MATERIAL DE APOYO PARA LA IMPARTICIÓN DE LA ASIGNATURA DE DISEÑO DE INTERFACES

TESIS QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

Presenta:
Darío Emmanuel Vázquez Ceballos

Directora De Tesis:
M. en D. Selene Marisol Martínez Ramírez

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS

Marzo 2013

MATERIAL DE APOYO PARA LA IMPARTICIÓN DE LA ASIGNATURA DE DISEÑO DE INTERFACES

Datos Del Jurado

Sinodal 1:
Dr. Gustavo de la Cruz Martínez

Sinodal 2:
Dra. Hanna Jadwiga Oktaba

Sinodal 3:
M. en D. Selene Marisol Martínez Ramírez (tutora)

Sinodal 4:
M. en C. Gustavo Arturo Márquez Flores

Sinodal 5:
M. en A. Karla Ramírez Pulido

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS

Marzo 2013

A mis padres y hermanas

con todo mi cariño

A mi directora de Tesis,
por su dirección y apoyo

A mis amigos y alumnos
de la Facultad de Ciencias

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, quiero agradecer a Word de Microsoft el haberme forzado a utilizar LaTeX.

Un agradecimiento especial a mi directora de Tesis y amiga Selene Martínez Ramírez por sus consejos, observaciones, paciencia y sobre todo por su siempre interés en este trabajo.

Un agradecimiento a mis familiares por su apoyo durante la licenciatura y durante la elaboración de este trabajo.

Índice

Agradecimientos VII

1. Introducción 1
- 1.1. Antecedentes . . . 1
  - 1.1.1. Diseño de Interfaces . . . 1
- 1.2. Objetivos del material . . . 2
- 1.3. Propuesta de solución. . . . 3
- 1.4. Estructura del material . . . 4

2. Material de Apoyo 5
- 2.1. GIMP y Photoshop . . . 6
  - 2.1.1. Práctica 01 Logotipos en Gimp . . . 7
  - 2.1.2. Práctica 02 Logotipos en Photoshop . . . 13
  - 2.1.3. Práctica 03 Botones en Photoshop . . . 18
- 2.2. UML . . . 29
  - 2.2.1. Práctica de UML . . . 29
- 2.3. Notas de CSS . . . 32
  - 2.3.1. Introducción a CSS . . . 32
  - 2.3.2. Selectores CSS . . . 41
  - 2.3.3. Modelo de caja . . . 58
  - 2.3.4. Posicionamientos . . . 87
  - 2.3.5. Complementos . . . 108
- 2.4. Notas de jQuery . . . 126
  - 2.4.1. Introducción a jQuery . . . 126
  - 2.4.2. Seleccionar Elementos . . . 129
  - 2.4.3. Selectores Avanzados . . . 144
  - 2.4.4. Funciones . . . 164
  - 2.4.5. AJAX . . . 187
  - 2.4.6. Complementos . . . 206

3. Asignaturas Relacionadas 217

3.1. Aplicaciones Web 217

Índice alfabético **220**

Acrónimos y glosario **220**

Índice de figuras

2.1. Logo de Megaupload [De Logo Megaupload, creada por el Autor de este trabajo (2011)] 7
2.2. Degradado circular y Texto. [Imágenes creadas por el autor de este trabajo (2011)] 8
2.3. Filtro luces y sombras. [Imagen creada por el autor de este trabajo (2011)] 8
2.4. Filtro inner shadow. [Imagen creada por el autor de este trabajo (2011)] 9
2.5. Efecto outer glow. [Imagen creada por el autor de este trabajo (2011)] 9
2.6. Efecto inner glow. [Imagen creada por el autor de este trabajo (2011)] 10
2.7. Efecto belvel and emboss. [Imagen creada por el autor de este trabajo (2011)] 10
2.8. Efecto satin. [Imagen creada por el autor de este trabajo (2011)] 11
2.9. Relleno y degradados. [Imágenes creadas por el autor de este trabajo (2011)] 11
2.10. Toques finales [Imagen creada por el autor de este trabajo (2011)] 12
2.11. Select-T, sombra y volumen. [Imágenes creadas por el autor de este trabajo (2011)] 14
2.12. Sombras y ajustes de la capa. [Imágenes creadas por el autor de este trabajo (2011)] 15
2.13. Rayos de luz. [Imágenes creadas por el autor de este trabajo (2011)] 16
2.14. Textura de arcilla. [Imágenes creadas por el autor de este trabajo (2011)] 16
2.15. Imagen con efectos de luz. [Imágenes creadas por el autor de este trabajo (2011)] 17
2.16. Degradados. [Imágenes creadas por el autor de este trabajo (2011)] 23
2.17. Filtro ruido [Imagen creada por el autor de este trabajo (2011)] 23

2.18. Crear botones. [Imágenes creadas por el autor de este trabajo (2011)] . . . 24
2.19. Rellenar los rectángulos. [Imagen creada por el autor de este trabajo (2011)] . . . 24
2.20. Sombra interior. [Imagen creada por el autor de este trabajo (2011)] . . . 25
2.21. Resplandor interior y opacidad. [Imagen creada por el autor de este trabajo (2011)] . . . 25
2.22. Bisel y relieve. [Imagen creada por el autor de este trabajo (2011)] . . . 26
2.23. Superposición de capas. [Imagen creada por el autor de este trabajo (2011)] . . . 26
2.24. Sombra en los botones. [Imagen creada por el autor de este trabajo (2011)] . . . 27
2.25. Botones con contorno. [Imagen creada por el autor de este trabajo (2011)] . . . 28
2.26. Bisel y relieve. [Imagen creada por el autor de este trabajo (2011)] . . . 28
2.27. Tipografía usanda en la página http://www.unam.mx [Imagen creada por el autor de este trabajo (2011)] . . . 34
2.28. Modificando la tipografía desde opciones del navegador [Imagen creada por el autor de este trabajo (2011)] . . . 35
2.29. Página de la UNAM con la nueva tipografía . . . 36
2.30. Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)] . . . 39
2.31. Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)] . . . 40
2.32. Partes de una regla de estilo [Imagen creada por el autor de este trabajo (2011)] . . . 41
2.33. Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)] . . . 54
2.34. Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)] . . . 55
2.35. Ejemplo de imagen con texto [Imagen creada por el autor de este trabajo (2011)] . . . 58
2.36. Border y padding [Imagen creada por el autor de este trabajo (2011)] . . . 59
2.37. Modelo de caja [Imagen creada por el autor de este trabajo (2011)] . . . 60
2.38. Margin y padding [Imagen creada por el autor de este trabajo (2011)] . . . 60
2.39. Márgenes [Imagen creada por el autor de este trabajo (2011)] 64

2.40. Márgenes cerrados [Imagen creada por el autor de este trabajo (2011)] . . . 66
2.41. Márgenes cerrados [Imagen creada por el autor de este trabajo (2011)] . . . 66
2.42. padding [Imagen creada por el autor de este trabajo (2011)] . 67
2.43. Estilos de borde [Imagen creada por el autor de este trabajo (2011)] . . . 70
2.44. Imagen de fondo [Imagen creada por el autor de este trabajo (2011)] . . . 75
2.45. Background [Imagen creada por el autor de este trabajo (2011)] 76
2.46. Divs anidados [Imagen creada por el autor de este trabajo (2011)] . . . 78
2.47. Medidas de divs aninados [Imagen creada por el autor de este trabajo (2011)] . . . 78
2.48. Resultado de medidas2.html [Imagen creada por el autor de este trabajo (2011)] . . . 79
2.49. Resultado en anchura.html [Imagen creada por el autor de este trabajo (2011)] . . . 80
2.50. Resultado en margin.html [Imagen creada por el autor de este trabajo (2011)] . . . 81
2.51. Resultado en border.html [Imagen creada por el autor de este trabajo (2011)] . . . 83
2.52. Resultado del Proyecto [Imagen creada por el autor de este trabajo (2011)] . . . 86
2.53. Elementos en bloque [Imagen creada por el autor de este trabajo (2011)] . . . 88
2.54. Elementos en bloque y en línea [Imagen creada por el autor de este trabajo (2011)] . . . 88
2.55. Contexto de formato en bloque [Imagen creada por el autor de este trabajo (2011)] . . . 90
2.56. Contexto de formato en línea [Imagen creada por el autor de este trabajo (2011)] . . . 91
2.57. Position relative [Imagen creada por el autor de este trabajo (2011)] . . . 93
2.58. Position absolute sin contenedores posicionados [Imagen creada por el autor de este trabajo (2011)] . . . 93
2.59. Position absolute con contenedores posicionados [Imagen creada por el autor de este trabajo (2011)] . . . 94
2.60. Position fixed [Imagen creada por el autor de este trabajo (2011)] 95
2.61. Párrafo flotado [Imagen creada por el autor de este trabajo (2011)] . . . 96

2.62. Ejemplo de párrafos flotados [Imagen creada por el autor de este trabajo (2011)] 97
2.63. Ejemplo del uso de clear [Imagen creada por el autor de este trabajo (2011)] 97
2.64. Visibility hidden [Imagen creada por el autor de este trabajo (2011)] 99
2.65. Display none [Imagen creada por el autor de este trabajo (2011)]100
2.66. Overflow: visible, hidden y auto [Imagen creada por el autor de este trabajo (2011)] 101
2.67. z-index [Imagen creada por el autor de este trabajo (2011)] . 102
2.68. Sección de comentarios [Imagen creada por el autor de este trabajo (2011)] 104
2.69. Elementos flotantes desbordados [Imagen creada por el autor de este trabajo (2011)] 106
2.70. Elementos flotantes abarcados [Imagen creada por el autor de este trabajo (2011)] 106
2.71. Sección para postear en CSS [Imagen creada por el autor de este trabajo (2011)] 107
2.72. Tipografía Serif [Imagen creada por el autor de este trabajo (2011)] 109
2.73. Tipografía Sans-Serif [Imagen creada por el autor de este trabajo (2011)] 109
2.74. Alineación vertical [Imagen creada por el autor de este trabajo (2011)] 115
2.75. La anchura del párrafo [Imagen creada por el autor de este trabajo (2011)] 121
2.76. Ejemplo de esquinas redondeadas [Imagen creada por el autor de este trabajo (2011)] 123
2.77. Ejemplo de bordes sombreados [Imagen creada por el autor de este trabajo (2011)] 125
2.78. Ejemplo de opacidad al 80 % [Imagen creada por el autor de este trabajo (2011)] 125
2.79. ejemplo de opacidad al 100 % 125
2.80. Página oficial de jquery [Imagen creada por el autor de este trabajo (2011)] 128
2.81. Resultado del ejemplo ready.html [Imagen creada por el autor de este trabajo (2011)] 146
2.82. Ocultar párrafos con fadeOut(). [Imágenes creadas por el autor de este trabajo (2011)] 152
2.83. resultados en contains.html [Imágenes creadas por el autor de este trabajo (2011)] 153

2.84. Resultado en atributos.html [Imagen creada por el autor de este trabajo (2011)] . . . 155
2.85. Resultado del ejercicio ready.html [Imagen creada por el autor de este trabajo (2011)] . . . 157
2.86. DOM.html. [Imágenes creadas por el autor de este trabajo (2011)] . . . 158
2.87. Algunos resultados de nth-child.html parte 1. [Imágenes creadas por el autor de este trabajo (2011)] . . . 159
2.88. Algunos resultados de nth-child.html parte 2. [Imágenes creadas por el autor de este trabajo (2011)] . . . 160
2.89. Resultados en contains.html. [Imagen creada por el autor de este trabajo (2011)] . . . 162
2.90. Resultados en atributos.html. [Imagen creada por el autor de este trabajo (2011)] . . . 163
2.91. Botones deshabilitados. [Imágenes creadas por el autor de este trabajo (2011)] . . . 163
2.92. Agregar un título a las imágenes en galeria.html [Imagen creada por el autor de este trabajo (2011)] . . . 165
2.93. Editar el título de las imágenes [Imagen creada por el autor de este trabajo (2011)] . . . 165
2.94. Agregar contenido HTML [Imagen creada por el autor de este trabajo (2011)] . . . 166
2.95. Concatenar texto con .text() [Imagen creada por el autor de este trabajo (2011)] . . . 167
2.96. Desplazar elementos con append() [Imagen creada por el autor de este trabajo (2011)] . . . 168
2.97. Desplazar elementos con before() y after() [Imagen creada por el autor de este trabajo (2011)] . . . 168
2.98. Agrupar elementos con wrap() [Imagen creada por el autor de este trabajo (2011)] . . . 169
2.99. event.stopPropagation [Imágenes creadas por el autor de este trabajo (2011)] . . . 174
2.100 Arreglo de párrafos [Imagen creada por el autor de este trabajo (2011)] . . . 176
2.101 galeria.html [Imagen creada por el autor de este trabajo (2011)] [Imágenes creadas por el autor de este trabajo (2011)] . . . 180
2.102 Resultado de aplicar las funciones en agregar_cont.html [Imagen creada por el autor de este trabajo (2011)] . . . 181
2.103 Agregar información a una galería [Imagen creada por el autor de este trabajo (2011)] . . . 181
2.104 Desplazar y agrupar elementos en moverElem.html [Imágenes creadas por el autor de este trabajo (2011)] . . . 182

2.105Sistema de pie de página. [Imágenes creadas por el autor de este trabajo (2011)] 184
2.106Eventos [Imágenes creadas por el autor de este trabajo (2011)] 186
2.107Convertir a un arreglo. [Imagen creada por el autor de este trabajo (2011)] 187
2.108Envío de parámetros en formato XML [Imagen creada por el autor de este trabajo (2011)] 197
2.109Ejemplo del uso de load() [Imagen creada por el autor de este trabajo (2011)] 200
2.110Cargar contenidos con AJAX [Imagen creada por el autor de este trabajo (2011)] 202
2.111Cargar noticias con AJAX [Imagen creada por el autor de este trabajo (2011)] 203
2.112Enviar parámetros al servidor con POST [Imágenes creadas por el autor de este trabajo (2011)] 205

Índice de Tablas

2.1. Anchura en CSS 61
2.2. Altura en CSS 61
2.3. Anchura mínima en CSS 62
2.4. Anchura máxima en CSS 62
2.5. Altura mínima en CSS 63
2.6. Altura máxima en CSS 63
2.7. Reglas individuales del margin 64
2.8. Regla resumida margin 64
2.9. Reglas individuales del padding 67
2.10. Regla resumida padding 68
2.11. Reglas individuales para el espesor del borde 69
2.12. Regla resumida para el espesor del borde 69
2.13. Reglas individuales para el estilo del borde 70
2.14. Regla resumida para el estilo del borde 71
2.15. Reglas individuales para el color del borde 71
2.16. Regla resumida para el color del borde 72
2.17. Reglas resumidas para cada borde 72
2.18. Regla resumida border 72
2.19. Regla para el color de fondo 73
2.20. Regla para colocar una imagen de fondo 73
2.21. Repetición de la imagen de fondo 74
2.22. Posición de la imagen de fondo 75
2.23. Regla resumida background 76
2.24. Posición de un elemento 89
2.25. Desplazamientos de los posicionados 90
2.26. Flotar un elemento 96
2.27. Regla clear 97
2.28. Propiedades de visibilidad 98
2.29. Propiedades de display 99
2.30. Propiedad overflow 100
2.31. Regla z-index 101

2.32. Fuentes disponibles 108
2.33. Fuentes compatibles Windows - Mac 108
2.34. Establece el color de letra 110
2.35. Establecer la tipografía 111
2.36. Establecer el tamaño de letra 111
2.37. Establecer el peso de letra 112
2.38. Establecer el estilo de letra 112
2.39. Letra versal 112
2.40. Regla resumida 112
2.41. Alineación del texto 113
2.42. Separación entre líneas de texto 113
2.43. Decoración del texto 113
2.44. Transformación del texto 113
2.45. Tabulación del texto 114
2.46. Separación entre letras 114
2.47. Separación entre palabras 114
2.48. Espacios vacíos 114
2.49. Alineación vertical 115
2.50. media css 116
2.51. Esquinas redondeadas 122
2.52. bordes con sombras 124
2.53. Opacidad de los elementos 125

Capítulo 1

Introducción

1.1. Antecedentes

Los cursos de Diseño de Interfaces, Aplicaciones Web (ver el anexo 3) y asignaturas a fines impartidas en la Facultad de Ciencias requieren de material teórico-práctico para cubrir los temas relacionados con el diseño de interfaces con enfoque Web.

1.1.1. Diseño de Interfaces

Durante los últimos 4 semestres que se ha impartido el curso de Diseño de Interfaces se deben cubrir los siguientes temas.

Temario

1.- Proceso de diseño de interfaces.

2.- Introducción IHC.

3.- Diseño Centrado en el Usuario.

4.- Principios y Guías de Diseño.

5.- Criterios para diseño de Interfaz.

6.- Como escribir para Web.

7.- Usabilidad.

8.- Evaluación.

Con los siguientes objetivos:

1.- Dar a conocer los elementos indispensables para constituir una interfaz gráfica de usuario usable y de calidad.

2.- Conocer los objetos de estudio de la disciplina conocida como IHC (Interacción Humano-Computadora).

3.- Valorar la importancia del diseño de interfaces de usuario en el contexto general del desarrollo de sistemas computacionales.

4.- Identificar los aspectos humanos y tecnológicos que impactan el desarrollo de interfaces.

5.- Conocer los paradigmas existentes para el diseño de interfaces humano-computadora.

6.- Conocer y aplicar técnicas de diseño y evaluación de interfaces de usuario.

7.- Crear una sistema Web/escritorio que sea usable.

Plan de trabajo. Durante el curso los alumnos formarán equipos pequeños de 2-3 personas y tendrán que investigar y diseñar interfaces sobre un tema en particular. Para cada tópico del curso los alumnos harán la investigación bibliográfica a través de Internet la cual será revisada y discutida en clase.

Durante las prácticas de ayudantía desarrollaran elementos que pueden ayudarles en el desarrollo de su proyecto final.

Los alumnos tendrán que generar documentación que justifique el diseño de las interfaces de su proyecto final. La documentación se hará disponible en Internet para la consulta pública. La documentación se evaluará en función de pertinencia y profundidad de la investigación.

Se evaluará el cumplimiento del proceso de Diseño de Interfaces y la calidad del sistema (proyecto final).

1.2. Objetivos del material

Los manuales teórico-prácticos del trabajo "Material de Apoyo para la Impartición de la Asignatura de Diseño de Interfaces" tienen como objetivo apoyar al docente de información y ejercicios que le permitan cubrir los temas relacionados a diseño de recursos gráficos, diseño de interfaces web y su implementación, y el uso de la tecnología JavaScript y AJAX.

Objetivos del material para con los estudiantes:

- Apoyar a los estudiantes en el aprendizaje del diseño de interfaces con un enfoque Web.

- Permita comprender los temas relacionados con el diseño de imágenes en GIMP y Photoshop. Con ello los estudiantes pueden implementar los recursos gráficos de su aplicación.

- Permita abordar de forma clara los temas relacionados con HTML y maquetación Web. Con el objetivo de crear una estructura web sólida y limpia usando buenas técnicas de programación Web.

- Abordar los temas relacionados con la apariencia de una página Web utilizando CSS 2.1 y un poco de CSS 3. Con la intención de crear interfaces limpias y usables sin utilizar tablas HTML para dar apariencia.

- Abordar los temas relacionados con JavaScript para dotar de dinámica visual al sitio Web y utilzar los gran variedad de plugins gratuitos sobre JavaScript y CSS.

- Cubrir de forma básica el Framework jQuery de JavaScript, para facilitar al diseñador Web el uso de la tecnología JavaScript y sobre todo el uso de AJAX.

- Que las prácticas propuestas les permitan ejercitar los conocimientos y técnicas adquiridas en el curso.

1.3. Propuesta de solución.

Se propone un conjunto de manuales prácticas (cap. 2) con ejemplos y material complementario en apoyo al instructor en la impartición de los temas; HTML, CSS, jQuery y GIMP/Photoshop. Estos manuales aportan a los alumnos información y ejemplos sobre dichas tecnologías que ayudan en sus tareas referentes a la web y en su proyecto final.

Adicionalmente a este trabajo se desarrolló un sitio web "emulando" un proyecto final de Diseño de Interfaces y/o Aplicaciones Web, con el objetivo de mostrar la utilidad de estos manuales en la práctica. Este proyecto utiliza tecnologías que se abarcan en el curso de aplicaciones web y conceptos de usabilidad de diseño de interfaces. Durante su desarrollo se siguieron las fases de diseño de interfaces y con ello una documentación de proyecto enfocada en el diseño de la interfaz por lo que esta documentación cubre los lineamientos solicitados en el curso y puede servir al estudiante como guía en la elaboración de su documentación final.

1.4. Estructura del material

Este libro, proporciona un proceso bien definido basado en buenas prácticas de programación Web. Ofrece conceptos y ejemplos prácticos sobre las tecnologías impartidas en el curso. Y conforma una guía práctica de apoyo en la impartición de los cursos afines al material.

Está compuesto por 3 manuales teórico-práctico, cada manual presenta secciones de teoría con sus ejemplos, en algunos casos se presenta un tutorial "paso a paso" a modo de ejemplo en el uso de una tecnología en particular, una sección de ejercicios a modo de práctica para evaluar los conocimientos adquiridos y algún material extra sobre la tecnología en cuestión.

1.- Manual de GIMP y Photoshop (cap. 2.1).

2.- Manual de CSS (cap. 2.3).

3.- Manual de JavaScript/jQuery (cap. 2.4).

Capítulo 2

Material de Apoyo

Resumen: En este capítulo se presentan un conjunto de manuales y prácticas con material de apoyo para la asignatura de Diseño de Interfaces y asignaturas afines impartidas en la Facultad de Ciencias de la UNAM. El material se enfoca en tecnologías relacionadas con el diseño e implementación de interfaces gráficas de usuario y aplicaciones Web.

En el contenido se encuentran tecnologías dedicadas a recursos gráficos como GIMP y Photoshop. La tecnología CSS dedicada a la implementación de la apariencia de las páginas Web. Tecnologías del lado del cliente como JavaScript y su biblioteca jQuery que es de gran utilidad y presenta un buen soporte entre los desarrolladores Web, en este campo también se incluye AJAX para la comunicación asíncrona con el servidor.

Durante el curso de Diseño de Interfaces impartido en la Facultad de Ciencias de la UNAM se cubren temas relacionados con tecnologías que permiten diseñar e implementar aplicaciones web y/o de escritorio. En el curso se abordan conceptos, buenas técnicas de implementación y ejemplos sobre dichas técnologías y al final de cada bloque se plantean ejercicios que miden los conocimientos adquiridos.

Durante el curso se suelen cubrir las siguientes tecnologías:

gimp/photoshop. Que permiten crear los layouts, los view templates, la botonería, recursos gráficos de contenido y de apariencia, entre otras.

CSS. Que se encargan de la apariencia del contenido web.

jQuery. Que simplifica el manejo de JavaScript de la página web.

JavaScript. Permite crear páginas Web dinámicas.

El curso se enfoca en implementar una interfaz gráfica de usuario que cumpla los criterios ergonómico de usabilidad sin poner énfasis en el lenguaje utilizado del lado del servidor. Se deja al instructor decidir que lenguaje emplear, en cursos anteriores se han utilizado PHP, Java y Ruby por lo que su uso esta recomendado.

Al final del curso el alumno debe implementar un sistema web, de escritorio o móvil, donde aplique los conocimentos teóricos expuestos en clase y las tecnologías vistas en estos manuales.

Los manuales prácticos son:

1.- GIMP/Photoshop.

2.- CSS.

3.- jQuery.

Dichos manuales presentan la siguiente estructura.

- Bloques teóricos según los temas.
- Material de apoyo y ejemplos.
- Algunos presentan tutoriales paso a paso.
- Bloque de ejercicios.

2.1. GIMP y Photoshop

GIMP *(GNU Image Manipulation Program).* Es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un software libre y gratuito y forma parte del proyecto GNU que está disponible bajo la Licencia pública general de GNU, como se indica en la wiki oficial de gimp.org (http://wiki.gimp.org/index.php/Main_Page).

Es el programa de manipulación de gráficos disponible en más sistemas operativos (Unix, GNU/Linux, FreeBSD, Solaris, Microsoft Windows y Mac OS X, entre otros).

Photoshop (*Adobe Photoshop*). Es una aplicación informática destinada a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits.

Es un software que permite sacar el máximo partido a las imágenes digitales y transformarlas, como lo indica la página oficial de Adobe (http://www.adobe.com/mx/products/photoshopfamily.html).

2.1.1. Práctica 01 Logotipos en Gimp

2.1.1.1. Enunciado

El logotipo de una página Web es un elemento muy importante en el diseño e implementación de un trabajo usable. La razón de su importancia radica en su finalidad; mostrar el título general de la Web y casi siempre se convierten en elementos característicos de la empresa y su filosofía. En la mayoría de las ocasiones estos elementos aportan conocimientos suficientes al usuario sobre el tipo de sistema en el que navega, es decir que tipo y finalidad tiene la página Web.

Por ejemplo. La página de *Megaupload* (figura 2.1), que es un sitio Web que proporciona el servicio de almacenamiento en línea, tiene en su logotipo y nombre un degradado de color con efectos de luces y sombras, lo que producen efectos de movimiento (transición) y los colores de rellenos negro-anaranjado siendo este último un color que transmite dinamismo, confirman la intención y finalidad de la página; almacenar y descargar archivos.

Figura 2.1: Logo de Megaupload [De Logo Megaupload, creada por el Autor de este trabajo (2011)]

2.1.1.2. Desarrollo

Esta práctica tiene como finalidad diseñar e implementar el logotipo de la página Web o aplicación solicitada durante el semestre (proyecto final), usando Gimp (versión 2.6.10) y/o Photoshop (CS3 o posteriores).

Los siguientes pasos pertenecen a un tutorial que ejemplifica la elaboración de un logotipo de la página Web "Blicken" desarrollada en el semestre 2009-2 en el curso de Diseño de Interfaces en la Facultad de Ciencias de la UNAM, utilizando la aplicación GIMP 2.6.

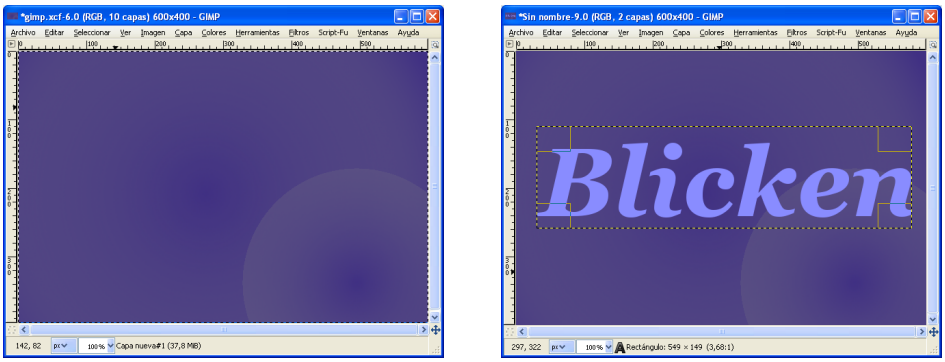
La aplicación GIMP se puede obtener desde su página oficial GIMP. La página también proporcina información importante sobre las versiones, herramientas, plug-in, noticias y documentación sobre la aplicación.

Tutorial de GIMP

1.- Crear una nueva imagen de 600px por 400px con fondo transparente. Después aplicar un degradado circular, para ello ir a Caja de Herramientas → Herramienta de mezcla → en la opción de forma usar Radial. El resultado se muestra en la figura 2.2a

2.- Ingresar el texto deseado. En el ejemplo se utiliza la fuente "Georgia Bold Italic" y como color el #898cff, sobre el texto "Blicken". Ir a Capa

→ y aplicar Capa a tamaño de imagen. El resultado se muestra en la figura 2.2b.



(a) Degradado circular (b) El texto

Figura 2.2: Degradado circular y Texto. [Imágenes creadas por el autor de este trabajo (2011)]

3.- Aplicar sobre el texto un filtro, para ello ir a Filtros → Luces y sombras → Sombra arrojada, con un desplazamiento de 0px en "x" e "y" y un radio de 10px. El resultado se muestra en la figura 2.3.

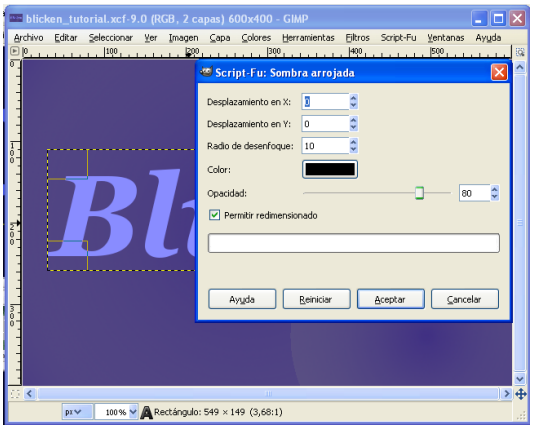


Figura 2.3: Filtro luces y sombras. [Imagen creada por el autor de este trabajo (2011)]

4.- Aplicar Script-Fu → Effects Layer → Inner Shadow, con un radio de 10 y una opacidad del 80 %. El resultado se muestra en la figura 2.4.

5.- Aplicar Script-Fu → Effects Layer → Outer Glow, con color blanco y una opacidad del 80 %. El resultado se muestra en la figura 2.5.

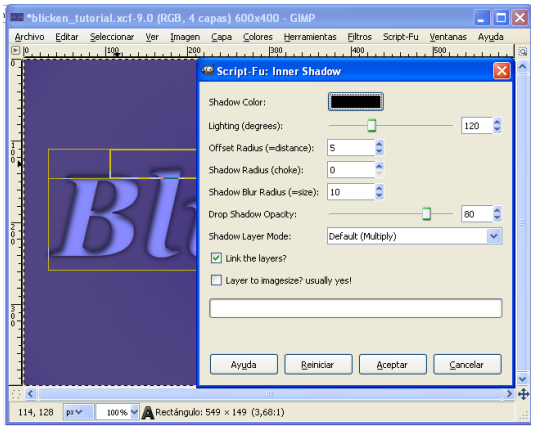


Figura 2.4: Filtro inner shadow. [Imagen creada por el autor de este trabajo (2011)]

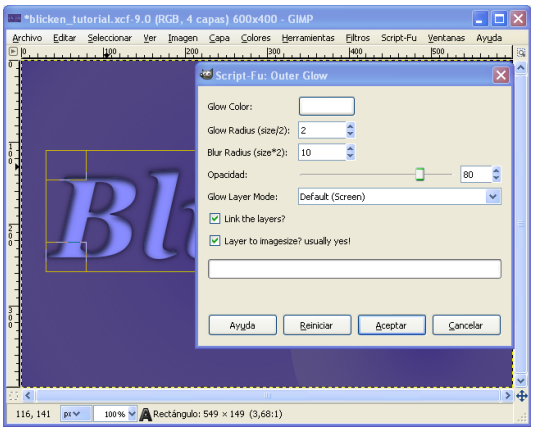


Figura 2.5: Efecto outer glow. [Imagen creada por el autor de este trabajo (2011)]

6.- Aplicar Script-Fu → Effects Layer → Inner Glow, con color blanco y una opacidad del 80 %. El resultado se muestra en la figura 2.6.

7.- Aplicar Script-Fu → Effects Layer → Belvel and Emboss, con los valores por defecto. El resultado se muestra en la figura 2.7.

8.- Aplicar Script-Fu → Effects Layer → Satin, con los valores por defecto. El resultado se muestra en la figura 2.8

9.- Seleccionar la herramienta de selección de color (Mayús + O), y posicionarse en la capa del texto, seleccionar el mismo. Borrar la selección (con Supr), y rellenar utilizando la herramienta de relleno (Mayús + B) con el color de frente, colocando la opacidad del relleno al 50 %.

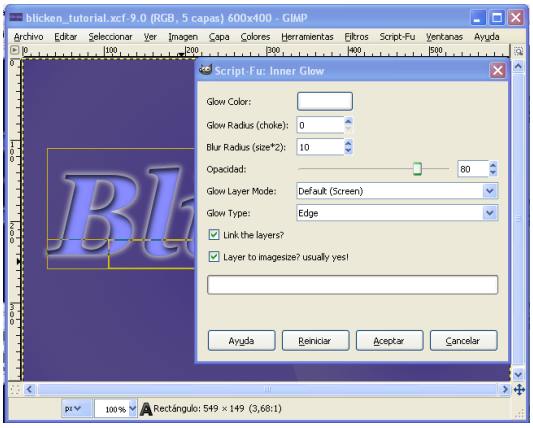


Figura 2.6: Efecto inner glow. [Imagen creada por el autor de este trabajo (2011)]

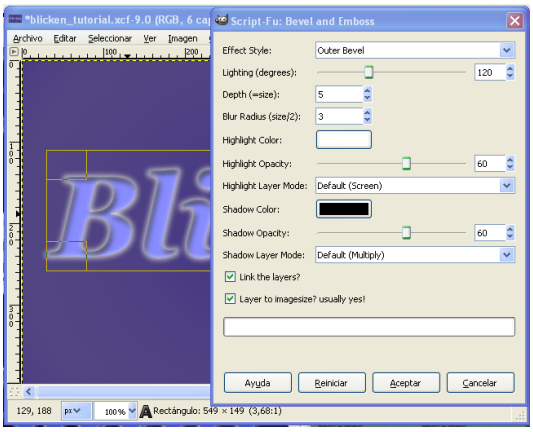


Figura 2.7: Efecto belvel and emboss. [Imagen creada por el autor de este trabajo (2011)]

10.- Nuevamente seleccionar el texto como se indica en el paso anterior. Colocar el color de frente en "blanco"(fff), tomar la herramienta de degradado (L) y aplicar un degradado de color de "frente a transparente" desde abajo hacia arriba en una nueva capa. Luego cambiar el modo de esta capa a "Solapar". El resultado se muestra en la figura 2.9a

11.- Repetir el paso anterior, pero esta vez realizar el degradado de arriba hacia abajo y colocar la opacidad de la capa al 40 % en lugar de cambiar su modo.

12.- Tomar la herramienta de selección elíptica (E) y seleccionar la zona superior del texto. Invertir la selección (I) y borrar usando la tecla

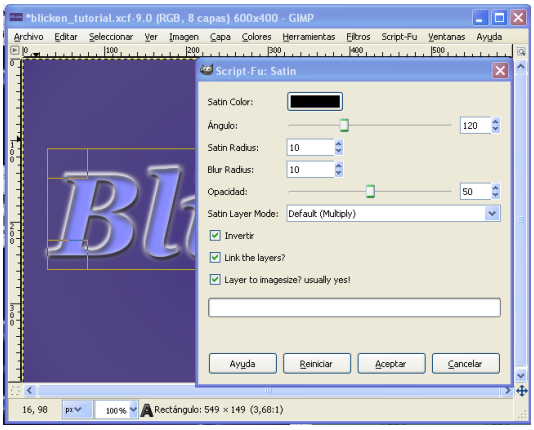
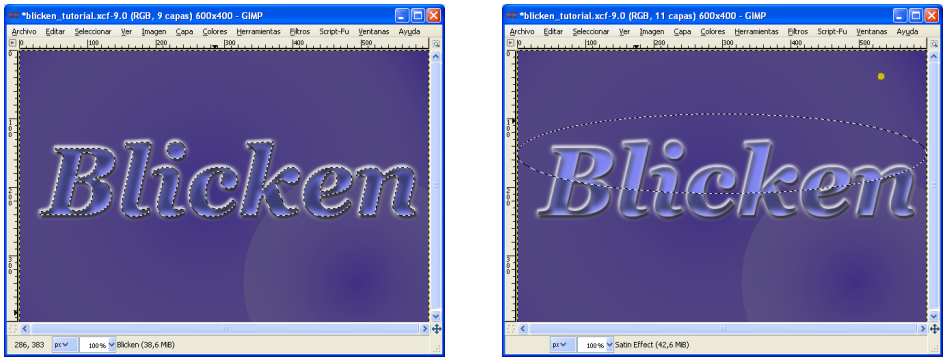


Figura 2.8: Efecto satin. [Imagen creada por el autor de este trabajo (2011)]

Supr. Resultado en la figura 2.9b



(a) Relleno y opacidad. (b) Degradado y opacidad.

Figura 2.9: Relleno y degradados. [Imágenes creadas por el autor de este trabajo (2011)]

13.- Tomar el pincel (P), seleccionar una brocha en forma de estrella de cuatro puntas (o crear una utilizando la opción “Pincel nuevo” del menú de pinceles) y realizar algunas estrellas en color blanco para dar un efecto de brillo y decorativo a la imagen.

2.1.1.3. Ejercicios de Entrega

El alumno debe realizar un trabajo similiar utilizando las técnicas presentadas en el tutorial.

- Capas.
- Filtros.

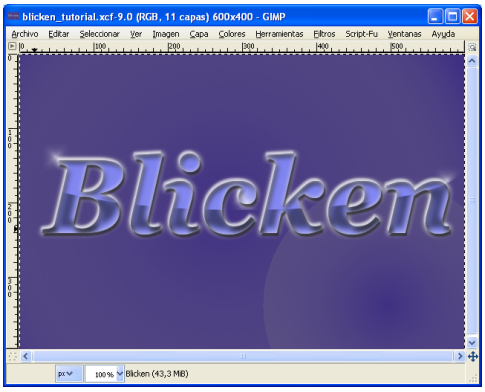


Figura 2.10: Toques finales [Imagen creada por el autor de este trabajo (2011)]

- Degradados.
- Máscaras.
- Herramientas de dibujo.
- otros.

El alumno debe entregar la imagen del logotipo final en formato png (jpg o gif) y el documento xcf (archivo nativo de Gimp) o el psd (archivo nativo de Photoshop). Este último archivo es importante, por que muestra el trabajo en capas de la imagen. También debe adjuntar un pdf donde explique su logotipo, así como las herramientas y técnicas de Gimp que utilizadas.

Estructura Clásica. El alumno debe generar una carpeta con la siguiente estructura de directorios:

◯ Carpeta "login de ciencias del estudiante".

→ Carpeta "imagenes".
- Trabajo xcf o psd realizado.
- Imagen en png, jpg o gif, exportada de la fuente original.

→ Carpeta "docs".
- Documento de reporte en formato pdf.

Script-Fu Para cargar los efectos en script-Fu:

1.- Descargar el script layerfx.scm de la página GIMP Plugin Registry

2.- Guardar el script en la ruta /home/usuario/.gimp-2.6/scripts varia según la versión de gimp, para confirmar la ruta abrir gimp ir a editar, preferencias, carpetas y Guiones ahi encontraran las rutas de los scripts.

3.- Para cargar los script ir a filtros, Script-Fu y le damos en Refrescar Guiones.

Nota Las versiones de Script-Fu pueden varias sus funcionalidades, es recomendable revisar su documentación.

2.1.2. Práctica 02 Logotipos en Photoshop

2.1.2.1. Enunciado

Esta práctica es un refuerzo de las técnicas y herramientas de la práctica anterior, con algunas técnicas nuevas. La intención es dar paso a técnicas útiles para el diseño e implementación de la página Web y recursos gráficos del proyecto final.

Con estás técnicas se dan las bases para diseñar los recursos gráficos de contenido de la página, es decir, imágenes que acompañan el contenido textual y que ellas mismas forman parte del mensaje a comunicar. Además se utilizan para diseñar los recursos gráficos encargados de la apariencia de la página usandos dentro de los archivos CSS.

2.1.2.2. Desarrollo

Los siguientes pasos muestran el desarrollo de un logotipo de la página Web "Blicken" usando efectos y técnicas de Photoshop.

Para desarrollar la práctica se requiere la aplicación Photoshop CS3 o superior. Para obtener una versión de prueba se tiene que dirigir a la página oficial de Adobe Photoshop CS. En la página oficial se detallan los requerimientos para obtener una versión de prueba de Photoshop.

El tutorial está implementado con Photoshop CS3 versión de prueba, pero las herramientas y pasos son similares en versiones superiores.

Tutorial 1 de Photoshop

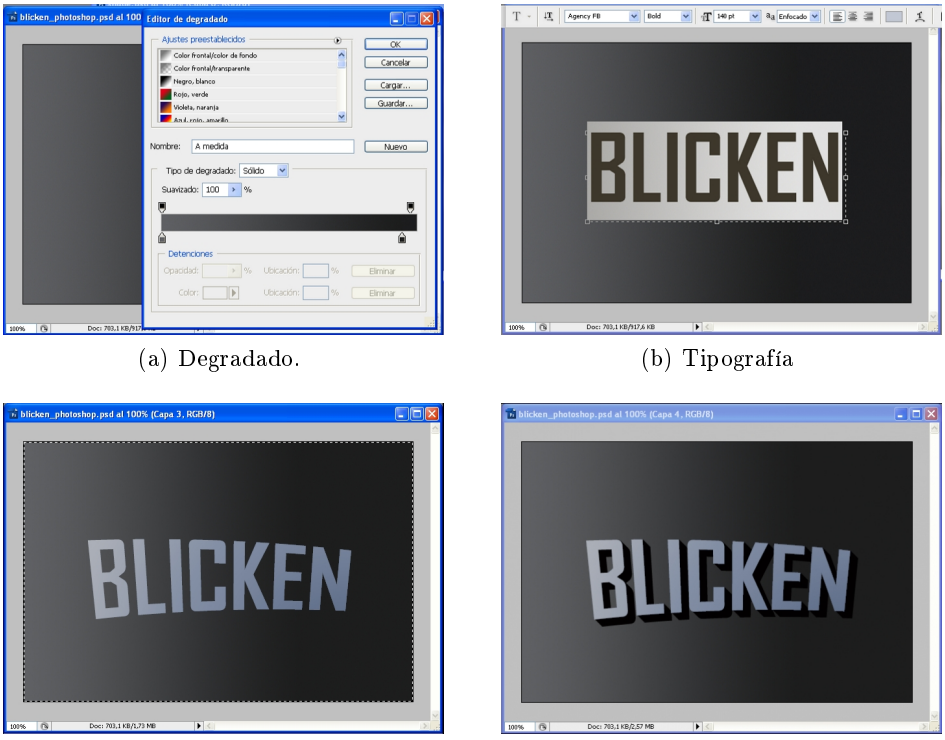
1.- Dibujar un degradado de gris-negro en un panel de 600px por 400px. El resultado se muestra en la figura 2.11a

2.- Ingresar el texto, en el tutorial se utiliza el tipo de letra Agency FB(se puede usar cualquier fuente), con un tamaño 140 pt, bold y un color gris #c2c8d4 (aplicar un efecto de arco al texto). Para escoger el tipo de letra es recomendable leer sobre las fuentes más usadas en el desarrollo WEB. El resultado se muestra en la figura 2.11b

3.- Presionando la tecla ctrl dar clic izquierdo del mouse en la capa de Texto (sobre la capa que tiene una T) obteniendo una selección (select-T) con forma del contorno del texto. Después crear una nueva capa y

posicionarse en ella, la selección select-T debe verse en dicha capa, de lo contrario repetir la acción. Aplicar un degradado de gris (#495a79) a transparente y en la nueva capa realizar el degradado de la parte inferior derecha a la parte superior izquierda, calculando para que se muestre un poco de luz en las primeras letras. El resultado se muestra en la figura 2.11c

4.- Restaurar los valores por defecto (con la tecla D). Después sin soltar ctrl dar clic derecho del mouse sobre la capa de texto obteniendo nuevamente la selección select-T, dado esto crear una nueva capa y ponerla debajo (orden de capas) de la capa de texto. Para dar el efecto de sombra y volumen se deben seguir los siguientes pasos, posicionarse en la capa nueva teniendo certeza de poder ver la capa y hacer un select-T. Con las teclas bajar una posición y desplazar a la derecha una posición, después pulsar ctrl + backspace (retroceso) se colorea una sombra de 1px de grosor, repetir estos pasos 5 veces. El resultado se muestra en la figura 2.11d



(a) Degradado. (b) Tipografía

(c) select-T (d) Sombra y volumen

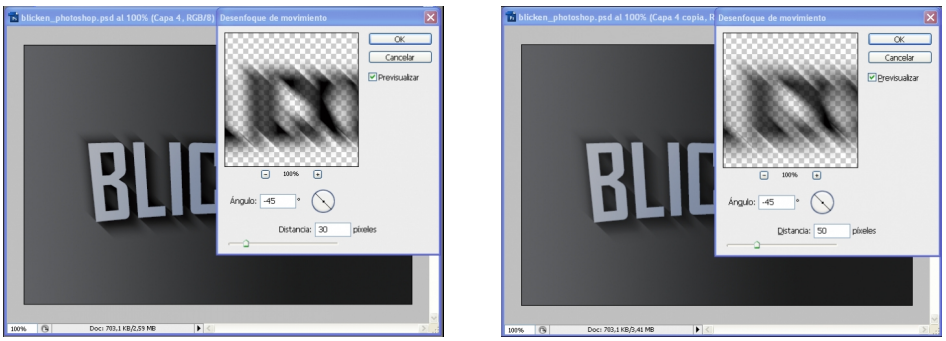
Figura 2.11: Select-T, sombra y volumen. [Imágenes creadas por el autor de este trabajo (2011)]

5.- En la capa de sombras aplicar un filtro, para ello ir a filtro→ Desenfocar

→Desenfoque de movimiento (Filter → Blur → Motion Blur). Poner como valores un ángulo de -45 y una distancia de 30px. El resultado se muestra en la figura 2.12a

6.- En la capa de sombra cambiar el tipo a "Multiplicar" y ajustar la opacidad al 40 %.

7.- Duplicar la capa de sombra, aplicar el filtro desenfoque de movimiento con una distancia de 50px. La capa debe ser del tipo "Multiplicar" y ajustar la opacidad al 20 % arrojando más sombra al fondo. El resultado se muestra en la figura 2.12b



(a) Filtro Motion Blur (b) Capa de tipo multiplicar

Figura 2.12: Sombras y ajustes de la capa. [Imágenes creadas por el autor de este trabajo (2011)]

8.- Crear una nueva capa por encima de todas las anteriores (orden de capas) y manteniendo presionado ctrl dar clic derecho en la capa de texto, para obtener select-T, llenar la selección de blanco de la nueva capa, después bajar 1px la selección y moverse 1px a la derecha, después pulsar supr para borrar el contenido, con esto se deja un pequeño semi-contorno blanco en select-T, poner opacidad de la capa a un 80 %. El efecto de luz produce una especie de texto tridimensional.

9.- Crear algunos efectos de rayos de luz, en una nueva capa por encima de todas las capas y dibujar 4 o 5 rectángulos de diferentes tamaños que vayan de extremo a extremo, (es recomendable hacerlo de arriba-abajo de menor grosor a mayor grosor). El resultado se muestra en la figura 2.13a

10.- Combinar las capas de los rayos en una sola capa, presionar ctrl+T para transformar, ampliar y rotar la capa, para dar por terminado los cambios presionar enter. Una forma rápida de modificar es dar clic derecho sobre la imagen de trabajo y hacer los cambios, en el ejemplo

se usaron las funciones de escalar (agrandar la capa), de rotación (dar el efecto de caída diagonal) y la perspectiva. El resultado se muestra en la figura 2.13b

11.- A la capa de rayos de luz aplicar una opacidad del 20 % y aplicar un filtro, para ello ir a filtro→desenfoque→desenfoque guassiano con un radio de 6px.

(a) Rectángulos horizontales (b) Efecto rayo de luz

Figura 2.13: Rayos de luz. [Imágenes creadas por el autor de este trabajo (2011)]

12.- Crear una nueva capa con encima de la capa de fondo con un relleno rosa (#9d506c). El resultado se muestra en la figura 2.14a

13.- Aplicar una opacidad de 20 % dando una textura de fondo como arcilla. El resultado se muestra en la figura 2.14b

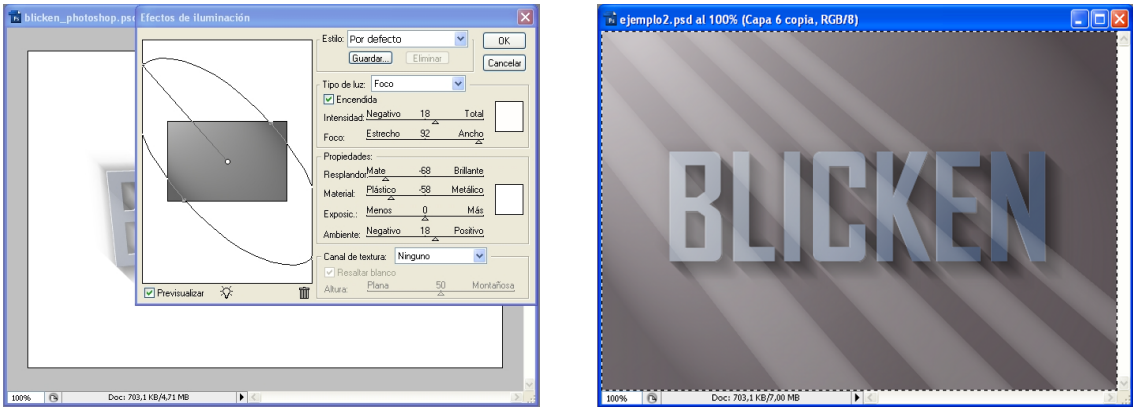
(a) Capa relleno rosa (b) Textura

Figura 2.14: Textura de arcilla. [Imágenes creadas por el autor de este trabajo (2011)]

14.- Crear una nueva capa por encima de la capa rosa y llenarla de color blanco, después aplicar un filtro, para ello ir a filtro→Interpretar→efectos

de iluminación. Establecer los siguientes valores: estilo → "por defecto", tipo de luz → "foco", en propiedades resplandor → -68, material → -58, Exposic → 0, ambiente → 18, intensidad → 18 y foco → 92. Como se muestran en la figura 2.15a.

15.- Duplicar la capa y pasarla por encima de todas las capas del proyecto con una opacidad del 40 %, para afectar al texto por encima, después crear una máscara para afectar únicamente una sección, enseguida aplicar un degradado de blanco a negro de la parte superior izquierda a la parte inferior derecha de la última capa.



(a) Foco de iluminación (b) Luz en capas

Figura 2.15: Imagen con efectos de luz. [Imágenes creadas por el autor de este trabajo (2011)]

2.1.2.3. Ejercicios de Entrega

El alumno debe realizar un trabajo similiar utilizando las técnicas presentadas en el tutorial.

- Capas.
- Filtros.
- Degradados.
- Máscaras.
- Herramientas de dibujo.
- otros.

El alumno debe entregar la imagen final en formato png (jpg o gif) y el documento xcf (archivo de Gimp) o el psd (archivo de Photoshop). Este último archivo es muy importante, ya que muestra el trabajo en capas de la imagen. También debe adjuntar un pdf donde explique el logotipo, las herramientas y técnicas de Gimp o Photoshop que se utilizaron.

Estructura Clásica. El alumno debe generar una carpeta con la siguiente estructura de directorios:

◯ Carpeta "login de ciencias del estudiante".

→ Carpeta "imagenes".

- Trabajo xcf o psd realizado.
- Imagen en png, jpg o gif, exportada de la fuente original.

→ Carpeta "docs".

- Documento de reporte en formato pdf.

2.1.3. Práctica 03 Botones en Photoshop

2.1.3.1. Enunciado

Los botones en una interfaz son elementos vitales, ya que en la mayoría de los casos aglutinan muchas de las capacidades de navegación que concibe el creador de un sitio web. Por ese motivo, adquieren un protagonismo importante, aun cuando, por regla general, son los elementos de menor tamaño dentro de una interfaz, además son los que comúnmente se utilizan dentro de una aplicación de software o página Web.

Por todo ello, una correcta utilización de estos elementos es vital de cara a obtener el uso óptimo de esa interfaz. A fin de lograrlo, presentamos una serie de recomendaciones desde el punto de vista de la usabilidad que ayudarán, a un mejor uso.

- Recomendaciones sobre el texto contenido por los botones
- Recomendaciones sobre las características de los botones

Principalmente, existen dos tipos de botones en los entornos web:

- Botones textuales.- Son aquellos que básicamente basan toda la información que transmiten sobre textos o a través de ellos.
- Botones icónicos.- Son aquellos que transmiten su información principalmente a través de una imagen siendo capaces de que la interpretación que el usuario dé a dicha imagen sustituya con éxito el mensaje textual que se les podría haber proporcionado. Así mismo, estos pueden disponer de información textual, la cual vendría dada por el etiquetado del icono.

Centrándonos en la información textual transmitida por los botones textuales, existen varias recomendaciones a tener en cuenta:

1.- Es necesario etiquetar todos los botones con verbos imperativos. Estos etiquetados han de tener siempre la primera letra en mayúscula, como por ejemplo 'Buscar', Éliminar', 'Guardar', etc. De este modo, habría que huir de botones con textos tales como Ácepta', Áccede', etc.

2.- Se recomienda usar el símbolo (...) o '...ál final de las etiquetas si la acción requiere más explicación para el usuario antes de la acción sea llevada a cabo. Por ejemplo "Guardar como ..." o "Buscar ...". Estos signos, no deberían añadirse en comandos tales como "Propiedades", "Preferencias", "Configuración", ya que estos comandos no requieren mayor explicación que la que ya expresan por medio de su etiquetado.

3.- Se recomienda utilizar siempre términos breves y representativos. Las frases largas de carácter descriptivo son desaconsejables.

4.- En el caso de que el etiquetado del botón tenga dos o más palabras, es crítico mantener la consistencia con el español. Un error muy común a la hora de etiquetar botones es importar errores tipográficos internacionales, la mayoría de procedencia anglosajona. De todos ellos, el más común es el de entremezclar mayúsculas y minúsculas dentro de una misma frase sin criterio alguno. Hay que hacer notar que si bien en la lengua inglesa no constituye falta de ortografía el comenzar con letra mayúscula cada palabra que no sea determinante, adverbio o preposición dentro de una misma frase, esto constituye una falta de ortografía en castellano. No obstante, poner mayúsculas a palabras situadas en mitad de frases castellanas se está convirtiendo en costumbre común. Un empeoramiento de esta situación lo constituye el situar dichas letras mayúsculas en algunas palabras de dicha frase y en otras no.

El segundo aspecto a tener en cuenta una vez se posee una idea más o menos clara sobre las cuestiones lingüísticas es la funcionalidad de dichos elementos. La funcionalidad de un botón en una interfaz web es casi tan importante como su etiquetado lingüístico. **El comportamiento de dicho botón proporcionará información crítica al usuario y le guiará por su interacción normal con el sitio web.**

En aras de una potenciación de la Accesibilidad de un sitio web, el acceso a las distintas secciones del mismo por medio de botones debería poder hacerse de dos maneras:

- Navegando por medio del ratón.- Es el uso más común por medio del cual interacciona el usuario con los sitios web.

- Navegando por medio del teclado.- Posibilidad necesaria para aquellas personas que tienen necesidad de hacer uso de los sitios web accesibles debido a algún tipo de discapacidad.

Acorde a estos usos, las recomendaciones que se pueden hacer son:

1.- En el caso de que se habilite una navegación por medio del teclado para el sitio Web, cada opción designada debería quedar asociada con los botones del teclado escogidos. De esta manera si para acceder al apartado Ínicio'se han de pulsar los botones $CTRL + I$, esta letra debería quedar resaltada de algún modo en el texto del botón. Ejemplo Inicio, Ventana, etc.

2.- Una vez que una función de navegación ha sido asociada a un botón, no se debe cambiar el botón por defecto manteniendo dicha función inamovible para el resto de la estructura. Se pueden añadir o eliminar el estado de esos botones si ello ayuda a prevenir que el usuario cometa errores. Pero el que los botones aparezcan o desaparezcan de la pantalla en función de las acciones que el usuario va realizando no logrará sino despistarlo.

3.- Es crítico etiquetar todos los botones a fin de que el usuario pueda obtener información acerca de la funcionalidad sin necesidad de pulsarlos. Adicionalmente, ello posibilitará la navegación en el caso de que los botones no se carguen, demoren en hacerlo o directamente, no existan por algún tipo de incompatibilidades con el navegador que está usando.

4.- Si el botón es capaz de generar texto, un icono o ambos, es necesario describir gráficamente lo que va a suceder. La generación de texto en la pantalla ocultará algunos elementos de la misma y ello podría propinar que el usuario se pierda. Es mejor advertir antes que subsanar.

Con respecto a la apariencia que se le debe dar a los botones dentro de una interfaz, la recomendación básica es la de mantener consistencia. La consistencia dotará al sitio Web de la facultad de los usuarios en poner en marcha sus aprendizajes previos así como le permite dotarse de pautas de actuación más o menos cómodas a la hora de interactuar con un sitio web. Por tanto, hay que tener en cuenta los siguientes puntos:

1.- No usar más de uno o dos anchos para los botones que componen una interfaz y, en cualquier caso, usar siempre el mismo alto. Ello ayudará a proporcionar un cómodo sistema visual para que la percepción del usuario sea capaz de clasificar los elementos que componen una interfaz.

2.- Aquellos botones cuya función, a causa de las acciones del usuario, quede inhabilitada, deberán quedar presentes, pero inhabilitados. Un ensombrecimiento de los mismos, ayudará sin duda a clarificar que a causa de esa acción, el botón ya no está disponible o aún no ha sido activado.

3.- En un texto, un botón puede constituir el botón por defecto de una interfaz. Para ello, tiene que estar resaltado con un borde distinto y debe volverse de la misma manera cuando el usuario regrese a la interfaz. Generalmente, este tipo de botones suelen ser los etiquetados con las funciones "Aceptar", "Adelante", "Siguiente", etc... Tan sólo en casos excepcionales "Cancelar" debería ser el botón activado por defecto de una función.

4.- Si un botón por defecto sólo va a poder ser utilizado una vez el usuario haya completado un cierto número de pasos (por ejemplo, rellenar un nombre de usuario y una contraseña), este botón NUNCA debería activarse hasta que el usuario ha realizado correctamente la tarea en cuestión. No tiene sentido posibilitar que el usuario lo pulse si ello le va a conducir a un error seguro.

Con respecto a la apariencia que los botones deben tener con respecto a la interfaz global del sitio web, se deben de tomar en cuenta los siguientes puntos:

1.- Cada vez que un botón se pulse, hay que dar feedback al usuario de que está siendo pulsado. Este feedback sí que admite las animaciones o los sonidos, siempre y cuando ayuden a aclarar al usuario que ese sonido está describiendo auditivamente el pulsado de un botón. En el caso de optar por dotar al botón de una animación a fin de resaltarlo, esta ha de estar hecha acorde a los principios básicos de la usabilidad, siendo lo más destacados:

- La animación solo debería activarse cuando se comience a ejecutar la acción que la desencadena y debería finalizar cuando dicha acción ha dejado de realizarse. Las animaciones que aparecen en una interfaz sin que el usuario tenga relación alguna con ellas, no hacen sino despistarle.
- Las animaciones han de ser sencillas y comunes al entendimiento del usuario. Un despliegue espectacular ante la realización de una acción sencilla, rompería la coherencia de actos en la interacción con el sitio web.
- De cara a mantener las acciones de accesibilidad, sería idóneo proporcionar algún tipo de feedback adicional a la acción de pulsado del botón, para que aquellos usuarios que no sean capaces

de percibir esa animación, puedan tener algún tipo de feedback que les proporciona similar sensación.

2.- La ubicación de los botones en la interfaz de un sitio web es un aspecto fundamental. Es recomendable que el botón con mayor posibilidad de ser pulsado sea mostrado en primer lugar. En este sentido, hay que tener en cuenta que las pautas de lectura escritura occidentales podrán ayudar en esto, ya que los usuarios leen de izquierda a derecha y de arriba abajo. Estas pautas nos van a proporcionar las reglas a seguir acorde a las ponderaciones en importancia otorgadas.

3.- En el caso de que la interfaz tenga ciertas particularidades por medio de las cuales no sea posible dotar al sitio web de indicadores claros de pulsado, puede ser aconsejable introducir alguna pista que enfatice los botones usados con más frecuencia o los más previsibles.

2.1.3.2. Desarrollo

Este tutorial es una guía que muestra algunas técnicas y funciones en Photoshop CS3 para el desarrollo de botones, como elementos Web o para una aplicación de escritorio. Las técnicas y herramientas del tutorial corresponden a la versión de Photoshop CS3, pero los pasos son similares para versiones superiores.

Tutorial botones en Photoshop

1.- Como primer paso crear un fondo para montar los botones.

1.1.- Crear una capa base de 400 x 400 píxeles.

1.2.- En las opciones de "Herramienta Degradado" crear un degradado del #212b37 al #3f5c7a. El resultado se muestra en la figura 2.16a

1.3.- Aplicar el degradado sobre toda la capa base, de izquierda a derecha, como se muestra en la figura 2.16b

1.4.- A este fondo aplicar un filtro de ruido, para ello ir a "Filtro → Ruido → Añadir ruido", con una cantidad de 1.5 % y una distribución Uniforme.

1.5.- Para terminar este paso cambiar el tipo de capa a "Multiplicar" o "Luz suave" con opacidad de 100 %. El resultado se muestra en la figura 2.17

2.- Crear una nueva capa con fondo transparente, nombrada "botones".

3.- Realizar 5 (o más) selecciones usando la "Herramienta Marco Regular (M)" con Estilo de "Tamaño fijo" con ancho de 210px y una altura 50px. Para hacer 5 selecciones rectangulares se mantiene presionado la

(a) Degradado. (b) Degradado de izquierda a derecha

Figura 2.16: Degradados. [Imágenes creadas por el autor de este trabajo (2011)]

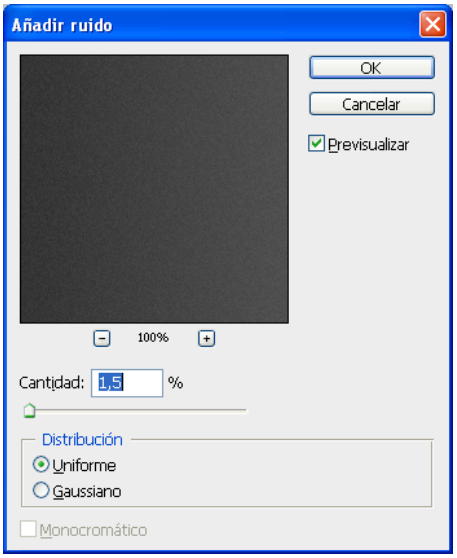
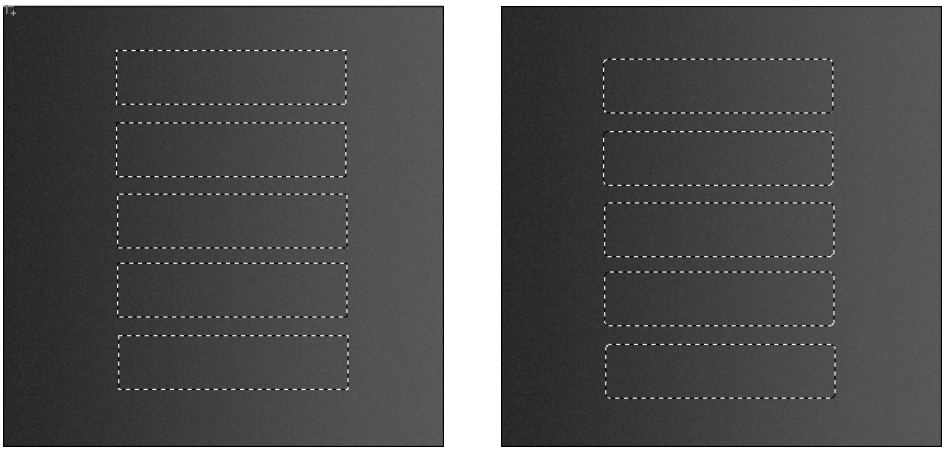


Figura 2.17: Filtro ruido [Imagen creada por el autor de este trabajo (2011)]

tecla shift o la opción "Añadir a selección" (icono con encimación de capas). Resultado en la figura 2.18a

4.- Redondear las esquinas de los rectángulos, para ellos ir a "Selección→ Modificar →Redondear", con radio de 3px. El resultado se muestra en la figura 2.18b

5.- Llenar los rectángulos con el color #2d3944 con el tipo de capa "Nor-



(a) Selecciones rectangulares. (b) Esquinas redondas.

Figura 2.18: Crear botones. [Imágenes creadas por el autor de este trabajo (2011)]

mal", como se muestra en la figura 2.19

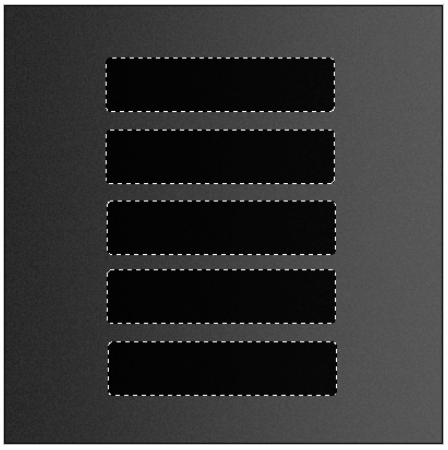


Figura 2.19: Rellenar los rectángulos. [Imagen creada por el autor de este trabajo (2011)]

6.- Aplicar efectos a los botones, para ello dar clic derecho sobre la capa botones y después en opciones de fusión:

6.1.- En la opción de Sombra interior, dar un ángulo de 106°, una distancia de 5px y tamaño de 5px. Como se muestra en la figura 2.20.

6.2.- En la opción de Resplandor exterior, establecer el color en blanco (#ffffff), con un tamaño de 4px y una opacidad al 30 %. Como se

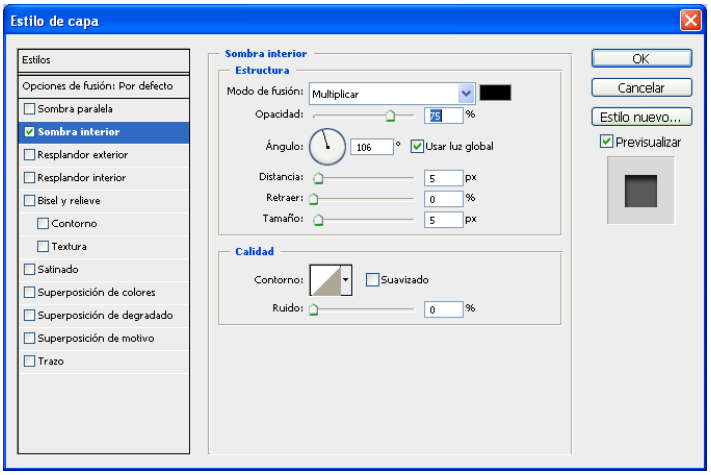


Figura 2.20: Sombra interior. [Imagen creada por el autor de este trabajo (2011)]

muestra en la figura **??**

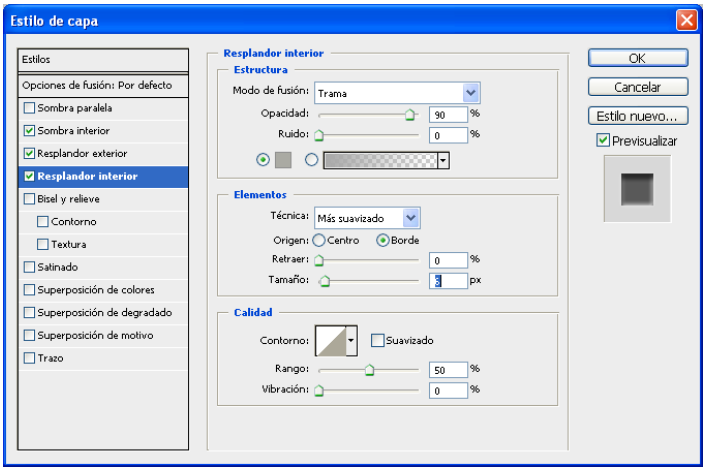


Figura 2.21: Resplandor interior y opacidad. [Imagen creada por el autor de este trabajo (2011)]

6.3.- En la opción de Resplandor interior, establecer el color #aaaaa3, una opacidad al 90 % y un tamaño de 6px. Como se muestra en la figura 2.21

6.4.- En la opción de Bisel y Relieve, poner una profundidad de 100 %, un tamaño de 7px, un ángulo de 106° y una opacidad del 75 % y seleccionar la opción de contorno con las opciones predefinidas. Como se muestra en la figura 2.22

6.5.- En la opción de Superposición de degradado, poner la capa en

modo Multiplicar, con una opacidad al 90 % y un ángulo de 72°. Como se muestra en la figura 2.23

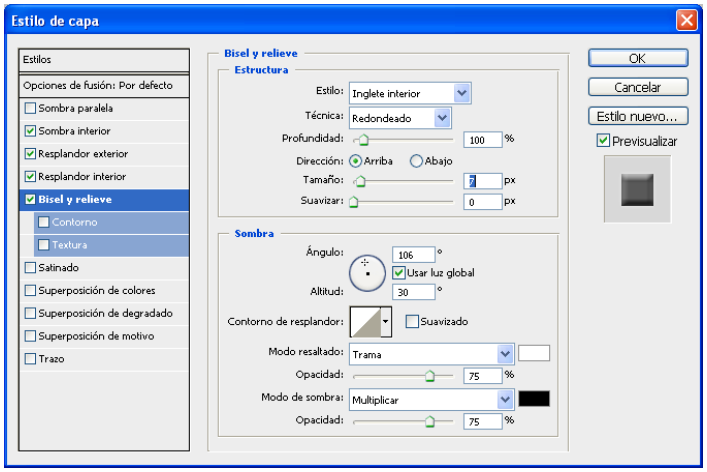


Figura 2.22: Bisel y relieve. [Imagen creada por el autor de este trabajo (2011)]

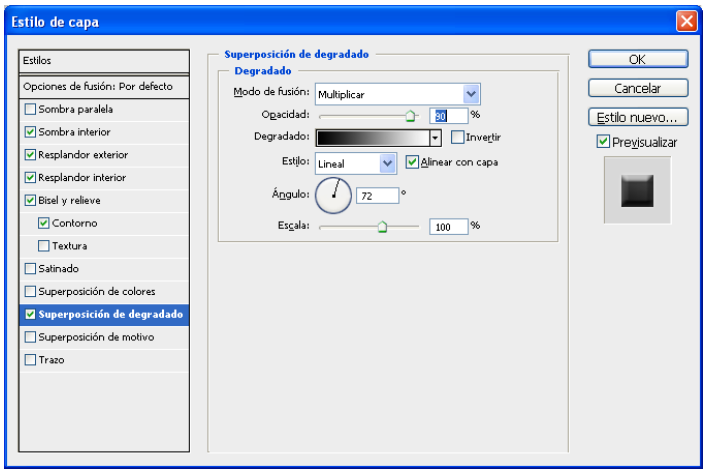


Figura 2.23: Superposición de capas. [Imagen creada por el autor de este trabajo (2011)]

7.- Crear una nueva capa antes de la capa de botones con nombre "sombra". Posicionados en la capa sombra dar la selección de los botones. Realizar algunos cambios a la selección, para ello ir a "Selección→Modificar→Expandir" dar como valor 7px, rellenar con el color #040508 y establecer el tipo de la capa como Multiplicar con una opacidad al 90 %. El resultado se muestra en la figura 2.24

Figura 2.24: Sombra en los botones. [Imagen creada por el autor de este trabajo (2011)]

8.- Crear una capa con los títulos de los botones con nombre "texto".

8.1.- Crear los botones TV, RADIO, MOVIL, REVISTA y VENTAS. El tipo de letra es Arial, con tamaño de 24pt y grosor Bold.

8.2.- Antes de unir el texto en una sola capa es necesario "Rasterizar capas" y luego "combinar capas".

9.- Crear una capa anterior a la capa de texto, de nombre "contorno", posicionados en ella hacer una selección del texto. Después en "Selección→ Modificar →Expandir" aplicar el tamaño a 3px. Cambiar la capa contorno a modo de Multiplicar y aplicar una opacidad del 90 %. El resultado se muestra en la figura 2.25.

10.- En la capa de texto realizar un ajuste en las opciones de fusión, en Bisel y relieve aplicar un tamaño de 9px, en estilo colocar "Inglete interior" a un tamaño de 9px. El resultado se muestra en la figura 2.26

11.- Para terminar crear una capa de nombre luz y aplicar un toque de luz únicamente a los botones, para ello hacer un degradado de blanco a transparente, y crear una máscara que afecte a los botones. Cambiar el tipo de capa a Luz intensa con una opacidad al 30 %.

Figura 2.25: Botones con contorno. [Imagen creada por el autor de este trabajo (2011)]

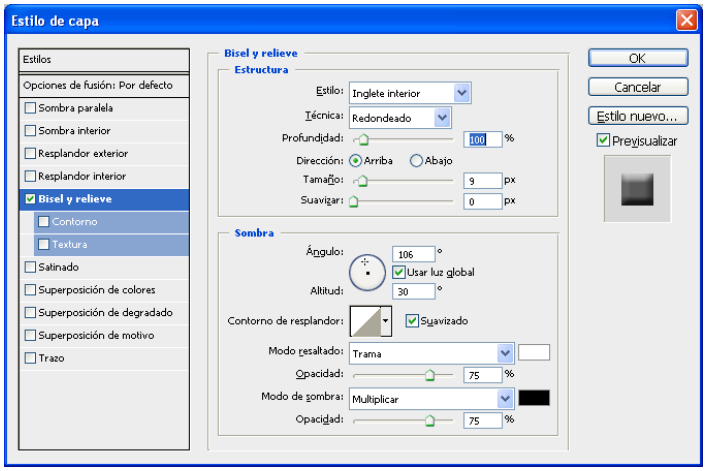


Figura 2.26: Bisel y relieve. [Imagen creada por el autor de este trabajo (2011)]

2.1.3.3. Ejercicios de Entrega

El alumno debe entregar el set de botones y el documento xcf (archivo de Gimp) o el psd (archivo de Photoshop). Este último archivo es muy importante, ya que muestra el trabajo en capas de las imágenes. También debe adjuntar un pdf donde explique las herramientas y técnicas de Photoshop que utilizó.

Estructura Clásica. El alumno debe generar una carpeta con la siguiente estructura de directorios:

- ◯ Carpeta “login de ciencias del estudiante”.
  - → Carpeta “imagenes”.
    - Trabajo xcf o psd realizado.
    - Imagen en png, jpg o gif, exportada de la fuente original.
  - → Carpeta “docs”.
    - Documento de reporte en formato pdf.

2.2. UML

UML (*Unified Modeling Language*, Lenguaje Unificado de Modelado) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group) Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un “plano” del sistema (o modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Se utiliza para definir un sistema, detallar los elementos del sistema y para documentar y construir el sistema. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el RUP (Proceso Unificado Racional)), pero no especifica en sí mismo qué metodología o proceso usar.

2.2.1. Práctica de UML

2.2.1.1. Enunciado

Usar UML como herrmienta de diagramación, para modelar y entender los requerimientos del sistema.

2.2.1.2. Práctica

Se solicita la implementación de los diagramas de casos de uso y casos de uso detallados que modelen el sistema que de solución a los requerimientos del siguiente problema.

Enunciado del problema Se necesita un sistema en donde varios usuarios puedan subir sus archivos de imágenes, solamente los usuarios registrados deben almacenar sus archivos, el único formato que se permite para el almacenamiento es jpg.

Si un usuario quiere ocupar el sistema primero se debe registrar, una vez cumplido este objetivo se le da un login y password, se solicita que proporcione un correo electrónico, por si el administrador quiere mandarle un aviso. El usuario puede administrar su contenido, lo cual puede borrar o cambiar de directorio sus archivos. También debe de tener la facultad de descárgalos cuando lo necesite. Si quiere el usuario puede compartir sus archivos a los demás usuarios que estén registrados. Todos comparten una solo carpeta para depositar los archivos que se comparten.

El administrador necesita poder borrar a usuarios del sistema y enviar un aviso a los usuarios registrados cuando lo necesite, es el único capacitado para borrar el contenido de la carpeta compartida de los usuarios.

Descripción del problema El Sistema de Compartición de archivos es un sistema que permite al usuario subir imágenes del tipo jpg o jpeg, compartirlas con otros usuarios, descargar imágenes propias y compartidas y mover sus imágenes entre directorios todo a través de un portal Web. El sistema presenta en su pantalla principal un mensaje de bienvenida describiendo los servicios que le ofrece al usuario junto con la opción de iniciar sesión o registrar un usuario en el sitio. En la pantalla se incluyen los campos de login y de password para que el usuario inicie sesión, datos que se proporcionaron en el registro del usuario. Estos datos deben ser validados para ingresar al perfil del usuario autenticado.

Para registrar un usuario, el sistema despliega una pantalla con campos de formulario que recolectan información del usuario, donde se incluye el login y el password que utilizará para ingresar al sistema como usuario autenticado. Antes de ingresar esos datos a la base, se deberán validar los campos según el tipo de datos que espera la base de datos y la disponibilidad de los datos de autenticación.

El sistema mostrará las imágenes en forma de mosaico según la carpeta en la que se encuentran, con una breve descripción de la imagen a su pie, datos que serán ingresados al subir una imagen; título de la imagen y breve descripción de ella. Otras operaciones relacionadas con las imágenes se encuentran al pie de la imagen, como son; eliminar imagen, editar información y compartir imagen con otros usuarios.

Al eliminar una imagen, el usuario logueado debe elegir la opción eliminar relacionada con la imagen en cuestión y aceptar la operación. Para compartir una imagen, el usuario logueado debe elegir la opción compartir relacionada con la imagen en cuestión, donde la aplicación desplegará una lista de usuarios pertenecientes a su entorno.

Para ingresar una nueva imagen de un usuario logueado, el sistema despliega un formulario con campos como título de la imagen, breve descripción de la imagen, un campo de archivo y el nombre del directorio en donde alojará la imagen.

Para agregar un usuario al entorno de un usuario logueado (ya sea como una relación de amistad, compañeros de trabajo, familiar, etc) el usuario accede a la sección de usuarios donde se despliega una lista de usuarios de la aplicación y un buscador de personas. En esta sección puede ocurrir lo siguiente:

1.- El usuario puede navegar por la lista de usuarios registrados en la aplicación. En la aplicación se despliega el nickname del usuario, su correo electrónico y la opción de invitación de amistad. Al elegir un usuario con la opción "enviar solicitud de amistad". el sistema envía una invitación al usuario elegido, dicha solicitud tiene como destino el perfil del usuario invitado. Con lo cual puede ocurrir que:

   1.1.- El usuario rechace la invitación con lo cual el sistema reenvía al usuario solicitante un mensaje, indicando que el usuario invitado rechaza la invitación de amistad.

   1.2.- El usuario acepta la invitación con lo cual el sistema reenvía al usuario solicitante un mensaje de aceptación y crea una carpeta en la cual se alojaran las imágenes compartidas entre usuarios.

2.- Que el usuario tenga el correo electrónico o el nickname del usuario que desea invitar, en dicho caso pude utilizar el buscador de usuarios. Al hacer una búsqueda el sistema arrojará una lista de usuarios que coinciden con los términos. En esta lógica puede ocurrir que el correo electrónico del usuario solicitado no se encuentra registrado en el sistema, en tal caso se le enviará una invitación para unirse a la comunidad.

Para eliminar a un usuario del entorno de amigos de un usuario logueado, el usuario debe ingresar a la sección de usuarios donde se despliega una lista de amigos. En esta sección las personas listadas presentan una breve información y acciones al pie; nickname, tipo de relación, eliminar del entorno. Al elegir eliminar, el usuario seleccionado deja de estar dentro del entorno del usuario solicitante.

Las actividades o acciones que el usuario logueado puede hacer son:

1.- Subir imágenes a una carpeta,

2.- Crear carpetas para almacenar imágenes,

3.- Descargar imágenes,

4.- Borrar imágenes, que no estén compartidas con otros usuarios,

5.- Borrar Carpetas, que no estén compartidas con otros usuarios.

6.- Compartir carpetas con otros usuarios.

7.- Mover imágenes entre carpetas.

8.- Renombrar imágenes.

9.- Invitar a un usuario a su entorno.

10.- Eliminar a un usuario de su entorno.

Las actividades o acciones que el usuario administrador puede hacer son:

1.- Borrar usuarios del sistema.

2.- Enviar correos electrónicos a los usuarios cuando lo necesite.

3.- Borrar el contenido de las carpetas compartidas entre usuarios.

4.- Borrar carpetas compartidas entre usuarios.

2.3. Notas de CSS

2.3.1. Introducción a CSS

CSS (*Cascading Style Sheets*, hojas de estilo en cascada) es un lenguaje de hojas de estilo creado para controlar la apariencia (aspecto) de los documentos escritos en HTML y XHTML.

El organismo (W3C) (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (*Cascading HTML Style Sheets*) y la SSP (*Stream-based Style Sheet Proposal*).

La propuesta CHSS fue realizada por Hakon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS.

Al utilizar CSS separamos la función de definir el aspecto de la página, de la función de definir el contenido (labor que desempeña HTML). Esta forma de diseñar presenta varias ventajas:

- Diseñar documentos HTML y XHTML bien definidos con significado completo (documentos semánticos).
- Mejora su accesibilidad del contenido.
- Facilita el mantenimiento del contenido y de las hojas de estilo.
- Permite que un mismo contenido se pueda visualizar en diferentes dispositivos (monitores, impresoras, dispositivos móviles, etc.)

Al diseñar una página utilizamos los documentos HTML para definir la estructura de la página, llenarla de contenido y "marcar" ese contenido. Marcar significa que se le asigna a cada elemento HTML su función dentro de la página.

Ejemplos:

- p $\longrightarrow$ Párrafo.
- h1, h2, ... , h6 $\longrightarrow$ Titulares.
- b, i, em, strong $\longrightarrow$ Destacar texto.

Una vez creado el contenido las reglas de CSS permiten definir el aspecto de cada elemento: color, tipo de letra, color de fondo, tamaño de letra, posición en la página etc.

2.3.1.1. Soporte de CSS

Los usuarios acceden a las páginas a través de los navegadores, por tal motivo es de gran importancia para el diseñador Web conocer el soporte o alcance de los principales navegadores respecto a los estándares de CSS.

Los navegadores Safari, Opera, Chrome, Firefox e IE 8 son los más avanzados en el soporte de CSS. Pero IE 6 y 7 siguen vigentes en el mercado y es casi seguro que te provocarán un dolor de cabeza en adaptar tus diseños.

Sin total control en el diseño

Los diseñadores Web no tienen total control sobre la apariencia final de la página en el navegador del cliente. Pongamos como ejemplo la tipografía usada en la página de la UNAM (http://www.unam.mx). En la figura 2.27 podemos ver que la tipografía usada es **Arial, Helvetica, sans-serif**

Para este ejemplo usamos Firefox 3.6.8, en las opciones de "preferencias", modificamos las opciones de "Contenido". Las opciones a modificar son la tipografía "FreeMono" y deshabilitamos la casilla que permite a las páginas utilizar la tipografía definida.

En la figura 2.29 se nota que la tipografía que muestra la página de la UNAM es muy diferente a la propuesta por sus diseñadores, esto no significa que se modificó la propiedad desde su CSS (eso sería un ataque al servidor), simplemente se está aplicando una nueva regla CSS que tiene mayor prioridad que las reglas propuestas en sus hojas de estilo y que se aplica a nivel local.

El poder y libertad que esto otorga a los usuarios va más allá de cambiar el tipo de fuente de una página; se puede cambiar colores, tamaños, bloquear imágenes, incluir hojas de estilo para dispositivos móviles, cambiar el estilo del documento para su impresión (principio de usabilidad), entre otros.

En la figura 2.29 se observa la aplicación firebug disponible para Firefox 3+ y google Chrome, la cual permite ver el código HTML de la página, las

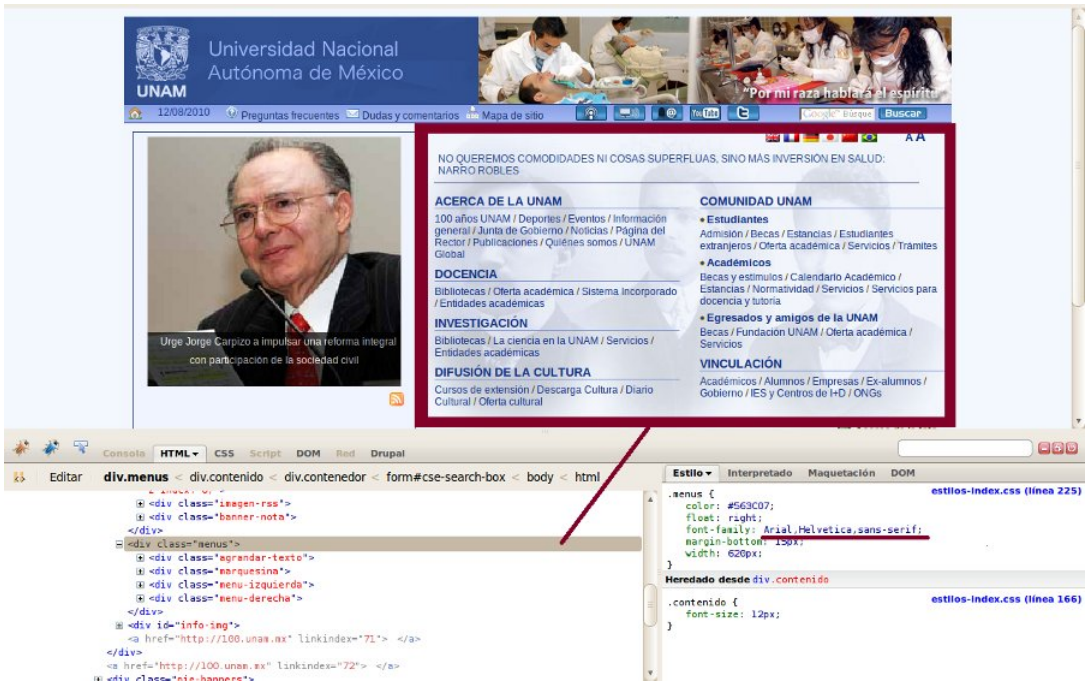


Figura 2.27: Tipografía usanda en la página http://www.unam.mx [Imagen creada por el autor de este trabajo (2011)]

hojas de estilo enlazadas al documento, el objeto DOM de la página y los script de JavaScript. Así como realizar modificaciones en los documentos y configurar pruebas de errores.

Una página Web no tiene el aspecto que el usuario ve en la ventana del navegador. El navegador recibe e interpreta el código fuente solicitado al servidor. Cuando el navegador del usuario carga la página completamente, crea el DOM, carga las imágenes, los archivos CSS, los archivos JavaScript y los archivos necesarios. Lo que el usuario ve es la interpretación que hace ese navegador de los archivos fuente.

Esto significa que las páginas web se visualizan de forma distinta porque el código fuente se interpreta de forma diferente en función del entorno; navegador, versión del navegador y necesidades del usuario final.

Crear una página que se muestre exactamente igual en todos los navegadores es casi imposible. Una página web no tiene que verse exactamente igual en todas partes, sino que debe funcionar y mantener accesible el contenido al usuario en todos los navegadores.

2.3.1.2. Funcionamiento del CSS

Antes del surgimiento de las hojas de estilo CSS, los diseñadores definian el aspecto de los elementos a través de la propiedad style de las etiquetas

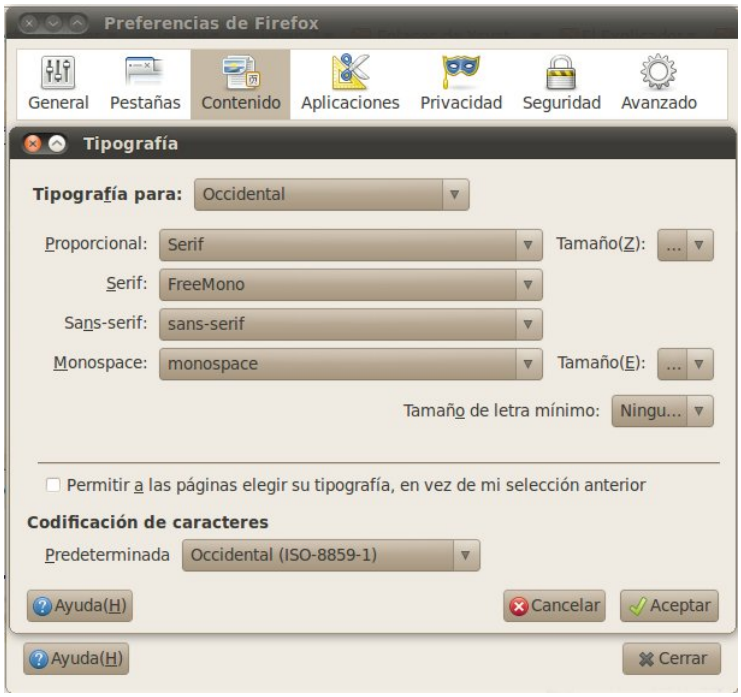


Figura 2.28: Modificando la tipografía desde opciones del navegador [Imagen creada por el autor de este trabajo (2011)]

HTML. El código 2.1 es un fragmento del documento HTML "funcionamiento_css_1.html" que utiliza la propiedad style:

Código 2.1: Uso propiedad style

```
<h1 style="color:#38005B; font-family:courier; font-size
    :28px;">
  Título de la página
</h1>
<p style="width:600px;color:#303030;background:#D0E68E;
    font-size:14px;">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Nam et elit. Vivamus placerat lorem. Maecenas sapien.
Integer ut massa. Cras diam ipsum,laoreet non, tincidunt
a, viverra sed, tortor.
</p>
```

El ejemplo anterior utiliza la propiedad style para modificar los atributos color, font-family y font-size del titular "h1" y los atributos width, color, background y font-size del párrafo.

Otra forma de aplicar estilos es desde la cabecera de la página HTML donde se puede definir estilos que se aplican a las distintas etiquetas HTML

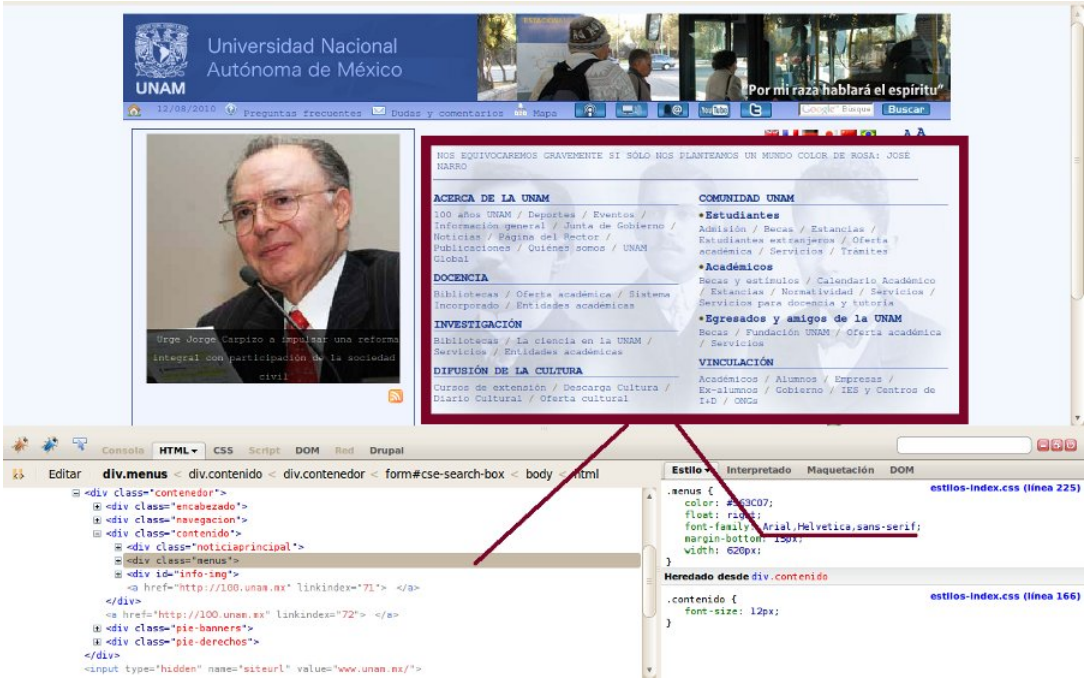


Figura 2.29: Página de la UNAM con la nueva tipografía
[Imagen creada por el autor de este trabajo (2011)]

de la página.

En el ejemplo anterior se tenía que definir los estilos a cada elemento, pero esto se puede solucionar al definirlos desde la cabecera usando la etiqueta style, como se muestra en el código 2.2 que es un fragmento del documento HTML "funcionamiento_css_2.html"

Código 2.2: Uso etiqueta style

```
<head>
<title>Ejemplo de estilos con CSS</title>
<style>
  h1{color:#38005B;
  font-family:Courier,monospace;
  font-size:28px;
  }
  p{width:600px;
  color:#303030;
  background:#D0E68E;
  font-size:14px;
  }
</style>
</head>
```

CSS permite separar los contenidos de la página y su aspecto. En el ejemplo anterior dentro de la propia página HTML se reserva una zona en la que se incluye toda la información relacionada con los estilos de la página.

En la etiqueta Style se indica que todas las etiquetas "h1" de la página deben poseer el color #38005B, con un tipo de letra Courier y un tamaño de letra de 28px. Además las etiquetas "p" de la página deben tener un color #303030, con fondo #D0E68E, un tamaño de letra de 14px y un ancho de 600px.

Definiendo los estilos de esta forma, no importa el número de elementos "p" que existan en la página, ya que todos tendrán el mismo aspecto establecido mediante CSS. Se puede observar que es mucho más eficiente que definir los estilos directamente sobre los elementos HTML.

2.3.1.3. Definir CSS en un archivo externo

La máxima optimización sobre la aplicación de reglas de estilo a un documento HTML es definirlas en un archivo externo que es enlazado desde la etiqueta $< link >$. Los archivos con reglas CSS tienen terminación ".css". Se pueden crear tantos archivos (reglas de estilo) como sean necesarios para llegar al aspecto deseado por el diseñador.

Cada documento HTML puede enlazar tantos archivos CSS como sea necesario y un archivo CSS puede ser enlazado por varios archivos HTML.

Para optimizar el archivos HTML anterior (`funcionamiento_css_2.html`) se puede crear un archivo CSS, de nombre "style.css" que contenga las reglas ejemplificadas en la etiqueta "head" para después enlazarlo. Los archivos del ejemplo se localizan en "complementos/funcionamento_css/ejemplos.

Código 2.3: Contenido del archivo style.css

```
h1{color:#38005B;
   font-family:Courier;
   font-size:28px;
}
p{width:600px;
  color:#303030;
  background:#D0E68E;
  font-size:14px;
}
```

Actual header del documento HTML `funcionamiento_css_2.html`:

Código 2.4: Vincular el archivo externo

```
<head>
<link rel="stylesheet" type="text/css" href="css/style.
   css" media="screen" />
```

```
<title>Ejemplo de estilos con CSS</title>
</head>
```

La etiqueta $< link >$ incluye cuatro atributos cuando se enlaza un archivo CSS:

- rel: indica la relación que tiene el recurso enlazado (en este caso, un archivo CSS) la página HTML.
- type: indica el tipo de recurso enlazado, para los archivos CSS su valor siempre es text/css
- href: indica la URL del archivo CSS que contiene los estilos.
- media (2.3.5.2): indica el medio en el que se van a aplicar los estilos del archivo CSS.

Generalmente se emplea la etiqueta $< link >$ para enlazar los archivos CSS externos, también se puede utilizar la etiqueta $< style >$ con ayuda del comando @*import*.

La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

El código 2.5 presenta una modificación en la etiqueta "head" para vincular el archivo "style.css" utilizando @*import*.
Actual header del documento HTML `funcionamiento_css_2.html`:

Código 2.5: uso de @*import*

```
<head>
<style>
   @import url('css/style.css');
</style>
<title>Ejemplo de estilos con CSS</title>
</head>
```

2.3.1.4. Práctica 01

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.3.1 Introducción a CSS, en cuyas secciones se tratan las técnicas básicas para agregar reglas de estilo a una página Web. En los ejercicios se plantean construir y aplicar estilos a un documento HTML a través de la propiedad style de los elementos HTML, agregar reglas de estilo utilizando la etiqueta "style" en el head de la página e incluir documentos de estilos externos.

Ejercicios Los documentos HTML de esta práctica se localizan en "complementos/funcionamiento_css/ejercicios".

1.- ¿Cuáles son las ventajas y desventajas de agregar estilos desde la propiedad style de las etiquetas html?

2.- Dar el estilo obtenido en la figura 2.30 al documento HTML funcionamiento_css.html, utilizando únicamente la propiedad style de sus elementos.[1]

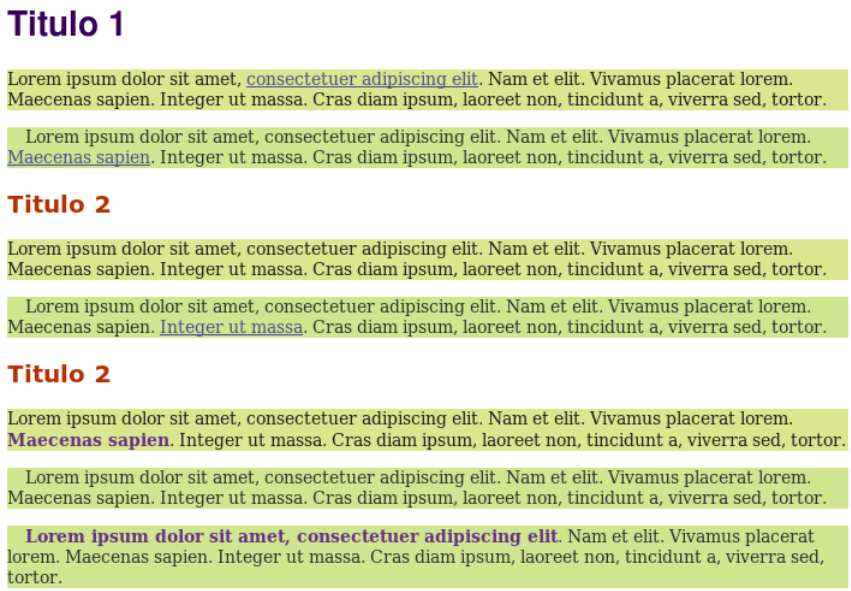


Figura 2.30: Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)]

3.- ¿Cuáles son las ventajas y desventajas de agregar estilos desde la cabecera del documento HTML usando la etiqueta style?

4.- Dar el estilo mostrado en la figura 2.31 al documento HTML funcionamiento_css2.html utilizando la etiqueta style.

5.- Crear los archivo css externos que contengan los estilos definidos en los ejercicios 2 y 4. No olvides que los archivos con extensión ".css" van dentro de una carpeta css (la carpeta de trabajo es "ejercicio5").

5.1.- Probar el aspecto de los documentos al intercambiar sus hojas de estilo entre sí.

5.2.- Optimizar las hojas de estilo, es decir, colocar en una hoja de estilo las reglas comunes a los dos documentos y en otra las individules.

[1] Como recomendación utilizar la aplicación gpick (linux) para obtener los colores exactos.

Figura 2.31: Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)]

6.- ¿Cuáles son las ventajas y desventajas de agregar estilos enlazando archivos externos al documentos HTML?

7.- Al documento HTML "ejercicio7/css_medios.html" definir los siguientes estilos CSS. Se recomienda ver el contenido 2.3.5.2 Medios css

- screen:

7.1.- Titular principal: color de letra #1A1A1A, tamaño de letra 30px.

7.2.- Titular secundario: color de letra #420B5D, tamaño de letra 24px y una separación de sus vecines de 5px.

7.3.- Párrafos: color de fondo #EBFDB8, color de letra #4D4D4D, un tamaño de 700px, tamaño de letra de 14px y separación interna de 3px.

7.4.- Listas: tamaño de letra 15px, separación entre vecinos 25px.

- print:

7.1.- Titular principal: color de letra #38005B, tamaño de letra 28pt.

7.2.- Titular secundario: color de letra #1C5A00, tamaño de letra 20pt.

7.3.- Párrafos: color de letra #000646, separación con el vecino superior de 10px.

7.4.- Listas: color de letra #1C5A00 y letras en negritas.

2.3.2. Selectores CSS

2.3.2.1. Reglas CSS

CSS define una estructura que permite describir cada una de las partes que componen los estilos CSS. El esquema 2.32 muestra las partes que forman una regla de estilo CSS muy básica:

Figura 2.32: Partes de una regla de estilo [Imagen creada por el autor de este trabajo (2011)]

- Regla: cada regla está compuesta por "Selectores" y entre llaves ({}) la "declaración".
- Selector: indica los elementos HTML a los que se aplica el estilo (ej. p, h1, h2, h3, em, pre, i, etc.).
- Declaración: indica los estilos que se aplican, formado por la propiedad y el valor.
- Propiedad: indica la propiedad a modificar, ej. color, width, font-size, background, etc.
- Valor: indica el valor del cambio a dicha propiedad, ej. #00f, 300px, 12pt, #0f0, etc.

Código 2.6: Ejemplo de encadenamiento de selectores

```
h1, h2, h3{
    color:#38005B;
    font-family:Courier;
    font-size:28px;
    font-weight:600;
    margin:0 auto;
}
```

Para entender la definición se suele decir que la declaración nos dice "que hay que hacer" y los selectores indican "a quienes tenemos que aplicarlo".

Normalmente un sito Web contiene muchas hojas de estilo CSS, cada CSS puede definir muchas reglas, cada regla puede contener muchos selectores y cada declaración puede definir muchos pares propiedad-valor. A un mismo elemento HTML (selector) se le puede asignar una infinidad de reglas y cada regla puede ser aplicada a un número infinito de selectores.

2.3.2.2. Selectores básicos

Selector Universal

El selector universal se representa con (*), y significa que selecciona cualquier elemento de la estructura de la página. En la práctica no es muy utilizada ya que es difícil encontrar diseños en donde sea necesario aplicar el mismo estilo a todos los elementos de la página.

Se utiliza en combinación con otros selectores para crear una regla más específica, así como en la aplicación de "hacks" para los navegadores IE.

Código 2.7: Ejemplo del selector *

```
* {
  margin: 0;
  padding:0;
}
```

En el ejemplo de código 2.7 se seleccionan todos los elementos de la página para eliminar su margen y relleno.
Aunque el selector universal no es muy usado es bastante útil y puede ahorrar líneas de código. Ya que sirve para dar un estilo a cualquier elemento, sea del tipo que sea. En combinación con otros selectores se obtienen resultados como el que se muestra a continuación.

Código 2.8: Fragmento de selectro_universal.html

```
// Reglas de CSS:
p * {
  color:#00F;
  font-style:italic;
}
<!-- Código HTML: -->
<p>
<span>Lorem ipsum dolor sit amet</span>, consectetuer
adipiscing elit.<b>Nam et elit</b>.Vivamus placerat
lorem.Maecenas sapien. Integer ut massa. Cras diam
ipsum,laoreet non, tincidunt a, viverra sed, tortor.
<a href="">tincidunta, viverra sed, tortor</a>
</p>
```

En el ejemplo de código 2.8 los elementos "span", "b", "a" que se encuentran dentro de un "p" reciben el estilo de color azul (#00F) y el estilo de letra cursiva.

Si el selector universal es el único componente de un selector simple se puede omitir. Por ejemplo:

- *[LANG=es] == [LANG=es]
- *.error == .error
- *#exito == #exito

Selector de etiqueta

El selector de etiqueta hace referencia al nombre de la etiqueta HTML a la que se desea aplicar la regla. El selector de etiqueta equivale a la instancia del tipo de elemento de la estructura del documento HTML.

Código 2.9: Selector de etiqueta

```
p{
  color:#00F;
}
h1{
  color:#0f0;
}
```

En el código 2.9 se seleccionan todos los elementos que coinciden con las etiquetas definidas en las reglas, en este caso las etiquetas "p" y "h1", a las que les aplica un estilo de color azul y verde respectivamente.

Si se desea aplicar un mismo estilo a dos o más etiquetas se pueden encadenar los selectores, utilizando comas para separarlos.
Supongamos las siguientes reglas de estilo para las etiquetas "h1", "h2" y "h3":

```
h1{
  color:#00F;
  font-family:Arial, sans-serif;
}
h2{
  color:#00F;
  font-family:Arial, sans-serif;
}
h3{
  color:#00F;
  font-family:Arial, sans-serif;
  font-style:italic;
```

```
13 }
```

En el ejemplo anterior los 3 selectores tienen un estilo en común, es decir, que se les aplica el color de fuente azul y tipografía Arial o sans-serif, asi que es posible separar ese estilo y encadenar los selectores para ahorrar líneas de definición, como se muestra en el código 2.10.

Código 2.10: Encadenamiento de selectores

```
1 h1, h2, h3{
2   color:#00F;
3   font-family:Arial, sans-serif;
4 }
5 h3{
6   font-style:italic;
7 }
```

En el ejemplo es notorio el uso de selectores encadenados para optimizar la hoja de estilo, al trabajar de esta manera se ahorra tiempo durante la carga de la página y se crean documentos CSS accesibles en el manteniemiento y en su reutilización.

Selector descendente

Selecciona los elementos que se encuentran **dentro** de otros elementos. Se dice que un elemento es descendiente de otro cuando su definición se encuentra entre las etiquetas de apertura y cierre del elemento contenedor.

En el código 2.11 el elemento "span" es descendiente del elemento "p", ya que se encuentra definido entre las etiquetas $< p >$ y $< /p >$. Siguiendo la misma lógica el elemento "span" es descendiente de "b" y este último es descendiente de "p". La regla CSS es aplicada a todos los elementos span descendientes de un elemento "p".

Código 2.11: Elementos descendentes

```
1 p span{
2   color:#00f;
3   font-style:italic;
4 }
5 <p>
6 Lorem ipsum ad his <b>scripta <span>blandit
7 partiendo</span>, eum fastidii</b> accumsan
8 euripidis in, eum liber hendrerit an. Qui ut
9 wisi vocibus suscipiantur,quo dicit ridens
10 inciderint id.Quo mundi lobortis reformidans.
11 </p>
```

En el ejemplo de código 2.11 la regla CSS aplica un color azul y un estilo de fuente cursiva a todos los elementos "span" descendientes de un elemento "p", aunque el "span" del ejemplo es descendiente de un elemento "b", la regla es aplicada por definición. Como el elemento "span" es descendiente de ambas etiquetas se aplican las reglas heredadas de sus "padres" y las directas, siendo las más específicas las últimas en aplicarse.
La sintaxis básica del selector descendente es:

```
elemento1 elemento2...elementoN{<declaracion>;<declaracion>;...<
    declaracion>}
```

Selector de clase

En muchas ocasiones se necesita aplicar un estilo a ciertos elementos en un contexto específico del diseño, por ejemplo, si se desea que todos los párrafos contenidos por un "div", tengan una sangría de 10px salvo el primer párrafo. Los selectores antes mencionados nos ayudan en aplicar la sangría a todos los párrafos, pero tenemos el problema específico de quitarle la sangría al primero. En esta situación necesitamos aplicar una clase a los primeros párrafos que cumplan con el contexto del diseño.

Para aplicar una clase a un elemento se utiliza el atributo class en HTML y en el documento CSS se indica anteponiendo un punto (.).
Sintaxis básica de selector por clase

```
.nombre_clase{<declaracion>;<declaracion>;...<declaracion>}
```

El código 2.12 muestra la situación planteada.

Código 2.12: Selector por clase

```
/*-----Reglas de CSS ------*/
div p{text-indent:15px;}
div .primer_parrafo{text-indent:0;}

<!--Código HTML -->
<div>
<p class='primer_parrafo'>Lorem ipsum ad
his scripta blandit partiendo...</p>
<p>Ius id vidit volumus mandamus, vide
veritus democritum te...</p>
<p>Blandit incorrupte quaerendum in quo,
nibh impedit id vis...</p>
</div>
```

En el ejemplo anterior se aplica la clase "primer_parrafo" al primer elemento "p" del grupo, enseguida se agregan las reglas CSS; la primera va

dirigida a todos los párrafos contenidos en un "div", a los que les aplica una sangría de 15px y la segunda regla va dirigada a todos los elementos HTML contenidos en un "div" cuyo valor en el atributo class sea "primer_parrafo", para quitarles la sangría.

Los selectores de clase son imprescindibles para diseñar páginas, ya que permiten aplicar estilos específicos con gran precisión. Pero, qué ocurre cuando varios elementos coinciden en algún valor en su atributo class.

Código 2.13: Ejemplo del selector por clase

```
/*-----Reglas de CSS ------*/
.importante{color:#4D4D4D; font-weight:bold;}

<!--Código HTML -->
<div>
<p class="importante">
Lorem ipsum ad his scripta blandit partiendo...
</p>
</div>
<div class="importante">
Eum hinc argumentum te, no sit percipit
adversarium, ne qui feugiat persecuti. Odio
omnes scripserit ad est, it vidit lorem
maiestatis his, putent mandamus gloriatur ne
ne his deserunt perpetua sententiae.
<\div>
```

En el ejemplo de código 2.13 se aplica el estilo a todos los elementos que coincidan con el valor "importante" en el atributo class.

Si se requiere aplicar el estilo únicamente a los párrafos cuyo valor del atributo class coincida con "importante", es necesario especificarlo en la aplicación de la regla, para ello se utiliza el nombre de la etiqueta seguida del nombre de la clase, el resultado se muestra en el código 2.14.

Código 2.14: Selector de clase más específico

```
/*-----Reglas de CSS ------*/
p.importante{color:#4D4D4D; font-weight:bold;}

<!--Código HTML -->
<div>
<p class="importente">
Lorem ipsum ad his scripta blandit partiendo...
</p>
</div>
<div class="importante">
Eum hinc argumentum te, no sit percipit
adversarium, ne qui feugiat persecuti. Odio
```

```
13  omnes scripserit ad est, it vidit lorem maiestatis
14  his, putent mandamus gloriatur ne pro. Oratio
15  iriure rationibus ne his, ad est corrumpit splendide.
16  Ad duo appareat moderatius, ei falli tollit denique eos.
17  <\div>
```

Algunas recomendaciones sobre el uso de clases en CSS.

- Es posible aplicar más de un estilo a un mismo elemento HTML, esto se consegue agregando más clases a un elemento utilizando comas en la definición. En el código 2.15 se aplican las clases negritas y literal al elemento "p".

Código 2.15: Múltiples clases

```
1  /*-----Reglas de CSS ------*/
2  .negritas{font-weight:bold; color:#4D4D4D;}
3  .literal{font-family:"Times New Roman";}
4
5  <!--Código HTML -->
6  <p class="negritas literal">Lorem ipsum ad his
7  scripta blandit partiendo...</p>
```

- Es posible utilizar estilos de forma jerárquica, es decir, aplicar un estilo a aquellos elementos que tengan una clase específica previa. En el código 2.16 el primer párrafo recibe el estilo negritas, el segundo "literal" y el último párrafo las clases.

Código 2.16: Jerarquía de clases

```
1  /*-----Reglas de CSS ------*/
2  .negritas{font-weight:bold; color:#4D4D4D;}
3  .literal{font-family:"Times New Roman";}
4  .literal.negritas{font-style:italic;}
5
6  <!--Código HTML -->
7  <p class="negritas">párrafo cuyo peso de
8  texto es bold</p>
9  <p class="literal">párrafo con estilo
10 literal impreso</p>
11 <p class="literal negritas">párrafo con
12 los anteriores estilos y en italica</p>
```

- Es posible aplicar una clase en función de una etiqueta específica.

En el código 2.17 se aplica la regla únicamente a los elementos "p" cuyo valor en el atributo class sea "literal", por lo tanto el "span" con esa misma clase no recibe los estilos.

Código 2.17: Clase a un elemento específico

```
/*-----Reglas de CSS ------*/
p.literal{font-family:"Times New Roman";}

<!--Código HTML -->
<p class="literal">párrafo cuyo peso de
texto es bold</p>
<span class="literal">párrafo con estilo
 literal impreso</span>
```

Selector ID

En ocasiones es necesario aplicar un estilo a un único elemento del documento, para ello se recurre al valor del atributo ID. Por definición este selector aplica el estilo a un único elemento del documento HTML, ya que el valor de id no se puede repetir en el documento HTML.
Sintaxis básica del selector por id.

```
#nombre_id{<declaracion>;<declaracion>;...<declaracion>}
```

Supongamos que se desea aplicar un estilo específico a un párrafo en particular, para lograrlo se utiliza el valor de su identificador. El código 2.18 es un ejemplo de esa situación.

Código 2.18: Selector por ID

```
/*-----Reglas de CSS ------*/
p{text-indent:15px;}
#primer_parrafo{text-indent:0;}

<!--Código HTML -->
<div>
<p id='primer_parrafo'>Lorem ipsum ad his
 scripta blandit partiendo...</p>
<p>Ius id vidit volumus mandamus, vide
veritus democritum te...</p>
<p>Blandit incorrupte quaerendum in quo,
nibh impedit id vis...</p>
</div>
```

En el código 2.18 la primera regla CSS aplica una sangría de 15px a

todos los párrafos del documento y la segunda regla quita la sangría (0px) al elemento cuyo id coincida con el valor "primer_parrafo".

A pesar de que el valor del atributo id de los elementos en un mismo documento HTML son diferentes entre sí, es posible precisar una regla de estilo en donde este involucrado el atributo id, esto se debe a que si se utilizan hojas de estilo importadas a diferentes documentos HTML estos pueden definir valores de id iguales pero aplicadas a etiquetas diferentes, ej. en un documento se define un id con valor "importante" a un titular principal ("h1") y en otro se define un id con ese mismo valor a un titular secundario ("h2") por supuesto a ambos documentos se les aplica la misma hoja de estilos, por lo tanto si deseamos precisar un estilo a un elemento específico, se debe anteponer el nombre de la etiqueta seguido de # y el valor del id con las declaraciones de la regla.
Sintaxis del selector id con etiqueta específica.

```
h1#nombre_id{<declaracion>;<declaracion>;...<declaracion>}
```

El código 2.19 muestra una situación similar.

Código 2.19: Selector ID más específico

```
/*-- Definida en style.css--*/
span#aviso{color#00f;}

<!--doc1.html vincula  style.css -->
<p id='aviso'>Lorem ipsum ad his scripta
blandit partiendo</p>

<!--doc2.html vincula style.css -->
<span id='aviso'>Lorem ipsum ad his scripta
blandit partiendo</span>
```

En el código 2.19 la regla definida en style.css solo es aplicada al elemento span del documento "doc2.html".

2.3.2.3. Selectores avanzados

En CSS 2.1 [2] se definen otros selectores que ahorran el tiempo simplificando estilos. El soporte de estos selectores es distinto en los diferentes navegadores, por lo que es recomendable conocer los selectores soportados por cada navegador.

[2]El navegador Internet Explorer 6 y sus versiones anteriores no soportan los selectores avanzados

Selector del descendiente inmediato

Se utiliza para seleccionar a los elementos que sean descendientes inmediatos de otro elemento.

Sintaxis básica de selector descendiente inmediato:

```
elementoA > elementoB{<declaracion>;<declaracion>;...<declaracion>}
```

En el código 2.20 aplica un color de letra azul al elemento "span" que sea hijo inmediato de un párrafo.

Código 2.20: Selector por denscendencia inmediata

```
p > span{color:#00f;}
```

Selector adyacente

Selecciona el elementoB de la siguiente sintaxis:

```
elementoA + elementoB{<declaracion>;<declaracion>;...<declaracion>}
```

- El elementoA y elemetoB están definidos al mismo nivel de la estructura y su elemento contenedor es el mismo.
- En la estructura HTML el elementoB debe estar definido inmediatemente después del elementoA.

Supongamos el caso de la sección de comentarios de un blog, se puede definir un titular secundario para el nombre del autor del comentario y un "span" en el cual se muestre la fecha de publicación. El código 2.21 ejemplifica esta situación.

Código 2.21: Ejemplo del selector adyacente

```
/*------Reglas de CSS ------*/
h2.autor + span{color:#00f; font-style:italic;
 font-size:12px;}

<!--Código HTML -->
<div class="comentario">
<h2 class="autor">Pancho Villa</h2>
<span>Publicado: 10 Sep 2010</span>
<p>Lorem ipsum ad his <span>scripta </span>
blandit partiendo<p>
<span>Relacionados <a>enlace 1</a> <a>enlace
 2</a></span>
</div>
```

El código 2.21 indica que se debe aplicar el estilo al elemento "span" que este definido inmediatamente después que un elemento "h2" cuyo valor del atributo class sea "autor" y que su elemento contenedor sea el mismo. El primer "span" recibe este estilo por que cumple con la definición del selector adyacente, pero el "span" definido dentro del párrafo y el definido después del párrafo, no cumplen con la adyacencia inmediata por lo tanto no reciben la regla.

Selector de atributos

Selecciona elementos a partir de su atributo o valor de su atributo HTML.

- [*nombre_atributo*] Selecciona todos los elementos que tienen establecido el atributo "nombre_atributo", sin importar el valor. Ejemplo: Aplica un borde de color azul y un grosor de 1px a las imagénes que tienen definidas el atributo "alt" sin importar el valor.

```
1 img[alt]{border:1px solid #00f;}
```

- [*nombre_atributo* = *valor*] Selecciona todos los elementos que tienen establecido el atributo "nombre_atributo" y el valor especificado. El código 2.22 aplica un borde de color verde con un grosor de 1px, un relleno de 3px a las imagénes que tienen definidas el atributo "alt" con el valor "fotografia".

Código 2.22: Selector por atributo y valor

```
1 img[alt="fotografia"]{border:1px solid #0f0; padding
    :3px;}
```

- [*nombre_atributo* ~= *valor*] Selecciona todos los elementos que tienen esteblecido el atributo "nombre_atributo" y al menos unos de los valores del atributo es "valor".

  En el código 2.23 aplica un borde de color verde con un grosor de 1px, un relleno de 3px a las imagénes que tengan definidas el atributo class y al menos uno de sus valores sea "galeria".

Código 2.23: El atributo coincida al menos con un valor

```
1 /*--- Reglas de CSS ---*/
2 img[class~="galeria"]{border:1px solid #0f0; padding
    :3px;}
3 <!-- Código HTML -->
```

```
4 <img class="fotografia galeria" src="" width=""
     height="">
```

En el código 2.23 el elemento "img" presenta dos clases por lo tanto cumple con tener el atributo class y al menos una de las clases coincide con el valor especificado en la regla.

- [*nombre_atributo*| = *valor*] Selecciona todos los elementos que tienen esteblecido el atributo "nombre_atributo" y cuyo valor es una serie de palabras que comienzan con valor y continúan separados por un guión (-).

  El código 2.24 aplica un borde de color verde con un grosor de 1px, un relleno de 3px a las imagénes que tengan definidas el atributo class y el valor comience con "galeria", es útil cuando se implementan galerías de imágenes.

Código 2.24: Valor del atributo que comienza galeria

```
1 /*--- Reglas de CSS ---*/
2 img[class |="galeria"]{border:1px solid #0f0;
     padding:3px;}
3 <!-- Código HTML -->
4 <img class="galeria-1" src="ejemplo1.png" width=""
     height="">
5 <img class="galeria-2" src="ejemplo2.png" width=""
     height="">
```

En el código 2.24 los elementos "img" presentan clases "galeria-1" y "galeria-2" respectivamente, por lo tanto cumplen con tener el atributo class e iniciar con la palabra "galeria".

Para conocer más sobre selectores leer la sección extra de selectores en 2.3.5.3.

2.3.2.4. Práctica 02

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.3.2 Selectores CSS, en cuyas secciones se tratan los selectores básicos y avanzados de CSS, se explica la forma para construir reglas de estilo, se definen y ejemplifican los conceptos básicos para seleccionar elementos de una página HTML. Además en la capítulo se defienen los selectores avanzados correspondientes al estándar CSS 2.1.

Ejercicios Los documentos HTML de esta práctica se localizan en "complementos/selectores_css/ejercicios".

1.- ¿Qué son los hacks de IE y cuál es la función del selector universal? Revisar la sección 2.3.2.2 Selector Universal

2.- Aplicar el siguiente estilo al documento HTML "selector_universal/ selector_universal.html":

- 2.1.- A todos los elementos incluidos en los enlaces aplicarles un color de letra azul.
- 2.2.- A todos los elementos dentro de una estructura de lista aplicarles un color de letra #4D4D4D y un tamaño de letra de 15px.
- 2.3.- A todos los elementos dentro de un elemento de lista aplicarles un color de letra #232323 y un peso de letra de 600.
- 2.4.- A todos los elementos dentro de un titular secundario aplicarles un color de letra #4848DE y un estilo de letra italic.
- 2.5.- A todos los elementos dentro de la etiqueta "b" aplicarles un color de letra #4D4D4D una decoración de texto subrayado y un estilo de letra italic.
- 2.6.- A todos los elementos dentro de la etiqueta "i" aplicarles un color de letra #8E3B3B y un peso de letra de 600.

3.- Defina cuatro propiedades diferentes a las propuestas usando el selector universal para el documento HTML anterior.

4.- ¿Cuáles son las diferencias entre los selectores descendentes y los selectores encadenados? ¿Se pueden combinar estos selectores? ejemplifica tu respuesta.

5.- Aplicar el siguiente estilo al documento HTML "selector_descendente/ selector_descendente.html": Revisar la sección 2.3.2.2 Selector descendente

- 5.1.- A todos los párrafos dentro de un elemento "div" aplicarles un color de letra #404040 y un ancho de 800px.
- 5.2.- A todos los enlace dentro de un párrafo aplicarles un color de letra #0006B2 y tamaño de letra de 16px.
- 5.3.- A todos los "span" dentro de un elemento "div" aplicarles un color de letra #A52A2A.
- 5.4.- A todos los "em" dentro de un párrafo aplicarles un color de letra #C80F00 y el texto debe estar subrayado.

5.5.- A todos los "span" dentro de cualquier elemento que a su vez esten dentro de un div aplicarles un tamaño de letra de 15px.

5.6.- A todos los "em" dentro de un "span" que a su vez esten dentro de cualquier elemento que se encuentre dentro de un "div" aplicarles un peso de letra "negritas".

5.7.- A todos los "em" dentro de una lista aplicarles un color de letra #840773.

5.8.- A todos los "em" dentro de cualquier elemento que a su vez este dentro de un elemento de lista definido en una lista aplicarles un rayado superior.

6.- Crear un ejemplo para cada situación, es recomendable revisar la sección 2.3.2.2 Selector de clase

- A dos elementos con clases coincidentes agregarles una situación jerárquica que les aplique un estilo adicional diferente entre si.
- A dos elementos con clases coincidentes agregarles una situación jerárquica que elimine los estilos de uno, pero sin afectar al otro.

7.- Encontrar el valor de las **Z** faltantes en el archivo "selector_clase/css/style2.css", para que el documento "selector_clase/selector_clase2.html" tenga el aspecto de la figura 2.33.

Figura 2.33: Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)]

8.- Crea una estructura HTML similar a la del ejercicio anterior con sus respectivas reglas de estilo, con una sección de comentarios en donde cada comentario puede tener respuesta por parte de otros usuarios. ej. la figura 2.34 muestra los comentarios en youtube.

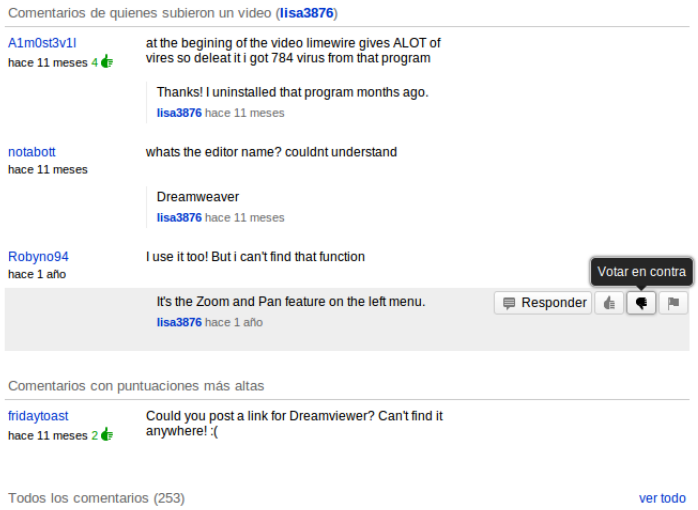


Figura 2.34: Encuentra las reglas de estilo [Imagen creada por el autor de este trabajo (2011)]

9.- Dar un ejemplo en el cual se aplica un estilo a un elemento según el elemento contenedor inmediato (padre). Una misma clase puede aplicar un estilo diferente en función de su elemento padre.

10.- ¿Es posible utilizar el selector de clase para aplicar un estilo únicamente a un elemento? ¿Qué es lo más recomendable para esta situación? Justifica tu respuesta.

11.- Describe con letra el significado de las siguientes reglas. Por ejemplo:

```
Aplica la <declaracion A> a todos los párrafos en el
documento cuyo valor del atributo id es principal
p#principal{<declaracion A>}
```

```
#aviso{<declaracion A>}
p #error{<declaracion A>}
p  span#ok{<declaracion A>}
.mensaje  p#aviso{<declaracion A>}
p.mensaje  span#error{<declaracion A>}
```

12.- Describe con letra el significado de las siguientes reglas.

```
p > span#aviso{<declaracion A>;<declaracion B>}
span > #error{<declaracion A>}
#ok > span{<declaracion A>}
.aviso span > b{<declaracion A>}
```

13.- Explica a que elementos HTML se les aplican las siguientes reglas de CSS.

13.1.-

```
/*--- Reglas de css ---*/
p a{<declaracion C>}
p > a{<declaracion A>;<declaracion B>}

<!-- Código HTML -->
<p>Lorem ipsum ad his <a>scripta </a>blandit
partiendo<p>
<p>Lorem ipsum <span> ad his <a>scripta </a>
blandit </span>
partiendo<p>
```

13.2.-

```
/*--- Reglas de CSS ---*/
.aviso #error{<declaracion D>}
.aviso p > #error{<declaracion A>;
<declaracion E>}

<!-- Código HTML -->
<p class="aviso">Lorem ipsum <a> ad his <span
 id="error">scripta </span> blandit partiendo
</a><p>
<div class="aviso">
<p>
Lorem ipsum <span id="error"> ad his <a>
scripta </a> blandit </span> partiendo
<p>
</div>
```

13.3.-

```
/*--- Reglas de CSS ---*/
.aviso p{<declaracion D>}
.aviso > p{<declaracion A>;<declaracion E>}

<!-- Código HTML -->
<div class="aviso">
<p>
Lorem ipsum ad his <b>scripta </b> blandit
partiendo.
```

```
<p>
<div>
<p>
Lorem ipsum ad his <b>scripta </b> blandit
partiendo.
</p>
</div>
</div>
<div class="aviso">
<p>
Lorem ipsum  ad his <a>scripta </a>blandit
partiendo
<p>
<p>
Lorem ipsum  ad his scripta blandit partiendo
<p>
</div>
```

14.- Utilizando el selector adyacente construir las siguientes situaciones, el resultado esperado se muestra en la figura 2.35. Es recomendable revisar la sección 2.3.2.3 Selector adyacente

- Una página Web que maneja mucho contenido textual distribuido por elementos p, aplicarles a todos los elementos p excepto al primero una sangría de 10px.
- En una estructura de galería se requiere que el párrafo inmediato a una imagen tenga un color de letra negro, con tamaño de 14px, una sangría de 10px, un estilo oblicuo, un color de fondo verde opaco y un peso de letra de 600, pero los demás párrafos deben conservar un estilo previamente especificado.

15.- Utilizando los selectores por atributo, construir la siguiente situación. Dada una galería de imagénes en la cual exhibe productos, fotografías y rutas. Se requiere aplicar el siguiente estilo a las fotografías; un borde color azul de 3px con relleno de 2px, a los productos un borde color verde de 3px con relleno de 2px y a las rutas un borde color violeta de 2px con relleno de 2px. A las fotografías y a las rutas un color de fondo de #FFE0A5 y a los productos un fondo #CCC. Es recomendable revisar la sección 2.3.2.3 Selector de atributos

16.- Dadas las siguientes reglas aplicadas a un mismo elemento HTML explicar que regla prevalece, es decir cuál es el estilo final. Para resolver el ejercicio se recomienda leer la sección 2.3.5.3 Colisiones de estilo

Lorem ipsum ad his ***scripta*** *blandit partiendo,* ***eum fastidii*** *accumsan euripidis in, eum liber hendrerit an. Qui ut wisi vocibus suscipiantur,quo dicit ridens inciderint id. Quo mundi lobortis reformidans eu, legimus senserit definiebas an eos. Eu sit tincidunt incorrupte definitionem, vis mutat affert percipit cu, eirmod consectetuer signiferumque eu per. In usu latine equidem dolores. Quo no falli viris intellegam, ut fugit veritus placerat per.*

Lorem ipsum ad his **scripta** *blandit partiendo,* **eum fastidii** accumsan euripidis in, eum liber hendrerit an. Qui ut wisi vocibus suscipiantur,quo dicit ridens inciderint id. Quo mundi lobortis reformidans eu, legimus senserit definiebas an eos. Eu sit tincidunt incorrupte definitionem, vis mutat affert percipit cu, eirmod consectetuer signiferumque eu per. In usu latine equidem dolores. Quo no falli viris intellegam, ut fugit veritus placerat per.

Figura 2.35: Ejemplo de imagen con texto [Imagen creada por el autor de este trabajo (2011)]

```
/*--- Reglas de  CSS ---*/
p {color:#00f;font-family:Verdana;
font-size:15px;}
p#mensaje{color:#0f0;font-family:Georgia;
font-size:14px;}
* {color:#ff0;font-family:Georgia;
font-size:16px;}

<!-- Código HTML -->
<p id="mensaje">Lorem ipsum ad his scripta
blandit.</p>
```

2.3.3. Modelo de caja

El modelo de caja (**box model**) es uno de los conceptos más importantes y posiblemente el más complicado de explicar y entender. Su importancia radica en que durante el diseño con su uso se distribuyen todos los elementos de la página, es decir un mal uso del box model se ve reflejado en la distribución final de la página.

En el box model todos los elementos HTML de la página se representan como cajas y con reglas CSS es posible manipular estas cajas para definirles; color de letra, color de fondo, imágenes, ancho, alto, espacio de relleno, separación entre elementos (margen), visualización, perspectiva en z, fluidez y definir un cierto comportamiento a los elementos adyacentes, etc.

Como todos los elementos HTML de los documentos son representados como cajas es posible "maquetar" la vista de cada pantalla que conforman un sistema Web. Para esto es necesario comprender que lo que maquetamos en

el papel no será posiblemente trasladado en un 100 % a CSS, lo importante es mandener la idea clara del diseño que se desea implementar y de las propiedades CSS que podemos y necesitamos implementar para cumplir con el diseño.

Elementos que conforman el modelo de caja de un elemento HTML.

1.- Contenido.- Se trata del contenido en forma de texto, imagen, ambos u otro elemento html.

2.- padding.- Se trata del relleno, el cual es una separación de la caja con su contenido, como se muestra en la figura 2.36.

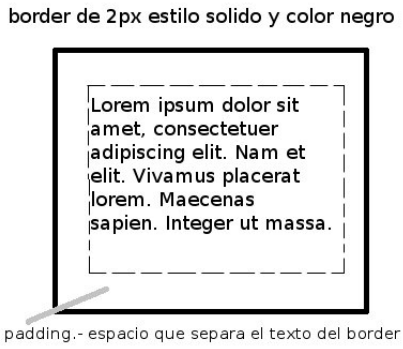


Figura 2.36: Border y padding [Imagen creada por el autor de este trabajo (2011)]

3.- border.- Es la línea delimitadora del contenido con los demás elementos del documento. Esta línea es la referencia que toma el padding y es utilizada para cerrar el contenido.

4.- background.- Esta propiedad permite colocar un color de fondo y/o una imagen. Esta propiedad abarca el espacio del contenido y del padding.

5.- margin.- Esta propiedad es la delimitadora de las cajas y sus adyacentes.

La figura 2.37 es un esquema del modelo de caja (Box model).

En el diagrama 2.37 se puede notar que el ancho de la caja está dado por la suma de los márgenes, bordes y rellenos izquierdos y derechos además del ancho del contenido. La altura está dada por la suma de los márgenes, bordes y rellenos superiores e inferiores, y la altura del contenido.

El padding y el margin no presentan propiedad de color ya que son transparentes, es decir si el elemento define un color de fondo (background-color) es posible ver el color de fondo. Si un elemento define color de fondo e imagen de fondo, la imagen tiene mayor importancia en la visualización, pero ambas

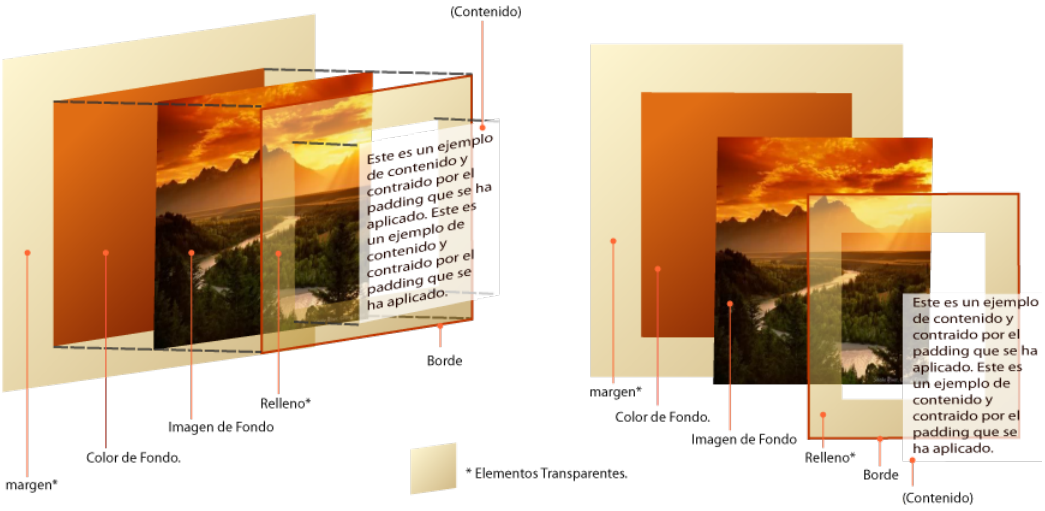


Figura 2.37: Modelo de caja [Imagen creada por el autor de este trabajo (2011)]

propiedades son aplicadas, esto se puede verificar si se coloca una imagen que no abarque todo el espacio del contenedor.

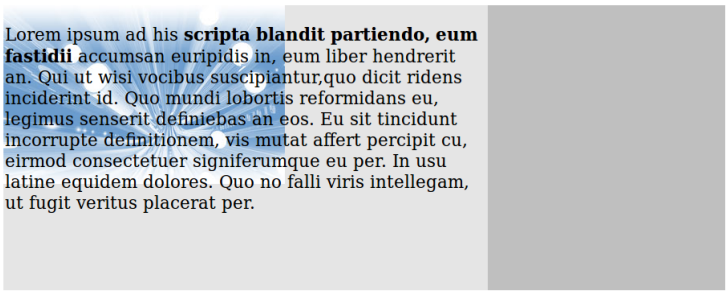


Figura 2.38: Margin y padding [Imagen creada por el autor de este trabajo (2011)]

En la figura 2.38 se muestra una caja contenedora con background-color #BFBFBF y dentro de ella una caja con un ancho menor y con la misma altura, a la que se le aplica un background-color y un background-image, es notorio que se aplican las dos propiedades ya que la imagen no abarca el tamaño de la caja de su contenedor y es visible el color (#E5E5E5). La ruta para ver este ejemplo es: "complementos/box_model/ejemplos/background"

2.3.3.1. Anchura y altura

Anchura

Es la propiedad que controla la anchura de los elementos.

Anchura	**width**
Posibles valores	medidas, procentajes, auto, inherit **medidas relativas**: div{width:900px;} **medidas absolutas**: span{width:6cm;} **porcentajes**: p{width:90 %;} **auto**: #content{width:auto;}calcula el valor según el espacio disponible. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos en bloque y solo a los siguientes elementos en línea; imágenes, filas y grupo de filas de una tabla.	

Tabla 2.1: Anchura en CSS

Width no acepta unidades negativas y los valores en porcentajes se calculan a partir del ancho del elemento padre.

Cuando no se establece un valor de forma explícita el valor es "auto", esto indica que el navegador debe calcular automáticamente la anchura del elemento, tomando en cuenta su contenido y el espacio disponible.

Altura

Es la propiedad que controla la altura de los elementos.

Altura	**height**
Posibles valores	medidas, procentajes, auto, inherit **medidas relativas**: div{height:400px;} **medidas absolutas**: p{height:3cm;} **porcentajes**: p{height:45 %;} **auto**: #content{height:auto;}calcula el valor según el espacio disponible. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos en bloque y solo a los siguientes elementos en línea; imágenes, columnas y grupo de columnas de una tabla.	

Tabla 2.2: Altura en CSS

Height no acepta unidades negativas y los valores en porcentaje se calculan a partir de la altura del elemento padre.

Cuando no se establece un valor de forma explícita el valor es "auto", esto indica que el navegador calcula la altura automáticamente tomando en cuenta su contenido y el espacio disponible.

En el código 2.25 se establece un ancho de 897px y una altura de 84px al elemento footer de una página Web, este elemento es muy común en los actuales diseños y forma parte del layout.

Código 2.25: Ejemplo de width y height

```
/*---------Reglas de CSS-----------*/
#footer{width:897px;height:84px;}
<!------- Código HTML ----------->
<div id="footer">
 <ul id="menu_bottom">
   <li><a href="">Bienvenidos</a></li>
   [...]
   <li><a href="">Contacto</a></li>
 </ul>
<div>
```

min/max-width/height

En el diseño de páginas Web se suelen utilizar layouts con una anchura/altura fija o con anchura/altura adaptable a la ventana. La mayoría de las veces es conveniente diseñar layouts con ciertas libertades pero con límites. Este tipo de diseño permite que el texto quede distribuido correctamente en su contenedor y así se pueda leer cómodamente.

Anchura mínima	**min-width**
Posibles valores	medidas, procentajes, inherit **medidas relativas**: div{min-width:650px;} **medidas absolutas**: span{min-width:2cm;} **porcentajes**: p{min-width:40 %;} **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos salvo filas y grupo de filas de una tabla	

Tabla 2.3: Anchura mínima en CSS

Anchura máxima	**max-width**
Posibles valores	medidas, procentajes, none, inherit **medidas relativas**: div{max-width:800px;} **medidas absolutas**: span{max-width:5cm;} **porcentajes**: p{max-width:90 %;} **none**: span{max-width:none;} no define un anchura máxima, este valor se puede omitir debido a que el valor inicial es none. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos salvo filas y grupo de filas de una tabla	

Tabla 2.4: Anchura máxima en CSS

Altura mínima	**min-height**
Posibles valores	medidas, procentajes, inherit **medidas relativas**: div{min-height:300px;} **medidas absolutas**: p{min-height:2cm;} **porcentajes**: p{min-height:25 %;} **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos salvo columnas y grupo de columnas de una tabla	

Tabla 2.5: Altura mínima en CSS

Altura máxima	**max-height**
Posibles valores	medidas, procentajes, none, inherit **medidas relativas**: div{max-height:800px;} **medidas absolutas**: p{max-height:6cm;} **porcentajes**: p{max-height:55 %;} **none**: span{max-height:none;} no define una altura máxima, este valor se puede omitir debido a que el valor inicial es none. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos salvo columnas y grupo de columnas de una tabla	

Tabla 2.6: Altura máxima en CSS

2.3.3.2. Margen y relleno

margin

El margin (margen) crea una separación transparente (no tiene color, por lo que permite ver el color de su elemento contenedor[3]) entre los elementos adyacentes.

Como todos los elementos se visualizan como cajas, cada una tiene cuatro lados y a cada lado se le asigna un margen, es decir; margen superior, margen derecho, margen inferior y margen izquierdo. Estas propiedades se pueden controlar de forma individual, mediante reglas separadas o utilizando la regla general margin. La figura 2.39 es un esquema sobre los márgenes de un párrafo.

Los márgenes verticales (margin-top y margin-bottom) solo se pueden aplicar a los elementos de bloque y a las imágenes, mientras que los márgenes laterales (margin-left y margin-right) se pueden aplicar a cualquier elemento.

Es posible especificar todos los márgenes en una sola propiedad, para ello se utiliza la regla resumida margin.

[3]Como se visualiza en el esquema 2.38

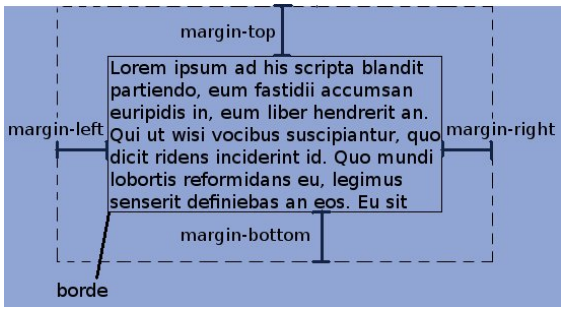


Figura 2.39: Márgenes [Imagen creada por el autor de este trabajo (2011)]

Margen superior	**margin-top**
Margen derecho	**margin-right**
Margen inferior	**margin-bottom**
Margen izquierdo	**margin-left**
Posibles valores	medidas, procentajes, auto, inherit **medidas relativas**: div{margin-top:15px;} medida con precisión **medidas absolutas**: span{margin-right:1cm;} **porcentajes**: p{margin-bottom:10 %;} para un diseño fluído **auto**: #aviso{margin:5px auto;}calcula el valor según el espacio disponible. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos, pero solo a los elementos en bloque y a las imágenes se les puede aplicar margin-top y margin-bottom.	

Tabla 2.7: Reglas individuales del margin

Margen	**margin**
Posibles valores	(medidas, procentajes, auto){1,4}, inherit **medidas relativas**: div{margin:1.5em;} para un diseño proporcional **porcentajes**: p{margin:10 % 15 %;} **auto**: #aviso{margin:5px auto;}calcula el valor según el espacio disponible. **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos	

Tabla 2.8: Regla resumida margin

La notación {1,4} indica que se admiten de 1 a 4 valores para esta propiedad. Si hay solo un valor, se aplica a todos los lados. Si hay dos valores, los márgenes superior e inferior son determinados por el primer valor y los márgenes derecho e izquierdo son determinados por el segundo. Si hay

tres valores, el superior es definido por el primer valor, el izquierdo y el derecho son definidos por el segundo, y el inferior es definido por el tercero. Si hay cuatro valores, ellos se aplican al superior, derecho, inferior e izquierdo, respectivamente.

En el primer bloque del código 2.26 se aplica a los elementos "div" un margen superior de 10px, inferior de 15px y a los laterales 20px. En el segundo bloque se utiliza la propiedad resumida "margin" con los valores 10px, 20px y 15px para los márgenes superior, laterales e inferiores respectivamente. Es decir, con una sola regla se aplican los 4 márgenes de un elemento.

Código 2.26: Regla resumida margin

```
/* Primer bloque  */
#container div{
  margin-top:10px;
  margin-bottom:15px;
  margin-left:20px;
  margin-right:20px;
}
/* Segundo bloque */
#container div{margin:10px 20px 15px;}
```

Márgenes cerrados

Los márgenes cerrados significan que cuando dos o más márgenes de elementos adyacentes o anidados, a los cuales ningún borde ni relleno los separa, se combinan para formar un solo margen. Los márgenes horizontales nunca se cierran.

Los márgenes verticales se pueden cerrar si:

- Dos o más márgenes verticales de cajas de bloques en el flujo normal se cierran. El tamaño del margen resultante es el máximo de los márgenes adyacentes. En el caso de márgenes negativos, el máximo de los negativos es restado al máximo de los positivos. Si no hay positivos, el máximo negativo es restado de cero.

- Los márgenes verticales entre un elemento flotante y cualquier otro elemento no se cierran.

- Los márgenes de elementos con posición absoluta o relativa no se cierran.

La siguiente situación ejemplifica el concepto de márgenes cerrados. Dado un elemento "div" que contiene cuatro párrafos, los párrafos reciben un margen de 10px en vertical y están centrados de forma horizontal. Si al segundo párrafo se le aplica la clase especial, la cual le asigna un margen de 20px en

vertical, se puede observar en la figura 2.40 que los elementos adyacentes al segundo párrafo presentan un margen vertical mayor que los adyacentes que no presentan esa clase (párrafos tercero y cuarto).

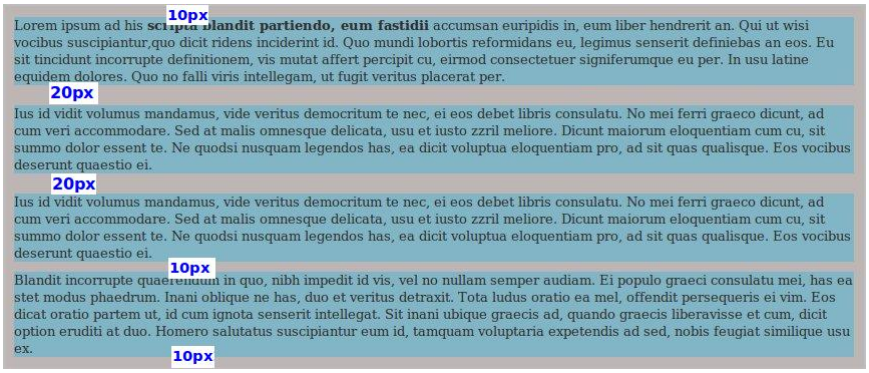


Figura 2.40: Márgenes cerrados [Imagen creada por el autor de este trabajo (2011)]

En la situación ejemplificada en la figura 2.41 un elemento "div" con id "seccion2", recibe un margen de 10px en vertical y centrado horizontalmente, este contiene un elemento "div" con id "contenido", al cual se le aplica un margen de 5px en vertical, que a su vez contiene párrafos a los cuales se les aplica un margen vertical de 20px. El margen aplicado a la "seccion2" es la separación con las cajas adyacentes a su mismo nivel de definición, asi que centrandos en el "div" contenido y en sus párrafos, como el elemento contenido no presenta borde ni relleno, sus márgenes se cierran con los márgenes de los párrafos, por lo tanto el margen final es el máximo entre el definido al contenido y el definido a los párrafos, en este caso el valor de los párrafos es de 20px que es superior a los 5px del contenido, dando como resultado una separación de 20px.

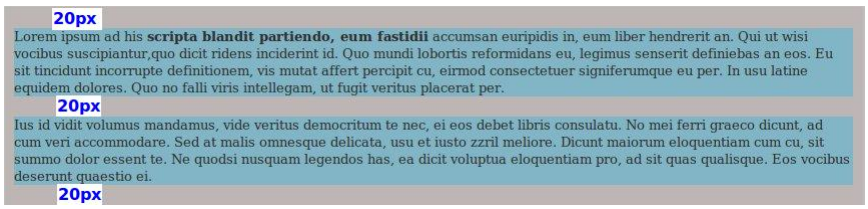


Figura 2.41: Márgenes cerrados [Imagen creada por el autor de este trabajo (2011)]

padding

La propiedad padding define el espacio entre el contenido y el borde. Esta propiedad es transparente por lo que permite ver el color de fondo de su

elemento contenedor. En la figura 2.42 se muestra un esquema del padding en un párrafo.

Al igual que con el margen se puede establecer el relleno de cada lado de forma individual.

Relleno superior	**padding-top**
Relleno derecho	**padding-right**
Relleno inferior	**padding-bottom**
Relleno izquierdo	**padding-left**
Posibles valores	medidas, procentajes, inherit **medidas relativas**: div{padding-top:10px;} **medidas absolutas**: span{padding-right:1.5cm;} **porcentajes**: p{padding-bottom:13 %;} **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos, excepto a los elementos de tablas como grupos de cabeceras y pies de tabla	

Tabla 2.9: Reglas individuales del padding

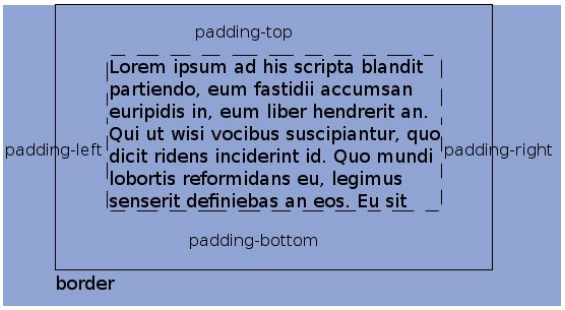


Figura 2.42: padding [Imagen creada por el autor de este trabajo (2011)]

En el código 2.27 se aplica un relleno vertical de 5px y un relleno lateral de 10px.

Código 2.27: Ejemplo de padding

```
#container div p{
  padding-top:5px;
  padding-bottom:5px;
  padding-left:10px;
  padding-right:10px;
}
```

Al igual que con margin, existe una regla resumida que permite establecer el relleno de cada lado en una sola definición.

Relleno	**padding**
Posibles valores	(medidas, procentajes){1,4}, inherit **medidas relativas**: div{padding:1em;} **porcentajes**: p{padding:5 % 10 %;} **inherit**: toma el valor heredado de su elemento contenedor.
Se aplica a todos los elementos, excepto a los grupos de cabeceras y pies de tablas.	

Tabla 2.10: Regla resumida padding

La notación {1,4} indica que padding admite de uno a cuatro valores. Si hay solo un valor, se aplica a todos los lados. Si hay dos valores, los rellenos verticales son determinados por el primer valor y los rellenos laterales son determinados por el segundo. Si hay tres valores, el superior es definido por el primer valor, los laterales son definidos por el segundo, y el inferior es definido por el tercero. Si hay cuatro valores, ellos se aplican al superior, derecho, inferior e izquierdo, respectivamente.

En el código 2.28 la primera regla le aplica a los párrafos contenidos en los "div", que a su vez están contenidos en "container" un padding vertical de 5px y un padding lateral de 10px, en la segunda regla a los "span" les aplica un padding de 3px a cada lado y en la última regla a los enlaces les aplica un padding superior de 5px, un inferior de 10px y lateral de 8px.

Código 2.28: Ejemplo de la regla padding

```
1 #container div p{padding:5px 10px;}
2 #container div span{padding:3px;}
3 #container div a{padding:5px 8px 10px;}
```

2.3.3.3. border y background

border

El borde o marco es el delimitador del contenido de una elemento con su margen, si no se cuenta con uno margen, es la separación con el resto de las demás cajas. En CSS se definen reglas que permiten establecer el estilo, el color, el ancho y por supuesto como es costumbre una regla resumida para el borde de los elementos.

border-width

El width indica el espesor del borde, en CSS se pueden especificar de forma individual o utilizando una regla resumida.

Espesor superior	**border-top-width**
Espesor derecho	**border-right-width**
Espesor inferior	**border-bottom-width**
Espesor izquierdo	**border-left-width**
Posibles valores	(medida, thin, medium, thick) **medida absoluta**: div{border-top-width:3px;} **medida relativa**: span{border-right-width:0.5em;} **thin**: p{border-bottom-width:thin;}
Valor de inicio	medium (el espesor depende del navegador).
Se aplica a todos los elementos	

Tabla 2.11: Reglas individuales para el espesor del borde

Establecer el espesor de un borde utilizando las palabras claves no es muy recomendable, esto se debe a que el navegador Internet Explorer define medidas diferentes a las propuestas por la mayoría de los navegadores. IE le asigna a la palabra clave medium un valor de 4px y el resto de los navegadores le asigna 3px, esto genera conflictos en el modelo de caja al momento de maquetar la página. Por esta razón es recomendable aplicar un espesor en px.

La regla resumida para aplicar el espesor al borde es border-width:

Espesor del borde	**border-width**
Posibles valores	(medidas, thin, medium, thick){1,4} **medidas relativas**: div{border-width:0.5em 1.5em;} **medidas absolutas**: p{border-width:1px 5px 2px;} **palabra clave**: #aviso{border-width:thin medium thick;}
Valor de inicio	medium (el espesor depende del navegador).
Se aplica a todos los elementos	

Tabla 2.12: Regla resumida para el espesor del borde

La notación {1,4} indica que se admiten de 1 a 4 valores para esta propiedad. Si hay solo un valor, se aplica a todos los lados. Si hay dos valores, el espesor superior e inferior son determinados por el primer valor y el espesor derecho e izquierdo son determinados por el segundo. Si hay tres valores, el superior es definido por el primer valor, el izquierdo y el derecho son definidos por el segundo, y el inferior es definido por el tercero. Si hay cuatro valores, ellos se aplican al superior, derecho, inferior e izquierdo, respectivamente.

Para que las reglas relacionadas al espesor del borde tengan efecto es necesario establecer previamente un estilo al borde.

border-style

Las siguientes reglas permiten establecer los estilos de los bordes. El estilo de los bordes solo se pueden indicar a través de palabras claves. La figura 2.43 es un esquema de los tipos de bordes.

Estilo superior	**border-top-style**
Estilo derecho	**border-right-style**
Estilo inferior	**border-bottom-style**
Estilo izquierdo	**border-left-style**
Posibles valores	none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, inherit **none**: div{border-left-style:none;} no aplica ningún borde. **hidden**: span{border-right-style:hidden;} similar a none. **double**: p{border-top-style:double;} aplica una doble línea continua. **solid**: p{border-bottom-style:solid;} aplica una línea continua.
Valor de inicio	none, no aplica ningún borde.
Se aplica a todos los elementos.	

Tabla 2.13: Reglas individuales para el estilo del borde

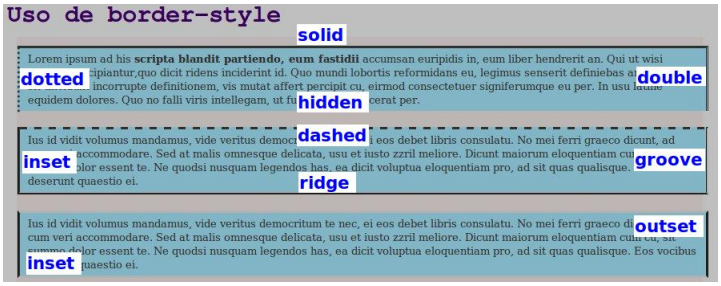


Figura 2.43: Estilos de borde [Imagen creada por el autor de este trabajo (2011)]

En el código 2.29 se aplica un estilo de borde a cada lado de la caja.

Código 2.29: Estilos de borde

```
#container div p{
  border-top-style:solid;
  border-right-style:double;
  border-bottom-style:hidden;
  border-left-style:dotted;
}
```

Si se requiere establecer el mismo estilo de borde o todos los estilos de borde en una sola definición se puede utilizar la regla resumida border-style:

Estilo del borde	**border-style**
Posibles valores	(none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset, inherit){1,4}, medida. **solid**: div{border-style:solid;} **double**: p{border-style:double;} **hidden**: #aviso{border-style:hidden;}
Se aplica a todos los elementos	

Tabla 2.14: Regla resumida para el estilo del borde

Los bordes más utilizados son solid, double, dashed y dotted, y por-supuesto "hidden" y "none" son indispensables para los diseños actuales. La diferencia entre hidden y none radica en la forma en la que se resuelven los conflictos de los elementos de tabla.

La palabra clave "none" obliga al navegador a aplicar un espesor de borde de 0. Las propiedades "groove", "inset", "outset" y "ridge" dependen de la propiedad border-color.

border-color

Las siguientes reglas permiten establecer el color de cada borde de forma individual.

Color superior	**border-top-color**
Color derecho	**border-right-color**
Color inferior	**border-bottom-color**
Color izquierdo	**border-left-color**
Posibles valores	color, transparent, inherit **color**: div{border-left-color:#0000FF;} azul al borde izquierdo. **color**: span{border-right-color:#FFFF00;} amarillo al borde derecho. **transparent**: p{border-top-color:transparent;} no aplica un color.
Valor de inicio	none, no aplica ningún color.
Se aplica a todos los elementos.	

Tabla 2.15: Reglas individuales para el color del borde

Si se requiere definir un mismo color o definir todos los colores de los bordes en una sola definición se utiliza la regla resumida border-color:

Color del borde	**border-color**
Posibles valores	(color, transparente){1, 4}, inherit **color**: div{border-color:blue;} **transparent**: p{border-color:transparent;}
Se aplica a todos los elementos	

Tabla 2.16: Regla resumida para el color del borde

Reglas resumidas del Borde

CSS define reglas resumidas para establecer las propiedades de color, estilo y espesor de los bordes.

Borde superior	**border-top**
Borde derecho	**border-right**
Borde inferior	**border-bottom**
Borde izquierdo	**border-left**
Posibles valores	(medida || color || estilo) | inherit **ej1**: div{border-left:1px solid #0000FF;} establece un espesor de 1px, un estilo de borde sólido y un color azul al borde izquierdo. **ej2**: span{border-right:2px dashed #FFFF00;} establece un espesor de 2px, un estilo dashed y color amarillo al borde derecho.
Valor de inicio	none, no aplica ningún color.
Se aplica a todos los elementos	

Tabla 2.17: Reglas resumidas para cada borde

Para completar todas las reglas CSS relacionadas con los bordes, se encuentra la regla general de bordes, la cual establece todos los atributos.

Borde	**border**
Posibles valores	(medida || color || estilo) | inherit **ej1**: div{border:1px solid #0000FF;} establece un espesor de 1px, un estilo de borde sólido y un color azul a todos los bordes. **ej2**: span{border:2px dashed #FFFF00;} establece un espesor de 2px, un estilo dashed y color amarillo a todos los bordes.
Se aplica a todos los elementos	

Tabla 2.18: Regla resumida border

Como el valor por defecto de la propiedad border-style es none, si una propiedad resumida no establece explícitamente el estilo de un borde, el borde no se muestra.

```
#container div p{
  border-top:1px solid #00f;
  border-right:1px solid #00f;
  border-bottom:1px solid #00f;
  border-left:1px solid #00f;
}
#container div p{border:1px solid #00f;}
```

Las reglas anteriores son semejantes debido a que la primera regla aplica un espesor de 1px, con estilo sólido y color azul a cada borde de forma individual y la regunda regla aplica el mismo estilo a todos los bordes en una sola defínción.

Background

El fondo de un elemento es visible en su contenido y en el espacio del padding, siendo el borde su límite de visualización. El fondo puede ser un color sólido o una imagen (degradados de colores, imágenes con elementos, colores sólidos, etc.). Es importante saber que todos los elementos de un documento HTML son transparentes por defecto.

CSS define varias propiedades para establecer el background incluyendo reglas resumidas.

Color de fondo	**background-color**
Posibles valores	(color | transparent | inherit) **color**: div{background-color:#00F;} establece un color de fondo azul. **transparent**: span{background-color:transparent;} fondo transparente.
Valor de inicio	transparent.
Se aplica a todos los elementos	

Tabla 2.19: Regla para el color de fondo

CSS define una propiedad para establecer una imagen como fondo:

Imagen de fondo	**background-image**
Posibles valores	(url | none | inherit) **url**: div{background-image:url(ímages/fondo.jpg');} establece la imagen fondo.jpg.
Valor de inicio	none. (no establece imagen)
Se aplica a todos los elementos	

Tabla 2.20: Regla para colocar una imagen de fondo

Esta propiedad establece una imagen de fondo para el elemento. La im-

agen siempre se muestra delante del color, así que si se define un color de fondo previamente este solo se muestra cuando:

- La imagen no abarca todo el espacio del elemento.
- La imagen no cargo (esto produce un error).
- La imagen abarca todo el espacio pero el usuario la posiciona de tal forma que muestra el color de fondo.
- La imagen abarca todo el espacio pero dicha imagen presenta un canal alfa.

Es recomendable crear una carpeta para alojar las imágenes utilizadas en CSS, esto permite llevar un orden en la estructura y facilita el acceso y mantenimiento del código y recursos del sistema Web.

En CSS es posible crear imágenes relativamente pequeñas y cubrir todo el espacio de un elemento, ahorrando tiempo y espacio al momento de cargar una página Web. Es decir se puede controlar la repetición y orientación de una imagen de fondo.

Repetición imagen	**background-repeat**
Posibles valores	(repeat | repeat-x | repeat-y | no-repeat | inherit) **repeat-x**: div{background-repeat:repeat-x;} repite la imagen en la dirección del eje x. **repeat-y**: p{background-repeat: repeat-y;} repite la imagen en la dirección del eje y. **no-repeat**: div{background-repeat: no-repeat;} coloca la imagen en (0,0) pero no la repite sobre los ejes.
Valor de inicio	repeat. (repite la imagen en la dirección del "eje y" y del "eje x")
Se aplica a todos los elementos	

Tabla 2.21: Repetición de la imagen de fondo

La figura 2.44 muestra cuatro ejemplos de uso del background-repeat, la imagen central es utilizada para repetir en el eje x, eje y en ambos ejes y sin repetir. En los tres primeros cuadros la forma de repetir la imagen permite ver el color de fondo, que para el ejemplo es #BFBFBF.

Como se mencionó es posible establecer la posición inicial en la que se coloca la imagen en background y junto con la propiedad repeat se consigue dar el estilo final del fondo.

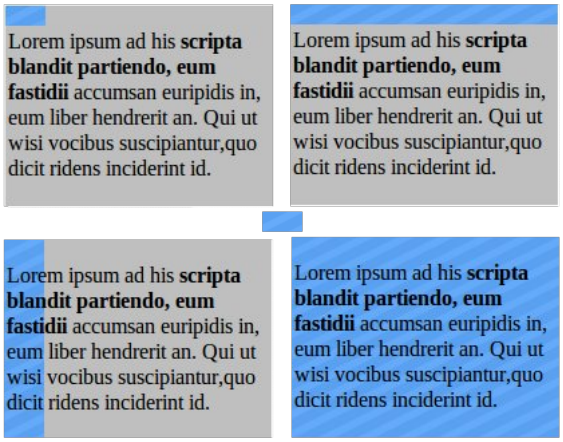


Figura 2.44: Imagen de fondo [Imagen creada por el autor de este trabajo (2011)]

Posición imagen	**background-position**
Posibles valores	(medida | porcentaje | left| center | right), (medida | porcentaje | top | center | bottom). div{background-position:right top;} en la parte superior derecha. p{background-position:center center;} en el centro. div{background-position:left bottom;} en la parte inferior izquierda. p{background-position:20 % 40 %;}
Valor de inicio	0 % 0 %
Se aplica a todos los elementos	

Tabla 2.22: Posición de la imagen de fondo

Cuando se utilizan porcentajes o medidas el primer valor indica el desplazamiento horizontal (de izquierda a derecha) y la segunda el desplazamiento vertical (de arriba a abajo) respecto al origen que se ubica en la parte superior izquierda. Cuando solo se indica una medida o un porcentaje, este indica el valor en horizontal y el valor en vertical se coloca automaticamente en 50 %.

La palabras claves tienen una medida porcentual.

- top == 0 % (segunda coordenada)
- left == 0 % (primera coordenada)
- center == 50 % (primera o segunda coordenada)
- bottom == 100 % (segunda coordenada)

- right == 100 % (primera coordenada)

Cuando se utilizan las palabras claves el orden no tiene importancia, ya que left o right hacen referencia a algo horizontal y top o bottom hacen referencia a algo vertical.

CSS cuenta con una regla resumida para establecer el color de fondo, url de la imagen, posición y repetición de una imagen en el background. El orden en el que se establecen las propiedades es indiferente, pero el estándar que vamos a seguir es; color, url de la imagen (si hay imagen), repetición (por defecto es repeat) y posición de la imagen (por defecto es 0 % 0 %).

Fondo	**background**
Posibles valores	(background-color || background-image || background-repeat || background-position | inherit) div{background:#00f url(ímages/fondo.png') repeat left top;} p{background: url(ímages/fondo.png') repeat-x center;} div{background: #0f0;} establece únicamente el color. p{background:#ff0 url(ímages/fondo.png') repeat-y 20 %;}
Valor de inicio	none
Se aplica a todos los elementos	

Tabla 2.23: Regla resumida background

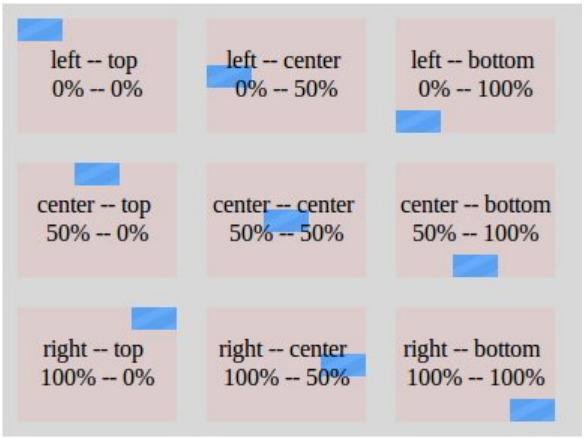


Figura 2.45: Background [Imagen creada por el autor de este trabajo (2011)]

2.3.3.4. Práctica 03

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.3.3 Modelo de caja, en cuyas secciones se trata el concepto de box model el cual es el más complejo de explicar. Se abordan las propiedades para establecer la anchura y altura de los elementos, la separación de los elementos con sus adyacentes, la separación interna del contenido con el borde de los elementos, entre otras propiedades relacionadas con el modelo de cajas de CSS.

Ejercicios Los documentos HTML de esta práctica se localizan en "complementos/box_model/ejercicios".

1.- Construye las reglas de estilo que se solicitan para sus respectivas construcciones HTML. Para resolver estos problemas se recomienda leer la sección de medidas 2.3.5.4

1.1.- Dado el código 2.30 definir un tamaño de letra de 15pt al elemento cuyo id sea "bloque", a sus titulares principales definirles el doble, a los párrafos el 90 % y a los elementos cuya clase sea "aviso" definirles un tamaño del 80 % del elemento contenedor. Si los párrafos contenidos en "bloque" tienen un ancho de 650px y esto es el 90 % del ancho definido a su elemento contenedor, cuál es su ancho original. Cómo quedan dichas reglas en CSS.

Código 2.30: Ejercicio 1.1

```
<div id="bloque">
<h1>Lorem impsum<\h1>
<p>
Ea mei nullam facete, omnis oratio offendit ius
cu. Doming takimata repudiandae usu an, mei
dicant taki mata id, pri eleifend inimicus
</p>
<p>
His vero singulis ea, quem euripidis abhorreant
mei ut, et populo iriure vix. Usu ludus affert
voluptaria ei, vix ea error definitiones movet
</p>
<span class="aviso">eos assum facilis corpora
</span>
<div>
```

1.2.- Construir el documento HTML y solo dos reglas CSS que cumplan con la siguiente situación. Dado tres párrafos anidados en estructuras "div" anidadas, contenidas en un "div" conocido como contenedor principal cuyo valor del atributo id es "contenido";

se les define un tamaño de letra de 14.4pt al primer párrafo, de 12.96pt al segundo párrafo y 11.664pt al último párrafo. El resultado esperado se muestra en la figura 2.46.

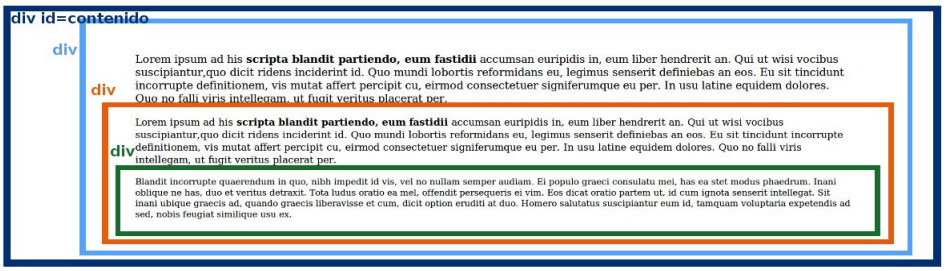


Figura 2.46: Divs anidados [Imagen creada por el autor de este trabajo (2011)]

1.3.- Construir un documento HTML el cual con solo dos reglas CSS debe cumplir con la siguiente situación. Un contenedor "div" con id "content" que contiene tres divs anidados entre sí, cuyas anchuras y alturas deben ser 630px y 280px respectivamente para el primero, 441px y 196px para el segundo y 308.7px y 137.2px para el último. El resultado se muestra en la figura 2.47

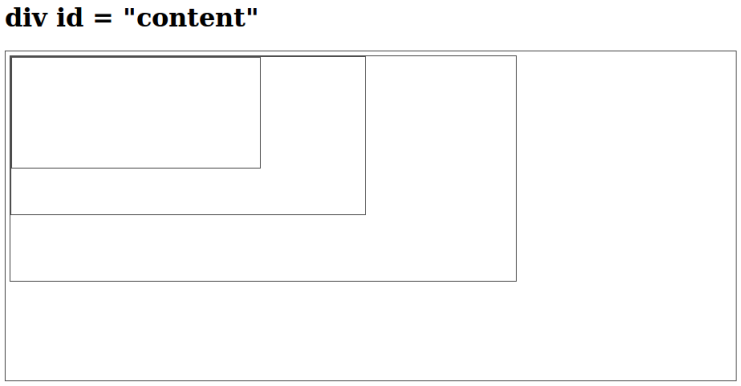


Figura 2.47: Medidas de divs aninados [Imagen creada por el autor de este trabajo (2011)]

1.4.- Construye las reglas de estilo indicadas en los párrafos del documento "medidas2/medidas2.html". El resultado esperado se muestra en la figura 2.48.

2.- ¿Cómo resolver el siguiente problema relacionado a medidas "Textos cada vez más grande" y "Textos cada vez más pequeños"?

Figura 2.48: Resultado de medidas2.html [Imagen creada por el autor de este trabajo (2011)]

3.- Construir las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "anchura/anchura.html". Es resultado esperado se muestra en la figura 2.49.

- 3.1.- #container.- un ancho de 900px, con un color de fondo #E5E5E5.
- 3.2.- h1.- aplicarles un ancho del 90 % de su elemento padre, un color de letra #38005B, una tipografía Courier y tamaño de letra de 2.8em.
- 3.3.- h2.- contenidos en #container aplicarles un ancho del 85 % y un color de fondo de #9494AD.
- 3.4.- p.- aplicarles un ancho del 98 % de su elemento padre, un color de fondo de #82B5C5, color de letra #303030 y tamaño de letra de 1.2em.
- 3.5.- div cuyos ids son seccionY con Y en {1, 2, 3}.- un ancho del 95 % de su elemento padre. ¿Cuánto equivale esto en píxeles? Además aplicarle un color de fondo de #C4C4C4.
- 3.6.- h2.- que se encuentran dentro de los divs inmediatos al "div" principal con valor de id container; definirles un ancho del 98 % del valor del padre. ¿Cuánto equivale esto en píxeles?
- 3.7.- p.- que se encuentran inmediatamente definidos a una imagen, además ambos están contenidos dentro de un "div" que sea hijo inmediato del "div" principal container; definirles un color de letra negro, un tamaño de letra de 1.4em, una sangría de 10px, un peso de letra de 600 y un color de fondo de color #D0E68E.

4.- Agregar las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "anchura/anchura.html".

- 4.1.- h2.- contenidos en #container aplicarles una altura mínima de 20px.

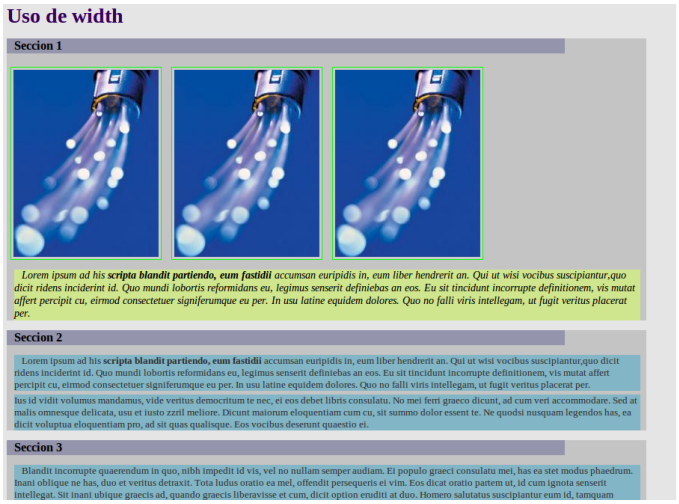


Figura 2.49: Resultado en anchura.html [Imagen creada por el autor de este trabajo (2011)]

4.2.- div cuyos ids son seccionY con Y en $\{1, 2, 3\}$.- una altura mínima de 150px.

4.3.- p.- aplicarles una altura mínima de 60px.

4.4.- p.- que se encuentran inmediatamente definidos a una imagen, además ambos están contenidos dentro de un div que sea hijo inmediato del div principal container; aplicarles una altura mínima de 50px, una altura máxima de 90px y un ancho de 70 %.

5.- Agregar las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "margin/margin.html". El resultado esperado se muestra en la figura 2.50.

5.1.- Poner el color de fondo del body en #E5E5E5.

5.2.- A los titulares principales un margen vertical de 0 y un lateral de 10px.

5.3.- Hacer que el container se muestre en el centro de la página.

5.4.- A los divs inmediatos al container aplicarles un margen vertical de 5px y un lateral de 10px.

5.5.- Los párrafos se tienen que mostrar en el centro con una separación vertical de 5px.

5.6.- Los párrafos definidos inmediatamente después de una imagen, aplicarles un margen de 10px.

5.7.- A las imágenes darles un margen vertical de 5px y un lateral de 10px.

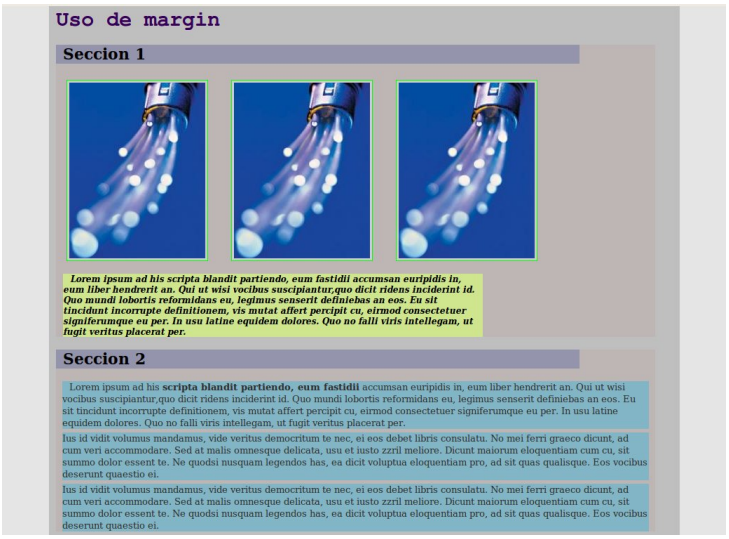


Figura 2.50: Resultado en margin.html [Imagen creada por el autor de este trabajo (2011)]

6.- Encontrar las reglas CSS de márgenes sobre el elemento “div” con id “ejemplo”, para las siguientes situaciones.

- 6.1.- Margen lateral de 0.5em y vertical de 1em.
- 6.2.- Todos los lados del “div” tenga una separación de 10px, pero la sepración inferior tiene que ser de 15px.
- 6.3.- Sin margen lateral y margen vertical de 10px.
- 6.4.- Un margen superior de 5px, el inferior y el izquierdo de 10px y el derecho de 15px.
- 6.5.- Un margen superior de 10px, el izquierdo y el derecho de 20px y el inferior de 15px;
- 6.6.- Centrado en la caja de su padre pero sin margen vertical.
- 6.7.- Centrado en la caja de su padre con una separación superior de 5px.
- 6.8.- Centrado en la caja de su padre con una separación interior y superior de 10px.

7.- Explica las siguientes propiedades CSS.

- 7.1.- margin:25px 30px;
- 7.2.- margin:25px 40px 75px;
- 7.3.- margin:30px;
- 7.4.- margin:40px; margin-bottom:30px;

7.5.- margin-left:35px; margin:30px;

7.6.- margin:10px 5px 10px 5px;

7.7.- margin:12px 7px; margin-top:20px;

8.- Agregar las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "margin-padding/margin-padding.html".

8.1.- Poner rellenos verticales de 0 y laterales de 4px a los titulares principales.

8.2.- Al "div" con id container quitarle el relleno.

8.3.- A los titulares secundarios contenidos en los divs definidos inmediatamente al container, aplicarles un relleno vertical de 3px y lateral de 10px.

8.4.- Los párrafos deben presentar en todos sus lados 3px de relleno.

8.5.- Los párrafos definidos inmediatamente después de una imagen, aplicarles un relleno de 5px.

8.6.- Las imágenes deben presentar rellenos de 5px en el superior, 4px en los laterales y 0 en el inferior.

9.- Explica las siguientes reglas CSS.

9.1.- padding:5px 15px;

9.2.- padding:5px 15px 7px;

9.3.- padding:30px 15px 0 20px;

9.4.- padding:15px; padding-left:25px;

9.5.- padding-top:15px; padding:25px;

9.6.- padding-left:12px; padding-top:10px; padding-bottom:12px; padding-right:10px;

10.- ¿Qué ocurre cuando una regla resumida relacionada con el borde no establece un estilo? ejemplifica.

11.- Establecer los siguientes estilo al documento "border/border.html". El resultado esperado se muestra en la figura 2.51.

11.1.- A los divs definidos inmediatamente al div con id "container" aplicarles un espesor de borde de 1px, a los bordes superiores un estilo sólido con un color #A6A3A3, a los derechos un estilo dotted con color #A16CC2, a los inferiores un estilo dotted con un color #A16CC2 y a los izquierdos un estilo sólido con un color #A6A3A3. Sólo se pueden utilizar 3 reglas para llegar a ese resultado.

11.2.- A los titulares secundarios definidos dentro de divs aplicarles un borde izquierdo de 3px de espesor, con estilo sólido y color #50506F.

11.3.- A los párrafos aplicarles un borde superior e inferior de 1px de espesor, con estilo dashed y color #669AAA. Utilizar únicamente 2 reglas.

11.4.- A los párrafos definidos inmedidamente a una imagen, aplicarles un color de borde #A1B46B. ¿Mantener el estilo aplicado a los párrafos?

11.5.- A las imágenes aplicarles un relleno de 5px y un borde de 1px de espesor, con estilo sólido y color #634078.

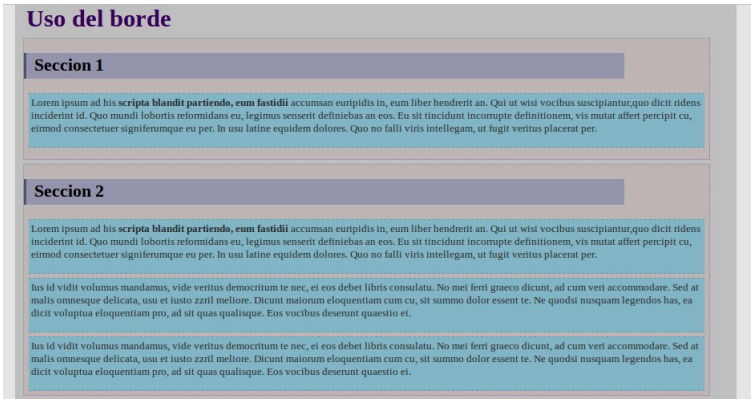


Figura 2.51: Resultado en border.html [Imagen creada por el autor de este trabajo (2011)]

12.- Resume las siguientes reglas sobre bordes.

12.1.-

```
#container div p{
    border-top:1px dashed #0f0;
    border-right:1px dashed #0f0;
    border-bottom:1px dashed #0f0;
    border-left:1px dashed #0f0;
}
```

12.2.-

```
#container div{
    border-top-style:dotted;
    border-bottom-style:dotted;
    border-right-style:dashed;
    border-left-style:dashed;
    border-top-width:2px;
    border-right-width:3px;
```

```
    border-bottom-width:3px;
    border-left-width:2px;
    border-color:blue red blue red;
}
```

12.3.-

```
#container div{
    border-top-style:dotted;
    border-bottom-style:double;
    border-right-style:dashed;
    border-left-style:dashed;
    border-top-width:2px;
    border-right-width:3px;
    border-bottom-width:4px;
    border-left-width:3px;
    border-top-color:blue;
    border-bottom-color:blue;
    border-right-color:red;
    border-left-color:red;
}
```

13.- ¿Cuál es la ventaja de establecer una imagen y un color de fondo a un elemento, en lugar de únicamente establecer una imagen?

14.- Construir las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "project/index.html" Se permite utilizar imágenes propias de diseño para aplicarlas en el ejemplo. Se recomienda la página stripegenerator para diseñarlas. Respecto a las imágenes de fondo cuando no se indica un valor referente a la propiedad repeat aplicar el valor por defecto, es decir, repeat (en el eje x y en el eje y). Las imágenes utilizadas en las hosjas de estilo CSS se deben alojar en la ruta: public/css/images. El resultado esperado se muestra en la figura 2.52

14.1.- Aplicar un color de fondo #EAEAEA y la imagen de fondo "fondo-body.jpeg" a toda la página (localizada en la carpeta images de css).

14.2.- Al "div" con id header aplicar un ancho de 100 %, una altura de 150px, el elemento se debe centrar en lo horizontal sin separación vertical y sin relleno.

14.3.- A los divs inmediatos a un "div" que a su vez sea inmediato al elemento con id header; aplicarles un ancho de 920px y se deben centrar horizontalmente y sin separación vertical.

14.4.- Al "div" con id sec-telefonos aplicarle una altura de 15px y un color de fondo #000.

14.5.- Al "div" con id sec-menu aplicar una altura de 90px y un color de fondo #272F18 y la imagen fondo_menu.png.

14.6.- A los elementos li del ul definido dentro de sec-telefonos aplicarles; un ancho de 90px, con un relleno vertical 0 y laterales de 3px, una separación superior de 6px, un borde izquierdo de 1px de espesor con color #D6D6D6 y estilo sólido.

14.7.- A los elementos "span" definidos dentro de los "li" (anteriormente definidos) aplicarles una separación de 2px.

14.8.- Al elemento con id container aplicarle una ancho de 920px y se debe centrar horizontalmente con una sepación vertical de 25px.

14.9.- A los titulares principales de la página aplicarles, una separación verticales de 5px y laterales de 15px, un tamaño de letra de 1.5em, un color #585858 y el texto debe estar en mayúsculas.

14.10.- A los párrafos de la página aplicarles un relleno de 5px, un color negro, un tamaño de letra de 1.1em, justificado y con una sangría de 10px.

14.11.- Al "div" con id content aplicarle un ancho de 100 % y una separación inferior de 30px.

14.12.- Al "div" con id general aplicarle un ancho de 613px y una altura de 400px.

14.13.- Al "div" con id texto aplicarle un ancho de 896px, una altura de 150px, una separación de 10px, un color de letra negro y un color de fondo blanco.

14.14.- A los divs con la clase "secciones" aplicarles un ancho de 260px, una altura de 390px, un relleno de 5px, una separación vertical de 10px y lateral de 15px, un borde con espesor de 1px de color #F5F5F5 y estilo sólido, un color de fondo de #FAFAFA y la imagen de fondo fondo-secciones.jpeg.

14.15.- A los divs inmediatos a los elementos con la clase "secciones" aplicarles una altura del 50 % y un border superior e inferior de 1px de espesor con estilo sólido y de color #FFF y #C6C6C6 respectivamente.

14.16.- A los titulares principales contenidos en los divs inmediatos a "secciones" aplicarles, una separación vertical de 5px y lateral de 15px, un relleno inferior de 3px y un color de letra #292929.

14.17.- A los párrafos contenidos en los divs inmediatos a "secciones" aplicarles una altura de 110px y un color de letra #C1C1C1.

14.18.- A todos los enlaces con valor de clase "ruta" contenidos en "secciones" aplicarles una separación de 5px, un relleno de 6px, un color de letra #FFF y un color de fondo #2D2D2D.

14.19.- Al "div" con id "texto" aplicarle una altura del 100 % y un color de fondo #FAFAFA y la imagen fondo-secciones.jpeg.

14.20.- A las imágenes contenidas en "texto" aplicarles una separación superior, derecha e inferior de 5px y sin separación izquierda, un relleno de de 6px y un borde derecho con espesor de 1px de color #C6C6C6 y estilo sólido.

14.21.- A los párrafos contenidos en "texto" aplicarles una anchura de 650px, una separación de 5px y un tamaño de texto de 1.2em.

14.22.- A los divs contenidos en "texto" aplicarles un borde izquierdo con espesor de 1px de color #FFF y estilo sólido, quitar el borde superior y el margen izquierdo.

14.23.- Al elemento con id "footer" aplicarles un ancho de 100 % con una altura mínima de 100px, un color de texto #FFF y un color de fondo de #3D3D3D y aplicar la imagen fondo-footer5.png.

14.24.- A los divs inmediatos al "footer" aplicarles un ancho de 920px y se deben centrar de forma horizontal sin separación horizontal.

14.25.- A los uls inmediatos al "footer" aplicarles un margen superior de 35px, sin relleno y sin estilo de lista.

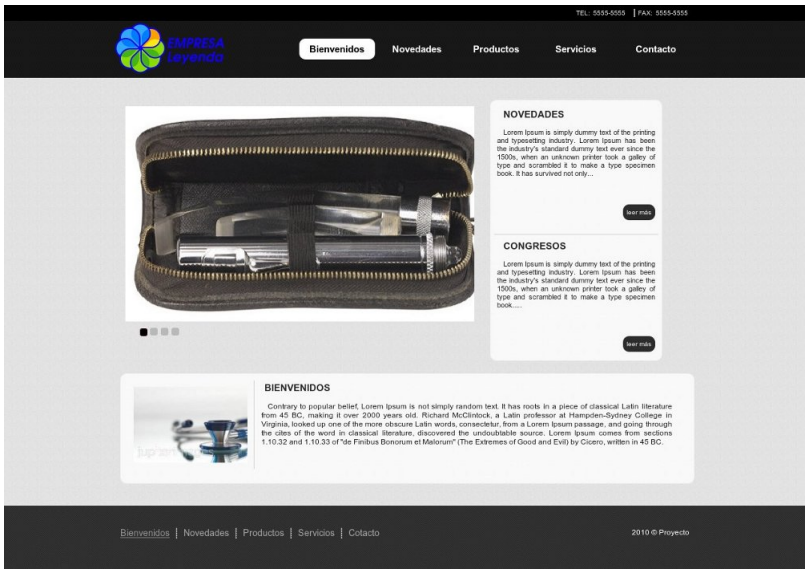


Figura 2.52: Resultado del Proyecto [Imagen creada por el autor de este trabajo (2011)]

2.3.4. Posicionamientos

2.3.4.1. Tipos de elementos

Cuando un usuario hace la petición al servidor para visualizar la página, este envía el código fuente de la página y es el navegador del usuario el encargado de interpretar y visualizar la página.

El navegador construye el modelo de cajas de la página, cada navegador interpreta este modelo de forma similiar pero algunas versiones de IE difieren en varios puntos críticos con los demás navegadores, esto tiene que ver con la forma en la que interpretan el width y el padding de los elementos.

Actualmente las compañías encargadas de los navegadores están cumpliendo en cubrir de la mejor forma la mayoría de los estándares de la W3C.

Cuando los navegadores construyen las cajas toman en cuenta los siguientes aspectos:

- Las anchuras y alturas de los elementos. Si no están establecidas son determindas por sus elementos contenedores y si se tratan de elementos en línea o en bloque.
- El tipo de elemento, es decir si se trata de un elemento en línea (ej span, a) o un elemento en bloque (ej. p, div).
- El posicionamiento de la caja.
- La relación con otros elementos, es decir, si se tratan de elementos anidados.
- El tamaño de las imágenes, el tamaño de la ventana del navegador, entre otras.

Elementos en bloque

Los elementos en bloque siempre comienzan en una nueva línea, es decir, si anterior a ellos se define cualquier elemento, comenzarán justo debajo (nueva línea) de dicho elemento, además ocupan todo el espacio que les permite su elemento contenedor.

Los elementos de bloque definidos por HTML son: address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul, dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.

En la figura 2.53 los párrafos abarcan todo el espacio que le permite el elemento contenedor y comienzan en una nueva línea.

Debido a sus características los elementos en bloque pueden contener cualquier cosa, es decir, tanto elementos en bloque como elementos en línea, pero los elementos en línea únicamente pueden contener otros elementos en línea.

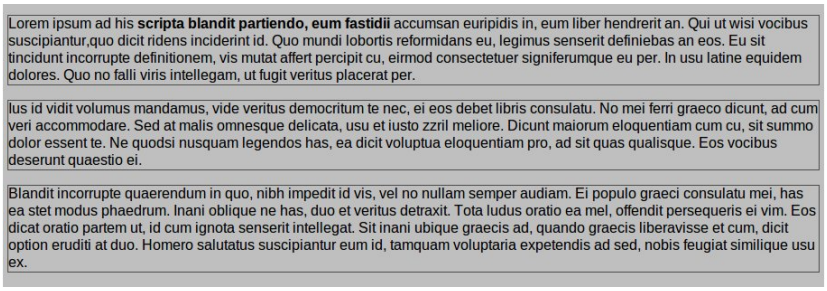


Figura 2.53: Elementos en bloque [Imagen creada por el autor de este trabajo (2011)]

Elementos en línea

Los elementos en línea no abarcan todo el espacio de su elemento contenedor, solo el espacio que necesita su contenido, es decir, tienen el ancho generado por su contenido, no inician en una nueva línea, esto es que entre elementos en línea su visualización es continúa, generan saltos de línea cuando su contenido es mayor al ancho del elemento contenedor.

Los elementos en línea definidos por HTML son: a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.

En la figura 2.54 se ve la diferencia entre párrafos y "span", los primeros al ser elementos en bloque abarcan todo el ancho de su elemento contenedor e inician en una nueva línea, los "span" al ser elementos en línea no inician en una nueva línea y tienen el ancho producido por su contenido. Se puede notar por el borde azul que los elementos en línea tienen un comportamiento continúo y cuando su contenido sobrepasa el tamaño del ancho del elemento contenedor generan un salto de línea.

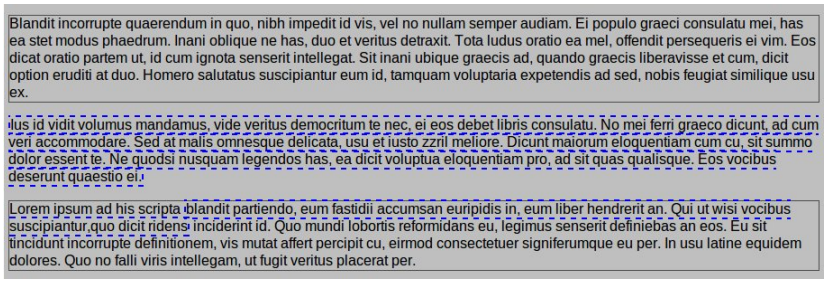


Figura 2.54: Elementos en bloque y en línea [Imagen creada por el autor de este trabajo (2011)]

2.3.4.2. Posicionamiento

Los navegadores crean y posicionan de forma automática todas la cajas de una página Web. Con reglas CSS es posible cambiar el comportamiento de cada caja, en el caso concreto, **nos permiten indicar la posición de una caja**.

El estándar de CSS define 5 posicionamientos:

- Estático o normal, es el utilizado por los navegadores a menos que se establezca lo contrario. En este posicionamiento los elementos se agrupan en forma de bloques uno debajo del otro.
- Relativo, es una variante del posicionamiento estático el elemento sigue en el flujo normal de la página pero se puede desplazar de su posición original.
- Absoluto, los elementos adyacentes ignoran su nueva posición y pasan a ocupar el espacio dejado por dicho elemento posicionado, se puede desplazar a partir de las coordenadas (0,0) del elemento contenedor más cercano posicionado (puede ser fijo, relativo, absoluto, pero para el estático no aplica).
- Fijo, es una variante del posicionamiento absoluto, también sale del fujo normal de la página y se puede desplazar a partir de las coordenadas (0,0) de la ventana del navegador. El elemento se convierte en una caja inamovible, siempre se mostrará en el mismo lugar.
- Flotante, no se establece con la propiedad position. La propiedad float desplaza a un elemento a la izquierda lo más posible o a la derecha según sea el caso. Los elementos salen del flujo normal de la página, es decir, los demás elementos desconocen su nueva posición y pasan a utilizar el espacio que deja.

Posición	**position**
Posibles valores	static, relative, absolute, fixed, inherit **relative**: div{position:relative;} Similar a static pero se puede desplazar. **absolute**: span{position:absolute;} Sale del fujo normal de la página y se puede desplazar. **fixed**: #aviso{position:fixed;} Similar a absolute pero siempre ocupa el mismo espacio en la ventana del navegador.
Valor de inicio	static. (el que aplican los navegadores)
Se aplica a todos los elementos.	

Tabla 2.24: Posición de un elemento

top **bottom** **left** **right**	**desplazamiento superior** **desplazamiento inferior** **desplazamiento desde la izquierda** **desplazamiento desde la derecha**
Posibles valores	medida, porcentaje, auto, inherit **medida**: div{top:25px; left:30px;} desplaza de arriba hacia abajo 25px. **porcentaje**: span{left:15 %;} desplaza de izquierda a derecha una distancia del 15 % del espacio disponible del ancho del elemento contenedor posicionado más cercano.
Valor de inicio	auto
Se aplica a todos los elementos.	

Tabla 2.25: Desplazamientos de los posicionados

Position static

Cuando los elementos en bloque no son posicionados, el navegador les aplica este comportamiento. Al no presentar un desplazamiento, debido a que en este posicionamiento las propiedades top, right, bottom y left están deshabilitadas únicamente se toman en cuenta si se tratan de elementos en bloque o en línea.

Referente a los elementos en bloque ocurre un comportamiento llamado "contexto de formato en bloque", el cual consiste en que todos los elementos se ven apilados, uno debajo del otro. Como se comentó anteriormente el ancho de estos elementos abarcan todo el espacio proporcionado por su elemento contenedor[4].

En la figura 2.55 se muestra que los párrafos contenidos en el div contenedor se encuentran apilados y abarcan todo el espacio horizontal disponible.

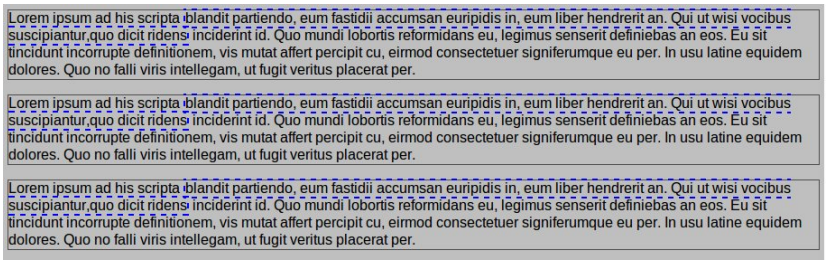


Figura 2.55: Contexto de formato en bloque [Imagen creada por el autor de este trabajo (2011)]

Respecto a los elementos en línea el comportamiento se conoce como "con-

[4]A los elementos en bloque cuando no se les define el ancho explícitamente, abarcan todo el espacio horizontal de su elemento contenedor

texto de formato en línea", en el cual los elementos fluyen horizontalmente. Cuando el contenido de un elemento termina inicia el siguiente, respetando los márgenes laterales, cuando el contenido de uno o de varios elementos en línea sobre pasan el ancho disponible por el contenedor generan un salto de línea.

En la figura 2.56 los "span" muestran un borde superior e inferior punteado de color azul, un borde izquierdo color naranja y en el borde derecho el color rojo para ejemplificar el flujo de los elementos, así es posible notar que el ancho de los elementos en línea es justo el generado por su contenido y que cuando este termina, inicia el otro elemento respetando los márgenes laterales.

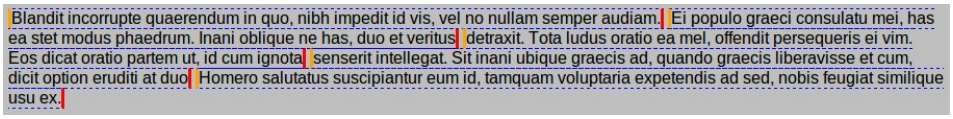


Figura 2.56: Contexto de formato en línea [Imagen creada por el autor de este trabajo (2011)]

Position relative

Este posicionamiento es una variante del posicionamiento static, el elemento permanece en el flujo normal de la página por lo que el resto de los elementos (elementos adyacentes) siguen conociendo su posición y respetan el espacio original que "usaba" el elemento posicionado.

Las propiedades de desplazamiento top, right, bottom y left toman como referencia las coordenadas de la posición inicial, es decir, la posición en static. Esto es muy conveniente debido a que si no se aplica algún desplazamiento, el elemento se comporta como elemento static, pero el navegador lo considera un elemento posicionado[5] sin salirse del flujo normal de la página y manteniendo el contexto de formato en bloque.

Esta propiedad es muy útil cuando los elementos son posicionados y se requiere desplazarlos hacia algún lugar de la página generando diseños innovadores, pero se debe considerar que el resto de los elementos seguirán respetando su espacio original.

A continuación se puntualizan algunos comportamientos referentes a las propiedades de desplazamiento.

- La propiedad **left** desplaza la caja de izquierda a derecha y la propiedad **right** la desplaza de derecha a izquierda. Estas propiedades son mutuamente excluyentes.

[5]Los elementos posicionados y anidados necesitan elementos contenedores posicionados, para generar un flujo que facilite el diseño de la página

  - Si ambas propiedades tiene el valor de **auto**, la caja no se desplaza.
  - Si solo left presenta el valor de auto el desplazamiento es de izquierda a derecha.
  - Si solo right presenta el valor de auto el desplazamiento es de derecha a izquierda.
  - Ocurre algo similar con las propiedades top y bottom.

- La propiedad **top** desplaza la caja de arriba hacia abajo y la propiedad **bottom** la desplaza de abajo hacia arriba.
- La propiedad -right, es el desplazamiento inverso de right, es decir, se comporta como left.
- La propiedad -left, es el desplazamiento inverso de left, es decir, se comporta como right.
- La propiedad -top se comporta como bottom.
- La propiedad -bottom se comporta como top.
- Si left y right tienen valores distintos a auto, una de las propiedades es anulada, eso depende del valor de dirección (direction). La mayoría de los países realizan sus documentos con la orientación de arriba hacia abajo y de izquierda a derecha (la forma de leer los documentos). Si la dirección es la descrita anteriormente y las propiedades de left y right tiene valores diferentes de auto, el valor de left predomina. Si la orientación es la opuesta right predomina y left es ignorada. Para top y bottom ocurre algo similar.

En la figura 2.57 (tomada del ejemplo "relative/relative.html") el primer párrafo está posicionado de forma relativa y se desplaza a partir de su posición original, el resto de los elementos respetan dicha posición y no invaden el espacio dejado por el elemento desplazado.

Position absolute

Se utiliza este posicionamiento para colocar un elemento en algún lugar de la página de una forma precisa, cuando el elemento se encuentra en su nueva posición el resto de los elementos (adyacentes) desconocen su posición original y pasan a ocupar el espacio dejado por dicho elemento, esto ocurre por que el elemento posicionado de forma absoluta sale del flujo normal de la página. Al desplazar un elemento posicionado es muy probable que ocurran solapamientos en los diseños inovadores es posible aprovechar este comportamiento.

Para desplazar un elemento posicionado en absolute se utilizan las propiedades de desplazamiento top, right, bottom y left, y a diferencia del posicionamiento relativo en el cual se toma como referencia la posición original

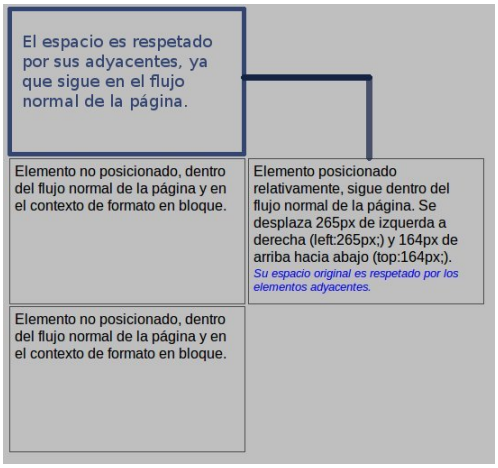


Figura 2.57: Position relative [Imagen creada por el autor de este trabajo (2011)]

(estático), en el absoluto la referencia depende del elemento contenedor más cercano posicionado. El posicionamiento del contenedor debe ser diferente de static, de no encontrar ningún elemento contenedor posicionado la referencia es el "body".

Para encontrar el elemento contenedor del cual se toma la referencia para desplazar un elemento posicionado de forma absoluta, el navegador localiza el elemento y verífica si presenta desplazamientos de ser el caso analiza todos los elementos contenedores (o padres) del elemento posicionado y el más cercano cuyo posicionamiento sea diferente a static se establece como el punto de referencia, de no encontrar ningún contenedor posicionado, se establece el body como punto de referencia.

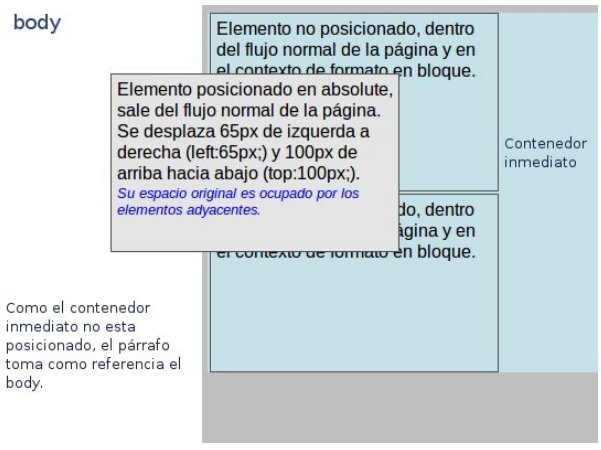


Figura 2.58: Position absolute sin contenedores posicionados [Imagen creada por el autor de este trabajo (2011)]

En la figura 2.58 el párrafo con id "absolute" ha sido posicionado de forma absoluta y se encuentra junto con otros dos párrafos contenidos en un "div" con id "contenedor_inmediato" que a su vez está contenido en el "div" container. Los divs container y contenedor no están posicionados, por lo que las propiedades de desplazamiento toman como referencia el body. En el ejemplo se puede notar que el párrafo sale del flujo normal de la página y el resto de los párrafos desconocen su posición y ocupan su lugar original, respecto a su movimiento se desplaza 65px de la parte superior del body hacia abajo y 100px de la parte izquierda del body hacia la derecha.

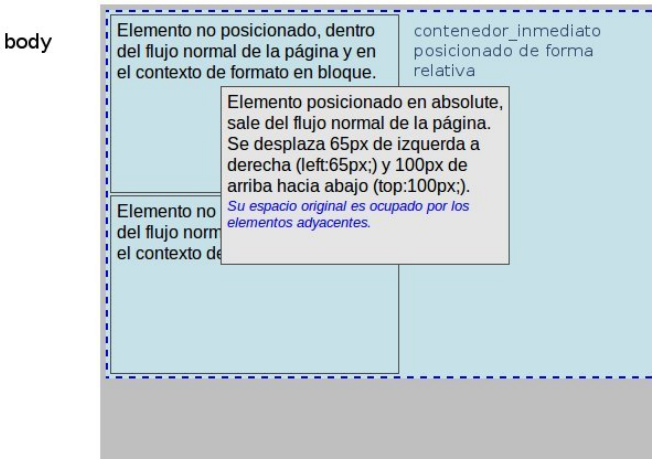


Figura 2.59: Position absolute con contenedores posicionados [Imagen creada por el autor de este trabajo (2011)]

En la figura 2.59 el párrafo "absolute" está posicionado de forma absoluta y el elemento "contenedor_inmediato" está posicionado de forma relativa por lo que el desplazamiento toma como referencia la parte superior izquierda del elemento posicionado relativamente.

Position fixed

Es una variante del posicionamiento absoluto la diferencia radica en que el elemento de referencia es el body y que al posicionar el elemento este siempre se mostrará en la misma zona del navegador aunque el usuario se desplace con el scroll de la ventana del navegador. Esto es que el elemento sale del flujo normal de la página y toma como referencia el body para desplazarse y se convierte en un elemento inamovible dentro de la ventana del navegador.

Cuando un elemento es posicionado de forma fixed y dicho estilo es aplicado a medios impresos en cada hoja que conforme el sitio web aparecerá dicho elemento en la posición especificada, esto es muy útil para mensajes, publicidad, logos y leyendas empresariales.

En la figura 2.60 el párrafo sale del flujo normal de la página y es posi-

cionado tomando como referencia el body.

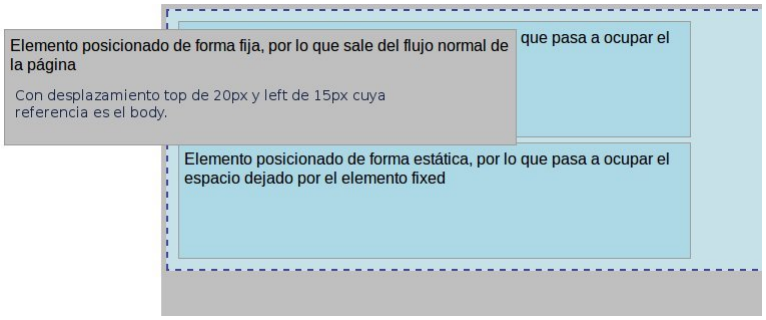


Figura 2.60: Position fixed [Imagen creada por el autor de este trabajo (2011)]

Float

Los elementos con posicionamiento float no están posicionados como tales, ya que las propiedades de desplazamiento top, right, bottom y left no influyen en sus desplazamientos, es mas son ignorados. Esta propiedad es una de las más complicadas del estándar CSS 2.1 y junto con el box model y los márgenes cerrados son las claves para generar diseños complejos.

Cuando un elemento es flotado sale del fujo normal de la página y se desplaza lo más a la izquierda o lo más a la derecha posible de su elemento contenedor, es decir, por definición los flotantes se desplazan por las laterales de su elemento contenedor y el contenido de los elementos adyacentes se reorganiza para fluir en su contorno.

Cuando un elemento presenta elementos flotados y no contienen elementos de flujo normal (es decir sin la propiedad float) los elementos flotados "salen" del contenedor, por lo que el contenedor pierde su altura. Si la altura fue establecida y es menor a la acumulada por los elementos flotados estos elementos se verán desbordados. Este comportamiento se debe a que el elemento contenedor desconoce a los elementos flotados y considera que se encuentra vacío, por lo tanto la altura desaparece ya que su contenido sale del flujo normal de la página y literalmente se encuentran en otro nivel, que no es reconocido por los elementos pertenecientes al flujo normal de la página.

La propiedad que se encarga de este comportamiento es float.

Flotante	**float**
Posibles valores	left, right, none, inherit **left**: div{float:left;} flotante la izquierda **right**: p{float:right;} flotante a la derecha
Valor de inicio	none
Se aplica a todos los elementos.	

Tabla 2.26: Flotar un elemento

En la figura 2.61 el párrafo se encuentra flotado a la izquierda y el resto de los párrafos adaptan su contenido para fluir a su alrededor. Sin importar sin son párrafos, listas o cualquier elemento en bloque, estos adaptan su contenido para fluir en el contorno de los elementos flotados y en el caso de los elementos en línea estos disminuyen su ancho para fluir alrededor de los elementos flotados.

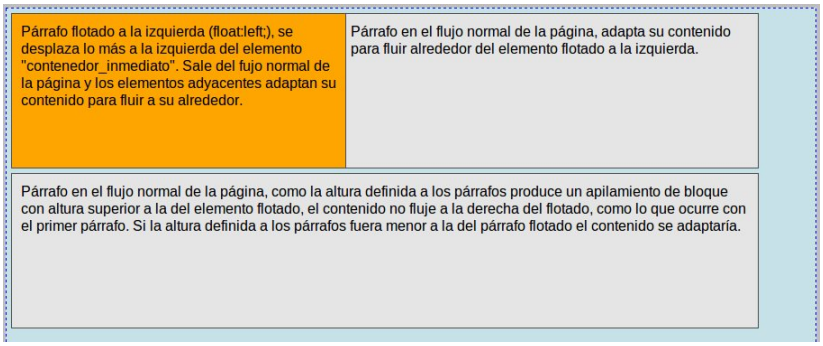


Figura 2.61: Párrafo flotado [Imagen creada por el autor de este trabajo (2011)]

Cuando varios elementos adyacentes están flotados estos se reconocen y se apilan unos después de otros mientras el ancho del elemento contenedor lo permite, de lo contrario generan un salto de línea y continúan con su comportamiento natural, se puede observar un ejemplo en la figura 2.62.

Los párrafos cercanos a un elemento flotado adaptan su contenido para fluir alrededor de dicho elemento, esto genera páginas Web muy parecidas a los documentos destinados a ser impresos (medios impresos), pero en ocasiones no se desea dicho comportamiento, para contrarrestar este comportamiento se utiliza la regla clear que obliga al elemento a no fluir entorno a los elementos flotados.

Cuando un elemento presenta clear left, se indica que ningún elemento flotante puede estar a su izquierda, de presentar un elemento flotante este se ubica justo debajo del elemento en cuestión. Cuando se presenta clear right, se indica que ningún elemento puede estar a su derecha y clear both indica que ningún elemento flotante puede estar en ambos lados.

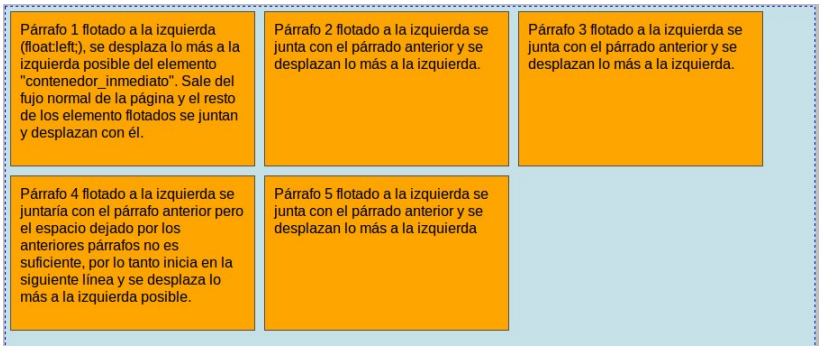


Figura 2.62: Ejemplo de párrafos flotados [Imagen creada por el autor de este trabajo (2011)]

Clear	**No fluir ante flotantes**
Posibles valores	left, right, none, both, inherit **left**: div{clear:left;} a la izquierda de los "div" no se deben presentar flotantes. **right**: p{clear:right;} a la derecha de los párrafos no se deben presentar flotantes.
Valor de inicio	none
Se aplica a todos los elementos en bloque.	

Tabla 2.27: Regla clear

En la figura 2.63 se puede observar como clear contrarresta el comportamiento de fluir alrededor de los elemento flotados.

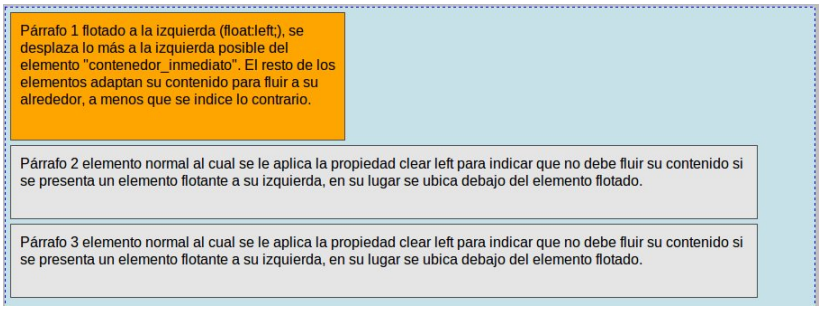


Figura 2.63: Ejemplo del uso de clear [Imagen creada por el autor de este trabajo (2011)]

2.3.4.3. Visualización

Las propiedades de CSS que se encargan de la visualización de los elementos son:

- visibility.- Permite visualizar u ocultar un elemento o grupo de elementos conservando su espacio original.
- display.- Permite visualizar o desaparecer un elemento o grupo de elementos de la página por lo que no conservan el espacio original, es decir, sale del flujo normal de la página y el resto de los elementos pasan a ocupar el espacio dejado.
- overflow.- Obliga a un elemento a ocupar y envolver todo su contenido para ser mostrado.
- z-index.- Indica el plano en z en el que se mostrará el elemento, es decir, entre más alto sea su valor más cercano estará a la vista del usuario.

Visibility

Permite que un elemento o grupo de elementos sean invisibles. El navegador construye el elemento y le proporciona las propiedades según el contexto al que pertenece dicho elemento, pero sin mostrarlo. Al tener el resto de las propiedades los elementos adyacentes conocen su posición y respetan su espacio de definición.

Visible	**visibility**
Posibles valores	visible, hidden, collapse, inherit **visible**: div{visibility:visible;} valor por default, se muestra el elemento. **hidden**: p{visibility:hidden;} hace invisible al elemento.
Valor de inicio	visible
Se aplica a todos los elementos.	

Tabla 2.28: Propiedades de visibilidad

En la figura 2.64 se muestra el uso de visibility con la opción hidden, en la cual el elemento 9 es invisible pero el navegador construye su espacio y forma parte del contexto de bloque de los elementos flotados, por lo que el resto de los elementos respetan su lugar. Para ejemplificar el comportamiento se hace uso de la herramienta firebug que permite analizar la estructura HTML y el CSS de cada elemento. En la figura se puede observar que el navegador construyó la caja del elemento 9 y únicamente ocultó su contenido.

Display

La propiedad display es parecida a visibility pero tiene más variantes y diferencias significativas. Con ella se puede ocultar un elemento de la página de tal forma que desaparece del la estructura de la página, además se puede conseguir que los elementos se comporten de otra forma, inclusive obtienen propiedades fuera de las pertenecientes a su estructura original. Al aplicarle

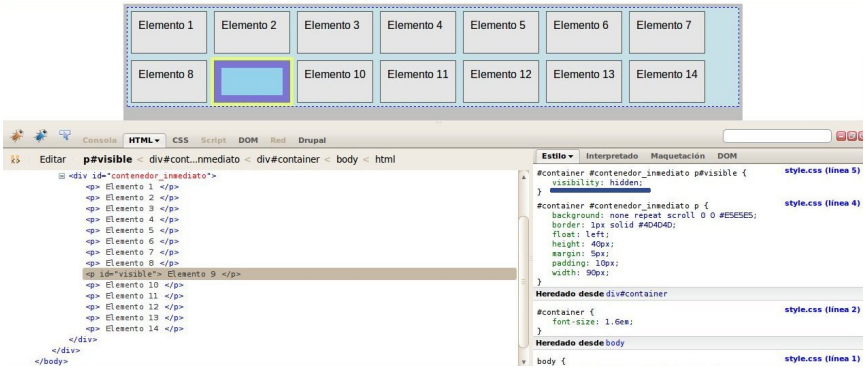


Figura 2.64: Visibility hidden [Imagen creada por el autor de este trabajo (2011)]

a un elemento display none este desaparece de la página y los elementos adyacentes pasan a ocupar el espacio dejado.

Propiedad display	**display**
Posibles valores	inline, block, none, list-item, run-in, inline-block, table, inline-table, table-row-group, table-row, table-column-group, table-column, table-cell, table-caption, inherit **block**: span{display:block;} cambia el comportamiento del elemento "span", el "span" pasa a comportarse como un elemento en bloque. **none**: p{display:none;} desaparece el elemento.
Valor de inicio	inline
Se aplica a todos los elementos.	

Tabla 2.29: Propiedades de display

En la figura 2.65 el elemento 10 desaparece de la estructura de la página por lo que el resto de los elementos pasan a ocupar el espacio dejado, usando firebug se observa dicho comportamiento y como el elemento desaparece de la página.

La propiedad display permite desaparecer un elemento de la página, además controla la forma en la que se visualizarán los elementos, es decir, se puede hacer que un elemento en bloque se comporte como uno de línea y viceversa.

Complicaciones entre float, position y display

Cuando un elemento define las propiedades position, display y float se obtiene el siguiente comportamiento.

1.- Si display tiene el valor de **none**, los valores de float y position son ignorados y el elemento no se muestra.

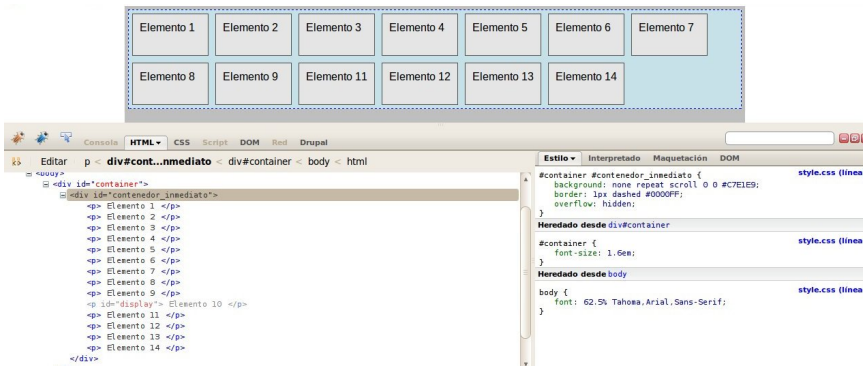


Figura 2.65: Display none [Imagen creada por el autor de este trabajo (2011)]

2.- Si position es fixed o absolute, el elemento se posiciona de forma absoluta y float vale none y display vale block, sin importar que tipo de elemento se trate.

3.- Si float es diferente de none, el elemento flota y se comporta como elemento de bloque sin importar el elemento.

Overflow

Cuando el contenido de los elementos anidados sobrepasan el espacio dejado por el elemento contenedor se produce un desborde, este comportamiento se puede contrarrestar usando la propiedad overflow, que permite indicar la forma de visualizar el contenido de los elementos que se desbordan de su elemento contenedor.

Este desborde ocurre cuando se define un ancho y un alto a un elemento de forma explícita y el contenido es mayor o cuando se utilizan etiquetas "pre" y la definición de anchos y altos no ayudan a abarcar dicho contenido.

Propiedad overflow	**overflow**
Posibles valores	visible, hidden, scroll, auto, inherit **visible**: p{overflow:visible;} si el contenido es mayor al espacio disponible se muestra el desborde. **hidden**: div{overflow:hidden;} si el contenido es mayor al espacio disponible oculta el contenido desbordado y deja visible solo el contenido abarcado.
Valor de inicio	visible
Se aplica a todos los elementos.	

Tabla 2.30: Propiedad overflow

Con "visible" se indica que se debe mostrar el desborde, "hidden" solo muestra el contenido abarcado, "scroll" oculta el contenido desbordado pero proporciona una barra de desplazamiento para acceder a él.

En la figura 2.66 se ejemplifica el resultado de aplicar overflow visible, hidden y auto respectivamente, en el primer párrafo el contenido se desborda y es visible dicho comportamiento, en el segundo se oculta el desborde y en el tercero se incluye un scroll-bar para acceder al contenido.

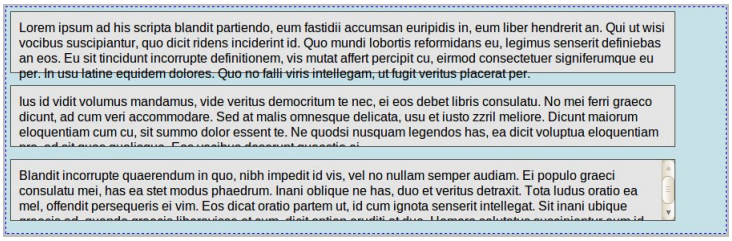


Figura 2.66: Overflow: visible, hidden y auto [Imagen creada por el autor de este trabajo (2011)]

z-index

Hasta el momento es posible colocar un elemento de forma horizontal y vertical usando posicionamiento, en los estándares CSS también es posible establecer la posición en el eje z, este comportamiendo sirve para indicar que tan cerca o lejos de la vista del usuario se deben colocar los elementos cuando se producen solapamientos. La condición para que z-index tenga efecto es que los elementos deben estar posicionados, por lo tanto cuando los elementos se solapan se puede indicar que elementos se vean por encima y cuales otros se vean por debajo (entiéndase "por debajo" al comportamiento de mostrar el contenido o parte del contenido de un elemento obstruido por el contenido de otro).

Eje z	**z-index**
Posibles valores	auto, número, inherit **número**: p{z-index:5;} **número**: p.encima{z-index:10;} al producir el solapamiento los párrafos con la clase "encima" mostrarán su contenido por encima de los párrafos normales.
Valor de inicio	auto
Se aplica a todos los elementos posicionados.	

Tabla 2.31: Regla z-index

En la figura 2.67 se observa el uso de z-index aplicado a tres párrafos posicionados, dos de forma relativa y el tercero de forma absoluta, se aplican desplazamientos para producir solapamientos y con las propiedades z-index y el uso de imágenes con canal alfa en el CSS, se consigue el efecto de ver a la caja con fondo verde claro por detrás de la caja color anaranjado, que a su vez se encuentra por detrás de la caja azul claro.

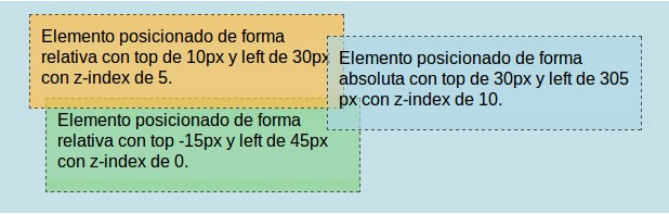


Figura 2.67: z-index [Imagen creada por el autor de este trabajo (2011)]

2.3.4.4. Práctica 04

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.3.4 Posicionamientos, en cuyas secciones se tratan los conceptos de elementos en bloque, elementos en línea y los contextos que forman. Se explican los tipos de posicionamiento y los métodos que CSS proporciona para desplazar, colocar, visualizar y ocultar elementos de la página. Además se explica el concepto de elementos flotados y la forma de trabajar con ellos para evitar los desbordamientos.

Ejercicios Los documentos HTML de esta práctica se localizan en "complementos/posicionamientos/ejercicios".

1.- ¿Qué ocurre cuando se anida un elemento de bloque en un elemento de línea?

2.- Qué ocurre cuando se aplican las propiedades top, left, right y bottom a un elemento con posición:

- static
- absolute
- fixed
- float

3.- ¿Cuáles son las diferencias entre elementos posicionados y los elementos flotados?

4.- Construir las reglas de estilo necesarias para cumplir los siguientes estilos, aplicados al documento "postear/postear.html" (se recomienda leer la sección 2.3.5.6 Un poco de CSS3). El resultado esperado se muestra en la figura 2.68.

4.1.- Aplicar un color de fondo al body de #ADD8E6 y con la imagen postear-fondo2.png.

4.2.- Los titulares principales deben presentar un color de #0A33CC y una separación de los adyacentes de 20px lateralmente y 10px de separación inferior.

4.3.- Aplicar al elemento container un ancho de 900px y un alto mínimo de 800px, un color de fondo #E4E4E4 con imagen fondo-body.jpeg. un tamaño de letra de 1.4em, un centrado horizontal sin separación vertical y un relleno de 5px.

4.4.- Los divs cuya clase sea "post" deben presentar un ancho de 840px, centrado horizontal con separación vertical de 5px, un relleno de 3px, un borde transparente de 1px de espesor, un color de fondo #FFF con imagen fondo-secciones.jpeg, esquinas redondeadas con las siguientes medidas; superior-izquierdo e inferior-derecho de 20px y superior-derecho e inferior-izquierdo de 5px, sombras con las siguientes especificaciones; sombra1 de color azul claro con difuminado de 4px con desplazamiento vertical de 2px y horizontal de 2px y sombra2 de color gris claro con difuminado de 6px con los desplazamientos invertidos a la sombra1.

4.5.- Los divs cuya clase sea "datos_user" deben presentar una anchura del 30 %, un tamaño de letra de 0.8em y un color de letra de #2E2F2F.

4.6.- Los párrafos contenidos en "datos_user" deben presentar un tamaño de letra de 1.2em, sin relleno y sin separación inferior.

4.7.- Los enlaces contenidos en "datos_user" deben presentar una separación vertical de 3px sin separación horizontal.

4.8.- Los span contenidos en "datos_user" deben presentar un estilo de letra italic, sin separación y sin relleno.

4.9.- Los enlaces en general deben poseer un subrayado solo cuando se pase el mouse sobre ellos y presentar una separación de 5px.

5.- Si un elemento posicionado presenta las propiedades de desplazamiento y valores de left y right con auto qué ocurre con su posición. Y si el elemento presenta valores diferentes a auto qué ocurre.

6.- ¿Para que sirve posicionar un elemento en relativo sino recibe valores de desplazamiento?

7.- ¿Existe algún incoveniente al posicionar un elemento en relative o en absolute?

8.- Explica lo que le ocurre al ancho de un elemento que no tiene definido ningún ancho (auto por definición) y está posicionado de forma absoluta y contiene varios párrafos cuyos anchos definidos son menores al del contenedor. Ejemplifica tu respuesta.

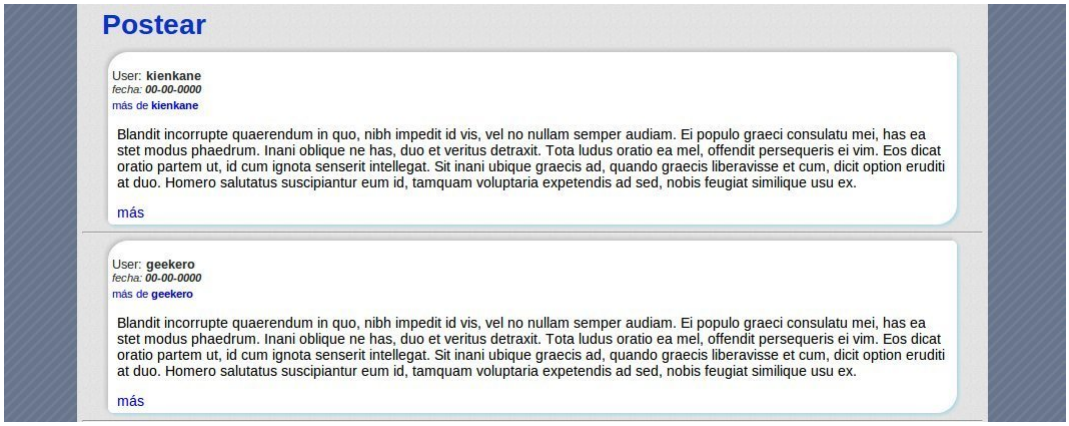


Figura 2.68: Sección de comentarios [Imagen creada por el autor de este trabajo (2011)]

9.- Si en el código 2.31 el elemento "absolute" está posicionado de forma absoluta, explicar que desplazamiento o que comportamiento se produce cuando: (La ruta del archivo se localiza en: "absolute/absolute.html").

Código 2.31: Ejercicio 9 práctica 04

```
<body>
<div id="container">
<div id="contenedor_inmediato">
<p id="absolute">
Lorem ipsum ad his scripta blandit partiendo,
eum fastidii accumsan euripidis in, eum liber
hendrerit an. Qui ut wisi vocibus suscipiantur.
</p>
<p>
Quo dicit ridens inciderint id. Quo mundi
lobortis reformidans eu, legimus senserit
definiebas an eos. Eu sit tincidunt
incorrupte definitionem.
</p>
<p>
Vis mutat affert percipit cu, eirmod consectetuer
signiferumque eu per. In usu latine equidem dolores.
Quo no falli viris intellegam, ut fugit.
</p>
</div>
</div>
</body>
```

9.1.- Si ninguno de sus contenedores están posicionados y se aplica algún desplazamiento al elemento "absolute".

9.2.- Si el elemento "contenedor_inmediato" está posicionado de forma relativa y se aplica (al "absolute") un desplazamiento left de 20px y un top de 30px.

9.3.- Si el elemento "container" está posicionado de forma relativa y se aplica (al "absolute") un desplazamiento left de 40px, top de 30px y right de 60px.

9.4.- Si el elemento "contenedor_inmediato" está posicionado de forma relativa y se aplica (al "absolute") un desplazamiento right de 60px y un top de 40px. ¿Qué ocurre con el punto de referencia superior izquierda? ¿Tiene algo que ver el valor de los desplazamientos con los puntos de referencia?

9.5.- Si el elemento "contenedor_inmediato" está posicionado de forma relativa y se aplica (al "absolute") un desplazamiento bottom de 60px y un right de 40px. ¿Cuál es el punto de referencia y porqué?

9.6.- Si el elemento "contenedor_inmediato" está posicionado de forma absoluta y se aplica (al "absolute") un desplazamiento bottom de 60px y un right de 40px. Explica lo que le ocurre al elemento "contenedor_inmediato".

9.7.- Si el elemento "contenedor_inmediato" está posicionado de forma relativa y no se aplica ningún desplazamiento al "absolute".

10.- ¿Qué diferencias hay entre absolute y fixed?

11.- ¿Qué soporte proporciona IE 6 y 7 para el posicionamiento fixed?

12.- ¿Qué ocurre cuando se le definen propiedades de desplazamiento a los elementos flotados?

13.- ¿Un elemento flotado pueder ser posicionado? Justifica tu respuesta.

14.- ¿Qué ocurre cuando uno o varios párrafos son adyacentes a un elemento con float left?

15.- Construir las reglas de estilo necesarias para cumplir la siguiente situación en el documento "float/float.html". Se recomienda leer la sección 2.3.4.2 ReferenciasCSS:floats.

El documento floats.html tiene un "div" con id "menu" el cual no tiene definida la altura y contiene dos elementos "span" flotados, uno a la izquierda y el otro a la derecha, debajo de ellos está definido un "div" para conseguir que el contenedor "menu" reconozca a los "span", es decir, evitar que se desborden los floats, como se muestra en la figura 2.69.

Figura 2.69: Elementos flotantes desbordados [Imagen creada por el autor de este trabajo (2011)]

- Que reglas se deben aplicar al "div" inmediato a los elementos flotados, para conseguir la apariencia mostrada en la figura 2.70, sin modificar las propiedades del elemento "menu" y de ningún otro elemento.

Figura 2.70: Elementos flotantes abarcados [Imagen creada por el autor de este trabajo (2011)]

- ¿Es necesario utilizar un "div" o se puede utilizar cualquier elemento para conseguir dicho resultado? Explica tu respuesta.
- Existe alguna razón para considerar este procedimiento inapropiado para resolver la situación. Explica tu respuesta.

16.- Construir las reglas de estilo necesarias para cumplir con el estilo mostrado en la figura 2.71 que pertenece al documento "postear2/ postear.html". Se recomienda leer la sección 2.3.5.6 opacity
.

a .- Imagen de fondo.

b .- Bordes sombreados en la sección principal.

c .- Opacidad en la imagen y un marco de color azul en el cual se distinguen dos azules.

d .- Los datos del usuario fluyen entorno a la imagen.

e .- Borde izquierdo de la sección de comentarios.

f .- Sombra superior color gris.

g .- Sombra inferior color azul claro.

h .- Enlace visible en la parte inferior derecha.

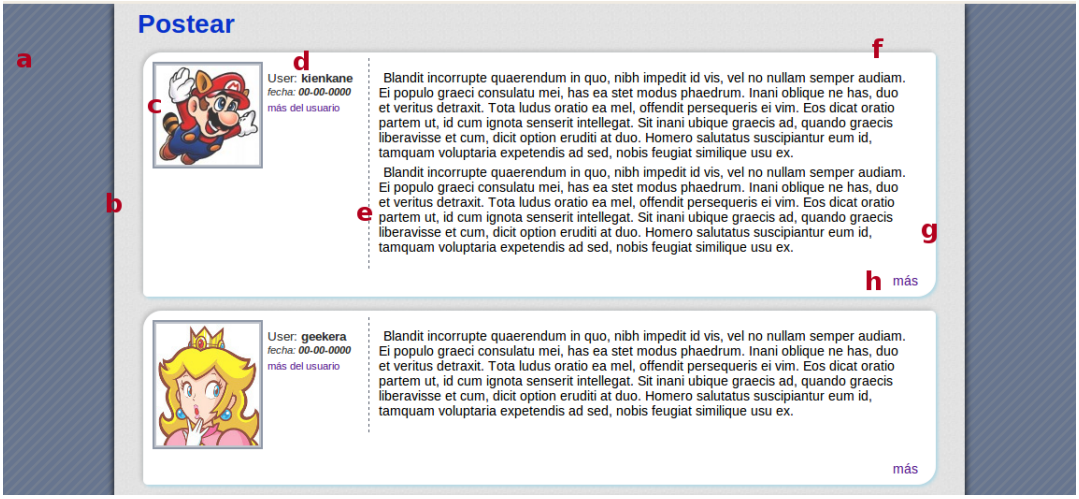


Figura 2.71: Sección para postear en CSS [Imagen creada por el autor de este trabajo (2011)]

17.- ¿Qué utilidad tienen las propiedades visibility y display para el diseño de una página Web? Ejemplifica tu respuesta.

18.- Menciona una optimización de visibility y display con JavaScript.

19.- ¿Qué usos tiene la propiedad display block e inline? Ejemplifica tu respuesta.

20.- ¿Qué comportamiento tiene la propiedad display inline-block y qué utilidad tiene en el diseño de páginas Web?

21.- ¿Es mejor usar la propiedad display table, que definir una tabla en HTML?

22.- ¿Qué utilidad tiene la propiedad overflow?

23.- Con los temas vistos hasta la sección 2.3.4.3 ReferenciasCSS:visualizacion particularmente overflow, indicar de que otra forma se puede solucionar el problema propuesto en el ejercicio 15 sin utilizar un elemento “div”.

24.- ¿Cuál es la condición para que z-index tenga efecto? ¿Qué utilidad tiene z-index?

25.- ¿Es posible utilizar valores negativos a z-index? ¿Qué utilidad tendría hacer eso?

2.3.5. Complementos

2.3.5.1. Tipografía

Las familias tipográficas disponibles en cada sistema operativo son diferentes, inclusive entre versiones diferentes de un sistema operativo. Puesto que arriba del 90 % de los usuarios de Internet utilizan Windows o Mac Os, parece lógico diseñar páginas Web buscando la mayor compatibilidad tipográfica entre ambos sistemas. Las fuentes instaladas por defecto en Windows y Mac OS son:

Windows	Mac OS
Arial	Courier
Verdana	Geneva
Times New Roman	Helvetica
MS Sans Serif	New York
MS Serif	Palatino
Courier	Times

Tabla 2.32: Fuentes disponibles

Por otra parte, Microsoft, Apple y las distintas empresas involucradas en Linux ofrecen a sus usuarios paquetes gratuitos de fuentes, por lo que el número de ellas presentes en una máquina puede aumentar considerablemente. Un ejemplo de estos paquetes es Windows Font Pack, que facilita fuentes tanto para entornos Windows y Mac.

Windows	Mac OS
Arial	Helvetica
Courier New	Courier
Times New Roman	Times
MS Sans Serif	Geneva
MS Serif	New York

Tabla 2.33: Fuentes compatibles Windows - Mac

Clasificación de fuentes serif y sans serif

Una clasificación de las familias de fuentes más utilizada en medios digitales es la que divide las familias tipográficas en Serif y Sans Serif.

Las fuentes serif o serifas tienen origen en el pasado cuando las letras se cincelaban en bloques de piedra, pero resultaba difícil asegurar que los bordes de las letras fueran rectos, por lo que el tallador desarrolló una técnica que consistía en destacar las líneas cruzadas para el acabado de casi todas las letras, por lo que las letras presentaban en sus extremos unos remates muy

característicos conocidos con el nombre de serif, como se muestra en la figura 2.72.

Figura 2.72: Tipografía Serif [Imagen creada por el autor de este trabajo (2011)]

Las tipografías romanas se basaban en círculos perfectos y formas lineales equilibradas, las letras redondas como la o, c, p, b tienen que ser un poco más grandes porque ópticamente parecen más pequeñas cuando se agrupan con otras letras.

Las fuentes serif incluyen todas las romanas. Son muy apropiadas para la lectura seguida de largos textos, ya que los trazos finos y los remates ayudan al ojo a fijar y seguir una línea en un conjunto de texto, facilitando la lectura rápida y evitando la monotonía.

Como ejemplos de fuentes serif podemos citar Book Antiqua, Bookman Old Style, Courier, Courier New, Century Schoolbook, Garamond, Georgia, MS Serif, New York, Times, Times New Roman y Palatino.

Las fuentes **sans serif** hacen su aparición en Inglaterra durante los años 1820 a 1830. No tienen remates en sus extremos (sin serif), entre sus trazos gruesos y delgados no existe apenas contraste, sus vértices son rectos y sus trazos uniformes, opticamente ajustados en sus trazos rectos. La figura 2.73 es un ejemplo de una letra sans serif.

Figura 2.73: Tipografía Sans-Serif [Imagen creada por el autor de este trabajo (2011)]

Asociados desde su inicio a la tipografía comercial, su legibilidad y durabilidad los hacían perfectos para impresiones de etiquetas, embalajes, envolturas y demás propósitos comerciales.

No están aconsejadas para textos largos, ya que resultan monótonas y difíciles de seguir.

Entre las fuentes sans serif se encuentran Arial, Arial Narrow, Arial Rounded MT Bold, Century Gothic, Chicago, Helvetica, Geneva, Impact, Monaco, MS Sans Serif, Tahoma, Trebuchet MS y Verdana.

Tipografías recomendadas

El CSS ha proporcionado muchísimas posibilidades en términos de ti-

pografías al diseño web, pero normalmente se utilizan las mismas 5 fuentes que se creen siempre están en todas las computadoras. Si la fuente especificada no está disponible no se mostrará el texto cómo se desea, para variar un poco en el diseño del tipo de letra en 3.7 design se ha hecho una recopilación de 8 fuentes que están disponibles en casi todas las computadoras y que se pueden utilizar con seguridad en los diseños, las fuentes son:

Palatino Linotype / Palatino Disponible en el 97.09 de los Windows y en el 78.86 de las Mac

Tahoma Disponible en el 96.09 de los Windows y en el 72.02 de las Mac

Impact Disponible en el 95.85 de los Windows y en el 88.07 de las Mac

Century Gothic Disponible en el 85.44 de los Windows y en el 42.5 de las Mac

Arial Black Disponible en el 97.73 de los Windows y en el 96.18 de las Mac

Arial Narrow Disponible en el 87.08 de los Windows y en el 91.01 de las Mac

Copperplate / Copperplate Gothic Light Disponible en el 58.13 de los Windows y en el 85.85 de las Mac

Gill Sans / Gill Sans MT Disponible en el 43.09 de los Windows y en el 90.82 de las Mac

Color del texto	**color**
Posibles valores	(nombre de color || color rgb || color en hexadecimal) **nombre**: div{color:blue;}, **rgb**: span{color:rgb(255,0,0);} **hex**: p{color:#00f;}
Se aplica a todos los elementos.	

Tabla 2.34: Establece el color de letra

Reglas CSS

El siguiente ejemplo establece a todos los elementos de la página un color de letra negro.

```
1 body{color:#000;}
```

Tipografía	**font-family**
Posibles valores	(nombre de la tipografía | familia de la tipografía)* | inherit) **nombre-tipografía**: div{font-family:Arial;}, **nombre-familia**: span{font-family:sans-serif;} **nombre-tipografía**: body{font-family:Arial, Verdana, Helvetica, sans-serif;}
Se aplican a todos los elementos.	

Tabla 2.35: Establecer la tipografía

- serif – Times New Roman
- sans-serif – Arial
- cursive – Comic Sans
- fantasy – Impact
- monospace – Courier New

Cuando se utiliza el nombre de la tipografía se indica que la página debe visualizarse con ese tipo de letra, esa opción depende de las fuentes instaladas en la computadora del usuario. Cuando se utiliza el nombre de la familia se indica el estilo de la letra y no una fuente en concreto, es decir, al especificar únicamente la familia, el navegador visualizará la página con una fuente instalada en el ordenador perteneciente a la familia indicada.

Es recomendable indicar el nombre de las fuentes a utilizar separadas por comas e indicar la familia a la que pertenecen, de esta forma si el ordenador no cuenta con alguna de las fuentes indicadas utilizará alguna perteneciente a la familia.

Tamaño de letra	**font-size**
Posibles valores	(medida relativa | medida absoluta | medida | porcentaje | inherit) **relativa**: div{font-size:1.2em;}, **absoluta**: span{font-size:14pc;} **medida**: p{font-size:larger;}
Se aplican a todos los elementos.	

Tabla 2.36: Establecer el tamaño de letra

Se puede establecer el tamaño de forma relativa utilizando las palabras claves “larger” y “smaller” que dependen del tamaño de letra de su elemento contenedor. También se pueden utilizar las palabras clave “xx-large”, “x-large”, “large”, “medium”, “small”, “x-small”, “xx-small”, estas medidas son absolutas y dependen del navegador. Para diseñar páginas Web dinámicas

y diseños flexibles se recomienda utilizar tamaños de letra en porcentajes o em.

Peso de la letra	**font-weight**
Posibles valores	(normal | bold | bolder | lighter | 100 - 900 | inherit) **normal**: div{font-weight:normal;}, **bold**: span{font-weight:bold;} **lighter**: p{font-weight:laghter;}
Valor de inicio	normal equivalente a 400 en numérico.
Se aplican a todos los elementos.	

Tabla 2.37: Establecer el peso de letra

Estilo de la letra	**font-style**
Posibles valores	(normal | italic | oblique | inherit) **normal**: div{font-style:normal;}, **italic**: span{font-style:italic;} **oblique**: p{font-style:oblique;}
Valor de inicio	normal.
Se aplican a todos los elementos.	

Tabla 2.38: Establecer el estilo de letra

Estilo alternativo	**font-variant**
Posibles valores	(normal | small-caps | inherit) **normal**: div{font-variant:normal;}, **small-caps**: span{font-variant:small-caps;}
Valor de inicio	normal.
Se aplican a todos los elementos.	

Tabla 2.39: Letra versal

Fuente	**font**
Posibles valores	(font-style | font-variant | font-weight)? font-size | (line-height)? | font-family | inherit **ej1**: body{font:normal 67 % Arial, Verdana, sans-serif;}, **ej2**: span{font:bold 1.4em/1.2em Helvetica, Arial, sans-serif;} **ej3**: #aviso{font:normal 16px/18px "Trebuchet MS", Arial, Helvetica, sans-serif}
Se aplican a todos los elementos.	

Tabla 2.40: Regla resumida

El orden de aplicación es:

1.- Se establecen la propiedades de style, variant y weight en cualquier orden.

2.- Se establece el tamaño y la altura del párrafo.

3.- Se establecen los tipos de letras y las familias.

Alineación del texto	**text-align**
Posibles valores	(left | right | center | justify | inherit **left**: p{text-align:left;}, alineación a la izquierda **right**: span{text-align:right;} alineación a la derecha **justify**: p{text-align:justify;} texto justificado
Valor inicio	left
Se aplican a todos los elementos.	

Tabla 2.41: Alineación del texto

Interlineado	**text-height**
Posibles valores	(normal | número | medida | porcentaje | inherit **normal**: p{text-height:normal;}, **número**: span{text-height:1.2;} valor que hace referencia a 1.2em o 120 % del tamaño de letra **porcentaje**: p{text-height:110 %;}
Valor inicio	normal
Se aplican a todos los elementos.	

Tabla 2.42: Separación entre líneas de texto

Decoración	**text-decoration**
Posibles valores	none | underline | overline | line-through | blink | inherit **none**: p{text-decoration:none;} sin decoración **underline**: span{text-decoration:underline;} texto subrayado
Valor inicio	none
Se aplican a todos los elementos.	

Tabla 2.43: Decoración del texto

Transformación	**text-transform**
Posibles valores	none | capitalize | uppercase | lowercase | inherit **none**: p{text-transform:none;} **capitalize**: span{text-transform:capitalize;} Pone en mayúsculas la primera letra de cada palabra.
Valor inicio	none
Se aplican a todos los elementos.	

Tabla 2.44: Transformación del texto

Para establecer sangrías en el texto se utiliza la propiedad text-indent.

Tabulación	**text-indent**
Posibles valores	medida | porcentaje | inherit **medida**: p{text-indent:10px;} **porcentaje**: span{text-indent:5 %;}
Valor inicio	0
Se aplican a todos los elementos.	

Tabla 2.45: Tabulación del texto

Se puede controlar el espacio entre letras y la separación entre palabras. Para ello se utilizan letter-spacing y word-spacing.

Separación de letras	**letter-spacing**
Posibles valores	medida | normal | inherit **ej1**: p{letter-spacing:5px;} **ej2**: span{letter-spacing:0.5em;}
Valor inicio	normal
Se aplican a todos los elementos.	

Tabla 2.46: Separación entre letras

Separación de palabras	**word-spacing**
Posibles valores	medida | normal | inherit **ej1**: p{word-spacing:8px;} **ej2**: span{word-spacing:0.4em;}
Valor inicio	normal
Se aplican a todos los elementos.	

Tabla 2.47: Separación entre palabras

Estos valores pueden ser positivos o negativos, cuando son positivos los elementos se separan y con valores negativos la separación entre elementos disminuye.

Espacios en blanco	**white-space**
Posibles valores	pre | normal | nowrap | pre-wrap | pre-line | inherit **pre**: p{white-space:pre;} como la etiqueta pre de HTML **pre-wrap**: p{white-space:pre-wrap;}
Valor inicio	normal
Se aplican a todos los elementos.	

Tabla 2.48: Espacios vacíos

- pre.- idéntico a la etiqueta “pre” de HTML.
- nowrap.- elimina los espacios en blanco y los saltos de línea, si el contenido es mayor al espacio de su contenedor se desborda.
- pre-wrap.- respeta los espacios en blanco y los saltos de línea pero ajusta el contenido al espacio de su contenedor.
- pre-line.- elimina los espacios en blanco pero respeta los saltos de línea ajustando el contenido al espacio de su contenedor.

Uno de los mayores problemas referente al texto es la alineación vertical cuando se encuentra una imagen de por medio. Un ejemplo se muestra en la figura 2.74.

Alineación	**vertical-align**
Posibles valores	baseline | sub | super | top | text-top | middle | bottom | text-bottom | porcentaje | medida | inherit **base-line**: a{vertical-align:base-line;} al pie de la imagen **middle**: span{vertical-align:middle;}
Valor inicio	baseline
Se aplica a los elementos en línea y celdas de una tabla	

Tabla 2.49: Alineación vertical

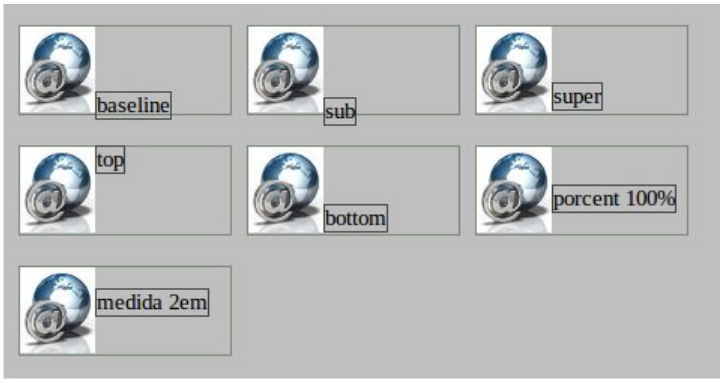


Figura 2.74: Alineación vertical [Imagen creada por el autor de este trabajo (2011)]

2.3.5.2. Medios css

Una de las características más importantes de las hojas de estilos CSS es que permiten definir diferentes estilos para diferentes medios o dispositivos por ejemplo:

Pantallas, impresoras, móviles, proyectores, etc.

CSS define algunas propiedades específicas para determinados medios, como por ejemplo la paginación y los saltos de página para los medios impresos y tipo de voz para los medios de audio.

Nombre y descripción del medio en CSS:

Medio	Descripción
all	Todos los medios
braille	Dispositivos táctiles que usan braille
embosed	Impresoras braille
handheld	Dispositivos móviles
print	Impresoras y "Vista Previa para Imprimir"
screen	Pantallas
speech	Sintetizadores de navegación por voz
tv	Televisores y dispositivos de resolución baja

Tabla 2.50: media css

Las reglas *@media* son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de *@media*. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

A continuación se muestra un ejemplo sencillo:

```
@media print{
    body{font-family:Verdana; font-size:16pt;}
    p{font-size:16pt; color:#0000FF; font-weight:500;}
}
```

Cuando se utilizan reglas de tipo @import para enlazar archivos CSS externos, se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL del archivo CSS:

```
@import url('css/style_print.css') print;
@import url('css/style_handheld.css') handheld;
```

Estas reglas indican que cuando el documento se imprima debe tener el aspecto definido por el archivo style_print.css y cuando se visualice en un dispositivo móvil debe tener el aspecto definido en style_handheld.css.

Además en el HTML a través de la etiqueta link (con tipo de archivo text/css y con relación stylesheet) se puede utilizar la propiedad media para indicar el tipo de medio o medios en los que se aplicarán los estilos a los documentos enlazados. A continuación se muestra un ejemplo de su uso:

```
<link rel="stylesheet" type="text/css" href="css/style.
    css" media="screen">
<link rel="stylesheet" type="text/css" href="css/
    style_print.css" media="print">
```

Si no se indica algún valor de media en la propiedad se interpreta como all.

2.3.5.3. Extras sobre selectores CSS

Es normal que en sistemas complejos se definan varios estilos aplicados a un mismo elemento en reglas separadas, esto no es aconsejable ya que demora el tiempo de carga y dificulta su mantenimiento.

por ejemplo:

```
/*--- Reglas CSS ---*/
p {border:1px solid #0f0;}
[...]
p {padding:3px;}
[...]
p {margin:5px;}
```

En situaciones como la anterior es recomendable agrupar las reglas en una sola.

```
/*--- Reglas CSS ---*/
p {border:1px solid #0f0;
   padding:3px;
   margin:5px;}
```

Para una mejor optimización es posible quitar los espacios en blanco, ya que son precisamente eso, espacios que el motor del navegador no toma en cuenta. Es recomendable hacer esto cuando el sistema se encuentra en tiempo de producción, es decir, se satisficieron todos los criterios

```
/*--- Reglas CSS ---*/
p {border:1px solid #0f0;padding:3px;margin:5px;}
```

Herencia

Este concepto hace referencia a la situación de que si algún un elemento se le aplica un estilo ese estilo o parte del estilo es heredado a algunos elementos descendientes (definidos dentro de sus etiquetas de definición).

Por ejemplo si aplicamos una tipografía al body ese estilo se hereda a todos los elementos, salvo que se defina otra tipografía de forma particular.

En el código 2.32 se aplica una fuente de texto Arial a todos los elementos del body, pero a los elementos cuyo valor del atributo class sea "consola" reciben una fuente Courier.

Código 2.32: Definir fuente

```
/*--- Reglas CSS ---*/
body {font-family:Arial;}
.consola{font-family:Courier, monospace;}
```

Colisiones de estilo

Cuando una hoja de estilo es compleja, es posible que contenga varias reglas CSS aplicadas a un mismo elemento HTML, pero el problema radica en que se pueden dar colisiones de estilo.

Código 2.33: Colisión de estilo

```
/*--- Reglas CSS ---*/
p {color:#00F;}
p {color:#0F0;}
```

En las reglas de estilo del código 2.33 se presenta una colisión, ya que se aplica un estilo diferente a una misma propiedad a todos los parrafos del body.

Para resolver estos conflictos se lleva acabo el siguiente proceso:

1.- Se localizan todas las declaraciones que son aplicadas al mismo elemento para el medio seleccionado (pantalla, impreso, pdas etc.).

2.- Se ordenan todas la declaraciones según su origen, es decir si se tratan de una regla proveniente del CSS del navegador, del usuario o del diseñador.

3.- Se ordenan las declaraciones según la especificidad del selector. Cuanto más específico sea el selector es más importante.

4.- Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó al final.

Este proceso se resume en las siguientes reglas:

1.- Cuanto más específico es un selector, más importancia tiene la regla asociada.

2.- A reglas con la misma especificidad se considera la última indicada.

2.3.5.4. Medidas en CSS

Las unidades en CSS se emplean para establecer el tamaño de letra, las alturas y anchuras de los elementos, el grosor de los bordes, la separación del relleno (padding), etc. Todas las unidades están formadas por un valor numérico seguido por una unidad de medida.

Cuando el valor de la medida es 0 la unidad se puede omitir (ej. margin:0px; y margin:0; son equivalentes) en cualquier otro caso se omite la regla y se considera un error.

En CSS hay dos grupos de medidas: relativas y absolutas.

Medidas Relativas

En las medidas relativas el valor real se obtiene a partir de operaciones con el valor indicado.

- em.- se calcula a partir del tamaño estándar de la tipografía usada. Aproximadamente el valor de 1em es la anchura de la letra M (eme mayúscula).
- ex.- similar a em, pero el valor de 1ex es la altura de la letra x (equis minúscula).
- px.- se calcula dependiendo de la pantalla con la que el usuario visualiza el documento.
- %.- siempre esta referenciando a otra medida, es decir a partir de una medida calcula el porcentaje indicado y lo aplica al elemento.

Por ejemplo si se utiliza en una página un tamaño de letra de 14pt (14 puntos) entonces 1em equivale a 14pt. Estos 14pt son establecidos por las hojas de estilo del navegador, así que se pueden redefinir los valores para obtener un tamaño al 90 % del original. Esto se consigue estableciendo un tamaño de letra de 0.9em[6] en el body, este estilo será heredado a todos los elementos del documento (contenidos en el body) a menos que reglas posteriores definan de forma particular otro tamaño.

```
1 /*--- Reglas CSS ---*/
2 body {font-size:.9em;} /*tamaño de 12pt*/
```

Al tratarse de una regla relativa es necesario conocer el valor de referencia para calcular el valor real, en el ejemplo anterior el navegador aplica un tamaño de 14pt y la regla aplica sólo el 90 % de esos 14pt, es decir $14pt*,9 = 12{,}6pt$

[6] Cuando el valor de una medida es menor a la unidad se puede omitir el cero, por lo que la unidad anterior queda como .9em

La clave para entender el comportamiento anterior radica en conocer el valor previo (heredado) del elemento contenedor (también llamado padre) ya que a partir de ese valor se calcula el valor real. En el ejemplo anterior el body es el elemento padre más cercano que define un tamaño de letra, por lo que se calcula el valor real a partir de ese medida. De aquí la importancia de redefinir el tamaño de letra de la etiqueta body desde las hojas de estilo del desarrollador, de lo contrario se aplicará el estilo definido por el navegador.

Los porcentajes se suelen utilizar para establecer el tamaño de la fuente, el ancho y alto de los elementos.

En el código 2.34 se establece un tamaño de letra del 90 % del establecido por el navegador a todos los elementos contenidos en el body, después la segunda regla redefine el tamaño de letra de los titulares principales (h1) al doble del establecido por la regla anterior (esto se debe a que el h1 esta contenido en el body y esta regla es heredada) esta regla es equivalente a definir 2em, la tercera regla redefine el tamaño de letra de los titulares secundarios (h2) a 1.5 veces a la definida en el body es decir 1.5em y la última regla redefine el tamaño de los párrafos contenidos en el body al tamaño definido por la primera regla, es decir, está regla se puede omitir.

Código 2.34: Tamaños de letra

```
1 /*--- Reglas CSS ---*/
2 body {font-size:90%;}
3 h1 {font-size:200%;}
4 h2 {font-size:150%;}
5 p {font-size:100%;}
```

Medidas Absolutas

Cuando se definen medidas absolutas el valor real es el indicado en la regla, ya que no dependen de otro valor.

- in.- pulgadas (una pulgada equivale a 2.54cm).
- cm.- centímetros.
- mm.- milímetros.
- pt.- puntos 1pt equivale a $1/72in$.
- pc.- picas 1pc equivale a 12pt.

De estas unidades los puntos se utilizan para indicar el tamaño de las letras cuando se muestran documentos para imprimir o vistas previas, es decir medio “print”.

Conclusiones

Es recomendable utilizar unidades relativas siempre que sea posible, ya que mejora la accesibilidad y mantenimiento de la página. Al utilizar medidas relativas se propicia a que las páginas se adapten fácilmente a los diferentes medios.

Se utilizan medidas en píxeles y porcentajes para establecer medidas de ancho y altura de los elementos y em y porcentajes para los tamaños de letra.

2.3.5.5. Extra sobre el box model

El ancho y alto de un elemento no están determinados únicamente por sus propiedades width y height. La medidas del margin, padding y border de un elemento influyen en el ancho y altura de los elementos.

En el código 2.35 el ancho del párrafo es de 250px, pero se deben agregar los márgenes laterales que suman 60px, los rellenos laterales que suman 20px y los espesores de los bordes laterales que suman 6px, esto implica que el ancho final del elemento es $250+60+20+6 = 336px$. El resultado se muestra en la figura 2.75.

Código 2.35: Anchura de un elemento

```
#container div p{
  width:250px;
  margin:20px 30px;
  padding:5px 10px;
  border:3px solid #000;
}
```

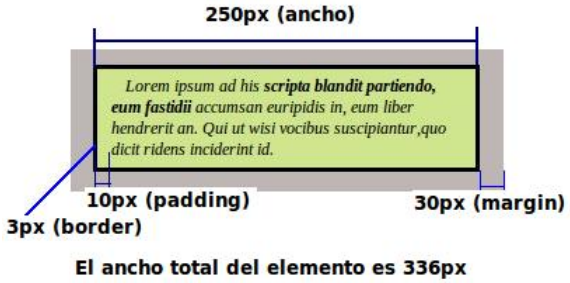


Figura 2.75: La anchura del párrafo [Imagen creada por el autor de este trabajo (2011)]

2.3.5.6. Un poco de CSS3

Esquinas Redondeadas

Una de las características interesantes de la Web 2.0 son las esquinas redondeadas que suelen presentar algunos sitios (entre otros como los degradados, los colores pastel, los reflejos, sombras, etc). Estos diseños llevan muchos años presentes, pero las técnicas para conseguirlos suelen utilizar muchas imágenes y divs para conseguir que una caja muestre esquina redondeadas, algo que no es muy recomendado y en ocasiones es necesario elegir entre lo funcional y la estética.

Con los estándares de la especificación CSS 3 se puede conseguir el resultado de una forma sencilla, el único inconveniente es el soporte que algunos navegadores le dan a estos estándares, en especial IE.

La propiedad del estándar CSS 3 encargada de generar esquinas redondeadas es border-radius.

Esquinas redondeadas	**border-radius**
Posibles valores	(medida){1-4} **ej1**: div{border-radius:5px;} aplica esquinas redondeadas de 5px a cada lado del div **ej2**: div{border-radius:5px 5px 0 0;} aplica esquinas redondeadas de 5px a las esquinas superiores izquierdas y derechas
Valor de inicio	none
Se aplica a todos los elementos.	

Tabla 2.51: Esquinas redondeadas

Es posible establecer las esquinas redondeadas de cada esquina de forma individual, utilizando las reglas para cada esquina.

- border-top-left-radius.- esquina superior izquierda.
- border-top-right-radius.- esquina superior derecha.
- border-bottom-right-radius.- esquina inferior derecha.
- border-bottom-left-radius.- esquina inferior izquierda.

Como el estándar CSS 3 está aun en desarrollo estas propiedades aun no son liberadas formalmente, pero los nevagadores más concientes de estos estándares han liberado sus reglas modificadas que consiguen el resultado deseado.

Mozilla (Firefox 3.5+, SeaMonkey, K-meleon)

- -moz-border-radius.- regla resumida.

- -moz-border-radius-topleft
- -moz-border-radius-topright
- -moz-border-radius-bottomright
- -moz-border-radius-bottomleft

Webkit (Safari, Chrome)

- -webkit-border-radius.- regla resumida.
- -webkit-border-top-left-radius
- -webkit-border-top-right-radius
- -webkit-border-bottom-right-radius
- -webkit-border-bottom-left-radius

Lamentablemente para IE8 (y anteriores) y Opera no hay reglas para conseguir el resultado, así que es necesario usar JavaScript e imágenes para conseguir el resultado.

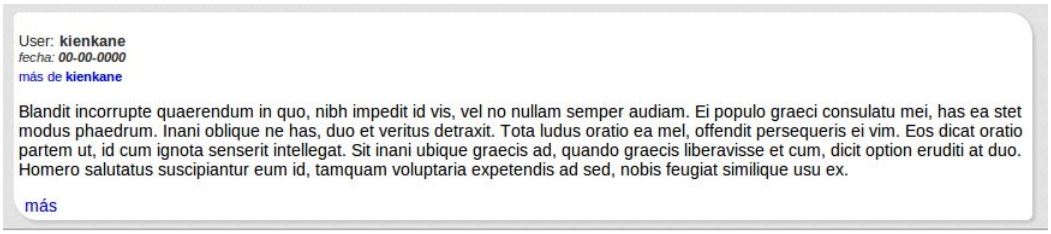


Figura 2.76: Ejemplo de esquinas redondeadas [Imagen creada por el autor de este trabajo (2011)]

Bordes sombreados

La propiedad box-shadow pertenece al estándar CSS 3 y permite agregar sombras a los elementos, siempre que definan un borde. Los navegadores que lo soportan son Firefox, Chrome, Safari y Opera, pero únicamente este último la reconoce con el nombre del estándar, para Firefox, Chrome y Safari se tiene que usar las reglas liberadas por dichos navegadores. La imagen 2.77 es un ejemplo de bordes sombreados.

Bordes sombreados	**box-shadow**
Posibles valores	((medida){3} && color){1-} medida 1 .- establece la separación vertical de arriba– abajo medida 2.- establece la separación horizontal de izquierda– derecha medida 3.- establece la difuminación. color.- establece el color de la sombra. **ej1**: div{box-shadow:8px 5px 0px #000;} desplaza la sombra 8px hacia abajo 5px a la derecha y sin difuminado, de color negro **ej2**: div{box-shadow:-8px -5px 20px #000;} desplaza la sobra 8px hacia arriba 5px a la izquierda y un difuminado de 20px color negro
Valor de inicio	none
Se aplica a todos los elementos	

Tabla 2.52: bordes con sombras

Es posible establecer más sombras a los elementos, para ello se tiene que separar cada grupo de definición por comas, como se muestra en el código 2.36.

Código 2.36: Sombras

```
1 div{
2 -moz-box-shadow: -8px -5px 20px #444, 8px 5px 20px #
      c3c3c3;
3 }
```

opacity

Propiedad que le aplica a los elementos una opacidad o semitransparencia, logrando diseños interesantes cuando se utilizan galerías o secciones de comentarios.

Para Firefox, Opera, Chrome, Safari, entre otros se utiliza la propiedad opacity y para IE se utiliza filter alpha opacity. Los valores posibles son de 0 a 1, entre más cercanos esten al cero más opaco se muestra el elemento y entre más cercano a 1 el elemento se muestra más transparente.

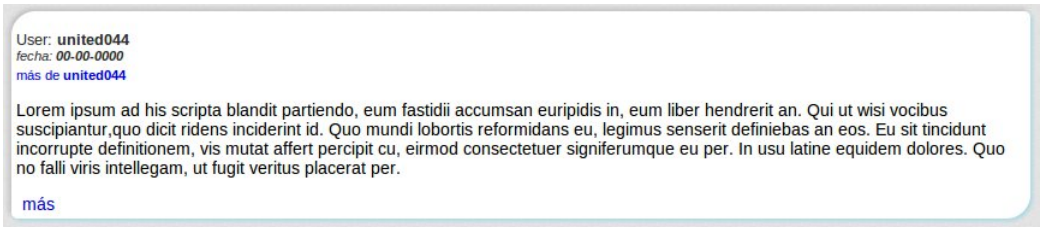


Figura 2.77: Ejemplo de bordes sombreados [Imagen creada por el autor de este trabajo (2011)]

Opacidad	**opacity o filter alpha opacity**
Posibles valores	(0.0 - 1.0) o (0 - 100) IE **ej1**: div{opacity:0.8;} muestra una semitransparencia del 80 % **ej2**: div{filter:alpha(opacity=90)} muestra una semitransparencia del 90 % versión para IE únicamente.
Valor de inicio	1
Se aplica a todos los elementos	

Tabla 2.53: Opacidad de los elementos

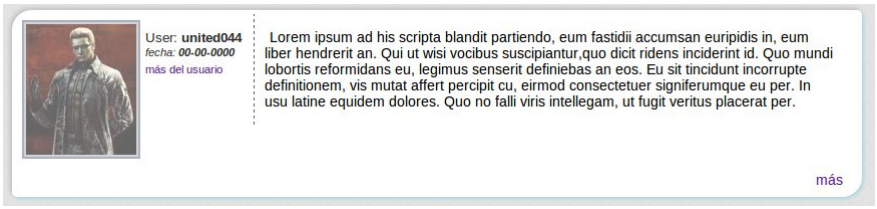


Figura 2.78: Ejemplo de opacidad al 80 % [Imagen creada por el autor de este trabajo (2011)]

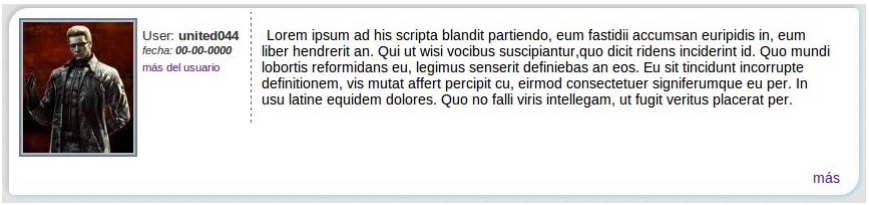


Figura 2.79: ejemplo de opacidad al 100 %

2.4. Notas de jQuery

2.4.1. Introducción a jQuery

Las aplicaciones Web son cada vez más complejas y hacen énfasis en incorporar recursos multimedia. Esta carga de contenidos y estructuras complican la prestación del servicio. La forma de establecer la comunicación entre el usuario y el servidor consiste en hacer peticiones que posteriormente se ejecutan. Este procedimiento hace que las aplicaciones complejas generen tiempos de respuesta ineficientes y sobrecargan el servidor.

Para solucionar estos impedimentos las tecnologías y sus avances se han enfocado en conseguir que las aplicaciones en línea tengan un comportamiento similar a las aplicaciones de escritorio, pero explotando los beneficios de ambas arquitecturas. Estas aplicaciones se les conoce como RIAs (Aplicaciones Ricas de Internet, Rich Internet Applications), que permiten que las acciones que ejecuta el usuario esten previamente cargadas por el navegador.

Las RIAs pueden visulizarse sobre cualquier navegador solo es necesario mantenerlo actualizado, esto permite que se pueda utilizar desde cualquier computadora con conexión a Internet sin importar del sistema operativo. Son menos propensas al contagio de virus. Tiene mejor capacidad de respuesta con el servidor ya que precarga lo necesario para evitar muchas recargas de la página y se comunica con el servidor para obtener o enviar datos.

Hay muchas herramientas y técnicas para el desarrollo de RIAs. Entre algunas de ellas se encuentra AJAX, Flex, Open Laszlo, Silverlight y JavaFX Script.

AJAX (Javascript Asíncrono y XML, Asynchronous JavaScript And XML) es nativo en los "navegadores web" y es el único "RIA framework" que puede ser encontrado por los diferentes motores de búsqueda. Fue creado en 2005 y es un conjunto de técnicas que hacen posible la creación de aplicaciones Web que se ejecutan en navegadores convencionales, sin que sea necesario recargar la página al comunicarse con el servidor, ya que realiza la comunicación como un proceso en paralelo sin que el usuario lo note, refrescando secciones de una página, trayendo datos del servidor o enviando peticiones.

JavaScript no está realmente preparada para la Web 2.0 y con ello para las RIAs. Esto radica en que el soporte de JavaScript varía del navegador Web y sus versiones, por lo que se convierte en una plataforma difícil de trabajar. Por esta razón se han creado muchas bibliotecas de JavaScript para facilitar su uso y desarrollo.

Es aquí donde entra jQuery, que es una de las bibliotecas de JavaScript más populares. Fue creada por John Resig durante sus días de universitario del Instituto Tecnológico de Rochester.

Un gran número se sitios Web de reconocimiento internacional utilizan jQuery por mencionar algunos están: Google, DELL, Bank of America, Major

League Baseball, Digg, NBC, CBS, Technorati, Mozilla.org, Wordpress y Drupal.

2.4.1.1. Capacidades y beneficios de jQuery

jQuery es Cross-Browser Los navegadores Web manejan JavaScript de forma diferente, esto es tan notable y crucial que se puede decir que no hay dos navegadores que manejen JavaScript de la misma forma. El código de JavaScript para acceder y manipular los elementos de una página varía de un navegador a otro, convirtiendo la programación de una función sencilla en una labor complicada, al tener que implementar la misma función con diferente código según el navegador, y así obtener un resultado similar en los diferentes navegadores. jQuery pone fin a esto incorporando un conjunto de funciones comunes a los navegadores, dando soporte y homogeneidad en el código sin importar el navegador.

Soporte a AJAX Una de las razones por las cuales se centró la atención en JavaScript y que ha hecho tan popular a muchas bibliotecas es AJAX. jQuery ofrece uno de los mejores soporte y manejos de estas herramientas.

Selectores en jQuery Permite acceder a cualquier elemento de una página usando selectores, tomando la misma base y principios que en los selectores de CSS, de tal modo que es fácil de aprender y utilizar.

Manejo de elementos de una página cargada En ocasiones definimos funciones en JavaScript que requieren acceder tan pronto como sea posible a los elementos de una página, pero dichas funciones se encuentran definidas en el $< head >$ y los elementos a acceder se encuentran definidos en el $< body >$ que se carga después del head, así que cuando la función en el head se ejecuta no existen elementos en html para acceder y modificar (aun no se han cargado). Al utilizar el evento onload del navegador se retrasa cualquier proceso hasta que todos los elementos de la página incluyendo las imágenes se hayan cargado completamente. jQuery permite acceder y modificar elementos de la página sin tener que esperar que se carguen todas las imágenes.

jQuery permite crear elementos HTML Es común que las buenas bibliotecas de JavaScript den soporte para crear, modificar y eliminar elementos de HTML.

Soporte a animaciones y efectos jQuery tiene una gran colección de funciones sobre animación y efectos visuales que proporcionan a sus páginas Web un estilo dinámico característico de las RIAs.

A continuación se describe brevemente los pasos de intalación de jQuery.

2.4.1.2. Instalar jQuery

jQuery es una biblioteca de Javascript por lo que su terminación es ".js".
Se puede descargar la última versión desde su página Web oficial (http://www.jquery.com), como se muestra en la figura 2.80

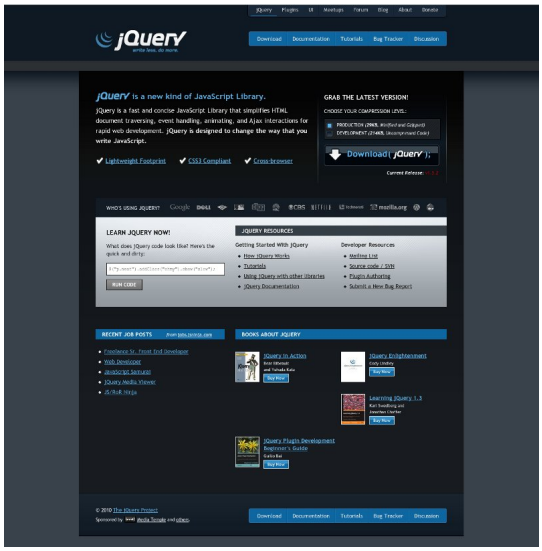


Figura 2.80: Página oficial de jquery [Imagen creada por el autor de este trabajo (2011)]

De la página se puede descargar la última versión en sus dos niveles de compresión, la versión en producción (terminación ".min.js") o la versión de desarrollo (terminación ".js") la primera tiene una organización interna que facilita su rápida carga en los navegadores pero es difícil de leer por los humanos y la segunda tiene una estructura que facilita la lectura al programador, pero al presentar tabulaciones, saltos de línea y comentarios su peso aumenta considerablemente, por tal razón, es una buena práctica de programación utilizar la versión de desarrollo cuando la aplicación Web se encuentre en implementación y utilizar la versión de producción cuando la aplicación sea liberada.

Una vez descargada la biblioteca se recomienda almacenarla en la carpeta que contiene los JavaScript, aunque la forma más sencilla (y poco recomendable) es el directorio raíz del servidor. Después se vincula con los documentos HTML que haran uso de ella. Para esto se inserta en una etiqueta $< script >$ dentro del $< head >$ en el HTML como se muestra en el código 2.37.

Código 2.37: Insertar jQuery en un documento HTML

```
1 <script type="text/javascript" src="js/jquery-1.5.2.js">
2 <\script>
```

Si no se quiere descargar se puede utilizar una versión en línea desde la página oficial de jQuery como se muestra en el código 2.38.

Código 2.38: Enlazar jQuery en línea a un documento HTML

```
<script type="text/javascript" src="http://code.jquery
.com/jquery-latest.js"><\script>
```

2.4.1.3. Objetivo

Los conocimientos y habilidades presentados en los siguientes cápitulos estan acompañados de ejemplos reales que pueden ser utilizados en cualquier página Web realizando los ajustes convenientes. Los ejercicios de las prácticas refuerzan los conocimientos adquiridos a través de problemas reales que se presentan al desarrollar páginas Web.

Este manual no garantiza que al terminar su lectura podrá implementar páginas Web usables, aplicaciones ricas de Internet (RIAs), ni le enseñará a implementar páginas Web deslumbrantes por dar alguna denotación. Tampoco se debe considerar como una API de jQuery, ya que la página oficial cuenta con una excelente API (http://api.jquery.com/). Los ejemplos y ejercicios no cubren todos los problemas con los que se enfrentan los desarrolladores Web actualmente, para esa labor son necesarias las experiencias previas y la constante actualización sobre las tecnologías, habilidades y técnicas en la materia.

El manual y su material estan enfocados en la inicilización de jQuery a través de ejemplos y ejercicios útiles que le permitan obtener experiencia y herramientas para abordar problemas futuros. Este manual es un material de apoyo a los temas abordados en el curso de Diseño de Interfaces de la Facultad de Ciencias de la UNAM junto con los diversos materiales proporcionados en la asignatura, donde se abordan criterios ergonómicos y principios de usabilidad enfocados al diseño e implementación de páginas Web usables.

2.4.2. Seleccionar Elementos

Las secciones de este capítulo presentan temas principales, los cuales se enfocan en explicar algunas formas para seleccionar y acceder a elementos de una página Web, para ejemplificar el uso de esos métodos y técnicas se acompañan con temas secundarios tales como métodos de jQuery para acceder a elementos de un arreglo, aplicar reglas de CSS, alternar clases y estilos de CSS, entre otros. Con la finalidad de avanzar y concretar ejemplos que se puedan utilizar en la vida cotidiana de un programador Web.

2.4.2.1. Seleccionar elementos por su tipo

jQuery permite seleccionar y acceder a elementos de una página para manipularlos, editar su contenido dinamicamente, aplicar o eliminar estilos de CSS, aplicarles efectos y animaciones, eliminar estructuras de HTML, entre otras acciones y procesos interesantes. Todos estos procesos, técnicas y estructuras que proporciona jQuery estan muy bien documentadas en la página oficial de jQuery (http://docs.jquery.com/Main_Page).

Al utilizar jQuery se requiere indicar al navegador que el código o ciertas partes del código hacen uso de la biblioteca jQuery, para ello se utiliza la función jquery() o su versión corta $( ), por comodidad este manual utilizará la versión corta.

En CSS existe el selector de tipo o etiqueta que permite seleccionar y aplicar ciertas reglas de estilo a todos los elementos de la página cuya etiqueta de HTML coincida con el valor del selector. Así en el siguiente ejemplo la regla de estilo, aplica a todos los párrafos de la página Web el color de fuente negro.

```
p {color:#000;}
```

De una forma muy parecida se utiliza el selector de tipo en jQuery que permite acceder y manipular un elemento o conjunto de elementos de una página cuya etiqueta coincida con la del selector. Para acceder a todos los párrafos de la página se utiliza el selector de tipo "p" a través de la siguiente instrucción $("p"). La cual devuelve el conjunto de todos los párrafos de la página almacenados como elementos de un arreglo. Aplicando el método size() se obtiene la cantidad de elementos de un arreglo y con el método alert() se muestra el resultado en una ventana de diálogo.

En el código 2.39 que es un fragmento del documento "selector_tipo.html", se usa un formulario para ejecutar la función count() que devuelve la cantidad de párrafos contenidos en el "div" con clase "result".

Código 2.39: Seleccionar por tipo

```
/*--- Código jQuery ---*/
<script type="text/javascript">
function count(){
 var total_p = $('.result p').size();
 alert("La cantidad de párrafos en result es " +total_p);
}
</script>

<!-- Código HTML -->
<div class = "result">
  <p>párrafo 1</p>
  <p>párrafo 2</p>
```

```
13      <p>párrafo 3</p>
14      <p>párrafo 4</p>
15  </div>
16
17  <form>
18  <input type="button" value="Cuenta Párrafos" onClick=
19  "count()">
20  </form>
```

2.4.2.2. Seleccionar un elemento por su ID

Similar a la selección de elementos por su tipo o etiqueta de HTML (ver apartado 2.4.2.1 Seleccionar elementos por su tipo), jQuery permite seleccionar y acceder a un elemento dado su valor de ID. La instrucción $('#id') devuelve un arreglo que contiene el elemento HTML cuyo valor de ID coincide con la del selector.

Una vez almacenado el elemento en cuestión se pueden aplicar funciones que agregan estructuras de HTML, modifican su contenido, incluyen efectos y animaciones o modifican sus reglas de estilo.

En la documentación y soporte técnico de la página oficial de jQuery, se encuentra una amplia variedad de métodos referentes al CSS. [7] Algunos de esos métodos son addClass(), removeClass() y toggleClass(). jQuery también proporciona métodos para modificar HTML y el contenido de una página, dos métodos importantes sobre el tema son text() y html().

.addClass("nombre_clase1 nombre_clase2 ... nombre_claseN") A los elementos involucrados les agrega las clases de CSS listadas en los parámetros.

.removeClass("nombre_clase1 nombre_clase2 ... nombre_claseN") Similar a addClass(), pero remueve las clases que coinciden de la lista de parámetros con las que presentan los elementos seleccionados.

.toggleClass("nombre_clase1 nombre_clase2 ... nombre_claseN") Alterna las clases de CSS (quita o agrega clases):

- Si los elementos en cuestión presentan alguna de las clases listadas, estas son removidas.
- De lo contrario las agrega.

.text( ) Método que devuelve la cadena de texto, de la concatenación del contenido de los elementos seleccionados (excluyendo el HTML). Puede ser usado para documentos XML y HTML.

[7] El sitio oficial es http://api.jquery.com/category/css/

.html( ) Método que devuelve una cadena de texto con el contenido del primer elemento que coincide con la selección.

En el código 2.40 que es un fragmento del documento "selector_id.html", se utilizan algunos métodos sobre CSS y HTML proporcionados por jQuery para trabajar con un elemento seleccionado por su ID. Se selecciona el párrafo con id "destacado" contenido en el div.result y mediante botones de formulario se aplican funciones que le agregan, remueven y alternan la clase "marcado" utilizando los métodos addClass(), removeClass() y toggleClass() de jQuery. La clase "marcado" de CSS aplica al párrafo un color de fondo amarillo. Trabajando sobre el mismo párrafo se crea una función que obtiene y despliega en un cuadro de mensaje el texto de dicho elemento.

Código 2.40: Seleccionar por ID

```
/*---- Reglas  CSS ----*/
<style type="text/css">
#container div.result p#destacado{
 font-size:18px;
 font-style:italic;
}
.marcado{
 background-color:#FFA500;
 font-style:none;
 font-variant:small-caps;padding:3px;
}
</style>
/*---- Código  jQuery ----*/
function agregar_clase(){
 $('.result p#destacado').addClass('marcado');
}
function remover_clase(){
  $('.result p#destacado').removeClass('marcado');
 }
function alternar_clase(){
  $('.result p#destacado').toggleClass('marcado');
}
function mostrar_contenido(){
  var cad = $('.result p#destacado').text();
  alert("El contenido del párrafo con id destacado
  es: \n"+cad);
}

<!-- Código HTML -->
<div class = "result">
 <p id="destacado" class="marcado">párrafo 1</p>
 <p>párrafo 2</p>
 <p>párrafo 3</p>
 <p>párrafo 4</p>
```

```
35  </div>
36  <form>
37  <input type="button" value="Agregar Clase" onClick=
38  "agregar_clase()">
39  <input type="button" value="Remover Clase" onClick=
40  "remover_clase()">
41  <input type="button" value="Alternar Clase" onClick=
42  "alternar_clase()">
43  <input type="button" value="Mostrar Contenido"
44   onClick="mostrar_contenido()">
45  </form>
```

2.4.2.3. Seleccionar por descendencia

En los ejemplos anteriores se presentó un tipo de selección muy útil y que juega un papel importante en la estructura de los ejemplos y ejercicios de este mánual de prácticas. Se trata del selector descendente, con el cual jQuery permite seleccionar uno o más elementos de un página dado su profundidad estructural tomando encuenta sus elementos predecesores. Este selector es muy específico y se puede combinar con selectores de clase, tipo, encadenamiento de selectores, etc. Lo que aumenta la especificidad del selector.

Se dice que un elemento es descendiente a otro cuando se encuentra dentro de las etiquetas de definición de otro elemento sin importar la profundidad.

Supongamos las siguientes selecciones:

1.- #container p.- Selecciona a todos los párrafos contenidos en el elemento con id container.

2.- #container div p.- Selecciona a todos los párrafos contenidos en un div que a su vez esté contenido en el elemento con id container.

En el anterior ejemplo la acción relacionada con la primera selección se aplicaría a las dos estructuras, ya que cualquiera de esos párrafos se encuentrán definidos dentro del elemento con id container, pero en la segunda selección se requiere un "div" intermediario.

Un ejemplo específico del uso de los selectores descendentes se puede ver en el código 2.41 que es un fragmento del documento "selector_descendente.html"

El ejemplo en cuestión esta formado por:

1.- Dos titulares un "h1" y "h2" en el mismo nivel estructural.

2.- Un "div" que contiene 3 párrafos con contenido "Lorem ipsum" inmediatos al titular "h1".

3.- Dos párrafos con contenido "Lorem ipsum" inmediatos al titular "h2".

4.- Se implementan dos clases de estilo CSS:

4.1.- resaltar.- pone un color de fondo #FFFF9A y un color de texto #4D4D4D.

4.2.- literario.- pone un color de fondo #ADD8E6, el texto justificado en itálica y una separación entre líneas de 1.2em.

5.- Se implementan dos funciones en jQuery con sus mecanismos de acción:

5.1.- resaltar().- Que utiliza el método toggleClass() para alternar la clase 'resaltará los párrafos contenidos en un "div" que a su vez esté contenido en el "div" con clase "result".

5.2.- aplica_p().- Similar a resaltar(), pero aplica la clase "literario" a todos los párrafos contenidos en un "div" con clase "result".

Código 2.41: Seleccionar por descendencia

```
/*---- Reglas CSS ----*/
#container div.result p#destacado {font-size:18px;
font-style:italic;}
.marcado{background-color:#FFA500; font-style:none;
font-variant:small-caps;padding:3px;}

/*--- Código  jQuery ---*/
function resaltar(){
 $('.result div p').toggleClass('resaltar');
}
function aplica_p(){
 $('.result p').toggleClass('literario');
}
```

En el código 2.41 la función aplica_p( ) le aplica a todos los párrafos contenidos en el div.result la clase "literario" y la función resaltar( ) aplica la clase homónima de CSS únicamente a los párrafos contenidos en un "div" descendiente al div.result.

Anteriormente se comentó que jQuery dispone de varios métodos interesantes sobre CSS, además de addClass(), removeClass() y toggleClass(), se encuentra hasClass().

- **.hasClass("nombre_clase1 nombre_clase2 ... nombre_claseN")** Método que devuelve True si los elementos seleccionados presentan al menos una de las clases listadas en los argumentos del método y False en otro caso.

2.4.2.4. Seleccionar por su estilo de CSS

jQuery permite seleccionar elementos HTML a través de las clases de CSS que presentan y en combinación con la selección descendente se obtiene mayor especificidad. Por ejemplo si una página cuenta con varios párrafos y algunos de ellos presentan la clase "destacado" de CSS que les aplica un estilo diferente para resaltar sus contenidos se puede acceder a esos párrafos mediante:

```
$('.result p.destacado')
```

Si estan contenidos en un "div", se puede mejorar la especificidad de la selección con:

```
$('.result div p.destacado')
```

Un ejemplo específico de la selección de elementos a través de sus clases se puede ver en el código 2.42 que es un fragmento del documento "introduccion_jquery/ selector_style.html". La estructura es similar al ejemplo anterior, pero con un párrafo extra. Los puntos importantes son:

1.- Se implementan dos clases de estilo CSS:

   1.1.- resaltar.- Pone un color de fondo #FFFF9A, un color de texto #4D4D4D y un borde con espesor de 1px de color #4D4D4D y estilo punteado.

   1.2.- destacado.- Pone un color de fondo #ADD8E6, el texto justificado en itálica, una separación entre líneas de 1.2em y un relleno de 5px.

2.- Se implementa una función en jQuery con su mecanismo de acción:

   2.1.- resaltar().- Que utiliza el método toggleClass() para alternar la clase "resaltar" a los párrafos que presentan la clase "destacado" contenidos en el "div" con clase "result".

Código 2.42: Seleccionar por clase

```
/*---- Reglas CSS ----*/
.destacado{background:#ADD8E6;text-align:justify;
font-style:italic;line-height: 1.2em;padding:5px;}
.resaltar{background:#FFFF9A;color:#4D4D4D;
border:1px dotted #4D4D4D;}

/*---- Código  jQuery ----*/
function resaltar(){
```

```
9  $(‘.result p.destacado’).toggleClass(‘resaltar’);
10 }
```

2.4.2.5. Seleccionar el primer y último elemento

jQuery permite acceder al primer y último elemento de un conjunto de elementos, usando los selectores posicionales first y last respectivamente. Así dado el conjunto de todos los párrafos se puede acceder al primer párrafo usando $(‘p:first’) y al último usando $(‘p:last’).

Una instrucción muy útil que proporciona jQuery es el método css(), que recibe como parámetro una regla de estilo de CSS que es aplicada al elemento o elementos coincidentes al selector.

- **.css({propiedad:“valor”, ..., propiedadN: “valorN”})** Aplica la regla (reglas) de estilo proporcionada como parámetro a los elementos seleccionados.

En el siguiente ejemplo se aplica un color de fondo #E6ECEF al primer párrafo descendiente a un contenedor con clase “result”.

```
1  $(‘.result p:first’).css({‘background’:‘#E6ECEF’});
```

En el segundo ejemplo el primer párrafo descendiente a un contenedor con la clase “result”, recibe un color de fondo #DEA7FF y un borde de color #6905A8 con espesor de 1px.

```
1  $(‘.result p:first’).css({background:‘#DEA7FF’,
2  ‘border’:‘1px solid #6905A8’});
```

Un dato importante a tomar en cuenta al momento de generar las reglas de estilo es que para jQuery las propiedades con nombres largos separados con guiones se deben empaquetar entre comillas (‘ ’, “ ”) o quitar los guiones y juntar las palabras del nombre cambiando a mayúsculas la primera letra de cada palabra después de la primera palabra, así las siguientes instrucciones obtienen el mismo resultado.

```
1  $(‘.result p:first’).css({‘background-color’:‘#E6ECEF’});
2  $(‘.result p:first’).css({backgroundColor:‘#E6ECEF’});
```

Un ejemplo específico se encuentra en el código 2.43 que es un fragmento del documento “selector_posicional.html”. Algunas especificaciones del ejemplo:

1.- Un titular “h1”.

2.- Tres párrafos con contenido “Lorem ipsum” inmediatos al titular “h1”.

3.- Se implementan dos funciones en jQuery con sus mecanismos de acción:

3.1.- aplica_firstp().- Que utiliza el método css() para aplicar el siguiente estilo de CSS al primer párrafo descendiente al contenedor con clase “result”:

1. Color de fondo #DEA7FF.
2. Borde de espesor 1px con estilo sólido y color #6905A8.

3.2.- aplica_lastp().- Le aplica al último párrafo descendiente del contenedor con clase “resul” el siguiente estilo:

1. Color de fondo #FFFF79.
2. Borde de espesor 1px con estilo sólido y color #FFA500.

Código 2.43: Seleccionar el primer y último elemento

```
/*---- Código  jQuery ----*/
< script type="text/javascript">
function aplica_firstp(){
$(‘.result p:first’).css({background:‘#DEA7FF’,
border:‘1px solid #6905A8’});
}
function aplica_lastp(){
$(‘.result p:last’).css({background:‘#FFFF79’,
border:‘1px solid #FFA500’});
}
</script>
```

2.4.2.6. Seleccionar dado un conjunto de elementos

Al seleccionar elementos con jQuery a través de la instrucción $() se crea un arreglo con todos los elementos coincidentes al selector proporcionado como parámetro, así $(‘p’) genera un arreglo con todos los párrafos de la página. Esta forma de manejar los elementos seleccionados por parte de jQuery es muy útil y permite acceder a elementos particulares de ese conjunto.

En el apartado 2.4.2.5 Uso de :first y :last, se indicó la forma para acceder al primer y último elemento utilizando el selector posicional, pero no es la única forma para conseguir eso. jQuery permite seleccionar un elemento accediendo por su índice en el arreglo de los elementos coincidentes utilizando el selector :eq() o el método .eq():

:eq(indice) Selecciona un elemento por su índice ocupado en el arreglo de los elementos coincidentes. Asignando el índice cero al primer elemento.

Como selector. $("p:eq(0)") selecciona al primer párrafo dado el arreglo obtenido de todos los párrafos de la página, resultado similar al conseguido con $("p:first-child").

Como método $("p").eq(0) mismo resultado que en el anterior ejemplo.

La principal diferencia entre el selector y el metodo eq(), es que el método acepta índices negativos, considerando el arreglo como una línea continúa, donde el último y el primer elemento del arreglo son consecutivos.

.eq(-indice) donde "-indice" indica la posición del elemento contando desde el último elemento del conjunto hacia atrás.

En el código 2.44 se aplica un color de fondo #DEA7FF al tercer "li" de la lista, ya que se cuentan dos lugares desde el último elemento hacia atrás.

Código 2.44: Uso método eq(index)

```
/*---- Código  jQuery ----*/
$('.result p').eq(-2).css({background:'#DEA7FF'});
<!-- Código HTML -->
<ul>
  <li>Elemento 1<\li>
  <li>Elemento 2<\li>
  <li>Elemento 3<\li>
  <li>Elemento 4<\li>
</ul>
```

Es recomendable usar el método .eq() ya que el selector :eq() es una extensión proporcionada por jQuery que no forma parte de las especificaciones de CSS.

Una vez abordado varios temas referente a las formas más comunes y sencillas para seleccionar elementos y acceder a elementos particulares del conjunto, jQuery incorpora métodos que aplican efectos visuales a los elementos seleccionados.

jQuery permite ocultar y mostrar elementos de la página a través del uso de hide() y show() respectivamente. Estos métodos únicamente desaparecen y aparecen elementos en una página, existen métodos que tienen el mismo resultado, pero presentan un efecto visual de deslizamiento como son los casos de slideUp(), slideDown() y slideToggle().

.hide(duracion, foo) Sin parámetros oculta los elementos seleccionados de forma inmediata. Al presentar el parámetro de duración, se demora la cantidad de tiempo indicada para ocultar el elemento, presentando un

efecto visual, en el cual el ancho y la altura se reducen gradualmente. Para indicar la duración se pueden usar las palabras reservadas "slow" y "fast" que equivalen a 600 y 200 milisegundos respectivamente o usar una cantidad numérica. Si se presenta el parámetro foo que es una función, se pueden encadenar efectos y procedimientos referentes al elemento seleccionado junto con el efecto de ocultar.

.show(duracion, foo) Efecto opuesto a hide(). También permite el uso de los valores "slow" y "fast" para mostrar los elementos seleccionados.

.slideUp(duracion, foo) Sin parámetros desliza los elementos seleccionados rápidamente hacia arriba con una duración de 400 milisegundos. Al presentar el parámetro de duración, se demora la cantidad de tiempo indicada para deslizar y ocultar el elemento. Para indicar la duración se pueden usar las palabras reservadas "slow" y "fast", antes mencionadas.

.slideDown(duracion, foo) Efecto opuesto a slideUp().

.slideToggle(duracion, foo) Alterna el deslizamiento de los elementos seleccionados, si el elemento esta oculto aplica slideDown() de lo contrario aplica slideUp().

Un ejemplo específico de la selección de elementos a través del índice que ocupa en el arreglo de los elementos coincidentes dada la selección, se encuentra en el código 2.45 que es un fragmento del documento "selector_posicion.html". Los puntos importantes del ejemplo son:

1.- Un titular "h1".

2.- Cuatro párrafos con contenido "Lorem ipsum" inmediatos al titular.

3.- Se implementan cinco funciones en jQuery con sus mecanismos de acción:

- 3.1.- oculta2().- Utiliza el método hide() con una duración de 300 milisegundos para ocultar el segundo párrafo.
- 3.2.- muestra2().- Utiliza el método show() con una duración de 300 milisegundos para mostrar el segundo párrafo.
- 3.3.- slideUp2().- Utiliza el método slideUp() con una duración de 300 milisegundos para deslizar el segundo párrafo y ocultarlo.
- 3.4.- slideDown2().- Utiliza el método slideDown() con una duración de 300 milisegundos para deslizar el segundo párrafo y mostrarlo.
- 3.5.- slideToggle2().- Utiliza el método slideToggle() que alterna los métodos slideDown() y slideUp(), con una duración de 300 milisegundos.

Código 2.45: Ejemplo con eq(index)

```
/*---- Código jQuery  ----*/
<script type="text/javascript">
function oculta2(){
  $('.result p').eq(1).hide(300);
}
function muestra2(){
  $('.result p').eq(1).show(300);
}
function slideUp2(){
  $('.result p').eq(1).slideUp(300);
}
function slideDown2(){
  $('.result p').eq(1).slideDown(300);
}
function slideToggle2(){
  $('.result p').eq(1).slideToggle(300);
}
</script>
```

2.4.2.7. Práctica 01

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.4.2 Seleccionar Elementos, en cuyos temas se tratan las formas básicas y más conocidas para seleccionar y acceder a elementos de una página Web. Así como algunos métodos proporcionados por jQuery para agregar y manipular las reglas de estilo de CSS, además se ejemplifican algunos efectos visuales tales como: ocultar y mostrar elementos de una página Web, desplegar para ocultar y mostrar elementos, contar elementos y desplegar cuadros de diálogo, entre otros.

Ejercicios Los documentos HTML de esta práctica se localizan en jQuery/ejercicios/introduccion_jquery.

1.- Además del método size(). ¿Qué proporciona jQuery para contar los elementos de una página Web y cómo funciona? Ejemplifica tu respuesta.[8]

2.- En el documento "selector_tipo.html" implementar las funciones con sus respectivos mecanismos de acción para contar los siguientes elementos descendientes al div.result:

2.1.- Los párrafos.

[8] Se pueden utilizar las estructuras base de los ejemplos HTML ubicados en: jQuery/ejemplos

2.2.- Los divs.

2.3.- Los titulares.

2.4.- Los elementos de HTML.

3.- ¿Cuáles son las diferencias entre los métodos text( ) y html( ) de jQuery? Ejemplifica el uso de html( ).

4.- Investiga otros métodos útiles sobre CSS que proporciona jQuery y construye dos funciones donde se apliquen.

5.- En el documento "selector_id.html" implementar lo que se pide con sus respectivos mecanismos de acción (botón de acción):

5.1.- Asignar el id frsElement al primer párrafo descendiente y aplicar un estilo que contenga:

1. Padding.
2. Margin.
3. Esquinas redondeadas.
4. Contorno sombreado.

5.2.- Implementar las clases "p_style" y "principal", que serán aplicadas a #frsElement.

1. p_style.- Aplica un fondo de color #58127F, con la imagen "css/images/fondo8.png"
2. principal.- Similar a la anterior pero usa el color de fondo #DF4949.

5.3.- Implementar funciones en jQuery aplicadas a #frsElement.

1. Función que agrega la clase "p_style".
2. Función que remueva la clase "p_style" o "principal".
3. Función que alterna las dos clases.

5.4.- Reacomoda los argumentos de las funciones anteriores para cambiar los efectos conseguidos al utilizar las funciones que agregan, remueven y alternan las clases de CSS.

5.5.- Función en jQuery que cuente la cantidad de caracteres del párrafo #frsElement y lo muestre en un cuadro de diálogo.

5.6.- Función en jQuery que cuente la cantidad de palabras del párrafo #frsElement y lo muestre en un cuadro de diálogo.

6.- En el documento "selector_descendente.html" implementar lo que se pide con sus respectivos mecanismos de acción (botón de acción):

6.1.- Implementar las clases de estilo de CSS en el documento "css/selector_descendente.css":

1. resaltar.- Aplica un color de fondo #FFFF9A y un color de texto #4D4D4D.
2. div_span.- Aplica un color de letra #1F1F9A y estilo de letra itálica.
3. result_a.- Aplica un color de letra #7B24B1.
4. encuadrar_resaltar.- Aplica un borde con espesor de 1px y de color #FFA500.

6.2.- Implementar las funciones en jQuery:

1. Función que alterna la clase "resaltar" a los párrafos contenidos en un "div" que a su vez este contenido en el "div" con clase "result".
2. Función que alterna la clase "div_span" a todos los "span" contenidos en los párrafos descendientes a un "div" que descienda a su vez del "div" con clase "result".
3. Función que alterna la clase "result_a" a todos los links contenidos en los párrafos con descendencia inmediata al "div" con clase "result".
4. Función que agrega la clase 'encuadrar_resaltará los párrafos contenidos en el div con clase 'result'que presentan la clase 'resaltar', cuando ocurra eso desplegar un cuadro de diálogo con un mensaje representativo. Si los párrafos no presentan la clase "resaltar" y el usuario da clic en el botón de acción se retira la clase "encuadrar_resaltar" y se despliega un mensaje representativo.

7.- En el documento "selector_style.html" implementar lo que se pide:

7.1.- Las siguientes clases de estilo de CSS en el documento "css/selector_style.css":

1. destacado.- Clase que aplica un color de fondo #ADD8E6, el texto justificado en itálica, una separación entre líneas de 1.2em y un relleno de 5px.
2. marcar.- Clase que aplica un color de fondo #FFFF9A, un color de texto #4D4D4D y un borde con espesor de 1px de color #4D4D4D y estilo punteado.
3. resaltar.- Clase que aplica un color de letra #4D4D4D, un borde con espesor de 1px de color #4D4D4D y estilo punteado, esquinas redondeadas de 5px y una sombra con inclinación inferior derecha de 5px de color #63777E.
4. A los párrafos que presenten las clases "destacado" y "marcar" se les aplica un color de fondo #55FFFE.

5. A los párrafos que presenten las tres clases anteriores se les aplica un color de fondo #FFACB9.
6. A los párrafos que presentan las clases "destacado" y "resaltar" y sean inmediatos a un "div" que descienda del "div" con clase "result", se les aplica un color de fondo #FFA666.
7. A los párrafos que presentan las clases "destacado" y "resaltar" y que desciendan del "div" con clase "result" se les aplica un color de fondo #ADFFAD.

7.2.- Implementar máximo dos funciones en jQuery con sus mecanismos de acción para conseguir:

1. Función que alterna la clase "marcar" a los párrafos con clase "destacado" que desciendan de un "div" que a su vez descienda del "div" con clase "result".
2. Función que alterna la clase "resaltar" a los párrafos con clase "destacado" que desciendan del div con clase "result".

8.- En el documento "selector_posicional.html" implementar lo que se pide:

8.1.- Implementar tres funciones en jQuery con sus mecanismos de acción para conseguir:

1. Función que aplica el siguiente estilo de CSS al primer párrafo contenido en un "div" descendiente al contenedor con clase "result":
   a. Color de fondo #A2D0DF.
   b. Borde de espesor 1px con estilo sólido y color #407485.
2. Función que le aplica al último párrafo contenido en un "div" descendiente al contenedor con clase "result" el siguiente estilo:
   a. Color de fondo #FFFF79.
   b. Borde de espesor 1px con estilo sólido y color #FFA500.
3. Función que le aplica el siguiente estilo de CSS a todos los primeros párrafos contenidos en los "div" descendientes al contenedor con clase "result":
   a. Color de fondo #DEA7FF.
   b. Borde de espesor 1px con estilo sólido y color #6905A8.

9.- ¿Cuáles son la diferencias entre :first y first-child?

10.- En el documento "selector_posicion.html" implementar lo que se pide:

10.1.- Cuatro funciones en jQuery con sus mecanismos de acción para conseguir:

1. Función que le aplica al primer link un color de #32327D y le cambia a mayúsculas la primera letra de cada palabra.
2. Función que le aplica al último link un color de #B75151 y le cambia a mayúsculas la primera letra de cada palabra.
3. Función que le aplica a los links intermedios un color de #490175 y les cambia a mayúsculas la primera letra de cada palabra.
4. Función en la cual el último párrafo se desliza y se oculta lentamente e inmediatamente después reaparece deslizandose con velocidad media junto con un color de fondo de #ADD8E6, un relleno de 5px y un borde de estilo sólido de espesor 1px y color #255767.

2.4.3. Selectores Avanzados

Siguiendo la metodología de trabajo presentada en el capítulo 2.4.2 Seleccionar Elementos, en el actual capítulo se abordan formas avanzadas para seleccionar y acceder a elementos de una página Web, por lo que se recomienda comprender y ejercitar los temas cubiertos en el capítulo anterior. Estos temas se acompañan con métodos secundarios que permiten: agregar contenido HTML a elementos de la página, agregar elementos HTML a la página, algunos manejos de eventos, contar elementos, alternar clases, agregar estilos CSS, entre otros.

2.4.3.1. Página en estado Ready

jQuery permite ejecutar un código cuando todos los elementos de la página hayan sido cargados, se dice que la página se encuentra en estado "Ready". Esta función es mejor que "onload" proporcionada por JavaScript del navegador, la cual se ejecuta sólo después de que se hayan cargado todos los elementos de la página incluyendo las imágenes. La función ready() de jQuery garantiza la ejecución del código después de que el DOM [9] haya sido cargado.

Cuando el código que se ejecuta modifica el valor de las propiedades de estilo CSS o incluye nuevas hojas de estilo o funciones de JavaScript es importante enlazarlas antes de la ejecución del código.

La sintaxis de la función ready() de jQuery es:

```
1 $(document).ready(function(){
2   //El código a ejecutar
3 });
```

Esta función también cuenta con una sintaxis corta que será la utilizada en el manual.

[9] Para más información sobre el DOM de una página revisar el apartado 2.4.6.1 DOM.

```
$(function(){
  //El código a ejecutar
});
```

Esta función permite que un código acceda a elementos y contenidos de una página tan pronto como esta haya sido cargada, sea para aplicar un estilo de CSS, una operación de cadenas, una operación de arreglos o una operación matemática, desplegar cuadros de diálogo, alternar clases o aplicar efectos.

En el código 2.46 se accede a todos los párrafos de la página tan pronto como sea posible (cuando el DOM haya sido cargado) y les aplica la clase de CSS "resaltar".

Código 2.46: Agregar una clase

```
/*---- Reglas CSS ----*/
.resaltar{background:#FFFF9A;color:#4D4D4D;}

/*-- Código jQuery --*/
$(function(){
  $('p').addClass('resaltar');
});
```

jQuery permite reemplazar el contenido HTML de un elemento de la página con uno nuevo, también es posible agregar nuevos elementos HTML a la página. Para ello se usa el siguiente método:

.html(htmlString) Método que reemplaza el contenido HTML de los elementos coincidentes con la cadena de texto "htmlString" proporcionada como parámetro. Este método no esta disponible para documentos XML.

El siguiente ejemplo utiliza la función .ready( ) y el método .html( ) para agregar un nuevo contenido HTML al primer párrafo descendiente al "div" con clase "result", además agrega la clase "literario" de CSS a los párrafos contenidos en un "div" descendiente al "div" con clase "result". El código del ejemplo se localiza en "ready.html".

En el cuadro de código 2.47 se muestra la función en jQuery del ejemplo "ready.html" y su resultado en la figura 2.81.

Código 2.47: ready.html

```
/*--- Código en jQuery ----*/
$(function(){
 $('.result div p').addClass('literario');
```

```
4    $('.result p').eq(0).html("< b>Nuevo texto para el
         primer párrafo < /b>");
5  });
6  function resaltar(){
7    $('.result p').toggleClass('resaltar literario');
8  }
```

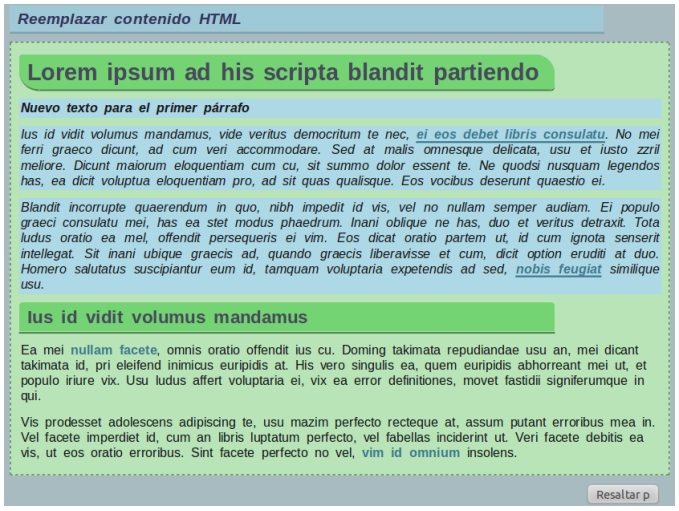


Figura 2.81: Resultado del ejemplo ready.html [Imagen creada por el autor de este trabajo (2011)]

2.4.3.2. Manipular el DOM

jQuery prorciona métodos que permiten manipular el DOM de una página. Algunos de esos métodos cambian únicamente los atributos de los elementos, otros modifican, eliminan y agregan propiedades de CSS. Otros modifican, eliminan, copian e insertan elementos o grupos de elementos HTML en la página. En esta categoría también se encuentran; attr(), html(), y val() que permiten acceder a información del elemento porporcionada por el DOM, para después utilizarla[10].

Agregar contenido HTML

Algunos métodos pertenecientes a la categoría de manipular el DOM de la página permiten agregar contenidos HTML, especificados como parámetros del método. Estos métodos son append() y appendTo() cuyas firmas son las siguientes:

[10] En la sección 2.4.6.1 Interacción DOM con HTML se muestran los métodos que proporciona DOM para realizar algunas de las tareas antes mencionadas.

.append(contenido)

.appendTo(contenido)

Donde "contenido" puede ser: un elemento DOM, una cadena de HTML o un objeto jQuery. También acepta un arreglo de elementos DOM, de cadenas de HTML o de objetos jQuery.

Estos métodos desempeñan la misma tarea, pero la principal diferencia radica en la posición en la cual se inserta el contenido. En append() el contenido se inserta el final de los elementos seleccionados, mientras que en appendTo() el contenido se inserta al inicio.

En el cuadro de código 2.48 se muestra el uso de append() para insertar una cadena de texto HTML y un objeto jQuery dentro de otros elementos de la página.

Código 2.48: Uso de append()

```
/*---- Código jQuery ----*/
$('#container div').append('<p>Párrafo de prueba</p>');
$('#container').append($('h2'));

<!-- Código HTML -->
<div id="container">
  <h2> Contenido </h2>
  <div>Lorem Ipsum</div>
  <div>Blandit incorrupte</div>
</div>
```

La primera instrucción agrega un párrafo al final de cada "div" descendiente al elemento con id "container", mientras que la segunda instrucción desplaza hacia el final del "container" la posición del "h2".

jQuery permite usar append() con otros métodos para agregar un contenido de texto a los elementos seleccionados. Así en el cuadro de código 2.49 se muestra el uso de document.createTextNode("str"), que se incluye como parámetro del método append( ) para agregar el texto "str" a los divs descendientes del elemento con id "container". Un ejemplo más elaborado de estas funciones se localiza en DOM.html.

Código 2.49: Uso de .createTextNode()

```
/*---- Código jQuery ----*/
$('#container div').append(document.createTextNode('Texto
    muestra'));

<!-- Código HTML -->
<div id="container">
  <h2> Contenido </h2>
```

```
7     <div>Lorem Ipsum</div>
8     <div>Blandit incorrupte</div>
9   </div>
```

Agregar contenido HTML en una posición

jQuery proporciona otros métodos parecidos a append() para insertar elementos HTML, objetos jQuery o elementos DOM a la página. Estos métodos y sus respectivas firmas son:

.before(contenido)

.after(contenido)

Donde el parámetro determina el contenido a agregar en la página y el método determina la posición. En before() el contenido se agrega antes de cada elemento seleccionado y en after() después de cada elemento seleccionado.

En el cuadro de código 2.50 la primera instrucción agrega un titular principal antes [11] de los elementos con clase "section" descendientes al #container y en la segunda instrucción se agrega un link después del último párrafo descendiente a un elemento con clase "section", el resultado se muestra en el cuadro de código 2.51.

Código 2.50: Uso de before() y after()

```
1  /*---- Código jQuery ----*/
2  $('#container .section').before(<h1>Título principal.
3  </h1>);
4  $('#container .section p:last-child).after(<a>Leer
5  más</a>);
6
7  <!-- Código HTML -->
8  <div id="container">
9    <div class="section">
10     <h2> Contenido </h2>
11     <p>Lorem Ipsum</p>
12     <p>Blandit incorrupte</p>
13   </div>
14 </div>
```

Código 2.51: Resultado del uso de before y after

```
1  <!-- Código HTML -->
2  <div id="container">
```

[11] Se refiere al orden de definición en el HTML.

```
3     <h1>Título principal.</h1>
4     <div class="section">
5       <h2> Contenido </h2>
6       <p>Lorem Ipsum</p>
7       <p>Blandit incorrupte</p>
8       <a>Leer más</a>
9     </div>
10  </div>
```

Al igual que con el método append() se puede utilizar la instrucción createTextNode("str") (vista en el cuadro de código 2.49) para insertar una cadena de texto antes o después de los elementos seleccionados, ver ejemplo en el cuadro de código 2.52 y su resultado en el 2.53.

Código 2.52: before() y createTextNode()

```
1  /*---- Código jQuery ----*/
2  $('#container .section p:first-child).before(document.
       createTextNode('Comienzan párrafos:'));
3
4  <!-- Código HTML -->
5  <div id="container">
6    <div class="section">
7      <h2> Contenido </h2>
8      <p>Lorem Ipsum</p>
9      <p>Blandit incorrupte</p>
10   </div>
11 </div>
```

Código 2.53: Resultado de before y createTextNode

```
1  <!-- Código HTML -->
2  <div id="container">
3    <div class="section">
4      <h2> Contenido </h2>
5      Comienzan párrafos:
6      <p>Lorem Ipsum</p>
7      <p>Blandit incorrupte</p>
8    </div>
9  </div>
```

La función each() proporcionada por jQuery permite iterar una función sobre los elementos seleccionados, es decir, aplica a cada elemento la misma función.

.each(function(indice, elemento))

La función each() comienza a recorrer el arreglo de los elementos seleccionados iniciando desde el índice cero y aplicando la función a cada elemento, para ello hace uso de la palabra reservada "this", que representa al elemento actual en el que se encuentra el apuntador del método al recorrer el arreglo. Un ejemplo de su uso se localiza en el cuadro de código 2.54.

jQuery permite registrar las acciones o eventos del usuario al interactuar con la página, estos métodos pertenecen a la categoría "events". Dentro de la cual se encuentra .click(). Este método enlaza una función o una acción al evento clic, que es la acción de posicionar el mouse sobre un elemento y dar clic sobre él, todos los elemento HTML pueden recibir este evento.

.click(foo) .- El parámetro "foo" es una función que se aplica a cada uno de los elementos seleccionados una vez se que haya completado el evento.

Para que la acción asociada tenga efecto se tiene que cumplir (completar el evento):

1.- Se posicione el mouse sobre la área del elemento en cuestión.

2.- Se presione el botón del mouse y se suelte sobre la área del elemento.

En el ejemplo 2.54 al dar clic sobre el "span" se aplica a cada elemento "li" un estilo de CSS, en el cual se pone en itálica el texto.

Código 2.54: Uso de each()

```
/*---- Código jQuery ----*/
$('span').click(function(){
  $('li').each(function(){
    $(this).css("font-style","italic");
  });
});

<!-- Código HTML -->
<span>Da clic aquí para cambiar la lista </span>
<li>Ea mei nullam facete</li>
<li>Ea mei nullam facete</li>
<li>Ea mei nullam facete</li>
```

2.4.3.3. Uso de nth-child

jQuery permite seleccionar al primer y último elemento del conjunto de los elementos seleccionados (visto en la sección 2.4.2.5 Uso de :first y :last) o seleccionar elementos dada la posición que ocupan en el arreglo (visto en la sección 2.4.2.6 seleccionar con eq()). Además de esos mecanismos jQuery proporciona el selector :nth-child() que permite seleccionar al n-ésimo elemento.

:nth-child(n) Donde n indica la posición en el arreglo del elemento a seleccionar, es decir, el n-ésimo elemento del arreglo.

En el cuadro de código 2.55 la instrucción en jQuery agrega un estilo de texto en itálica al primer párrafo descendiente al "div" con clase "section". En esta pseudo-clase el arreglo comienza a contar desde 1 a diferencia del selector :eq() que comienza en 0. Para conseguir el mismo resultado se utiliza :eq(0).

Código 2.55: Ejemplo de nth-child()

```
/*---- Código jQuery ----*/
$('.section p:nth-child(1)').css("font-style","italic");
$('.section p:eq(0)').css("font-style","italic");

<!-- Código HTML -->
<div class="section">
  <p>Lorem Ipsum</p>
  <p>Ea mei nullam facete</p>
  <p>Blandit incorrupte</p>
</div>
```

Otra característica interesante que presenta la pseudo-clase es que permite obtener los elementos del arreglo a través de una expresión de la forma (xn+a) donde n es el índice del arreglo comenzando en cero y "x" y "a" números que forman una expresión matemática para calcular los elementos del arreglo de forma precisa. En el cuadro de código 2.56 :nth-child(2n) selecciona todos los elementos en el arreglo cuya posición sea múltiplo de 2.

Código 2.56: Ejemplo de nth-child(2n)

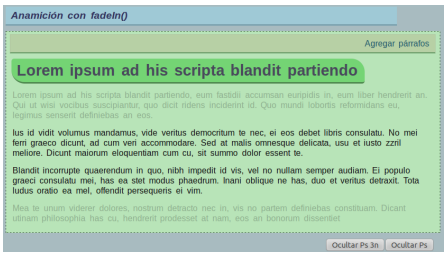
```
/*---- Código jQuery ----*/
$('.section p:nth-child(2n)').css("font-style","italic");

<!-- Código HTML -->
<div class="section">
  <p>Lorem Ipsum</p>
  <p>Ea mei nullam facete</p>
  <p>Blandit incorrupte</p>
  <p>Ea mei Blandit</p>
</div>
```

jQuery proporciona métodos que aplican animaciones a los elementos de la página, algunos métodos se localizan en la categoría de efectos (effects) en la página oficial. Dentro de la categoría se encuentran métodos que ajustan o utilizan la opacidad para mostrar u ocultar los elementos seleccionados, a estos métodos se les llaman "fading" (o desvanecer) de los cuales se encuentran fadeIn(), fadeOut(), fadeTo() y fadeToggle().

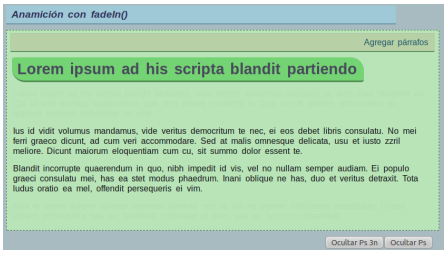
.fadeIn(duracion, foo)

Donde el primer parámetro determina la duración de la animación en milisegundos, cuanto mayor sea este número la animación será más lenta y viceversa. También acepta las cadenas de texto "fast" y "slow" las cuales representan 200 y 600 milisegundos respectivamente. Sino se específica una cadena o un valor el método utiliza un valor de 400 milisegundos. El parámetro "foo" es una función que se ejecuta a cada uno de los elementos seleccionados cuando la animación se ha completado.

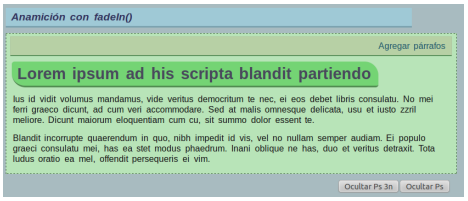
Un ejemplo específico del uso de nth-child() y de algunos métodos fading se localiza en "nth-child.html". Además se utiliza el evento click() para ejecutar una función que muestra a los párrafos inmediatos al "div" con clase "result" y a través de botones de formulario y del método fadeOut() se ocultan los párrafos. El resultado de aplicar un fadeOut() en una página se muestra en la figura 2.82.



(a) Desvanecimiento.



(b) Se ocultan los elementos



(c) Resultado final.

Figura 2.82: Ocultar párrafos con fadeOut(). [Imágenes creadas por el autor de este trabajo (2011)]

2.4.3.4. Seleccionar elementos con un texto específico

jQuery permite seleccionar elementos que contienen un texto específico. El texto puede aparecer directamente en los elementos seleccionados, en sus elementos descendientes o en su combinación. El texto dentro del método puede aparecer entrecomillado o sin comillas y el resultado debe ser el mismo.

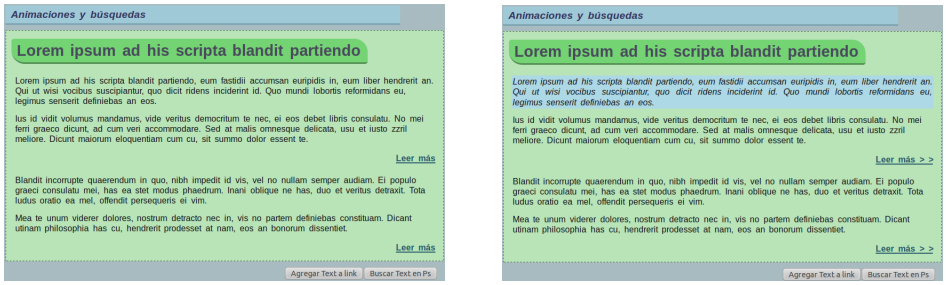
En el cuadro de código 2.57 se muestra un ejemplo del uso de :contains("str"), en el cual se seleccionan párrafos descendientes al "div" con clase

"section" que contienen el texto "Lorem ipsum" y les agrega una regla de estilo CSS que pone el texto en justificado y en itálica. En el ejemplo únicamente el primer párrafo coincide con la búsqueda.

Código 2.57: Ejemplo de :contains('str')

```
/*---- Código jQuery ----*/
 $(".section p:contains('Lorem ipsum')").css({textAlign:
"justify",fontStyle:"italic"});

<!-- Código HTML -->
<div class="section">
<p>Lorem ipsum ad his scripta blandit partiendo</p>
<p>Ius id vidit volumus mandamus, vide veritus
democritum.</p>
<p>Mea te unum viderer dolores, nostrum detracto nec
 in.</p>
</div>
```

Un ejemplo más elaborado se localiza en el documento "contains.html", en el cual se utilizan los métodos addClass() y append() para buscar un texto específico entre los párrafos y agregarles una clase de estilo, también agregar un texto al final de ciertos links. Los resultados de este ejemplo se muestran en la colección 2.83.



(a) Página original. (b) Después de aplicar los métodos.

Figura 2.83: resultados en contains.html [Imágenes creadas por el autor de este trabajo (2011)]

2.4.3.5. Seleccionar por atributo

Las especificaciones de CSS permiten seleccionar elementos por sus atributos y sus valores, pero no son soportados por todos los navegadores. jQuery proporciona selectores por atributo los cuales permiten seleccionar elementos HTML dado el nombre o nombres de ciertos atributos con o sin sus valores, además son independientes del navegador.

[nombre].- Selecciona los elementos que presentan el atributo especificado "nombre" sin importar el valor.

[nombre = "valor"].- Selecciona los elementos que presentan el atributo especificado "nombre" y su valor sea igual a "valor" o comience con la cadena "valor" seguido de un guión (-) y cualquier otra cadena.

[nombre $*=$ "valor"].- Selecciona los elementos que presentan el atributo especificado y su valor contiene la subcadena "valor". Este es el selector por atributo más generoso ya que selecciona a los elementos si la cadena aparece en cualquier parte del valor del atributo.

[nombre $\sim=$ "valor"].- Selecciona los elementos que presentan el atributo especificado y su valor contiene la palabra "valor" delimitada por espacios. En ocasiones es mejor utilizar $\sim=$ a $*=$.

[nombre $ = "valor"].- Selecciona los elementos que presentan el atributo especificado y su valor termina con la cadena "valor". La comparación es sensible a mayúsculas y minúsculas.

[nombre ^ = "valor"].- Selecciona los elementos que presentan el atributo especificado y su valor inicia con la cadena "valor".

[nombre = "valor"].- Selecciona los elementos que presentan el atributo especificado y su valor es exactamente la cadena "valor".

[nombre1 = "valor1"][nombre2 = "valor2"].- Selecciona los elementos que presentan los atributos especificados con los valores especificados respectivamente.

Una página puede contener enlaces internos y externos, los primeros son aquellos que direccionan a una sección o subsección del mismo sitio y los enlaces externos son lo que llevan fuera del sitio.

Supongamos que se desea añadir el valor "_blank" al atributo "target" de los enlaces que direccionan fuera del sitio, es decir, que la página se despligue en otra ventana en blanco, conservando la página de origen. [12] Los enlaces externos inician su cadena de direccionamiento con "http://", pero el dominio debe ser diferente al del sitio original (enlace interno). Para conseguir ese resultado se utilizan los métodos attr(), not() y document.domain de jQuery.

.attr("nombreAtributo", "valor") .- Del primero de los elementos coincidente se obtiene el valor del atributo proporcionado. Si se pasan los dos parámetros se aplica el "valor" al atributo proporcionado.

[12] Este comportamiento no es recomendable ya que se pierde el foco de la ventana.

.not() .- Remueve los elementos listados del conjunto de los elementos coincidentes. Los elementos listados pueden ser elementos DOM, cadenas con expresiones de selección o una función callback.

document.domain .- Obtiene el nombre del dominio de la página.

En el cuadro de código 2.58 se muestra un fragmento del ejemplo atributos.html que hace uso de los métodos antes mencionados junto con el selector por atributo "a[href ^ = 'http://']", para agregar el atributo "target" con valor "_blank" a los links externos de la página, el resultado se observa en la figura 2.84.

Código 2.58: Uso del selector por atributo

```
/*---- Código jQuery ----*/
function linkWebExt(){
var linksExt =$("a[href^='http://']").not("a[href^=
'http://"+document.domain +"']");
linksExt.attr({target: "_blank"});
}
<!-- Código HTML -->
<div class="result">
[...]
<a href='http://www.fciencias.unam.mx'>Web Facultad
de Ciencias</a>
<a href='http://www.unam.mx'>Web UNAM</a>
<a href='atributos_continuacion.html'>Leer más</a>
</div>
```

```
<div class="result">
    <h1 class="esq_red">Lorem ipsum ad his scripta blandit partiendo</h1>
    <div>
        <p> Lorem ipsum ad his scripta blandit partiendo, eum fastidii accumsan euripidis in, eum liber
        hendrerit an. Qui ut wisi vocibus suscipiantur, quo dicit ridens inciderint id. Quo mundi lobortis
        reformidans eu, legimus senserit definiebas an eos. </p>
        <p> Ius id vidit volumus mandamus, vide veritus democritum te nec, ei eos debet libris consulatu. No
        mei ferri graeco dicunt, ad cum veri accommodare. Sed at malis omnesque delicata, usu et iusto zzril
        meliore. Dicunt maiorum eloquentiam cum cu, sit summo dolor essent te. </p>
        <a href="http://www.fciencias.unam.mx" target="_blank"> Web Facultad de Ciencias </a>
        <a href="http://www.unam.mx" target="_blank"> Web UNAM </a>
        <a href="atributos_continuacion.html"> Leer más </a>
    </div>
</div>
```

Figura 2.84: Resultado en atributos.html [Imagen creada por el autor de este trabajo (2011)]

2.4.3.6. Práctica 02

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.4.3 Selectores Avanzados, en cuyos temas se tratan algunos métodos proporcionados por jQuery para manipular el DOM de la página, para seleccionar elementos

dada su posición, dado los valores de sus atributos o de un texto específico. Además se explica el uso del método ready().

También se recomienda acceder a la sección 2.4.6.1 DOM para comparar los métodos de jQuery con los que proporciona el DOM, para manipular la estructura de una página y comprobar que jQuery es una herramienta que facilita el trabajo del programar en gran medida. Además cuenta con la sección 2.4.6.1 Ejercicios DOM
donde se propone resolver algunas cuestiones comunes de una página Web, utilizando únicamente los métodos proporcionados por DOM.

Ejercicios Los documentos HTML de esta práctica se localizan en jQuery/ejercicios/selectores_avanzados.

1.- En el documento "ready.html" implementar las siguientes funciones en jQuery que modifican elementos descendientes al div.result (Un ejemplo del resultado solicitado se muestra en la figura 2.85):

1.1.- Tres funciones en jQuery activadas con ready():

1. La primera función concatena al contenido HTML de cada párrafo con el siguiente contenido: << Contenido final del párrafo N >> donde N es el número de la posición del párrafo en la página.
2. Similar a la anterior pero con los links de la página.
3. Implementar un controlador que aplique una función parecida a las anteriores recibiendo únicamente dos parámetros: el tipo y los elementos a manipular, es decir, **foo(tipo, elemento)** donde:

   tipo Es el tipo de los elementos, es decir, links, párrafos o divs.

   elemento Es el conjunto de elementos seleccionados, relacionado con el tipo.

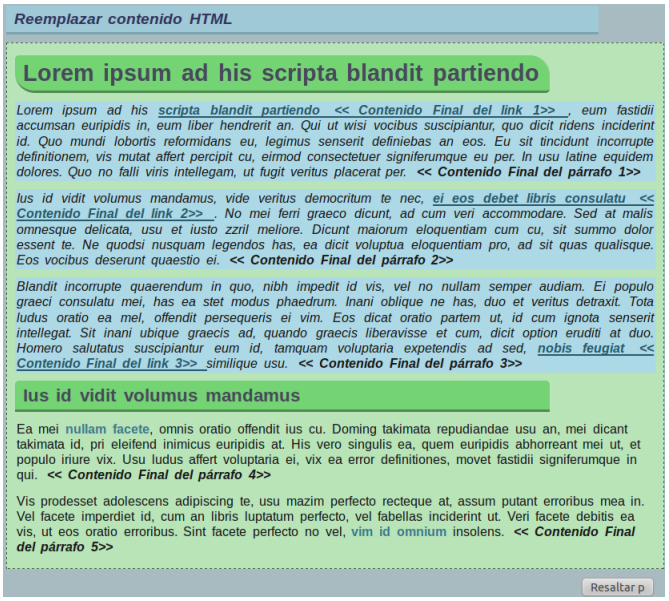


Figura 2.85: Resultado del ejercicio ready.html [Imagen creada por el autor de este trabajo (2011)]

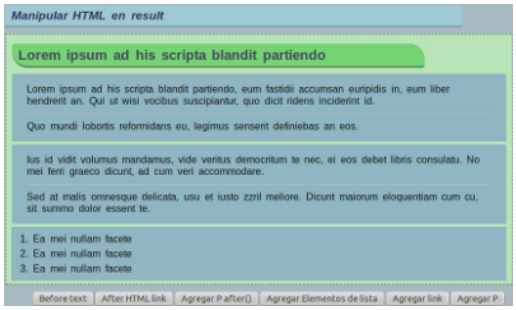
2.- En el documento "DOM.html" implementar las siguientes funciones en jQuery y reglas de estilo que modifiquen los elementos descendientes al div.result (Un ejemplo del resultado solicitado se muestra en la colección 2.86):

2.1.- Estilos de css:

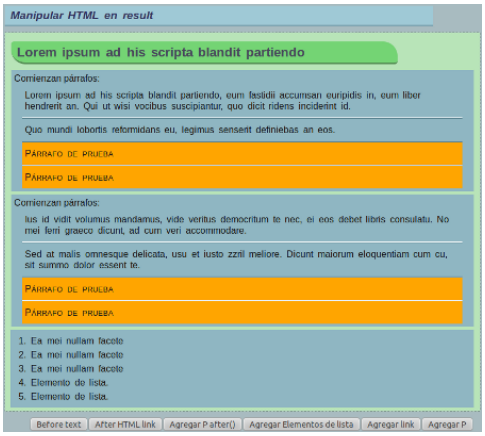
1. A los párrafos contenidos en los divs inmediatos se les define un borde superior azul y uno inferior blanco a excepción del primer párrafo el cual no debe presentar el borde superior y el último párrafo no debe presentar el borde inferior.
2. A los párrafos recien agregados se les debe aplicar una clase que los resalte para diferenciarlos de los párrafos adyacentes.
3. Los elementos en lista deben presentar el estilo mostrado en la imagen del ejemplo.
4. Los links a agregar deben presentar un estilo mostrado en la imagen del ejemplo. Si el alumno lo desea también puede implementar los estilos cuando el link esta en hover y visitado.

2.2.- Funciones en jQuery:

1. Método que agrega un párrafo al final de cada "div" contenido en div.result, sin olvidar aplicar las reglas de estilo referentes a los párrafos. La segunda parte del método es que una vez que se agrega el link al "div", cada vez que se llame a la función



(a) Página original.



(b) Resultado de aplicar las funciones.

Figura 2.86: DOM.html. [Imágenes creadas por el autor de este trabajo (2011)]

que agrega nuevos párrafos el link debe permanecer al final del div como se muestra en la figura 2.86b.

2. Método que agrega un link con el texto "ver más" al final de cada div descendiente al div.result. El método permite agregar únicamente un link por "div".
3. Método que agrega un elemento de lista a la lista descendiente al div.result. Si el alumno cree conveniente aplicar un estilo de CSS que resalte a cada elemento de lista agregado se debe comentar en el código dicha tarea.
4. Método que realiza las mismas funciones que el primer método, utilizando los métodos each(), after() y append(). También debe satisfacer los requerimientos de la segunda parte del método.
5. Método que agrega un texto antes de los primeros párrafos descendientes a un "div" que a su vez descienda de un elemento con clase "result".

6. Método que realiza la misma función que el segundo método, es decir que agrega un link con el texto “Leer más”, pero utilizando el método after().

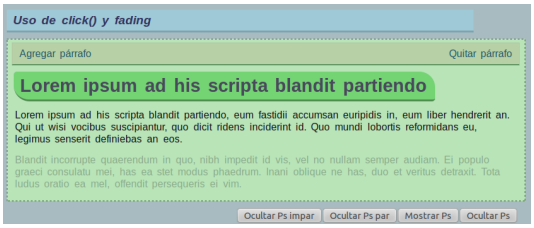
3.- En el documento “nth-child.html” implementar las siguientes funciones en jQuery y reglas de estilo que modifiquen los elementos descendientes al div.result (El funcionamiento de algunos ejercicios solicitados se muestra en la colección 2.88):

3.1.- Estilos de css:

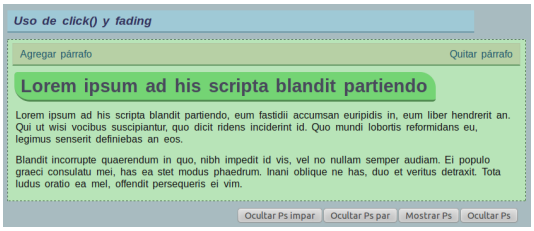
1. Los estilos relacionados con la clase “control” se localizan en “css/style.css”
2. Todos los párrafos descendientes directos del “div” con clase “result” presentan display none (están ocultos).

3.2.- Funciones en jQuery:

1. Al hacer clic sobre el span “Agregar párrafo” se muestran los párrafos inmediatos al “div” con clase “result” uno por cada clic, siendo el orden de definición el orden de aparición. Un ejemplo del resultado esperado se muestra en las figuras 2.87a y 2.87b.



(a) Agregar Párrafo.



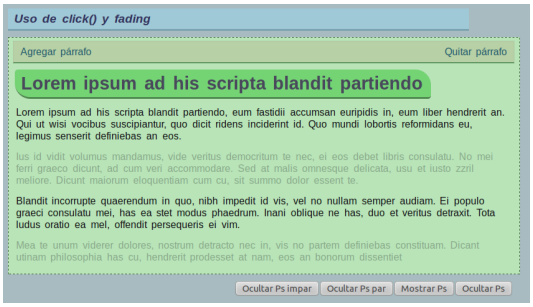
(b) Párrafo Agregado.

Figura 2.87: Algunos resultados de nth-child.html parte 1. [Imágenes creadas por el autor de este trabajo (2011)]

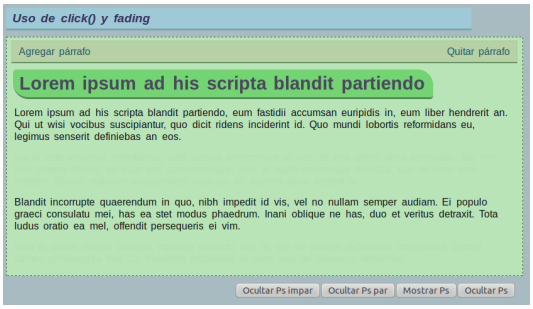
2. Al hacer clic sobre el “span” con clase “Quitar párrafos” se van ocultando los párrafos inmediatos al “div” con clase “result” uno por cada clic, ocultando siempre el último párrafo

agregado. Las funciones "Agregar párrafo" y "Quitar párrafo" deben estar relacionados.

3. Una función que muestre todos los párrafos inmediatos al "div" con clase "result" con su mecanismo de acción. Esta función debe estar relacionado con "Quitar Párrafo" para que en el momento en el que se agreguen los párrafos funcione correctamente, es decir, oculte párrafo a párrafo a partir del último en la definición.
4. Una función con su mecanismo de acción que oculte todos los párrafos inmediatos. Esta función debe estar relacionado con "Agregar Párrafo" para que en el momento en el que se le de clic se vayan agregando los párrafos correctamente.
5. Una función con su mecanismo de acción que oculte los párrafos pares. Un ejemplo del resultado esperado se muestra en las figuras 2.88a y 2.88b.
6. Una función con su mecanismo de acción que oculte los párrafos impares.



(a) Ocultar párrafos pares



(b) Párrafos Ocultados

Figura 2.88: Algunos resultados de nth-child.html parte 2. [Imágenes creadas por el autor de este trabajo (2011)]

4.- ¿Qué son los métodos "traversing"? ¿Pará que sirven los métodos parent(), parents(), children(), prev(), next() y siblings()?

5.- En el documento "contains.html" implementar las siguientes funciones en jQuery y reglas de estilo que modifiquen los elementos descendientes al div.result (El funcionamiento de algunos ejercicios solicitados se muestra en la figura 2.89):

5.1.- Estilos de css:

1. La clase de estilo "literario" que principalmente pone el texto en justificado y en itálica y un color de fondo #ADD8E6.
2. Los divs inmediatos al "div" con clase "result" se les escapan los elementos flotados y se aplica un relleno.
3. A los links que cumplen con div.result los divs se ponen flotados a la derecha con una separación entre elementos.
4. La clase de estilo "censurado" que posiciona al elemento de forma absoluta, con un color de fondo rojo y un color de letra amarillo en negritas.
5. A los "span" inmediatos al "div" con clase "censurado" se les aplica un tamaño de letra de 22px.

5.2.- Funciones en jQuery:

1. Al hacer clic sobre "Buscar Text en PS" se aplica la clase "literario" a los párrafos descendientes al "div" con clase "result" que contengan el texto "Lorem ipsum".
2. Al hacer clic sobre "Agregar Text a link" se agrega un texto al final de los links con clase "link_leer_mas". La función puede agregar el texto solo una vez y después el botón de acción se torna como "desactivado" (con cierta opacidad), y aunque el usuario de clic sobre el botón no se obtendrán más inserciones.
3. Al hacer clic sobre "Censurar párrafo" se agrega un "div" con clase "censurado" a los párrafos descedientes al "div" con clase "result", que contengan el texto "XYX!!". De tal forma que el div.censurado abarque todo el párrafo y oculte el contenido, mostrando únicamente el texto "Censurado".

6.- En el documento "atributos.html" implementar las siguientes funciones en jQuery que modifiquen los elementos descendientes al div.result (El funcionamiento de los ejercicios solicitados se muestra en la figura 2.90):

6.1.- Estilos de css:

1. Para cumplir con el estilo y resultado solicitado de los siguientes ejercicios se deben comprender las reglas de estilo sobre el bloque con clase "result_html" y sus contenidos definidos en el archivo "css/style.css". Si el alumno desea cambiar el estilo

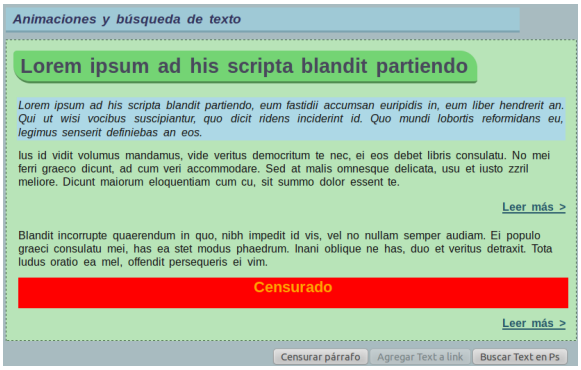


Figura 2.89: Resultados en contains.html. [Imagen creada por el autor de este trabajo (2011)]

y proponer otros es necesario comentar esos cambios y de ser posible no modificar el archivo "style.css".

6.2.- Funciones en jQuery:

1. Al cargar la página se deben agregar bloques con clase "descripcion" al "div" con id "antes" contenido en el "div" con clase "result_html", estos bloques deben contener la siguiente información sobre cada enlace externo de la página:
   a. El atributo href y su valor.
   b. El atributo target y su valor.
2. Al hacer clic sobre "Links externos _blank" se ejecuta una función que egrega el atributo "target" con el valor "_blank" a los links externos de la página. Con ello los links se abren en una ventana en blanco. Además se deben agregar bloques con clase "descripcion" al "div" con id "despues" contenido en el div con clase "result_html", estos bloques deben contener la siguiente información sobre cada enlace externo de la página:
   a. El atributo href y su valor.
   b. El atributo target y su valor.
3. Después de aplicar la función "Links externos _blank" el botón de acción se debe deshabilitar para evitar la inserción de más bloques de información. Se permite cualquier mecanismo para llegar al resultado.

7.- Crear una página Web que contenga el código 2.59 e implementar una función que agregue el texto "Botón deshabilitado" a los botones con el atributo y valor "disabled", después de aplicar la función el botón de acción se debe deshabilitar, para evitar la inserción del mismo texto. (El funcionamiento del ejercicio solicitado se muestra en la colección 2.91):

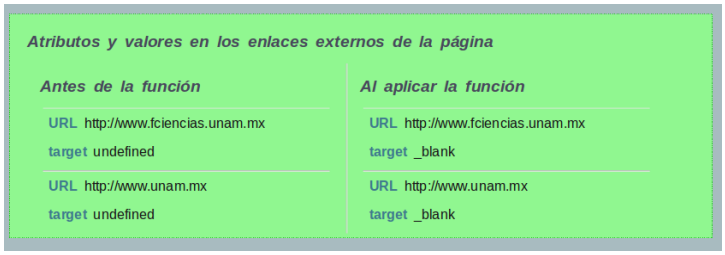


Figura 2.90: Resultados en atributos.html. [Imagen creada por el autor de este trabajo (2011)]

Código 2.59: Selector por atributo

```
<!-- Código HTML -->
<div class="section">
[...]
<div><button disabled="disabled"> Botón Interno
</button> </div>
<div><button disabled="disabled"> Botón Formulario
</button></div>
<div><button> Botón Externo</button></div>
[...]
</div>
```

(a) Página original.

(b) Después de aplicar los métodos.

Figura 2.91: Botones deshabilitados. [Imágenes creadas por el autor de este trabajo (2011)]

2.4.4. Funciones

Los temas de este capítulo explican algunos métodos útiles que proporciona jQuery en el desarrollo de aplicaciones Web. Dentro de la cuales se abordan el método "each()" para recorrer los elementos seleccionados y aplicar una función a cada uno de ellos, el método "html()" para acceder a los elementos y editar su contenido HTML o desplazarlos, así como el método "wrap()" que permite agrupar una estructura dentro de otra.

Otro tema de relevante abordado son los métodos de manejo de eventos, siendo este tema uno de los más importantes en el desarrollo de aplicaciones Web dinámicas. En el capítulo se presentan algunas circunstancias en las cuales se abordan el manejo de eventos; como la interacción del usuario con la página a través de "clicks", del movimientos del mouse o desplazamientos del "scroll bar", entre otros.

Para terminar se ejemplifican algunos métodos útiles relacionados con las operaciones de arreglos, así como la conversión de objetos a elementos parecidos a los arreglos. Además se menciona el uso de jQuery.support para considerar diseñar páginas Web pensadas en las funcionalidades de los navegadores.

2.4.4.1. Aplicar una función a cada elemento del conjunto

Al aplicar un método jQuery a los elementos seleccionados, por ejemplo .hide(), jQuery oculta cada elemento del conjunto, es decir, le aplica el método solicitado a cada elemento. Para las funciones implementadas por el programador jQuery proporciona el método each(), que permite iterar una función a cada elemento del conjunto selección.

Por ejemplo, si se desea agregar un valor al atributo "title" de las imágenes de una galería, se puede utilizar el método each() para iterar una función que desempeñe dicha tarea, como se muestra en el cuadro de código 2.60, en el cual se agrega el texto "Fotografía N de la Galería Fciencias" donde N es un contador que toma valores a partir de 0. Si las imágenes presentan un valor previo en el atributo "title" al aplicar la función se sobrescribe dicho valor. En la figura 2.92 se muestra el resultado de aplicar esta función.

Código 2.60: Agregar un título a las imágenes

```
1 /*---- Código jQuery ----*/
2 $('#galeria img').each(function(cnt){
3   $(this).attr('title', 'Fotografía '+ (++cnt) +
4   'de la Galería Fciencias');
5 });
```

Supongamos que las imágenes de la galería presentan un título y se desea editar el texto original concatenando un texto. En el cuadro de código 2.61 se

```
<div style="border-right: 1px solid rgb(229, 229, 229);">
    <p class="caption">Edificio 0</p>
    <img width="260" height="195" alt="" title="Fotografía 1 de la Galería Fciencias" src="../../images
    /galeria/edificio0.jpeg">
</div>
<div style="border-left: 1px solid rgb(77, 77, 77);">
    <p class="caption">Prometeo</p>
    <img width="260" height="195" alt="" title="Fotografía 2 de la Galería Fciencias" src="../../images
    /galeria/prometeo.jpeg">
</div>
```

Figura 2.92: Agregar un título a las imágenes en galeria.html [Imagen creada por el autor de este trabajo (2011)]

accede al valor del atributo "title" de las imágenes utilizan el método attr(), que además recibe como parámetro una función que edita el título original agregando un nuevo texto, como se muestra en la figura 2.93.

Código 2.61: Editar el título de las imágenes

```
/*---- Código jQuery ----*/
$('#galeria img').each(function(){
  var tituloPrevio = $(this).attr('title');
  $(this).attr('title', tituloPrevio + ' (fotografía
  del autor)');
});
```

```
<div style="border-right: 1px solid rgb(229, 229, 229);">
    <p class="caption">Edificio 0</p>
    <img width="260" height="195" alt="" title="Edificio 0 de FCiencias (Fotografía de
    kienkane)" src="../../images/galeria/edificio0.jpeg">
</div>
<div style="border-left: 1px solid rgb(77, 77, 77);">
    <p class="caption">Prometeo</p>
    <img width="260" height="195" alt="" title="Fuente de la FCiencias (Fotografía de
    kienkane)" src="../../images/galeria/prometeo.jpeg">
</div>
```

Figura 2.93: Editar el título de las imágenes [Imagen creada por el autor de este trabajo (2011)]

2.4.4.2. Editar el contenido HTML

jQuery permite agregar un nuevo contenido HTML o sobrescribir uno previo utilizando el método .html() (visto en la sección 2.4.3.1 Página en estado Ready), esta función es muy útil para agregar nuevas estructuras HTML junto con su contenido o reemplazar el contenido de las estructuras seleccionadas. Además jQuery proporciona el método .text() (visto en la sección 2.4.2.2 Seleccionar por ID) que permite modificar únicamente el texto de los elementos HTML.

En el cuadro de código 2.62 se muestra una alternativa para cambiar el contenido HTML de los primeros párrafos inmediatos a un elemento "div"

agregando un elemento "span", seguido del contenido original utilizando el método .html() el resultado se puede visualizar en la figura 2.94

Código 2.62: Agregar contenido .html()

```
/*---- Código jQuery ----*/
var cont_span = "<span>Agregar Contenido</span>";
var prev = $('div > p:first-child').html();
$('div > p:first-child').html(cont_span + prev);
```

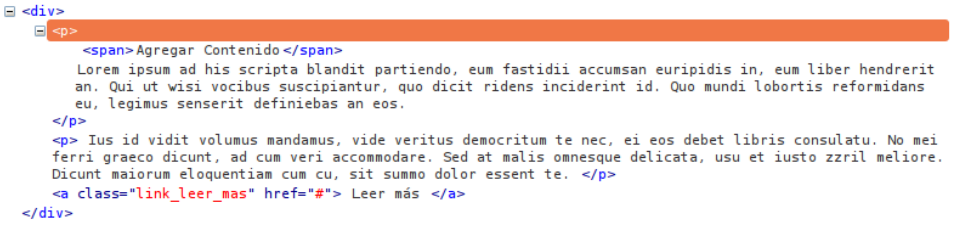


Figura 2.94: Agregar contenido HTML [Imagen creada por el autor de este trabajo (2011)]

Cuando únicamente se desea modificar el texto de los elementos es recomendable utilizar el método text() en lugar de la opción html(). En el cuadro de código 2.63 que es un fragmento del documento "agregar_contenido.html", se muestra el uso del método para concatenar un texto que enumera todos los links, el resultado se muestra en la figura 2.95

Código 2.63: Agregar contenido .html()

```
/*---- Código jQuery ----*/
var prev = $('.result a');
for(var i = 0; i <= prev.length; i++){
  prev.eq(i).text(prev.eq(i).text() + " ( link "+
  (i+1) +" )");
}
```

2.4.4.3. Desplazar y agrupar elementos HTML

Con los métodos append() y appendTo() de jQuery es posible agregar o cambiar de posición elementos HTML de la página. Usando la función append() se colocan los elementos al final del contenido del elemento seleccionado o al principio usando el método appendTo().

En el cuadro de código 2.64 que es un fragmento del documento "mover Elem.html" se muestran dos funciones, la primera agrega un link externo al "div" con clase "links" y la segunda un elemento de lista al elemento "ol" (lista ordenada). Además contiene dos funciones que desplaza elementos de

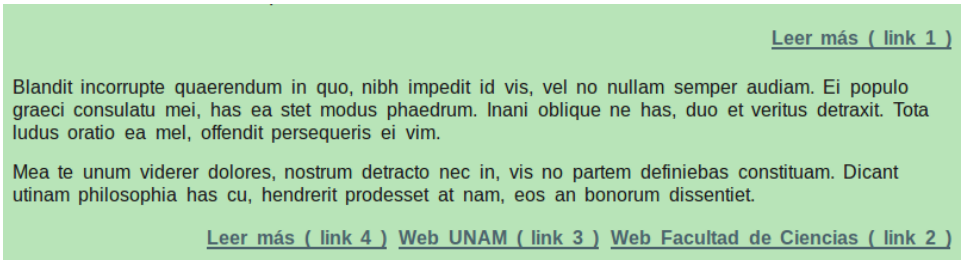


Figura 2.95: Concatenar texto con .text() [Imagen creada por el autor de este trabajo (2011)]

la página la primera desplaza la lista ordenada al final del documento y la segunda desplaza el "div" con clase "links" al final del documento. El resultado se visualiza en la figura 2.96.

Código 2.64: Agregar y desplazar con .append()

```
/*---- Código jQuery ----*/
function agregarLink(){
$('.result .links').append("<a href='http://www.unam.mx'>
Web UNAM </a>")
}
function agregarItem(){
$('.result ol:first').append("<li>Inani oblique ne has,
duo veritus detraxit. Tota ludus oratio ea mel.</li>");
}
function desplazarLink(){
$('.result').append($('.result .links'));
}
function desplazarList(){
  $('.result p').append($('.result ol'));
}
```

En la sección 2.4.3.2 Agregar contenido HTML en una posición se ejemplífica el uso de los métodos before() y after() los cuales permiten agregar elementos HTML o desplazarlos, antes o después (respectivamente) de los elementos seleccionados.

En el cuadro de código 2.66 se muestra el uso de before() para desplazar la lista ordenada antes del "div" con clase "links" pero al mismo nivel estructural. Y after() para desplazar el div con clase "links" después del último párrafo en "result", pero al mismo nivel estructural. El resultado de aplicar esas funciones se muestra en la figura 2.97.

Código 2.65: Desplazar con .before() y .after()

```
/*---- Código jQuery ----*/
```

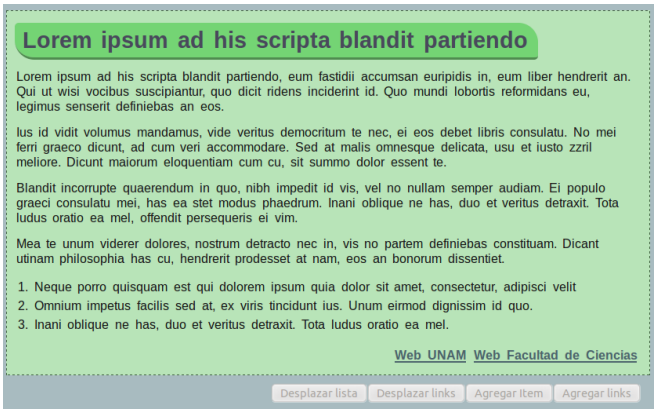


Figura 2.96: Desplazar elementos con append() [Imagen creada por el autor de este trabajo (2011)]

```
2  function desplazarLink2(){
3    $('.result > p:last-child').after($('.result .links'));
4  }
5  function desplazarList2(){
6    $('.result > .links').before($('.result ol'));
7  }
```

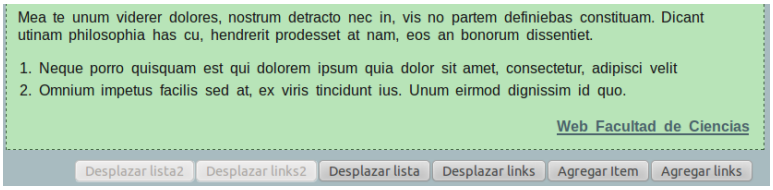


Figura 2.97: Desplazar elementos con before() y after() [Imagen creada por el autor de este trabajo (2011)]

jQuery permite agrupar y envolver elementos seleccionados utilizando el método wrap(), útil para construir y ordenar estructras de forma dinámica.

1.- **wrap(Elemento_agrupador)**

2.- **wrap(function(indice))**

En la primera firma del método el elemento agrupador es una etiqueta HTML, objeto jQuery o elemento DOM con el cual se envuelven los elementos seleccionados. En la segunda se hace referencia a una "función callback" que ejecuta su contenido sobre cada elemento seleccionado.

En el cuadro de código 2.66 se muestra una función que utiliza wrap() para envolver cada párrafo dentro de un "div" con clase "wrapP", el resultado se muestra en la figura 2.98.

Código 2.66: Desplazar con .before() y .after()

```
/*---- Código jQuery ----*/
function envolverP(){
$('.result div.links').after("<div class='wrapPs'>
</div>");
$('.result > *').not('h1, ol, ul, .links').
wrap($('div.wrapPs'));
}
```

```
<div class="wrapP">
    <p> Lorem ipsum ad his scripta blandit partiendo, eum fastidii accumsan euripidis in, eum liber
    hendrerit an. Qui ut wisi vocibus suscipiantur, quo dicit ridens inciderint id. Quo mundi lobortis
    reformidans eu, legimus senserit definiebas an eos. </p>
</div>
<div class="wrapP">
    <p> Ius id vidit volumus mandamus, vide veritus democritum te nec, ei eos debet libris consulatu. No mei
    ferri graeco dicunt, ad cum veri accommodare. Sed at malis omnesque delicata, usu et iusto zzril meliore.
    Dicunt maiorum eloquentiam cum cu, sit summo dolor essent te. </p>
</div>
<div class="wrapP">
    <p> Blandit incorrupte quaerendum in quo, nibh impedit id vis, vel no nullam semper audiam. Ei populo
    graeci consulatu mei, has ea stet modus phaedrum. Inani oblique ne has, duo et veritus detraxit. Tota
    ludus oratio ea mel, offendit persequeris ei vim. </p>
</div>
```

Figura 2.98: Agrupar elementos con wrap() [Imagen creada por el autor de este trabajo (2011)]

2.4.4.4. Manejo de eventos

Con JavaScript es posible implementar páginas Web dinámicas con una perspectiva de respuesta similar a las aplicaciones de escritorio, en esta dinámica se encuentra el manejo de las acciones que el usuario realiza sobre el sistema, como los clics sobre elementos de la página, los movimientos del mouse (como el "hover") y las combinaciones de teclas (como en los formularios de petición). A este control de acciones se lo conoce como manejo de eventos siendo los clicks, mouse movements y keystrokes (clics, movimientos del mouse y combinaciónes de tecla respectivamente) los eventos a capturar, analizar y dar respuesta.

Este manejo de eventos es una de las razones por las cuales JavaScript tiene tanto éxito e importancia en el desarrollo de aplicaciones dinámicas, pero su administración e implementación son la principal dificultad ya que dependen de cada navegador (browser). Unos de los principales pilares de jQuery es el de unificar el manejo de eventos para cada navegador (cross-browser pág. 127).

Hasta el momento se ha explicado el evento click (pág. 150) que ocurre cuando el usuario hace clic sobre la área del elemento en cuestión y completa la acción, es decir, suelta el botón del mouse dentro de dicha área. Otro evento visto es el que ocurre cuando al cargar la página se ejecuta un código, es decir, el uso de la función $(document).ready() (pág. 144).

Cuando un evento ocurre jQuery conecta ese evento con el manejador de eventos (handler), por ejemplo, al dar clic con el mouse sobre un enlace o botón de la página, el manejador de eventos es llamado y se ejecuta el código relacionado con dicho evento. Cuando se desea enlazar un evento con una acción o función particular jQuery porporciona el método bind().

.bind(eventType, handler(eventObject)) Método que enlaza un evento (eventType) o conjunto de eventos con una función (handler).

eventType .- Es el tipo de evento a conectar, por ejemplo "click", "submit" o "blur" (tipos de eventos de JavaScript).

handler(eventObject) .- Es una función que se ejecuta en el momento en el que se completa el evento ocurrido.

jQuery también proporciona métodos independientes para cada tipo de evento estándar de JavaScript sin necesidad de usar el método bind(), se conecta el evento en cuestión con una acción o acciones. La siguiente lista muestra algunos de esos eventos:

1.-	blur	9.-	click	17.-	mouseleave
2.-	focus	10.-	dblclick	18.-	change
3.-	focusin	11.-	mousedown	19.-	select
4.-	focusout	12.-	mouseup	20.-	submit
5.-	load	13.-	mousemove	21.-	keydown
6.-	resize	14.-	mouseover	22.-	keypress
7.-	scroll	15.-	mouseout	23.-	keyup
8.-	unload	16.-	mouseenter	24.-	error

Como se indica en el listado jQuery proporciona el manejo de los eventos "mouseover" y "mouseout", que son equivalentes a "onmouseover" y "onmouseuot" de JavaScript. El primero lanza la acción programada cuando el puntero del mouse pasa por encima de la área en cuestión y cuando el puntero sale de dicha área se acciona el segundo evento. Además jQuery proporciona el método "hover" que desempeña la misma acción. En el cuadro de código 2.67 que es un fragmento del documento "eventos.html" se muestra el uso de hover y de mouseover/mouseout.

El método hover recibe dos funciones la primera se dispara cuando el puntero del mouse ingresa a la área de los elementos seleccionados y la segunda se dispara cuando el puntero sale, conjuntando las acciones de mouseover y mouseout.

Código 2.67: Hover y eventos

```
/*-- Código jQuery uso de hover() --*/
$(document).ready(function(){
 $('.result ul li').hover(
 function(){
  $(this).css({background:"#FFFFBF",fontWeight:"bold",
  padding:"2px"});
 },
 function(){
  $(this).css({background:"transparent",fontWeight:"400",
  padding:"0"});
  }
 );
)}
/*-- bind con mouseover y mouseout --*/
$(document).ready(function(){
 $('.result ul li').bind({
 mouseover: function(){
  $(this).css({background:"#FFFFBF",fontWeight:"bold",
  padding:"2px"});
 },
 mouseout:function(){
  $(this).css({background:"transparent",fontWeight:"400",
  padding:"0"});
 }
 });
)}
/*-- bind con mouseover, mouseout y una función. --*/
$(document).ready(function(){
 $('.result ul li').bind("mouseover mouseout",function(){
  $(this).toggleClass('resaltar');
 });
});
```

En el segundo bloque de código (en 2.67) se utiliza el método bind() para enlazar dos eventos con diferentes funciones a los elementos seleccionados, a lo que se conoce como enlazar eventos múltiples. Una variante del manejo de eventos múltiples se muestra en el último bloque de código en el cual se listan los eventos a manejar seguidos por la función que alterna una clase de estilo CSS a los elementos seleccionados al dispararse los eventos listados.

Cuando una función se dispara al llevarse acabo un evento, el "Objeto Evento" de JavaScript (event object) es pasado como el primer parámetro de la función callback y proporciona métodos y propiedades del proceso en cuestión.

Las siguientes propiedades estan disponibles para el objeto evento, sus valores dependen del evento en cuestión por lo que algunos de estos valores pueden no estar definidos.

1.- altKey
2.- attrChange
3.- attrName
4.- button
5.- charCode
6.- clientX
7.- clientY
8.- ctrlKey
9.- currentTarget
10.- data
11.- detail
12.- which
13.- handler
14.- layerX
15.- layerY
16.- metaKey
17.- pageX
18.- pageY
19.- prevValue
20.- relatedTarget
21.- screenX
22.- screenY
23.- shiftKey
24.- target
25.- toElement
26.- view

jQuery proporciona un soporte cross-browser (manejo coherente en los principales navegadores) para las siguientes propieades.

1.- target
2.- relatedTarget
3.- pageX
4.- pageY
5.- which
6.- metaKey

Además de las propiedades el objeto evento proporciona métodos que se pueden utilizar dentro de las funciones callback.

1.- **event.preventDefault** Si se ejecuta detiene la acción predeterminada asociada con el evento disparado. Por ejemplo en el cuadro de código 2.68 el enlace dirige a la página oficial de la UNAM, pero al accionar la función preventDefault() se detiene dicho comportamiento.

2.- **event.isDefaultPrevented()** Regresa "True" si el método event. preventDefault() ha sido llamado, como se muestra en el código 2.68.

Código 2.68: event.preventDefault

```
<!-- Código HTML. -->
<a id="link" href="http://www.unam.mx/"
target="_blank">UNAM</a>

/*-- event.isDefaultPrevented() --*/
$('a#link').click(function(e){
 alert(e.isDefaultPrevented()); //el resultado es
    False
 e.preventDefault();
 alert(e.isDefaultPrevented()); //el resultado es
    True
```

```
10 });
```

3.- **event.stopPropagation()** Si se ejecuta detiene la propagación de la acción asociada con el resto de los elementos del árbol DOM coincidentes de la selección. En el ejemplo 2.69 se ejecuta el método event. stopPropagation() y produce que la regla de CSS se aplique únicamente al "div" que recibió el clic de forma directa, como se muestra en la figura 2.99a, de lo contrario si el clic se realizará en el "div" más profundo la regla también afectaría a los "div" contenedores (que coincidan con el selector), como se muestra en la figura 2.99b.

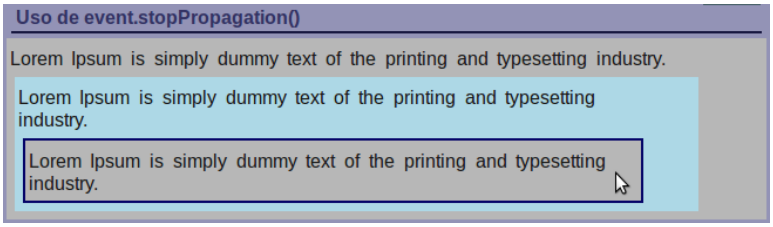
4.- **event.isPropagationStopped()** Regresa "True" si el método event. stopPropagation() ha sido llamado, como se muestra en el código 2.99.

Código 2.69: event.stopPropagation()

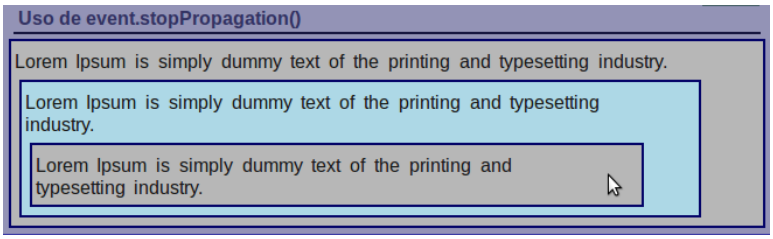
```
1  <!-- Código HTML. -->
2  <div>
3   <p>Lorem Ipsum is simply dummy text of the printing
4   and typesetting industry.</p>
5   <div>
6    <p>Lorem Ipsum is simply dummy text of the
7    printing and typesetting industry.</p>
8    <div>
9     <p>Lorem Ipsum is simply dummy text of the
10    printing and typesetting industry.</p>
11   </div>
12  </div>
13 </div>
14
15 /*---- event.stopPropagation()----*/
16 $('#SPropagation, #SPropagation div').click(
17  function(event){
18   alert(event.isPropagationStopped());
19   event.stopPropagation();  /*Si se comenta esta
20   línea la acción se propaga a los contenedores*/
21   $(this).css("border","2px solid #040468");
22   alert(event.isPropagationStopped());
23 });
```

2.4.4.5. Métodos Útiles

jQuery proporciona métodos útiles que permiten hacer tareas de JavaScript de forma rápida, fácil y soportada por los navegadores. Los nombres de estas



(a) Se detiene la propagación de la acción.



(b) La acción se propaga a través del DOM.

Figura 2.99: event.stopPropagation [Imágenes creadas por el autor de este trabajo (2011)]

funciones llevan la siguiente estructura "jquery.nombreFuncion()" o "$.nombreFuncion()" donde "nombreFuncion", es el nombre de la función que se ejecuta. Por ejemplo la función "$.trim()", que elimina espacios en blanco al principio y final de una cadena de texto.

Estos métodos forman parte de la biblioteca jQuery y son códigos generados en base y para JavaScript, por lo que cada programador puede implementar sus propias funciones que desempeñen las mismas tareas, lo que sería reinventar la rueda. jQuery proporciona estos métodos para facilitar el uso de JavaScript en el desempeño de muchas tareas comunes en la creación de sistemas Web, dejando al programador implementar las tareas personalizadas o propias de cada proyecto.

A continuación se abordan métodos útiles que proporciona jQuery para facilitar algunas acciones comunes en la implementación de páginas Web, así como funciones que pueden resultar de gran utilidad.

Métodos sobre arreglos

jQuery ofrece muchas funciones últiles en el manejo de arreglos. Por ejemplo, para manejar los elementos de una página, sus características y propiedades suele ser más sencillo trabajarlos por colecciones de elementos, es decir, convertir los elementos en forma de arreglos.

Se puede convertir un objeto o conjunto de elementos en un arreglo o en una estructura parecida a un arreglo utilizando el método "makeArray()". Se dice que es una estructura similar por que adquiere propiedades y métodos propios de los arreglos nativos que le permiten calcular su tamaño, acceder

al contenido, entre otros métodos más.

$.makeArray() Método que convierte los elementos seleccionados en un arreglo.

jQuery proporciona muchos métodos que devuelven objetos parecidos a los arreglos, el más utilizado es $() que devuelve un objeto con propiedades y características propias de los arreglos como la forma para obtener el tamaño o acceder a los valores, etc. Pero el resultado no es exactamente un arreglo, ya que hay otros métodos como .pop() y .reverse() que no se pueden aplicar a los objetos resultantes. Para comprobar si un objeto es un arreglo jQuery proporciona el método $.isArray(), que devuelve "true" si el objeto pasado es un arreglo y "false" de lo contrario.

En el cuadro de código 2.70 que es un fragmento del documento "arreglos.html" se utiliza el método $.makeArray para convertir los párrafos en elementos de un arreglo y setTimeout para demorar 1200 milisegundos la tarea de agregar los elementos impares al "div" con clase "testResult". El resultado se muestra en la figura 2.100. Además se despliega un cuadro de diálogo indicando si la estructura generada se trata de un arreglo.

Código 2.70: Uso de $.makeArray()

```
/*---- Código jQuery ----*/
window.setTimeout(function() {
 var ps = $('.result .contenedorTest .test
 p:nth-child(odd)');
 var arrayPs = $.makeArray(ps);
 for(var i=0; i< arrayPs.length; i++){
  $('.result .contenedorTest .testResult').
  append(arrayPs[i]);
 }
 if($.isArray(arrayPs)){
  alert("El objeto generado es un arreglo");
 }else{
  alert("El objeto generado NO es un arreglo");
 }
}, 1200);
```

Entre algunas de las operaciones comunes y útiles de los arreglos se encuentran; la búsqueda de elementos en el arreglo, la construcción de un arreglo a partir de reglas a modo de filtros que se aplican a otro arreglo y el mapeo de elementos, el cual permite trabajar con cada elemento del arreglo y modificar su valor creando un nuevo arreglo.

$.inArray(term, array) Método que permite buscar un elemento particular en una arreglo, sea un número o una cadena. Cuando lo encuentra

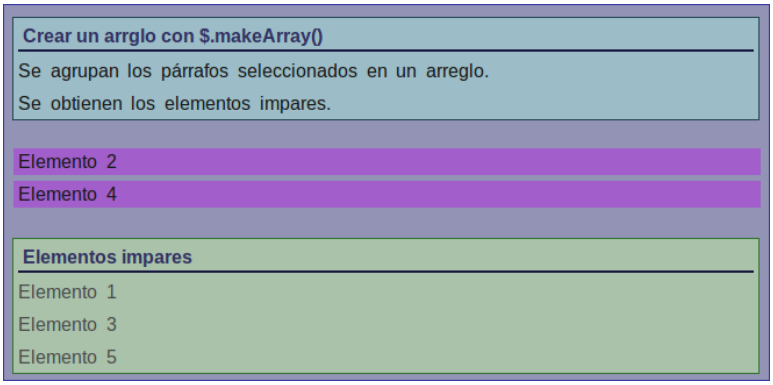


Figura 2.100: Arreglo de párrafos [Imagen creada por el autor de este trabajo (2011)]

devuelve el índice que ocupa el elemento en el arreglo (siendo el 0 el índice base), de lo contrario devuelve -1. Donde "term" es el término a buscar y "array" el arreglo en cuestión.

$.grep(array, function(v, i)) Método que permite crear un nuevo arreglo a partir de aplicar reglas de filtro a otro arreglo. Donde "array" es el arreglo en cuestión, "v" es el elemento e "i" es el índice que ocupa el elemento en el arreglo.

$.map(array, function(v, i)) Método que crea un nuevo arreglo a partir de modificar el valor de los elementos de otro arreglo. Donde "array" es el arreglo en cuestión, "v" es el elemento e "i" es el índice que ocupa el elemento en el arreglo.

En el cuadro de código 2.71 que es un fragmento del documento "arreglos.html" se muestra un ejemplo del uso de estos métodos. En el primer bloque se busca un elemento dentro del arreglo e imprime el índice en el que se localiza. El segundo bloque crea un nuevo arreglo a partir de los elementos cuyos índices sean mayores a 1. Y el último bloque concatena una cadena de texto a cada elemento de un arreglo.

Código 2.71: Funciones útiles

```
1 /*---- Código jQuery uso de $.inArray() ----*/
2 function bArr(){
3  [...]
4  var res = $.inArray("Elemento 2", arrayPs);
5  $('.result .contenedorTest .testResult:eq(1)')
6  .append("<p> El elemento se localiza en el índice"
7  + res+ "</p>");
8 }
9 /*--- Código jQuery uso de $.grep() ---*/
```

```
10 function filArr(){
11  var arrTemp = arr;
12  var res = $.grep(arrTemp, function(v, i){
13   return i > 1;
14  });
15  for(var i=0; i< res.length; i++){
16   $('.result .contenedorTest .testResult:eq(2)')
17   .append(res[i]);
18  }
19 }
20 /*--- Código jQuery uso de $.map() ---*/
21 function modArr(){
22  [...]
23  var j=1;
24  var res = $.map(arrayPs, function(v, i){
25  return "<p>"+ v + " cadena extra "+ (j++) +"</p>";
26  });
27  for(var i=0; i< res.length; i++){
28   $('.result .contenedorTest .testResult:eq(3)')
29   .append(res[i]);
30  }
31 }
```

2.4.4.6. Métodos sobre navegadores

jQuery proporciona métodos y propiedades relacionadas con los navegadores, indicando sus características y defectos (propiedades que no soportan). Es una buena práctica de programación identificar primero si el navegador con el cual esta accediendo el usuario a la página, soporta las propiedades de CSS avanzadas (dado el caso de que las presente la página), métodos de JavaScript, métodos de manipulación del DOM, AJAX, entre otras. Con esta identificación es posible asegurar un mejor desempeño de la página.

Para esta tarea jQuery proporciona propiedades que son utilizadas en la detección de estas caraterísticas a través del método jQuery.support. Para condicionar los scripts de la página a las funcionalidades soportadas por el navegador y no al navegador en cuestión.

La siguiente lista muestra el uso de $.support para obtener algunos valores de las propiedades y características del navegador. Para cada variable se accede de la forma $.support.NombrePropiedad, donde "NombrePropiedad" puede tomar los siguientes valores.[13]

- **boxModel** Devuelve "True" si el navegador soporta el modelo de cajas de CSS como lo específica la W3C.

[13] La lista completa se puede observar en la página oficial de jQuery

- **cssFloat** Devuelve "True" si el nombre de la propiedad que contiene el valor de "float" de CSS es ".cssFloat", tal y como lo define la especificación de CSS de la W3C. [14]
- **opacity** Devuelve "True" si el navegador interpreta la propiedad de opacidad de CSS dada la especificación de CSS de la W3C.[1]
- **style** Devuelve "true" si los estilos en línea de un elemento pueden ser accedidos a través del atributo DOM "style". De ser el caso con ".getAttribute('style')" se puede recuperar ese valor. Para los navegadores IE se utiliza ".cssText".
- **scriptEval** Devuelve "true" si los scripts en línea son automáticamente evaluados y ejecutados cuando se insertan en el documento utilizando métodos de manipulación DOM, como ".appendChild()" y ".createTextNode()".
- **AJAX** Devuelve "true" si el navegador es capaz de crear el objeto XMLHttpRequest.
- **checkOn** Devuelve "true" si el valor por defecto de los checkbox es "on" cuando no se específica un valor.

2.4.4.7. Práctica 03

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.4.4 Funciones, donde se tratan temas relacionados con el manejo de estructuras y agrupamientos, desplazamientos y edición de contenidos, manejo de eventos, operación con arreglos e identificación de las características y defectos de los navegadores.

Ejercicios Los documentos HTML de esta práctica se localizan en jQuery/ejercicios/funciones.

1.- En el documento "galeria.html" implementar las siguientes funciones en jQuery que modifican elementos descendientes al div.result (Un ejemplo del resultado solicitado se muestra en la figura 2.101):

1.1.- Referente a los estilos CSS

1. Las reglas de estilo de la galería se localizan en el documento "css/galeria.css".
2. Al cargar la página, aplicar una línea divisoria vertical que pase por el centro de la galería, como se muestra en la figura 2.101a.

[14] Devuelve falso para los navegadores de IE

1.2.- Funciones en jQuery:

1. Sobrescribir el código de la función agregarTitleCtn() sin utilizar el método attr(), manteniendo el mismo resultado.
2. Sobrescribir el código de la función editarTitleEach() sin utilizar el método attr(), manteniendo el mismo resultado.
3. Implementar el código de la función editarTitle() accionada por el botón "Editar Título", que desempeñe la misma tarea que el método editarTitleEach() sin utilizar el método each() y ejecutando el método attr() una vez por cada elemento seleccionado.
4. Implementar un mecanismo que una vez ejecutada cualquiera de las funciones agregarTitleCtn(), editarTitleEach() o editarTitle() se inhabiliten sus funciones o sus mecanismos de acción, para evitar agregar la misma información. Un posible resultado se muestra en la figura 2.101b.
5. Implementar el código de la función agregarAlt() que agrega al atributo "alt" de las imágenes, la cadena de texto formada por "Imagen " concatenada con el nombre de la imagen sin su extensión obtenida a partir del valor del atributo "src". El resultado esperado se muestra en la figura 2.101b.

2.- En el documento "agregar_cont.html" implementar las siguientes funciones en jQuery que modifican elementos descendientes al div.result (Un ejemplo del resultado solicitado se muestra en la figura 2.102):

2.1.- Funciones en jQuery:

1. Sobrescribir el código de la función agregar_span(id) para utilizar el método html() solo una vez, conservando el mismo resultado.
2. Sobrescribir el código de la función agregar_text(id) para utilizar el método text() solo una vez, conservando el mismo resultado.
3. Implementar una función que se invoque una vez ejecutada cualquiera de las funciones anteriores para deshabilitar su mecanismo de acción y evitar su ejecución nuevamente. El resultado esperado se muestra en la figura 2.102.

3.- En el documento "galeria.html" implementar las siguientes funciones en jQuery que agregan estructuras HTML con información referente a las imágenes de la galería (Un ejemplo del resultado solicitado se muestra en la figura 2.103):

3.1.- Referente a los estilos CSS

(a) Deshabilitar las funciones relacionadas con "title"

```
<div id="galeria">
  <div style="border-right: 1px solid rgb(229, 229, 229);">
      <p class="caption">Edificio 0</p>
      <img width="260" height="195" alt="imagen edificio0 " title="Edificio 0 de FCiencias" src="../..
      /images/galeria/edificio0.jpeg">
    </div>
  <div style="border-left: 1px solid rgb(77, 77, 77);">
      <p class="caption">Prometeo</p>
      <img width="260" height="195" alt="imagen prometeo " title="Fuente de la FCiencias" src="../../images
      /galeria/prometeo.jpeg">
    </div>
  <div style="border-right: 1px solid rgb(229, 229, 229);">
      <p class="caption">Tlahuizcalpan</p>
      <img width="260" height="195" alt="imagen tlahuizcalpan " title="Edificio Tlahuizcalpan" src="../..
      /images/galeria/tlahuizcalpan.jpeg">
    </div>
  <div style="border-left: 1px solid rgb(77, 77, 77);">
      <p class="caption">Tlahuizcalpan desde adentro</p>
      <img width="260" height="195" alt="imagen tlahuizcalpan_dentro " title="Tlahuizcalpan desde
      adentro" src="../../images/galeria/tlahuizcalpan_dentro.jpeg">
    </div>
</div>
```

(b) Resultado de aplicar agregarAlt().

Figura 2.101: galeria.html [Imagen creada por el autor de este trabajo (2011)] [Imágenes creadas por el autor de este trabajo (2011)]

1. Las reglas de estilo de la galería se localizan en el documento "css/galeria.css"

3.2.- Funciones en jQuery:

1. Implementar una función que obtenga los siguientes datos de las imágenes:
   a. El título de la imagen, obtenida del atributo "title".
   b. El texto alternativo de la imagen, obtenida del atributo "alt".
   c. Y la anchura y altura obtenidas de los atributos "width" y "height" respectivamente.
2. Utilizando el método html(), agregar los datos obtenidos de cada imagen en forma de lista debajo de ella, como se muestra en la figura 2.103.
3. Implementar un mecanismo para que una vez ejecutada la función se deshabilite, evitando agregar la misma información. Un posible resultado se muestra en la figura 2.103.
4. Modificar el ejercicio anterior para que siempre entregue los

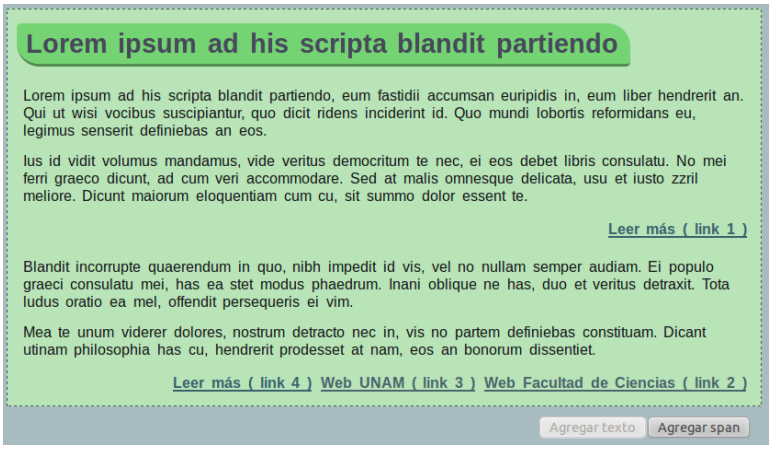


Figura 2.102: Resultado de aplicar las funciones en agregar_cont.html [Imagen creada por el autor de este trabajo (2011)]

datos reales sobre el width y height, aunque no hayan sido establecidos en los atributos.

Figura 2.103: Agregar información a una galería [Imagen creada por el autor de este trabajo (2011)]

4.- Resolver el ejercicio 3 utilizando un método y/o mecanismo diferente a .html() para conseguir al mismo resultado. Además implementar una función que desplace el párrafo con clase "caption" de cada "div" inmediato justo después de su elemento "img".

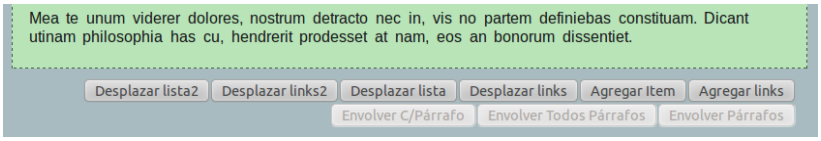
5.- En el documento "moverElem.html" implementar las siguientes funciones en jQuery (Un ejemplo del resultado solicitado se muestra en la colección 2.104):

5.1.- Implementar la función "disabledBotton(id)" que deshabilita el elemento cuyo valor de id coincide con el proporcionado en la

firma. Aplicar esta función para deshabilitar el botón de acción de las funciones relacionadas con agregar, desplazar y agrupar, una vez que estas se hayan ejecutado. El resultado deseado se muestra en la figura 2.104a

5.2.- Implementar el código de la función "envolverPs(id)" que agrupe a todos los párrafos inmediatos al "div" con clase "result" dentro de un "div" con clase "wrapPs", utilizando variaciones del método wrap(). El resultado deseado se muestra en la figura 2.104a

5.3.- Implementar el código de la función "envolverP(id)" que desempeña la misma tarea del ejercicio anterior, sin utilizar el método wrap() ni sus variantes.



(a) Deshabilitar los botones de acción.

```
<div class="wrapPs">
    <p> Lorem ipsum ad his scripta blandit partiendo, eum fastidii accumsan euripidis in, eum liber
    hendrerit an. Qui ut wisi vocibus suscipiantur, quo dicit ridens inciderint id. Quo mundi lobortis
    reformidans eu, legimus senserit definiebas an eos. </p>
    <p> Ius id vidit volumus mandamus, vide veritus democritum te nec, ei eos debet libris consulatu. No mei
    ferri graeco dicunt, ad cum veri accommodare. Sed at malis omnesque delicata, usu et iusto zzril meliore.
    Dicunt maiorum eloquentiam cum cu, sit summo dolor essent te. </p>
    <p> Blandit incorrupte quaerendum in quo, nibh impedit id vis, vel no nullam semper audiam. Ei populo
    graeci consulatu mei, has ea stet modus phaedrum. Inani oblique ne has, duo et veritus detraxit. Tota
    ludus oratio ea mel, offendit persequeris ei vim. </p>
    <p> Mea te unum viderer dolores, nostrum detracto nec in, vis no partem definiebas constituam. Dicant
    utinam philosophia has cu, hendrerit prodesset at nam, eos an bonorum dissentiet. </p>
  </div>
</div>
```

(a) Agrupar elementos como una sola entidad.

Figura 2.104: Desplazar y agrupar elementos en moverElem.html [Imágenes creadas por el autor de este trabajo (2011)]

6.- En el documento "piePagina.html" implementar las siguientes funciones en jQuery que modifican elementos descendientes al div.result (Un ejemplo del resultado solicitado se muestra en la figura 2.105):

6.1.- Referente a los estilos CSS

1. Los estilos referentes al pie de página se localizan en "css/piePagina.css". Analizar los estilos para tomarlos en cuenta al momento de implementar las funciones.
2. Si el alumno desea cambiar los estilo y/o agregar nuevos estilos es importante documentar esos cambios.

6.2.- Implementar una función que se ejecute al cargar el DOM de la página y construya notas numeradas al final del contenido del "div" con clase "result" (pie de página). Estas notas que en un

principio se localizan dentro de los párrafos se deben desplazar al final del contenido y dejar el número correspondiente a su cardinalidad con el cual se identifica a la nota en cuestión al final del "div". Para obtener el resultado deseado (ver figura 2.105a) se recomienda tomar encuenta los siguientes "tips".

1. Al final del contenido del "div" con clase "result", crear un "div" con clase "blockPiePagina" que contenga una lista ordenada sin elementos.
2. Desplazar cada nota a la lista ordena (contenida en blockPiePagina).
3. Envolver cada "span" en un elemento de lista.

6.3.- Implementar una función accionada con la anterior que permita:

1. Convertir el número dejado por el "span" con clase "piePagina" en un link interno que permita desplazar al lector a la nota en cuestión al dar clic.
2. Si el lector da clic en algún número de nota durante la lectura el sistema debe redireccionar a la nota en cuestión y aplicar una clase de CSS que la resalte.
3. El resaltado de notas debe ser excluyente, así que el sistema debe manejar el caso en cual si el lector visitó varias notas únicamente la más reciente deberá estar resaltada (ver figura 2.105b).

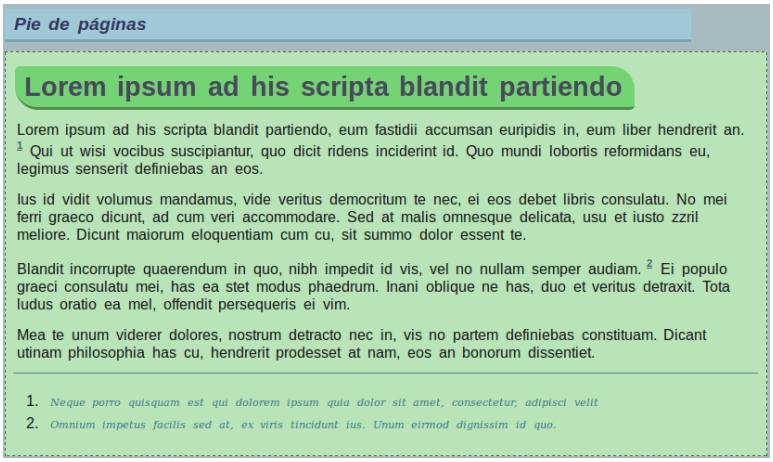
6.4.- Implementar una función vinculada con el sistema de pie de página, que permita el doble redireccionamiento, es decir, del identificador de nota contenido en el cuerpo del texto hacia la sección de notas al final de la página y viceversa. Esta función también debe ser excluyente en el resaltado.

7.- En el documento "eventos.html" implementar las siguientes funciones en jQuery (Un ejemplo del resultado solicitado se muestra en la figura 2.106):

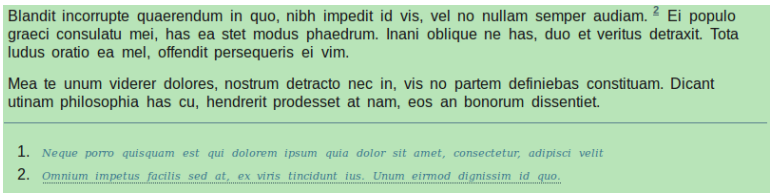
7.1.- Referente a los estilos CSS

1. Los estilos referentes a las listas y del historial se localizan en el "head" de la página.
2. Si el alumno desea cambiar los estilo y/o agregar nuevos estilos es importante documentar esos cambios.

7.2.- Implementar un código en el cual al pasar sobre los elementos de lista contenidos en el "div" con clase "result", se disparen los eventos mouseover, mouseout y click, ejecutando las siguientes acciones:



(a) Construcción de las notas al pie de página.



(b) Resaltado excluyente de las notas.

Figura 2.105: Sistema de pie de página. [Imágenes creadas por el autor de este trabajo (2011)]

1. Al pasar el mouse sobre el elemento se aplica un color de fondo, una separación interna y el texto se cambia a "negritas", como se muestra en la figura 2.106a.
2. Al salir del elemento su estilo de CSS regresa a la normalidad.
3. Al dar clic sobre el elemento en cuestión se aplica un color de fondo y de letra diferente y un borde seccionado, como se muestra en la figura 2.106a.
4. El código debe soportar que al dar clic sobre un elemento los eventos mouseover y mouseout queden desactivados.
5. Implementar un pequeño historial de los elementos de lista seleccionados en el orden en el que ocurrieron los clics, como se muestra en la figura 2.106a.

7.3.- Implementar las mismas funciones del ejercicio anterior utilizando la estructura:

```
1  .bind("mouseover mouseout click", function(e){});
```

7.4.- Al "div" con id "areaClic" implementar un código que maneje los

eventos necesarios para cumplir con (Un ejemplo del resultado solicitado se muestra en la figura 2.106b):

1. Al pasar el mouse sobre el rectángulo se actualizan las coordenas X e Y en tiempo real, en los "span" con id CooX y CooY respectivamente.
2. Al salir el puntero de "areaClic" las coordenadas se borran.
3. Implementar un pequeño historial de los clics ocurridos dentro de "areaClic", en el cual se registren las coordenadas del clic, como elementos de una lista ordenada.
4. Al dar clic sobre "areaClic" se debe mostrar un efecto de destello en dicha área.

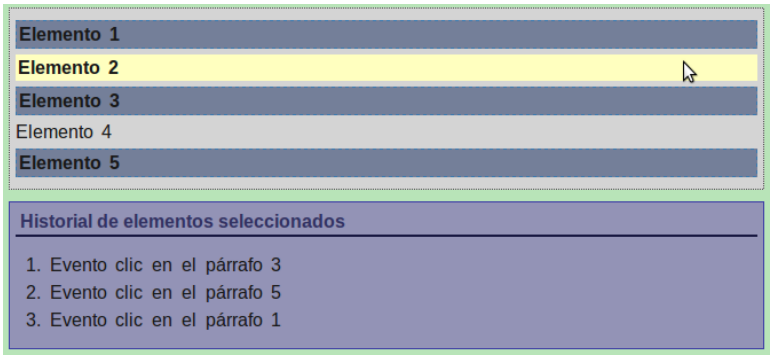
7.5.- Implementar una función genérica que capture un triple click sobre los elementos HTML. El tiempo entre cada clic no debe pasar de 500 milisegundos.

1. Crear un código que al ocurrir el evento "triple clic" en el "div" con id "ejemplo3C" se agregue un párrafo con la frase "Triple clic" en su interior, como se muestra en la figura 2.106c.
2. El contenedor con id "ejemplo3C" debe presentar un efecto de destello al ocurrir el evento.
3. Una vez ocurrido el "triple clic" y agregado el párrafo informativo se debe eliminar de forma automática pasados dos segundos (con efecto desvaneciente).

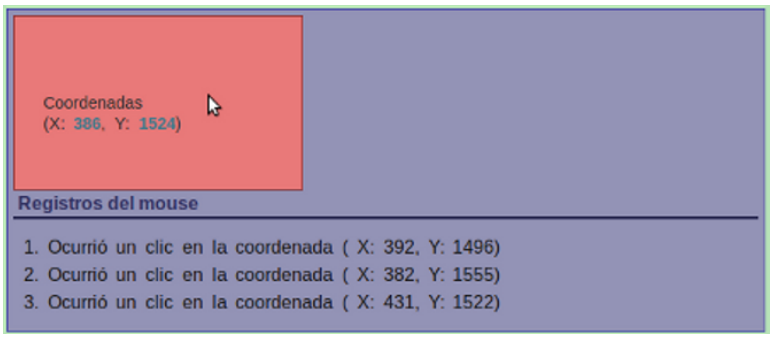
8.- En el documento "galeria_arreglos.html" implementar las siguientes funciones en jQuery (Un ejemplo del resultado solicitado se muestra en la figura 2.107):

8.1.- Implementar una función y funciones auxiliares necesarias que desempeñen:

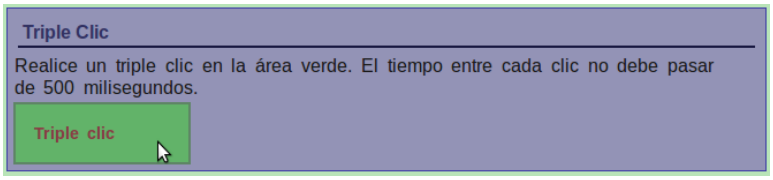
1. Al dar clic sobre una imagen se agrega como elemento de un arreglo el texto del párrafo con clase "caption" adyacente a la imagen que recibió el click.
2. La imagen que recibe un clic adquiere el siguiente estilo de CSS: borde punteado color azul y una opacidad del 60 % que se aplica lentamente. Además el párrafo con clase "caption" adyacente a la imagen en cuestión recibe un color de letra azul en negritas.
3. Si el botón de acción de cualquiera de la funciones involucradas con el arreglo de imágenes se encuentra deshabilitado, se debe detener el proceso de agregar elementos al arreglo junto con las acciones involucradas.



(a) Manejar 3 eventos múltiples.



(b) Clic y su registro.



(c) Triple clic.

Figura 2.106: Eventos [Imágenes creadas por el autor de este trabajo (2011)]

8.2.- Implementar una función que agregue los elementos del arreglo como elementos de una lista en el primer contenedor con clase "resTest".

8.3.- Implementar una función que agregue los elementos del arreglo sin repetición de entradas como elementos de una lista en el segundo contenedor con clase "resTest".

8.4.- Una vez accionada una función se debe deshabilitar su botón para evitar agregar información duplicada.

Figura 2.107: Convertir a un arreglo. [Imagen creada por el autor de este trabajo (2011)]

2.4.5. AJAX

AJAX es un conjunto de tecnologías que permiten crear aplicaciones Web interactivas. Las aplicaciones que utilizan AJAX se pueden comunicar con el servidor en segundo plano si necesidad de refrecar la página. Esta característica permite que las aplicaciones Web se vean como aplicaciones de escritorio.

Cuando la aplicación requiere información nueva para la página, se conecta en segundo plano al servidor y descarga los datos sin refrescar la página y con ayuda del DHTML (HTML dinámico) actuliza los elementos de la página con los datos descargos, también sin refrescar la página.

AJAX son las siglas de Asynchronous JavaScript and XML (JavaScript asíncrono + XML), en la práctica son un conjunto de tecnologías que se unen para realizar aplicaciones sorprendentes. Dentro de las tecnologías se encuentran:

- DHTML, XHTML y CSS.
- DOM. Para la manipulación de las estructuras de la página.
- JSON, XML y XSLT. Para realizar el intercambio de la información.
- XMLHttpRequest. Para realizar el intercambio asíncrono de la información.
- JavaScript une todas las tecnologías, permite el manejo del código en la página.

Las siglas de AJAX provienen del siguiente comportamiento: Las peticiones HTTP de la página al servidor se convierten en peticiones JavaScript. Las peticiones simples no requieren intervención del servidor por lo que las respuestas son inmediatas. Las peticiones que requieren del servidor se realizan de forma asíncrona a través de AJAX, en este caso el navegador no requiere refrescar la página para obtener siempre un contenido sin esperas largas.

Es XML por que originalmente cuando AJAX fue desarrollado los datos enviados desde el servidor estaban en formato XML, actualmente cualquier formato basado en texto funciona, esto se debe a que el HTTP que los navegadores usan para comunicarse con los servidores esta basada en texto.

2.4.5.1. AJAX estándar

Una aplicación AJAX sencilla consiste en una aplicación de JavaScript que descarga un archivo del servidor en segundo plano, después obtiene el contenido del archivo y lo carga en el cuerpo de la página sin refrescar.

En el cuadro de código 2.72 que es un fragmento del documento "AJAX/ajaxEstandar/AjaxEjemploSencillo.html", se muestra una aplicación sencilla de AJAX, la cual consiste en 4 etapas:

1.- **Instanciar el objeto XMLHttpRequest**.- Todas las aplicaciones que utilizan AJAX deben instanciar primero el objeto XMLHttpRequest, ya que es el objeto que permite realizar la comunicación en segundo plano con el servidor, sin recargar la página. En la línea de código 13 se crea el objeto XMLHttpRequest, pero este objeto depende de cada navegador. Para los navegadores que cumplen con los estándares (Firefox, Safari, Opera, Chrome, IE 7 y 8) se construye el objeto de forma nativa por lo que se obtiene a través del objeto window (línea de código 5) y para los navegadores obsoletos como IE 6 y anteriores, se construye como un objeto de tipo ActiveX (línea de código 8).

2.- **Asociar la función de respuesta**.- Una vez obtenida la instancia del objeto XMLHttpRequest, se asocia la función (líneas de código 17-22) que se encarga de procesar la respuesta del servidor. La propiedad "onreadystatechange" permite indicar la función directamente como una función anómina (línea 18) o indicar una referencia a una función independiente. Las líneas de código 17-22 indican que cuando reciban la respuesta del servidor se ejecuta el código definido.

3.- **Realizar la petición al servidor**.- En las líneas 16 y 23 se realiza la petición más sencilla que se puede enviar a un servidor. Se trata de una petición de tipo "GET" simple que no envía ningún parámetro al servidor. La petición HTTP se crea a través de la instrucción "open()" que recibe el tipo de petición en este caso GET y la URL del recurso

solicitado. Una vez creada la petición HTTP se envía al servidor a través del método send() (línea 23).

4.- **Ejecutar la función de respuesta**.- Por último cuando se recibe la respuesta del servidor se ejecuta la función establecida en la líneas 17-22. La función comprueba que se haya recibido la respuesta del servidor a través de la propiedad "readyState". Si ha recibido alguna respuesta comprueba que sea válida y correcta, esto lo hace comprobando que el código de estado HTTP devuelto sea igual a 200. Después de las comprobaciones agrega el contenido del texto recibido al div cuyo id coincida con el proporcionado en la firma del método.

Código 2.72: Ejemplo sencillo con AJAX

```
1  /*--- Código JavaScript usando AJAX ---*/
2  var peticionHttp;
3
4  function crearHttpRequestObject(){
5   if(window.XMLHttpRequest) {
6    return new XMLHttpRequest();
7   }
8   else if(window.ActiveXObject) {
9    return new ActiveXObject("Microsoft.XMLHTTP");
10  }
11 }
12 function cargarContenidoAjax(rutaRecurso, id){
13   peticionHttp = crearHttpRequestObject();
14   if(peticionHttp) {
15    var rutaCR = "http://"+location.host+"/jQuery
16    /ejercicios_resueltos/AJAX/"+rutaRecurso;
17    peticionHttp.open("GET", rutaCR);
18    peticionHttp.onreadystatechange = function() {
19     if (peticionHttp.readyState == 4 &&
20         peticionHttp.status == 200) {
21      var obj = document.getElementById(id);
22      obj.innerHTML = peticionHttp.responseText;
23     }
24    }
25    peticionHttp.send(null);
26  }
27 }
28
29 <!-- Código HTML -->
30 <div class="result">
31 [...]
32 <button type="button" name="cargarContAjax"
33 value="Cargar contenido" onClick=
34 "cargarContenidoAjax('ajaxEstandar/recursosTexto
```

```
35  /datosExternos.txt',‘contenidoExterno')">
36  Cargar contenido con AJAX</button>
37  [...]
38  </div>
```

2.4.5.2. Objeto XMLHttpRequest

El objeto XMLHttpRequest posee varios métodos y propiedades, a continuación se listan las propiedades del objeto XMLHttpRequest:

- **readyState**.- Valor numérico que almacena el estado de la petición.
  - **0**.- No inicializado, el objeto se ha creado pero no se ha invocado al método open().
  - **1**.- Cargando, el objeto se ha creado, pero no se ha invocado el método send().
  - **2**.- Cargado, se ha invocado el método send(), pero el servidor aún no ha respondido.
  - **3**.- Interactivo, se han recibido algunos datos, pero no se puede utilizar la propiedad responseText.
  - **4**.- Completo, se han recibido todos los datos de respuesta del servidor.
- **responseText**.- Es el contenido de la respuesta del servidor en forma de texto.
- **responseXML**.- Es el contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM.
- **status**.- Es el código del estado HTTP devuelto por el servidor, 200 para una respuesta correcta, 404 para "no encontrado", 500 para un error de servidor, etc.
- **statusText**.- El código del estado HTTP devuelto por el servidor en forma de texto, "OK", "Not Found", Internal Server Error", etc

Métodos del objeto XMLHttpRequest:

- **abort()**.- Detiene la petición actual.
- **getAllResponseHeaders()**.- Devuelve una cadena de texto con todas las cabeceras de la respuesta del servidor.
- **getResponseHeader("cabecera")**.- Devuelve una cadena de texto con el contenido de la cabecera solicitada.

- **onreadystatechange**.- Se invoca cada vez que se produce un cambio en el estado de la petición HTTP. Normalmente es una referencia a una función JavaScript.

- **open("metodo", "url")**.- Establece los parámetros de la petición que se realiza al servidor. Los parámetros necesarios son el método HTTP empleado y la URL del recurso.

- **send(contenido)**.- Realiza la petición HTTP al servidor. El argumento indica la información que se va a enviar al servidor junto con la petición HTTP. Si no se envían datos, se debe indicar un valor "null". En caso contrario se debe indicar una cadena de texto, un arreglo o un objeto DOM.

- **setRequestHeader("cabecera","valor")**.- Permite establecer cabeceras personalizadas en la petición HTTP. Antes se debe invocar al método open().

2.4.5.3. Envío de parámetros al servidor

El objeto XMLHttpRequest pertime enviar parámetros al servidor junto con la petición HTTP. Para ello utiliza los métodos GET y POST de HTTP. Con estos métodos se envían los datos como una serie de pares clave/valor concatenados por el símbolo &.

En el método GET los datos se concatenan a la URL presentando un límite en la cantidad de datos que se puede enviar de 512 bytes. Si se intenta enviar más de 512 bytes a través de la URL el servidor devuelve un error 414 con el mensaje "Request-URI Too Long" (La URI de la petición es demasiado larga). El método GET se utiliza cuando se accede a un recurso que requiere de la información proporcionada por el usuario.

En el método POST los parámetros se envían en el cuerpo de la petición. Este método se utiliza para crear o actualizar información

Si se utiliza un formulario HTML para obtener la información del usuario y enviarla al servidor se requiere construir la cadena de los parámetros manualmente.

A continuación se muestra un ejemplo del funcionamiento del envío de parámetros al servidor utilizando un formulario con 2 campos de texto cuyos valores se concatenan para formar la cadena de parámetros a enviar. El resultado de procesar la respuesta del servidor se carga en el "div" con id "respuesta". En el código 2.73 que es un fragmento del documento "ajaxEstandar/form.html", se muestra el formulario HTML para obtener los datos por parte del usuario y el "div" donde se cargará el texto devuelto.

Código 2.73: Formulario

```
<!-- Código HTML -->
[...]
<form class="esq_red">
<div>
<label for="nombre">Nombre:</label>
<input type="text" id="nombre" name="nombre"/><br/>
</div>
<div>
<label for="telefono">Teléfono:</label>
<input type="text" id="telefono" name="telefono"/><br/>
</div>
<input type="button" id="enviar" value="enviar datos" />
</form>
[...]
<div id="respuesta"></div>
```

En el código 2.74 se detalla el código JavaScript necesario pare realizar la conexión al servidor y enviar los parámetros en segundo plano utilizando un recurso en PHP que devuelve una cadena de texto que será cargada en el elemento con id "respuesta".

Código 2.74: Código JavaScript para el formulario

```
/*--- Código JavaScript usando AJAX ---*/
window.onload = function(){
document.getElementById("enviar").onclick = enviar;
}
var peticionHttp = null;
var locRecurso = "http://"+location.host+":8888/jQuery/
ejemplos/AJAX/ajaxEstandar/lib/";
function instanciaXHttpReques(){
 if(window.XMLHttpRequest){
   return new XMLHttpRequest();
 }
 else if(window.ActiveXObject){
   return new ActiveXObject("Microsoft.XMLHTTP");
 }
}
function crearCadParam(){
 var nombre = document.getElementById("nombre");
 var telefono = document.getElementById("telefono");
 return "nombre=" + encodeURIComponent(nombre.value)
        +&telefono=" + encodeURIComponent(telefono.value)
        +"&nocache=" + Math.random();
}
function enviar(){
  peticionHttp = instanciaXHttpReques();
  if(peticionHttp){
   peticionHttp.onreadystatechange = procesaRespuesta;
```

```
27      var rutaRec = locRecurso+"validaDatos.php";
28      peticionHttp.open("POST", rutaRec, true);
29      peticionHttp.setRequestHeader("Content-Type",
30      "application/x-www-form-urlencoded");
31      var cadPeticion = crearCadParam();
32      peticionHttp.send(cadPeticion);
33     }
34  }
35  function procesaRespuesta(){
36   if(peticionHttp.readyState == 4){
37    if(peticionHttp.status == 200){
38     document.getElementById("respuesta").innerHTML =
39     peticionHttp.responseText;
40     }
41    }
42  }
```

Con las líneas 1-3 se le indica al navegador que al cargar la página se vincule la función "enviar" al evento click del botón.

De la línea 8 a la 15 se indica la función encargada de instanciar el objeto XMLHttpRequest. Esta función es llamada en el cuerpo de la función "enviar" en la línea 24. En la línea 26 se indica la función "procesaRespuesta" que se encargada de procesar los datos de respuesta del servidor.

En la línea 28 se crea la petición HTTP utilizando el método POST e indicando la ruta del recurso del servidor el cual es un archivo en PHP que devolverá una cadena de texto. En las líneas 29 y 30 se establece la cabecera Content-Type, sin la cual el servidor descarta cualquier dato enviado por POST al servidor. Sin el Content-Type correcto la aplicación del servidor que espera trabajar con los parámetros no recibe ninguno. Para enviar parámetros por POST al servidor es obligatorio incluir la cabecera Content-Type usando la propiedad setRequestHeader del objeto XMLHttpRequest, en el ejemplo la instrucción queda así:

```
1  peticionHttp.setRequestHeader("Content-Type","application/
2  x-www-form-urlencoded");
```

De la línea 16 a la 22 se establece la función encargada de construir la cadena con los parámetros a enviar al servidor. En la función se utiliza el método "encodeURIComponent" que evita problemas con algunos caracteres especiales. La función reemplaza todos los caracteres que no se pueden utilizar de forma directa en las URL, por su representación hexadecimal. Las letras, números y los caracteres: {- _ . ! ~ * ( )} no se modifican, pero el resto de los caracteres se sustituyen por su equivalente en haxadecimal. JavaScript incluye una función que realiza la tarea inversa llamada "decodeURIComponent".

En la función encargada de crear la cadena de parámetros se concatena

un parámetro adicional llamado "nocache" con valor aleatorio, esta estrategia es muy común entre los desarrolladores para evitar problemas con la caché de los navegadores. Con este valor cambiante en cada petición se asegura que el navegador realice la petición directa al servidor sin hacer uso de su caché.

En el ejemplo el servidor devuelve una cadena con la información ingresada por el usuario en el formulario y enviada en segundo plano al servidor. En las aplicaciones reales se recomienda realizar dos validaciones a los datos ingresados por el usuario, una validación en la parte del usuario utilizando JavaScript y AJAX, y una en el servidor.

2.4.5.4. Enviar parámetros en formato XML

El objeto XMLHttpRequest además de permitir enviar parámetros en forma de cadena de texto, permite enviar parámetros al servidor en formato XML. En el siguiente ejemplo, se envían los datos en una cadena de texto que representa un documento XML.

A continuación se modifican partes del código del ejemplo anterior para que el envío de los datos ingresados por el usuario en el formulario se realice en forma de un documento XML junto a la petición HTTP.

El código 2.75 es el resultado de modificar la función encargada de crear y enviar la petición POST al servidor, así como de invocar la función encargada de construir la cadena de los parámetros a enviar en formato XML. Después la cadena de texto de los parámetros y la petición HTTP es enviada en segundo plano al documento "validaDatosXML.php" en el servidor.

Código 2.75: Función que envía la petición

```
1  /*--- Código JavaScript usando AJAX ---*/
2  function enviar() {
3   peticionHttp = instanciaXHttpRequest();
4    if(peticionHttp) {
5      peticionHttp.onreadystatechange = procesaRespuesta;
6      var rutaRec = locRecurso+"validaDatosXML.php";
7      peticionHttp.open("POST", locRecurso+
8      "validaDatosXML.php", true);
9      peticionHttp.setRequestHeader("Content-Type",
10     "application/x-www-form-urlencoded");
11     var cadXMLPeticion = 'cadXMLPeticion='+
12     crearCadXMLParam();
13     peticionHttp.send(cadXMLPeticion);
14   }
15 }
```

El código 2.76 se encarga de construir la cadena de los parámetros a enviar al servidor como representación de un documento XML. El uso del carácter \ en el cierre de las etiquetas XML es debido a que las etiquetas de

cierre de los documentos XML y HTML se interpretan en el mismo lugar en el que se encuentran, con lo cual al utilizar \ el código se valida siguiendo el estándar XHTML. En el código se construye un pequeño documento XML con etiqueta principal "parametros" que contiene el valor del nombre en la etiqueta "nombre" y el valor del teléfono en la etiqueta "telefono" valores ingresados por el usuario en el formulario HTML.

Código 2.76: Construcción del documento XML

```
/*--- Código JavaScript usando AJAX ---*/
function crearCadXMLParam() {
  var nombre = document.getElementById("nombre");
  var telefono = document.getElementById("telefono");
  var cadXML =
    "<parametros>"
      +"<nombre>"+nombre.value+"<\/nombre>"
      +"<telefono>"+telefono.value +"<\/telefono>"
    +"<\/parametros>";
  return escape(cadXML);
}
```

El código 2.77 que se encuentra en el documento "validaDatosXML.php", se encarga de convertir la cadena enviada junto con la petición en un objeto XML utilizando la función "simplexml_load_string" después obtiene los valores de los parámetros enviados. La segunda parte del código arma una cadena que representa un documento XML que almacena un mensaje y los valores obtenidos de la petición previamente analizada, con el proposito de reenviarla como respuesta a la página de origen.

Código 2.77: Obtener la datos XML en PHP

```
/*-- Código PHP  --*/
$cad = $_REQUEST['cadXMLPeticion'];
$xml = simplexml_load_string("<?xml version='1.0'
encoding='ISO-8859-1'?>".$cad);
  $nombre = $xml->nombre;
  $telefono = $xml->telefono;

$cadXML = "<?xml version='1.0' encoding='ISO-8859-1'?>"
."<respuesta>"
 ."<mensaje>Datos estructurados en el servidor</mensaje>"
 ."<nombre>". $nombre ."</nombre>"
 ."<telefono>".$telefono."</telefono>"
."</respuesta>";
echo $cadXML;
```

El código 2.78 se encarga de procesar la respuesta del servidor, para ello primero comprueba que se ha recibido la respuesta y que esta sea va-

lida. Después se procesa la respuesta como un elemento XML utilizando la propiedad "responseXML" y se accede a su contenido utilizando las funciones DOM. Trabajar la respuestas del servidor como documentos XML requiere de muchas funciones DOM convirtiendo el proceso en laborioso y pesado, pero muy útil para procesar respuestas complejas.

En la línea 6 se obtienen los elementos cuya etiqueta sea "mensaje" y se accede al primer elemento que coincide con esa etiqueta utilizando la propiedad "childNodes", luego se obtiene su valor utilizando la propiedad "nodeValue". El proceso es silimar para obtener el nombre y el teléfono ingresados por el usuario en el formulario. Para terminar se obtiene la referencia del "div" con id "respuesta" y se agrega como contenido la cadena de la respuesta procesada. El resultado se muestra en la figura 2.108.

Código 2.78: Procesa respuesta del servidor

```
1  /*--- Código JavaScript usando AJAX ---*/
2  function procesaRespuesta(){
3   if(peticionHttp.readyState == 4){
4    if(peticionHttp.status == 200){
5      var resXML = peticionHttp.responseXML;
6      var mensaje = resXML.getElementsByTagName
7        ('mensaje')[0].childNodes[0].nodeValue;
8      var nombre = resXML.getElementsByTagName
9        ('nombre')[0].childNodes[0].nodeValue;
10     var telefono = resXML.getElementsByTagName
11       ('telefono')[0].childNodes[0].nodeValue;
12     var cadRes = "Mensaje obtenido: "+ mensaje+
13       "<br>"+"Nombre: "+nombre +"<br>"+"Teléfono:"
14       +telefono+"<br>";
15     var agregarRes = document.getElementById
16       ("respuesta");
17     agregarRes.innerHTML = cadRes;
18     }
19   }
20 }
```

2.4.5.5. Respuesta en formato JSON

JSON es un formato compacto mucho más ligero que el formato XML, además es más sencillo de entender y de construir. El formato JSON es cada vez más soportado por los diferentes lenguajes del lado del servidor; como Java, PHP, C#, Perl, etc.

A continuación se modifica el código del ejemplo anterior para trabajar con la respuesta del servidor en formato JSON. El código 2.79 construye la petición POST, prepara la función encargada de procesar la respuesta del servidor e invoca la función encargada de crear la cadena de texto con los

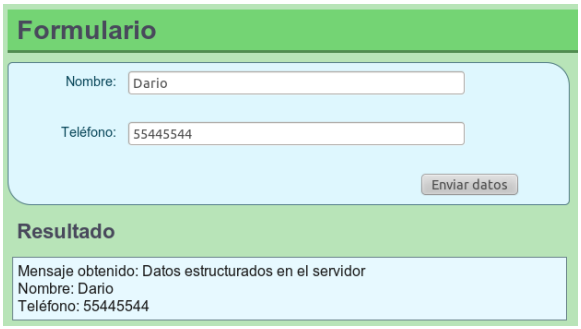


Figura 2.108: Envío de parámetros en formato XML [Imagen creada por el autor de este trabajo (2011)]

parámetros a enviar al servidor.

Código 2.79: Crear y enviar petición POST

```
/*--- Código JavaScript usando AJAX ---*/
function enviar() {
  peticionHttp = instanciaXHttpRequest();
  if(peticionHttp) {
   peticionHttp.onreadystatechange = procesaRespuesta;
   peticionHttp.open("POST", locRecurso+
     "validaDatosJSON.php", true);
   peticionHttp.setRequestHeader("Content-Type",
     "application/x-www-form-urlencoded");
   var cadPeticion = crearCadParam();
   peticionHttp.send(cadPeticion);
  }
}
```

El código 2.80 construye la cadena de parámetros a enviar al servidor como una serie de pares clave/valor concatenados con el símbolo & y utilizando la variable "nocache" para obligar al navegador a ir directamente al servidor.

Código 2.80: Construcción de parámetros a enviar

```
/*--- Código JavaScript usando AJAX ---*/
function crearCadParam(){
 var nombre = document.getElementById("nombre");
 var telefono = document.getElementById("telefono");
 return
   "nombre=" + encodeURIComponent(nombre.value)
  +"&telefono=" + encodeURIComponent(telefono.value)
  +"&nocache=" + Math.random();
}
```

El código 2.81 procesa la respuesta del servidor, para ello recibe los datos del servidor en forma de una cadena de texto que representa un formato JSON y envuelve la cadena entre paréntesis (para realizar la evaluación de forma correcta) después se utiliza la función "eval" que convierte la cadena en un objeto JSON. Una vez realizada la conversión los datos se pueden obtener con la notación punto, como si se tratarán de métodos o propiedades de JavaScript.

Código 2.81: Procesar respuesta JSON

```
/*--- Código JavaScript usando AJAX ---*/
function procesaRespuesta(){
 if(peticionHttp.readyState == 4) {
  if(peticionHttp.status == 200) {
    var resJSON = peticionHttp.responseText;
    var objetoJSON = eval("("+resJSON +")");
    var mensaje = objetoJSON.mensaje;
    var nombre = objetoJSON.nombre;
    var telefono = objetoJSON.telefono;
    var cadRes = "Mensaje obtenido: "+ mensaje
    + "<br>"+"Nombre: "+nombre +"<br>"
    +"Teléfono: "+telefono+"<br>";
    var agregarRes =
      document.getElementById("respuesta");
    agregarRes.innerHTML = cadRes;
  }
 }
}
```

2.4.5.6. AJAX con jQuery

AJAX es una herramienta que permite implementar aplicaciones ricas de Internet (RIA). Con AJAX la página Web del lado del cliente, se puede comunicar en segundo plano con el servidor y descargar recursos de forma asíncrona que posteriormente serán utilizados en la página.

Gracias al uso de AJAX los flashes o recargas constantes que realiza la página al traer y actualizar información son menos constantes, dejando la recarga de la página cuando realmente es necesaria.

Trabajar AJAX implementando el código en JavaScript desde cero involucra mucho código. Primero se debe instanciar el objeto XMLHttpRequest que es fundamental, ya que es el encargado de realizar la comunicación en segundo plano con el servidor. Después se requiere implementar y asociar la función encargada de procesar la respuesta del servidor. El siguiente paso consiste en crear la petición HTTP usando el método "open()" y enviar la petición utilizando el método "send()", junto con la petición se pueden enviar parámetros al servidor, por lo que se requiere implementar el código

encargado de armar los parámetros ya sea en forma de cadena de texto, cadena de texto que representa un documento XML o una cadena en formato JSON. Por último se ejecuta la función encargada de procesar la respuesta del servidor, primero se debe comprobar que se ha recibido la respuesta del servidor y que esta sea válida y correcta.

Trabajar AJAX con jQuery es mucho más sencillo, ya la función "load()" encapsula los pasos antes mencionado.

2.4.5.7. Función load jQuery

jQuery cuenta con una gran variedad de funciones relacionadas con AJAX siendo "load()" una de las más utilizadas.

.load(url, parametros, foo) método que descarga un recurso del servidor en segundo plano y carga su contenido en el elemento solicitante.

Donde "url" es la ruta del recurso en el servidor, "parametros" es un objeto JavaScript con los parámetros a enviar al servidor y "foo" es una función "callback"encargada de procesar la respuesta del servidor.

El código 2.82 es un fragmento del documento "ajaxjQuery/ejemploLoad.html" que ejemplifica el uso del método "load()", en el cual se descarga un documento de texto del servidor y carga su contenido en el "div" con id "contenidoExterno". El método recibe como parámetro la ruta del recurso a descargar y es accionado al dar clic sobre el botón "cargar contenido". Al lanzar la función se sustituye el contenido del "div" con la información obtenida del recurso como se muestra en la figura 2.109.

Código 2.82: Ejemplo de load()

```
/*--- Código jQuery AJAX ---*/
function cargarContenidoAjax(){
$("#contenidoExterno").load(locRecurso+
  'datosExternos.txt');
}
<!-- Código HMTL -->
[...]
<button type="button" name="cargarContAjax" value="Cargar
 contenido" onClick="cargarContenidoAjax()">Cargar
contenido</button>
 <div id="contenidoExterno">
   <p>Esperando contenido externo.</p>
</div>
```

jQuery proporciona métodos asociados un con el proceso AJAX, como "ajaxStart()" y "ajaxStop()". El primero ejecuta una función cuando inicia la petición al servidor y permite ejecutar una función mientras el proceso de

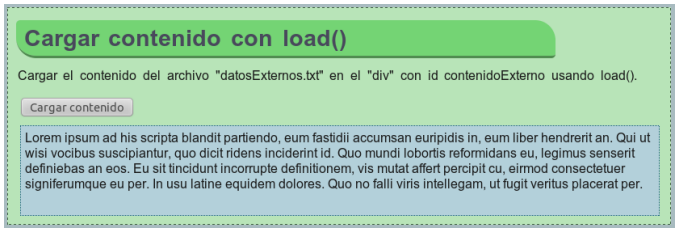


Figura 2.109: Ejemplo del uso de load() [Imagen creada por el autor de este trabajo (2011)]

comunicación y respuesta del servidor se lleva acabo. El segundo se ejecuta cuando finaliza la petición de datos al servidor.

.ajaxStart(foo)

.ajaxStop(foo)

El código 2.83 es una modificación del ejemplo anterior, el cual descarga un recurso en PHP y muestra su contenido en el "div" con id "contenidoExterno" el proceso de comunicación y respuesta del servidor debe presentar una demora de 1 segundo esto se consigue con la función "sleep()" que se presenta en el documento de PHP. Durante ese tiempo de demora la función "ajaxStart()" ejecuta la función "cargando()" que muestra un texto indicando que se está cargando un recurso desde el servidor.

Código 2.83: Ejemplo de load() con retraso

```
1  /*--- Código jQuery AJAX ---*/
2  function cargarContenidoAjaxSleep(){
3   $("#contenidoExterno").ajaxStart(cargando);
4   $("#contenidoExterno").load(locRecurso2 +
5    'datosExternos.php');
6  }
7  function cargando(){
8   $("#contenidoExterno").html("<p class='cargando'>
9     Cargando...</p>");
10  $("#contenidoExterno .cargando").css({color:'#2F748A',
11    fontSize:'1.2em'});
12 }
```

2.4.5.8. Práctica 04

Objetivos Aplicar los conocimientos adquiridos en el capítulo 2.4.5 AJAX, donde se tratan temas relacionados con AJAX estándar, el manejo de las propiedades y métodos proporcionados por el objeto XMLHttpRequest, las peticiones utilizando los métodos GET y POST, el envío de parámetros en formato XML y trabajar con respuestas en formato XML y JSON por parte del servidor.

Ejercicios Los documentos HTML de esta práctica se localizan en jQuery/ejercicios/AJAX y es necesario que todos los documentos de esta sección esten almacenados en un servidor con disponibilidad de PHP y permisos para obtener recursos de las páginas Web proporcionadas. Si el servidor no tiene disponible PHP, los scripts incluidos en los problemas requieren de conocimientos básicos de programación estructural y programación orientada a objetos, para convertirlos en instrucciones del lenguaje de lado del servidor de su preferencia.

1.- En el documento "ajaxEstandar/ejercicioAjax1.html" se debe implementar una aplicación de JavaScript que utilice AJAX y que cargue el contenido de un documento HTML o de algún recurso contenido en el servidor en una sección específica de la página, además debe mantener un registro de los estados de la petición. Un ejemplo del resultado solicitado se muestra en la figura 2.110). Para ello se debe cumplir con:

1.1.- Referente a los estilos CSS

1. Los estilos referente a los elementos contenidos en el div con clase "result" se localizan en la etiqueta style en el "head".

1.2.- Código JavaScript:

1. Al cargar la página, el campo de texto con id recurso debe tener como valor por defecto la URL del documento actual. También puede recibir una URL de algún recurso contenido en el servidor.
2. Al pulsar el botón "Cargar Contenido" se debe descargar el archivo solicitado por la URL proporcionada por el usuario o en su defecto tomar la ruta del documento actual (según el punto anterior), todo mediante peticiones AJAX, el contenido de la respuesta recibida del servidor se debe mostrar en el div con id "contenido".
3. Se debe mantener un registro de los estados de la petición, indicando el momento en milisegundos y el estado en el que se encuentra la petición. Los estados de petición regresan un

valor numérico por lo que se debe manejar ese comportamiento para relacionar esos índices con las frases; No inicializada, cargando, cargado, interactivo y completado.

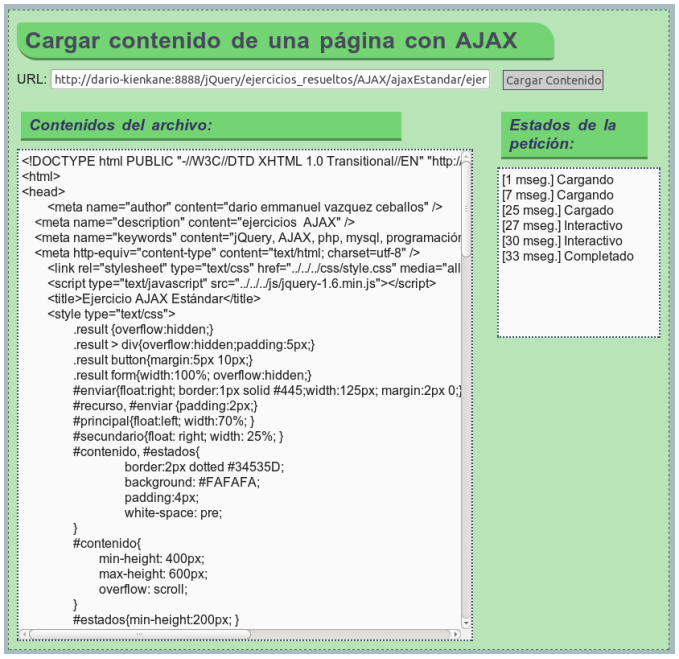


Figura 2.110: Cargar contenidos con AJAX [Imagen creada por el autor de este trabajo (2011)]

2.- Dados los documentos "ajaxEstandar/noticiasAjax.html" y "lib/generaNoticias.php", implementar el código necesario para cargar las noticias generadas por el documento "generaNoticias.php", en la sección con id "divNoticia" (un ejemplo del resultado solicitado se muestra en la figura 2.111). Para ello se debe cumplir con:

2.1.- Referente a los estilos CSS

1. Los estilos referente a los elementos contenidos en el "div" con clase "result" se localizan en el documento css/contenidoDinamico.css.

2.2.- Código JavaScript:

1. De forma periódica cada 4 segundos, se debe realizar una petición al servidor a través de AJAX, para obtener la noticia siguiente del arreglo de noticias. Esa noticia se debe cargar en la sección con id divNoticia.
2. Las noticias se van actualizando cada 4 segundos hasta traer todas las noticias del documento generaNoticias.php. Cuando

las noticias se agotan se debe deshabilitar el boton con id detener (ya que su funcionalidad pierde sentido).

3. Las noticias recibidas del servidor se deben almacenar, ya que cuando se pulsen los botones de anterior y siguiente, la petición al servidor para traer noticias se detiene y es posible navegar por las noticias almacenadas. Siguiendo la lógica de acción el botón anterior devuelve la noticia anterior y la carga en el div especificado y el botón siguiente realiza una operación similar pero con la noticia siguiente.
4. Si aún no se han obtenido todas las noticias y se accionan los botones de anterior o siguiente, la petición al sevidor se detiene y el botón "detener" pasa a poseer el valor de "iniciar", el cual debe realizar la tarea de reanudar las peticiones obteniendo la siguiente noticia, dada la última noticia solicitada. Además es posible navegar por las noticias almacenadas a través de los botones de anterior y siguiente.
5. Al cargar una noticia en el "div" con id divNoticia, se debe presentar un efecto de parpadeo o flash de color amarillo claro, el cual debe ocurrir en menos de un segundo. Este efecto también ocurre cuando se van pasando las noticias usando los botones anterior y siguiente..

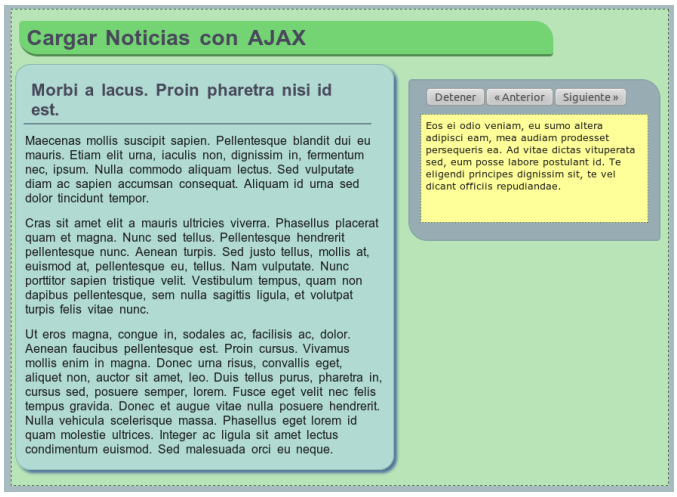


Figura 2.111: Cargar noticias con AJAX [Imagen creada por el autor de este trabajo (2011)]

3.- Dados los documentos "ajaxEstandar/form.html" y "lib/validaDatos.php", implementar el código necesario para que la aplicación valide con JavaScript que ningún campo del formulario se encuentre vacío al presionar el botón de enviar. Cuando los campos se encuentren llenos y el usuario

presione el botón de enviar, los datos se deben enviar al servidor en segundo plano usando el método POST, el documento validaDatos.php es el encargado de recibir los datos y construir una cadena de texto con esos datos, después debe reenviar la cadena de texto a la aplicación. Una vez devuelta esa cadena se carga en el "div" con id respuesta.

3.1.- Referente a los estilos CSS

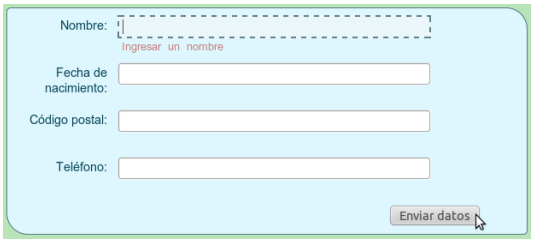
1. Los estilos referente a los elementos contenidos en el formulario se localizan en el documento css/formulario.css.

3.2.- El código en JavaScript debe presentar:

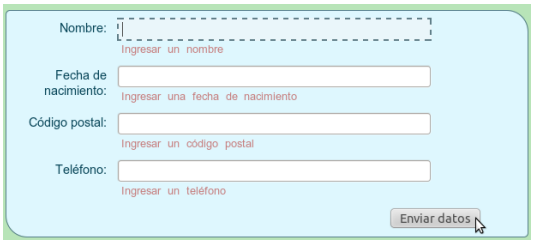
1. La función encargada que inicializar el objeto XMLHttpRequest dado el navegador del usuario.
2. La función encargada de crear la cadena de los parámetros concatenados utilizando una variable nueva llamada nocache, para obligar al navegador a realizar la petición directamente al servidor sin utilizar el caché.
3. La función que se encarga de procesar la respuesta del servidor y cargar el resultado en el "div" con id resultado.
4. La función encargada de validar que ningún campo del formulario este vacío.
   a. Para la primera versión cada campo de texto debe cumplir con: Si el campo está vacío se debe lanzar un cuadro de texto indicando que se debe ingresar un valor y posicionar el cursor en el campo correspondiente. Cuando un campo está vacío no se puede ir avanzando en la validación de los demás campos, es decir, debe ser una validación campo por campo. Cuando ya no existen campos vacíos se puede invocar a la función que envía la petición.
   b. Para la segunda versión cada campo de texto debe cumplir con: Si el campo está vacío se debe agregar un texto de advertencia en el "span" con id correspondiente al mensaje de error del campo encuestión (ej. para el campo nombre el "span" correspondiente tiene id "errorNombre") y colocar el cursor en el campo correspondiente. Debe ser una validación campo por campo. Cuando un campo deja de estar vacío el texto del mensaje de advertencia debe desaparecer. Cuando ya no existen campos vacíos se puede invocar a la función que envía la petición. El resultado esperado se muestra en la figura 2.112c
   c. Para la tercera versión se debe modificar la versión anterior para que la validación de los campos sea completa y mostrar el mensaje de error en todos los campos vacíos

en una sola ejecución. El resultado esperado se muestra en la figura 2.112b.

5. La función que crea y envía la petición HTTP utilizando el método POST.



(a) Validación campo por campo



(b) Validación completa

(c) Resultado

Figura 2.112: Enviar parámetros al servidor con POST [Imágenes creadas por el autor de este trabajo (2011)]

4.- Modificar los documentos "ajaxEstandar/form.html" y "lib/validaDatos.php", para enviar los parámetros al servidor en formato XML. Las funciones que validan los campos vacíos deben funcionar correctamente.

5.- Modificar los documentos "ajaxEstandar/form.html" y "lib/validaDatos.php", para procesar la respuesta del servidor en formato JSON. Las funciones que validan los campos vacíos deben funcionar correctamente.

6.- Dados los documentos "ajaxjQuery/ejercicioLoad.html" y "recursosTexto/datosExternos.txt", implementar el código necesario para que la aplicación descargue el recurso de texto del servidor y cargue su contenido en "div" con id "contenidoExterno" sin recargar la página.

6.1.- Referente a los estilos CSS

1. Los estilos referente a los elementos contenidos en el "div" con clase "result" se localizan en la etiqueta style en el "head".

6.2.- El código en jQuery debe presentar:

1. Antes de presionar el botón se muestra un texto que indica que aun no se ha realizado la petición.
2. Cuando se lanza la petición el texto inicial desaparece y se debe mostrar una imagen de "cargando" centrada en el "div" con id "contenidoExterno" el proceso de la petición deben presentar un retardo de 2 segundos (independientemente de lo que demore el proceso AJAX). Una vez cargado el texto la imagen desaparece y el nuevo contenido reemplaza al anterior.
3. Se puede utilizar la imagen "images/ajax-load.gif" para el ejercicios o cargar el recurso gráfico de su preferencia. El centrado de la imagen se puede realizar con JavaScript y/o CSS.

2.4.6. Complementos

2.4.6.1. DOM

Document Object Model, es un conjunto de funciones y utilidades que permiten manipular los documentos XML, XHTML y HTML de forma rápida y eficiente.

Antes de emplear las funciones en una página, DOM transforma el documento original en una estructura jerárquica de nodos. De esta forma es más sencillo acceder a los elementos de una página y manipularlos.

Por ejemplo en una página HTML se genera una estructura de nodos donde el nodo raíz es un nodo de tipo documento HTML. A partir de ahí se derivan dos nodos con el mismo nivel, el primero dado por la etiqueta $< head >$ y el segundo por $< body >$. De estas etiquetas se derivan otras y así sucesivamente.

Una vez construida dicha estructura se pueden utilizar las funciones del DOM, por esto es imprescindible que la página junto con todos sus elementos (imágenes, tablas, entre otros recursos) hayan sido cargados.

A continuación se muestran los tipos de nodos más importantes generados a partir de un documento:

- **Document**.- Es el nodo raíz de todos los documentos XML, XHTML y HTML. De él derivan el resto de los nodos.
- **DocumentType**.- Contiene la representación del DTD, determinado por el DOCTYPE.
- **Element**.- Representa el contenido definido entre las etiquetas de apertura y de cierre.

- **Attr**.- Representa el par atributo/valor.
- **Text**.- Almacena el texto que se encuentra entre las etiquetas de apertura y de cierre.
- **CDataSection**.- Representa una sección de tipo $<![CDATA[]] >$.
- **Comment**.- Representa un comentario de XML.

Una vez formada la estructura de nodos es posible utilizar las funciones para manipular el contenido de los nodos. Para ello JavaScript crea el objeto "Node" que contiene propiedades y métodos para procesar y manipular los contenidos y estructuras de una página.

El objeto "Node" define las siguientes constantes para identificar a los distintos tipos de nodos.

- Node.ELEMENT_NODE = 1
- Node.ATTRIBUTE_NODE = 2
- Node.TEXT_NODE = 3
- Node.CDATA_SECTION_NODE = 4
- Node.ENTITY_REFERENCE_NODE = 5
- Node.ENTITY_NODE = 6
- Node.PROCESSING_INSTRUCTION_NODE = 7
- Node.COMMENT_NODE = 8
- Node.DOCUMENT_NODE = 9
- Node.DOCUMENT_TYPE_NODE = 10
- Node.DOCUMENT_FRAGMENT_NODE = 11
- Node.NOTATION_NODE = 12

Además de las constantes el objeto "Node" define las siguientes propiedades y métodos. Estos métodos son específicos de XML, pero se pueden aplicar a los documentos XHTML. Para los documentos HTML los navegadores lo tratan como si fuera XML.[15]

- **nodeName**.- Devuelve una cadena con el nombre del nodo.

[15]Existen extensiones específicas para XHTML y HTML

- **nodeValue**.- Devuelve una cadena con el valor del nodo (no está definido para algunos tipos de nodo).
- **nodeType**.- Devuelve el número correspondiente con el tipo de nodo dada la lista anterior de constantes.
- **ownerDocument**.- Devuelve una referencia del documento al que pertenece el nodo.
- **firstChild**.- Devuelve el primer nodo de la lista childNodes (o la referencia al primer nodo).
- **lastChild**.- Devuelve el último nodo de la lista childNodes (o la referencia al último nodo).
- **childNodes**.- Devuelve una lista de todos los nodos hijo del nodo actual.
- **previousSibling**.- Devuelve la referencia del nodo hermano anterior o null si este nodo es el primer hermano.
- **nextSibling**.- Devuelve la referencia del nodo hermano siguiente o null si este nodo es el último hermano.
- **hasChildNodes()**.- Devuelve true si el nodo actual tiene uno o más nodos hijo.
- **attributes**.- Se emplea con nodos de tipo Element. Contiene objetos de tipo Attr que definen todos los atributos del elemento.
- **appendChild(nodo)**.- Añade un nuevo nodo al final de la lista childNodes.
- **removeChild(nodo)**.- Elimina un nodo de la lista childNodes.
- **replaceChild(nuevoNodo, anteriorNodo)**.- Reemplaza el nodo "anteriorNodo" por el nodo "nuevoNodo".
- **insertBefore(nuevoNodo, referenciaNodo)**.- Inserta el nodo "nuevoNodo" antes de la posición del nodo "referenciaNodo" dentro de la lista "childNodes".

Interacción DOM con HTML

El uso de DOM está limitado a las posibilidades que ofrece el navegador con el cual accede el usuario a la página. Los navegadores que implementan las especificaciones del DOM en niveles 1, 2 y parte del 3, son Safari, Firefox

y Chrome [16], mientras que otros navegadores como Internet Explorer con versiones de la 7 y anteriores, no son capaces de ofrecer una implementación completa de la versión DOM nivel 1.

Para los siguientes ejemplos se puede utilizar cualquiera de los tres navegadores antes mencionados que soportan las especificaciones DOM nivel 1 y 2, o si lo prefiere investigar si el navegador de su preferencia le ofrece un buen soporte.

En el cuadro de código 2.84 que es un fragmento del documento "complementos/EjemplosDOM.html" se muestra la forma de acceder a algunos nodos, por ejemplo la variable "nodoRaiz" almacena el elemento raíz de la página HTML y muestra el tipo de nodo utilizando el operador "nodoRaiz.nodeName", que despliega el tipo "HTML". Para acceder a los hijos del nodo raíz (sus dos nodos descendientes) se utiliza el operador childNodes[i] con i tomando los valores de 0 y de 1, para acceder a los nodos "head" y "body" respectivamente. El operador "length" permite conocer la cantidad de nodos hijo con los que dispone un nodo utilizando "childNodes", el ejemplo despliega la cantidad de nodos hijo del nodo "body".

Código 2.84: Acceso a nodos

```
/*--- Código jQuery con DOM ---*/
$(document).ready(function(){
  var nodoRaiz = document.documentElement;
  alert(nodoRaiz.nodeName);
  //para acceder a los nodos hijos de la raíz
  var primerNodo =  nodoRaiz.childNodes[0];
  var segundoNodo = nodoRaiz.childNodes[1];
  alert(primerNodo.nodeName);
  alert(segundoNodo.nodeName);
  // acceso directo al nodo body
  var nodoBody = document.body;
  alert(nodoBody.nodeName);
  //despliega la cantidad de nodos hijo del body
  alert(nodoBody.childNodes.length);
)}
```

Los nodos de tipo "Element" contienen la propiedad "attributes" que permite acceder a los atributos de cada elemento. DOM proporciona métodos para operar con los atributos de los elementos.

- **getNamedItem(nombre)**.- Devuelve el nodo cuya propiedad "nodeName" coincida con el valor "nombre".
- **removeNamedItem(nombre)**.- Elimina el nodo cuya propiedad "nodeName" coindica con el valor "nombre".

[16] Son los navegadores más importantes en mantener un soporte y una constancia de actualización de los estándares de la W3C

- **setNamedItem(nodo)**.- Agrega el nodo a la lista "attributes".
- **item(posicion)**.- Devuelve el nodo cuya posición coincida con el valor numérico "posicion".
- **getAttribute(nombre)**.- Equivalente a getNamedItem(nombre).
- **setAttribute(nombre, valor)**.- Aplicar el valor indicado al atributo que coincida con "nombre".
- **removeAttribute(nombre)**.- Equivalente a removeNamedItem(nombre).

El cuadro de código 2.85 que es un fragmento del documento "complementos/EjemplosDOM.html" ejemplifica la forma de acceder al valor del atributo "id" del "div" inmediato al "body" y al valor del "id" del primer "h12 inmediato al "div" con id "container". Para terminar se agrega la clase "esq_red" al primer "h1" inmediato al "container" y se utiliza el método "removeAttribute()" para eliminar su atributo id.

Código 2.85: Atributos de los nodos

```
1  /*--- Código jQuery con DOM ---*/
2  $(document).ready(function(){
3    var divNodoBody= document.body.childNodes[1];
4    alert(divNodoBody.getAttribute(?'id'));//devuelve
       container
5    //accede al primer titular h1
6    var h1Container = divNodoBody.childNodes[1];
7    alert(h1Container.getAttribute('id'));//devuelve
       titularPrincipal
8    //agregar la clase esq_red al primer titular h1
9    h1Container.setAttribute('class', 'esq_red');
10   //Elimina el atributo id del primer h1
11   h1Container.removeAttribute('d');
12 })
```

Además de los métodos anteriores para acceder a los nodos, DOM proporciona métodos de acceso más directos.

- **getElementsByTagName("nombreEtiqueta")**.- Devuelve un arreglo con todos los elementos de la página cuyo tag de HTML coincide con "nombreEtiqueta".
- **getElementsByName("AtributoName")**.- Devuelve un arreglo con todos los elementos de la página cuyo valor del atributo "name" coincide con "AtributoName". Este método es muy útil para acceder a todos los elementos radiobutton que están relacionados.

- **getElementById('valorId')**.- Devuelve el elemento cuyo valor de "id" coincide con "valorId". Como el valor de id es único para cada página HTML, es el método indicado para elegir un elemento concreto de la página.

Además de los métodos DOM para acceder a los nodos y a sus propiedades. DOM proporciona métodos para crear, eliminar y modificar nodos.

- **createAttribute("nombre")**.- Crea un nodo de tipo attributo con el nombre indicado.
- **createComment("texto")**.- Crea un nodo del tipo comentario con el contenido "texto".
- **createElement("nombreEtiqueta")**.- Crea un elemento del tipo indicado con "nombreEtiqueta".
- **createTextNode("texto")**.- Crea un elemento de tipo texto con el contenido indicado en "texto".

Ejercicios DOM

A partir de la página "complementos/EjemplosDOM.html" y utilizando las funciones DOM, implementar las funciones que se solicitan:

1.- Implementar una función que despliegue en un cuadro de diálogo la cantidad de párrafos en la página.

2.- Implementar una función que despliegue en un cuadro de diálogo la cantidad de links de la página.

3.- Implementar una funcion que despliegue en un cuadro de diálogo la siguiente información sobre los párrafos.

   3.1.- Donde direcciona el primer link.

   3.2.- Cuántos links direccionan a http://EjemplosDOM.html/.

   3.3.- Hacer que todos los links abran en una ventana nueva y desplegar en un cuadro de diálogo que los links presentan dicho atributo.

4.- Implementar una función que agregue párrafos al final del documento de forma dinámica sin importar el contenido.

5.- Implementar una función que agregue la clase de estilo "parrafoImpar" a los párrafos impares. La clase implica un color de fondo #FBD58E y un estilo de letra en cursiva y en negritas.

6.- Implementar una función que agregue el texto "Texto agregado con métodos DOM", a los párrafos pares de la página.

7.- Implementar una función que elimine cada tercer párrafo de la página.

8.- Aplicar un borde de 1px de estilo sólido color #4C764C a los "div" inmediatos al "div" con id "container".

2.4.6.2. BOM

El Browser Object Model, es un concepto introducido por los navegadores IE y Netscape Navigator ambos en su versión 3, que permite acceder y modificar las propiedades de las ventanas del navegador (como redimensionar y mover), modificar el texto de la barra de estado, entre otras más. Estas operaciones no están relacionadas con el contenido.

El principal inconveniente del BOM es que no existe alguna entidad encargada de estandarizarlo en los diferentes navegadores.

A continuación se listan algunas operaciones y características del BOM:

1.- Permite crear, mover, redimensionar y cerrar ventanas.

2.- Obtener información del navegador.

3.- Administrar las cookies.

4.- Manejo de objetos ActiveX para IE.

5.- Conocer propiedades de la pantalla del usuario (como su resolución).

Cuando se carga la página se genera el objeto window, que esta formado por varios objetos relacionados entre sí; document, location, navigator y screen.

Objeto window

El objeto window permite mover, redimensionar, manipular, abrir y cerrar ventanas de navegador. Cada ventana de navegador genera su propio objeto window.

En el código JavaScript no se requiere indicar de forma explícita el objeto window ya que todos los objetos derivan de él. Así para acceder al objeto se puede prescindir de la palabra window, como se muestra en el cuadro de código 2.86.

Código 2.86: Objeto window

```
1 window.document == document
2 window.navigator == navigator
```

A continuación se listan algunos métodos proporcionados por el objeto window:

- **open(url, target, propiedades)**.- Permite crear una nueva ventana de navegador. Donde "url" indica la dirección del documento, "target" indica el valor que utiliza la propiedad TARGET y "propiedades" puede tomar elementos de la siguiente lista, separados por comas.
  - toolbar [=yes/no]
  - location [=yes/no]
  - status [=yes/no]
  - menubar [=yes/no]
  - scrollbars [=yes/no]
  - resizable [=yes/no]
  - width = píxeles
  - height = píxeles
- **close()**. Cierra la ventana actual. Despliega una ventana de confirmación.
- **alert()**.- Despliega una ventana de diálogo.
- **prompt(mensaje,valorDefecto)**.- Despliega una ventana con un campo de texto para llenar, en el que el usuario puede teclear. El valor incial del campo es el especificado por "valorDefecto". Cuando el usuario acepta la cadena es almacenada de lo contario devuelve el valor null.
- **[identificador =]setTimeout(funcion,tiempo)** .- Ejecuta la función cuando hayan transcurrido el tiempo especificado en milisegundos.
- **clearTimeout(identificador)**.- Detiene el proceso anterior.
- **scroll(x,y)**.- Desplaza el documento en la ventana hasta la posición especificada por las coordenadas "x" e "y", que indican el desplazamiento horizontal y vertical respectivamente en píxeles.
- **moveBy(x,y)**.- Desplaza la posición de la ventana "x" píxeles a la derecha e "y" píxeles hacia abajo. Permite desplazamientos negativos.
- **moveTo(x,y)**.- Desplaza la ventana del navegador hasta que la esquina superior izquierda se localice en la posición (x,y) de la pantalla. Permite desplazamientos negativos.
- **resizeBy(x,y)**.- Redimensiona la ventana del navegador de forma que su anchura sea igual a la anchura original más "x" píxeles y su nueva altura sea igual a la altura original más "y" píxeles. Permite valores negativas.
- **resizeTo(x,y)**.- Redimensiona la ventana del navegador hasta que su anchura sea igual a "x" y su altura sea igual a "y". No permite valores negativos.

Objeto document

El objeto document pertenece tanto al BOM como al DOM[17]. Este objeto proporciona información sobre la página Web.

A continuación se listan las propiedades más importantes definidas por el objeto document.

- **referrer**.- Es la URL desde la cual se accedió a la página. (Es la página anterior en el arreglo history)
- **title**.- El contenido de la etiqueta $< title >$
- **URL**.- La URL de la página actual.
- **lastModified**.- La fecha de la última modificación de la página.

La importancia del objeto document radica en que hace referencia a objetos image, form, link, entre otros, los cuales permiten acceder a las imágenes, campos de formulario y a los enlaces de la página respectivamente. A continuación se muestran estos arreglos:

- **images**.- Contiene todas las imágenes de las página. Permite acceder al valor de las propiedades de las imágenes como es el "src".
- **forms**.- Contiene todos los formularios de la página. Permite aplicar métodos a cada formulario del arreglo.
- **links**.- Contiene todos los enlaces de la página del tipo "href".
- **anchors**.- Contiene todas las anclas de la página, del tipo name = "nombreAncla" (referencias internas).
- **applets**.- Contiene todos los applets de la página.

Objeto navigator

Este objeto permite obtener información sobre el propio navegador. Lamentablemente es uno de los objetos menos estandarizados, pero algunas de sus propiedades son comunes en casi si todos los navegadores. El objeto navigator se emplea para detectar el tipo y versión del navegador con el cual accede el usuario, así como para detectar si el navegador tiene habilitado java y las cookies.

A continuación se muestran algunas de esas propiedades:

- **appName**.- Contiene el nombre oficial del navegador.
- **appCodeName**.- Contiene el nombre del navegador.

[17] Para más información sobre el DOM revisar la sección 2.4.6.1 DOM

- **appVersion**.- Contiene la versión del navegador.
- **browserLanguage**.- Contiene el idioma del navegador.
- **cookieEnabled**.- Contiene un valor boolean que indica si las cookies están habilitadas
- **cpuClass**.- Sólo para IE, indica el tipo de CPU del usuario (ej. x86, PPC, other).
- **javaEnabled**.- Contiene un valor boolean que indica si java está habilitado.
- **mimeTypes**.- Arreglo que contiene los tipos MIME registrados por el navegador.
- **platform**.- Contiene la plataforma sobre la que se ejecuta el navegador.
- **plugins**.- Arreglo que contiene los plugins instalados en el navegador.
- **preference()**.- Sólo para Firefox, método empleado para establecer preferencias en el navegador.
- **oscpu**.- Sólo para Firefox, contiene el sistema operativo o CPU.

Objeto screen

Se utiliza para obtener información sobre la pantalla del usuario. El dato más importante que proporciona es la resolución del monitor del usuario.

A continuación se muestran algunas propiedades disponibles del objeto screen:

- **height**.- Altura de la pantalla en píxeles.
- **width**.- Anchura de la pantalla en píxeles.
- **colorDepth**.- Profundidad del color de la pantalla.
- **availwidth**.- Anchura de la pantalla disponible para las ventanas.
- **availHeight**.- Altura de la pantalla disponible para las ventanas.

Ejercicios BOM

A partir de la página "complementos/EjemplosBOM.html" y utilizando las funciones BOM, implementar las funciones que se solicitan:

1.- Implementar una función que desplace la ventana del navegador "x" píxeles a la derecha e "y" píxel hacia abajo, tomando como referencia la esquina superior izquierda. Los valores de "x" e "y" son ingresados por el usuario.

2.- Implementar una función que cambie la anchura y altura de la ventana a partir de los valores ingresados por el usuario a través de ventanas desplegables.

3.- Implementar una función que redimensione la ventana con los valores ingresados por el usuario a través de ventanas desplegables.

4.- Implementar una función que modifique el título de la página y despliegue un cuadro de diálogo mostrando el anterior título y el nuevo que lo sustituye.

5.- Implementar una función que despliegue la fecha de la última modificación de la página.

6.- Implementar una función que despliegue las urls de los links de la página.

7.- Implementar una función que redimensione la ventana del navegador al máximo tamaño posible según la pantalla del usuario.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.
Miguel de Cervantes Saavedra

[1] Steven Holzner, *Visual Quickstart Guide jQuery*, Peachpit Press, (2009)

[2] B.M. Harwani, *jQuery Recipes*, Apress, (2010)

[3] jQuery, *jQuery API*, http://api.jquery.com/ , (2012)

[4] Arman Danesh, *JavaScript in 10 Simple Steps or Less*, Wiley Publishing, Inc., (2004)

[5] Jason van Gumster and Robert Shimonski, *GIMP Bible*, Inc. Wiley Publishing, (2010)

[6] Philip Andrews, *The Adobe Photoshop CS4 Dictionary*, Focal Press, (2009)

[7] Richard Lynch, *The Adobe Photoshop CS4 Layers Book*, Focal Press, (2009)

[8] Javier Eguíluz Pérez, *Introducción a CSS*, librosweb.es, (2009)

[9] Thomas Powell, *Web Design: The Complete Reference Second Edition*, McGraw-Hill/Osborne, (2002)

[10] Wilbert O. Galitz, *The Essential Guide to User Interface Design*, John Wiley Sons, Inc., (2002)

Acrónimos y glosario

AJAX ..... *(Asynchronous JavaScript And XML)* Javascript Asíncrono y XML

CHSS ..... *Cascading HTML Style Sheets*

CSS ..... *Cascading Style Sheets*, hojas de estilo en cascada

DOM ..... *(Document Object Model)* Modelo de Objetos del Documento.

FEEDBACK ..... Retroalimentación

GIMP ..... *GNU Image Manipulation Program*, software libre de edición de imágenes.

HTTP ..... *Hypertext Transfer Protocol*, Protocolo de transferencia de hipertexto

JAVA ..... Lenguaje de programación orientado a objetos.

JAVASCRIPT ... Lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

JQUERY ..... Biblioteca de JavaScript.

LAYOUT ..... *Layout*, Diseño general de la plantilla web

OMG ..... Object Management Group

PHOTOSHOP ... *Adobe Photoshop*, software de edición de imágenes.

PHP ..... *Hypertext Pre-processor, (Personal Home Page Tools).* Lenguaje de programación de script del lado del servidor diseñado para el desarrollo web de contenido dinámico.

RUBY ..... Lenguaje de programación interpretado, reflexivo y orientado a objetos.

RUP ..... Proceso Unificado Racional

SSP ..... *Stream-based Style Sheet Proposal*

UML ..... *Unified Modeling Language*, Lenguaje Unificado de Modelado

URI ..... *Uniform Resource Identifier*, Identificador uniforme de recurso

URL ..... *Uniform Resource Locator*, Localizador de recursos uniforme

VIEW TEMPLATE *View template*, Diseño de una vista web

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes