

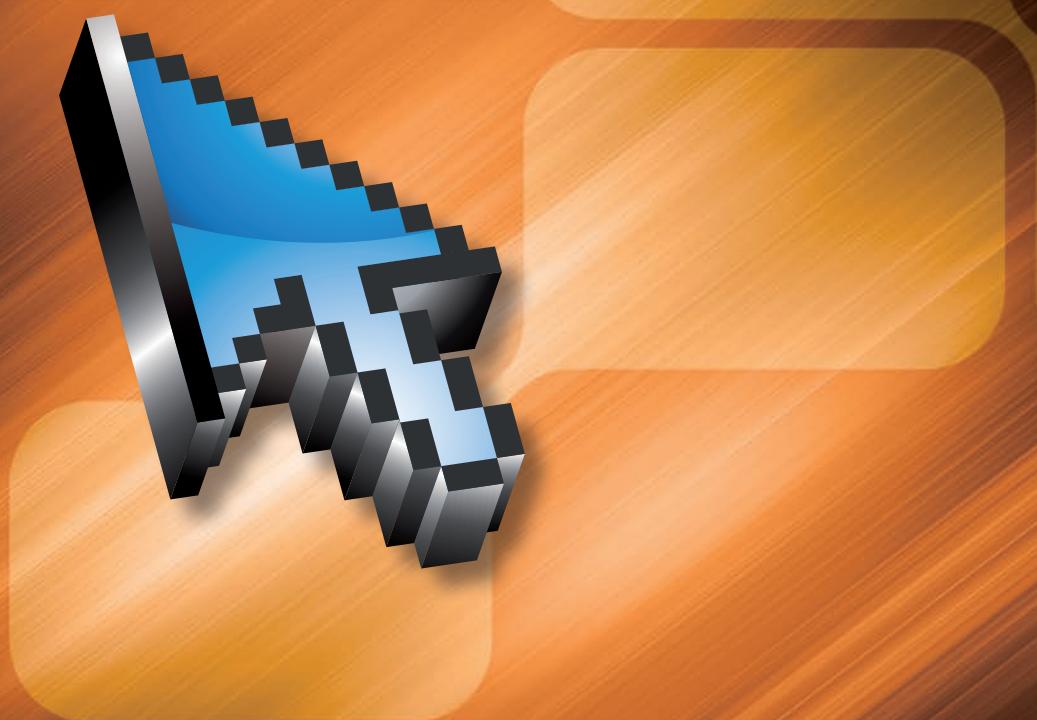
CF

GRADO SUPERIOR

CICLOS FORMATIVOS

R.D. 1538/2006

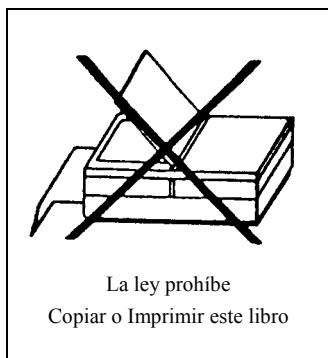
Diseño de Interfaces Web



**JOSE EDUARDO CÓRCOLES TENDERO
FRANCISCO MONTERO SIMARRO**

 **Ra-Ma®**

www.ra-ma.es/cf



La ley prohíbe
Copiar o Imprimir este libro

DISEÑO DE INTERFACES WEB

© José Eduardo Córcoles Tendero, Francisco Montero Simarro

© De la Edición Original en papel publicada por Editorial RA-MA

ISBN de Edición en Papel: 978-84-9964-154-6

Todos los derechos reservados © RA-MA, S.A. Editorial y Publicaciones, Madrid, España.

MARCAS COMERCIALES. Las designaciones utilizadas por las empresas para distinguir sus productos (hardware, software, sistemas operativos, etc.) suelen ser marcas registradas. RA-MA ha intentado a lo largo de este libro distinguir las marcas comerciales de los términos descriptivos, siguiendo el estilo que utiliza el fabricante, sin intención de infringir la marca y solo en beneficio del propietario de la misma. Los datos de los ejemplos y pantallas son ficticios a no ser que se especifique lo contrario.

RA-MA es una marca comercial registrada.

Se ha puesto el máximo empeño en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso ni tampoco de cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa o de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro sin autorización previa y por escrito de RA-MA; su contenido está protegido por la Ley vigente que establece penas de prisión y/o multas a quienes, intencionadamente, reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica.

Editado por:

RA-MA, S.A. Editorial y Publicaciones
Calle Jarama, 33, Polígono Industrial IGARSA
28860 PARACUELLOS DE JARAMA, Madrid
Teléfono: 91 658 42 80
Fax: 91 662 81 39
Correo electrónico: editorial@ra-ma.com
Internet: www.ra-ma.es y www.ra-ma.com

Maquetación: Gustavo San Román Borrueco
Diseño Portada: Antonio García Tomé

ISBN: 978-84-9964-363-2

E-Book desarrollado en España en Septiembre de 2014

Diseño de Interfaces Web

JOSÉ EDUARDO CÓRCOLES TENDER
FRANCISCO MONTERO SIMARRO



Descarga de Material Adicional

Este E-book tiene disponible un material adicional que complementa el contenido del mismo.

Este material se encuentra disponible en nuestra página Web www.ra-ma.com.

Para descargarlo debe dirigirse a la ficha del libro de papel que se corresponde con el libro electrónico que Ud. ha adquirido. Para localizar la ficha del libro de papel puede utilizar el buscador de la Web.

Una vez en la ficha del libro encontrará un enlace con un texto similar a este:

“Descarga del material adicional del libro”

Pulsando sobre este enlace, el fichero comenzará a descargarse.

Una vez concluida la descarga dispondrá de un archivo comprimido. Debe utilizar un software descompresor adecuado para completar la operación. En el proceso de descompresión se le solicitará una contraseña, dicha contraseña coincide con los 13 dígitos del ISBN del libro de papel (incluidos los guiones).

Encontrará este dato en la misma ficha del libro donde descargó el material adicional.

Si tiene cualquier pregunta no dude en ponerse en contacto con nosotros en la siguiente dirección de correo: ebooks@ra-ma.com

*Dedicado a mi tío Juan
(1937-2012)*

*A mi mujer Nieves y a mis hijos Hugo y Marco,
por darme la paz que necesito.*

*A mi familia, por quererme.
A Yolanda, por supuesto.*

Índice

INTRODUCCIÓN	9
CAPÍTULO 1. PLANIFICACIÓN DE INTERFACES GRÁFICAS.....	11
1.1 ELEMENTOS DE DISEÑO: PERCEPCIÓN VISUAL	12
1.2 FUNDAMENTOS DE LA COMPOSICIÓN.....	13
1.2.1 El equilibrio visual	14
1.2.2 La tensión compositiva	14
1.3 COLOR, TIPOGRAFÍA, ICONOS	15
1.3.1 Color.....	15
1.3.2 Tipografía.....	17
1.3.3 Iconos	18
1.4 COMPONENTES DE UNA INTERFAZ WEB.....	19
1.4.1 Cabecera	20
1.4.2 Los sistemas de navegación	21
1.4.3 El cuerpo de la página	24
1.4.4 El pie de página	24
1.4.5 Los espacios en blanco	25
1.5 LENGUAJES DE MARCAS	27
1.6 MAQUETACIÓN WEB. ELEMENTOS DE ORDENACIÓN.....	29
1.6.1 Distribución de elementos en la interfaz: capas, marcos	29
1.7 MAPA DE NAVEGACIÓN. PROTOTIPOS.....	31
1.8 INTERPRETACIÓN DE GUÍAS DE ESTILO. ELEMENTOS.....	33
1.9 APLICACIONES PARA DESARROLLO WEB	35
1.10 GENERACIÓN DE DOCUMENTOS Y SITIOS WEB	36
1.11 PLANTILLA DE DISEÑO	37
1.12 INTERACCIÓN PERSONA-ORDENADOR	38
1.13 CONCLUSIÓN	39
RESUMEN DEL CAPÍTULO	39
EJERCICIOS PROPUESTOS.....	40
TEST DE CONOCIMIENTOS	41
CAPÍTULO 2. USO DE ESTILOS.....	43
2.1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET)	44
2.2 SELECTORES: ESTILOS EN LÍNEA BASADOS EN ETIQUETAS, EN CLASES Y EN IDENTIFICADORES.....	45
2.2.1 Selectores basados en etiquetas.....	45
2.2.2 Selectores basados en clases	47
2.2.3 Selectores basados en identificadores	49

2.3	AGRUPACIÓN Y ANIDAMIENTO DE SELECTORES	52
2.3.1	Agrupamientos	52
2.3.2	Anidamientos.....	53
2.4	BUENAS PRÁCTICAS AL ESCRIBIR CSS	57
2.5	ATRIBUTOS. MODELO DE CAJAS.....	60
2.5.1	Unidades de medida	61
2.5.2	Atributos de posición	62
2.5.3	Atributos <i>margin</i>	63
2.5.4	Atributos <i>padding</i>	64
2.5.5	Atributos <i>border-top</i> , <i>border-bottom</i> , <i>border-right</i> , <i>border-left</i>	65
2.5.6	Atributos del contenido	67
2.6	ELEMENTOS: COLORES DE FONDO, TEXTOS, ENLACES, LISTAS, TABLAS, VISIBILIDAD, IMÁGENES	72
2.7	SUPERPOSICIÓN Y PRECEDENCIA DE ESTILOS	79
2.7.1	Superposición de cajas.....	79
2.7.2	Precedencia de estilos.....	83
2.8	CREAR Y VINCULAR HOJAS DE ESTILO.....	85
2.9	CREAR Y VINCULAR HOJAS DE ESTILO EN CASCADA EXTERNA	86
2.10	HERRAMIENTAS Y TEST DE VERIFICACIÓN	88
2.11	CONCLUSIÓN Y PROPUESTAS PARA AMPLIAR	90
	RESUMEN DEL CAPÍTULO.....	91
	EJERCICIOS PROPUESTOS.....	92
	TEST DE CONOCIMIENTOS	92
	CAPÍTULO 3. IMPLANTACIÓN DE CONTENIDO MULTIMEDIA	95
3.1	DERECHOS DE LA PROPIEDAD INTELECTUAL. LICENCIAS. LEY DE LA PROPIEDAD INTELECTUAL. DERECHOS DE AUTOR.....	96
3.1.1	Derechos de la propiedad intelectual.....	96
3.1.2	Licencias	97
3.1.3	Propiedad intelectual	99
3.1.4	Derechos de autor	99
3.2	IMÁGENES: MAPA DE BITS, IMAGEN VECTORIAL. SOFTWARE PARA CREAR Y PROCESAR IMÁGENES. FORMATOS DE ARCHIVOS DE IMÁGENES	100
3.2.1	Tipos de imágenes en la web	100
3.2.2	Software para crear y procesar imágenes.....	102
3.3	OPTIMIZACIÓN DE IMÁGENES PARA LA WEB. RESOLUCIÓN	103
3.4	AUDIO: INSERTAR AUDIO EN UNA WEB	104
3.4.1	Formatos	104
3.4.2	Conversión de formatos.....	106
3.4.3	Insertando audio en una web.....	107
3.4.4	Consideraciones para el uso de audio en un sitio web.....	113
3.5	VÍDEO: INSERTAR VÍDEO EN UNA WEB	114
3.5.1	Formatos	114
3.5.2	Conversión de formatos.....	114
3.5.3	Insertar vídeo en una web	116

3.6	ANIMACIONES.....	120
3.6.1	Formatos de animaciones.....	120
3.6.2	Herramientas de programación.....	122
3.7	CONCLUSIÓN Y PROPUESTAS PARA AMPLIAR.....	129
	RESUMEN DEL CAPÍTULO.....	129
	EJERCICIOS PROPUESTOS.....	130
	TEST DE CONOCIMIENTOS	130
	CAPÍTULO 4. INTEGRACIÓN DE CONTENIDO INTERACTIVO	133
4.1	ELEMENTOS INTERACTIVOS BÁSICOS Y AVANZADOS	134
4.1.1	Introducción a jQuery.....	135
4.1.2	Manejo de eventos	136
4.2	CAMBIO DE LAS PROPIEDADES DE UN ELEMENTO.....	140
4.2.1	Cambio de las propiedades CSS.....	140
4.2.2	Añadir nuevos elementos	140
4.2.3	Añadiendo texto.....	142
4.2.4	Otros métodos	142
4.3	EJECUCIÓN DE SECUENCIAS DE FUNCIONES.....	142
4.4	COMPORTAMIENTO DE LOS ELEMENTOS. EFECTOS VISUALES	144
4.4.1	Efectos básicos	145
4.4.2	Efectos <i>Fading</i> (opacidad-transparencia de los elementos)	147
4.4.3	Efectos <i>Sliding</i> (desplazamiento)	149
4.4.4	Otros efectos	150
4.5	COMPORTAMIENTOS INTERACTIVOS	155
4.5.1	Eventos de ratón.....	155
4.5.2	Eventos de teclado	160
4.5.3	Eventos de ventana	162
4.6	REPRODUCCIÓN DE SONIDO, VÍDEO Y ANIMACIÓN	164
4.7	CONCLUSIONES.....	166
	RESUMEN DEL CAPÍTULO.....	167
	EJERCICIOS PROPUESTOS.....	167
	TEST DE CONOCIMIENTOS	168
	CAPÍTULO 5. DESARROLLO DE WEBS ACCESIBLES.....	169
5.1	CONCEPTO DE ACCESIBILIDAD	170
5.2	EL CONSORCIO WORLD WIDE WEB (W3C).....	171
5.3	PRINCIPIOS GENERALES DE DISEÑO ACCESIBLE	173
5.4	PAUTAS DE ACCESIBILIDAD AL CONTENIDO EN LA WEB	175
5.5	CRITERIOS PARA SATISFACER LOS REQUISITOS DEFINIDOS EN LAS WCAG	177
5.6	PRIORIDADES. PUNTOS DE VERIFICACIÓN. NIVELES DE ADECUACIÓN	181
5.6.1	Perceptibilidad.....	182
5.6.2	Operatividad.....	185
5.6.3	Comprensibilidad	188
5.6.4	Robustez	190
5.7	MÉTODOS PARA REALIZAR REVISIÓNES PRELIMINARES Y EVALUACIONES DE ADECUACIÓN O CONFORMIDAD DE DOCUMENTOS WEB	191

5.8 HERRAMIENTAS DE ANÁLISIS DE ACCESIBILIDAD WEB	194
5.8.1 Software y herramientas <i>on line</i>	194
5.8.2 Chequeo de la accesibilidad web desde diferentes navegadores	196
5.8.3 Chequeo de la accesibilidad web desde dispositivos móviles.....	197
5.9 CONCLUSIONES.....	197
RESUMEN DEL CAPÍTULO.....	198
EJERCICIOS PROPUESTOS.....	198
TEST DE CONOCIMIENTOS	199
CAPÍTULO 6. IMPLEMENTACIÓN DE LA USABILIDAD EN LA WEB. DISEÑO AMIGABLE	201
6.1 CONCEPTO DE USABILIDAD	202
6.2 ANÁLISIS DE LA USABILIDAD. TÉCNICAS	204
6.3 PRINCIPIOS PARA CONSEGUIR WEBS AMIGABLES.....	205
6.4 IDENTIFICACIÓN DEL OBJETIVO DE LA WEB	208
6.5 TIPOS DE USUARIO. NECESIDADES	210
6.6 BARRERAS IDENTIFICADAS POR LOS USUARIOS E INFORMACIÓN FÁCILMENTE ACCESSIONEABLE.....	211
6.7 VELOCIDAD DE CONEXIÓN	213
6.8 IMPORTANCIA DEL USO DE ESTÁNDARES EXTERNOS	215
6.9 NAVEGACIÓN FÁCILMENTE RECORDADA FRENTE A NAVEGACIÓN REDESCUBIERTA	216
6.10 FACILIDAD DE NAVEGACIÓN EN LA WEB	219
6.11 VERIFICACIÓN DE LA USABILIDAD EN DIFERENTES NAVEGADORES Y TECNOLOGÍAS	220
6.12 HERRAMIENTAS Y TEST DE VERIFICACIÓN	221
6.13 CONCLUSIONES.....	222
RESUMEN DEL CAPÍTULO.....	223
EJERCICIOS PROPUESTOS.....	223
TEST DE CONOCIMIENTOS	224
ÍNDICE ALFABÉTICO	227

Introducción

Este libro surge con el propósito de acercar al lector a los aspectos más importantes que encierra el diseño de interfaces web ante la demanda de personal cualificado para su desarrollo. Con tal propósito, puede servir de apoyo también para estudiantes del Ciclo Formativo de Grado Superior de **Desarrollo de Aplicaciones Web** y para estudiantes universitarios del **Grado de Informática**.

Actualmente, el diseño de interfaces web es una disciplina muy demandada dentro del mundo del desarrollo de aplicaciones en Internet. El éxito o fracaso de una aplicación web, en cualquier dispositivo móvil o fijo, no depende tanto de la funcionalidad o información que ofrece como de lo fácil y sencillo que es acceder a ella. Sin embargo, el diseño de interfaces no debe entenderse solo como una disciplina asociada al arte y al diseño gráfico propia de expertos en esas disciplinas. El diseño de interfaces tiene mucho de estructura, organización, reglas y normativas que solo un técnico en Informática con conocimientos de programación y lenguajes puede aplicar y optimizar. Por lo tanto, los contenidos de este libro, lejos de ser un tratado de diseño y buen gusto, se centran en ofrecer técnicas y herramientas para conseguir interfaces web usables y accesibles, que sigan los estándares de desarrollo, reales y de facto, impuestos en la red y que estén estructuradas y organizadas acorde a la funcionalidad que sirve el sistema que hay detrás de la interfaz. Evidentemente, si a esto se le unen competencias propias del diseño gráfico, el éxito de cualquier desarrollo estará asegurado.

El libro pretende servir de referencia a docentes y estudiantes de Ciclos Formativos o Universidad. Los docentes pueden utilizar los contenidos aquí expuestos para inculcar aspectos genéricos del Diseño de Interfaces Web o, simplemente, centrarse en preparar a fondo alguno de ellos. Además, las actividades resueltas integradas en los contenidos, los ejercicios propuestos y los test de evaluación sirven para desarrollar y afianzar conocimientos en cualquier secuencia didáctica llevada a cabo en el aula. También las sugerencias y enlaces web para ampliar conocimientos incluidos en cada capítulo pueden servir de gran ayuda para desarrollar actividades de ampliación de conocimientos o concretar más específicamente un tema de interés. Por su parte, los estudiantes pueden utilizar la clara y concreta exposición de los contenidos, las actividades resueltas y los enlaces para ampliar conocimientos como guía para afianzar su aprendizaje.

Ra-Ma pone a disposición de los profesores una guía didáctica para el desarrollo del tema. En ella se incluyen las soluciones a los ejercicios expuestos en el texto. Puede solicitarlo a editorial@ra-ma.com, acreditándose como docente y siempre que el libro sea utilizado como texto base para impartir las clases.

1

Planificación de interfaces gráficas

OBJETIVOS DEL CAPÍTULO

- ✓ Identificar y reconocer las principales componentes de una página web.
- ✓ Clasificar sitios web atendiendo a sus objetivos.
- ✓ Identificar, clasificar y combinar aplicaciones de desarrollo web.

En los últimos años la cultura Web ha calado en la sociedad actual. Cualquier persona, sobre todo estudiantes de Diseño de aplicaciones Web, sabe o ha oído hablar de lo que es un *sitio web*. Sin ser muy formal, un sitio web es *un conjunto de páginas web agrupadas bajo un dominio y que comparten una dirección en la Web*. Actualmente, muchos son los sitios web que se pueden encontrar en Internet. De hecho, hoy en día cualquier institución pública o privada posee un sitio en la Web. Un ejemplo de sitio web es el de esta editorial¹ o el de la Universidad de Castilla-La Mancha, que usamos en varias figuras de este capítulo².

Una característica muy común en la mayoría de los sitios web es que tienen una página principal, *home* o *homepage* desde la que se puede acceder a todos los contenidos ofrecidos por el sitio. Desde el punto de vista del diseño, que es el objetivo de este libro, una máxima es que todas las páginas que componen el sitio web cumplan *criterios de homogeneidad y consistencia*. Esto se justifica porque un usuario visitante de un sitio web podrá acceder al sitio web a través del *home* o a través de cualquiera de las páginas web que lo compongan. De alguna manera el diseño web es una *marca comercial o personal*, y esté donde esté el usuario dentro del sitio, las páginas web deben siempre identificarse con esa *marca comercial o personal*.

Es importante destacar que cuando una persona, empresa o institución crea un sitio web es porque está interesado en comunicar algo a los demás, ya sea con fines comerciales o simplemente informativos. Por lo tanto, el objetivo es comunicar, pero con un objetivo específico implícito que se puede resumir en *venderse bien a los demás*. Otra máxima del diseño web es que el sitio web debe ser *atractivo y funcional*. Evidentemente, el diseñador web no puede controlar la información que él o su cliente quiere poner en el sitio, pero sí decide cómo está organizado el sitio y cómo es esa información mostrada.

En este capítulo identificaremos y describiremos componentes característicos que deben planificarse y diseñarse antes de construir un sitio web. Sin duda, este capítulo es básico para entender los conceptos y el vocabulario técnico mostrados en el libro. En las secciones correspondientes se hará referencia a los capítulos que tratarán los contenidos más destacables.

1.1 ELEMENTOS DE DISEÑO: PERCEPCIÓN VISUAL

A la hora de desarrollar un sitio web hay que tener en cuenta qué interfaz web ofrecer y qué elementos incluir en ella. Cada uno de esos elementos contribuirá a la percepción que sus potenciales usuarios tendrán de la aplicación.

Una de las primeras impresiones que causará la interfaz de usuario será visual. Aunque suene duro, en función de la forma, tamaño, ubicación, color, tipografía, etc., que se le asigne a cada uno de los elementos de la interfaz se influirá, de una manera u otra, en el usuario o visitante de un sitio web. El diseñador ha de tener en cuenta constantemente a lo largo de todo su trabajo estas circunstancias y saber valorar la relación directa que puede identificarse entre sus diseños y cómo estos serán percibidos. A modo de ejemplo, ¿algún aficionado al fútbol se puede imaginar una página del Real Madrid diseñada con los colores azul y grana? ¿Cómo lo percibirían los aficionados del Real Madrid? O, ¿un

¹ <http://www.ra-ma.es/>

² <http://www.uclm.es>

lector empedernido interesando en comprar libros en Amazon imagina no encontrar fácilmente la opción de *comprar* un libro?

El diseñador debe buscar un equilibrio entre los elementos que constituyen la interfaz, a fin de poder así hallar un adecuado sentido gráfico de su diseño, lo que a su vez le permitirá conseguir una comunicación eficaz. Debemos tener muy presente que, por principio, nada debe ser totalmente superfluo en un diseño, aunque tampoco es conveniente excederse en la utilización de elementos por el mero hecho de ponerlos, ya que esto puede producir un *excesivo ruido* o distracciones que pueden enmascarar el mensaje de la comunicación. Por ejemplo, un sitio web con demasiados elementos animados distrae al usuario olvidándose por un momento de que hay un texto que leer.

Una vez asimilada toda la información sobre aquello que quiere comunicar, el diseñador ha de empezar a generar soluciones de diseño adecuadas al propósito. Lo primero que determinará es el área de diseño, es decir, qué tamaño se asignará al espacio del que se dispone para la composición gráfica. Una composición gráfica puede estar formada por muchos o pocos elementos. Puede componerse exclusivamente de la presencia de texto o solo de imágenes; puede poseer grandes espacios en blanco o constituir una combinación equilibrada de elementos gráficos. Pero, en cualquier caso, debe ser adecuada con lo que se quiere comunicar. Por ejemplo, ¿es lo mismo diseñar el programa de un concierto de música clásica que el cartel de un concierto de *rock*? Y, ¿es lo mismo un portal de venta de ropa que un portal tipo buscador Google?

A continuación se profundizará en cuáles son los fundamentos de la composición.

1.2 FUNDAMENTOS DE LA COMPOSICIÓN

Con lo visto anteriormente, se puede concluir que hacer una composición gráfica es ordenar todos los elementos de nuestro diseño, ya sean texto o ilustraciones, destinados a lograr los objetivos propuestos, es decir, impactar visualmente al público receptor de nuestro mensaje. Aunque no hay ninguna norma específica que garantice el éxito de una composición, sí existen una serie de pautas a las que el diseñador se puede adecuar para obtener soluciones eficaces, todas ellas muy relacionadas con la *percepción*. El diseñador ha de tener un profundo conocimiento de los factores que rigen el fenómeno de la percepción para poder establecer sus composiciones de un modo sólido y fundamentado. Algunos de estos factores son:

- **Componentes psicosomáticos del sistema nervioso:** nos facilitan el contacto visual con nuestro mensaje gráfico haciendo uso del mecanismo de percepción llamado vista. Con ella recogemos información visual (percibimos distintas formas, ubicaciones, longitudes de onda de un color, etc.) que luego nuestro cerebro interpreta como contornos, texturas, dimensiones, etc., dotándolas de un significado gráfico definido.
- **Componentes de tipo cultural:** influyen en la interpretación que hacemos de los estímulos desde un punto de vista cultural y educacional. Por ejemplo, el color que en Occidente está relacionado con el luto es el negro mientras que en los países orientales este mismo significado se le asigna al color blanco.
- **Experiencias compartidas con el entorno:** por ejemplo, conceptos altamente arraigados como: hierba/verde, azul/cielo, hielo/frío. Todas ellas van constituyendo una serie de dualidades que el hombre va aprendiendo desde su infancia y que, posteriormente, serán utilizadas por él como patrones con los que interpretar y dotar de significado la realidad.

Todos los factores anteriormente señalados proporcionan una clara orientación sobre cómo una determinada composición puede llegar a afectar a nuestra percepción y, consecuentemente, a la interpretación final que hagamos del mensaje. Sin embargo, hay más factores relacionados con la disposición de los elementos para conseguir una composición adecuada.

1.2.1 EL EQUILIBRIO VISUAL

Antes de hablar del *equilibrio visual* es necesario definir dos conceptos previos de mucha importancia: el **equilibrio formal** y el **informal**. El equilibrio formal se basa en la *bisimetría*. Se busca con él un centro óptico dentro del diseño y no tiene por qué coincidir con el centro geométrico de la composición. El punto de equilibrio formal suele estar ubicado un poco por encima del centro geométrico. Una composición que decida seguir este esquema compositivo reflejará estabilidad, calma y estatismo. No supone una composición muy audaz o creatividad, aunque lo que sí asegura es una distribución armónica de los elementos. El equilibrio informal, por el contrario, está altamente cargado de fuerza gráfica y dinamismo. Prescinde por completo de la simetría y el equilibrio se consigue aquí en base a contraponer y contrastar los *pesos visuales* de los elementos, buscando diferentes densidades, tanto formales como de color, que consigan armonizar visualmente dentro de una asimetría intencionada.

Las formas pequeñas poseen menor peso visual que las más grandes. Si, además, la forma de la figura no es regular, su peso aumenta notablemente. Los colores también juegan un papel importante en lo que respeta al peso visual: cuanto más luminosos sean, mayor peso compositivo tendrán. Entre elementos con el mismo tamaño pero colores de diferente intensidad tiene más peso visual el de color intenso. Pero, entre elementos del mismo color, tiene más peso visual el de más tamaño.

El último elemento importante de equilibrio es la posición. Dependiendo de dónde se coloquen los elementos se podrá conseguir un mayor equilibrio y se apreciarán mejor por parte del usuario. Es evidente (en Occidente), por ejemplo, que los elementos situados en la parte superior de la página tienen más protagonismo que los situados en la parte inferior derecha.

En resumen, para conseguir un equilibrio adecuado hay que estar al tanto de todos los factores compositivos que intervienen, tales como el peso, el tamaño y la posición.

1.2.2 LA TENSIÓN COMPOSITIVA

Lo opuesto al equilibrio desde el punto de vista estructural es la *tensión compositiva*. La tensión tiene como finalidad dirigir la mirada y conseguir fijar la atención del observador. La tensión se puede conseguir con la combinación de líneas y formas agudas e irregulares. Hay algunas técnicas para provocar la tensión y conseguir captar la atención del usuario. Las principales técnicas son:

- **Técnica sugestiva:** consiste en dirigir intencionadamente la atención a un punto determinado utilizando elementos de apoyo. Por ejemplo, imágenes de personas que miran hacia un punto determinado (que sería el punto de interés).
- **Técnica rítmica:** basada en la tendencia innata del ojo humano a completar secuencias de elementos (ya sean números, formas, figuras geométricas o colores), agrupando aquellos que poseen formas semejantes.

ACTIVIDADES 1.1



► Observe y navegue por los siguientes sitios web. ¿Qué siente al verlos? Describa, con sus propias palabras, lo que resaltaría como negativo en cada uno de ellos.

<http://art.yale.edu/>. Escuela de arte de Yale en Connecticut (EE.UU.).

http://www.vatican.va/holy_father/special_features/hf_jp_ii_xxv_en.htm. Página publicada en el Vaticano.

1.3 COLOR, TIPOGRAFÍA, ICONOS

Dentro de las composiciones para diseñar sitios web, los elementos más destacados que podemos encontrar en todas ellas son: colores, tipografía e iconos. Los tipos y cantidad de estos elementos, así como su variación, dependerá lo que pretendemos comunicar con el sitio web (y de la creatividad del diseñador³). A continuación se describen estos tres elementos.

1.3.1 COLOR

En los entornos gráficos digitales, los colores se forman a partir de tres básicos, el **rojo, verde y azul**, que se denominan *componentes*⁴. Generalmente, la intensidad de cada componente se expresa (en diseño web) como un número hexadecimal del 00 al FF (del 0 al 255 en base diez). Por ejemplo, el color rojo se representa como #FF0000, porque tiene toda la intensidad de rojo y nada de verde y azul.

Los colores básicos son:

- ✓ #FF0000 - Rojo
- ✓ #00FF00 - Verde
- ✓ #0000FF - Azul
- ✓ Otros colores son:
 - ✓ #FFFFFF - Blanco
 - ✓ #000000 - Negro
 - ✓ #FFFF00 – Amarillo (mezcla de rojo y verde)

Para hacer un color más oscuro se reduce la intensidad del componente, dejando los otros dos iguales. De esta forma, el rojo (#FF0000) se hace más oscuro así: #CC0000, #990000, #660000, #330000, etc.

Actualmente, la gran mayoría de entornos que permiten el trabajo con colores ofrecen la equivalencia de los colores en este formato hexadecimal. La Figura 1.1 muestra una típica paleta de colores en Windows 7, mostrando el color negro (00 rojo, 00 verde y 00 azul).

³ En el Capítulo 2 se trata el tema de CSS (*Hojas de estilo*), un lenguaje clave para determinar la apariencia de un sitio web.

⁴ En inglés, este modelo se llama RGB (*Red, Green and Blue*), de rojo, verde y azul.

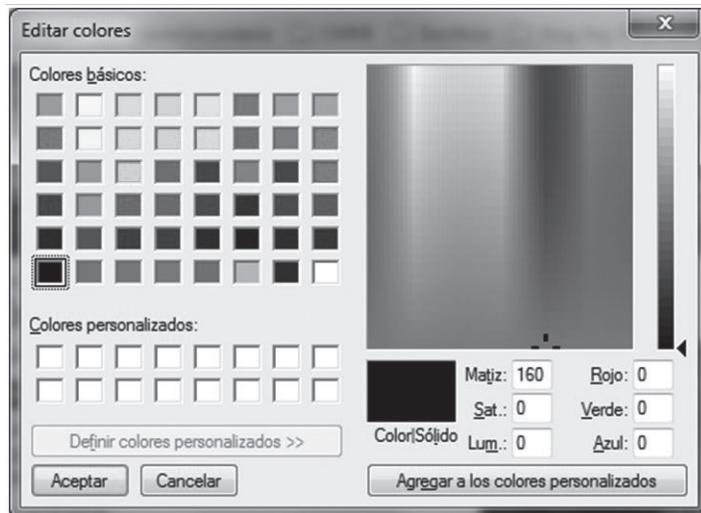


Figura 1.1. Tabla de colores de Windows 7

Aunque parezca lo más sencillo, elegir una combinación de colores apropiada para un diseño es una de las tareas más difíciles. Para algunos expertos en diseño, la combinación adecuada de colores requiere un *gen artístico* que no todo el mundo tiene. Para otros, la combinación adecuada de colores se puede calcular con ecuaciones matemáticas que combinan colores, tonos y saturación para crear composiciones *artísticas*.

Sin embargo, para que un diseñador web no tenga que ir de un extremo a otro para poder crear una buena composición, existe software que ayuda en esta labor de crear combinaciones armónicas y placenteras. Estas herramientas suelen estar muy orientadas a facilitar el trabajo del diseñador *rescatando* combinaciones de sitios o imágenes ya creadas:

- Crear varias combinaciones de colores a partir de un color de referencia.
- Obtener combinaciones de colores presentes en una imagen ya creada. Útil cuando se quiere sacar una combinación, por ejemplo, de una foto.
- Obtener el código de un color y el valor hexadecimal (RGB y CMYK) de cualquier color que se ve en tu pantalla. Esto es interesante cuando se desean sacar los colores de, por ejemplo, un sitio web ya creado.
- Buscar imágenes que satisfagan un patrón de colores concreto. Útil cuando se quiere encontrar imágenes que combinen con los colores de la web.

Algunas de las herramientas actuales (en inglés⁵) con esta y otras funcionalidades son:

- **ColorPix:** es un pequeño software que descargas en tu ordenador y te permite conocer los códigos, las coordenadas y el número de píxeles de cada color presente en tu pantalla. ColorPix traduce automáticamente

⁵ Pueden usarse las utilidades de Google Traslator para traducir las páginas.

cada color en los códigos RGB, HEX, HSB y CMYK. Permite, incluso, que hagas *zoom* a un área de la pantalla. Funciona con Windows y es gratuito⁶.

- **Color Schemer Online:** es una aplicación web gratuita útil para crear las mejores combinaciones de colores posibles. Basta con seleccionar los valores RGB o HEX del color con el cual quieras comenzar. Entonces, aparecerá un esquema de colores más armonioso (obtenidos según ecuaciones matemáticas) con los códigos HEX y RGB relativos⁷.
- **Whats Its Color:** es un servicio web gratuito útil para encontrar los colores complementarios para una imagen. Basta con subir una foto al sitio (*upload*) o encontrar una en la web. Para esa imagen aparecerá una combinación compuesta de los colores primarios de la imagen. Sobre esos colores la aplicación propondrá los diez mejores colores únicos, entre los complementarios y los dominantes de la imagen⁸.

1.3.2 TIPOGRAFÍA

Sin duda, los textos son la base de la gran mayoría de los sitios web. Transmitir información mediante *letras* es lo más común y, por tanto, requiere una especial atención.

Cuando se habla en diseño web de *fuente* se hace referencia a un conjunto de caracteres con un estilo o modelo gráfico particular. De alguna manera, una fuente es sinónimo de *tipo de letra*. A la hora de manejar fuentes en un sitio web hay que tener en cuenta una serie de limitaciones y características que complican el diseño. Entre esas limitaciones, la más destacada es que las fuentes disponibles en cada sistema operativo son diferentes. Aunque las versiones actuales de los navegadores instalan un conjunto de fuentes similar en Windows, Linux, MacOs, etc., hay que tener en cuenta que existen otros navegadores y otros sistemas operativos, por lo que es importante asegurarnos de que los contenidos textuales tendrán el mismo aspecto con independencia del navegador que interprete el sitio web. Más allá de esto, las otras limitaciones están relacionadas con la adecuación, con lo que se quiere comunicar su legibilidad y, como ocurre con los colores, si son o no combinadas con *buen gusto*.

Las fuentes más comunes suelen ser las llamadas *Sans Serif*, destacando entre ellas *Verdana*, *Arial* y *Helvetica*, aunque hay una fuente concreta con ese nombre, *Sans Serif*, que hace referencia a un tipo genérico. Estas fuentes son adecuadas para mostrar texto en pantallas. Si se desea que los textos se puedan imprimir, es conveniente sustituir las fuentes anteriores por alguna tipo *Serif* (con remates en sus extremos), ya que son más legibles en documentos impresos y menos monótonas. Entre estas fuentes tipo *Serif* destacan las conocidas *Times New Roman*, *Courier* y *Courier New*, aunque también hay una fuente concreta llamada *Serif* que hace referencia a un tipo genérico. Como se verá más adelante, usando CSS es posible indicar que para un mismo texto se usen fuentes diferentes, una para ver en pantalla y otra para que se muestre impresa.

⁶ http://www.colorschemer.com/colorpix_info.php

⁷ <http://www.colorschemer.com/online.html>

⁸ <http://whatsitscolor.com/>

Para concluir, un sitio web no es aconsejable que use más de tres fuentes. Es una recomendación bastante extendida en los sitios web actuales. Algunos ejemplos de fuentes se muestran en la siguiente figura:



Figura 1.2. Ejemplo de fuentes

1.3.3 ICONOS

La palabra *ícono* se utiliza para designar a las imágenes gráficas generalmente pequeñas y que suelen ser metáforas de las acciones que se pueden hacer. Por lo general, se trata de mantener una relación entre el ícono y lo que representa, es decir, que lo que se identifica con dicho ícono está ligado de alguna manera al ícono que lo está representando. Respecto a esto, existen algunos *estándares de facto*, como por ejemplo, el ícono de un disquete sustituye a la orden “guardar”; el de una lupa, a la orden “buscar” y el de una carpeta representa a los archivos.

Con estos dibujos evitamos leer textos y obtenemos de una manera más rápida las opciones que nos presentan. Una buena elección de estos íconos es muy importante, puesto que si un usuario no es capaz de determinar su significado no hemos conseguido nuestro propósito de ahorrarle tiempo en la visualización de la página. Un ícono debe contener la menor cantidad de detalle posible, únicamente dejar los imprescindibles para la comprensión de su significado.

Otro punto importante en la elección de un ícono es la estandarización, o mejor dicho, a lo que están acostumbrados los usuarios. Es muy arriesgado innovar con estos temas puesto que los usuarios son muy reticentes a los cambios y tendría que ser muy bueno el ícono para que no despiste al usuario.

La siguiente figura muestra parte del *home* del sitio web Amazon⁹, un ejemplo de portal que no usa muchos íconos pero sí usa un ícono asociado a la cesta de la compra para conseguir más impacto visual, ayudando al usuario a localizar bien cómo acceder a comprar.

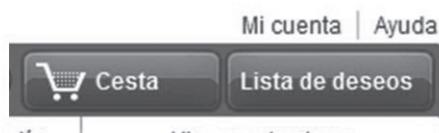


Figura 1.3. Ícono combinado con texto

⁹ <http://amazon.es>

Aunque pueda parecer lo contrario, los iconos tienen sus limitaciones en la web. Estas limitaciones están relacionadas con la falsa creencia de que un ícono es interpretado más rápido por un usuario que un texto. A veces, eso no es así y los iconos, lejos de facilitar, complican más la web, al no estar el usuario familiarizado con lo que quiere representar el ícono. Esto es debido a que los íconos son siempre subjetivos, están sujetos a la interpretación individual y subjetiva de cada persona a partir de su experiencia. Nunca son totalmente claros e inequívocos y existe riesgo de malentenderlos. Por esta razón no se recomienda usar íconos para operaciones críticas y se recomienda mejor un texto con una fuente adecuada y legible, o una combinación de ambos, como la imagen anterior de Amazon.

Según varios estudios, debido a los riesgos de interpretación de los íconos, su adecuado diseño no puede depender únicamente de la inspiración o preferencias de diseñadores o los responsables del sitio. Es necesaria la creación de varios diseños o prototipos para cada ícono y la realización de test con usuarios reales en un proceso iterativo de diseño-test-rediseño.

ACTIVIDADES 1.2



- Desde el punto de vista de los colores, fuentes e íconos, compare estos dos sitios web. Escriba al menos tres aspectos positivos y tres negativos de cada uno.

<http://www.lingscars.com/>. Agencia de alquiler de coches.

<http://www.avis.es>. Agencia de alquiler de coches.

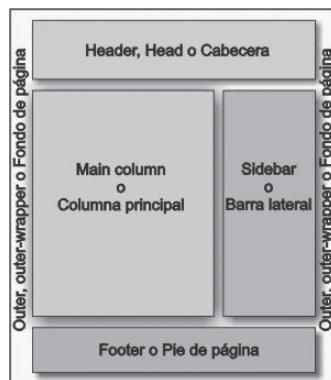
1.4 COMPONENTES DE UNA INTERFAZ WEB

Un punto muy importante a la hora de introducir lo que es el diseño de interfaces web es definir cuáles son actualmente sus componentes, es decir, qué partes forman un sitio web.

Desde las primeras páginas web hasta la actualidad, los diseños han evolucionado hacia la homogeneidad, ofreciendo unas interfaces bien definidas, con un conjunto de componentes gráficos y funcionales similares que hacen posible que, sea cual sea el usuario que accede a un sitio web cualquiera, la comunicación entre ellos sea posible y efectiva. En esta evolución, como si de una selección natural se tratase, se han asentado elementos que han demostrado su utilidad y su comprensión por los usuarios. Algunos ejemplos son: sistemas de navegación, los pies de página o los formularios de entrada de datos, etc., que normalmente encontraremos en todas las páginas web y cuyo diseño y funcionalidad son similares en todas ellas.

La Figura 1.4 muestra la estructura general de un sitio web (o página web) marcando los nombres de sus elementos principales. Conocer estas partes es importante por sí mismo, pero también es fundamental porque forma parte del vocabulario al que se hace referencia en las guías de estilo, introducidas también en este capítulo¹⁰. A continuación, se describen esos elementos.

¹⁰ En el Capítulo 2 se utiliza CSS (Hojas de estilo) para diseñar la estructura de una web.

**Figura 1.4.** Partes de una página

1.4.1 CABECERA

Se entiende por *cabecera* una zona de la interfaz web situada en la parte superior de la misma (de ahí su nombre), de anchura generalmente igual a la de la página y altura variable, en la que se ubica generalmente el logotipo del sitio web o de la empresa propietaria, acompañado generalmente de un texto identificador de la misma y de otros elementos de diseño, como fotografías (simples o formando un montaje), formularios de *login* (entrada de claves de acceso al sistema), *banners* publicitarios, etc.¹¹

Un ejemplo de cabecera (con fondo de página en blanco) para el sitio web de la Universidad de Castilla-La Mancha es mostrado en la siguiente figura. Observar que aparece un menú en la parte derecha.

**Figura 1.5.** Ejemplo de cabecera

El objetivo principal de la cabecera está muy relacionado con el de las cabeceras en las portadas de la prensa escrita, por ejemplo, diarios:

- Identificar el sitio web con la empresa a la que representa mediante el logotipo y el nombre del mismo, de la empresa propietaria o de la marca que representa.
- Identificar y homogeneizar todas las páginas pertenecientes al sitio web, ya que la cabecera suele ser común en todas ellas, creando con ello un elemento de referencia común.
- Crear una separación visual entre el borde superior de la interfaz y el contenido central de la misma, haciendo más cómoda su visualización y lectura.

¹¹ Esta definición válida para cualquier sitio web, pero coincide exactamente con el uso que se le da en gestores de contenidos como Joomla, introducido en el punto 1.5.

El motivo por el que la cabecera se encuentra situada en la zona superior de la interfaz y el logotipo en su parte izquierda obedece a consideraciones de jerarquía visual. En la cultura occidental estamos acostumbrados a leer de arriba hacia abajo y de izquierda a derecha, por lo que la parte superior izquierda de una página es la primera a la que dirige el usuario la vista, con lo que situando en ella el logotipo nos aseguramos que sea el primer elemento gráfico que el espectador observe.

La cabecera no es obligada en un sitio web, pero es habitual usarla. La forma más común de la cabecera es rectangular, pero conforme avanzan el diseño gráfico, se pueden encontrar de muchas formas y colores, asociándose generalmente al impacto que se quiere causar en el usuario. En cualquier caso, el diseño (colores, tipografía, etc.) de la cabecera *nunca debe ocultar el logo y el texto* que se muestra en ella.

La cabecera no tiene siempre que ocupar todo el ancho de la página, puede ocurrir que tan solo ocupe una parte del mismo, generalmente la izquierda, en la que se suele situar en una banda vertical común con un menú de navegación. Por otro lado, también es posible encontrar páginas sin cabecera, generalmente en páginas de inicio que sirven como presentación del sitio y que presentan un diseño especial, diferente al del resto de páginas que lo forman. También ocurre en páginas de diseño vanguardista, que intentan huir de los patrones clásicos, muchas de ellas desarrolladas en tecnología *Flash*. En estos casos el logotipo puede estar situado en cualquier zona de la interfaz, generalmente en la parte inferior izquierda de la misma.

1.4.2 LOS SISTEMAS DE NAVEGACIÓN

Los sistemas de navegación son los elementos de una interfaz que permiten la navegación por las diferentes secciones y páginas que componen el sitio web.

Generalmente se presentan como menús formados por diferentes opciones, con las que el usuario puede interaccionar al seleccionarlas, pasando a una nueva página o documento.

Un ejemplo de menú es mostrado en la siguiente imagen:



Figura 1.6. Ejemplo de menú (en dos partes)

Los menús pueden tener textos, gráficos o ambos (el de la imagen es solo texto), todo ello combinado también con efectos dinámicos para acentuar el carácter interactivo de las mismas. Un tipo de efecto es el *rollover*, en el que todos los componentes, una opción o algunos de ellos cambian de aspecto al situar el usuario el puntero sobre ella.

Con capas, CSS y JavaScript (DHTML) es posible crear también menús dinámicos en los que aparecen y desaparecen porciones del mismo según las acciones que haga el usuario sobre sus opciones principales. De este tipo son los conocidos menús de árbol, similares al que ofrece el Explorador de Windows para navegar entre los discos duros y sus carpetas, y los menús de cortinillas, en los que aparecen y desaparecen capas con grupos de opciones. Un ejemplo de menú de árbol es mostrado en la siguiente figura. La opción “Grado” tiene descendientes que se muestran al seleccionarla.



Figura 1.7. Menú en árbol (desplegado para la opción Grado)

Otro tipo de menú muy aceptado es el de pestañas, que simula el aspecto de un clásico archivador de carpetas, apareciendo en primer plano la pestaña activa, en un color diferente y unido visualmente a la base común o al cuerpo de la página. Un ejemplo de este tipo de menú es mostrado en la siguiente figura:



Figura 1.8. Ejemplo de menú de pestañas

Un formato de menú muy extendido es el “estás aquí”. Este tipo de enlace presenta en forma textual una serie de enlaces que describen la ruta que ha seguido el usuario para llegar a la página actual a partir de la *home* o página de inicio, permitiendo regresar a cualquiera de ellas rápidamente. Estos menús poseen la ventaja adicional de ubicar al visitante en el total del sitio, con lo que éste sabe en cada momento dónde se encuentra y cómo ha llegado allí. Un ejemplo es el mostrado en la siguiente figura.



Figura 1.9. Ejemplo de menú “estás aquí”

Los menús son un elemento principal en todo sitio web porque permite que el usuario sepa en todo momento cómo moverse por el sitio y saber también dónde está. Por lo tanto, la ubicación de los menús es un aspecto muy importante en el diseño. Ésta debe permitir un cómodo acceso a las opciones que lo forman, pero sin llegar a estorbar al resto de elementos.

Los menús tipo lista y los de árbol se sitúan generalmente en la zona lateral izquierda de la página, mientras que el tipo pestaña o “estás aquí” es más habitual verlos en la parte superior, debajo de la cabecera. Esta distribución se ha convertido en un estándar de facto entre los diseñadores, pero el origen de esta costumbre está más ligado a cuestiones técnicas (muy asociada a la resolución y a HTML) que a motivos de usabilidad, funcionalidad o estética.

Si la altura de la página es tal que el usuario tiene que utilizar la barra de desplazamiento vertical tanto que pierde de vista el menú, es conveniente situar una versión reducida del menú principal en el pie de página, para que pueda acceder directamente desde esa posición a las partes del sitio.

Si el menú ofrece un número excesivo de opciones (cinco o más), es aconsejable utilizar menús dobles o menús en forma de árbol que jerarquice las opciones. Esto permitirá que el usuario encuentre la opción deseada con mayor facilidad, tal y como se muestra en la Figura 1.6.

En caso de ser necesario, el segundo menú (menú secundario) deberá diseñarse de forma que se identifique claramente como tal, siendo habitual mantener el menú principal como elemento general de navegación del sitio web completo y utilizar el menú secundario para permitir la navegación entre las diferentes páginas de una sección o nivel concreto.

Ejemplo de un sistema doble muy común es el formado por un menú principal lateral y uno secundario ubicado en la zona superior del cuerpo principal de la página, que puede ser de tipo “estás aquí”. Otra modalidad común es la formada por un menú principal horizontal bajo la cabecera y uno secundario en el lateral, aunque es posible cualquier combinación lógica y funcional.

ACTIVIDADES 1.3



- Analice los siguientes sitios web desde el punto de vista de la navegación. ¿Cómo de fácil es moverse por el sitio?

<http://www.zinccbistroaz.com/>
<http://www.thinkingforaliving.org>
<http://www.ucm.es>

1.4.3 EL CUERPO DE LA PÁGINA

El *cuerpo* es la parte de la página web donde se presenta al usuario toda la información referente a los contenidos de la página. Lo que aparece en el cuerpo suele ser el objetivo del sitio, lo que el usuario quiere ver. Por lo tanto, el espacio destinado a ella debe ser el mayor de todos, ocupando generalmente entre el 50% y el 85% del total. Su ubicación es siempre central, bajo la cabecera (si la hay) y al lado del menú lateral de navegación (si lo hay).

Los contenidos específicos del cuerpo de la página variarán según sea una página textual, un formulario, una ficha, una tabla o una página mixta, pero aparte de estas particularidades, existirán algunos elementos característicos de esta zona, que deberán estar presentes generalmente en todos los casos.

Es habitual que el cuerpo central lleve un título que identifique claramente la página a la que ha accedido el usuario. Este título se situará en la parte superior de esta zona y puede ser reforzado mediante un menú de navegación tipo “estás aquí”. El tamaño de las letras del título de página debe ser superior al del resto de los contenidos (como ocurre en los periódicos), con la finalidad de resaltar. Sin embargo, ésta no es la única manera de resaltar el título con respecto al resto del contenido. Otra alternativa es cambiar el color del título con respecto al contenido. Si el contraste es significativo entre ambos colores, se consigue también resaltarlo.

Es importante que todos los elementos gráficos que situemos dentro del cuerpo de página presenten un aspecto similar al del resto de elementos de la interfaz, respetando el estilo de todo el sitio. La siguiente imagen muestra parte de una página en la que el título se resalta con colores y tamaño distintos del contenido y donde el contenido tiene el mismo diseño que el resto de partes de la página (cabecera y menú).



Figura 1.10. Cuerpo respetando el estilo de la cabecera

1.4.4 EL PIE DE PÁGINA

El *pie de página* es la parte de una interfaz web situada en la parte inferior de la misma, bajo el cuerpo de página. En principio no parece tener una misión muy importante, sin embargo tiene mucha utilidad por la información que muestra y por ayudar a una percepción más estructurada del sitio.

Un uso muy común del pie de página es para mostrar enlaces a servicios muy particulares del sitio web, como contratación de publicidad, formulario de contacto, ofertas de empleo, condiciones de uso, políticas de seguridad, etc. Otro uso común es para mostrar información sobre la empresa propietaria del sitio web o de su responsable directo¹².

Como se comentó al hablar de los menús, si la página necesita de mucho movimiento vertical para poder visualizarse entera (usando barra de desplazamiento) el pie de página suele contener un menú auxiliar que permita al usuario continuar navegando por el sitio web sin tener que volver a buscar el menú principal.

Los contenidos del pie de página pueden aparecer alineados de cualquiera de las formas aceptadas (a la izquierda, centrados, a la derecha o justificados), aunque lo normal es que aparezcan centrados en pantalla. Si aparecen alineados de otra manera, siempre deberá estar en consonancia con el resto de elementos de la página. Las siguientes figuras muestran dos ejemplos de pie de página: uno para una institución pública (Universidad de Castilla-La Mancha) y otro del portal *amazon.es*.



Figura 1.11. Pie de página de la UCLM



Figura 1.12. Pie de página de Amazon.es

1.4.5 LOS ESPACIOS EN BLANCO

Aunque parezca mentira, un elemento de especial importancia en un diseño web son los *espacios en blanco*. Los espacios en blanco se definen como todas esas zonas de la interfaz en las que no hay ningún otro elemento gráfico. Entre sus objetivos está el compensar el peso visual del resto de elementos, crean márgenes o separaciones entre ellos, encuadrándolos de forma adecuada, y marcan los límites que estructuran la composición, haciendo la interfaz más equilibrada, limpia y bella.

¹² Esto es debido en España a la nueva Ley de Servicios de la Sociedad de la Información (LSSI) y del comercio electrónico, que obliga a todos los sitios web que generen beneficios directos (ventas) o indirectos (publicidad) a mostrar en cada página el nombre de la empresa o responsable y una dirección física o de correo electrónico válidas. Esta información se suele complementar con el número de teléfono y fax e información sobre *copyright* de los contenidos de la web.

Para muchos expertos en diseño web, la forma correcta es diseñar considerando desde el principio a los espacios en blanco como un elemento gráfico más, concibiendo su presencia y su ubicación desde el principio. Los espacios en blanco establecen el lugar, la rejilla base de la composición, que delimita las zonas en las que vamos a situar el resto de elementos, los márgenes y separaciones que van a existir entre ellas. A continuación se muestran algunas consideraciones concretas sobre los espacios en blanco.

Si existe un menú lateral de navegación es conveniente dejar siempre un espacio blanco o libre entre éste y el cuerpo de la página. Habrá que dejar, al menos, el mismo espacio entre la cabecera y el cuerpo de página. Si no existe cabecera, la separación será entre el cuerpo y el borde superior de la ventana útil del navegador. Si hemos diseñado una página con dos menús laterales, uno a cada lado, la separación entre estos y el cuerpo de la página será la misma en ambos casos, así como la separación entre los dos menús y los bordes de la ventana. De la misma manera, deberá existir un espacio en blanco de margen entre el dintel o el menú superior y el cuerpo de la página, así como entre éste y el pie de página, que deben tender a ser del mismo alto, buscando la simetría en la composición.

Todas estas separaciones son necesarias para conseguir un diseño poco sobrecargado en el que se delimitan bien las partes de la página. Observar el importante papel que juegan los espacios en el portal de iGoogle, dando una estructura muy bien definida a todo el sitio web.

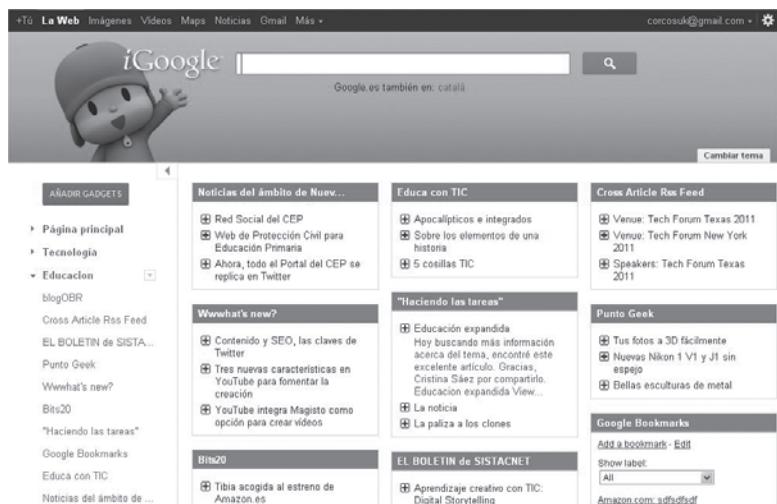


Figura 1.13. Los espacios en blanco en iGoogle

ACTIVIDADES 1.4



► Identifique en estos sitios web los elementos descritos en esta sección.

- <http://stonelab.osu.edu/>. Universidad de Ohio (EE.UU.).
- <http://www.educa.jccm.es>. Consejería de Educación de C-LM.
- <http://www.gio.upm.es/>. GIO – Universidad Politécnica de Madrid.
- <http://www.zara.com>. Tiendas Zara.

1.5 LENGUAJES DE MARCAS

Una vez se han visto los elementos principales de un sitio web, es necesario introducir los lenguajes de creación de la web. De alguna manera, la web es un programa que se ejecuta en el ordenador y que se define mediante lenguajes de programación que entiendan los navegadores. Por sus características, estos lenguajes son de un tipo especial y reciben el nombre de lenguajes de marcas. HTML; XML o RDF son algunos de esos lenguajes.

Un lenguaje de marcado o lenguaje de marcas es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. Es decir, un lenguaje de marcas es un tipo de lenguaje que combina texto con información extra acerca del texto. Esta información extra se entremezcla con el texto.

Los lenguajes de marcado son un tipo de lenguajes de programación. Sin embargo, no son exactamente lo mismo, ya que el lenguaje de marcado no tiene funciones aritméticas o variables, como sí poseen los lenguajes de programación.

El lenguaje de marcas más conocido en la actualidad es el HTML (*HyperText Markup Language*, Lenguaje de marcado de hipertexto), que se utiliza en las páginas web y fue propuesto por Tim Berners Lee en 1989, considerado el padre de la Web.

El lenguaje HTML (*Hyper Text Markup Language*) es lo que nos permite describir los contenidos de una página web de forma textual y estructurada; es un lenguaje para que el navegador conectado a Internet sepa cómo visualizar una página web; es decir, para que se pueda saber si un texto es un título, si éste está centrado o si tiene un tamaño determinado, también permite definir vínculos o enlaces a otras páginas o documentos y gestionar imágenes.

La primera versión de HTML solo mostraba texto con estilo, es decir, contenía etiquetas para títulos, párrafos, listas y viñetas, entre otras características. Muchas empresas utilizaron esta versión de lenguaje como punto de partida, pero incorporando sus propias interpretaciones de las etiquetas. Para organizar el crecimiento de HTML y de la propia Web se creó el World Wide Web Consortium (W3C), que se encargó desde entonces de estandarizar todos los temas relacionados con la Web. Así, en 1995 se publicó la versión 2.0 de HTML.

El lenguaje HTML 2.0 aportaba compatibilidad con los navegadores y etiquetas adicionales con las que incorporar imágenes, vínculos, tablas y formularios. Con estos últimos se lograba incorporar facilidades de interactividad a nivel de intercambio de información entre el navegador, utilizado por parte del usuario, y un servidor suministrador de las páginas web.

Una versión adicional del estándar HTML, la 3.0, apareció para dar respuesta a nuevas incorporaciones y propuestas de más etiquetas procedentes de distintas empresas, como Nestcape y o Microsoft.

Paralelamente, otros lenguajes para Internet fueron apareciendo, como PHP (*Hypertext Preprocessor*) y ASP (*Active Server Pages*), que se propusieron para funcionar con bases de datos y aprovechar la interactividad que puede ofrecer la Web.

HTML 4.0 se publicó el 24 de Abril de 1998, entre sus novedades destacaron la utilización de hojas de estilos CSS, la posibilidad de incluir pequeños programas o *scripts*, la mejora de la accesibilidad de las páginas diseñadas, tablas complejas y agilidad en los formularios. Con este nuevo estándar llegó una plataforma para entretenér a los navegantes de Internet con juegos y aplicaciones desarrollados en otros lenguajes, como Java.

En el año 2004, las empresas Apple, Mozilla y Opera se organizaron para formar la asociación Web Hypertext Application Technology Working Group (WHATWG), cuya actividad se centra en el actual estándar HTML5. HTML5 es una realidad actualmente y ya son muchos los navegadores importantes que lo implementan. HTML5 es una revolución en el desarrollo web al integrar en el propio lenguaje de marcas tanto el diseño como la inclusión de elementos multimedia¹³, entre otras funcionalidades. Un ejemplo de uso de HTML5, como sustituto de Adobe Flash para hacer animaciones y juegos, es Cut the Rope, un juego de ingenio implementado en 2012 como promoción de Internet Explorer¹⁴.

La Web, igual que los lenguajes que se utilizan para su desarrollo, ha pasado por diferentes etapas y hay otras fases en el horizonte. En la siguiente figura se muestra la evolución de la Web atendiendo a retos que ésta pretende acometer. Así se habla de Web 1.0 (Web informativa), Web 2.0 (Web colaborativa), Web 3.0 (Web semántica) y Web 4.0 (Web ubicua).

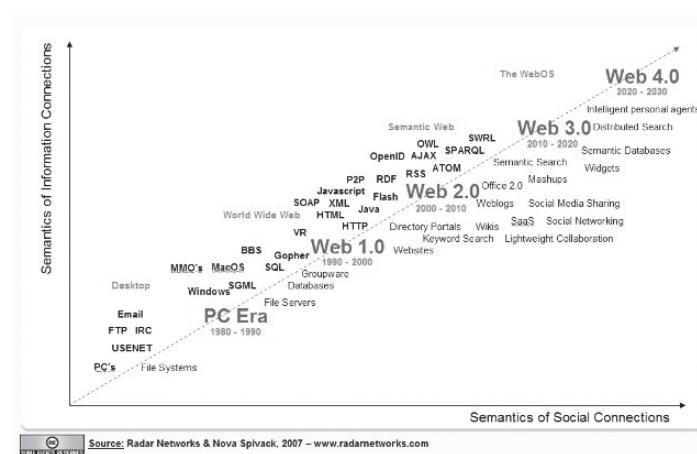


Figura 1.14. Evolución de la Web y sus lenguajes de marcas

En este punto, hay que entender que entre las competencias de un diseñador web no está solo en saber usar lenguajes de programación para crear sitios web, sino que es igual o más importante que el diseñador sepa plasmar lo que cada usuario quiere comunicar con su sitio web, además de dar soluciones factibles a los cambios que se puedan producir durante el desarrollo del sitio o posteriormente.

Por lo tanto, para el desarrollo de un sitio web no hay que comenzar a crearlo con el lenguaje de marcas HTML o cualquier otro. Antes de programar el diseñador debe planificar: debe saber qué quiere ofrecer y qué objetivos persigue el sitio web (qué quiere comunicar) y, posteriormente, trabajar la elaboración de distintas soluciones de diseño antes de decantarse por una opción u otra. En las siguientes secciones se muestran opciones para planificar correctamente un sitio web, de manera que los cambios que se quieran hacer sobre el diseño o los contenidos sea posible hacerlos en un tiempo adecuado.

¹³ El uso de HTML5 para incluir elementos multimedia es tratado en el Capítulo 3.

¹⁴ <http://www.cuttherope.ie/>

1.6 MAQUETACIÓN WEB. ELEMENTOS DE ORDENACIÓN

Por maquetación web se entiende la distribución, en el espacio considerado y disponible, de los elementos que conforman una página web. En otras palabras, maquetar es colocar las diferentes partes de una página dentro de sus límites. Evidentemente, esta idea está muy ligada con lo visto en las secciones anteriores.

La ventaja principal de maquetar es mantener separado el contenido de la página de la presentación, es decir, que si hay cambios en los contenidos no tenga que tocarse el diseño y viceversa. De este modo, se hace más sencillo el mantenimiento y los cambios al contenido y diseño que se tengan que hacer. Sin embargo, ésta no es la única ventaja. Por el lenguaje que se usa actualmente para la maquetación de sitios web, ésta es interesante para reducir el tiempo de desarrollo y el tiempo que el usuario debe esperar a que se cargue completamente el sitio.

Hace unos años, la maquetación de las páginas web se realizaba utilizando tablas (etiquetas <table><tr><td> de HTML). Una vez entendido este proceso podía resultar sencillo, aunque si no se dominaban las tablas, podía convertirse en algo tedioso. El problema de las tablas es que generaban una página muy encorsetada y el código se volvía complejo de entender. Además, algunos buscadores encontraban problemas al analizar la estructura de la página y su uso puede causar problemas de accesibilidad e interpretación de los contenidos organizados en ellas.

Actualmente, la maquetación con tablas ha caído en desuso y se realiza utilizando capas (etiqueta <div> de HTML), también llamadas divisiones o contenedores. La colocación de capas en una página web se realiza a través de hojas de estilo o CSS, concepto que se tratará en el segundo capítulo de este libro y que es clave para entender el diseño actual de los sitios web. Las capas permiten, por ejemplo, que se pueda pasar de un diseño con un menú lateral a otro con el menú en la parte superior, solo cambiando la hoja de estilos.

1.6.1 DISTRIBUCIÓN DE ELEMENTOS EN LA INTERFAZ: CAPAS, MARCOS

En esta sección se describen dos de los elementos clave en la maquetación web: **capas** y **marcos**.

Las *capas*, también llamadas DIV o *layout*, son como contenedores donde se colocan imágenes, textos o incluso, otras capas. Las principales características de las capas son las siguientes:

- Las capas pueden estar anidadas, es decir, pueden estar unas dentro de otras. Básicamente, lo que se hace es definir cómo se posiciona en la página web, su colocación y su tamaño.
- Las capas son bloques con contenido HTML que pueden posicionarse de manera dinámica y anidarse. Las ventajas que ofrecen las capas solo se pueden aprovechar al cien por cien utilizando estilos CSS.

En realidad, las capas no se definen completamente mediante el lenguaje HTML, sino que necesitan del lenguaje de definición de estilos CSS¹⁵ (que se verá en el siguiente capítulo). Con uno y otro lenguaje se pueden incluir en las páginas web elementos móviles, ocultables y, en general, manipulables de forma dinámica.

¹⁵ En el Capítulo 2 se trata este tema ampliamente.

Por ejemplo:

```
<STYLE TYPE="text/css">
    .CapaFija {position: absolute; top:100px; left:20px; width:200; height: 100}
</STYLE>
```

Con esto hemos definido un tipo de capa, denominada *CapaFija*, cuya altura es de 100 píxeles (un píxel depende de la resolución y tamaño de la pantalla, por ejemplo, en una pantalla de 1024×768 , un píxel en horizontal son 0,329 mm y en vertical son 0,35 mm) y la anchura de 200 px. Además, está situada a 100 px de la parte superior y a 20 px del margen izquierdo de la página. Con el código anterior se ha definido una clase *capa*, pero todavía no se ha construido la capa. Para construirla se utiliza la etiqueta `<DIV>` y el atributo ID, tal y como se muestra a continuación:

```
<DIV ID="capa">
    <H1>Esto es contenido</H1>
    <P>Aquí sigue más contenido HTML </p>
    ...
</DIV>
```

Cualquier bloque `<DIV>` con ID="capa" estará en esa posición y con ese tamaño.

Esta capa puede colocarse en cualquier parte de la ventana, su posición es absoluta (*absolute*). Pero también podemos definir capas de posicionamiento relativo, es decir, que más que definir las coordenadas de su posición respecto a la ventana, describimos su posición respecto al lugar donde aparezca en el texto. En otras palabras: describimos el desplazamiento de la capa respecto de donde la ponemos. Se definen así:

```
<STYLE TYPE="text/css">
    .CapaRelativa {position: relative; left: 20px; top: 100px;}
</STYLE>
```

Los ejemplos anteriores definen y construyen dos capas, una absoluta y otra relativa. Esta última está construida con una etiqueta, ``, para evitar el salto de línea propio de los bloques.

Además de las capas, otra alternativa para la maquetación son los *marcos (frames)*, representados en HTML con etiqueta `<frameset>` y `<frame>`. Su uso mueve y ha movido controversia entre algunos diseñadores y adhesión por parte de otros. Los marcos son una forma de insertar varias páginas web en una sola. Los marcos dividen la página web en varias partes y dentro de cada parte se incluye otra página web. La idea es parecida a las capas, pero dentro de cada marco en vez de haber texto, imagen u otra capa, hay una página web.

Mal utilizados pueden arruinar la mejor página web, puesto que la pantalla del monitor está físicamente limitada. Cada marco que compone la página poseerá sus propios bordes y barras de desplazamiento, comportándose como ventanas independientes. Su situación en la página es rígida, no podemos colocarlos en las posiciones que deseemos, si tenemos cuatro marcos se situarán en cada uno de los cuatro cuadrantes de la pantalla. Si tenemos dos la pantalla se dividirá en dos filas o en dos columnas para alojarlos.

El problema principal de los *marcos* es que algunos navegadores no lo pueden manejar. Esto requiere que el diseñador, mediante código incrustado en la página, controle esta posible situación. El siguiente código utiliza marcos e incluye código para controlar limitaciones del navegador:

```
<HTML>
<HEAD>
    <TITLE>Los frames: páginas multiventana</TITLE>
</HEAD>
<FRAMESET COLS="20%, 80%">
    <FRAME NAME="indice" SRC="indice.htm">
    <FRAME NAME="principal" SRC="principal.htm">
    <NOFRAMES>
        <P align="center">Al parecer tu navegador
        no soporta marcos, actualízate.</P>
    </NOFRAMES>
</FRAMESET>
</HTML>
```

Dentro de un `<FRAMESET>` se definen los marcos que componen el conjunto y la acción alternativa para navegadores que no soporten marcos. A cada uno de los marcos se le adjudica un nombre y se especifica qué página HTML se mostrará en él (etiqueta `<FRAME>`). En el ejemplo solo queda definir lo que verá el usuario en el supuesto de que su navegador no soporte marcos (etiqueta `<NOFRAMES>`).

Comparado con las capas, los marcos requieren de mucha habilidad en el diseño para ser la base de una maquetación. Sin embargo, en algunas situaciones pueden ser muy útiles. Por ejemplo, si se desea compartir un cierto contenido por todo el sitio web, la barra de navegación o la cabecera, los marcos son de gran ayuda.

Una excelente alternativa a los *marcos fijos* son los marcos flotantes, actualmente soportados por todos los navegadores. La idea de este elemento, ideado por Microsoft, sigue siendo la misma: incluir una página externa dentro de otra, pero en este caso el marco puede quedar totalmente integrado en la página contenedora.

1.7 MAPA DE NAVEGACIÓN. PROTOTIPOS

Como se puede apreciar de lo visto en las secciones anteriores, los sitios web pueden contener muchas páginas, todas ellas accesibles desde algún punto del sitio y todas con todos o algunos enlaces a las demás. Esta estructura de enlaces hace, en muchos casos, difícil que el diseñador o el usuario del sitio sepan qué páginas llevan a cuáles. Por ello, antes de diseñar un sitio web se debe realizar un esquema que permita anticipar cuáles son las secciones en las que estará dividida el sitio web y la relación entre los diferentes bloques de contenidos. Ese esquema recibe el nombre de *mapa de navegación* y es algo parecido al índice de contenidos de un libro, es decir, un manera de que el diseñador de un sitio web estucture bien los contenidos antes de crear el sitio y de que los usuarios encuentren más rápidamente lo que buscan una vez creado el sitio.

La siguiente figura muestra un mapa del sitio web de un Hotel-Restaurante. El mapa muestra cómo están relacionados los diferentes grupos de información. Un ejemplo de mapa de navegación o mapa de un sitio web ligado a una temática de institución educativo se muestra en la siguiente figura:

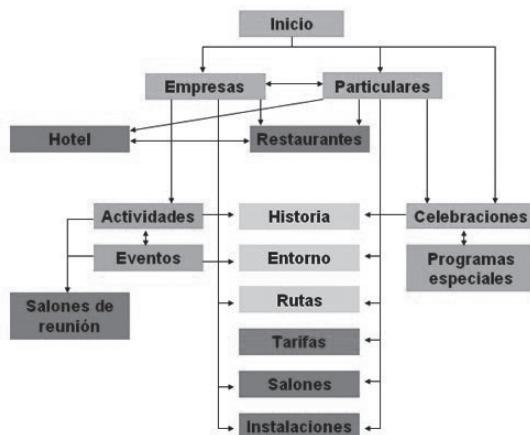


Figura 1.15. Mapa web de un Hotel-Restaurante

A la hora de realizar mapas de navegación hay que tener en cuenta que las páginas que forman un sitio web no deben aportar todas ellas la misma información ni cubrir el mismo objetivo, se podrá partir de una página principal (*home* o *homepage*) y, desde ella, poder acceder al resto de páginas que conforman el sitio web.

Además de los mapas de navegación, la propia complejidad del diseño hace que, en muchos casos, sea difícil entender qué es lo que el usuario quiere transmitir al resto del equipo de desarrollo y qué es lo que tienen que hacer. Por tal motivo, los *prototipos* son herramientas muy interesantes para ahorrar tiempo a la hora de determinar qué es lo que hay que hacer, ya que muestran un esquema de cómo quedará el sitio web, pero empleando mucho menos tiempo que si hubiese que hacerlo realmente.

Más detalladamente, un prototipo web es un borrador o modelo inicial a partir del cual se empieza a concebir y desarrollar la idea original del diseño de un sitio web. Hacer un prototipo es más sencillo y económico que hacer una web real y luego modificarlo hasta alcanzar lo que se busca. El prototipado de las páginas web resuelven básicamente los siguientes aspectos:

- Qué elementos deben conformar la interfaz de cada página.
- Qué elementos o características serán comunes a lo largo de las distintas páginas del sitio web.
- Cuántos elementos deben conformar la interfaz para que haya suficiencia en la información/interacción, pero evitando la saturación de elementos (de cada página).
- Cómo debe organizarse el mapa de navegación (en qué orden y disposición van las páginas).
- Qué extensión (superficie visual o tamaño) adecuada deben tener aprovechando eficientemente el espacio bidimensional disponible.
- Qué aspectos deben tenerse en cuenta a la hora de desarrollar el sitio web. Entre los aspectos especialmente interesantes, por su repercusión en los usuarios finales que usen o visiten un sitio web, están los aspectos técnicos, de usabilidad y de accesibilidad.

A la hora de realizar prototipos se puede separar la interfaz gráfica en dos grupos de elementos o componentes:

- Los elementos o componentes abstractos y comunes a toda página web, como pueden ser los presentados en la sección 1.6 de este mismo capítulo, como son las cabeceras, barras de navegación (vertical u horizontal), los pies de página, los formularios, etc.
- Los elementos concretos específicos de una parte o del total de una página web, que se utilizan con un objetivo y una apariencia concreta, por ejemplo, botones, enlaces, campos de texto, imágenes, texto, etc.

La siguiente figura muestra un prototipo de poco detalle comparado con el resultado final:



Figura 1.16. Prototipo de mucho detalle

Una vez que el diseñador ha realizado prototipos y el mapa de navegación, y estos han sido aprobados, el siguiente paso es abordar el desarrollo más detallado de un sitio web. En este paso se discuten otras características de los sitios web, como son las ligadas a la maquetación o el diseño detallado de dichos sitios. Para ello se puede trabajar con plantillas ya elaboradas o trabajar en el desarrollo de una propia. A este aspecto volveremos más adelante en este mismo capítulo.

1.8 INTERPRETACIÓN DE GUÍAS DE ESTILO. ELEMENTOS

Por su complejidad, para diseñar eficazmente interfaces web, son necesarias dos actividades: la **planificación** de qué se quiere hacer y la **coordinación** del equipo de desarrollo que se encarga del diseño. En la sección anterior se han visto dos técnicas para facilitar esta tarea. En esta sección se trata otra que también tiene relación con el diseño de sitios web dentro de entornos de desarrollo: la creación de una *guía de estilo*.

La *guía de estilo* es un documento (o varios) que *define las pautas y normas de calidad que debe seguir una interfaz web para un determinado sitio web*. Gracias a la guía de estilo se garantiza la coherencia del sitio, integrando toda la interfaz con un aspecto y uso homogéneos. La guía de estilo abarca aspectos de calidad de uso, accesibilidad, diseño

gráfico, marketing, etc., tocando temas como los colores y otros elementos de diseño, como estándares (de usabilidad, accesibilidad, etc.). Más concretamente, se puede decir que una guía de estilo para la interfaz de usuario sirve como:

- Una herramienta para garantizar la coherencia de un sitio web a través de las páginas web del sitio.
- Una técnica para conseguir integrar en un mismo objetivo a todos los miembros de un equipo de trabajo, ya que se establecen las pautas que todos deben seguir. Además, ayuda a la formación de nuevos miembros de un equipo de trabajo.

Como antes se ha mencionado, el objetivo de una guía de estilo es desarrollar sitios web coherentes. Pero, ¿qué es ser coherente? La coherencia tiene varias interpretaciones: coherencia con las expectativas del usuario, coherencia en todos los sitios web que están relacionados, coherencia en todos los sitios web que no están relacionados pero que provienen de la misma empresa, coherencia con las normas de facto (por ejemplo, el uso de enlaces azules para denotar los enlaces no visitados), coherencia de la terminología, coherencia de la interacción, coherencia visual, coherencia entre las páginas/diálogos/ventanas, coherencia en el uso de los iconos o coherencia de los mensajes de error.

No existe una estructura única que deban seguir las guías de estilo. Sin embargo, algunas de las preguntas que debe responder son: ¿Qué colores tendrá la web y tonos? ¿Qué fuentes se usarán? ¿Qué formato de fuente se usará para los títulos, subtítulos, encabezados y el texto principal? ¿Cuál será la estructura? ¿Habrá encabezado, pie de página o menús? ¿Habrá un menú o varios? ¿Cuántos y dónde colocarlos? ¿Qué imágenes se mostrarán? ¿Dónde se colocarán? ¿Habrá logotipo? ¿Dónde se colocará? ¿Se tratarán la accesibilidad de la página y criterios de calidad de uso?

Aunque gran parte de los elementos que forman una guía de estilo se tratan de forma implícita en este libro, en este punto se quiere hacer énfasis en su utilidad para todos los participantes en un desarrollo web, usuarios, desarrolladores o, incluso, el propio negocio vinculado al sitio web. Estos beneficios están relacionados con aspectos como: reducir cambios, facilitar el uso o mejorar la coordinación del equipo de trabajo.

ACTIVIDADES 1.5



- Acceda a las guías de estilo de los siguientes sitios web y haga una lista de los elementos comunes que reflejan todas ellas.

<http://www.ua.es/es/internet/estilo/guia/estilo.htm>. Universidad de Alicante.

http://es.wikipedia.org/wiki/Wikipedia:Manual_de_estilo. Wikipedia.

http://www.upv.es/entidades/ASIC/manuales/guia_estilos_upv.pdf. Universidad Politécnica de Valencia.

1.9 APPLICACIONES PARA DESARROLLO WEB

En la actualidad son muchas las herramientas que pueden utilizarse para facilitar tareas relacionadas con la planificación, diseño, desarrollo y mantenimiento de sitios web. El diseñador web no suele partir de cero, sino que se apoya en estas herramientas para desarrollar un sitio, ahorrando esfuerzo y focalizándolos en aspectos menos automatizables.

Las herramientas para desarrollar sitios web pueden clasificarse atendiendo a muchos criterios: su propósito, su coste, su alcance o la fase del proceso de desarrollo a la que den soporte, etc. Algunos grupos de aplicaciones útiles para abordar el desarrollo para la Web serían los siguientes:

- **General:** se incluirían en este grupo aquellos programas cuya utilidad es de interés general y de uso no solo exclusivo para los desarrolladores. Podrían incluirse en este apartado navegadores, herramientas software que faciliten realizar labores de planificación, tratamiento de imágenes o transferencia de ficheros (clientes FTP).
- **Diseño:** serían aquellos programas útiles para diseñar páginas web. Diseño web y diseño en general. Dentro de estos programas, existen programas comerciales, no comerciales y programas que permiten hacer prototipos iniciales para discutir distintas posibilidades de diseño antes de llevar esos diseños a su programación final. Ejemplos de este tipo de programas podrían ser Balsamiq¹⁶, Pencil¹⁷ o SketchFlow, de Microsoft.
- **Multimedia:** serían programas orientados a la gestión o creación de animaciones y otros componentes con los cuales se puede dar más dinamismo a los sitios web desarrollados. Ejemplos de este tipo de entornos son: Adobe Flash Professional, Silverlight o JavaFX.
- **Programación:** son programas enfocados a desarrolladores y programadores, con los cuales se elaboran páginas y sitios web. Dichos programas suelen estar ligados a lenguajes concretos o tecnologías para la Web. Dentro de esos lenguajes encontramos el lenguaje HTML, un lenguaje de marcas que será presentado en la siguiente sección; javascript (jQuery), PHP, ASP, ASP.NET, JSP o Ruby.
- **Editores y validadores HTML:** son programas para la edición de código HTML y para su comprobación, que ofrecen ayudas visuales específicas para construir webs, como editores WYSIWYG (*What You See Is What You Get*, lo que ves es lo que consigues). Ejemplos de ellos podrían ser Dreamweaver de Adobe o Kompozer (<http://www.kompozer.net/>), aunque cualquier editor de textos, serviría para escribir HTML.
- **Editores y validadores CSS:** programas que facilitan la creación, edición y comprobación de código CSS (hojas de estilo en cascada). Ejemplos de ellos serían Stylizer (<http://www.skybound.ca/>), Xyle Scope (<http://culturedcode.com/xyle/>) o CSS Toolbox (<http://www.blumentals.net/csstool/>).

Dentro de estas herramientas también se encuentran los gestores de contenidos CMS, los cuales se describen más ampliamente en la siguiente sección.

¹⁶ <http://www.balsamiq.com>

¹⁷ <http://pencil.evolus.vn>

1.10 GENERACIÓN DE DOCUMENTOS Y SITIOS WEB

Como se ha comentado antes, crear sitios web no tiene que ser en todos los casos algo que se empiece desde cero. Desde que aparecieron en escena los Gestores de Contenidos, conocidos por CMS (*Content Management Systems*), muchos son las empresas o instituciones que tienen creados sitios web con ellos, sin partir de cero. En España, a día de hoy se puede afirmar que Joomla¹⁸, OpenCMShispano¹⁹ o Drupal²⁰ son unos de los gestores de contenidos más extendidos.

Un gestor de contenidos se define como una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. De esa manera, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores.

El éxito de los gestores de contenido radica principalmente en que alejan los aspectos técnicos de desarrollo del diseño de la interfaz y, ambos, de la generación de la información y documentación que se quiere comunicar en el sitio. De alguna manera, se puede afirmar que con un gestor de contenidos, un administrador puede crear contenidos sin necesidad de saber nada allá que manejar un procesador de textos.

Los gestores de contenidos más extendidos suelen estar basados en tecnología web con lenguaje PHP/HTML y gestores de bases de datos MySQL. Esto hace que estos CMS sean de código y licencia libre (Joomla, por ejemplo, es GPL). Además, esta tecnología hace que su funcionalidad se puede extender para adaptarse a las necesidades de un determinado negocio.

Más concretamente, los gestores de contenidos guardan tanto los elementos de las páginas web como las especificaciones del diseño en bases de datos. La construcción de un sitio web se hace utilizando elementos de diseño predefinidos, llamados *plantillas*. Todos los elementos son leídos desde la base de datos, cargados automáticamente, puestos en el sitio preciso del diseño y presentados al usuario como página web. Esto garantiza aislar el diseño de los contenidos y la distribución de los componentes, pudiendo así cambiar el diseño sin tocar ninguno de los otros aspectos²¹.

Aunque los gestores de contenidos están muy extendidos como base de sitios web, es un error pensar que al existir este tipo de plataformas el desarrollo web, como negocio, no tiene sentido. En contra de lo que se pueda pensar, este tipo de plataformas no hace otra cosa que potenciar la necesidad de desarrolladores y diseñadores web. Actualmente, existen muchas empresas software que desarrollan componentes y módulos concretos para ser usados en estos gestores de contenidos y, al mismo tiempo, existen también empresas interesadas en diseñar plantillas (*templates*) para ser incluidas en estos entornos. Sin duda, el uso extendido de los gestores de contenidos ha abierto un abanico de posibilidades para los desarrolladores web.

¹⁸ <http://www.joomlaspanish.org/>

¹⁹ <http://www.opencmshispano.com>

²⁰ <http://drupal.org.es/>

²¹ En <http://www.iesleonardodavinci.com/> se puede ver el portal de un centro educativo de Albacete, soportado en Joomla 1.6.

ACTIVIDADES 1.6



- Acceda a los siguientes portales de centros educativos hechos con Joomla, y saque al menos 5 elementos comunes en todos ellos (observar que a pie de página aparece la versión de Joomla con la que están hechos).

<http://edu.jccm.es/ies/losbatanes/>
<http://www.iesleonardodavinci.com/>
<http://edu.jccm.es/ies/molina/>

1.11 PLANTILLA DE DISEÑO

Las plantillas de diseño web son la mejor opción para disponer de un sitio web diseñado de forma profesional y atractiva sin necesidad de realizar una inversión elevada en tiempo en su desarrollo²².

Las plantillas son sitios web prediseñados, de forma que ya disponen de una estructura definida y solo hay que incorporar los contenidos particulares del sitio web y desarrollar todas las páginas que lo conforman sin preocuparse del aspecto. Por lo tanto, permiten desarrollar el sitio web de una forma mucho más ágil y rápida que los diseños a medida (proyectos en que se parte de cero).

Las plantillas web son adecuadas para aquellos sitios web que no van a requerir de una estructura compleja y en los que su función principal será la de mostrar información general sobre la propia empresa, negocio o servicios que ofrece.



Figura 1.17. Plantillas web

²² En el Capítulo 2 se trata, al final, el tema de las plantillas de diseño disponibles en Internet.

La figura previa muestra distintas páginas web elaboradas utilizando plantillas. Dichas plantillas aportan la componente visual y llevan asociadas hojas de estilo y código HTML, desarrollado independientemente del contenido que se muestre en dicho sitio web.

ACTIVIDADES 1.7



- Busque en un motor de búsqueda las palabras "*plantillas diseño web html5*" y encuentre 2 plantillas que le gusten, que se puedan descargar gratuitamente, y 2 de pago.

1.12 INTERACCIÓN PERSONA-ORDENADOR

Una vez vistos los elementos principales que afectan al diseño de sitios web, en este punto se muestra el *marco* que afecta a todo desarrollo web que incluya interacción con los usuarios. La Interacción Persona-Ordenador (IPO), que así se llama ese marco, *es la disciplina que estudia el intercambio de información entre las personas y los ordenadores* (el término en inglés es *HCI, Human Computer Interaction*). Su objetivo es que este intercambio sea más eficiente, minimizando errores e incrementando la satisfacción. La IPO ofrece un marco empírico con todo lo referente a calidad de uso e interacción de interfaces de usuario. Aunque parezca que todo lo relacionado con interfaces de usuario sea algo exclusivo de técnicos en desarrollo de aplicaciones o ingenieros relacionados con la Ingeniería del Software, realmente, la IPO es la disciplina que sin duda más tiene que decir en este campo.

Los orígenes de la IPO hay que buscarlos en la rama de la Psicología Aplicada que estudia la Interacción Persona-Ordenador. La Psicología es la disciplina que estudia la percepción, la memoria, la adquisición de habilidades y el aprendizaje, la resolución de problemas, el movimiento, las tareas de juicio, de búsqueda o procesamiento de información y de la comunicación, es decir, procesos todos cuyo conocimiento se requiere para el adecuado diseño de mecanismos de interacción del usuario. Aunque la Psicología Cognitiva es una ciencia muy joven en lo que respecta a investigaciones de carácter básico y sistemático, existen actualmente suficientes hallazgos basados en resultados empíricos que permiten el desarrollo de la IPO y, por ende, de sitios web adaptados a los usuarios.

La IPO es una disciplina joven, sin embargo, los estudios han permitido dar una base teórica al diseño y a la evaluación de aplicaciones informáticas (en los que se incluyen, claro está, el desarrollo web). La importancia de esta disciplina para cualquier diseñador/desarrollado web se pone sobre relieve al leer artículos sobre el tema escritos hace cuarenta años en los que se predecían elementos de interacción de los que se disponen actualmente. Una de las asociaciones más influyentes en este campo es la ACM SIGCHI (*Association for Computing Machinery's Special Interest Group on Computer-Human Interaction*) que desde 1982 reúne a los mejores especialistas en IPO.

Para un diseñador web, es muy importante la labor que se hace dentro del marco de la IPO, ya que gracias a las investigaciones punteras en este campo, se mejora día a día la manera de interaccionar con los sistemas informáticos. La lectura de publicaciones o estándares desarrollados dentro de la disciplina IPO permite al diseñador web estar actualizado con los avances de la disciplina y conocer el futuro de su profesión.

1.13 CONCLUSIÓN

El capítulo introduce los elementos básicos a la hora de planificar un diseño web. Los colores, fuentes, navegación o estructura de una página son algunos de esos elementos. Sin embargo, no basta con conocerlos, sino también es necesario colocarlos en el sitio adecuado, según las recomendaciones de la IPO y de los estándares de facto de los sitios web actuales.

Por otro lado, el diseño no es algo aislado del desarrollo de aplicaciones web, sino que forma parte de él. Por ello, es necesario que el diseño venga acompañado de una guía de estilo que permite a cualquier involucrado o interesado en el sitio web conocer las reglas básicas de diseño de ese sitio. Además, la tecnología empleada en el desarrollo web condiciona mucho las posibilidades a la hora de planificar el diseño web, por lo que es necesario conocer aplicaciones y lenguajes relacionados que puedan limitar o ampliar el abanico de posibilidades del diseño.

En conclusión, los contenidos de este capítulo son necesarios para, por un lado, entender muchos de los conceptos del resto del libro y, por otro, saber integrar el diseño en el complejo entramado de tecnología que conlleva el desarrollo de aplicaciones web.



RESUMEN DEL CAPÍTULO



Planificar la interfaz de un sitio web es una de las partes más importantes dentro del desarrollo de aplicaciones web. El diseño gráfico elegido y la manera en la que los usuarios interactúan con la aplicación son algunos de los aspectos más destacados para determinar el grado de aceptación de un sitio y, por tanto, de la institución o empresa a la que el sitio web representa.

El capítulo ha introducido los conceptos básicos del diseño web desde dos perspectivas: la perspectiva propia del diseño y la perspectiva del desarrollo de aplicaciones web.

Respecto a la primera perspectiva, el capítulo hace una introducción a los elementos básicos del diseño web: colores, fuentes, cabecera o navegación son algunos de los puntos que se tratan. La manera en la que el diseñador configura estos elementos influye en el resultado final y, por tanto, el capítulo incluye recomendaciones actuales para un empleo de los elementos más eficaz y efectivo.

Respecto a la segunda perspectiva, el capítulo trata las guías de estilo, documento de comunicación entre el diseño y el resto de involucrados en un desarrollo web. Pero, además, por estar condicionado el diseño web con la tecnología empleada en el desarrollo, el capítulo analiza aplicaciones que relacionan el desarrollo y el diseño. Un ejemplo es el uso de gestores de contenidos - CMS.



EJERCICIOS PROPUESTOS



■ 1. Acceda a la página de MoWes (<http://www.chsoftware.net/en/mowes/mowesmixer/mowesmixer.htm>) y descárguese en local (pendrive si lo prefieres) un servidor de Joomla!. Para ello necesita Apache, MySQL y PHP, además del propio Joomla!. Hecho esto, póngalo en marcha según las instrucciones de MoWes. Para más información sobre el manejo de Joomla! se puede consultar:

- <http://edu.jccm.es/joomla15/index.php/introduccion.html>. Portal de la Consejería de Educación de Castilla-La Mancha, con vídeo-tutoriales sobre su manejo.
 - <http://www.joomlaspanish.org/>. Comunidad Joomla! en Español.
- 2. Descargue las últimas versiones de los navegadores: Mozilla Firefox, Google Chrome, Safari e Internet Explorer. Compruebe con el servicio web <http://html5test.com/> qué características de HTML5 soporta cada uno. ¿Soportan todos las mismas?
- 3. Por parejas, se propone que cada miembro del grupo selecciona en Internet un sitio web que le llame la atención. Argumente a su compañero por qué su sitio elegido tiene un mejor diseño que el suyo: apunte en una hoja qué características del diseño destaca en su argumentación. ¿Los colores?, ¿la rapidez de carga?, ¿las imágenes?, ¿cómo muestra los contenidos?, ¿lo fácil que es encontrar información?, etc.
- 4. Continuando con el ejercicio anterior: cada miembro del grupo tiene que hacer una tabla en la que refleje aspectos que considere positivos y negativos del sitio web que ha elegido respecto a:

- Colores.
- Estructura.
- Iconos.
- Tipografía.
- Animaciones.
- Facilidad de navegación.
- Visionado de los contenidos que muestre.
- Tecnología empleada para su desarrollo.
- Animaciones.

Sobre esa tabla, entre los dos miembros del grupo, atendiendo a las ventajas y desventajas de cada sitio web, debe asignar una puntuación (0-5) a cada ítem y luego sumar el resultado. Con esta tabla ya puntuada, ¿es más fácil argumentar cuándo un sitio es mejor que otro?

- 5. Diseñe e implemente una página web para un grupo musical que le guste. El tamaño de la página no debe ser mayor que lo que se puede ver en el navegador sin usar la barra de desplazamiento. Asigne los colores, tipografía e iconos que le parezcan oportunos. Incluya imágenes y vídeos si lo cree oportuno. Una vez terminado, pase su diseño a otros cuatro compañeros para que lo valoren según los criterios del ejercicio anterior. ¿La valoración de los compañeros es positiva o negativa? ¿Diría que puntúan según criterios objetivos o subjetivos? ¿Se puede sacar una opinión común entre los cuatro compañeros para todos los ítems valorados?



TEST DE CONOCIMIENTOS



1 ¿Qué es la IPO?

- a) Un organismo para hacer sitios web.
- b) Una disciplina que estudia la interacción entre personas y máquinas.
- c) Una publicación sobre avances en el campo de la interacción entre personas y máquinas.

2 Con la aparición de CMS:

- a) No tiene sentido desarrollar portales web desde cero. Conocer CSS y HTML pierde su sentido.
- b) El diseño de las páginas web viene hecho con plantillas. El diseñador no tiene nada que hacer.
- c) Acelera el desarrollo de portales web, aunque todos los portales al final requieren cierta personalización.

3 Para el desarrollo de un sitio web, lo más optimo es:

- a) Apoyarse en herramientas software que faciliten la generación del código y la realización de diseños.
- b) Usar solamente un editor de textos y programar desde HTML directamente.
- c) Atarse a una tecnología y utilizarla al menos durante 10 años.

4 Una guía de estilos de un sitio web:

- a) Es una pérdida de tiempo, ya que la mayoría de los sitios web los desarrolla una única persona.
- b) No se suele utilizar ya que lleva mucho trabajo confeccionarla. En el desarrollo web hay que eliminar tareas superfluas y centrarse en hacer líneas de código.
- c) Es necesaria para mantener un sitio web siempre con el mismo estilo, con independencia de que las personas que los mantengan cambien.

5 HTML5:

- a) Es un lenguaje de marcas muy extendido actualmente.
- b) No mejora mucho con respecto a HTML4.
- c) Es un lenguaje de marcas propietario con licencia de Adobe.

6 Un portal web, con independencia de para qué se use, siempre debe estructurarse:

- a) Con pie de página, cabecera, cuerpo y navegación, obligatoriamente.
- b) Con cabecera y menú, obligatoriamente.
- c) Con pie de página, cabecera, cuerpo y navegación, pero siempre opcionalmente. Todo depende de qué se quiere mostrar con él.

2

Uso de estilos

OBJETIVOS DEL CAPÍTULO

- ✓ Identificar las posibilidades de modificar las etiquetas HTML.
- ✓ Ser capaz de definir estilos de forma directa.
- ✓ Ser capaz de definir y asociar estilos globales en hojas externas.
- ✓ Ser capaz de definir hojas de estilos alternativas.
- ✓ Identificar las distintas propiedades de cada elemento.
- ✓ Ser capaz de crear clases de estilos.
- ✓ Ser capaz de utilizar herramientas de validación de hojas de estilos.

Las hojas de estilo en cascada o CSS (*Cascading Style Sheet*) son una parte muy importante dentro del diseño de aplicaciones web. CSS es un estándar del Consorcio WWW o W3C, ampliamente reconocido y utilizado debido a dos aspectos importantes que ofrece: ahorrar tiempo en el diseño de sitios web y conseguir efectos potentes, soportados por la mayoría de los navegadores.

Este capítulo se centra en los aspectos básicos de CSS. Se mostrarán los elementos más comunes y se ilustrarán con ejemplos prácticos. Todo ello para que un diseñador conozca este lenguaje y pueda desarrollar sus propias plantillas y/o interpretar plantillas CSS ya existentes. Sin embargo, CSS es un estándar muy extenso, y no es objetivo de este capítulo trabajarla en su totalidad. Una vez el lector entienda el funcionamiento de CSS, podrá utilizar Internet para consultar el estándar en W3C, sitios web o bibliografía especializada en CSS para conocer aspectos avanzados (trucos y soluciones específicas).

2.1 INTRODUCCIÓN A HOJAS DE ESTILO EN CASCADA (CSS, CASCADING STYLE SHEET)

Entre todos los avances que en la última década se han producido en el campo del diseño web, la aparición de CSS (*Cascading Style Sheet*, Hojas de estilos en cascada) ha sido uno de los más significativos. Su principal ventaja es que permiten separar en el desarrollo de un sitio web lo que es el diseño (apariencia) de lo que son los contenidos (información que se quiere transmitir). Esto tiene como efecto inmediato un desarrollo y mantenimiento más eficiente de sitios web.

Más técnicamente, la filosofía de las CSS se puede resumir así: usar la etiqueta `<body>` de HTML para definir las estructuras de los contenidos que se muestran en el sitio (encabezados, párrafos, viñetas, etc.) y, luego, en otro archivo o en el `<head>` del HTML, se define la apariencia de cada página usando el lenguaje de CSS. Esto permite que se puedan cambiar los contenidos que se pongan en el `<body>` sin afectar a la apariencia definida en el archivo CSS (o en el `<head>` del HTML), o que se cambie la apariencia en el CSS sin que afecte a nada de lo puesto en el `<body>` del HTML.

Otra de las grandes ventajas actuales de los CSS es la adaptación de los sitios web a los dispositivos con los que será visualizado. Un sitio web puede ser visto desde un iPad, un móvil con Android, un ordenador personal o cualquiera de los dispositivos disponibles actualmente. La combinación de CSS y HTML permite crear sitios web personalizados para cada dispositivo. Cuando el servidor detecta el dispositivo cliente (*Media Queries*), éste aplica una hoja de estilos para un mejor visionado. Evidentemente, cada una de las hojas de estilos para cada dispositivo debe haber sido diseñada a propósito por el diseñador, aquí no hay magias.

La organización W3C fue la encargada de estandarizar la versión CSS1. W3C definió CSS como hojas de estilo en cascada ya que es posible que un mismo contenido pueda ser regido por varios archivos CSS que controlen su apariencia. Lo que el estándar CSS1 de W3C pretendía era establecer los criterios por los que se da preferencia a un estilo respecto a otro (por eso lo de cascada). Un mismo elemento, como el encabezamiento `<h1>` de la página puede estar definido en un archivo CSS externo para aparecer como texto azul; en la propia página, definirlo como texto rojo. En este caso (muy común) el navegador tiene que optar por uno u otro estilo. Por lo tanto, la especificación W3C marca las pautas de preferencia que debe respetar un navegador compatible con CSS.

CSS1 alcanzó el status de recomendación por la W3C en el año 1996. Las reglas CSS1 tienen un soporte adecuado en prácticamente todos los navegadores modernos más conocidos. Las reglas CSS 2 alcanzaron el status de recomendación en el año 1998. En junio de 2011, el modulo de colores de CSS 3 Color ha sido publicado.

El objetivo de que W3C estandarice las CSS, HTML o cualquier otro lenguaje de Internet, es garantizar que el creador de un sitio web no tiene que hacer uno a propósito (ad hoc) para cada uno de los navegadores existentes (IEexplorer, Firefox, Chrome, Opera, etc.) y, además, que los usuarios no vean un sitio web con apariencia diferente dependiendo del navegador que empleen.

En las siguientes secciones se muestran los detalles de CSS lo que permitirá, que al final del capítulo, el lector sepa generar CSS para un sitio web y entender CSS de sitios web ya creados o creadas éstas con herramientas que automatizan el proceso.

2.2 SELECTORES: ESTILOS EN LÍNEA BASADOS EN ETIQUETAS, EN CLASES Y EN IDENTIFICADORES

Una hoja de estilo CSS está formada por *reglas* que indican la manera en la que se visualizará la página (reglas de estilo). Cada regla está compuesta de *selectores* los cuales indican a qué elemento o parte de una página se aplica un determinado estilo.

2.2.1 SELECTORES BASADOS EN ETIQUETAS

Para dar los primeros pasos en la definición de reglas de estilo, la manera más sencilla es usar las propias etiquetas HTML como selectores. Esto consiste en asociar a cada etiqueta una *declaración* del estilo que se le aplica al selector (etiqueta HTML). Las declaraciones tienen esta forma:

```
selector { atributo:valor }
```

El *atributo* hace referencia a la característica que se quiere modificar de la etiqueta, por ejemplo *color*. El valor hace referencia a la instancia del atributo, por ejemplo, *blue*.

Así, por ejemplo, *h1* podría ser un selector para aplicar un estilo a la etiqueta *<h1>*. Para indicar entonces el estilo de *<h1>* se pondría:

```
h1 {color:blue}
```

Los selectores se escriben omitiendo las llaves *<>*, es decir, simplemente *h1*, *h2*, etc. La declaración *{atributo:valor}* ha de ir encerrada en llaves *{ }* .

A cualquier etiqueta HTML se le puede asignar un estilo pero la descripción de las reglas debe ajustarse a la sintaxis definida anteriormente (según la especificación CSS). Si un navegador encuentra un selector cuya sintaxis no comprende, éste ignorará la declaración entera y continúa con la siguiente, con independencia de si el error afecta a la propiedad, al valor o a toda la descripción.

CSS contempla sintaxis para definir atributos para varios selectores y selectores con varios atributos.

Selector1, Selector2, {atributo1:valor1; atributo2:valor2}

Así, por ejemplo, si el mismo estilo se le quiere aplicar a dos selectores distintos, la sintaxis sería:

```
h1 , h2 {color:blue}
```

y si al mismo selector se le quiere aplicar atributos diferentes:

```
h1 {color: blue; background-color:red }
```

La propiedad *background-color* hace referencia al color de fondo del texto que por defecto es como el de la página.

Una vez conocida la sintaxis de las reglas, la siguiente pregunta sería ¿dónde se debe poner esas declaraciones para que el navegador las lea y las interprete? Una respuesta válida (en la sección 2.8 se muestran más) es en la cabecera `<head> </head>`. Todas las declaraciones basadas en etiquetas se incluyen en el `<head>` del fichero HTML entre etiquetas `<style></style>` (trataremos estas etiquetas y su utilización en las Secciones 2.8 y 2.9 de este mismo capítulo). Así, por ejemplo, las declaraciones del ejemplo anterior quedarían así:

```
<head>
<title></title>
<style>
  h1 {color: blue; background-color:red }
</style>
</head>
```

Cuando en navegador lee la cabecera del HTML interpreta que todas las etiquetas `<h1>` incluidas en el `<body>` tendrán color azul y color de fondo rojo.

ACTIVIDADES 2.1



- Cree un fichero HTML con dos textos, uno entre etiquetas `<h1></h1>` y otro entre `<h2></h2>`. Luego incluya en el `<head>` un estilo que afecte al color de las etiquetas `h1` (color) y al color de fondo (background-color) de los `h2`. De tal manera que quede como la Figura 2.1 (el texto entre `<h1>` es azul y el `<h2>` en negro con el *background* rojo):

Esto está entre etiquetas H1

Esto está entre etiquetas H2

Figura 2.1. Mi primer CSS

2.2.2 SELECTORES BASADOS EN CLASES

Los selectores basados en etiquetas tienen un uso muy claro, lo que las hace fáciles de entender. Sin embargo, desde el punto de vista de la flexibilidad y la reutilización, esta alternativa no es muy ventajosa.

Un ejemplo que muestra esa falta de flexibilidad es si se desea establecer diferentes estilos según el tipo de párrafo que se ponga, de manera que se distinga el encabezado de la página del resto del texto. Con lo visto en la sección anterior, si se hace una declaración del tipo `p {color:blue}` siempre que aparezca un texto entre etiquetas `<p></p>` éste se mostrará en color azul. Pero, si lo que se desea es que unos párrafos respondan a una regla de estilo y otros a otra esta alternativa no es válida. Es necesario usar *selectores basados en clases*.

Mediante las clases se pueden definir estilos abstractos, es decir, que no estén asociados directamente a una etiqueta HTML. Las clases permiten aplicar estilos a etiquetas HTML, con el mismo efecto que usando selectores de etiquetas, pero también a cualquier otro elemento de la página.

Hay diferentes tipos de clases: unas asociadas directamente a una etiqueta HTML (por ejemplo `h1.verde {color:green}`) y otras más genéricas que se pueden aplicar a cualquier etiqueta (por ejemplo, `.citas {color:grey}`).

Las clases asociadas a etiquetas HTML se definen con el

```
Nombreetiqueta.nombreclase {atributo:valor; atributo:valor; ... }
```

Esta alternativa es más potente que la vista con selectores basados en etiquetas, ya que permite aplicar a las mismas etiquetas diferentes estilos.

```
<head>
<style>
h1.roja {color: red}
h1.verde {color: green}
h1.azul {color: blue}
</style>
</head>
```

Para indicar en cada etiqueta `<h1>` qué estilo se le quiere aplicar se usa el atributo *class* de la siguiente manera:

```
<body>
<h1 class="roja"> Un encabezamiento rojo </h1>
<h1 class="azul"> ahora azul </h1>
<h1 class="verde"> Y ahora verde </h1>
</body>
```

El resultado quedaría como en la Figura 2.2: textos de tamaño grande pero de diferente color (el cambio de color se apreciará al ejecutar el código en el navegador).

Un encabezamiento rojo

ahora azul

Y ahora verde

Figura 2.2. Ejemplo clases basadas en etiquetas

Las clases más genéricas no se aplican a ninguna etiqueta HTML, por lo que en su descripción se emite en el selector el nombre de ninguna etiqueta.

```
.nombrecalse {atributo:valor; atributo:valor; ... }
```

Por ejemplo:

```
.verde {color:green;}
```

Una clase así definida puede aplicarse a cualquier elemento de la página. Del ejemplo siguiente, la primera declaración se aplica a todo el párrafo, la segunda a todo el encabezado `<h1>` y la tercera a todo el bloque `<div>`:

```
<p class="verde"> ...
<h1 class="verde"> ...
<div class="verde"> ...
```

Con la siguiente declaración:

```
<head>
<style>
.roja {color: red}
.verde {color: green}
.azul {color: blue}
</style>
</head>
```

Y aplicando las clases a las diferentes etiquetas `<h1>`:

```
<body>
<h1 class="roja">Un encabezamiento rojo</h1>
<h1 class="azul">ahora azul </h1>
<h1 class="verde">Y ahora verde</h1>
</body>
```

se obtienen un resultado idéntico al de la Figura 2.2. Sin embargo, con estas soluciones, no solo se limita aplicar estas clases a las etiquetas `<h1>` sino que se pueden aplicar a cualquier otra, por ejemplo a una etiqueta `<p>`.

```
<p class="roja">Aquí está el párrafo en rojo </p>
```

Si esa línea HTML se incluye en el `<body>` del ejemplo anterior quedaría como muestra la Figura 2.3 (con la línea nueva insertada):

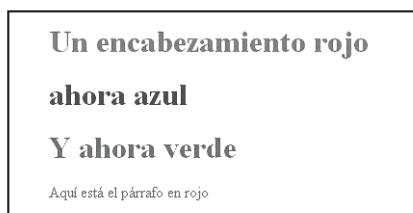


Figura 2.3. Ejemplo de clases abstractas

Una característica muy interesante del uso de clases es que se puede asignar más de una clase a una misma etiqueta, siempre y cuando no hay conflicto entre ellas. Por ejemplo, con la siguiente definición:

```
<head>
<style>
.textorojo { color: red }
.fondoazul { background-color: blue }
</style>
</head>
```

Se podría hacer el siguiente contenido en el <body>:

```
<h3 class="textorojo fondoazul">titulo en rojo, fondo azul</h3>
<p class="textorojo">color en rojo; fondo: el que herede de la pagina.</p>
```

El resultado sería el de la Figura 2.4 (los colores usados los explica el propio texto):

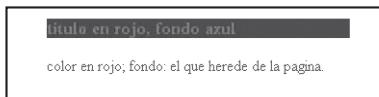


Figura 2.4. Ejemplo de uso de varias clases en la misma etiqueta

ACTIVIDADES 2.2



➤ Cree un fichero HTML con dos textos, uno entre etiquetas <h1></h1> y otro entre <p></p>. Luego incluya en el <head> estilos con clases abstractas que muestren: el texto en <h1> en verde con fondo azul y el texto de <p> en azul con fondo verde. Resuelva la actividad de dos maneras:

- La primera, usando una declaración de estilos con 4 etiquetas (2 para los colores y dos para los fondos).
- La segunda usando dos etiquetas: una que se puede llamar *.textoverdefondoazul* y otra que se puede llamar *.textoaazulfondoverde*

2.2.3 SELECTORES BASADOS EN IDENTIFICADORES

A modo de resumen de esta sección anterior, las clases definen propiedades que pueden ser compartidas por uno o varios elementos. Esto hace que una clase, por ejemplo *.roja* vista anteriormente, puede ser usada en un elemento <h1 class="roja"></h1> y en un <p class="roja"></p>. Es decir, *las clases se pueden utilizar en varios elementos*.

Concluido esto, en esta sección se explicarán los selectores basados en identificadores, los cuales tienen una función y sintaxis muy parecida, pero que se diferencian en algo muy útil: *los identificadores solo se pueden usar en un único elemento*.

Un selector basado en identificador se define dentro de las etiquetas `<style></style>` con la siguiente sintaxis:

```
Nombreetiqueta#nombreclase {atributo:valor; atributo:valor; ... }  
#nombreclase {atributo:valor; atributo:valor; ... }
```

Como puede apreciarse, esta declaración es idéntica a la usada para definir selectores de clase. La única diferencia es que en vez de usar un punto “.” se usa una almohadilla “#”. Sin embargo, el significado, la semántica de ambas expresiones es muy parecida.

Luego, para indicar en cada etiqueta `<h1>` qué estilo se le quiere aplicar se usa el atributo *id* en la etiqueta (de la misma manera que se usa *class* para las clases). Así, el siguiente ejemplo muestra una definición de estilos usando selectores de identificador.

```
<head>  
<style>  
#rojo { color:red; }  
</style>  
</head>  
  
<body>  
<p id="rojo"> el párrafo va en rojo </p>  
</body>
```

Este código el navegador lo interpreta mostrando en color rojo “el párrafo va en rojo”.

En el ejemplo anterior, si cambiamos *id* por *class* y “#” por “.” en vez de aplicar selectores de identificador aplicamos selectores de clase. Aparentemente no hay diferencia entre identificadores y clases. Sin embargo, sí que las hay, muy sutiles, pero importantes. La diferencia entre identificadores y clases es la misma que entre clases y objetos en un modelo orientado a objetos. A grandes rasgos las clases definen un patrón que deben cumplir todos los objetos de la clase. Cada objeto se identifica con un identificador único que lo diferencia de los demás objetos de esa clase.

En CSS las clases se pueden usar en uno o varios elementos. Por ejemplo, en una etiqueta `<h1>` y en una `<h2>` y en otra `<h1>` diferente y en una `<p>`. Sin embargo, y esta es la diferencia, los identificadores solo se deben usar en un único elemento. Esto es porque un identificador es como si hiciese referencia a un objeto de estilo y los objetos son únicos, por tanto solo un elemento puede “coger” ese objeto de estilo.

Un ejemplo que muestra la diferencia de uso es el siguiente: La clases son habituales usarlas de esta manera, que muestra cómo la clase *textorojo* se usa en dos elementos `<div>`:

```
<div class="textorojo">  
    El texto hereda las propiedades de una clase textorojo  
</div>  
  
<div class="textorojo">  
    Se vuelve a heredar las propiedades de la misma clase textorojo  
</div>  
  
<div class="textonegrrita">  
    El texto hereda las propiedades de las clases textonegrrita.  
</div>
```

Sin embargo, los identificadores se suelen usar en casos como el siguiente en donde los identificadores *cabecera*, *contenidos* y *piepagina* definen el estilo de esas tres partes de una página web. Los tres objetos `<div>` son asociados a tres identificadores diferentes ya que no tiene sentido que esos identificadores se repitan en varios elementos (no tiene sentido que una página tenga por ejemplo dos o más elementos `<div>` con un estilo tipo *cabecera* en una misma página web).

```
<div id="cabecera"></div>
<div id="contenido"></div>
<div id="piepagina"></div>
```

En definitiva, las clases se usan cuando el estilo se quiere aplicar a más de un elemento, mientras que los identificadores se definen cuando lo que se busca es “exclusividad” ya que solo será aplicada a un elemento. Una práctica habitual de los identificadores es usarlos para nombrar los bloques o secciones principales de un sitio web. El resto de estilos se hacen con clases.

Realmente, en muchos de los navegadores actuales, si se usa un identificador en varios elementos estos se interpretan y devuelven un resultado idéntico al que se devuelve si se hiciese con clases. Sin embargo, son “*buenas prácticas*” de *diseñador usar los identificadores y las clases es su momento*, con vistas a seguir un estándar de uso y también para que el diseñador se pueda beneficiar de las ventajas de los identificadores. Por ejemplo, CSS permite dar como valor un identificador al atributo *href* de la etiqueta `<a>` de la siguiente manera:

```
<style>
#cabecera
{
    background:#CCC;
    border:1px solid #093;
    margin:10px 12px 20px 15px;
}
</style>
</head>

<body>

<div id="cabecera"> Aquí está la cabecera </div>
...
...
<a href="#cabecera"> Ir a la cabecera </a>
</body>
```

En este otro ejemplo la etiqueta `<a>` define un enlace al elemento que usa el identificador “cabecera” que es un `<div>`. Obviamente, esto se puede hacer por la características deben tener los identificadores de usarse en un único elemento. Si se usan clases o se usa un identificador en varios elementos (no recomendable) el navegador no sabría a qué elemento definido con estilo “cabecera” saltar. Esa es solo una de las muchas ventajas que tiene usar identificadores según se recomienda en la especificación CSS.

ACTIVIDADES 2.3



- Cree un fichero HTML que defina un selector de clase y un selector de identificador. Pruebe a usar ID y CLASS en uno y varios elementos HTML y compruebe su efecto. ¿Qué pasa si no se cumple la especificación y se repite un identificador en varios elementos? ¿Cómo responde el navegador? Y en otro navegador, ¿responde igual?

2.3 AGRUPACIÓN Y ANIDAMIENTO DE SELECTORES

Los selectores se pueden agrupar y anidar para conseguir estilos CSS, por un lado más concretos y definidos, y por otro lado para tener un fichero CSS más optimizado y fácil de entender por el equipo de desarrollo.

2.3.1 AGRUPAMIENTOS

El término agrupamiento hace referencia a la manera en la que se pueden escribir las reglas de estilo (selectores) para conseguir un CSS más claro y fácil de entender. De esta manera, los errores son más fáciles de detectar y corregir. El uso de agrupamientos, como ocurría con el uso de los *id* y *class* en la sección anterior, responden a buenas prácticas en la especificación de CSS. Como se verá, algunas de las reglas de agrupamiento han sido ya comentadas en la sección 2.2.

Cualquier selector se puede agrupar siguiendo esta sintaxis:

Selector1, Selector2, {atributo1:valor1; atributo2:valor2;...}

De esta manera se puede aplicar el mismo estilo a un conjunto de selectores al mismo tiempo. Por ejemplo, si se quiere aplicar un tipo de fuente *Arial* a etiquetas *<h1>*, *<p>* y *<h3>*, la regla sería:

h1, p, h3 {Font-family:arial}

La versión menos optimizada de esa misma regla sería:

```
h1 {font-family: arial;}  
p {font-family: arial;}  
h3 {font-family: arial;}
```

De la misma manera se podría hacer para selectores de clase, pero teniendo en cuenta que si son abstractos es necesario primero poner un “.”. El siguiente ejemplo le asigna un color en hexadecimal (#0000FF) a tres clases: *mensaje*, *énfasis* y *subtítulo*.

.mensaje, .subtitulo, .enfasis {color:#0000FF}

Lo mismo se puede hacer para selectores de identificador, pero recordando que se usa la almohadilla. El siguiente ejemplo asigna a los identificadores un color verde (#00FF00):

```
#cabecera, #piepagina {color:#00FF00}
```

Como se ha comentado antes, las agrupaciones tienen como objetivo simplificar y dejar más claro un CSS. Por lo tanto, también se pueden combinar diferentes tipos de selectores para agrupar según estilos. Así, por ejemplo, la clase *.cabecera*, la etiqueta *h2* y el identificador *#micolor* comparten el mismo color en hexadecimal (#FF0000):

```
.cabecera, h2, #micolor {color:#FF0000}
```

2.3.2 ANIDAMIENTOS

Los selectores se pueden anidar con el fin de conseguir estilos más concretos y definidos. Este anidamiento es lo que se llama en CSS *selectores contextuales* que permiten aplicar un estilo a un elemento dependiendo de los elementos que tenga alrededor.

Selector anidado común

Se usa para crear reglas sobre elementos que están rodeados de otros elementos. La sintaxis general de este tipo de anidamiento para dos selectores es la siguiente:

```
SelectorX SelectorY {atributo1:valor1; atributo2:valor2;...}
```

Observar que entre ambos selectores hay un espacio en blanco.

Este tipo de anidamiento es útil cuando, por ejemplo, se desea ver en rojo un texto en negrita siempre y cuando esté incluido dentro de una etiqueta *<h1>* y en cursiva *<i>* (aunque entre medias haya otras etiquetas).

Si se definen los selectores de esta manera:

```
b {color:red}
```

El siguiente código HTML visualizará ambos textos en rojo, con independencia de la etiqueta *<h1>* y *<i>*:

```
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>
<b> Un texto en negrita y rojo </b>
</body>
```

Sin embargo, usando un anidamiento en la definición de la regla se consigue el efecto deseado:

```
<head>
<style>
h1 i b {color:red}
</style>
</head>
```

Las Figuras 2.5 y 2.6 muestran las salidas del HTML con los estilos anteriores (en la Figura 2.6 el anidamiento hace que “Un texto en negrita y rojo” aparezca en negro en el navegador).

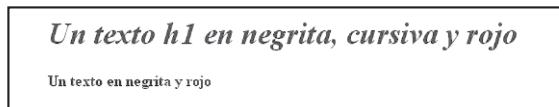


Figura 2.5. Ejemplo sin anidamiento

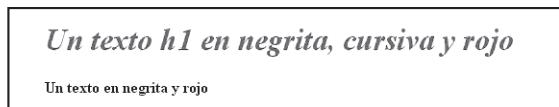


Figura 2.6. Ejemplo con anidamiento

En principio no hay límite en el número de anidamiento que se pueden hacer, sin embargo no es recomendable un anidamiento de más de 4 ó 5 por motivos de complejidad a la hora de interpretar el CSS, tanto por el navegador como por los diseñadores. Por otro lado, el anidamiento en los ejemplo de arriba se ha hecho con selectores basados en etiqueta, pero también pueden usarse con *class*, *id*, o combinaciones

Enlazando con los agrupamientos vistos anteriormente, este tipo de selectores se pueden agrupar. Por ejemplo, si suponemos que hemos definidos varias reglas con anidamiento *h1 i b* y *h1 b*, el agrupamiento sería así:

```
h1 i b, h1 b { color: red; },
```

que equivaldría a esto:

```
h1 i b {color:red}  
h1 b {color:red}
```

Anidamiento de selectores hijos

El anidamiento común explicado anteriormente se comporta igual si las etiquetas están consecutivas o si hay etiquetas intermedias. Por ejemplo, el siguiente anidamiento mostraría los dos textos en verde.

```
<head>  
<style>  
h1 b {color:red}  
</style>  
</head>  
<body>  
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>  
<h1><b> Un texto en negrita y rojo </b></h1>  
</body>
```

El motivo es que para el navegador, tanto `<h1><i>` como `<h1>` cumplen el anidamiento definido `h1 b {color:red}`.

Si lo que se desea es restringir que las etiquetas, además de estar en el mismo contexto, estén seguidas unas de otras, entonces la sintaxis que se tiene que usar en la definición es la siguiente (para anidamiento de dos selectores):

`SelectorX > SelectorY {atributo1:valor1; atributo2:valor2;...}`

De esta manera, el siguiente código mostrará en rojo solo el texto que tiene una etiqueta `` dentro de `<h1>` sin ninguna entre medias tal y como muestra la Figura 2.7 (solo el segundo texto aparecerá en rojo al probarlo en un navegador).

```
<head>
<style>
h1>b {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b> Un texto en h1,negrita y rojo </b></h1>
</body>
```

Un texto h1 en negrita, cursiva y negro
Un texto en h1,negrita y rojo

Figura 2.7. Anidamiento de un hijo

El siguiente código muestra un ejemplo que incluye a tres etiquetas, tal y como muestra la Figura 2.8 (solo el segundo texto aparecerá en rojo al probarlo en un navegador):

```
<head>
<style>
h1>b>i {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b><i><u> Texto en h1,negrita, cursiva, rojo y subrayado </u></i></b></h1>
</body>
```

Un texto h1 en negrita, cursiva y negro
Texto en h1,negrita, cursiva, rojo y subrayado

Figura 2.8. Anidamiento de dos hijos

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con *class*, *id*, o combinaciones.

Anidamiento de selectores adyacentes

Este tipo de anidamiento se usa cuando se quiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro elemento en el mismo nivel de anidamiento en HTML. La sintaxis es la siguiente (para dos selectores):

SelectorX + SelectorY {propiedad1:valor1; propiedad2:valor2;...}

El siguiente ejemplo muestra este estilo de anidamiento sobre dos etiquetas *<i>* y **. Solo se aplicará el estilo sobre ** si hay una etiqueta adyacente *<i>*. En este ejemplo en concreto, la palabra *advertencia* será la única que aparezca en rojo.

```
<head>
<style>
i+b {color:red}
</style>
</head>
<body>
<p> <i>Nota</i>, esto es una <b>advertencia</b> </p>
<b> Leer detenidamente </b>
</body>
```

El resultado se puede ver en la Figura 2.9 (la palabra “advertencia” aparece en rojo al probarlo en un navegador):

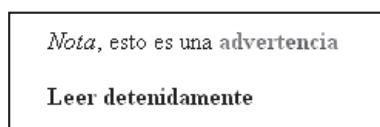


Figura 2.9. Anidamiento de adyacentes

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con *class*, *id*, o combinaciones.

ACTIVIDADES 2.4



► Interprete qué hace el siguiente código. ¿De qué color aparecerán los elementos de la lista? ¿De qué tamaño? Una vez interpretado compruebe el resultado en el navegador.

```
<head>
<style>
i+b {color:red}
.nivel1 {color:green}
.nivel2 {color:blue}
ul ul .nivel2 { font-size: x-small }
ul ul li {font-size: x-small; color:red}
</style>
</head>

<body>
<h1><p> <i>Título</i>: <b> Anidamiento y agrupamiento </b> de selectores </p></h1>
<ul>
<li class="nivel1"> Agrupamiento</li>
<li class="nivel1"> Anidamiento </li>
<ul>
<li class="nivel2"> Anidamiento de hijos</li>
<li class="nivel2"> Anidamiento de adyacentes </li>
<li> Anidamiento común</li>
</ul>
</ul>
</body>
```

2.4 BUENAS PRÁCTICAS AL ESCRIBIR CSS

Hasta el momento los ejemplos usados en este capítulo eran muy básicos, por tanto, no necesitaban de un número elevado de reglas. Sin embargo, en un trabajo real, las reglas CSS se multiplican para cualquier cosa que se quiera hacer de manera profesional en una web. Si a eso se le suma que cada selector puede tener del orden de 5 o más atributos, el CSS resultante puede ser ilegible si no se estructura con cuidado.

Como se comprobará en ejemplos posteriores, el gran número de reglas que puede tener un CSS necesita que el desarrollador haga un esfuerzo a la hora de escribir los selectores con el fin de que sea más fácil hacer modificaciones posteriores. Las siguientes recomendaciones no están definidas en el estándar W3C, son solo *buenas prácticas* que la mayoría de los desarrolladores de CSS suelen tener en cuenta. Seguir estas prácticas no solo ayuda a un desarrollo propio más claro, sino que también ayudan a entender mejor desarrollo de otros autores.

En la Actividad 2.4 se ha usado como selector de clase el siguiente: `:nivel1 {color:green}`, pero también se podría haber usado este otro: `::ff8 {color:green}`. Ambas declaraciones tienen algo en común: no son muy descriptivas de lo que quieren definir. A continuación, se muestra una batería de propuesta de buenas prácticas:

- **Los selectores se nombran en minúsculas, nunca empezando por caracteres especiales o numéricos:** como en cualquier otro lenguaje de etiquetas, hacerlo así da más claridad al selector (además de que la especificación CSS no lo permite para nombre de selectores de clase e identificadores). CSS en principio no distingue entre mayúsculas y minúsculas, salvo en los nombres de selectores de clase e identificadores, pero se eliminan errores si todo se pone en minúsculas. Por último, aunque en las últimas versiones se permite usar “_” en los nombres, es su lugar (como luego veremos) es preferible usar “-”, ya que no todos los navegadores soportan “_”.
- **El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva:** los ejemplos anteriores se podrían sustituir por `lista-nivel1{color:green}`, de esa manera se entiende mejor que lo que se pretender es dar un estilo para un elemento de una lista que está a `nivel1`.
- **El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición:** es mejor no usar nombre asociado a color, tamaño o posición porque hace que el selector soporte peor los cambios. Si a una regla se le llama `.rojo{color:rojo}`, entonces si por cualquier motivo (que los hay) se cambiar el color de la clase, también se debería cambiar el del selector, con los problemas que eso lleva al tener que actualizar todas las referencias a esa clase en el HTML.
- **Los nombres deben seguir más una visión semántica que estructural:** por el mismo criterio de facilitar los cambios, no se deben usar nombre de selectores según la localización si no se dan otras alternativas en el mismo CSS. Por ejemplo, si se usa un selector de clase menú-izquierda{...} para referirse a un menú situado a la izquierda, si por cualquier motivo se quiere cambiar a la derecha, entonces hay que cambiar también el nombre del selector y de las referencias HTML. Es mejor usar menu-navegacion {...} para definir este selector y no asociarlo a la izquierda. Otra cosa, sería si se desea hacer un menú a la izquierda y otro a la derecha, y que sea el usuario el que seleccione el que más le gusta (como ocurre en CMS como Joomla!).

Una alternativa semántica define los selectores según su función y no la estructura donde estará alojada. Por ejemplo, a un `<div>` es preferible asociarle una clase llamada `.main` que una llamada `.left-content` (contenidos de la izquierda) ya que si por cualquier motivo se cambia de lugar se tienen que cambiar sus propiedades y las referencias en el HTML.

Si en cualquier caso el desarrollador piensa que es necesario definir una clase única y exclusivamente para alinear una imagen o un párrafo y llamarla con esa acción, lo importante es documentarlo apropiadamente, para que él u otros desarrolladores no tengan problemas en el futuro. En CSS los comentarios se ponen como en Lenguaje C:

/*texto del comentario */

- **Separa las palabras mediante guiones o mayúsculas:** así se le da más claridad a los nombre de los selectores. Por ejemplo `menu-superior` en vez de `menú-navegacion` lo hace más legible.
- **No hacer uso excesivo de clases:** esto es útil ya que, a menudo, es más sencillo utilizar selectores contextuales o anidar selectores que no alteren el HTML, o incluso usar selectores de etiquetas para conseguir lo mismo. Los selectores de clase se suelen dejar para las partes más relevantes de la estructura.

Por ejemplo, en vez de usar:

```
<div class="main">
<div class="main-title">...</div>
<div class="main-paragraph">...</div>
</div>
```

Usar esta estructura ya que lo que se busca es el mismo efecto:

```
<div class="main">
<h1>...</h1>
<p>...</p>
</div>
```

- **Agrupar las reglas según su selector siempre que sea posible:** cuando hay varias reglas con el mismo selector (sea del tipo que sea) es interesante agruparlas unas debajo de otras. Por ejemplo:

```
[...]
table {border:double}
table.miembros {border:solid}
table.empleados {border:groove}
[...]
```

- **Al principio de un CSS es aconsejable definir los selectores de etiquetas:** además se usan comentarios para dejar claro cuál es la parte que define los selectores de etiquetas y cuál es la parte que definen las clases y otros elementos. Por ejemplo:

```
/* Etiquetas HTML */

html {font-family:arial, verdana, sans serif; font-size:13px;}
h1, h2, h3, h4, h5, h6, form, input, text-area{
border:0; padding:0; margin:0;
font-family:arial;}
h1{font-size:24px; color:#000000;}
h2{font-size:18px; color:#666666;

/* FIN Etiquetas HTML */
```

- **Estructurar visualmente los atributos:** si un elemento solo tiene tres atributos se pueden poner en la misma línea. Pero si hay más, se ponen en líneas diferentes sangrados con tabuladores. En el ejemplo anterior se puede ver este uso.

Estas son solo algunas propuestas de buenas prácticas. Sin embargo, conforme el diseñador profundiza en el desarrollo de CSS adquirirá muchas otras que darán más legibilidad a sus CSS.

2.5 ATRIBUTOS. MODELO DE CAJAS

Hasta el momento, todos los ejemplos anteriores estaban relacionados con la sintaxis de CSS. Por ello, en los ejemplos siempre se ha jugado con propiedades muy sencillas y fáciles de entender, como el color (*color*), la fuente (*font-family*) o el tamaño de la fuente (*font-size*). Estas tres propiedades (*color*, *font-family* y *font-size*) se llaman *atributos*.

La potencia de los CSS radica principalmente en la gran cantidad de atributos que se pueden usar para dar estilo a las páginas web. Estos van desde la posición que puede ocupar un determinado elemento hasta la colocación de una imagen de fondo.

En esta sección se explicarán aspectos básicos de localización de elementos en las páginas y el significado de los atributos que definen esas distancias.

Para entender los estilos CSS lo más adecuado es pensar en cualquier elemento (*<h1>*, *<p>*, *<div>*, etc.) como una caja. Para cada caja las dimensiones pueden ser controladas para producir una gran variedad de efectos.

Con esta definición, una página es una caja de cajas, y a esta simplificación se le llama *modelo de cajas*. La Figura 2.10 muestra los atributos de una caja. Este esquema es básico para conocer cómo afecta un determinado valor a un contenido. Es importante destacar, que en principio este esquema es seguido por todos los navegadores. Sin embargo, algunos navegadores como Internet Explorer lo interpretan de diferente manera. Por ejemplo, para Internet Explorer 6.0 el ancho del contenido va desde el *margin-left* hasta *margin-right*, sin tener en cuenta el ancho del borde ni del relleno (*padding*). En las siguientes explicaciones no haremos excepciones y se explicarán los atributos según indica el estándar W3C.

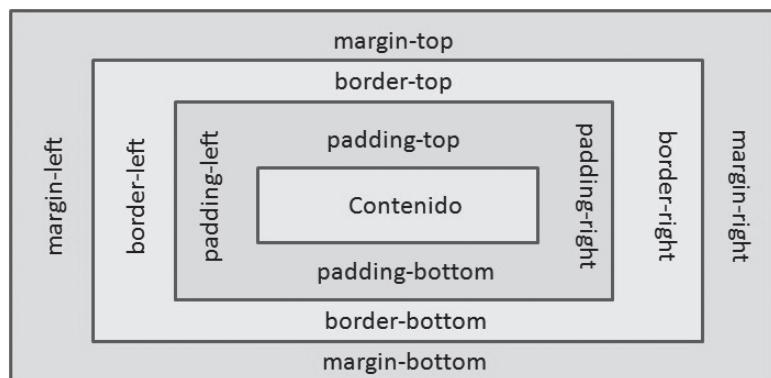


Figura 2.10. Atributos de una caja

Las cajas tienen varios atributos relacionados con los colores, imágenes, etc. Sin embargo esos atributos se verán en la siguiente sección.

Esta sección se describen los atributos que afectan a las dimensiones y posición de la caja:

- Atributos de posición de la caja.
- Los atributos de los márgenes, que asignan un borde externo a la caja.
- Los atributos de relleno (*padding*) asignan un espacio interno dentro de la caja para separar el contenido de los márgenes.
- Los atributos de los bordes, que definen las líneas gráficas alrededor de la caja.

2.5.1 UNIDADES DE MEDIDA

Los valores de estos atributos se pueden expresar en varias unidades absolutas: pulgadas, centímetros, milímetros, puntos, picas, y unidades relativas: em, ex y px.

- pulgadas (in). Una pulgada = 2,54 cm.
- centímetros (cm).
- milímetros (mm).
- puntos (pt). Un punto = 1/72 de pulgada.
- picas (pc). Una pica = 12 puntos.

El problema de las unidades absolutas es que siempre dependerán del entorno para el que fueron desarrolladas, y no siempre son portables a otros entornos. Por ejemplo, la medida punto depende de la resolución de la pantalla del usuario.

Por lo tanto, se recomienda el uso de unidades relativas. La menos aconsejable de estas son los píxeles ya que dependen de la resolución de pantalla y del tipo de ordenador: un sistema operativo Windows mantiene una equivalencia de 96px por pulgada y un Macintosh de 72 px por pulgada.

La unidad más aconsejable es em. Esta unidad es igual a la altura (*font-size*) de la letra del elemento en el que se usa. Por ejemplo, si para un párrafo especificamos una sangría de 2em el largo de la sangría será igual a dos veces el tamaño de la letra de ese párrafo. En el siguiente ejemplo la sangría es de 22 px (11px*2 del em).

```
p { font-size:11px; text-indent: 2em; }
```

Para el tamaño de letra. si el párrafo está contenido en un elemento <div>, el tamaño de la letra sería de 18px (un 20% mayor que el especificado para dicho div). Si no estuviera contenido en un <div>, un 20% mayor que el tamaño de letra del elemento del que descienda (por ejemplo, <body>).

```
div { font-size:15px; }  
p { font-size:1.2em; }
```

Por último están los porcentajes que son muy utilizados. Un valor de porcentaje se forma por un número y el signo %. No hay espacios en un valor de porcentaje. Los valores de porcentaje se fijan en relación a otro. Generalmente, el valor de porcentaje es relativo al tamaño de fuente del elemento:

ACTIVIDADES 2.5



- Compruebe en un navegador las diferencias de tamaño con los siguientes códigos. Observar que en este caso el estilo no está definido en el `<head>` del documento, sino que incorporado en la propia etiqueta (atributo `style`). Este método viene bien para poner ejemplos rápidos como en este caso, pero no para diseñar debido a su poca escalabilidad.

```
<body>
<div style="font-size:0.2in;">Texto de 0.2 pulgadas</div>
<div style="font-size:0.3in;">Texto de 0.3 pulgadas</div>
<div style="font-size:0.4in;">Texto de 0.4 pulgadas</div>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
<div style="font-size:10px;">Texto de 10 píxeles</div>
<div style="font-size:12px;">Texto de 12 píxeles</div>
<div style="font-size:14px;">Texto de 14 píxeles</div>
<div style="font-size:16px;">Texto de 16 píxeles</div>
<div style="font-size:18px;">Texto de 18 píxeles</div>
<div style="font-size:20px;">Texto de 20 píxeles</div>
<div style="font-size:8mm;">Texto de 8 milímetros</div>
<div style="font-size:9mm;">Texto de 9 milímetros</div>
<div style="font-size:10mm;">Texto de 10 milímetros</div>
<div style="font-size:11mm;">Texto de 11 milímetros</div>
<div style="font-size:1pc;">Texto de 1 picas</div>
<div style="font-size:2pc;">Texto de 2 picas</div>
<div style="font-size:3pc;">Texto de 3 picas</div>
<div style="font-size:4pc;">Texto de 4 picas</div>

</body>
```

2.5.2 ATRIBUTOS DE POSICIÓN

Los atributos de posición son principalmente `top`, `left`, `right` y `bottom`. `Top` (desde arriba) y `bottom` (desde abajo) indican la distancia en vertical donde se colocará la capa y `left` (desde la izquierda) y `right` (desde la derecha) la horizontal. Sin embargo, esta distancia está supeditada al atributo `position` que define el tipo de posición de la capa. Si el atributo `position` es `absolute` (o `fixed`), `top` indica la distancia del borde superior de la capa con respecto al borde superior de la página. Si el atributo `position` era `relative`, `top` indica la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa.

En la sección 2.7 se tratan más en profundidad el atributo `position` y sus posibilidades.

2.5.3 ATRIBUTOS MARGIN

Como se puede apreciar en la Figura 2.10, los atributos *margin-left*, *margin-right*, *margin-top*, *margin-bottom* marcan la separación entre otra caja y el borde de la que se representa. El siguiente ejemplo muestra estos atributos definidos en dos cajas: una para *<body>* y otra para *<div>*. El ejemplo se hace para las dos porque de esa manera se puede ver que las medidas se hacen relativas a la caja que la contiene, en este caso las medidas de los márgenes del *<div>* se hacen relativas al *<body>*. Para visualizar mejor el efecto se ha incluido un atributo *border* que, como veremos más adelante, dibuja el borde de 3 px en rojo para el *<div>* y azul para el *<body>*.

```
<head>
<style>
body {
    margin-top: 100px;
    margin-right:100px;
    margin-bottom: 100px;
    margin-left: 100px ;
    border : 3px dotted blue ;
}
div {
    margin-top: 15px;
    margin-right:15px;
    margin-bottom: 15px;
    margin-left: 15px;
    border : 3px dotted red ;
}
</style>

</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
```

El resultado de este código es mostrado en la Figura 2.11:



Texto de 0.2 pulgadas

Figura 2.11. Márgenes entre body y div

Al no haber relleno (*padding*) la distancia entre el borde del cuadro exterior (azul) y el más interno (rojo) es de unos 15 px. La distancia de 100 px es entre el borde exterior (azul) y los bordes de la vista del navegador. Si se incrementan los márgenes de 15 px a 150 px se puede observar que la distancia es mayor entre ambos bordes. También se puede poner una distancia de -15 px en vez de 15 px para conseguir que el borde interior (rojo) salga quede por fuera 15 px del exterior (azul). Sin embargo, no es muy aconsejable usar medidas negativas para las distancias.

Para simplificar también se puede usar un atributo *margin* que tiene 4 valores separados por espacios en blanco y cuyo orden es el siguiente (a favor de las agujas del reloj): el primer margen superior (*margin-top*), el derecho (*margin-right*), el inferior (*margin-bottom*) y el izquierdo (*margin-left*). Si se proporcionan solo dos o tres, los que faltan se asignan automáticamente según la relación con otras cajas. Para el ejemplo anterior, *body* se podía haber definido de esta manera:

```
body {  
    margin: 100px 100px 100px 100px ;  
    border : 3px dotted blue ;  
}
```

Los valores de estos atributos (y de cualquier de los siguientes que veremos) también pueden ser *auto* o *inherit*. Auto calcula automáticamente la mínima distancia según la relación con otros elementos. Esto es útil cuando las relaciones se hacen con medidas relativas. Por su lado, *inherit* hereda el valor del mismo atributo en la caja que lo contiene, es decir, hereda los valores del padre.

2.5.4 ATRIBUTOS PADDING

Padding se puede traducir como relleno. Como se puede apreciar en la Figura 2.10, estos atributos indican la distancia entre el borde y los elementos que se encuentran en el interior. Dicho de otro modo, tienen una función opuesta a la vista anteriormente para el atributo *margin*. Las medidas que se usan son las mismas que para *margin* y los nombres son muy parecidos: *padding-top* (relleno superior), *padding-right* (relleno derecho), *padding-bottom* (relleno inferior) y *padding-left* (relleno izquierdo).

Observar como en la Figura 2.11 la distancia entre la “T” de *Texto de 0.2 pulgadas* y el borde rojo es inexistente. Esto es debido a que el *padding-left* no ha sido definido, con lo que se coge el valor mínimo. El siguiente ejemplo asigna *padding-left* de 10 px para apreciar la diferencia.

```
<head>  
<style>  
body {  
    margin: 100px 100px 100px 100px ;  
    border : 3px dotted blue ;}  
div {  
    margin-top:15px;  
    margin-right:15px;  
    margin-bottom: 15px;  
    margin-left: 15px;  
    padding-left: 10px;  
    border : 3px dotted red ; }  
</style>
```

```
</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
```

El resultado se puede ver en la Figura 2.12 (el borde del cuadro más exterior es azul y el más interior es rojo como podrá verse al probarlo en un navegador):



Figura 2.12. div con padding-left de 15 px

Como ocurre con el atributo *margin*, también hay una atributo *padding* que simplifica el escribir el nombre de los 4 atributos. Un ejemplo de uso es el siguiente:

```
body {
  padding: 100px 100px 100px 100px ;
  border : 3px dotted blue ;
}
```

2.5.5 ATRIBUTOS BORDER-TOP, BORDER-BOTTOM, BORDER-RIGHT, BORDER-LEFT

Como se puede apreciar en la Figura 2.10, estos atributos definen el estilo y color del borde de la caja. Algunas de las palabras clave que tienen son: *none*, *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset* y *outset*:

En el ejemplo de la Figura 2.11 se ha usado un *atributo border: 3px dotted red* ; para indicar que todo el borde de la caja <div> sea rojo, punteado con puntos de 3 píxeles.

Se puede separar la asignación de color y estilo usando dos atributos separados: *border-color* y *border-style*. Estos atributos aplican respectivamente un color y un estilo a todos los bordes de la caja.

También se puede especificar un color y estilo para cada uno de los bordes (*top*, *left*, *right*, *bottom*) por separado con *border-top*, *border-bottom*, *border-right*, *border-left*.

Los atributos mostrados no son todos los que definen CSS (versión 3) para manejar bordes. Existen muchos más, como por ejemplo, *border-radius* que se usa para hacer bordes redondeados. Este atributo se utiliza en el siguiente ejemplo.

Por último, también se puede definir por separado el ancho de un borde con la propiedad *border-width* para todos los bordes de la caja o con *border-top-width*, *border-bottom-width*, *border-right-width*, *border-left-width* para cada uno por separado. Los anchos (*width*) pueden tener como valor un tamaño en cualquier unidad como los visto anteriormente, o valores predefinidos: *thin* (estrecho), *medium* (mediano) o *thick* (ancho).

El siguiente ejemplo muestra el uso de muchos de los atributos visto para hacer una caja dentro de otra con borde redondeado.

```
<head>
<style>
body {
    margin: 100px 100px 100px 100px ;
    border-style:inset;
    border-color:blue; /* color del borde azul */
    border-radius: 15px; /*borde redondeado de radio 15px */
    border-width:thick; /* Un borde grueso */
    padding: 15px 15px 15px 15px;
}

div {
    margin-top:15px;
    margin-right:15px;
    margin-bottom: 15px;
    margin-left: 15px;
    padding-left: 10px;
    border-top : 3px dotted red ; /*estilo, tamaño y color incluidos en el mismo
atributo */
    border-right : 2px solid blue ;
    border-bottom : 3px double green ;
    border-left : 3px groove red ;
}
</style>
```

```
</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
```

El resultado se puede ver en la Figura 2.13 mostrada en Chrome 14.0 (el cuadro más exterior es de color azul y el interior tiene el lado derecho azul, el inferior verde y los otros dos lado rojos, como se puede ver al probarlo en un navegador).



Figura 2.13. Aplicación del atributo Border

2.5.6 ATRIBUTOS DEL CONTENIDO

También se puede concretar el ancho y alto de un contenido (ver Figura 2.10) con el atributo *width* y *height*. *Width* establece la distancia entre el límite del *padding-left* y el *padding-right* (así es como viene definido en el CSS, aunque navegadores como Internet Explorer 6 no los considera así). *Height* igual pero entre el *padding-bottom* y el *padding-top*.

En esta a sección se ha pretendido hacer una introducción a los atributos más importantes que participan en el modelo de cajas, con la idea de que el desarrollador se forme un mapa mental de cómo pueden estar distribuidos los elementos en una web. Sin embargo, no se han incluido todas las posibilidades que ofrece CSS en este tipo de atributos. Se recomienda consultar las siguientes URL para conocer más sobre el tema.

Todos los atributos disponibles en CSS 2.1 de la W3C:

<http://www.w3c.es/divulgacion/guiasreferencia/css21/#modeloCajas>

Esta otra URL muestra las nuevas incorporaciones de CSS3, en la que se puede ver *border-radius*.

<http://www.css3.info/preview/>

ACTIVIDADES 2.6



El siguiente código muestra una estructura básica de página web con un encabezado una parte para el menú y otra parte para los contenidos. Este es el esqueleto básico de una web para una Restaurante. Como se puede observar, esta estructura está hecha con clases e identificador con el fin de dar mayor flexibilidad.

```
<head>
<style>

div#body /*Define una identificador "body" asociado a un elemento <div>. Es la
posición del cuerpo de la página (barra lateral más contenido */
{
margin:auto auto auto auto; /* top right bottom left: equivale a poner un solo
auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la
posición de la cabecera */
{
margin:10px auto 10px auto;
width:710px;
border:dotted 1px #ccc;
}
div#sidebar
```

```
{  
display:none;  
border:solid 1px #CCC;  
}  
body.main-sidebar div#main, body.sidebar-main div#main  
{  
width:490px;  
}  
body.main-sidebar div#sidebar, body.sidebar-main div#sidebar  
{  
display:block;  
width:200px;  
border:solid 1px #CCC;  
}  
body.main-sidebar div#main, body.main-sidebar div#sidebar /*ms hace referencia a  
main-sidebar. Este estilo coloca el sidebar a la derecha y el main a la izquierda */  
{  
float:left;  
border:solid 1px #CCC;  
}  
body.sidebar-main div#main, body.sidebar-main div#sidebar /*sm hace referencia a  
sidebar-main. Este estilo coloca el sidebar a la izquierda y el main a la derecha*/  
{  
float:right;  
border:solid 1px #CCC;  
}  
  
</style>  
</head>  
<body class="main-sidebar">  
<div id="header">Cabecera. RESTAURANTE: El muslito</div>  
<div id="body">  
<div id="main">Este restaurante es una de las referencias gastronómicas del esta  
zona. [...] </div>  
<div id="sidebar">Menú principal</div>  
</div>  
</body>
```

- Analice el código. Se han definido dos clases asociadas al *body*: *body.main-sidebar* y *body.sidebar-main*. Dependiendo de qué clase se use en la etiqueta *<body>* el *div* asociado a la barra lateral (*sidebar*) se pondrá a la derecha o izquierda. Por lo tanto, se puede apreciar cómo de fácil es cambiar una distribución con solo cambiar *<body class="main-sidebar">* por *<body class="sidebar-main">*.

El resultado aplicado *<body class="main-sidebar">* puede verse en la Figura 2.14.

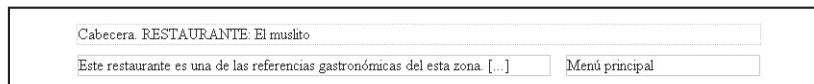


Figura 2.14. Esqueleto de una web para una Restaurante

► Analice el atributo *float*. El valor de *float:left* indica que la caja con este estilo se colocan a la izquierda y el resto de cajas que se crean posteriores en el código HTML, y que están en la misma posición, se colocan a la derecha de ésta. Esto es lo que le da el efecto de una caja *main* a la izquierda y el *sidebar* apoyada a su derecha. Cuando es *float:right* lo que se indica es que esa caja se coloca a la derecha. Esto da el efecto de una caja *main* a la derecha y el *sidebar* apoyada a su izquierda.

El efecto de un *float* sigue hasta que se usa en otra caja un estilo *clear*.

El problema surge si posteriormente se desea colocar una caja que ya no se apoye a la derecha (o izquierda) de la flotante sino que se coloque debajo como se hace por defecto (sin atributo *position*). En este caso, esa nueva caja se tiene que crear un el estilo *clear*. Si se pone *clear:left* se indica que ya no se desea colocar esa caja adyacente a una definida con *float:left*. Si pone *clear:right*, se indica que ya no se quiere colocar esa caja adyacente a una definida con *float:right*. El estilo *clear* elimina el efecto del *float*.

► Modifique este código para conseguir los siguientes efectos:

- Que el borde de la cabecera sea en azul, sólido, redondeado con un ancho de 3 px.
- Que el borde del menú principal sea de 1 px, en verde y punteado (*dotted*) y con la palabra "Menú Principal" separada 15 px del borde de la caja (solo el borde izquierdo).
- ¿Qué ocurre si el borde es de tamaño 5 px?
- Define un pie de página igual de ancho y alto que el encabezado pero de color de borde verde (PISTA: Usar en ese nueva caja un estilo con *clear*).
- ¿Qué ocurre si no se usa un estilo *clear* en el pie de página?

El resultado debe ser algo similar a la Figura 2.15 usando el navegador Chrome 14.0 (el cuadro de borde redondeado en el encabezado es azul y el del pie de página es verde). Internet Explorer 8.0 no muestra el efecto curvo, sino que usa cajas con ángulos rectos (es decir, no interpreta *border-radius*):

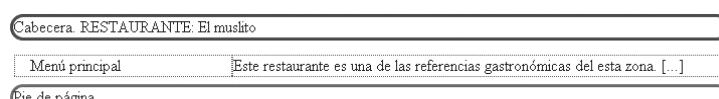


Figura 2.15. Esqueleto con pie de página no flotante

Solución

El siguiente código muestra la solución a la actividad.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<style>
```

```
div#body /*Define una identificador "body" asociado a un elemento <div>. Es la
posición del cuerpo de la página (barra lateral más contenido) */
{
margin:auto auto auto auto; /* top right bottom left: equivale a poner un solo
auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la
posición de la cabecera */
{
margin:10px auto 10px auto;
width:710px;
border:solid 3px blue;
border-radius: 15px; /*borde redondeado de radio 15px */
}
div#sidebar
{
display:none;
border:solid 1px #CFC;
padding-left:15px; /*Se modifica el ancho*/
}
body.sidebar-main div#footer /*Define una identificador "footer" asociado a un
elemento <div>. Es la posición del pie de página */
{
margin:25px auto 25px auto;
width:710px;
border:solid 3px green;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:right; /* Deja sin utilidad la opción de float:right */
}

body.main-sidebar div#footer /*Define una identificador "footer" asociado a un
elemento <div>. Es la posición del pie de página */
{
margin:15px auto 15px auto;

width:710px;
border:solid 3px blue;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:left; /* Deja sin utilidad la opción de float:left */
}
body.main-sidebar div#main, body.sidebar-main div#main
{
width:490px;
```

```
border:solid 4px green;

}

body.main-sidebar div#body, body.sidebar-main div#body
{
margin:auto auto auto auto;
width:710px;
border:solid 4px #FFF;
}

body.main-sidebar div#sidebar, body.sidebar-main div#sidebar
{
display:block;
width:200px;
border:solid 1px #CCC;
}

body.main-sidebar div#main, body.main-sidebar div#sidebar /*Main-sidebar. Este
estilo coloca el sidebar a la derecha y el main a la izquierda */
{
float:left;
border:dotted 1px green;
}

body.sidebar-main div#main, body.sidebar-main div#sidebar /*Sidebar-main. Este
estilo coloca el sidebar a la izquierda y el main a la derecha*/
{
float:right;
border:dotted 1px green;
}


```

</style>

</head>

<body class="sidebar-main">

<div id="header">Cabecera. RESTAURANTE: El muslito</div>

<div id="body">

<div id="main">Este restaurante es una de las referencias gastronómicas del esta
zona. [...] </div>

<div id="sidebar">Menú principal </div>

</div>

<div id="footer">Pie de página</div>

</body>

</html>

2.6

ELEMENTOS: COLORES DE FONDO, TEXTOS, ENLACES, LISTAS, TABLAS, VISIBILIDAD, IMÁGENES

Una vez vistos los atributos asociados a las clases. En esta sección se muestran atributos adicionales de carácter más general y relacionados con la apariencia de textos, listas, tablas, enlaces e imágenes.

Atributos de fuentes

- **Color:** *RGB o nombre de color*. Sirve para indicar el color del texto. Lo admiten casi todas las etiquetas de HTML. Admite nombres de colores en inglés para algunos colores y valores RGB para todos.
- **font-size:** *unidades | xx-small | x-small | small | medium | large | x-large | xx-large*. Sirve para determinar el tamaño de una fuente.
- **font-family:** *serif | sans-serif | cursive | fantasy | monospace*. Con este atributo indicamos la familia de tipografía del texto. Los primeros valores son genéricos, es decir, los navegadores las comprenden y utilizan las fuentes que el usuario tenga en su sistema. También se pueden definir con tipografías normales, como ocurría en HTML. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.
- **font-weight:** *normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900*. Sirve para definir la anchura de los caracteres, es decir, efecto de negrita. Normal y 400 son el mismo valor, así como *bold* y 700.
- **font-style:** *normal | italic | oblique*. Es el estilo de la fuente. El estilo *oblique* es similar al *italic*.

Atributos de párrafos

- **line-height:** *normal | unidades*. El alto de una línea, y por tanto, el espacio entre líneas. Es una de esas características que no se pueden modificar con HTML.
- **text-decoration:** *none | underline | overline | line-through*. Para establecer la decoración de un texto, es decir, si está subrayado, sobre-rayado o tachado.
- **text-align:** *left | right | center | justify*. Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque no funciona en todos los navegadores.
- **text-indent:** *Unidades*. Un atributo que sirve para hacer sangrado o márgenes en las páginas.
- **text-transform:** *capitalize | uppercase | lowercase | none* *text-transform | none*. Permite transformar el texto haciendo que tenga la primera letra en mayúsculas de todas las palabras, todo en mayúsculas o minúsculas.

Atributos de fondo

- **Background-color:** *RGB o nombre de color*. Sirve para indicar el color de fondo de un elemento de la página.
- **Background-image:** nombre de la imagen con su camino relativo o absoluto.

Atributos tablas

- **caption-side:** *valores top | bottom.* Posición del título.
- **table-layout:** *auto |fixed.* Control del algoritmo usado para el formato de las celdas, filas y columnas.
- **border-collapse:** *collapse | separate.* Selección del modelo de los bordes.
- **border-spacing:** *unidades.* Espaciado entre los bordes de celdas adyacentes.
- **empty-cells:** *show | hide.* Visibilidad de los bordes de celdas sin contenido (ocultar o mostrar).

Atributos visibilidad

- **Overflow:** *visible |hidden |scroll |auto.* Comportamiento del contenido si se desborda en la caja
- **Clip:** *rect (top,right,bottom, left) | auto.* Especifica la región visible del elemento mediante las dimensiones de un rectángulo que hace de ventana de visualización.
- **Visibility:** *visible |hidden |collapse.* Visibilidad de las cajas. No se reorganizan las cajas de alrededor, solo se oculta.
- **Display:** *inline | block | none | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | inherit.* Muestra una caja con diferentes estilos. El más común es *none*, que se diferencia de *visibility:hidden* en que en este caso las cajas de alrededor se reorganizan cuando se oculta.

Atributos de listas

- **list-style-type:** *disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none.* Estilo aplicable a los marcadores visuales de las listas.
- **list-style-image:** *url("http://...") | none.* Imagen aplicable a los elementos de las listas.
- **list-style-position:** *inside | outside.* Posición dentro de la lista de los elementos marcadores de las listas.

Atributos de enlaces

Los enlaces en CSS se pueden manejar con bastante libertad comparado con HTML. CSS permite definir estilos en los enlaces, quitando el subrayado o hacer enlaces en la misma página con distintos colores. Para aplicar estilo a los enlaces debemos definirlos para los distintos tipos de enlaces, que son:

- **Enlaces normales:** *A:link {atributos}*
- **Enlaces visitados:** *A:visited {atributos}*
- **Enlaces activos:** *A:active {atributos}* (Los enlaces están activos en el preciso momento en que se pulsa sobre ellos).
- **Enlaces hover:** *A:hover {atributos}* (Cuando el ratón está encima de ellos).

El atributo para definir enlaces sin subrayado es *text-decoration:none*, y para darles color es con el conocido atributo *color*.

El siguiente ejemplo muestra una definición personalizada de un menú de enlaces basados en los atributos vistos. Para los enlaces visitados se quita el subrayado y se pone en color. Si el ratón se coloca encima de un enlace aparece subrayado (*underline*), si se seleccionan todos pasan a color gris rodeados por un marco.

```
<html>
<head>
<style>
a.menus:link
{text-decoration:none;
color: #000000;
border:#FFFFFF 1px solid;
} /* Link no visitado*/



a.menus:visited {
text-decoration:none;
color:#cccccc;
} /*Link visitado*/



a.menus:active {
text-decoration:none;
color: #003399;
background: #green;
border:#FFFFFF 1px solid;
} /*Link activo*/



a.menus:hover {
text-decoration:underline;
color: #003399;
background: #red;
border:#FFFFFF 1px solid;}
/*Ratón sobre el link*/



</style>
</head>
<body>
<a href="#" class="menus">Enlace uno</a>
<a href="#" class="menus">Enlace dos </a>
<a href="#" class="menus">Enlace tres</a>
</body>
</html>
```

La Figura 2.16 muestra los enlaces ya visitados y uno de ellos con el ratón encima (*hover*). El color de fondo del último enlace es rojo.



Figura 2.16. Ejemplo de menú

Aunque se han mostrado en esta sección los atributos más usados, faltan muchos por conocer. Se recomienda consultar las siguientes URL para conocer más sobre el tema. Todos los atributos disponibles en CSS 2.1 de la W3C pueden consultarse en <http://www.w3c.es/divulgacion/guiasreferencia/css21> y en esta otra URL se presentan las nuevas incorporaciones de CSS3 <http://www.css3.info/preview/>.

ACTIVIDADES 2.7



► Modifique el código del esqueleto mostrado en la Actividad 2.6 para incluir los siguientes elementos:

- Dividir el encabezado *header* en dos partes. Colocar un logo de Superman en una de ellas y un texto `<h1>` a lado del logo.
- Colocar una imagen Superman en el contenidos *main*.
- Hacer un menú con 4 enlaces en el *sidebar* que no lleve a ningún enlace pero que si tenga atributos para sus enlaces: activo, *hover*, etc.

El resultado debe quedar algo como muestra la Figura 2.17 (usando el navegador Chrome 14.0).

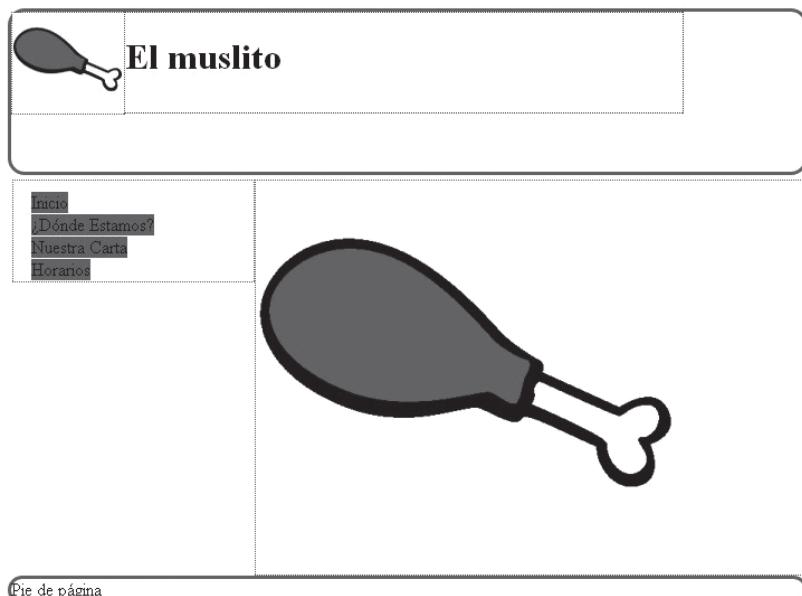


Figura 2.17. Esqueleto mejorado con menú e imágenes

Solución: (el siguiente código muestra la solución a la actividad)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<style>

div#body /*Define una identificador "body" asociado a un elemento <div>. Es la posición del cuerpo de la página (barra lateral más contenido) */
{
margin:auto auto auto auto; /* top right bottom left: equivale a poner un solo auto */
width:710px; /* El ancho del estilo lo coloca en 710px */
clear:left; /*En el body quito el flotante para que la siguiente caja quede debajo */
}
div#header /*Define una identificador "header" asociado a un elemento <div>. Es la posición de la cabecera */
{
margin:auto auto auto auto;
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
}
div#sidebar
{
display:none; /*este div no lo hace visible */
border:solid 1px #CFC;
padding-left:15px; /*Se modifica el ancho*/
}
body.sidebar-main div#footer /*Define una identificador "footer" asociado a un elemento <div>. Es la posición del pie de página */
{
margin:25px auto 25px auto;
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:right; /* Ya que el div encima es tipo float:right, es necesario decirle ahora que coloque este div cuando no hay ningún div que este flotando a la derecha */
}

body.main-sidebar div#footer /*Define una identificador "footer" asociado a un elemento <div>. Es la posición del pie de página */
{
margin:15px auto 15px auto;
```

```
width:710px;
border:solid 3px #815608;
border-radius: 15px; /*borde redondeado de radio 15px */
clear:left; /* Ya que el div encima es tipo float:left, es necesario decirle ahora
que coloque este div cuando no hay ningún div que este flotando a la izquierda*/
}

body.main-sidebar div#main, body.sidebar-main div#main
{
width:490px;
border:solid 4px green;

}

body.main-sidebar div#body, body.sidebar-main div#body
{
margin:auto auto auto auto;
width:710px;
border:solid 4px #FFF;
}

body.main-sidebar div#sidebar, body.sidebar-main div#sidebar
{
padding-top:10px;
display:block;
width:200px;
border:solid 1px #CCC;
}

body.main-sidebar div#main, body.main-sidebar div#sidebar /*Main-sidebar. Este
estilo coloca el sidebar a la derecha y el main a la izquierda */
{
float:left;
border:dotted 1px green;
}

body.sidebar-main div#main, body.sidebar-main div#sidebar /*Sidebar-main. Este
estilo coloca el sidebar a la izquierda y el main a la derecha*/
{
float:right;
border:dotted 1px green;
}

div#header div#logo /*Se define el logo flotante a la izquierda*/
{
margin-bottom:inherit;
border:dotted 1px green;
float:left; /*Hago estas caja flotante, y las siguientes cajas que estén en la misma
```

```
posición también serán flotantes a la izquierda. No hace falta ponerle a div#header  
div#main-title también esta propiedad*/  
}  
div#header div#main-title /*Es el título del encabezado. Al ser el logo flotante  
este también lo será */  
{  
width:600px;  
margin-bottom:inherit;  
border:dotted 1px blue;  
padding-bottom:10px;  
  
/* DEFINE LOS ENLACES DE MEnU*/  
}  
a.menus:link  
{text-decoration:none;  
color:#000000;  
background-color:#815608;  
border:#FFFFFF 1px solid;  
} /* Link no visitado*/  
  
a.menus:visited {  
text-decoration:none;  
color:#cccccc;  
} /*Link visitado*/  
  
a.menus:active {  
text-decoration:none;  
color: #003399;  
background: green;  
border:#FFFFFF 1px solid;  
} /*Link activo*/  
  
a.menus:hover {  
text-decoration:underline;  
color: #003399;  
background: red;  
border:#FFFFFF 1px solid;}  
/*Ratón sobre el link*/  
  
</style>  
</head>  
  
<body class="sidebar-main">  
<div id="header">
```

```
<div id="logo"> </img> </div>
<div id="main-title"><h1> El muslito </h1></div>
<p>&nbsp;</p>
</div>

<div id="body">
<div id="main"> </div>
<div id="sidebar">
<div><a href="#" class="menus">Inicio</a></div>
<div><a href="#" class="menus">¿Dónde Estamos?</a></div>
<div><a href="#" class="menus">Nuestra Carta</a></div>
<div><a href="#" class="menus">Horarios</a></div>

</div>
</div>
<div id="footer">Pie de página</div>
</body>
</html>
```

2.7 SUPERPOSICIÓN Y PRECEDENCIA DE ESTILOS

Para terminar con el uso de CSS, en esta sección se tratarán dos importantes aspectos: la *superposición de cajas*, que está relacionada con lo visto en la sección 2.5 y la *precedencia de estilos*. Controlar estos aspectos supone entender bien el funcionamiento del CSS y conseguir resultados muy precisos y personales.

2.7.1 SUPERPOSICIÓN DE CAJAS

En el apartado 2.5 se mostró todo lo relacionado con el posicionamiento de las cajas, tanto vertical como horizontalmente. Sin embargo, además de estas dos dimensiones, CSS permite controlar la profundidad de las cajas determinando el orden de superposición de éstas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas, haciendo efectos de solapamiento.

El atributo *z-index* permite definir el nivel de profundidad de una caja. Su valor es un número entero. En principio el estándar W3C permite números negativos, pero generalmente, el valor 0 suele tomarse como en nivel más bajo. Cuanto más alto sea el valor, más *cerca* se mostrará la capa al usuario en la web, es decir, una caja con *z-index=10* se mostrará por encima de una con *z-index=9*.

El atributo *z-index* solo tiene efecto si va acompañado de otro muy importante para determinar la posición de una caja respecto a las otras. Este atributo se llama *position* y su funcionalidad está muy ligada con lo visto en la sección 2.5.

El atributo position puede tener como valores: *static*, *absolute*, *relative*, *fixed* o *inherit*. A continuación se describen cada uno de ellos:

- **Static:** es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, *static* no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos *top*, *left*, *right* y *bottom* no se tendrán en cuenta.
- **Absolute:** el valor *absolute* permite posicionar cajas de manera absoluta, esto es de manera definida por valores de los atributos *top*, *left*, *bottom* y *right*. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML no viéndose afectadas por el lugar en donde aparezcan dentro del HTML y tampoco afecta estas cajas a otras del flujo normal del HTML.

Es importante destacar que los valores *top*, *left*, *bottom* y *right* son una distancia con respecto al primer elemento contenedor que tenga un valor de position distinto de *static*. Si todos los contenedores donde esté la capa posicionada con estilo *absolute* (todos sus padres hasta llegar a <body>) son *static*, simplemente se posiciona con respecto al lado superior de la página, para el caso de *top*, el inferior para *bottom*, del lado izquierdo para *left* o el derecho, en el caso de utilizar *right*.

El siguiente código muestra el resultado de la Figura 2.18. Una modificación para comprobar el efecto de *static* podría ser la siguiente: poner las tres capas del ejemplo a *static*, eso colocaría una debajo de las otras según han sido definidas en el HTML, sin tener en cuenta los valores *top*, *right*, *bottom* y *left*.

```
<body>
<div style="position: absolute; width: 300px; height: 140px; top: 100px; left: 30px;
background-color: #ff8800; color: #fff; padding: 15px;z-index: 2;">
Esta capa tiene posicionamiento absoluto. Permite especificar top y left para colocarla
con respecto a la esquina superior izquierda.
</div>

<div style="position: static ; width: 820px; height: 30px; padding: 10px; background-
color: #ddf; top: 150px; left: 10px; z-index: 1;">Posicionamiento static. Las otras dos
capas absolutas se posicionan después de la static que se coloca arriba.</div>

<div style="position: absolute; width: 100px; height: 20px; padding: 10px; background-
color: #ddf; bottom: 10px; right: 10px;">Posicionamiento absoluto con atributos bottom
y right</div>
</body>
```

Posicionamiento static. Las otras dos capas absolutas se posicionan después de la static que se coloca arriba.

Esta capa tiene posicionamiento absoluto.
Permite especificar top y left para colocarla con
respecto a la esquina superior izquierda.

Posicionamiento
absoluto con
atributos bottom
y right

Figura 2.18. Cajas static y absolute

- **Relative:** indica que la capa sí forma parte del flujo normal de elementos de la página, por lo que su posición dependerá del lugar donde esté en el código y el flujo HTML. Además, las capas con posicionamiento *relative*, admiten los valores *top* y *left* para definir la distancia a la que se colocan con respecto al punto donde esté en ese momento el flujo normal del HTML. Como afectan al mencionado flujo del HTML, los elementos colocados después de las capas estilo *relative*, tendrán en cuenta sus dimensiones para continuar el flujo y saber dónde colocarse.

La Figura 2.19 muestra el resultado del siguiente ejemplo con diferencias entre *static* y *relative*.

```
<body>

<h1>NO define posición</h1>

<div style="background-color: #606; color:#ffc; padding:10px; text-align: center; width: 300px;">No define posicion, por lo que es static</div>

<div style="position: relative; width: 300px; padding: 10px; background-color: #066; color:#ffc; top:100px; left: 30px;">Capa de posicionamiento relative<br>Se tiene en cuenta esta capa para posicionar las siguientes.</div>

<h2>La última en ponerse</h2>

</body>
```

NO define posición

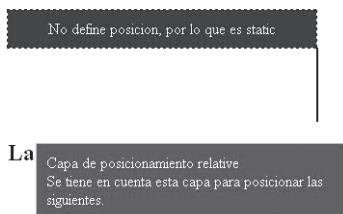


Figura 2.19. Cajas static y relative

Las etiquetas <h1> y <h2> respetan el flujo HTML y el <div> que no establece posición (por defecto es *static*), por tanto, también es afectada por el flujo. Hay una capa estilo *relative*, en el segundo elemento <div>, que también se posiciona con respecto al flujo normal. Como tiene un *top* y *left*, aparece un poco desplazada del lugar que le tocaría con respecto al flujo. El último <h2> que aparece se coloca teniendo en cuenta al flujo y tiene en cuenta la capa *relative*, por eso deja un espacio en blanco arriba, pero no atiende a la posición real de ésta, que se marcó con los atributos *top* y *left*.

- **Fixed:** este atributo sirve para posicionar una capa con posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición. El lugar donde se “anclará” la capa siempre es relativo al cuerpo (el espacio disponible del navegador para la página). Si utilizamos *top* y *left*, estaremos marcando su posición con respecto a la esquina superior izquierda y si utilizamos *bottom* y *right* su posición será relativa a la esquina inferior derecha.

Las capas *fixed* son útiles para hacer zonas en la página web que no se muevan ni bajando la barra de desplazamiento.

- **Inherit:** indica que el valor de *position* tiene que heredarse del elemento padre. Este valor no es soportado por muchos navegadores por lo que no se utiliza mucho en la actualidad.

ACTIVIDADES 2.8



- Modifique el código siguiente para colocar la capa delante del fondo.

```
<body>

<h1>NO define posición</h1>
<div style="z-index:9; position: absolute; background-color: #606; color:#ffc;
padding:10px; text-align: center; width: 300px;">No define posicion, por lo que es
static</div>
<div style="z-index:17; position: relative; width: 300px; padding: 10px; background-
color: #066; color:#ffc; left: 30px;">Capa de posicionamiento relative<br>Se tiene
en cuenta esta capa para posicionar las siguientes.</div>
<h2>La última en ponerse</h2>

</body>
```

ACTIVIDADES 2.9



► Analice el siguiente código e interprete cómo quedarán las capas. Haga un bosquejo de la pantalla resultado.

```
<body>  
<h1>NO define posición</h1>  
  
<div style="background-color: #606; color:#ffc; padding:10px; text-align: center; width: 300px;">No define posición, por lo que es static</div>  
  
<div style="position: absolute; width: 300px; padding: 10px; background-color: #066; text-align:right; color:#ffc; top:100px; left: 300px;">Capa Absoluta (solo tiene como contenedor a "body") por lo tanto se relativa a body. </div>  
  
<div style="position: relative; width: 300px; padding: 10px; background-color: #033; color:#ffc; top:0px; left: 30px;">Capa de posicionamiento relative. Se coloca según la posición que le tocaría en el flujo HTML.</div>  
  
<div style="position: static; width: 300px; padding: 10px; background-color: #096; color:#ffc; top:10px; left: 30px;">Capa Fija se coloca como una absoluta</div>  
  
<div style="position: absolute; width: 350px; height:100px; padding: 10px; background-color: #666; text-align:right; color:#ffc; top:300px; left: 500px;">  
    <div style="position: absolute; width: 300px; padding: 10px; background-color: #111; text-align:right; color:#ffc; top:0px; left: 5px;"> Capa absoluta, se referencia según a su contenedor porque no es un static </div>  
</div>  
  
<h2>La última en ponerse</h2>  
</body>
```

2.7.2 PRECEDENCIA DE ESTILOS

La precedencia de estilos es una manera de indicar que un estilo definido prevalece por encima de otro definido en la misma o en CSS diferentes. Esto es necesario principalmente cuando hay dos o más estilos que actúan sobre los mismos atributos pero con diferente valor.

La precedencia de estilos va asociado con el concepto de *especificidad* de una regla. La especificidad se refiere al peso que toman cada uno de los elementos de una hoja de estilo. Cuanto más peso más especificidad. Cuanta más especificidad tenga un regla menos problemas a la hora de garantizar que será esa y no otra la regla que se aplique sobre un determinado contenido.

Un cálculo sencillo para calcular la especificidad de una regla es sumar los puntos según el tipo de selectores que contenga:

- Se da un valor de 1 punto a un *selector de etiqueta* (por ejemplo, `<h1>`, `<p>`, `<div>`).
- A un *selector de clase* se le da el valor de 10 puntos.
- A un *selector de identificador* se le da un valor de 100 puntos.
- A un *atributo de estilo* a los que se les da un valor de 1.000 puntos. Este atributo es `style` y se han usado en la Actividad 2.5).

Con este cálculo, si se desea agregar un estilo a los párrafos `<p>` de una página agregando un selector `p{estilo}`, este será utilizado para dar estilo a todos los párrafos del HTML, pero solo se le dará el valor total de 1 punto. Un punto es muy poca especificidad. Cualquier regla con especificidad > 1 se aplicaría antes.

Si se define una clase `.parrafo {estilo}` que se aplique sobre los párrafos tendrá una especificidad de 10 puntos. Con lo cual, los párrafos `<p>` que usen el estilo de la clase `.parrafo` tendrán como estilo lo de la clase antes que los definidos con `p{estilo}`.

De la misma manera, si en vez de clase se define un selector de identificador `#id-parrafo{estilo}` esta tomará el valor de 100 puntos de especificidad, por lo que será la más importante que los de clase y el de etiqueta definidos antes.

Es importante destacar que, en el caso de que dos o más reglas en conflicto tengan el mismo valor de especificidad, se aplicará la última definida.

El siguiente código muestra las especificidades de los selectores definidos:

```
<style>
p{background: crimson;} /* Especificidad de 1 puntos */
.parrafo{background: pink;} /*Especificidad de 10 puntos*/
p.parrafo{background: maroon;} /*Especificidad de 11 puntos*/
#id-parrafo{background: orange;}/*Especificidad de 100 puntos*/
p#id-parrafo{background: red;}/*Especificidad de 101 puntos*/
p.parrafo#id-parrafo{background:green;}/*Especificidad de 111 puntos*/
</style>
```

El siguiente código HTML aplica estos estilos:

```
<body>
<p>El fondo de este párrafo será color carmesí</p>
<p class="parrafo">El fondo de este párrafo será color granate</p>
<div class="parrafo">Este div tendrá el fondo de color rosa</div>
<p id="id-parrafo" class="parrafo">El fondo de este párrafo será de color verde</p>
<p id="id-parrafo" style="background: black;">El fondo de este párrafo será negro porque
usa un atributo de estilo (con peso 1000) dentro de la etiqueta</p>
<p id="id-parrafo">El fondo de este párrafo será rojo</p>
</body>
```

Además de la especificidad, se de obligar a que un atributo de una regla se aplique por encima del resto de reglas. Esto se hace usando la declaración *!important*. Para utilizar *!important* en una regla de estilo, siempre se coloca en la parte del valor del atributo, antes del punto y coma “;”. Por ejemplo:

```
p{  
background: red !important;  
background: crimson ;  
}
```

Tenemos una declaración de estilos para los párrafos `<p>`, donde definimos dos veces el atributo *background*. En condiciones normales, se tendría en cuenta el valor definido en segundo lugar. Sin embargo, al estar el primer *background* definido como *!important* en realidad lo que ocurrirá es que prevalezca este atributo sobre el otro y se aplique un color de fondo rojo sobre todos los párrafos.

Si se sustituye esta declaración de `p` vista en el ejemplo anterior (`p{background: crimson;}`) por esta otra (`p{background: red !important; background: crimson ;}`) el resultado será que todos los párrafos tendrán color de fondo rojo (el `div` seguirá siendo rosa porque no es una párrafo y no se aplica sobre él esta regla).

2.8 CREAR Y VINCULAR HOJAS DE ESTILO

En este punto se describen las alternativas existentes para asociar un código HTML (contenido) con un estilo determinado y definido en CSS. Existen tres alternativas principales, y todas ellas han sido mostradas en los diferentes ejemplos del capítulo.

La primera alternativa es usando el atributo *style* dentro de las etiquetas HTML (reglas de estilos integradas). Por ejemplo:

```
<p style="background: black;">El fondo de este párrafo será negro </p>
```

Esta alternativa presenta alta prioridad tiene frente a otras reglas. Sin embargo esta aparente ventaja se convierte en desventaja cuando, al usarla, se pierde la posibilidad de reutilizar reglas entre elementos, teniendo que escribir todos los atributos de nuevo cuando se desea aplicar el mismo estilo a otros elementos. Además, otra desventaja añadida es que se dificulta el mantenimiento de las CSS haciendo más complicados los cambios y la localización de errores.

Otra alternativa es usando la etiqueta `<style>` dentro del mismo fichero (reglas de estilo incrustadas). En este caso la etiqueta `<style>` se coloca en la cabecera `<head>` del documento. La etiqueta `<style>` tiene varios atributos opcionales:

- **Type** (requerido): se usa para indicar que se aplica un estilo formato CSS.
- **Media**: atributo para indicar sobre qué dispositivo se aplicarán los estilos. Algunos de los valores posibles son: *handheld* (para dispositivos móviles), *print* (para salida por impresora), *projection* (para presentación en proyectores), *screen* (para pantallas), *tv* (para televisores), *braille* (para presentación en dispositivos braille) u *all* (para todos los dispositivos). Si se desean especificar varios valores se puede hacer separados por comas (,).
- **Title**: nombre que se le da al estilo. Útil cuando se vinculan CSS externas.

Un ejemplo de declaración es el siguiente:

```
<style type="text/css" media="screen, tv" title="Mi Estilo 1" >
```

En la mayor parte de los ejemplos utilizados en este capítulo se ha sido la etiqueta `<style>` aplicada a un único HTML. Este método tiene sentido cuando un único documento tenga un único estilo. Si la misma hoja de estilo se usa en múltiples documentos o páginas web, entonces sería más apropiado disponer de una hoja de estilo externa tal y como se verá en la siguiente sección.

2.9

CREAR Y VINCULAR HOJAS DE ESTILO EN CASCADA EXTERNA

La ventaja de utilizar hojas de estilo externas (CSS externas) es que se pueden reutilizar en varios documentos HTML, permitiendo así, por ejemplo, que todo un sitio web se rija por las misma CSS. De hecho, esta alternativa es la más empleada profesionalmente.

Una hoja de estilo externa puede ser enlazada a un documento HTML mediante la etiqueta `<link>` que se coloca en el `<head>` de la página. El siguiente ejemplo muestra cuatro enlaces a hojas CSS. Los atributos `type` y `media` tienen la misma semántica que en la etiqueta `<style>` vista en la sección 2.8:

```
<link rel=stylesheet href="estilo.css" type="text/css" media=screen>
<link rel=stylesheet href="color-8b.css" type="text/css" title="estilo de color 8-bit"
media="screen, print">
<link rel="alternate stylesheet" href="color-24b.css" type="text/css" title="estilo de
color 24-bit" media="screen, print">
<link rel=stylesheet href="aural.css" type="text/css" media=screen>
```

Cuando se define una CSS externa ésta no debe contener ninguna etiqueta HTML como `<head>` o `<style>`. La hoja de estilo solo debería consistir de reglas de estilo o sentencias. Un archivo que solo consista de la siguiente línea podría utilizarse como hoja de estilo externa.

```
p { margin: 2em }
```

El atributo `rel` se usa para definir la relación entre el archivo enlazado y el documento HTML. `rel=stylesheet` especifica un estilo persistente o preferido. Un estilo persistente es aquel que siempre se aplica si están activas las hojas de estilo. Un estilo preferido es uno que se aplica automáticamente, como en la segunda etiqueta `<link>` en el ejemplo. La combinación de `rel=stylesheet` y un atributo `title` especifica un estilo preferido. Los autores no pueden especificar más de un estilo preferido.

Por otro lado `rel="alternate stylesheet"` define un estilo alternativo. Un estilo alterno se indica por `rel="alternate stylesheet"`. La tercera etiqueta `<link>` en el ejemplo define un estilo alternativo, que el usuario podría elegir para reemplazar la hoja de estilo preferido. Debe tenerse en cuenta que algunos navegadores carecen de la capacidad de elegir estilos alternativos.

Un estilo simple también puede ser dado mediante múltiples hojas de estilo. Todas ellas contribuyen a dar el estilo al HTML que las importa:

```
<link rel=stylesheet href="basico.css" title="miestilo">
<link rel=stylesheet href="tablas.css" title="miestilo">
<link rel=stylesheet href="formas.css" title="miestilo">
```

En este ejemplo, tres hojas de estilo son combinadas en un estilo llamado “miestilo” que se aplica como una hoja de estilo preferido. Para combinar múltiples hojas de estilo en un estilo único, se debe usar el mismo *title* con cada hoja de estilo.

Como se ha comentado al principio, una hoja de estilo externa es ideal cuando el estilo se aplica a muchas páginas. Un autor podrá cambiar la apariencia de un sitio completo mediante el cambio de un solo archivo. Además, la mayoría de navegadores guardan en caché las hojas de estilo externas, evitando así una demora en la presentación una vez que la hoja de estilo se ha guardado en caché.

Otra alternativa para enlazar CSS externos es usar la regla *@import*. Esta regla va incluida dentro de las etiquetas `<style>`. Un ejemplo de uso de esta regla es el siguiente:

```
<style>@import url("estilos.css");</style>
```

El funcionamiento es igual que usar `<link>`. La diferencia es que *@import* no es soportada por todos los navegadores y, además, `<link>` ofrece mejores alternativas si se desea usar hojas de estilo preferentes, alternativas o preferidas. Aún así, *@import* permite hacer ciertas cosas como elegir cuál importar dependiendo del medio (media) al que se vaya a aplicar.

```
<style>
  @import url("impresora.css")print;
  @import url("normal.css")screen;
</style>
```

ACTIVIDADES 2.10



- Cree un fichero de texto con extensión “css” que contenga el interior de las etiquetas `<style>` de la Actividad 2.7. Guárdelo con el nombre “miestilo.css”.
- Cree otro fichero HTML llamado “micontenido.html” que tenga el código HTML restante de la Actividad 2.7. En ese HTML importa los estilos de “miestilo.css”: Primero usando la regla *@import* y otra con la etiqueta `<link>`.

2.10 HERRAMIENTAS Y TEST DE VERIFICACIÓN

Como se comentó al principio del capítulo CSS es un estándar de W3C. Para que los desarrolladores puedan comprobar que los estilos que definen cumplen ese estándar, el consorcio de estándares web W3C (*World Wide Web Consortium*) proporciona herramientas para validar tanto el código HTML como las hojas de estilo CSS, comprobando si éstas son correctas según las gramáticas publicadas.

A este servicio de validación se puede acceder a través de la página web: <http://jigsaw.w3.org/css-validator/>. Su uso es muy sencillo, el usuario solo tiene que introducir la URL del sitio web que pretenda evaluar y seleccionar las opciones que desea considerar.

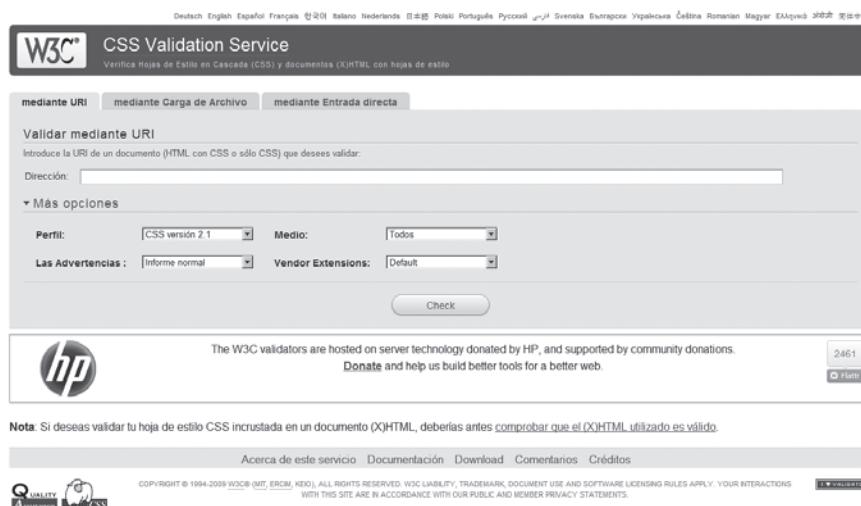


Figura 2.20. Portal de la W3C donde se ofrece un servicio de validación de CSS

Si el proceso de validación no encuentra errores, sus autores pueden incluir en la página web un ícono como el siguiente. Con la inclusión de esta imagen los visitantes verán que los desarrolladores se han preocupado por crear un sitio web interoperable y acorde al estándar.



Además de la W3C las principales herramientas de desarrollo de sitios web también ofrecen facilidades para el desarrollo y validación de hojas de estilo.

Otras herramientas de validación que pueden utilizarse para comprobar nuestras hojas de estilo podemos encontrarlas ligadas a determinados navegadores; por ejemplo *Firefox* ofrece un complemento que puede instalarse y permite validar un sitio web. Dicho complemento puede descargarse en <https://addons.mozilla.org/es-es/firefox/addon/css-validator/>.

XHTML-CSS (<http://xhtml-css.com/>) es otra herramienta de validación que puede utilizarse indistintamente para comprobar la bondad (lo adecuado) de nuestro código XHTML y el CSS. Los informes ofrecidos por estas herramientas, en muchas ocasiones, están relacionados con distintas situaciones que deben ser comprobadas por el desarrollador o diseñador del sitio web, pero que no necesariamente son errores que haya que subsanar obligatoriamente. Dichas situaciones son identificadas como *warnings*, es decir, se trata de código cuyo bondad o no debe ser comprobada en última instancia por un humano, ya que automáticamente no hay evidencias totalmente objetivas y automáticas que permitan afirmar que hay una vulneración en el uso y construcción de la hoja de estilo.

Además de herramientas de validación, también hay otras ligadas a los navegadores (*plugins*) que permiten examinar CSS y detectar errores y editar código al instante. Algunos ejemplos son:

- ✓ *Firebug* (<http://getfirebug.com/>): es un *plugin* para Firefox y Chrome (Firebug Lite) entre otros navegadores, que permite, por ejemplo, analizar CSS en cascada, editar “en vivo” reglas y atributos (propiedades) y autocompletar valores de atributos con sugerencias de contexto. Esta herramienta no es solo para ayudar en el desarrollo de CSS, sino que es un asistente para todo lo que conlleva el desarrollo web. Por ejemplo, permite inspeccionar códigos HTML y Javascript.
- ✓ *Pendule* (<https://chrome.google.com/webstore/detail/gbkffbkamcejhkcaocmkdeiicpmjfdi>): es un *plugin* de Chrome que está más especializado en CSS, permitiendo validar CSS pero también visualizar reglas, deshabilitarlas, mostrar los colores usados, etc. Es una herramienta simple para comprobar problemas en un sitio web (depurar).

Por otro lado, en Internet hay una gran cantidad de herramientas de uso gratuito que permiten generar (y ayudar a generar) código CSS automáticamente. Algunos ejemplos son:

- ✓ *List-o-matic* (<http://www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/>): permite crear código CSS para menús. Se indica el texto que aparecerá en los enlaces, se selecciona el diseño preferido y la herramienta crear el CSS que lo representa, para que el desarrollador lo copie y incorpore en su web.
- ✓ *CSS Layout Generator* (<http://www.pagecolumn.com/>): permite crear CSS para páginas web con diferentes capas, distribuidas de la manera que el usuario desee (varias columnas, horizontales, verticales, etc.). Una vez seleccionado el diseño preferido, la herramienta crear el código para copiar y pegar. Se puede elegir solo el CSS o también el HTML que lo usa.
- ✓ *CSS Text Wrapper* (<http://www.csstextwrap.com/>): es una herramienta que permite colocar mediante CSS un texto dentro de la forma que se deseé. Lo normal es colocar los textos en rectángulos. Pero esta herramienta genera código para colocarlo según la forma (circular, trapezoidal, triangular, etc.) que el usuario elija visualmente. Es una utilidad muy adecuada para hacer textos aparentes en un web.

Por último, la gran cantidad de sitios web que ofrecen de manera gratuita o de pago plantillas CSS y HTML ya creados son una gran ayuda para la creación de sitios web. Los desarrolladores, ante la idea de empezar un sitio web desde cero, pueden optar por usar una plantilla ya creada y modificarla para que obtenga toda la funcionalidad que un cliente quiera (personalización).

- ✓ *FreeCSSTemplates* (<http://www.freecsstemplaes.org/>) es uno de los sitios web que ofrecen plantillas CSS3-HTML5 gratuitas. Un desarrollador puede elegir de este sitio la plantilla que más se asemeja a las especificaciones del cliente, descargarla, y modificarla para conseguir un sitio personalizado.

Los vistos son solo algunos ejemplos de herramientas gratuitas disponibles en Internet para ayudar en la creación de CSS. Conforme pasa el tiempo, la aparición de estas herramientas se incrementa, facilitando mucho a los diseñadores la generación de código, ahorrando tiempo de desarrollo, al mismo tiempo que se crean sitios aparentes y actuales.

2.11 CONCLUSIÓN Y PROPUESTAS PARA AMPLIAR

En este capítulo se han mostrado las características más destacables de CSS. Si lugar a duda, se podría concluir que lo visto en las secciones 2.2, 2.5y 2.9 es la esencia de CSS. Comprender bien las actividades propuestas y los ejemplos de esas secciones ayuda a que el diseñador pueda controlar perfectamente todos los elementos incluidos en su web: desde el punto de vista del tamaño, posición y aspecto.

Terminado el capítulo, el lector debe entender que no ha sido posible sintetizar *todo CSS* en estas páginas. CSS tiene muchos aspectos avanzados que no han sido comentados, se han quedado en el tintero. Sin embargo, para facilitar que el lector profundice en algunos de esos aspectos no vistos, a continuación se enumeran los más destacados con el fin de facilitar al lector la búsqueda de información relativa a ellos:

- Selectores universales, que son aplicables a todas las etiquetas de un HTML.
- Atributos específicos de CSS3.
- Selectores aplicables solo a etiquetas HTML que tengan un determinado atributo. Por ejemplo:
 - Si se quiere definir un selector que solo se aplique a etiquetas `` que tengan definidos un atributo `alt`, se podría: `img [alt] { estilo que se quiera }`.
 - Si se quiere afinar más y que se aplique a `` con atributo `alt` que tenga un valor determinado (*vacaciones*) entonces sería: `img [alt="vacaciones"] { estilo que se quiera }`.
- Nomenclatura *de facto* para las partes más reconocibles de una web. Por ejemplo, a las reglas que definen la cabecera se les llama *header*, a los pies de página *footer*, etc.

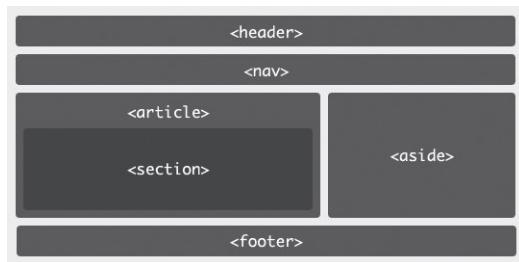


Figura 2.21. Estructura con nomenclatura “de facto”

- Encabezado: *header*; Pie:*footer*; Zona de navegación: *nav*; Secciones de la página: *section*; Artículos: *article*; Barra lateral: *sidebar*.
 - Ya que esta estructura está muy extendida, HTML5 tiene etiquetas específicas para definirlas, simplificando así el trabajo con CSS: <header> <footer> <nav> <article> <section> y <aside> (barra lateral). La Figura 2.21 muestra este esquema.
- Estilos de autor y de lector. Diferentes ámbitos de personalización.



RESUMEN DEL CAPÍTULO



En este capítulo se ha hecho una introducción al manejo y funcionalidad básica de CSS. Se han mostrado todos los elementos principales que permiten al diseñador explotar sus características desde una buena base teórica. Se ha hecho especial hincapié en el modelo de cajas y el posicionamiento de las capas usando CSS.

Con lo visto en el capítulo se deduce que, hoy en día, en el diseño web con CSS, los desarrolladores no tienen que empezar desde cero. Hay muchas herramientas, libres y de pago, que ayudan a la generación automática de código CSS a partir de modelos visuales. Además, también hay plantillas creadas por otros autores que pueden servir como base para la creación de diseños personalizados.

Todas las herramientas y posibilidades de CSS no sirven para nada si los desarrolladores no adquieren desde el principio “buenas prácticas” a la hora de generar su propio código CSS, ya que esa es la base para hacer diseños fáciles de mantener, compartir y de reutilizar entre un proyecto y otro.



EJERCICIOS PROPUESTOS



- 1. Busque en Internet al menos cuatro herramientas gratuitas que permitan generar automáticamente código CSS a partir de diseños visuales. Éstas deben ser diferentes a las mostradas en este capítulo.
- 2. En parejas, desarrollad una hoja de estilos CSS para un sitio web de un grupo musical. Cada miembro debe utilizar individualmente, por lo menos, dos herramientas que generen código CSS automáticamente (de las encontradas en el ejercicio anterior o de las mostradas en el capítulo). La unificación de los códigos generados por cada miembro debe hacerse conjuntamente para obtener el resultado final.
- 3. En parejas, utilizad una herramienta libre para verificar la bondad del código CSS realizado e interpretar sus resultados. Además, comprobar que el resultado de la plantilla es el mismo en los principales navegadores: IEXplorer, Firefox, Chrome y Safari.
- 4. Busque en Internet algún sitio web, diferente a los mostrados en este capítulo, que ofrezcan plantillas CSS3 de pago. Seleccione al menos cuatro plantillas de ese sitio que te llamen la atención. ¿Enumere las características técnicas más importantes que destacaría de las plantillas seleccionadas? Reflexione sobre qué es lo que las hace atractivas para un diseñador como para pagar por ellas.
- 5. En parejas, descargad una plantilla basada en CSS disponible en Internet. Se pueden usar los enlaces mostrados en el capítulo o cualquier otro que los miembros conozcan. Modificad esa plantilla y personalizadla para que sea una buena propuesta para una web de un restaurante de comida tradicional. Se valorará el número de cambios de la plantilla final con respecto a la original, así como las diferencias visuales entre ambas.



TEST DE CONOCIMIENTOS



- 1 Se considera buena práctica para nombrar a los selectores:
- a) Que el nombre describa una característica visual como el color.
 - b) Que el nombre no esté asociado a la localización de un elemento (salvo que se ofrezcan otros selectores con otras alternativas de localización).
 - c) Que el nombre solo contenga minúsculas y no empiece por un carácter especial.

- 2 El *padding* en el modelo de cajas:
- a) Es la separación entre el borde de esa caja y otras cajas adyacentes.
 - b) Es la separación entre el borde de esa caja y los elementos que hay en su interior.
 - c) Se puede decir que es el ancho y alto de la caja.

3 Si se desea poner un elemento que no se oculte moviendo las barras de desplazamiento:

- a) Se puede definir como *position:absolute*.
- b) Se puede definir como *position:fixed*.
- c) Se puede definir como *relative:no_movement*.

4 En la definición *p{background: red; background: crimson;}*:

- a) Los párrafos aparecerán con color fondo *crimson* por estar en segundo lugar.
- b) Los párrafos aparecerán con color *red* por estar en primer lugar.
- c) CSS dará un error por no repetir el *background* y no usar *important* en alguno de ellos.

5 A la hora de trabajar con CSS es un inconveniente:

- a) Que no haya plantillas en Internet ya hechas y disponibles para bajar y modificarlas.
- b) Que no haya herramientas visuales que permitan crear CSS automáticamente.

c) Que cuando el código CSS es muy extenso es complicado de gestionar si no se lleva un orden adecuado y se respetan las buenas prácticas para su escritura.

6 Respecto a la posición de las cajas es falso que:

- a) *Clear* elimina el efecto del *float*.
- b) Los elementos de una página se posicionan por defecto como *static*.
- c) Los valores *top*, *left*, *bottom* y *right* son una distancia con respecto al primer elemento contenedor que tenga un valor de *position* distinto de *absolute*.

3

Implantación de contenido multimedia

OBJETIVOS DEL CAPÍTULO

- ✓ Reconocer las implicaciones de las licencias y los derechos de autor en el uso de material multimedia.
- ✓ Identificar los formatos de imagen, audio y vídeo a utilizar.
- ✓ Analizar las herramientas disponibles para generar contenido multimedia.
- ✓ Utilizar herramientas para el tratamiento digital de la imagen.
- ✓ Utilizar herramientas para manipular audio y vídeo.
- ✓ Realizar animaciones a partir de imágenes fijas.
- ✓ Importar y exportar imágenes, audio y vídeo en diversos formatos según su finalidad.
- ✓ Verificar el funcionamiento de contenidos multimedia en diferentes navegadores.

En la Web abunda el material interactivo, gráfico y multimedia, pero su fácil acceso y su disponibilidad no significan que todo ese material sea gratuito ni que se pueda utilizar como si fuera propio. En este capítulo se introducen formatos y herramientas ligados a la manipulación y conversión de material multimedia y su inclusión en sitios web (imágenes, audio, vídeo y animaciones). Además, se identificará la legislación relacionada con la salvaguarda y el respeto a los derechos de autor. Dicha legislación regula, a nivel nacional e internacional, los derechos que tiene el material que se ofrece y que ofrecemos en la Web, y no supone una cortapisa a su uso sino un reconocimiento de sus legítimos propietarios y de sus derechos.

3.1 DERECHOS DE LA PROPIEDAD INTELECTUAL. LICENCIAS. LEY DE LA PROPIEDAD INTELECTUAL. DERECHOS DE AUTOR

Los derechos de propiedad intelectual y de autor son esenciales para proteger la creatividad humana al ofrecer a los autores incentivos en forma de reconocimiento y recompensas económicas equitativas. Este sistema de derechos garantiza a los creadores la divulgación de sus obras sin temor a que se realicen copias no autorizadas o actos de piratería. A su vez, ello contribuye a facilitar el acceso y a intensificar el disfrute de la cultura, los conocimientos y el entretenimiento en todo el mundo.

El ámbito y alcance de estos dos derechos, el de *propiedad intelectual* y el de autor, sin embargo es ligeramente diferente. La *propiedad intelectual* es un concepto más amplio, que abarca tanto a los derechos de autor como los llamados derechos conexos a los derechos de autor (que son otras facultades previstas a favor de otros agentes que intervienen en la creación de una obra, como los artistas o intérpretes, los productores de fonogramas y de grabaciones audiovisuales, las entidades de radiodifusión, etc.). En esta sección se van a delimitar en mayor detalle estos conceptos y cómo se pueden identificar esos derechos en la Web.

3.1.1 DERECHOS DE LA PROPIEDAD INTELECTUAL

Los derechos que conforman la propiedad intelectual se dividen en dos grupos: los *derechos morales* y los *derechos económicos*:

- **Derechos morales:** frente a los sistemas de corte anglosajón, la legislación española es claramente defensora de los derechos morales, reconocidos para los autores y para los artistas-intérpretes. Estos derechos son irrenunciables e inalienables, acompañan al autor o al artista durante toda su vida y a sus herederos al fallecimiento de aquellos. Entre ellos destaca el derecho al reconocimiento de la condición de autor de la obra o del reconocimiento del nombre del artista sobre sus interpretaciones o ejecuciones, y el de exigir el respeto a la integridad de la obra o actuación y la no alteración de las mismas.
- **Derechos de carácter económico:** hay que distinguir entre:
 - Derechos relacionados con la explotación de la obra o prestación protegida, que a su vez se subdividen en *derechos exclusivos* y en los *derechos de remuneración*. Los *derechos exclusivos* son aquellos que permiten a su titular autorizar o prohibir los actos de explotación de su obra o prestación protegida por el usuario, y a exigir de este una retribución a cambio de la autorización que le conceda.

— Los *derechos de remuneración*, a diferencia de los *derechos exclusivos*, no facultan a su titular a autorizar o prohibir los actos de explotación de su obra o prestación protegida por el usuario, aunque si obligan a este al pago de una cantidad dineraria por los actos de explotación que realice, cantidad esta que es determinada, bien por la ley o en su defecto por las tarifas generales de las entidades de gestión.

■ **Derechos compensatorios:** como el derecho por copia privada que compensa los derechos de propiedad intelectual dejados de percibir por razón de las reproducciones de las obras o prestaciones protegidas para uso exclusivamente privado del copista.

Las preguntas que surgen en función de los derechos de propiedad intelectual y de tus necesidades como creador y autor de sitios web pasarían por ser capaz de responder a preguntas como las siguientes: ¿Dónde encuentro recursos digitales para enriquecer mi trabajo? ¿Puedo utilizarlos libremente? ¿Tengo obligación de citar la fuente u origen de los recursos que utilice? ¿Cómo sé qué tipo de uso puedo hacer de los recursos que he encontrado en la Web? ¿Hay alguna forma de enriquecer con mi trabajo a la comunidad web? ¿Qué derechos tengo sobre los materiales que produzco? Muchas de estas preguntas tienen su respuesta en la licencia con la que se ofrezcan los recursos en la Web.

3.1.2 LICENCIAS

Una *licencia de software* es un contrato entre el autor/titular de los derechos de explotación/distribuidor y el usuario consumidor/usuario profesional o empresa, para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Las licencias de software pueden establecer entre otras cosas: la cesión de determinados derechos del propietario al usuario final sobre una o varias copias del programa informático, los límites en la responsabilidad por fallos, el plazo de cesión de los derechos, el ámbito geográfico de validez del contrato e incluso pueden establecer determinados compromisos del usuario final hacia el propietario, tales como la no cesión del programa a terceros o la no reinstalación del programa en equipos distintos al que se instaló originalmente.

El *copyright* es la formula anglosajona para designar únicamente los derechos de explotación de una obra, no hace relación a los derechos morales. El símbolo © asociado a un nombre indica titularidad de derechos de explotación. Normalmente, aunque no necesariamente, va seguido de la expresión “todos los derechos reservados”

El *copyleft* es un movimiento social y cultural alternativo al sistema tradicional del *copyright* que aboga por el uso de licencias libres para compartir y reutilizar las obras de creación. Hay diferentes tipos de licencias libres entre las que se puede elegir según el ámbito que se trate (software, obra científica, música, arte, etc.). Las más utilizadas son las licencias *Creative Commons* de origen norteamericano, pero también existen otros modelos, por ejemplo las **licencias Coloriuris**, de iniciativa española, que se están abriendo paso en la comunidad de habla hispana.

En cualquier caso, no se puede confundir una licencia *Creative Commons* con una *Copyleft* o “todo libre”. *Creative Commons* plantea un paso intermedio entre el férreo sistema de *copyright* y el “libre total” (*Copyleft*). *Creative Commons* pone a disposición de los autores licencias “a la carta”, cuya redacción se incorpora en el sitio web o soporte donde esté la obra y donde se regulan los usos autorizados por el autor con respecto a la referida obra. Esto, evidentemente, también permite que los autores puedan decidir que su obra pase automáticamente a dominio público.

Más concretamente, en las licencias *Creative Commons* el autor otorga a la comunidad una mayor libertad de uso sobre su obra aunque bajo determinadas condiciones. Estas condiciones son escogidas por el propio autor, de modo

que frente una obra con todos los derechos reservados las licencias *Creative Commons* proponen “algunos derechos reservados”. Tras un proceso de adaptación tenemos en España desde el año 2004 diferentes modelos posibles de licencias *Creative Commons* a elegir, en función de lo que se pretenda. Entre ellas están las siguientes (la iconografía utilizada para distinguir estas licencias se muestran en la Figura 3.1.):

- **Reconocimiento:** ⓘ El creador permite copiar, distribuir y comunicar públicamente la obra mientras se reconozca y cite adecuadamente al autor original.
- **No comercial:** ⓘ Se permite copiar, distribuir y comunicar públicamente la obra mientras no sea utilizada con fines comerciales.
- **Prohibición de obras derivadas:** ⓘ El creador permite copiar, distribuir y comunicar públicamente copias inalteradas de la obra, pero no realizar trabajos derivados de ellas.
- **Redistribución bajo la misma licencia:** ⓘ Se permite distribuir obras derivadas solo bajo una licencia idéntica a la que regula la obra original.

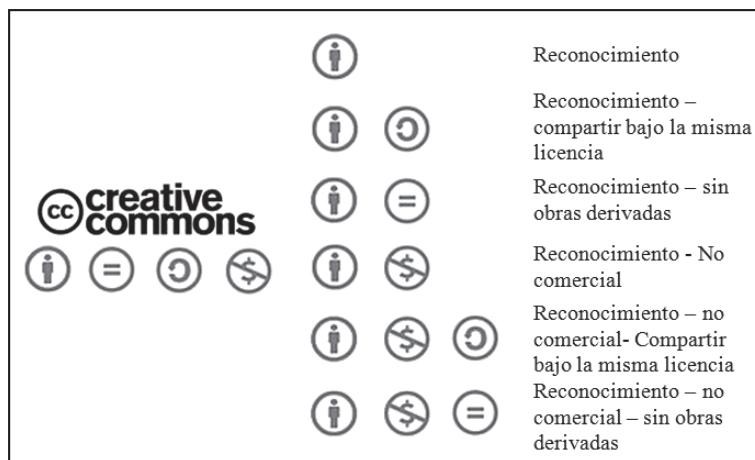


Figura 3.1. Licencia Creative Commons y diferentes modelos de licencia

Algunos sitios web donde se pueden buscar y descargar imágenes con este tipo de licencia son los siguientes: *Picfindr*²³, *search.creativecommons.org*²⁴, *Pixsy*²⁵, *Flickr*²⁶ o *Everystockphoto*²⁷.

²³ <http://www.picfindr.com/>

²⁴ <http://search.creativecommons.org/>

²⁵ <http://www.pixsy.com/>

²⁶ <http://www.flickr.com/>

²⁷ <http://www.everystockphoto.com/>

3.1.3 PROPIEDAD INTELECTUAL

La *propiedad intelectual* es el conjunto de derechos de autor, personales (morales) y patrimoniales (económicos) que corresponden a los autores sobre las obras de su creación.

En otras jurisdicciones y para la OMPI (Organización Mundial de la Propiedad Intelectual) la expresión *intellectual property* engloba tanto los derechos de propiedad industrial (marcas, patentes, diseño industrial, denominaciones de origen) como los derechos de propiedad intelectual (derechos de autor y derechos afines).

La norma nacional principal en lo que se refiere a Propiedad Intelectual es el Texto Refundido de la Ley de Propiedad Intelectual, aprobado por Real Decreto Legislativo 1/1996 de 12 de abril, que ha sido objeto de algunas modificaciones posteriores.

También de consideración son las Directivas Comunitarias sobre la materia, especialmente la Directiva 2001/29/CE sobre derechos de autor y derechos afines en la sociedad de la información.

Normas internacionales importantes son: el Tratado de la Unión de Berna, firmado en 1886 y revisado en varias ocasiones, y los Tratados de la OMPI de 20 de diciembre de 1996, uno sobre derechos de autor y otro sobre derechos afines.

Las obras que son objeto de propiedad intelectual son todas las creaciones originales literarias, artísticas o científicas expresadas por cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro. Lo que no puede incluirse en la definición anterior son las ideas, la información, y todo conocimiento que es patrimonio común y no es susceptible de apropiación.

Expresamente la ley excluye las disposiciones legales y reglamentarias, sus correspondientes proyectos, las resoluciones de órganos jurisdiccionales, actos, acuerdos, deliberaciones y dictámenes de organismos públicos y traducciones oficiales de todos ellos.

3.1.4 DERECHOS DE AUTOR

Los *derechos de autor*, al igual que se ha comentado para los derechos de propiedad intelectual, son de dos clases:

- *Derechos morales, irrenunciables e inalienables*, como el derecho de reconocimiento de autoría y el derecho de integridad de la obra, entre otros.
- *Derechos económicos, transferibles* y de duración limitada en el tiempo, básicamente los derechos de explotación, aunque hay otros (i.e. derecho a remuneración por copia privada).

Se considera *autor* a la persona natural que aparece como tal en la obra. En algunos casos previstos por la ley las personas jurídicas pueden tener algunos derechos económicos de propiedad intelectual.

En las obras en colaboración, es decir, en aquellas en las que hay varios autores, los derechos pertenecen a todos los autores, sin embargo en las obras colectivas corresponden a la persona bajo cuya iniciativa y coordinación se edita y divulga la obra.

3.2 IMÁGENES: MAPA DE BITS, IMAGEN VECTORIAL. SOFTWARE PARA CREAR Y PROCESAR IMÁGENES. FORMATOS DE ARCHIVOS DE IMÁGENES

3.2.1 TIPOS DE IMÁGENES EN LA WEB

Uno de las principales decisiones a la hora de incluir imágenes en una web es elegir el formato correcto para cada tipo de imagen de manera que se logre una correcta relación entre la calidad visual de la misma y su peso en bytes. Básicamente, existen distintos tipos de imágenes: las que son mapas de bits, las vectoriales y las imágenes animadas. Estas últimas se tratarán en la Sección 3.6 dedicada a las animaciones. Esta sección se centra en los otros dos grupos de imágenes (vectoriales y mapas de bits).

Cuando se habla de formatos de imagen en la Web, tradicionalmente se han contemplado las imágenes de mapa de bits, dentro de las que se encuentran imágenes que se pueden identificar a través de sus extensiones: *.GIF*, *.JPG* y *.PNG*. Las principales características de los tres tipos de imágenes mencionados son las siguientes:

- Las imágenes *.GIF* (*Graphic Image File Format*) hacen uso de un formato más adecuado para aquellas imágenes sencillas, de formas simples y en las que no existe un elevado número de colores. Sus características son:
 - Número de colores: de 2 a 256 de una paleta de 24 bits.
 - Formato de compresión sin pérdida basado en el algoritmo LZW.
 - Carga progresiva en el navegador.
 - Máscara de transparencia de 1 bit.
 - Permite la animación simple. Se comentará en la Sección 3.6.1 dedicada a las animaciones.
- El formato en JPEG (*Joint Photographic Experts Group*) fue diseñado para la compresión de imágenes fotográficas, basándose en el hecho de que el ojo humano no es perfecto y no es capaz de captar toda la información que se puede almacenar en una imagen de 24 bits. El formato JPEG intenta eliminar la información que el ojo humano no es capaz de distinguir, por eso se dice que posee un formato de compresión con pérdida, porque elimina información. Por regla general, es el más indicado para aquellas imágenes que son fotografías. Las características de este formato son:
 - Número de colores: 24 bits color o 8 bits B/N.
 - Elevado grado de posibilidad de compresión.
 - Formato de compresión con pérdida.
 - No permite transparencias.
 - No permite la animación.

■ El formato PNG (*Portable Network Graphics*) es el más adecuado para imágenes renderizadas, es decir que provienen de un modelo, ya que se logran unos degradados más suaves y una buena definición de las líneas. Este formato proporciona una compresión de imágenes sin pérdida, a diferencia del formato JPG. Las principales características de este formato son:

- Color indexado hasta 256 colores y TrueColor hasta 48 bits por píxel.
- Mayor compresión que el formato GIF (+10%).
- Compresión sin pérdida a diferencia del formato JPG.
- Canal alfa (transparencia variable).
- No permite animación, a diferencia del formato GIF.

Como ya se ha resaltado, GIF, JPEG y PNG son formatos gráficos de mapa de bits, es decir, que su estructura se basa en una matriz de puntos, cada punto correspondiente a un punto virtual del gráfico, y cada uno de ellos incorpora la información correspondiente al color que le es propio.

Las limitaciones de este tipo de gráficos son evidentes, ya que ofrecen muchas dificultades a la hora de hacer modificaciones sobre el gráfico original (por ejemplo, cambiar texto, color, etc.), o simplemente a la hora de redimensionar la imagen, ya que se redimensionan los píxeles y no los elementos independientes, haciendo que estos pierdan definición y calidad.

Todas esas limitaciones se solucionan con otro tipo de imágenes, las imágenes vectoriales. Las imágenes vectoriales son representaciones de entidades geométricas tales como círculos, rectángulos o segmentos. Están representadas por fórmulas matemáticas (un rectángulo está definido por dos puntos; un círculo, por un centro y un radio; una curva, por varios puntos y una ecuación). El procesador de un ordenador traduce estas formas en información que la tarjeta gráfica pueda interpretar. La ventaja de las imágenes vectoriales es doble, ya que a la posibilidad de rediseño posterior de la imagen hay que sumar que muchas veces el tamaño de estos gráficos puede ser muchísimo menor.

Los formatos de gráficos vectoriales para la Web se pueden dividir en dos tipos: los soportados mediante el *plugin* del navegador (complemento) correspondiente y los basados en lenguajes de marcado.

Los primeros son los gráficos vectoriales clásicos, que necesitan del *plugin* correspondiente para poder ser visualizados en un navegador web concreto.

El segundo tipo, los basados en lenguajes de marcado, son la última alternativa en la implantación de imágenes vectoriales. En las imágenes vectoriales basadas en lenguajes de marcado, los elementos de esas imágenes se definen con texto, que después es interpretado por el navegador web, al igual que un archivo HTML.

Los estándares más populares de formatos de imagen vectorial son: *.eps* (archivo *postscript*), *.ps*, *.pdf*, *.fla* y *.swf* (definidos por Adobe), *.wmf* (definido por Microsoft Windows), y *.svg* (definido por la W3C).

Las imágenes se incluyen en las páginas web con la etiqueta ``, independientemente de su tipo, y otra serie de elementos cuyo soporte depende de cada navegador, pero que el propio HTML5 no especifica, ya que queda en manos de cada uno de ellos. Los principales atributos ligados a la etiqueta `` son los siguientes:

- **src**: este atributo es obligatorio al añadir una imagen e indica la dirección URL donde se encuentra el elemento a mostrar.
- **alt**: este elemento es el “texto alternativo” en una imagen. Es un atributo importante ya que está muy relacionado con el soporte a la característica de accesibilidad en la Web²⁸. Dependiendo de una serie de combinaciones hay que ponerlo o no.
- **width/height**: las imágenes pueden llevar el tamaño de las mismas. Por norma general es interesante indicarlas ya que de esta forma el navegador no tendrá que esperar a finalizar la carga de la página para poder acabar de renderizarla correctamente.
- **usemap**: este elemento, si existe, indica la información del mapa asociado y será el nombre del mapa.
- **ismap**: Si la imagen es un mapa y se encuentra dentro de un enlace, entonces hay que indicarlo con este parámetro booleano. Con esto el enlace quedaría en entredicho ya que necesitará de un mapa.

Un ejemplo de imagen definida en una página web utilizando la etiqueta y los atributos anteriormente presentados es el siguiente:

```

```

3.2.2 SOFTWARE PARA CREAR Y PROCESAR IMÁGENES

Para la creación, manipulación y tratamiento de imágenes existe una gran cantidad de oferta de herramientas, tanto comerciales como gratuitas, algunas de estas últimas se ofrecen a través de la Web. Seguidamente se identifican algunas de ellas:

Una de las herramientas comerciales más conocidas para el tratamiento digital de imágenes, de hecho algunas personas han llegado a asociar retoque de imágenes y esta herramienta, es Adobe Photoshop²⁹. Photoshop es un producto software para la creación, edición y retoque de imágenes. Los formatos propios de Photoshop son PSD y PDD, que guardan capas, canales, guías y cualquier modo de color. Además de los formatos comentados también soporta la mayoría de tipos de imágenes comentados en secciones anteriores.

Como alternativa en el ámbito de software libre a Photoshop está GIMP. GIMP³⁰ (*GNU Image Manipulation Program*) es un programa de edición de imágenes digitales en distintos formatos. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU.

Algunas herramientas software para manipular, optimizar y crear imágenes de acceso a través de la Web son las siguientes:

- **Pixlr**³¹: es un editor de imágenes *on line*, fácil de manejar y permite realizar las operaciones básicas de tratamiento de imágenes. Tiene menor potencia que Photoshop o GIMP, pero es más fácil de manejar.

²⁸ Se volverá a considerar este tema en los Capítulos 5 y 6

²⁹ <http://www.adobe.com/>

³⁰ <http://www.gimp.org/>

³¹ <http://pixlr.com/>

- **Cellsea Free Web Photo Editor**³²: Cellsea es un editor de imágenes y fotografía *on line* bastante completo. Tiene todo tipo de herramientas y, en lo que a retoque de imágenes se refiere, poco tiene que envidiarle a herramientas comerciales como Photoshop.
- **Fauxto**³³: otra herramienta muy recomendable para dar soporte a la edición de fotos e imágenes *on line*. Se ofrece a través de una interfaz de usuario casi igual a la de Photoshop y tiene las mismas herramientas, permite utilizar capas (*layers*) y filtros. Es muy usada por profesionales y no es tan simple como la anterior herramienta.
- **ImageTool**³⁴: herramienta para crear logos *on line*. El usuario tiene que elegir el color de fondo y las propiedades del texto. Con esos datos es suficiente.

3.3 OPTIMIZACIÓN DE IMÁGENES PARA LA WEB. RESOLUCIÓN

Al crear un sitio web es muy recomendable que los archivos que contienen las imágenes ocupen el menor número posible de bytes para agilizar su descarga y visualización por Internet. Esto garantizará el acceso de aquellos clientes que utilicen conexiones con anchos de banda modestos.

El tamaño de un archivo gráfico viene determinado, entre otros, por los siguientes factores:

- Dimensiones de la imagen.
- Profundidad o paleta de colores.
- Resolución.
- Formato de archivo (JPG, GIF, PNG).

Recomendaciones de optimización

Conviene definir una resolución de imagen no superior a 96 ppp. Es la resolución que usan las pantallas de ordenador. No interesa optar por valores mayores, ya que aumenta considerablemente el peso del archivo a descargar y el usuario no lo aprecia.

En ocasiones puede interesar reducir el número de colores de la paleta porque ello supone disminuir el tamaño del archivo.

Conviene utilizar un programa de tratamiento de imágenes para definir las dimensiones concretas de una imagen antes de insertarla en una página web.

Es recomendable guardar los originales de las imágenes favoritas en formato BMP, TIFF ó JPEG sin comprimir. A partir de ellas se puede crear una copia en formato GIF (PNG) o JPEG con las dimensiones, resolución y paletas optimizadas para publicarlas en la web.

³² <http://editor.cellsea.com/>

³³ <http://www.fauxto.com/>

³⁴ <http://www.imageTool.net/>

Las imágenes GIF son más adecuadas para dibujos, gráficos y logotipos. Son aquellas donde predominan los colores sólidos y una paleta con un número reducido de colores.

Las imágenes JPEG se adaptan mejor a fotografías e imágenes con degradados complejos. Admiten color de 24 bits y gracias a su compresión ofrecen una imagen más brillante que ocupa menos espacio.

La correcta combinación de aspectos relacionados con la resolución, el color, el número de bits y el formato, es muy importante para la optimización de una imagen. Puesto que en la web se deben cumplir, por encima de todo, dos principios: que las imágenes se vean bien y que pesen poco, para que no tarden mucho en cargarse.

Herramientas de optimización

La mayoría de las herramientas consideradas en la sección anterior nos permiten optimizar las imágenes para utilizarlas en la Web, pero hay herramientas específicas y relacionadas con actividades de optimización y reducción del tamaño, por ejemplo:

- **Image Optimization**³⁵: con esta herramienta *on line* se puede cargar una imagen de mapa de bits (GIF, JPG o PNG) y optimizarla para su uso en la Web, sin sacrificar calidad.
- **DoSize**³⁶: permite cambiar el tamaño de una imagen y adaptarlo a las necesidades del diseñador.

ACTIVIDADES 3.1



- Busque en Internet imágenes libres que puedan ser incorporadas en el esqueleto del Restaurante e incorpore alguna en alguna de sus partes con características adecuadas según las recomendaciones vistas en ésta y las anteriores secciones.

3.4 AUDIO: INSERTAR AUDIO EN UNA WEB

3.4.1 FORMATOS

Al hablar del audio en Internet, el formato más conocido es *mp3*. Éste es un formato de compresión de audio digital patentado que usa un algoritmo con pérdida para conseguir un menor tamaño de archivo. Es un formato de audio común usado para música tanto en ordenadores como en reproductores de audio portátil. Más concretamente, *mp3* fue desarrollado por el *Moving Picture Experts Group* (MPEG) para formar parte del estándar MPEG-1 y del posterior y más extendido MPEG-2. Un *mp3* puede comprimirse usando una mayor o menor tasa de bits por segundo, resultando directamente en su mayor o menor calidad de audio final, así como en el tamaño del archivo resultante.

³⁵ <http://www.sitereportcard.com/imagereducer.php>

³⁶ <http://www.dosize.com/>

Es importante destacar que, en lo que respecta al desarrollo web y al uso del audio para ser interpretado por los diferentes navegadores, *mp3* no es único y universal, es decir, hay más formatos iguales o mejores y no todos los navegadores lo soportan (por ejemplo, Firefox no interpreta *mp3* por sí solo). Por lo tanto, hay otras posibilidades de formatos que hay que conocer, sobre todo para, como luego veremos, permitir que la reproducción de audio sea lo más universal posible entre todos los navegadores actuales (al margen de sus batallas empresariales).

Cuando se habla de archivos comprimidos es importante saber distinguir entre formato y códec. El formato es la estructura del archivo que guarda el audio, mientras que códec hace referencia al tipo de compresión que se utiliza o ha utilizado para comprimir un audio en un archivo. Comparado con la imágenes, el formato es cómo se guarda la imagen (*jpg*, *tiff*, *bmp*, etc.) y el códec es lo equivalente a usar una compresión *rar* o *zip* para esos archivos. Para reproducir un audio en un formato que es comprimido (codificado) con un códec determinado se necesita ese códec para decodificarlo.

Los siguientes formatos no son los únicos que existen, pero sí son los más conocidos en Internet, y los más usados en la distribución de audio en sitios web.

- **Ogg:** es un formato válido para audio y vídeo. Es un formato contenedor, desarrollado por la Fundación *Xiph.org*. Su principal ventaja con respecto a *mp3* es que es un formato libre de patentes y abierto diseñado para dar un alto grado de eficiencia en el *streaming*³⁷ y la compresión de archivos. De manera incorrecta, muchas veces los archivos *ogg* se les llama *Vorbis* ya que éste fue el primer códec desarrollado para leer este formato. Sin embargo, hay muchos otros códec además de *Vorbis* que interpretan *ogg*. Actualmente, Firefox y Chrome interpretan *ogg*.
- **Real Audio:** sin duda, este formato es el rey del *streaming*. La gran mayoría de las páginas que usan *streaming* utilizan el formato Real Audio para difundir sus contenidos. RealNetworks es la compañía que ha creado Real Audio y, por supuesto, también ha creado un reproductor (*player* en inglés) para su formato, el RealPlayer (del que existe una versión gratuita y otra comercial).
- **Windows Media Audio (*wma*):** es un formato desarrollado por Microsoft. Ofrece gran calidad de sonido en un tamaño reducido. Y como ventaja para las compañías y los músicos, el formato de Microsoft incluye un sistema de gestión de los derechos de autor para evitar la piratería y la creación de copias no autorizadas de las canciones. Por las características de calidad y tamaño *wma* resulta muy adecuado para la reproducción de archivos en *streaming*. El reproductor que Microsoft ha desarrollado para escuchar su formato de audio es el Windows Media Player que se encuentra disponible en su web. La desventaja de *wma* es que, al ser de Microsoft, el navegador IExplorer lo soporta, pero no se puede asegurar en el resto.
- **wav o wave (WAVEform audio file format):** es un formato de audio que se suele utilizar sin compresión de datos. wav es propiedad de Microsoft y de IBM. Este formato es muy conocido en entornos Windows, pero no se suele usar en Internet por ocupar mucho espacio al ser usado sin compresión.
- **vqf:** el formato de audio *vqf* ha sido desarrollado por la compañía Yamaha y presenta algunas ventajas respecto al *mp3*: mayor calidad de sonido y archivos de menor tamaño (se puede comprimir hasta un 30% más que *mp3*). El reproductor más asociado a este archivo es el Yamaha SoundVQ, aunque hay más que lo soportan. El mayor inconveniente del *vqf* es que no está muy extendido (por lo que muchos navegadores no lo soportan), además, su reproducción necesita de más recursos que cualquiera de los formatos anteriores.

³⁷ Lectura del archivo conforme se descarga, con lo que no es necesario grabarlo antes de escucharlo, por ejemplo, las emisoras de radio en Internet usan *streaming* para su difusión.

3.4.2 CONVERSIÓN DE FORMATOS

Un hándicap importante a la hora de trabajar con audio en páginas web es saber cómo implementa cada navegador los reproductores. No todos los navegadores tienen reproductores para todos los formatos, sino que hay preferencias y lo que un navegador puede reproducir otro no. Esta situación lo único que hace es complicar el trabajo del desarrollador, ya que éste debe intentar garantizar que los audios que inserte en un sitio web puedan ser reproducidos por todos los navegadores, y si eso no es posible, por aquellos más utilizados por el usuarios objetivo del portal.

La Tabla 3.1 muestra una comparativa de qué formatos son reproducidos por los 5 navegadores más destacados actualmente (bajo Windows). Esta lista no es definitiva, ya que la vertiginosa actualización de los navegadores hace que las prestaciones que ofrecen en una versión sean mejoradas o reducidas en las siguientes versiones.

Tabla 3.1 Formatos de audio y navegadores

Formato	IE 9	Firefox 7	Opera 11.5	Chrome 14.0	Safari 5.0
Ogg	No	Sí	Sí	Sí	No
Mp3	Sí	No	No	Sí	Sí
Wav	No	Sí	Sí	Sí	Sí

Los valores de Tabla 3.1 hay que tenerlos en cuenta a la hora de saber qué formatos insertar como audio en un sitio web. Como se verá en los siguientes puntos, HTML5 ofrece una alternativa para insertar varios formatos de un audio dependiendo de cuál es el que soporta un determinado navegador. El desarrollador guarda el mismo audio en diferentes formatos, y en HTML5 se referencian todos ellos para que cada navegador elija el que puede reproducir.

Para poder tener un audio en varios formatos, es necesario usar herramientas de conversión. Estas herramientas suelen soportar una gran cantidad de formatos y suelen ser sensibles respecto a los parámetros de compresión en aquellos que lo soportan.

Herramientas de conversión hay muchas disponibles en Internet. Algunas gratuitas son:

- **Free Studio**³⁸: es una potente herramienta para convertir todo tipo de archivos (vídeo, audio, imagen). Es fácil de usar y muy adecuada para un diseñador que trabaja con audio en la web. Solo versión para Windows.
- **Audacity**³⁹: es una herramienta avanzada para grabar y editar. Es libre y de código abierto. Su funcionalidad es mucha, y entre ella está la de convertir archivos. Es muy adecuada si el diseñador desea crear sus propios sonidos. Para Linux y Windows.

También hay conversores *online*, servicios web a los que se le sube el archivo de audio, se selecciona el formato al que se quiere convertir y el resultado se envía por *email*. Un ejemplo de este tipo de herramientas es *Switchr*⁴⁰, sin embargo hay muchas más disponibles en Internet.

³⁸ <http://www.dvdvideosoft.com/es/free-dvd-video-software.htm>

³⁹ <http://audacity.sourceforge.net/?lang=es>

⁴⁰ <http://switchr.net/>

3.4.3 INSERTANDO AUDIO EN UNA WEB

Vistos los diferentes formatos de audio adecuados para Internet, cuáles son soportados por los navegadores más adecuados, y herramientas de conversión de formatos, en este punto se muestran diferentes maneras de introducir audio en una web con HTML.

Audio de fondo (en HTML4)

Poner un sonido de fondo para una página web usando etiquetas HTML no es la opción más versátil. Existen dos etiquetas que, en teoría, lo permiten <bgsound> y <embed>.

- <bgsound> es una alternativa aparentemente adecuada pero con el inconveniente de que es solo entendible por Internet Explorer. Evidentemente, esto la hace muy poco portable por lo que no se recomienda su uso, salvo en situaciones muy concretas. Un ejemplo de uso para ejecutar de fondo un sonido *audios/Beethoven.wma* (solo válida para IExplorer):

```
<bgsound src="audios/Beethoven.wma" loop="2" volume="20" />
```

Lo más importante de esta etiqueta es que para indicar el fichero a reproducir hay que utilizar el atributo **src**. Otros atributos de interés son **loop**, que permite que la canción vuelva a empezar cada vez que acaba (se le puede indicar cuántas veces se repetirá, con **INFINITE** no hay límite); **delay**: es un valor para retrasar el inicio de la música (debe ser un número positivo en segundos); **volumen**, determina la intensidad del sonido de fondo, con valores entre -10.000 (el más débil) a 0 (el más alto).

- <embed> es una alternativa para insertar complementos (*plugins*) de audio y vídeo que es compatible para todos los navegadores. Sin embargo, no está incluida dentro del estándar W3C por lo que su implementación depende de los navegadores, y no todos las interpretan igual. Además, dependiendo del formato del archivo, así lo interpretarán los navegadores. Por ejemplo, en Google Chrome no funciona el reproductor por defecto cuando la extensión es *wma* (Microsoft) pero sí con *mp3*. Un ejemplo de código es el siguiente:

```
<embed src="audios/Aretha.mp3" height="0" type="audio/mpeg" loop="false" controller="false">
```

El atributo **src** sirve para especificar la ruta del archivo (como en <bgsound>). Los audios insertados a través de esta etiqueta se reproducen automáticamente al cargarse la página, y se reproducen solamente una vez. Esto puede cambiarse a través de los atributos *autoplay* y *loop*. El atributo *autoplay* indica si el archivo se reproducirá automáticamente al cargarse la página, y puede tomar los valores *true* o *false* (aunque no funciona correctamente para todos los navegadores). El atributo *loop* indica si el archivo se reproducirá continuamente en un bucle, y también puede tomar los valores *true* o *false*. Los atributos *width* y *height* sirven para especificar el tamaño de la consola de control de sonido (o vídeo). Estos atributos pueden tomar como valor un número, que indica el tamaño en píxeles. Si no se especifican estos atributos, la consola de control de vídeo se mostrará con el tamaño más adecuado al tamaño del sonido. *Controller* indica si aparece o no la consola de controles del reproductor. *Type* es un atributo importante, que declara el tipo de fichero de audio que estamos usando, con lo que el navegador web puede ejecutar el complemento o *plugin* adecuado para la reproducción del fichero. Puede ser *audio/midi*, *audio/wav* o *audio/mpeg*, entre otros.

En cualquier caso, porque depende del navegador y del *plugin* necesario para su inclusión, ninguna de estas dos alternativas ofrece una solución interoperable entre todos los navegadores, por tanto, es poco recomendable su uso.

Descargar audio por el usuario (en HTML4)

La manera más sencilla de incluir sonidos es dejando al usuario la decisión de escucharlos o no en local. En realidad es más una opción *descargar* que *reproducir*. Para ello se puede colocar el nombre del archivo de sonido en una atributo *href* de la etiqueta `<a>`. El siguiente código muestra un ejemplo.

```
<a href="audios/Beethoven.wma">Audio de Beethoven</a>
```

En este ejemplo, dependiendo de la versión del navegador que se esté usando, se le da al usuario la opción de descargar el archivo en local o se carga un complemento (*plugin*) que lo ejecuta. Pero todo necesita de descargarse en local y ejecutarlo con un reproductor (*player*).

Insertar sonido (HTML5)

En HTML5 la inclusión de sonidos en páginas web intenta paliar muchos de los inconvenientes vistos con las etiquetas `<bgsound>` o `<embed>`. De alguna manera, HTML5 ofrece una alternativa más sencilla y eficaz para que el desarrollador pueda insertar audio. Sin embargo, la realidad es que cada navegador implementa de HTML5 lo que le quiere, por lo que tampoco con esta opción se consigue una solución universal.

La propuesta de HTML5 utiliza la etiqueta `<audio>`. Esta etiqueta es nueva en HTML5 y no existe para versiones de HTML4. Evidentemente, eso obliga a que los navegadores que la interpreten deban implementar HTML5 en general (y esta etiqueta en particular).

Los atributos de esta etiqueta son muy parecidos a los vistos para `<embed>`.

Autoplay especifica que el audio debería empezar automáticamente tan pronto como esté listo para reproducirse (el único valor de este atributo es *autoplay*). *Controls* indica que los controles de reproducción serán visibles (el único valor posible de este atributo es *controls*). *Loop* indica que el audio se repetirá automáticamente cuando termine (el único valor posible de este atributo es *loop*). *Preload*, especifica si el audio debería ser cargado cuando la página se carga o no, es decir, se empieza a cargar cuando el usuario lo quiera reproducir (el valor *auto* indica que se cargará todo el archivo cuando se cargue la página, el valor *meta* indica que solo se cargarán los metadatos del archivo y *none* que no habrá precarga). Por último, *src* especifica la URL de archivo de audio. Además de estos atributos, `<audio>` soporta también atributos globales de HTML5 y eventos⁴¹.

Un ejemplo del uso de esta etiqueta para obtener un reproductor para un audio que se inicia solo, sería:

```
<audio src="audios/Aretha.mp3" controls="controls" autoplay></audio>
```

Este ejemplo funciona correctamente en IExplorer 9 y en Google Chrome 14. La apariencia de los reproductores es diferente, pero ambos reproducen el archivo *mp3* obedeciendo al atributo *autoplay*. Es importante destacar que el uso de HTML5 abre las puertas de navegadores de dispositivos móviles como IPad, iPhone y los basados en Android o Windows 7. El caso de IPad (que hasta la fecha) no interpreta archivos Flash, la etiqueta `<audio>` ofrece una alternativa muy elegante y sencilla para los desarrolladores.

Una mejora para hacer más versátil esta opción es usar la etiqueta `<source>` dentro de `<audio>`. De esta manera, el navegador intenta cargar la primera línea `<source>`. Si falla o no es soportada esa reproducción, entonces pasa a la siguiente. Por ejemplo, si IExplorer, Safari y Google Chrome leen archivos *mp3* y Firefox lee archivos *ogg* (no soporta

⁴¹ Ver la especificación de HTML5 para conocer más sobre atributos y eventos en HTML5 (<http://www.w3.org/TR/html5/>).

mp3), el siguiente código hace una solución para los cuatro navegadores. Cuando el navegador Firefox llegue al primer `<source>` y no pueda interpretar *mp3* entonces pasará al segundo con *ogg*. Sin embargo, Internet Explorer, Safari y Google Chrome se quedarán con el primer `<source>`. El siguiente ejemplo empieza definiendo el tipo de documento (`<!DOCTYPE html>`) que es: navegadores como IEExplorer lo necesitan para interpretara bien el contenido.

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

<audio controls="controls" preload="auto" >
  <source src="Aretha.mp3" type="audio/mpeg" />
  <source src="Aretha.ogg" type="audio/ogg" />
    El navegador NO soporta el audio
</audio>

</body>
</html>
```

La Figura 3.2. muestra el resultado en Google Chrome.



Figura 3.2. Reproductor en Google Chrome con etiqueta `<audio>`

En resumen, este código funcionará en los navegadores más conocidos: IEExplorer 9, Firefox 7, Google Chrome 14 y Safari 5, pero no funciona en IEExplorer8. Si se quitara la línea con el archivo *ogg* entonces Firefox 7 no lo soportaría ya que no lee archivos *mp3*, pero sí en el resto de navegadores antes enumerados. Por otro lado, si se quitara la línea con el archivo *mp3*, en Google Chrome 14 y Firefox 7 funcionaría (porque leen *.ogg*), pero en IEExplorer 9 y Safari 5 no.

A modo de conclusión, el uso de la etiqueta `<audio>` hace que insertar audio en la web sea bastante interoperable entre diferentes navegadores (aquellos que interpretan HTML5). Sin embargo, un diseñador, debería hacer soluciones que también la soporten otros navegadores más antiguos. Es decir, en el ejemplo anterior ¿qué pasa si un usuario tiene IEExplorer8? Si ese usuario no puede ver correctamente los elementos de su sitio web lo más seguro es que lo abandone defraudado.

Una alternativa incluida dentro de HTML4 para insertar objetos no soportados por el navegador es el uso de la etiqueta `<object>`. Combinando `<object>` con `<embed>` se puede, por ejemplo, insertar un reproductor (Flash) para

aquellos navegadores que no soporten el código mostrado en el ejemplo anterior (en nuestro caso solo era IExplorer8). Esto ofrece una solución más interoperable, aunque tampoco es universal.

El siguiente código muestra un ejemplo de uso de `<object>` con `<audio>`. Si ninguna de las opciones ofrecidas en `<source>` es soportada por el navegador entonces intentará embeber el reproductor *player_mp3_mini.swf* (éste no es el único reproductor Flash, hay muchos más disponibles en Internet).

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>

<audio controls="controls" preload="auto" >
  <source src="Aretha.mp3" type="audio/mpeg" />
  <source src="Aretha.ogg" type="audio/ogg" />
  <object type="application/x-shockwave-flash"
    data="player_mp3_mini.swf" width="200" height="20">
    <param name="movie" value="player_mp3_mini.swf" />
    <param name="bgcolor" value="#085c68" />
    <param name="FlashVars" value="mp3=Aretha.mp3" />
    <embed href="player_mp3_mini.swf" bgcolor="#085c68" width="200"
      height="20" name="movie" align=""
      type="application/x-shockwave-flash" flashvars="mp3=Aretha.mp3">
  </embed>
</object>
</audio>
</body>
</html>
```

Para que este código funcione es necesario que el archivo *player_mp3_mini.swf* esté disponible en la misma carpeta (local o en web) que la página HTML. Este archivo es el que carga `<object>` cuando ninguna de las otras opciones son soportadas. `<object>` tiene varios parámetros que definen el tamaño del elemento que quiere embeber, en este caso el reproductor Flash. Sin embargo, es con la etiqueta `<embed>` con la que realmente se está incluyendo el archivo *mp3* que se quiere cargar en ese reproductor. De alguna manera, con este código se pretende insertar un audio *mp3* con la etiqueta `<embed>` vista anteriormente, pero añadiendo adrede el reproductor, y no suponer que el navegador lo tiene.

La etiqueta `<object>` es muy conocida en HTML4 por lo que no es objetivo de este libro tratarla en profundidad. Sin embargo, si es interesante resaltar que los valores de los atributos coincidentes entre `<object>` y `<embed>` deben tener los mismos valores para garantizar un funcionamiento adecuado.

A la etiqueta `<audio>` también se le puede aplicar estilos CSS. Por ejemplo, el siguiente código define una estilo para hacer que una etiqueta audio que use la clase `audio-fondo` aparezca con una color de fondo amarillo y una tamaño determinado.

```
<style>
  .audio-fondo {
    width: 160px;
    height: 36px;
    background: #FFFF00;
  }
</style>
```

A modo de conclusión de los visto, aunque no todos los navegadores actuales soportan HTML5 con la madurez deseada, y no todos los usuarios usan los navegadores debidamente actualizados para garantizar que soportan HTML5, la etiqueta `<audio>` ofrece muchas ventajas que aconseja su uso futuro a la hora de insertar audio en sitios web. HTML5 permite que los navegadores yo no necesiten complementos (*plugins*), del estilo Adobe Flash, Microsoft Silverlight, para reproducir audio y eso le facilita el desarrollador no tener que preocuparse si un navegador reproduce o no un audio determinado.

Los dispositivos móviles actuales integran navegadores que soportan HTML5 por lo que es cuestión de poco tiempo que todos los usuarios tengan navegadores HTML5 y no sea necesario buscar alternativas con `<object>` o *JavaScript*⁴² para detectar que audio puede reproducir un complemento determinado de un navegador concreto. Aunque será inevitable que en la etiqueta `<audio>` se den opciones variadas respecto al tipo de archivo (*mp3, ogg*).

Por último, como opción de diseño avanzada, se puede usar *JavaScript* y HTML5 para personalizar el reproductor. El siguiente código utiliza botones para reproducir, en vez de un *player*. Esta opción da mucho juego en el diseño del reproductor al poder cambiar los botones por imágenes acordes con el estilo del sitio.

```
<!DOCTYPE html>
<html>
<head>

</head>
<body>
<audio id="player" src="Aretha.mp3"></audio>
<div>
<button onclick="document.getElementById('player').play()">Play</button>
<button onclick="document.getElementById('player').pause()">Pausa</button>
<button onclick="document.getElementById('player').volume+=0.1">+ Volumen</button>
<button onclick="document.getElementById('player').volume-=0.1">- Volumen</button>
</div>
</body>
</html>
```

⁴² En JavaScript existen librerías, como `audio.js` (<http://kolberg.github.com/audiojs/>), que permite reproducir audio en todos los navegadores como con la etiqueta `<audio>` aunque no soporten HTML5.

ACTIVIDADES 3.2



► Modifique el esqueleto del Restaurante para incluir en la parte inferior derecha de la cabecera un reproductor cuyos botones sean imágenes estáticas. Las funciones deben ser *play*, *stop*, subir volumen y bajar volumen y deben funcionar en Chrome 14 o superior (*mp3*) y en Firefox 7 o superior (*ogg*). Cuando el usuario presione una de las imágenes se ejecutará la función deseada. La clave de esta actividad son dos:

- Colocar con CSS las imágenes (controles) en la posición deseada sin modificar las capas existentes. Para estar prevenidos de un posible cambio en el tamaño del navegador, la caja con los controles no deberá ser absoluta.
- Asociar a las imágenes la funcionalidad para reproducir.

La Figura 3.3 muestra un posible resultado en Google Chrome:

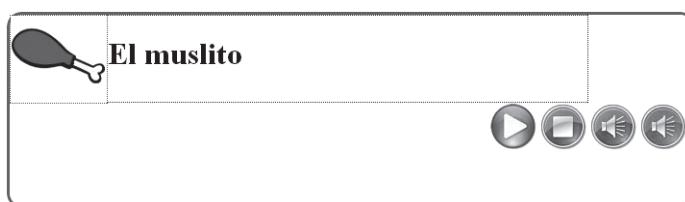


Figura 3.3. Reproductor con botones en la web del Restaurante

SOLUCIÓN

Las líneas que se deben incluir al ejemplo de la Actividad 2.7 son las siguientes:

CSS

```
div.reproductor /*Se define el reproductor a la derecha*/
{
margin-left:500px;
margin-bottom:inherit;
clear:left;
position:realitive;
}
```

HTML

```
<div class="reproductor">
<audio id="player">
<source src="Aretha.mp3" type="audio/mpeg" />
<source src="Aretha.ogg" type="audio/ogg" />
</audio>
```

```
</img>
</img>
 </img>
 </img>
</div>
```

3.4.4 CONSIDERACIONES PARA EL USO DE AUDIO EN UN SITIO WEB

En la sección anterior se han mostrado diferentes alternativas para insertar audio con HTML, con la idea de hacer sitios web más funcionales y “atractivos”. El adjetivo “atractivo” aparece entre comillas para indicar que su semántica es subjetiva y que no siempre una página web con audio es más atractiva que otra que no lo tenga. El audio tiene varias ventajas que cualquier usuario de Internet conoce: pueden hacer sitios web más accesibles, más sorprendentes, explican cosas que de otra manera no se puede, etc.

En general, el audio en sitios web se puede usar de dos maneras:

- Como contenido y por tanto tiene un lugar en un sitio web: Por ejemplo, un sitio web de una cadena de radio necesita insertar sonido para que los usuarios pueda acceder a programas de días anteriores.
- Como elemento decorativo, por ejemplo, música de fondo.

El primer uso es inevitable y está asociado con el objetivo del sitio web, por lo que debe implementarse para que pueda ser interpretado por la mayor cantidad de navegadores.

Sin embargo, el segundo uso es meramente ornamental, no imprescindible, con el que se puede conseguir un efecto positivo (por ejemplo, llamar la atención del usuario) o negativo, contrario al que se espera de él: Un ejemplo clásico del audio como elemento decorativo negativo es cuando se usa como fondo. Actualmente su uso está “pasado de moda”, los diseños actuales no suelen usar música de fondo, ya que en muchos casos son una molestia para los usuarios (por ejemplo, la música aparece cuando el volumen de los altavoces está en los más alto). En cualquier caso, si aparece sonido de fondo, siempre debe existir la posibilidad de que el usuario lo apague (y/o inicie) cuando quiera, por lo que deberían aparecer los controles del reproductor cuando se muestre.

En conclusión, los audios son archivos con un tamaño considerable. Por lo tanto, su uso se debe restringir para cuando es imprescindible, y siempre optimizando su reproducción (jugando, por ejemplo, con *preload* de la etiqueta `<audio>`).

3.5 VÍDEO: INSERTAR VÍDEO EN UNA WEB

3.5.1 FORMATOS

Mucho de lo visto en la sección anterior respecto a los formatos de audio es aplicable a vídeo. El vídeo tiene por un lado el formato de los archivos (*avi*, *mpeg*, etc.) y por otro los códec con los que se comprime (codifica) y descomprimen (decodifica) y que puede haber varios para un mismo formato. Este proceso de compresión tiene más sentido en vídeo que en audio ya que no es posible difundir vídeo por Internet si éste no está comprimido (debido al espacio que ocuparía).

A continuación se hace un resumen de los formatos más conocidos en Internet y los códec asociados a ellos. Estos formatos son en realidad formatos contenedores, es decir, un tipo de formato de archivo que almacena información de vídeo, audio, subtítulos, capítulos, meta-datos e información de sincronización siguiendo un formato preestablecido en su especificación:

- **.mp4**: es un formato que puede contener vídeo en formato MPEG-4, es decir el creado con los códec: DivX, Xvid, QuickTime y H.264, entre otros.
- **.swf**, **.flv** y **.f4v**: son los sucesivos formatos que Adobe ha ido definiendo para el Flash vídeo desde 2002 hasta ahora. Las últimas versiones soportan los códec Sorenson Spark, VP6 y H.264. El .f4v solo soporta H.264.
- **.ogg** y **.ogv**: *ogg* es el correspondiente contenedor Open Source de la Fundación Xiph.Org. Apropiado para contener el formato Theora.
- **.mkv** (Matroska): es un formato Open Source que puede contener casi cualquier tipo de formato de vídeo. Muy usado originalmente para comprimir películas que se han de compartir por Internet.
- **.webm** (WebM): es un contenedor de vídeo Open Source desarrollado por Google, muy dirigido para usarse con HTML5. Está compuesto por el códec VP8 y el códec de audio Vorbis (*ogg*) dentro de un contenedor multimedia Matroska.
- **.avi**: formato contenedor propietario de Microsoft. Además de los formatos de Microsoft soporta otros muchos, entre ellos MPEG-4. No tiene un uso muy extendido en páginas web.
- **.mov**: es el fichero contenedor de QuickTime propietario de Apple. En realidad es casi idéntico a *.mp4*, ya que el MPEG se basó en QuickTime para definirlo; Pero al ser propietario de Apple tiene menos soporte en otras plataformas.

3.5.2 CONVERSIÓN DE FORMATOS

Como ocurre con el audio, existen muchas aplicaciones que permiten convertir de un formato de vídeo. En el desarrollo web, estas aplicaciones ofrecen una alternativa sencilla para poder dar soluciones flexibles para la mayor cantidad posible de navegadores.

Al igual que con el audio, existe una “batalla” entre fabricantes de navegadores sobre qué formato de vídeo debe de ser el estándar y por supuesto no todos reproducen los mismos formatos de forma nativa. Al usar HTML5 ocurre igual que con los ficheros de audio, unos navegadores contemplan de manera nativa unos códec y otros apuestan por otros, y las diferencias son irreconciliables. La siguiente tabla muestra los códec en los principales navegadores:

Tabla 3.2 Formatos de vídeo y navegadores

Códec	Tipo	IE 9	Firefox 7	Chrome 14.0	Safari 5	Opera 11.5
Ogg Theora	Libre	No	Sí	Sí	No	Sí
H.264	Propietario	Sí	No	No	Sí	No
VP8	Libre	No	Sí	Sí	No	Sí

Los formatos más usados con HTML5 (y en los ejemplos siguientes) usan los siguientes códec para vídeo y audio:

$$mp4 = H.264 \text{ (vídeo)} + AAC \text{ (audio)}$$

$$ogg/ogv = Theora \text{ (vídeo)} + Vorbis \text{ (audio)}$$

$$webm = VP8 \text{ (vídeo)} + Vorbis \text{ (audio)}$$

De esta tabla, se puede interpretar que, por ejemplo, *ogv* es un formato adecuado para Firefox 7, pero que IE Explorer 9 no lo soporta, y le va mejor un formato *mp4* como lo hace Safari 5.

Herramientas de conversión hay muchas disponibles en Internet. Algunas gratuitas son:

- **Miro Video Converter**⁴³: es una utilidad muy sencilla para convertir a cualquier formato de vídeo. Incluye *ogv* y *webm*.
- **Free Studio**⁴⁴: es una potente herramienta para convertir todo tipo de archivos (vídeo, audio, imagen). Es fácil de usar y muy adecuada para un diseñador que trabaja con audio y vídeo en la web. Solo versión para Windows. La versión 5.2.1 y anteriores no soporta conversión a *ogv*.
- **AtubeCatcher**⁴⁵: es una utilidad muy interesante. Además de convertir archivos a una gran cantidad de formatos, permite descargar (en varios formatos) vídeos de los repositorios más conocidos (Youtube, Google Videos o Vimeo).

⁴³ <http://www.mirovideoconverter.com/>

⁴⁴ <http://www.dvdvideosoft.com/es/free-dvd-video-software.htm>

⁴⁵ <http://atube-catcher.dsnetwb.com>

3.5.3 INSERTAR VÍDEO EN UNA WEB

Para aprender a insertar vídeo en páginas web se empezará por las posibilidades de HTML5. Tal y como se vio en la Sección 3.4.3, HTML5 ofrece soluciones mucho más ventajosas para insertar audio que su predecesor HTML4. Y para el caso del vídeo ocurre lo mismo. En HTML5 hay una etiqueta <video> que permite embeber archivos de vídeo de forma nativa sin necesidad de complementos (*plugins*) adicionales.

La etiqueta <video> es muy parecido a <audio>: dispone de los atributos *autoplay*, *loop* y *preload*, con la misma sintaxis y semántica que en <audio>. También se puede especificar la fuente de un archivo bien usando el atributo *src* o bien usando la etiqueta <source>. Así mismo, se puede utilizar los controles que ofrece el navegador de forma nativa utilizando el atributo *controls* o bien puedes ofrecer tus propios controles en JavaScript (como se mostró al final de la Sección 3.4.3).

Sin embargo, a diferencia del audio, el vídeo necesita un tamaño para reproducirse, tal y como ocurre con las imágenes. Para ello se pueden usar los atributos *height* (alto) y *width* (ancho) con más sentido que en <audio>. Un ejemplo de uso de esta etiqueta con un archivo *mp4* es el siguiente:

```
<video poster= "images/portada.png" src="videos/recetapollo.mp4" controls width="360"  
height="240"></video>
```

Además, de los atributos vistos, la etiqueta <video> también tiene una atributo *poster* en el que se indica una imagen que se usará como portada antes de que el vídeo empiece a reproducirse.

El ejemplo anterior usa un archivo *mp4* lo que hace que su reproducción sea nativa solo en Safari 5 y no en IE Explorer 9, Firefox 7 ni Google Chrome 14. Entonces, para hacer la reproducción adecuada para la mayor cantidad de navegadores posible, se puede utilizar la etiqueta <source> igual que se hizo en <audio>. En el siguiente ejemplo se ha convertido el formato *mp4* en *ogv* y *webm* con la utilidad *Miro Converter*.

```
<!DOCTYPE html>  
<html><head>  
</head>  
  
<body>  
<video controls="" width="360" height="240">  
<source src="videos/recetapollo.mp4" type="video/mp4"><!-- Safari 5 -->  
<source src="videos/recetapollo.ogv" type="video/ogg; codecs='theora, vorbis'"><!--  
Firefox 5 y Google Chrome ^4 -->  
<source src="videos/recetapollo.webm" type="video/webm"><!-- Firefox 5 y Google Chrome  
14 -->  
</video>  
</body>  
</html>
```

Sin esta solución es solo válida para navegadores que soportan HTML5. Si se desea tener en cuenta a versiones anteriores, por ejemplo, a IE Explorer 8, se puede utilizar etiquetas `<object>` (o usar JavaScript⁴⁶). Esta solución es similar a la vista para audio.

```
<!DOCTYPE html>
<html><head>
</head>

<body>
<video controls="" width="360" height="240">
<source src="videos/recetapollo.mp4" type="video/mp4"><!-- Safari 5 -->
<source src="videos/recetapollo.ogv" type="video/ogg; codecs='theora, vorbis'"><!--
Firefox 5 y Google Chrome º4 -->
<source src="videos/recetapollo.webm" type="video/webm"><!-- Firefox 5 y Google Chrome
14 -->
<object type="application/x-shockwave-flash" data="player_flv_maxi.swf" height="240"
width="360">
<param name="movie" value="player_flv_maxi.swf">
<param name="FlashVars" value="flv=videos/recetapollo.flv">
<embed type="application/x-shockwave-flash" width="360" height="240" src="player_flv_
maxi.swf" flashvars="flv=videos/recetapollo.flv">
</embed>
</object>
</video>

</body>

</html>
```

Para que este código funcione es necesario que el archivo `player_flv_maxi.swf` esté disponible en la misma carpeta (local o en web) que la página `html`. Este archivo es el que carga `<object>` cuando ninguna de las otras opciones son soportadas. `<object>` tiene varios parámetros que definen el tamaño del elemento que quiere embeber, en este caso el reproductor Flash.

⁴⁶ En JavaScript existen librerías, como `video.js` (<http://videojs.com/>), que permiten ejecutar vídeo en cualquier navegador como con la etiqueta `<video>` aunque no soporte HTML5.

Es importante destacar, que no todos los vídeos que se insertan en páginas web deben estar en propiedad del creador. También es posible utilizar vídeos disponibles en cualquier repositorio de vídeo como por ejemplo, *Youtube* o *Vimeo*. Estos repositorios ofrecen código para incrustar un vídeo determinado en una página web. Actualmente, el código proporcionado utiliza la etiqueta <iframe> en sustitución a <object> con el fin de dar soporte más optimizado a páginas web para dispositivos móviles tipos *iPad*.

Como ocurre en audio, también a la etiqueta <video> se le pueden aplicar estilos con CSS.

ACTIVIDADES 3.3



- Modifique el esqueleto del Restaurante para incluir en la parte central un vídeo con, por ejemplo, una receta de cocina. La solución debe funcionar en Chrome 14 o superior, Safari 5 IExplorer 8 y Firefox 7 superiores. Debajo de éste se debe embeder directamente un vídeo de Vimeo con otra receta.

Utilice una clase en CSS para centrar los vídeos.

La Figura 3.4 muestra un posible resultado en Google Chrome:



Figura 3.4. Reproductor de vídeo local y en Vimeo (iFrame)

SOLUCIÓN

Las líneas que se deben incluir al ejemplo de la Actividad 3.2 son las siguientes:

CSS

```
div.reproductor-video /*Se define el reproductor centrado*/
{margin:0 auto 0 auto; width:390px;}
```

HTML

```
<div id="main">

    <!-- VIDEO ALMACENADO EN EL SERVIDOR -->
    <div class="reproductor-video">
        <video controls="" width="360" height="240">
            <source src="videos/recetapollo.mp4" type="video/mp4"><!-- Safari 5 -->
            <source src="videos/recetpollo.ogv" type="video/ogg; codecs='theora, vorbis'"><!--
Firefox 5 y Google Chrome º4 -->
            <source src="videos/recetpollo.webm" type="video/webm"><!-- Firefox 5 y Google
Chrome 14 -->
        <object type="application/x-shockwave-flash" data="player_flv_maxi.swf"
height="240" width="360">
            <param name="movie" value="player_flv_maxi.swf">
            <param name="FlashVars" value="flv=videos/recetpollo.flv">
            <embed type="application/x-shockwave-flash" width="360" height="240" src="player_
flv_maxi.swf" flashvars="flv=videos/recetpollo.flv">
        </embed>
    </object>
</video>

    <!-- VIDEO con IFRAME de VIMEO -->
    <iframe src="http://player.vimeo.com/video/868018?title=0&byline=0&portrait=0"
width="360" height="240" frameborder="0" webkitAllowFullScreen allowFullScreen>
</iframe>

</div>
</div>
```

3.6 ANIMACIONES

Uno de los aspectos más vistosos y atractivos de un sitio web es la inclusión de animaciones. *Una animación* (según Wikipedia) es un proceso utilizado para dar la sensación de movimiento a imágenes o dibujos. Para el caso concreto del diseño web, se puede completar esta definición añadiendo, además de imágenes, textos y audios que pueden visualizarse directamente desde un navegador, dando todo ello sensación de movimiento (animando la web).

El objetivo de esa sección es conocer la manera de incluir y generar animaciones para un sitio web, mostrando la tecnología y las tendencias actuales en este campo, pero sin profundizar en el uso de una herramienta concreta.

Las animaciones web se usan para muchos fines: para mostrar un contenido (por ejemplo, un Sistema solar), para publicidad, como *banners*, detalles de diseño (por ejemplo, marcos o cenefas), efectos, objetos animados (por ejemplo, botones y opciones e menú). Actualmente la web ofrece muchas alternativas para la creación de estas animaciones. Algunas de esas alternativas son de pago y otras son libres, algunas son implementadas por todos los navegadores, otros son exclusivas de unos pocos, algunas siguen el estándar W3C y otras no. La variedad es mucha, y el desarrollador tiene que elegir en cada momento la solución que mejor se adapte a sus necesidades, a la tecnología empleada en el desarrollo del sitio web y a la tecnología que permiten los navegadores.

El desarrollador y diseñador de un sitio web debe conocer todas las alternativas actuales para la creación de animaciones para sitios web, sin embargo, con eso no se está diciendo que el diseñador web (desde una perspectiva del desarrollo) tenga que ser necesariamente creativo, ni un artista del diseño digital. Muchas veces ocurre que un desarrollador sabe manejar una herramienta del tipo Adobe Flash, pero eso no quiere decir que lo que diseña con ella sea atractivo.

3.6.1 FORMATOS DE ANIMACIONES

Desde el punto de vista de los formatos, muchas son las posibilidades para incluir animaciones. Las más lógicas con los formatos de vídeo mostrador en la Sección 3.5, *mp4*, *ogv*, *webm*, *avi*, etc. Los vídeos no son otra cosa que una secuencia de imágenes estáticas con o sin audio. Por lo tanto, en sí son animaciones.

Pero además de los formatos de vídeo, hay otros formatos más específicos que siguen la misma filosofía que el vídeo, pero que tiene otra finalidad.

Gifs animados

Un *gif* animado es un formato creado en 1987. Consiste, en una serie de imágenes, en formato *gif* que están colocados consecutivamente y se muestran en pantalla durante un intervalo de tiempo determinado. Esta secuencia se suele hacer en modo bucle (*loop*) para se repita indefinidamente.

Las ventajas de un *gif* animado es su rápida descarga, nitidez, que permite usar transparencias, y que se puede insertar en html como cualquier otra imagen. Pero tiene como desventaja que las imágenes deben tener un número fijo de colores (un máximo de 256). Al tratarse de un formato de mapa de bits si la animación tiene muchas imágenes estáticas el tamaño del fichero resultante puede ser excesivo para que sea práctico.

En la red existen muchos sitios web que ofrecen *gifs* animados ya creados. Y también hay herramientas para crearlos y convertirlos a otros formatos (*.avi*, *.mpeg*, *.swf*, etc.). Algunas son las siguientes:

- **GIF Construction Set Professional**⁴⁷: es una herramienta de escritorio muy utilizada por su gran funcional. Permite crear *gifs* y conversión entre diferentes formatos.
- **GIMP**⁴⁸: una herramienta de diseño gráfico Open Source (en el segmento de Adobe Photoshop) permite, además de editar todo tipo de imágenes, crear *gifs* animados poniendo capa imagen estática en una capa diferente.
- **Picasion**⁴⁹: es una herramienta *on line* (Web 2.0) que permite crear *gif* animados a partir de un máximo de 10 imágenes. Su funcionalidad es muy limitada, pero permite poner la velocidad de transmisión y la calidad final. Además, las imágenes pueden ser cogidas de repositorios de fotos como *Flickr.com*). Otra herramienta similar es la proporcionada con GifMake⁵⁰.

Archivos Flash: *swf* (*fla*)

Durante más de una década, *.swf* ha sido el formato más atractivo de la web (los archivos *.fla* son los archivos editables). Tanto es que muchas webs han sido creadas con solo animaciones Flash (*.swf*). Los archivos flash (*.swf*) son creados, inicialmente, con la herramienta Adobe Flash (antes Macromedia Flash), aunque existen otras herramientas que también pueden crearlos, pero con menos funcionalidad.

La principal ventaja de *.swf* es que es un formato vectorial (ocupa poco espacio) e integra tanto imagen, texto, vídeo (*Flash video*) y audio. Además, incluye un lenguaje de programación *ActionScript* que hace que estas animaciones puedan interactuar, al reconocer, por ejemplo, eventos de ratón).

Actualmente, el uso de este formato en la web está viéndose reducido por dos razones principales:

- La dificultad de que el contenido de los *.swf* puedan ser indexados en su totalidad por los motores de búsqueda. Esto, desde el punto de vista SEO⁵¹ es una limitación importante.
- Por otro lado, desde el auge del iPad el uso de *.swf* en páginas web se ha reducido considerablemente. El motivo es que, hasta la fecha, Apple se niega a introducir tecnología Flash en sus dispositivos (iPhone, iPad y iPod). Esto hace que los navegadores de estos dispositivos (Safari, por ejemplo) no puedan visualizar archivos *.swf*, con el consiguiente quebradero de cabeza para los diseñadores. Como más adelante veremos, la solución adoptada antes este problema de intereses empresariales, ha sido la aparición de HTML5 y *conversores de swf a HTML5*.

⁴⁷ <http://www.mindworkshop.com/gifcon.html>

⁴⁸ <http://gimp-win.sourceforge.net/>

⁴⁹ <http://www.picision.com/>

⁵⁰ <http://gifmake.com/>

⁵¹ SEO, *Search Engine Optimization*. Posicionamiento de buscadores. Ver más en http://es.wikipedia.org/wiki/Posicionamiento_en_buscadores

Algunas herramientas disponibles para crear *.swf* son las siguientes:

- **Adobe Flash Professional**⁵²: y, en general, todos los productos para el desarrollo web de la familia Adobe permiten editar (*.fla*) y crear (*.swf*) con toda la funcionalidad.
- **SWF Easy**⁵³: es una herramienta sencilla con gran funcionalidad para crear archivos *.swf*. Permite textos e imágenes estáticas, y además, la posibilidad de usar Action Script 2.0 para permitir interacción.

Shockwave

Es un formato muy popular para presentar contenidos animados más complejos que con Flash (juegos, entre otros). Realmente, ambos fueron creados por la misma compañía (Macromedia) aunque ahora están bajo el nombre de Adobe. Shockwave permite crear aplicaciones más complejas que con Flash. Sin embargo, tiene como contrapartida que el software Adobe para crear shockwave es más caro que el que se usa para Flash. Además, a la hora de insertarlo en una web, Flash ofrece soluciones más extendidas.

Vistos estos formatos, es importante ahora resumir cuándo es preferible usar archivos Flash (*shockwave*) o *gifs* animados.

- **Escenario 1:** animaciones con mucho movimiento de solo algunos objetos (o elementos) de la escena. En este caso es preferible Flash (*.swf*) ya que las herramientas ofrecen muchas posibilidades sencillas que si se quisieran hacer con *gifs* animados sería muy complicado.
- **Escenario 2:** para animaciones a gran escala, con muchos elementos. El tamaño en Flash ocupa poco espacio al ser vectorial (salvo si se usan imágenes tipo mapa de bits en la animación), por ello, poner animaciones a pantalla completa para ver los detalles sería muy sencillo con las herramientas para crear Flash. Hacer lo mismo con *gifs* animados sería muy poco práctico.
- **Escenario 3:** para logotipos es preferible Flash, ya que con solo los efectos que permite se pueden hacer logotipos muy atractivos y creativos. Con un *gif* animado, simular muchos de esos efectos daría como resultado un archivo de enorme tamaño.
- **Escenario 4:** si se desea una interacción avanzada entre la animación y el usuario (por ejemplo, efectos cuando el usuario pase el ratón por encima de la animación) es preferible archivos Flash con ActionScript.
- **Escenario 5:** si se desea una animación sencilla, que no tenga interacción, por ejemplo, flechas asociadas a entradas de menú) es preferible *gifs* animados.

3.6.2 HERRAMIENTAS DE PROGRAMACIÓN

Además de los formatos vistos en la sección anterior, existen muchas otras maneras de incluir animaciones en un sitio web usando código de programación. En esta sección se mostrarán dos de las más utilizadas actualmente, aunque existen muchas más, dependiendo del lenguaje que se quiera utilizar para el desarrollo.

⁵² <http://www.adobe.com/es/>

⁵³ <http://www.sothink.com/product/swfeasy/>

HTML y Javascript

HTML dinámico (JavaScript y otros): una primera alternativa para crear animaciones que pueda interactuar con los usuarios en una página web es combinar HTML con, por ejemplo, Javascript. La potencia de Javascript como lenguaje de programación (con eventos), ayuda a crear animaciones fácilmente soportables por los navegadores de manera nativa.

Un caso concreto de este tipo de solución es usar HTML con jQuery. jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto (licencia MIT y GNU v2). jQuery ahorra tiempo y espacio en desarrollo de animaciones comparado con usar JavaScript directamente. En el Capítulo 4 se detalla jQuery para hacer animaciones.

En Internet existen una gran cantidad de tutoriales⁵⁴ y código gratuito⁵⁵ para hacer animaciones con esta tecnología. Además, el Capítulo 4 se centra en ella para implementar la interacción en la web. Un ejemplo de animaciones con jQuery es el siguiente: un visor de fotografías en las que se pueda trabajar como si estuviesen encima de una mesa (moverlas, colocarlas encima, etc.). La captura de la Figura 3.5. ha sido obtenida del blog <http://www.marcofolio.net>.



Figura 3.5. Resultado de jQuery para animaciones

A continuación se muestra un ejemplo de uso de jQuery para hacer una animación. Ésta consiste en que cuando el usuario se coloca con el puntero del ratón encima de una foto, ésta se aparta para dejar ver el texto que hay debajo. También existe la posibilidad de pulsar para acceder a un enlace.

⁵⁴ Un ejemplo de iniciación es el siguiente: <http://mundogeek.net/archivos/2010/04/21/tutorial-rapido-de-jquery/>

⁵⁵ Dos ejemplos son: <http://www.creativosonline.org/blog/35-tutoriales-de-animationes-en-jquery.html> y <http://pixaf.com.ar/blog/archives/1223>

Para que este ejemplo funcione es necesario tener descargada la librería jquery-1.2.6.min.js⁵⁶ de jQuery y colocarla en la misma carpeta en la que está la página web.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html>

<head>
    <!-- imagenes sacadas de http://www.menudiet.es/ -->
    <!--Ejemplo adaptado de http://www.incg.nl/blog/2008/hover-block-jquery/ -->

    <script type="text/javascript" src="jquery-1.2.6.min.js"></script>
    <script type="text/javascript">

        $(function() {
            $('ul.hover_block li').hover(function(){ //cuando se baja
                $(this).find('img').animate({top:'182px'}, {queue:false,duration:500});
            }, function(){ //cuando se recoge
                $(this).find('img').animate({top:'0px'}, {queue:false,duration:500});
            });
        });

    </script>
<style media="screen">
    body { background: #666; }
    ul.hover_block { display: block; overflow: hidden; height: 1%; padding-bottom: 15px;
    }
    ul.hover_block li {
        list-style:none;
        float:left;
        background: #fff;
        padding: 10px;
        width:300px; position: relative;
        margin-right: 20px; }

```

⁵⁶ <http://code.google.com/p/jqueryjs/downloads/list>

```
ul.hover_block li a{  
    display: block;  
    position: relative;  
    overflow: hidden;  
    height: 150px;  
    width: 268px;  
    padding: 16px;  
    color: #000;  
    font: 1.6em/1.3 Helvetica, Arial, sans-serif;}  
ul.hover_block li a { text-decoration: none; }  
ul.hover_block li img {  
    position: absolute;  
    top: 0;  
    left: 0;  
    border: 0;      }  
  
</style>  
</head>  
  
<body>  
  
<ul class="hover_block">  
  
    <li><a href="http://misrecetas.com"> Pollo al Chilindrón</a></li>  
    <li><a href="http://misrecetas.com"> Habichuelas con almejas</a></li>  
</ul>  
  
</body>  
  
</html>
```

El resultado se muestra en la Figura 3.6. Ejemplo de animación con jQuery:



Figura 3.6. Ejemplo de animación con jQuery

Este código desliza las imágenes hacia abajo y muestra el texto que hay debajo. Si se desea que se deslicen hacia la derecha, solo habría que cambiar: `animate({top:'182px'})` por `animate({left:'300px'})` (para apartar) y `animate({top:'0px'})` por `animate({left:'0px'})` (para volver a su sitio cuando se quita el puntero de encima) en el código jQuery. En el Capítulo 4 se detalla el uso de estos métodos.

ACTIVIDADES 3.4



- Modifique el esqueleto del Restaurante (código de la Actividad 3.2) para incluir esta animación con un tamaño adecuado. Pista: una distribución en vertical será más apropiada.

HTML5, CSS3 y JavaScript

Como ya se ha comentado al hablar de los archivos `.swf`, Flash ha sido durante más de una década el elemento estrella de la web. Sin embargo, desde la aparición de los dispositivos móviles (concretamente iPad e iPhone) y su rápida extensión, Flash ha perdido fuelle. Esto ha coincidido con la aparición e implementación por parte de los navegadores de HTML5. Este lenguaje, combinado con CSS3 y JavaScript permite crear animaciones tan potentes como Flash. Además, HTML5 tiene como ventaja respecto a Flash que es casi soportado por los navegadores actuales de forma nativa y que la web tiene en un futuro a ser implementado completamente en todos.

De alguna manera, aunque no actualmente, HTML5 dejará de lado a Flash en un futuro no muy lejano, facilitando a los desarrolladores la creación de webs animadas para cualquier navegador.

El uso de HTML5, CSS3 y JavaScript tiene una estructura muy parecida a lo mostrado en el punto anterior. Cualquier desarrollador con conocimientos de estos lenguajes puede escribir directamente el código para animar sus creaciones. Además, si se apoya en la multitud de páginas web disponibles en Internet con trucos para hacer animaciones o en librerías de alto nivel (como jQuery), el resultado es aún más rápido.

Sin embargo, las nuevas tendencias para crear animaciones con estos tres elementos no van por el camino de que el desarrollador escriba todo el código. Lo que muchas aplicaciones proponen son actualmente, y seguramente sea una pauta para el futuro son dos alternativas:

- **Entornos gráficos** (estilo Adobe Flash): para crear animaciones visualmente. Luego, estos entornos permiten guardar (exportar) las animaciones creadas en HTML5, CSS3 y JavaScript. Es decir, de la misma manera que se pueden guardar en un formato determinado (por ejemplo, SWF) se puede generar el código HTML5 correspondiente.

Actualmente grandes empresas como Google o Adobe⁵⁷ ofrecen herramientas gráficas para este fin:

- **Adobe Edge**⁵⁸: que ya se puede descargar tanto para Windows como para Mac, es un programa que sirve para crear animaciones web y otros contenidos interactivos mediante HTML5, CSS3 y Javascript. Tiene una interfaz intuitiva y clara, muy inspirada en Adobe Flash. Permite importar documentos HTML, CSS y soporta los formatos de imagen más populares (SVG, PNG, JPG y GIF). Además tiene una amplia galería de efectos. Esta herramienta no es gratuita.

Cooties (Blue Dojo)⁵⁹: Es una herramienta soportada dentro de las aplicaciones de Google (se necesita cuenta de Google para ser usada) que permite crear animaciones al estilo Adobe Flash. Aunque su funcionalidad no es tan amplia como Adobe Flash, ésta es gratuita y con un compromiso de sus desarrolladores para ampliarse.

- **Hype**⁶⁰: es otra herramienta con similares características destinada a ordenadores Mac OS X. Esta herramienta no es gratuita.

Estas tres herramientas sin duda marcan la tendencia de lo que será programar animaciones HTML5 en un futuro cercano.

- **Sencha**⁶¹: Es otra herramienta similar pero para crear animaciones en CSS3. Está disponible para varios sistemas operativos (Windows, Mac y Linux) pero no es gratuita.

Aplicaciones de conversión de archivos Flash (SWF): a HTML5, CSS3 y JavaScript

Este tipo de aplicaciones de conversión se presentan como herramientas de transición entre archivos Flash y HTML5. Muchas webs actuales con animaciones Flash no puede ser visualizadas correctamente en algunos dispositivos móviles. Por ello, estas herramientas se presentan como solución temporal de cambio entre ambas tecnologías.

- **Wallaby**⁶²: es una aplicación de Adobe para convertir archivos (.fla, Flash editable) a HTML5. De momento esta herramienta no convierte .swf y obliga a que la animación esté en código editable .fla.
- **Swiffy**⁶³: es una utilidad gratuita de Google para convertir .swf a HTML5.

⁵⁷ Parece un poco paradójico que Adobe apueste por una alternativa de Flash cuando es su propio producto. Sin embargo tiene lógica después del, ya comentado, decremento en el uso de archivos Flash a favor del menos problemático HTML5. De alguna manera, Adobe se cubre las espaldas para un futuro que parece inevitable.

⁵⁸ <http://labs.adobe.com/technologies/edge/>

⁵⁹ <http://cooties.bluedojo.com/>

⁶⁰ <http://tumultco.com/hype/>

⁶¹ <http://www.sencha.com/products/animator/>

⁶² <http://labs.adobe.com/technologies/wallaby/>

⁶³ <http://www.google.com/doubleclick/studio/swiffy/>

Todas estas herramientas, como se ha dicho, marcan la tendencia de lo que será el desarrollo de animaciones en un futuro. Sin embargo, aunque puedan parecer una panacea actual, todavía quedan muchas cosas que el diseñador tiene que tener en cuenta a la hora de usarlas, debido al estado tan inicial en el que se encuentra HTML5 en los navegadores.

La mayoría de estas herramientas crean el HTML5 usando el motor de renderizado⁶⁴ *WebKit*. Este motor open source es la base de los navegadores Safari (Apple) y Google Chrome (Google). Por ellos, para estos navegadores estas aplicaciones pueden garantizar que el código HTML5 generado se muestra correctamente. Sin embargo, para todos otros navegadores con otros motores de renderizado, el resultado es más incierto actualmente, aunque en un futuro no muy lejano se solucionará debido a su atractivo empresarial.

Pese a todo, a continuación se muestran algunos ejemplos de animaciones hechas con tecnología HTML5, CSS3 y JavaScript (en algunos casos usando alguna de las herramientas antes mencionadas:

ACTIVIDADES 3.5



► Visualice las siguientes animaciones hechas con HTML5, CSS3 y JavaScript en los navegadores más conocidos en sus últimas versiones: IExplorer, Firefox, Google Chrome, Safari y Opera.

¿Se puede ver en todos directamente y de la misma manera? ¿Alguno necesita usar el *plugin* de Flash, ya que no soporta ese HTML5?

<http://bomomo.com/>

<http://tumultco.com/hype/gallery/BirthdayParty/BirthdayParty.html>

<http://tumultco.com/hype/gallery/WooGrit2/WooGrit2.html>

<http://peterned.home.xs4all.nl/3d/>

<http://www.openrise.com/lab/FlowerPower/>

⁶⁴ Un *motor de renderizado* de un navegador es el encargado de coger el contenido de una página (HTML, imágenes, etc.) y el formato (CSS, XSL, etc.) y visualizar el resultado en pantalla. Algunos motores son Gecko (Firefox), Presto (Opera) o webKit (Chrome y Safari). No todos trabajan de la misma manera, por lo que algunas tecnologías referentes a animaciones son específicas para usar un determinado motor de renderizado.

3.7

CONCLUSIÓN Y PROPUESTAS PARA AMPLIAR

En este capítulo se han mostrado cómo trabajar con elementos multimedia en diseño web (imágenes, audio, vídeo y animaciones). Comprender bien las actividades propuestas y los ejemplos de cada sección es clave para que el diseñador pueda incorporar elementos multimedia en su web, con garantías de que se visualizará en todos los navegadores de la misma manera.

Terminado el capítulo, el lector debe entender que no ha sido posible sintetizar todo en estas páginas. Las tecnologías para incluir multimedia en la web son muy variadas y muy cambiantes. Lo que se ha pretendido ha sido dar una visión actual de cuáles son las tendencias en esta línea. Sin embargo, muchas cosas se han quedado en el tintero ya que es imposible entrar a detallar cómo se utiliza o implementa cada tecnología.

Si el lector está interesado en profundizar más en el tema y especializarse en una tecnología en particular, se recomienda que lo haga en lo relativo al uso de HTML5, CSS3 y JavaScript (jQuery) para desarrollar animaciones. El capítulo siguiente trata en exclusiva sobre jQuery. Además, la web tiene una gran cantidad de tutoriales y bibliografía relacionada que ayudará al lector a profundizar en este interesante campo de aplicación con una gran proyección futura.



RESUMEN DEL CAPÍTULO

En este capítulo se ha hecho una introducción a la inclusión de elementos multimedia en un sitio web. Aunque muchas son las tecnologías disponibles para hacer esta labor, el capítulo se ha centrado en el uso de HTML5 principalmente. Esto es debido a que en un futuro no muy lejano esta será la tecnología más extendida.

El capítulo no pretende mostrar cómo se utilizan todas las tecnologías actuales (eso daría un libro de cientos de páginas). Sin embargo, si pretende dar al lector una muestra de las ventajas y desventajas entre las tecnologías más extendidas, dejando en su mano el profundizar en una o varias en particular.



EJERCICIOS PROPUESTOS



- **1.** Busque en Internet sitios web que incluyan algún tipo de audio o vídeo y analice (con su código fuente) qué técnica ha utilizado para incluirlo en la web.
- **2.** Descargue un vídeo de Youtube, conviértalo en formatos soportados por las últimas versiones de los navegadores Safari, IExplorer, Chrome y Firefox, e inclúyalo en una página web. Compruebe que todos los navegadores web muestran el vídeo adecuadamente.
- **3.** Busque en alguno de los repositorios de imágenes mostrados en el capítulo una imagen con licencia *Creative Commons* de tipo Reconocimiento - No copias derivadas. Incorpórela en la web creada en el ejercicio anterior con los elementos adecuados que exige la licencia. ¿Es correcto legal y moralmente lo que ha hecho?
- **4.** Busque en alguno de los repositorios de imágenes mostrados en el capítulo una imagen con licencia *Creative Commons* de tipo Reconocimiento - No comercial. Modifique alguna característica de la imagen con algún editor gráfico de los vistos en el capítulo. Convierta la imagen a formatos .jpg, .png y .gif e insértelas en la página web creada en el ejercicio anterior. ¿Es correcto legal y moralmente lo que ha hecho?
- **5.** En parejas, utilizando como base la página creada en el ejercicio 5 del Capítulo 2, insertad un vídeo y dos imágenes estáticas que cumplan que: (1) todo sea visible por los navegadores actuales más conocidos, y, (2) que estén en consonancia con el diseño del portal y lo que representa. Una vez hecho, enumere las dificultades técnicas encontradas para hacer esta actividad.



TEST DE CONOCIMIENTOS



1 Respecto a las licencias de software:

- a) *Creative Commons* es la implementación del copyleft anglosajón.
- b) *Copyright* atiende a los derechos morales de los autores.
- c) *Creative Commons* permite diferentes modelos de licencias que diferentes niveles de restricción.

2 Respecto a las imágenes en la web:

- a) El tamaño de una imagen solo depende de su resolución. Una imagen con más resolución que otra tiene siempre un tamaño mayor.
- b) Una imagen con degradados complejos es mejor comprimirla en formato GIF.
- c) En la Web lo importante es que las imágenes se vean bien y se carguen rápido.

3 Los formatos de audio son importantes por no todos los navegadores soportan por defecto todos ellos, así:

- a) .ogg es un formato que no soporta Internet Explorer 9.
- b) .mp3 es un formato que solo soporta Safari y Chrome.
- c) Si se usa .ogg y .mp3 se garantiza que lo leerá cualquier navegador.

4 A la hora de insertar (no enlazar) vídeo en una web:

- a) El formato no importa, no ocurre lo mismo que con los formatos de audio. Todos los navegadores leen todos los formatos.
- b) Utilizar la etiqueta <video> es lo más recomendable, incorporando los formatos más extendidos y un reproductor Flash para versiones obsoletas.
- c) La etiqueta <video> en sí misma hace que todos los vídeos se vean en todas los navegadores, al convertir los formatos a mp4.

5 Respecto a las animaciones:

- a) En la Web predominan las animaciones Flash, aunque con HTML5 se empiezan a utilizar menos ya que se ofrece como una alternativa.
- b) El problema de Flash para animaciones es que no permite hacer efectos tan vistosos HTML5.
- c) Flash es siempre la mejor alternativa para hacer animaciones para un sitio web.

6 El multimedia (audio, vídeo, animaciones) disponible en Internet tiene ventajas como:

- d) Al estar en Internet es libre y se puede usar como se quiera.
- e) No tiene limitaciones técnicas. Si está en la web, cualquier navegador lo puede ver.
- f) Tiene limitaciones técnicas y legales que hay que conocer antes de incorporarlo en un sitio web profesional.

4

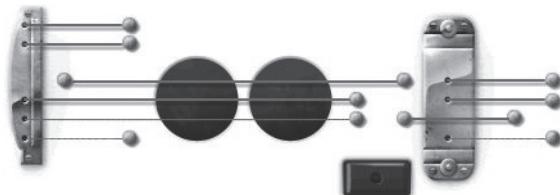
Integración de contenido interactivo

OBJETIVOS DEL CAPÍTULO

- ✓ Identificar las necesidades actuales respecto al contenido interactivo.
- ✓ Añadir animaciones a una página web.
- ✓ Añadir interactividad a los diseños web.
- ✓ Desarrollar y agregar animaciones para distintos navegadores.
- ✓ Verificar el funcionamiento de animaciones en diferentes navegadores.

4.1 ELEMENTOS INTERACTIVOS BÁSICOS Y AVANZADOS

La interacción es, junto a la inclusión de elementos multimedia, otra de las claves del desarrollo web. La interacción, según Wikipedia, es *el proceso que establece un usuario con un dispositivo, sistema u objeto determinado. Es una acción recíproca entre el elemento con el que se interacciona y el usuario*. Para entender bien esta idea lo mejor es interactuar con la guitarra que desarrolló Google Logos en honor al nacimiento de Les Paul⁶⁵ y que la Figura 4.1 muestra con una captura estática.



[Back to Google Logos](#)

Figura 4.1. Guitarra interactiva Google Logos (les Paul)

Esta Logo interactivo fue desarrollado con HTML5, CSS3 y JavaScript, aunque tenga la apariencia de un archivo .swf hecho con Adobe Flash.

Al igual que ocurría en el capítulo anterior con los elementos multimedia, la interacción en un futuro inmediato pasa por los lenguajes antes citados: HTML5, CSS3 y JavaScript. Los archivos Flash seguirán teniendo su protagonismo, pero cada vez en menor medida⁶⁶.

En ese capítulo nos centraremos en la interacción de objetos con jQuery, usando inevitablemente HTML5 y CSS3 (en la sección 3.6.2 ya se han visto algunos ejemplos de animaciones con este lenguaje). jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es un software libre y de código abierto (licencia MIT y GNU v2) y su misión es ahorrar tiempo y espacio en desarrollo de animaciones comparado con el uso de JavaScript directamente.

jQuery es un lenguaje muy extenso, pero este capítulo se centrará únicamente en lo más relacionado con las interfaces de usuario y la interacción, aunque es inevitable empezar con los conceptos básicos del lenguaje. Para un correcto aprendizaje de lo mostrado en el capítulo, el lector debe estar familiarizado con la tecnología actual de desarrollo web, por ejemplo, XML y DOM.

⁶⁵ Se puede interactuar con esta guitarra en <http://www.google.com/logos/2011/lespaul.html>

⁶⁶ En la sección 3.7 se habla más profundamente sobre Flash vs HTML5 y el futuro de la web.

4.1.1 INTRODUCCIÓN A JQUERY

Para utilizar jQuery, solamente es necesario descargar la librería⁶⁷ y enlazarla en la página HTML de la siguiente manera:

```
<script type="text/javascript" src="jquery.js"></script>
```

La librería se puede descargar en dos versiones: comprimida y descomprimida. La primera es más adecuada para que la descargue el cliente ya que ocupa menos espacio. La segunda es más adecuada cuando se está trabajando en el desarrollo.

Una de las primeras instrucciones que todo código jQuery debe tener es:

```
$(document).ready( function() { ... } );
```

Esta instrucción indica que el código jQuery se ejecute cuando el código HTML se ha cargado completamente y el árbol DOM se ha generado⁶⁸. Esto debe ser así ya que jQuery trabaja con los elementos que hay en la página, referenciados a través del DOM. Si los objetos no se han cargado no se debe poder ejecutar ningún código jQuery ya que daría un error.

La función más usada en jQuery es \$().⁶⁹ Esta función se utiliza para seleccionar un elemento del árbol DOM a través del identificador, a través del nombre de la clase o de la etiqueta (CSS). Por ejemplo:

Seleccionar un elemento con id=#miIdentificador:

```
$("#miIdentificador");
```

Seleccionar todos los elementos <a> (enlaces) de un HTML:

```
 $("a");
```

Se pueden seleccionar varios al mismo tiempo, separados por coma:

```
$("#miParrafo, a");
```

Seleccionar mediante el nombre de la clase CSS:

```
 $(".miClase");
```

En el ejemplo mostrado en la sección 3.6.2, se selecciona los elementos ‘ul.hover_block li’ como selectores CSS, mediante el siguiente código:

```
 $('ul.hover_block li').hover(.....
```

⁶⁷ Por ejemplo de <http://blog.jquery.com/2011/09/12/jquery-1-6-4-released/>

⁶⁸ Este comando no espera a que se carguen las imágenes, con los elementos definidos en el HTML es suficiente.

⁶⁹ Esta función es equivalente al `getElementsByName()` y `getElementsByName()` y `getElementById` de JavaScript.

También se pueden seleccionar los elementos que tenga un cierto atributo. Por ejemplo, todos los elementos <a> con atributo *rel*.

```
$( "a[rel]" );
```

Esta consulta se puede completar pidiendo aquellos con un cierto valor para un atributo. Por ejemplo todos los elementos <input> con atributo *type = radio* (es decir, que son radio-buttons)

```
$( "input[@type=radio]" );
```

O incluso solo aquellos radio-buttons que estén seleccionados:

```
$( "input[@type=radio][@checked]" );
```

Además de poder seleccionar elementos por el nombre de las etiquetas o los selectores CSS, también se pueden seleccionar usando el lenguaje XPath. Así, por ejemplo, se pueden seleccionar todos los párrafos del documento con un camino XPath:

```
$( "/html/body//p" );
```

O también elementos que tienen un determinado atributo con un valor. Por ejemplo, los elementos <a> con el atributo *ref='nofollow'*.

```
$( "//a[@ref='nofollow']" );
```

Estos son solo algunos ejemplos de la flexibilidad que da jQuery para seleccionar los elementos sobre los que luego se aplicarán ciertas acciones. Existen más alternativas, incluso jQuery proporciona sus propias etiquetas para simplificar aún más la selección, sin embargo con los anteriores se abarca un amplio abanico de consultas.

4.1.2 MANEJO DE EVENTOS

Además del evento *ready()* usando en el primer ejemplo, jQuery dispone de varias funciones relacionadas con la gestión de los eventos. De hecho, jQuery tiene un modelo de eventos muy completo que facilita la programación de interacciones entre el usuario y los objetos de la página web.

Para cada evento disponible hay dos posibilidades de manejo:

1. Se le puede pasar una función que determine su comportamiento.
2. No se le pasa función, entonces se ejecuta el evento JavaScript del elemento. jQuery tiene tantas funciones como eventos estándar de JavaScript. El nombre de cada función es el mismo que el del evento, pero sin el habitual prefijo “on” de los eventos:

Un ejemplo del primer caso es el siguiente:

```
$( "p" ).click(function() { alert( $(this).text() ) ;});
```

Este código muestra una ventana de alerta cuando se hace clic (*onclick*) en cualquier párrafo del documento. Esta función ha sido definida expresamente para atender este evento.

Un ejemplo del segundo caso es: el siguiente:

```
$(“p”).click();
```

En este caso, se ejecuta la función *click()* que es la asociada por defecto al evento *onclick*.

bind() y unbind()

Un método interesante para trabajar con eventos de manera óptima es *bind()*. Este método permite asociar a un elemento un número ilimitado de eventos, todo de una vez. Esto método facilita la asignación de eventos a los elementos, haciendo un código más legible y una ejecución más óptima.

La sintaxis es:

```
bind(“tipo_de_evento1 tipo evento2 ... tipo evento n”,manejador (handler) )
```

Un ejemplo de uso es el siguiente:

```
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $("p").bind("click mouseenter mouseleave", function(e) {
        if ($(this).css("color") != "rgb(255, 0, 0)")
            $(this).css("color", "rgb(255, 0, 0)");
        else
            $(this).css("color", "rgb(0, 0, 255)");
    })
});
```

Este código averigua para todos los elementos *p* de qué color son cuando el usuario hace clic sobre él (evento *clic*) o entra o sale con el ratón (*mouseenter* y *mouseleave*). Si no es de color *rojo* entonces le asigna ese color y si lo es entonces lo cambia a *azul*.

Además de *bind()*, existe el método contrario *unbind()*, que elimina un evento previamente asignado a un elemento o varios de una página. Si se utiliza *unbind()* sin ningún parámetro se eliminan todos los manejadores de cualquier tipo de evento. Esto es útil cuando se quiere quitar la interacción a unos elementos de una página por cualquier motivo.

```
$(“p”).unbind();
```

Sin embargo, lo más habitual es quitar la atención de un tipo de evento en particular. Por ejemplo, para eliminar el manejo de un evento *onclic* sobre un elemento *p*, se escribiría:

```
$(“p”).unbind(“click”);
```

Para no eliminar todo el manejo de eventos de un tipo determinado se puede especificar la función que se desea descartar en la lista de parámetros de la llamada a `unbind()`. Para ello habría que colocar la función dentro de una variable⁷⁰, para poder referirse a ella como parámetro de `unbind()`. El siguiente ejemplo muestra la definición de la función y cómo se desactiva con `unbind()` la atención al evento `onclic` de `p` al hacer clic sobre el botón con `id="desactivar-evento"`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html>
<head>

<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">

$(document).ready(function() {

    var funcionqueManeja = function() {
        if ($(this).css("color")!="rgb(255, 0, 0)")
            $(this).css("color", "rgb(255, 0, 0)");
        else
            $(this).css("color", "rgb(0, 0, 255)");
    }

    $("p").bind("click mouseenter mouseleave",funcionqueManeja);
    $("#desactivar-evento").bind("click", function(){
        $("p").unbind("click", funcionqueManeja);
    });
});

</script>
<style media="screen">body { background: #666; } </style>
```

⁷⁰ En la sección 4.3 se habla más sobre cómo definir una función.

```
</head>
<body>
<p> El texto cambia de color al entrar o salir con el ratón o hacer click</p>
<input id="desactivar-evento" type=button value="Quitar el cambio de color cuando se
hace click" >
</body>

</html>
```

Se puede observar en el código que, al desactivar el manejo de *click()*, el cambio de color sigue funcionando cuando el puntero entra y sale del párrafo ya que *mouseenter* y *mouseleave* no se han desactivado.

toggle()

Toggle() es otro método muy usado en jQuery. A este método se le pasan dos funciones. En función de las veces que se hace clic sobre el elemento que lo define, se ejecutará la primera función o la segunda. La primera vez que se hace clic sobre el elemento (y todas las veces impares), se ejecuta la primera función y la segunda vez que se hace clic el elemento (y todas las veces pares) se ejecuta la segunda función:

El siguiente ejemplo usa el método *toggle()* para cambiar el color de un párrafo *p* cuando se hace clic. Se puede observar en el código que en este caso las funciones que manejan el evento *onclick()* (*poner_Rojo*, *poner_Azul*) han sido definidas fuera en forma de variable.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">
```

```
<html>
<head>
    <title>Ejemplo uso toggle()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
        var poner_Rojo= function() {$(this).css("color", "rgb(255, 0, 0))"; }
        var poner_Azul= function() {$(this).css("color", "rgb(0, 0, 255))"; }
        $(document).ready(function() {
            $("p").toggle(poner_Rojo,poner_Azul);
        });
    </script>
    <style media="screen">      body { background: #666; }</style>
</head>
<body>
    <p> El texto cambia de color hacer click</p>
</body>

</html>
```

4.2 CAMBIO DE LAS PROPIEDADES DE UN ELEMENTO

Una vez vistos las funciones básicas de jQuery y cómo se seleccionan elementos sobre los que aplicar esas funciones, en esta sección se explicará cómo se pueden cambiar propiedades de los elementos (propiedades definidas principalmente con CSS) así como añadir nuevos elementos y texto.

4.2.1 CAMBIO DE LAS PROPIEDADES CSS

Si se desea obtener el valor de una propiedad para un elemento determinado se utiliza el método `.css()`. El parámetro de este método es el nombre de la propiedad `css` que se desea obtener.

Por ejemplo, si se desea saber con qué color está definido los elementos `<p>` se pondría:

```
$(“p”).css(“color”);
```

Además, `.css()` permite establecer el valor de una propiedad determinada. Por ejemplo, si se desea cambiar el color (propiedad `color`) de los elementos `<p>` se pondría:

```
$(“p”).css(“color”, “red”);
```

El valor del color admite todas las posibilidades que da CSS.

También se pueden establecer varias propiedades a la vez. Por ejemplo, el siguiente código modifica las propiedades CSS (`color, background, font-weight`) de los elementos `<p>`.

```
$(“p”).css({ color: “red”, background: “blue”, font-weight: “bold” });
```

4.2.2 AÑADIR NUEVOS ELEMENTOS

Además de cambiar propiedades, jQuery permite insertar nuevos elementos en un HTML haciéndolo (dinámico).

`.append()` permite insertar un contenido como el último hijo (árbol DOM) de cada elemento de la colección seleccionada. Si lo que se desea es insertarlo como primer hijo, entonces se usa `.prepend()` de la misma manera.

Por ejemplo, si se desea añadir una etiqueta `<p>` dentro de dos `<div>` llamados “misdivs” se pondría:

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type=“text/javascript” src=“jquery-1.6.4.js”></script>
<script type=“text/javascript”>
$(document).ready(function() {
    $(“.misdivs”).bind(“click mouseenter mouseleave”, function(e) {
        $(“.misdivs”).append(“<p><b>Texto</b>insertado en los div</p>”);
    });
});
```

```
        })
    });
</script>
<style media="screen">
    body { background: #666; }
    .misdivs {background: #623; }
</style>
</head>
<body>
    <div class="misdivs">Mi div 1 </div>
    <div class="misdivs">Mi div 2</div>
</body>
</html>
```

De la misma manera que se añade contenido se puede añadir elementos existentes en otra posición del HTML. Por ejemplo, el siguiente código mueve una etiqueta `<h1>` al interior de los `<div>` al hacer clic con el ratón.

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $(".misdivs").bind("click", function(e) {
            $(".misdivs").append($("#h1"));
        })
    });
</script>
<style media="screen">
    body { background: #666; }
    h1{color:#00CC00;}
    .misdivs {background: #623; }
</style>
</head>
<body>
    <h1> Titulo </h1>
    <div class="misdivs">Mi div 1 </div>
    <div class="misdivs">Mi div 2</div>
</body>
</html>
```

4.2.3 AÑADIENDO TEXTO

Para añadir texto a un elemento se utiliza el método `.text()`.

`.text()`. Este método permite obtener o añadir un texto a un elemento determinado (que permita texto). Puede ser utilizado tanto en HTML como en XML⁷¹:

En la forma `.text()` obtiene el texto de un elemento (no es válido para elementos de formularios tipo `<input>` ya que para ello se utiliza `.val()`).

En la forma `.text(cadena_de_texto)` escribe ese texto en el elemento que lo invoca.

Por ejemplo, si se desea escribir dentro de los párrafos `<p>` de un HTML la cadena “`nuevo texto`” se podría:

```
$("p").text("<b>Some</b> new text.");
```

Es importante destacar que el texto, aunque sea HTML no se interpreta como tal, y que aparecería en pantalla el texto tal y como está.

4.2.4 OTROS MÉTODOS

Otros métodos relacionados con propiedades de los elementos son:

- `.innerWidth()` e `.innerHeight()`⁷²: para un elemento devuelve sus dimensiones internas (anchura y altura, respectivamente) contando el *padding* del elemento pero no el borde.
- `.outerWidth()` y `outerHeight()`: devuelve las dimensiones externas (anchura y altura, respectivamente) del elemento contando el *padding* del elemento y su borde.
- `.offset()`⁷³: indica la posición del elemento real, teniendo en cuenta los márgenes del elemento. Lo devuelve en un objeto con dos atributos *top* y *left*.

4.3 EJECUCIÓN DE SECUENCIAS DE FUNCIONES

Antes de entrar de lleno en métodos relacionados directamente con la interacción, en esta sección se trata otro de los aspectos de jQuery más utilizados y que hay que tener en cuenta para una correcta programación: los *callbacks*.

En jQuery cuando se trabaja de cara a la interfaz, es habitual utilizar secuencia de métodos o funciones apilados para lograr efectos más elaborados.

⁷¹ Aunque en este capítulo solo se ha trabajado con HTML, jQuery se puede utilizar también para generar y modificar XML.

⁷² En el API de jQuery se puede encontrar más sobre estos métodos: <http://api.jquery.com/category/dimensions/>

⁷³ En el API de jQuery se puede encontrar más sobre este método: <http://api.jquery.com/offset/>

Por ejemplo, si se desea que unos <div> con identificador “#misdivs” se desvanezca tardando 2 segundos, luego se cambie su color y termine apareciendo de nuevo en 2 segundos, se podría usar la siguiente secuencia:

```
$("#misdivs").fadeOut(2000);
$("#misdivs").css({background: "blue"});
$("#misdivs").fadeIn(2000);
```

Esto mismo se hubiese conseguido haciendo una secuencia de este tipo:

```
$("#misdivs").fadeOut(2000).css({background: "blue"}).fadeIn(2000);
```

Para las dos sintaxis el problema es el mismo: para jQuery todo se hace al mismo tiempo. Por eso, como los efectos de aparecer y desvanecerse tardan 2 segundos, y cambiar la propiedad es casi inmediato, entonces lo que se verá será que se cambia la propiedad (color de fondo azul) y luego aparece y desaparece el <div>.

La solución es utilizar efectos *.delay()* (que se verán en la sección 4.4.4) o utilizar la pila de ejecución de funciones. Aquí se explicará esta segunda opción:

Cualquier método o función en jQuery puede recibir como parámetro la función que se ejecutará después de ejecutar el método o función correspondiente. Esta función se llama “*callback*”. Al usar *callback*, en el ejemplo anterior se puede ejecutar el primer desvanecimiento, y en el *callback* de ese método colocar el cambio de la propiedad con *.css()* y que aparezca de nuevo el elemento. El código sería:

```
$("#micapa").fadeOut(1000, function() {
  $("#micapa").css({'top': 300, 'left':200});
  $("#micapa").fadeIn(1000);
});
```

Sin embargo, también se puede hacer un código más elegante definiendo primero la función y luego invocándola:

```
var ocultar=function () {
  $("#misdivs3").css({background: "green"});
  $("#misdivs3").fadeIn(2000);
};
```

Para llamarla se haría:

```
$("#misdivs3").fadeOut(2000, ocultar);
```

El siguiente código muestra los problemas comentados para el <div> “#misdivs1” y la solución con *callback* definido dentro del propio método “#misdivs2” y definida fuera “#misdivs3”.

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="../jquery-1.6.4.js"></script>
<script type="text/javascript">
  var ocultar=function () {
```

```
$("#misdivs3").css({background: "green"});
$("#misdivs3").fadeIn(2000);
};

$(document).ready(function() {

    $("#misdivs").fadeOut(2000).css({background: "blue"}).fadeIn(2000);
    $("#misdivs2").fadeOut(2000, function() {
        $("#misdivs2").css({background: "red"});
        $("#misdivs2").fadeIn(2000);
    });
    $("#misdivs3").fadeOut(2000, ocultar);
});

</script>
<style media="screen">
    body { background: #666; }
    h1{color:#00CC00;}
    .misdivs {background: #623; }
</style>
</head>
<body>
    <h1> Titulo </h1>
    <div id="misdivs">mis divs</div>
    <div id="misdivs2">mis divs</div>
    <div id="misdivs3">mis divs</div>
</body>
</html>
```

4.4 COMPORTAMIENTO DE LOS ELEMENTOS. EFECTOS VISUALES

Una vez vistos los elementos de jQuery más básicos para una correcta programación, esta sección detalla los métodos asociados con efectos visuales de los elementos. De esta manera se pueden conseguir animaciones modificando el comportamiento de los elementos.

A los efectos que se muestran a continuación se les puede añadir el método *toggle()* visto en la sección 4.1.2 para poder intercalar funciones dependiendo del número de clic hechos sobre un elemento, y conseguir con ello efectos visuales muy comunes en los desarrollos web actuales⁷⁴.

⁷⁴ Los ejemplos mostrados utilizan el evento *click()* para tratar la interacción, sin embargo, en la sección 4.5 se mostrarán el resto de eventos que maneja jQuery.

4.4.1 EFECTOS BÁSICOS

jQuery permite que los sitios web incorporen pequeños efectos visuales y animaciones que, bien empleados, mejoran la interacción y la usabilidad del sitio. Los efectos visuales más extendidos son los siguientes:

```
.show ([duración] [, easing] [, callback])
```

Muestra un elemento que estaba oculto (generalmente con una propiedad CSS). Cuando un elemento se muestra (*show*) y se oculta (*hide*) se queda con los valores originales (de las propiedades CSS) que tenía antes de mostrarse.

- **duración:** una cadena (*slow* o *fast*) o un número (milisegundos) para determinar el tiempo que la animación se ejecutará.
- **easing:** es la función que se usará para el tipo de transición. Por defecto, *show* anima al mostrar y ocultar modificando el alto, ancho y la opacidad del objeto. Para ver más funciones de transición es necesario un *plugin*⁷⁵.
- **callback:** una función para llamar una vez finalizada la animación (tal y como se mostró en el apartado 4.3).

Todos los parámetros de este método son opcionales (por eso están entre []).

Un ejemplo para mostrar todos los párrafos lentamente sería: `$(“p”).show(“slow”);`

```
.hide ([duración] [, easing] [, callback])
```

En esta función los parámetros tienen el mismo significado que en *show()*, pero con la acción contraria, es decir, la de ocultar un elemento.

El siguiente ejemplo muestra el uso de para mostrar textos que luego puede ser ocultado con *.hide()* a una velocidad de 500ms. Hay dos botones, uno para mostrar los y otro para ocultarlos (uno detrás de otro). Los elementos se ocultan con velocidad 500ms. Este ejemplo está inspirado en el mostrado en el API de jQuery⁷⁶. La Figura 4.2 muestra el resultado del código.

```
<html>

<head>

<title>Ejemplo de animación con Show() y Hide()</title>

<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
var ocultar=function () { $(this).prev().hide("fast", ocultar); }
```

⁷⁵ <http://jqueryui.com/> Esta librería implementa efectos específicos para Interfaz de Usuario.

⁷⁶ <http://api.jquery.com/show/>

```
$(document).ready(function() {  
    $("#hidr").click(function () {  
        $("span:last-child").hide("fast", "ocultar");  
    });  
    $("#showr").click(function () {  
        $("span").show(500);  
    });  
});  
  
</script>  
<style media="screen">  
body { background: #666; }  
span { background:#def3ca; padding:3px; float:left; }  
</style>  
  
</head>  
<body>  
    <button id="hidr">Ocultar</button>  
    <button id="showr">Mostrar</button>  
    <div>  
        <span>Había</span>  
        <span>una </span>  
        <span>vez</span>  
        <span>un</span>  
        <span>desarrollador </span>  
        <span>Web</span>  
        <span>...</span>  
    </div>  
</body>  
</html>
```

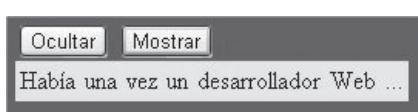


Figura 4.2. Ejemplo de uso de show() y hide()

4.4.2 EFECTOS FADING (OPACIDAD-TRANSPARENCIA DE LOS ELEMENTOS)

Fading son efectos de desvanecimiento de elementos. El *fading* juega con la opacidad y la transparencia para lograr un gran atractivo visual a la hora de mostrar y ocultar elementos.

La sintaxis de los efectos Fading de los que dispone jQuery es la siguiente:

```
.fadeIn ([duración] [, easing] [, callback])
.fadeOut ([duración] [, easing] [, callback])
.fadeToggle ([duración] [, easing] [, callback])
```

Todos los parámetros son opcionales y tienen el mismo significado que los vistos en *show()* y *hide()*.

- *fadeIn()*: muestra un elemento con opacidad gradual (hasta llegar a 1).
- *fadeOut()*: oculta un elemento con opacidad gradual(hasta llegar a 0).
- *fadeToggle()*: al igual que Toggle, alterna el proceso de aparecer y desvanecer según se haga clic sobre el elemento un número par de veces o un número impar.

```
.fadeTo (duración, opacidad [, easing] [, callback])
```

Ese método coloca el elemento a una opacidad determinada (*opacidad*) en un tiempo establecido (*duración* - número que indica milisegundos, *fast* o *slow*). La opacidad toma como valor un valor entre 0 y 1. Tanto *duración* como *opacidad* son parámetros obligatorios en este método.

El siguiente ejemplo⁷⁷ muestra el uso de *fadeOut()*. En un texto se seleccionan las palabras que son *adjetivos*, y al hacer clic sobre ellas, éstas desaparecen y se muestra un mensaje en un <div> (initialmente vacío).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">
<html>
<head>
<title>Ejemplo de animación con FadeOut()</title>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
  $("span").hover(function () { $(this).addClass("hilite"); }, function () { $(this).
removeClass("hilite"); });

  $("span").click(function () {
    $(this).fadeOut(1000, function () {
      $("div").text("'" + $(this).text() + "' ;Se ha desvanecido!");
      $(this).remove();
    });
  });
}); //Function del Click
}); //Function del Click
```

⁷⁷ Código inspirado en los ejemplos del API jQuery: <http://api.jquery.com/fadeOut/>

```

        });
</script>
<style media="screen">
    span { cursor:pointer; }
    span.hilite { background:yellow; }
    div { display:inline; color:red; } /*Se pone inline para que al quitarlo se ocupe su hueco */
</style>

</head>
<body>
<h3>Encuentra los adjetivos <div></div></h3>
<p> La chaqueta <span>roja</span> el lo mejor cuando hace un tiempo <span>cálido</span>
¿verdad?</p>
</body>
</html>

```

Encuentra los adjetivos 'cálido' ¡Se ha desvanecido!

La chaqueta roja el lo mejor cuando hace un tiempo ¿verdad?

Figura 4.3. Ejemplo de uso de `fadeOut()`

La Figura 4.3 muestra el resultado. En el ejemplo la función `$(“span”).hover` activa una nueva clase asociada a `` que lo pone con fondo amarillo al entrar el ratón y lo deja normal (`removeClass`) al sacar el ratón del `` (en la sección 4.5.1 se muestra el evento `hover` más ampliamente). Por su parte, `$(this).fadeOut` desvanece el `` y después rellena el `<div>` con el nombre del `` desvanecido y elimina (`remove`) el `` del árbol DOM. Al estar el `` definido en el estilo como *inline*, al eliminarlo, todo el texto se desplaza ocupando su hueco.

ACTIVIDADES 4.1



- Modifique el código que muestra como resultado la Figura 4.3., para que cuando el usuario haga clic sobre el texto que aparece en el `<div>` todo los `` ocultos aparezcan de nuevo en su posición gradualmente (`FadeIn()`).

SOLUCIÓN

La solución de la actividad se consigue con las siguientes modificaciones del código.

```

$("span").click(function () {
    $(this).fadeOut(1000, function () {
        $("div").text("'" + $(this).text() + "' ;Se ha desvanecido!");
        //$(this).remove(); SE ELIMINA PORQUE SI DESPARECE DEL DOM YA NO LO PUEDO MOSTRAR
        //CON FADEIN()
    });
    //function del Fade
});
//Function del Click
$("div").click(function () { $("span").fadeIn(); });

```

4.4.3 EFECTOS SLIDING (DESPLAZAMIENTO)

Sliding son efectos de desplazamiento de elementos. Los *sliding* permiten hacer efectos de plegado y despliegue.

La sintaxis de los efectos Sliding de los que dispone jQuery es la siguiente:

```
.slideUp ([duración] [, easing] [, callback])  
.slideDown ([duración] [, easing] [, callback])  
.slideToggle ([duración] [, easing] [, callback])
```

Todos los parámetros son opcionales y tienen el mismo significado que los vistos en *show()* y *hide()*.

- *slideUp()*: recoge los elementos en altura de manera gradual.
- *slideDown()*: despliega los elementos en altura de manera gradual.
- *slideToggle()*: al igual que Toggle, alterna el proceso de plegado y desplegado en altura según se haga clic sobre el elemento un número par de veces o un número impar.

En el siguiente ejemplo⁷⁸ se muestran cuatro elementos <div> que se despliegan hacia abajo. Tres de ellos tienen en su interior texto y uno tiene un botón. Al hacer clic en el texto *¡Pulsa aquí!* se despliegan los <div>. A hacer otra vez, de vuelven a contraer (con *hide()*). El código es el siguiente:

```
<html>  
  <head>  
    <title>Ejemplo de animación con Show() y Hide()</title>  
    <script type="text/javascript" src="jquery-1.6.4.js"></script>  
    <script type="text/javascript">  
      $(document).ready(function() {  
        $(document.body).click(function () {  
          if ($("#div:first").is(":hidden")) {  
            $("div").slideDown("slow");  
          } else {  
            $("div").hide();  
          }  
        });  
      });  
    </script>  
    <style>  
      div { background:#de9a44; margin:3px; width:80px; height:140px; display:none;  
      float:left; }  
    </style>
```

⁷⁸ Código inspirado en los ejemplos del API jQuery: <http://api.jquery.com/slideDown/>

```

</head>
<body>
    ¡Pulsa Aquí!
    <div>Los efectos no son solo para divs o spans</div>
    <div>También son para cualquier elemento </div>
    <div><button id="hidr">Botones también</button></div>
    <div><p> aunque no estén incluidos en un div </p></div>
</body>
</html>

```

La Figura 4.4 muestra el resultado al hacer clic en ¡Pulsa Aquí!. Si se hace clic de nuevo se vuelve a la situación original.

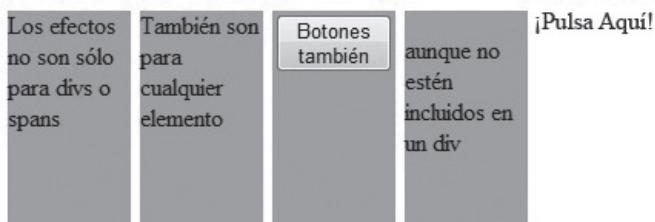


Figura 4.4. Ejemplo de uso de `slideDown()`

ACTIVIDADES 4.2



- Modifique el código que muestra como resultado la Figura 4.4 para que cuando el usuario haga clic sobre el texto ¡Pulsa Aquí! por segunda vez se recojan los <div> hacia arriba (`slideUp()`).

SOLUCIÓN

La solución pasa por sustituir `$(“div”).hide();` por `$(“div”).slideUp(“slow”);`. En ambos casos los <div> quedan ocultos (`hidden=true`).

4.4.4 OTROS EFECTOS

Además de los efectos vistos en las secciones anteriores existen otros que aumentan las prestaciones a la hora de hacer animaciones.

`.animate (propiedades, [duración] [, easing] [, callback])`

Este método anima un elemento según las propiedades CSS que se pongan en el parámetro *propiedades*. Los parámetros opcionales *[duración]*, *[easing]* y *[callback]* tienen el mismo significado que en los métodos vistos en las secciones anteriores.

animate() es similar a usar el método *.css()*⁷⁹ salvo que el rango de posibilidades en *animate()* es muy menor: la propiedades CSS que se utilicen deben tener valores numéricos, aquellas que no son numéricas no puedes animarse. Por ejemplo, *width* o *height* reciben valores numéricos y pueden animarse, sin embargo, *background* no puede animarse si se le da un valor no numérico. Los valores de medida son tratados en píxeles (px) a menos que se indique lo contrario (em o %)⁸⁰.

Es interesante destacar que el valor de las propiedades puede ser relativo al que tenga actualmente. Para ello se utiliza “+=” o “-=”.

El siguiente código⁸¹ utiliza un *<div>* para cambiarlo de tamaño (al hacer clic sobre él) y para moverlo cuando se usen los botones (*left* y *right*)

```
<html>
<head>
    <title>Ejemplo de animación con animate()</title>

    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $("#right").click(function() { //Animación del botón derecho.
                $(".block").animate({ "left": "+=50px"}, "slow");
            });
            $("#left").click(function() { //Animación del botón izquierdo.
                $(".block").animate({ "left": "-=50px"}, "slow");
            });
            $("div").click(function() { //al hacer click en div, cambia de tamaño
                $("div").animate({
                    width: "70%",
                    opacity: 0.5,
                    marginLeft: "0.6in",
                    fontSize: "3em",
                    borderWidth: "10px"
                }, 1500 );
            });
        });
    </script>
    <style>
        div {
```

⁷⁹ El método *css()* ha sido explicado en la sección 4.2.1

⁸⁰ En el Capítulo 2 se tratan las propiedades CSS más ampliamente

⁸¹ Inspirado en el ejemplo disponible en el API jQuery: <http://api.jquery.com/animate/>

```

        position:absolute;
        background-color:#abc;
        left:50px;
        width:90px;
        height:90px;
        margin:5px;
    }

```

</style>

</head>

<body>

<button id="left">«</button> <button id="right">»</button>

Pulsa AQUÍ también

</body>

</html>

La Figura 4.5 muestra el resultado después de una ejecución:

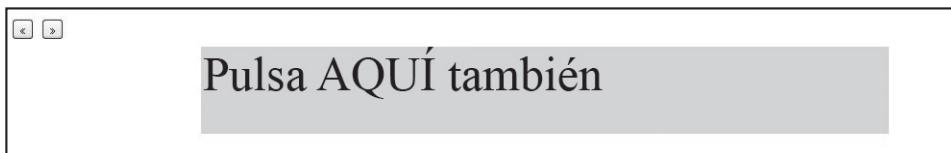


Figura 4.5. Ejemplo de uso de animate()

.delay(duración [, NombreCola])

Del método `.delay()` ya se comentó algo en la sección 4.3. Este método retrasa un tiempo especificado (duración) la animación de los elementos que se encuentran en una estructura cola (NombreCola), a la espera de ser animados. El significado de sus parámetros es el siguiente.

- *duración*: un número en milisegundos que indica cuánto tiempo se retrasará el comienzo de una animación con respecto al siguiente elemento de una cola.
- *NombreCola*: una cadena que contiene el nombre de la cola que contiene los elementos que se retrasarán .Por defecto existe la cola estándar *fx* que es la que se usa si no se pone valor a este parámetro⁸².

El siguiente código, al hacer clic sobre cualquiera de los párrafos, se recogen todos (`slideUp(300);`) ellos, aparece (`fadeIn(800);`)los que tienen identificador “antes” y más tarde (`delay(800);`) los que tienen identificador “después”.

```

<html>
<head>
    <title>Ejemplo de animación con delay()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>

```

⁸² Para saber más sobre colas (*queue*) y su métodos (*clearQueue()*, *doQueue()*, etc.) ver: <http://api.jquery.com/category/effects/>

```
<script type="text/javascript">
$(document).ready(function() {
    $("div").click(function() {
        $("div.anteriores").slideUp(300);
        $("div.anteriores").fadeIn(800);
        $("div.siguientes").slideUp(300);
        $("div.siguientes").delay(800);
        $("div.siguientes").fadeIn(800);
    });
});
</script>
<style>
    div.anteriores { background-color:#aac; }
    div.siguientes {background-color:#abc; }
</style>
</head>
<body>
    <h3> Pulsa en cualquiera de los textos </h3>
    <div class="anteriores"><p> Antes </p> </div>
    <div class="siguientes"><p> Después </p> </div>
    <div class="anteriores"><p> Antes </p> </div>
    <div class="siguientes"><p>Después </p> </div>
</body>
</html>
```

La Figura 4.6. muestra los párrafos *Antes* y mientras comienza a mostrarse los párrafos *Después* (*fadeIn(800)*).

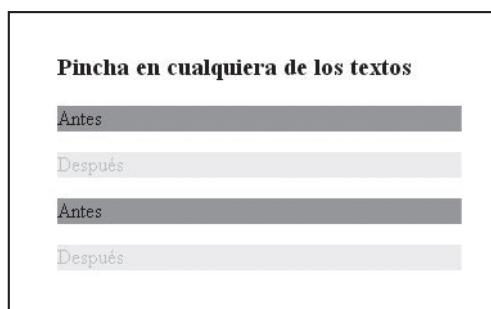


Figura 4.6. Ejemplo de uso de *delay()*

Junto con *delay()* otro método interesante es *stop()* que para la animación del elemento seleccionado si está actualmente en proceso.

ACTIVIDADES 4.3



- Modifique el código que muestra como resultado la Figura 4.6 para que cada uno de los párrafos aparezcan secuencialmente al cargarse la página. Cada uno con un retraso de 1 segundo (1.000 ms) respecto al anterior.

SOLUCIÓN

Una posible solución sería:

```
<html>
<head>
    <title>Ejemplo de animación con delay()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {

            $("div.primer").fadeIn(300);
            $("div.segundo").delay(1000);
            $("div.segundo").fadeIn(800);
            $("div.tercer").delay(2000);
            $("div.tercer").fadeIn(800);
            $("div.cuarto").delay(3000);
            $("div.cuarto").fadeIn(800);

        });
    </script>
<style>
    div.primer {background-color:#aac;display:none; }
    div.segundo {background-color:#abc;     display:none; }
    div.tercer {background-color:#abc;display:none;      }
    div.cuarto {background-color:#abc;     display:none; }
</style>
</head>
<body>
    <h3> Pulsa en cualquiera de los textos </h3>
    <div class="primer"><p>    Primero </p> </div>
    <div class="segundo"><p>    Segundo    </p> </div>
    <div class="tercer"><p>    Tercero    </p> </div>
    <div class="cuarto"><p>    Cuarto   </p> </div>
</body>
</html>
```

4.5 COMPORTAMIENTOS INTERACTIVOS

Una vez vistos los efectos más comunes usando jQuery, en esta sección se muestran los diferentes eventos (ratón y teclado) que atiende jQuery para la interacción del usuario con objetos de la web. Además, en la última parte se describen los eventos asociados a las ventanas.

4.5.1 EVENTOS DE RATÓN

Los eventos de ratón ejecutan una función cuando el usuario utiliza el puntero del ratón en un elemento determinado.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Sobre un *elemento*, cuando el usuario lanza un *evento* lo atiende (se ejecuta) la *función*. A continuación se describen los eventos de ratón disponibles. En la descripción se supone una configuración de ratón para diestros, donde el botón izquierdo de ratón es el que se usa para seleccionar.

- **.click()**. Se lanza un evento clic cuando el usuario, colocado sobre un elemento (objeto) presiona el botón izquierdo del ratón y lo levanta, todo dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.dblclick()**. Es el llamado doble clic. Este evento se lanza cuando se repite dos veces la secuencia de un clic, es decir, sobre un elemento (objeto) el usuario presiona el botón izquierdo del ratón, lo levanta, lo vuelve a presionar y lo levanta, toda la secuencia dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.focusin()**. Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout()**. Es el evento contrario a *focusin()*, se lazán cuando un elemento (objeto) pierde el foco.
- **.mousedown()**. Este evento se lanza cuando el usuario presiona el botón del ratón sobre un elemento (objeto). La diferencia con respecto a *click()* es que éste no se lanza hasta que se suelta el botón dentro de ese mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.mouseup()**. Este evento se lanza cuando el usuario levanta el botón de ratón (previamente presionado). Cualquier elemento HTML puede recibir este evento.
- **.mouseenter()**. Se lanza este evento cuando el puntero del ratón entra en un elemento (objeto). No hace falta presionar ningún botón para que este evento se lance cuando el puntero entra en la región definida por el elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseleave()**. Se lanza este evento cuando el puntero del ratón sale de la región delimitada por un elemento (objeto). Es el evento opuesto a *mouseenter()*. Cualquier elemento HTML puede recibir este evento.

- **.mousemove()**. Este evento se lanza cuando el puntero de ratón se mueve dentro de un elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseover()**. Es parecido a *mouseenter()*, pero se diferencia cuando hay elementos anidados (por ejemplo, un *<p>* dentro de un *<div>* *hijo* anidado dentro de un *<div>* *padre*): se lanza el *mouseover()* del padre al pasar de *<p>* a *<div>* *hijo* o de *<div>* *hijo* a *<p>* o de *<div>* *hijo* al *<div>* *padre*.
- **.mouseout()**. Es el opuesto de *mouseover()*. Es parecido al *mouseleave()* pero con la diferencia con objetos anidados mostrada con *mouseover()*.

El siguiente ejemplo, cuando se ejecuta, muestra la diferencia entre *mouseover()* y *mouseenter()*.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.first { position:relative;background-color: #3f3; left: 0px; height:400px; width:400px; }
div.second { position:relative;background-color: #a3f; top:100px;left:100px;height:200px; width:200px; }
p {background:#33CC99; }
</style>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    var cuenta_enter=0;
    var cuenta_over=0;
    $("div.first").mouseenter(function() {
        $("p:first").text("con_mouseEnter "+ cuenta_enter++);
    });
    $("div.first").mouseover(function() {
        $("p:last").text("con_mouseOver "+ cuenta_over++);
    });
});
</script>

</head>
<body>
<div class="first">
    <div class="second">
        <p></p>
        <p></p>
    </div>
</div>

```

```
</div>
</div>
</body>
</html>
```

En el ejemplo, al pasar el ratón de los párrafos `<p>` al `<div>` hijo aumenta la cuenta de `cuenta_over`. Lo mismo al pasar del `<div>` hijo al `<div>` padre. Sin embargo, la `cuenta_enter` solo se incremente al pasar de fuera de los `<div>` al `<div>` padre.

`.hover()`. Este evento se lanza cuando el puntero está sobre un elemento o sale de él. Su principal ventaja es que permite definir una función para cuando el puntero entra en el elemento y otra para cuando salga. Es un evento muy usado con menús, cuando se desea hacer, por ejemplo, seleccionar opciones de menú, destacando alguna de sus propiedades o menús que se despliegan. Su sintaxis es:

```
Elemento.hover ( función_entrada (handler), función_salida (handler));
```

El siguiente ejemplo muestra un submenú que aparece al colocarse con el ratón en la opción “localización”.

```
<!DOCTYPE html>
<html>
<head>

<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {

    $("li#localizacion").hover(function () {
        $("span#opcion_loc").show();
    }, function () {
        $("span#opcion_loc").hide();
    });

});

</script>
<style>
ul { margin-left:20px; color:blue; }
li { cursor:default; }
span { display:none }
```

```
</style>
</head>
<body>
<ul>
    <li id="localizacion">Localización</li>
    <span id="opcion_loc">
        <ul>
            <li>Mapa</li>
            <li>Dirección</li>
            <li>Teléfonos</li>
        </ul>
    </span>
    <li id="recetas">Recetas</li>
</ul>

</body>
</html>
```

ACTIVIDADES 4.4



- Modifique el código anterior, apoyándose en los eventos vistos en esta sección, para añadirle la siguiente funcionalidad:
- Que al hacer clic sobre una opción se despliegue el submenú y al hacer clic de nuevo se vuelva a ocultar.
 - Que al moverse el puntero del ratón por encima de cualquier opción (menú o submenú) ésta cambie de color (a rojo), y al quitarse vuelva a su color original (a azul).

La Figura 4.7 muestra cómo podría quedar el resultado con los dos menús desplegados:

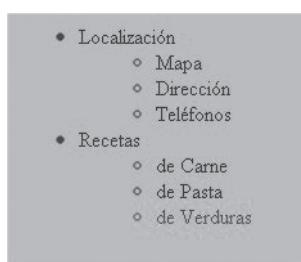


Figura 4.7. Ejemplo de menú desplegable

SOLUCIÓN

La solución para mostrar el código de la Figura 4.7 es el siguiente:

```
<!DOCTYPE html>
<html>
<head>

<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {

    $("li").hover(function () {
        $(this).css("color", "red");
    }, function () {
        $(this).css("color", "blue");
    });

    $("li#localizacion").toggle(function () {
        $("span#opcion_loc").show();
    }, function () {
        $("span#opcion_loc").hide();
    });

    $("li#recetas").toggle(function () {
        $("span#opcion_rec").show();
    }, function () {
        $("span#opcion_rec").hide();
    });
});;

</script>
```

```
<style>
body { background:#99CCFF; }
ul { margin-left:20px; color:blue; }
li { cursor:default; }
span { display:none; background:#CCCCFF ; }
</style>
</head>
<body>
<ul>
<li id="localizacion">Localización</li>
<span id="opcion_loc">
<ul>
<li>Mapa</li>
<li>Dirección</li>
<li>Teléfonos</li>
</ul>
</span>
<li id="recetas">Recetas</li>
<span id="opcion_rec">
<ul>
<li>de Carne</li>
<li>de Pasta</li>
<li>de Verduras</li>
</ul>
</span>
</ul>

</body>
</html>
```

4.5.2 EVENTOS DE TECLADO

Los eventos de teclado ejecutan una función cuando el usuario utiliza las teclas del teclado en un determinado elemento.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Sobre un *elemento*, cuando el usuario lanza un *evento* lo atiende (se ejecuta) la *función*. A continuación se describen los eventos de teclado disponibles. La mayoría de ellos van asociados a formularios ya que suelen ser usados cuando el usuario entra mediante teclado.

- **.focusin()**. Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout()**. Es el evento contrario a *focusin()*, se lazan cuando un elemento (objeto) pierde el foco.
- **.keydown()**. Se lanza el evento cuando el usuario presiona una tecla. El evento se lanza cuando el elemento tiene el foco. Para determinar que tecla ha sido presionada se examina el objeto *event* que se le pasa a la función que atiende el evento. jQuery ofrece la propiedad *.which* de *event* para recuperar el código de la tecla que se presiona. Si lo que se desea es obtener el texto que se ha introducido es mejor opción usar *.keypress()*.
- **.keyup()**. Se lanza el evento cuando el usuario libera una tecla. El evento se lanza cuando el elemento tiene el foco.
- **.keypress()**. Se lanza el evento cuando el usuario presiona una tecla pero con la idea de registrar qué carácter se ha pulsado. También se puede conseguir con *.keydown()*, sin embargo si se pulsa una tecla y se mantiene, *keydown()* solo registra un evento para esa tecla, mientras que *keypress()* registra uno por cada carácter introducido.

El siguiente ejemplo muestra un sencillo detector de qué tecla se ha presionado en cada momento. El parámetro *e* es de tipo *event* y es el que contiene la tecla pulsada, obtenida con *e.which*. Con *e.preventDefault()*; se consigue no se escriba nada en el *textarea*, es decir se inhabilita el comportamiento habitual del evento, que es escribir las teclas en el *textarea*.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {background:#33CC99;}
</style>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $('#objetivo').keypress(function(e) {
        e.preventDefault();
        $("#salidapress").html(e.which + ": " + String.fromCharCode(e.which))
    });
});
</script>
</head>
<body>
<form>
<fieldset>
<label for="objetivo">Teclea cualquier cosa</label>
```

```
<input id="objetivo" type="text" />
<div id="salidapress"></div>
</fieldset>
</form>
</body>
</html>
```

4.5.3 EVENTOS DE VENTANA

Los eventos principales de ventana son: *scroll* y *resize*. La estructura básica de los dos es la misma que en los eventos anteriores de ratón y teclado.

- **.scroll()**. Este evento se atiende cuando sobre un elemento que tiene barras de desplazamiento se mueve una de ellas. El evento se suele aplicar a elemento “ventana” pero también sobre *frames* o elementos con la propiedad CSS *overflow* puesta a *scroll*.
- **.resize()**. El evento se lanza cuando a un elemento tipo ventana se le cambia el tamaño.

El siguiente ejemplo muestra el comportamiento de ambos eventos. *Scroll()* se activa cuando se mueven las barras de desplazamiento o se pulsa en el texto “*PULSA Y lazo el evento pero no hago el scroll (desplazamiento)*”. Al activarse aparece un texto “se hace *Scroll*”, pero este se desvanece (*fadeout*). Por otro lado, cuando se modifica el tamaño de la ventana del navegador, si el ancho de la ventana es mayor de 500 px se pone un color de fondo para la caja de texto y si es menor otro.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {overflow: scroll;
      width: 300px;
      height: 100px;
      font-family: Arial, Helvetica, sans-serif;
      color: #888888;
      padding: 7px 3px 0 4px;
      font-size: 12px;

}

span {display:inline;}
</style>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
```

```
$ (document) .ready(function() {  
    $('#prueba') .scroll(function() {  
        $("span#textoevento") .css("display", "inline") .fadeOut("slow"); });  
    $('#repite') .click(function() { $('#prueba') .scroll();  
    });  
    $(window) .resize(function() {  
  
        if ($(window) .width() > 500)  
        {  
            $('div') .css("background", "#99CC99");  
        }  
        else  
        {  
            $('div') .css("background", "#9999CC");  
        }  
  
    });  
});  
</script>  
</head>  
<body>  
<div id="prueba" >  
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
    sed do eiusmod tempor incididunt ut labore et dolore magna  
    aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
    ullamco laboris nisi ut aliquip ex ea commodo consequat.  
    Duis aute irure dolor in reprehenderit in voluptate velit  
    esse cillum dolore eu fugiat nulla pariatur. Excepteur  
    sint occaecat cupidatat non proident, sunt in culpa qui  
    officia deserunt mollit anim id est laborum.  
</div>  
<span id="textoevento">  
    Se hace Scroll  
</span>  
<span id="repite"> PULSA Y lazo el evento pero no hago el scroll (desplazamiento)  
</span>  
</body>  
</html>
```

4.6 REPRODUCCIÓN DE SONIDO, VÍDEO Y ANIMACIÓN

Como se comentó en el Capítulo 3, actualmente la reproducción de vídeo y audio está recayendo sobre las etiquetas `<video>` y `<audio>` de HTML5. Sin embargo, en algunos casos sustituyendo y en otros complementando, existen *plugins* específicos para jQuery para insertar vídeo y audio en la web.

Estos *plugins*, en algunos casos, gracias a la flexibilidad que ofrecen, son una alternativa personalizada a `<video>` y `<audio>` que permite diseños más potentes y visuales. Sin embargo, no siempre es así, y en otros muchos casos usar estos *plugins* perjudica (por rendimiento o por tiempo de desarrollo) más que ayuda, y es preferible usar `<video>` `<audio>`.

A continuación se muestran algunos de los *plugins* mejor valorados para manejar vídeo y audio con jQuery.

- **Acorn Media Player**⁸³: es un *plugin* de jQuery que permite personalizar el reproductor de HTML5 `<video>` haciendo hincapié en la accesibilidad del recurso. Está destinada a la reproducción de vídeo.
- **Video JS**⁸⁴: es un reproductor muy conocido para JavaScript y jQuery. Está destinada a la reproducción de vídeo.
- **jPlayer**⁸⁵: otro reproductor muy versátil adecuado para un gran número de navegadores y formatos tanto de vídeo como de audio.
- **Flare Video**⁸⁶: otro reproductor que ofrece solución para todos los navegadores. Está destinada a la reproducción de vídeo.
- **Media Element**⁸⁷: un reproductor jQuery fácil de usar. Es adecuado tanto para vídeo como para audio.
- **Simple Player**⁸⁸: un reproductor solo de audio. Soporta CSS para definir la apariencia del reproductor. Es solo adecuado para los navegadores que soportan HTML con formatos *mp3* y *ogg*.

En los sitios web de cada reproductor se puede obtener información y ejemplos sobre cómo usar cada *plugin* e insertarlo en las páginas.

⁸³ <http://ghinda.net/acornmediaplayer/>

⁸⁴ <http://videojs.com/>

⁸⁵ <http://www.jplayer.org/>

⁸⁶ <http://flarevideo.com/>

⁸⁷ <http://mediaelementjs.com/>

⁸⁸ <http://jay-notes.blogspot.com/2010/06/simple-player-very-simple-html5-audio.html>

ACTIVIDADES 4.5



► Modifique el código de la Actividad 2.7 para incluir los siguientes efectos:

- a. Al cargar la página solo debe aparecer el pie de página. El resto de elementos deben permanecer ocultos.
- b. Dentro del pie de página insertar 4 <div>, cada uno de un color.
- c. Cuando el usuario hace clic (un número impar de veces) en cada uno de los tres primeros <div> aparecerá el correspondiente *encabezado, menú y cuerpo de la página*.
- d. Cuando el usuario hace clic (un número par de veces) en cualquier de los tres <div> desaparecerá el correspondiente *encabezado, menú y cuerpo de la página*.
- e. Para el cuarto <div> cuando el usuario hace clic, desaparecen los 4 <div> de forma gradual quedando en el pie de página un texto “aquí va el nuevo pie de página”.

Si el usuario hace clic en el último <div> antes de mostrar todos las partes de la página, ya no se podrán mostrar ya que todos los <div> estarán invisibles.

La Figura 4.8 muestra la página al inicio, la Figura 4.9 la página al hacer clic en los tres primeros <div> y luego en el último.

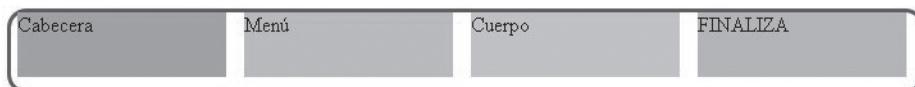


Figura 4.8. Antes de la animación

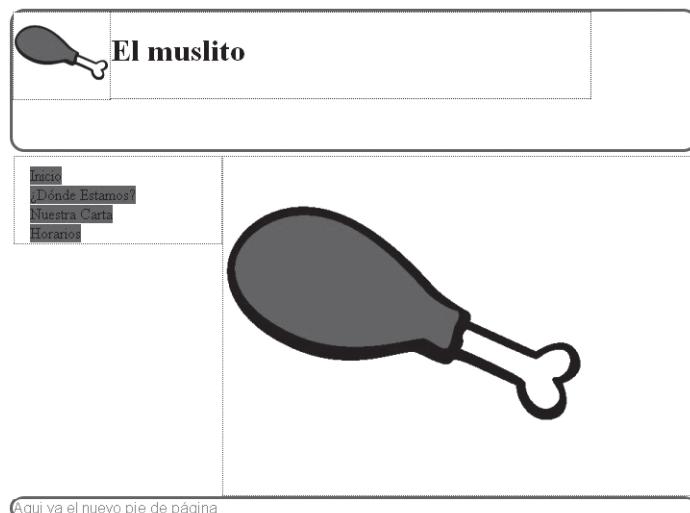


Figura 4.9. Al final de la animación

SOLUCIÓN

El código de solución está disponible en el material adicional: *Actividad final_solucion.html*

4.7 CONCLUSIONES

En este capítulo se ha hecho una introducción a jQuery desde el punto de vista del diseño y creación de interfaces de usuario. Eventos de ratón, teclado, efectos y animaciones han sido los puntos principales que se han tratado. La mayor parte de la funcionalidad ha sido ilustrada con códigos de ejemplo, lo cual es una ayuda no solo para entender los aspectos de interfaz, sino también de cómo se incluye el código en una página web.

En la web hay gran cantidad de código realizado por la comunidad jQuery que ofrece desarrollos de interfaz bastante completos: botones, menús, animaciones, etc. El lector familiarizado con jQuery podrá interpretar esos ejemplos y modificarlos para poder incluirlos en sus páginas (si la licencia lo permite). Un ejemplo es mostrado en Onextrapixel⁸⁹ para el desarrollo de un botón con JQuery y CSS.

Sin embargo, con todo lo visto, jQuery es un lenguaje mucho más amplio, ya que como lenguaje de programación ofrece alternativas relacionadas con las condiciones, bucles, acceso a datos, etc. Junto a esto, JavaScript en general y jQuery en particular es el lenguaje con el que se implementa la tecnología AJAX (Asynchronous JavaScript And XML) lo que hace que este lenguaje sea todavía más extenso y completo.

Por lo tanto, se recomienda que lector interesando en el desarrollo de aplicaciones web profundice más en el tema, abarcando el conocimiento de otras funcionalidades proporcionadas por jQuery, *plugins* sobre jQuery y su trabajo con AJAX. En la web hay gran cantidad de tutoriales y bibliografía relacionada con esta tecnología. Aquí se muestra solo algunos ejemplos en inglés:

- **jQuery API**⁹⁰: toda la documentación on line sobre el API de jQuery con ejemplos.
- **w3schools**⁹¹: tutorial *on line* sobre las funciones jQuery.
- **Canal**⁹² de vídeo-tutoriales: vídeos sobre el API de jQuery en *YouTube.com* (inglés).

⁸⁹ <http://www.onextrapixel.com/2010/12/06/creating-3d-buttons-with-css-jquery/>

⁹⁰ <http://api.jquery.com/>

⁹¹ <http://www.w3schools.com/jquery/default.asp>

⁹² http://www.youtube.com/watch?v=GNb8T5NBdQg&list=PL0EFA1232C66601D7&index=1&feature=plpp_video



RESUMEN DEL CAPÍTULO



En este capítulo se ha mostrado mediante ejemplos el uso de jQuery para desarrollar animaciones y efectos en la web. Aunque hasta hace poco tiempo, las animaciones en la Web y la interacción era exclusividad de aplicaciones Flash, actualmente la Web tiende a utilizar jQuery, HTML5 y CSS para desarrollar animaciones fácilmente soportadas por todos los navegadores.

El capítulo no pretende mostrar todo lo relacionado con jQuery, ni siquiera todo lo relacionado con el desarrollo de interfaces de usuario con jQuery (eso también daría para un libro de cientos de páginas). Sin embargo, sí pretende dar al lector una muestra de qué permite hacer y cómo, acercándolo de manera práctica al uso de esta tecnología.



EJERCICIOS PROPUESTOS



- **1.** Busque en Internet dos herramientas libres diferentes a las descritas en el capítulo que permitan crear código jQuery automáticamente para hacer animaciones. Describa su funcionalidad.
- **2.** En parejas, crear dos animaciones con alguna de las herramientas de generación de código mostradas en el capítulo. Incluir cada miembro del equipo esa animación en el código de la web creada en el ejercicio 5 del Capítulo 2. Las animaciones creadas deben estar en consonancia con el diseño de la plantilla.
- **3.** En grupos de dos, para la web creada en el ejercicio anterior, crear con jQuery directamente una pantalla de bienvenida que se muestre al principio de cargar la página y que sirva de introducción. Esa página debe tener animación. Cuando la animación termina desaparece y aparece la web lista para usarse. Enumeren qué dificultades técnicas se han encontrado al hacer esta actividad.
- **4.** En grupos de dos, buscar en Internet dos diseños de botones creados con JQuery+CSS y HTML5. Los botones seleccionados deben estar en consonancia con el diseño de la plantilla seleccionada para la Web del ejercicio anterior.
- **5.** Continuando con el ejercicio anterior, cada miembro debe insertar el botón descargado en la Web y asociarle una función. ¿Qué dificultades se han encontrado al hacer la integración?



TEST DE CONOCIMIENTOS



1 No se puede decir sobre jQuery:

- a) Es una librería basada en JavaScript muy adecuada para hacer animaciones en un contexto de HTML5.
- b) Es un lenguaje alternativo al uso de Flash como herramienta para crear animaciones.
- c) Es un lenguaje sencillo que no necesita de herramientas que generen código jQuery automáticamente a partir de modelos visuales.

2 En jQuery es falso que:

- a) Se pueden cambiar propiedades HTML pero no se pueden modificar propiedades CSS.
- b) Se pueden modificar propiedades CSS usando el método `css()`.
- c) Se puede añadir texto a un HTML usando el método `text()`.

3 Respecto a los efectos de animación en jQuery:

- a) `delay()` permite retrasar un tiempo especificado la ejecución para un elemento de un efecto.
- b) Si se desea transformar una elemento a una opacidad determinada se utiliza `Fade()`.
- c) El método `SlideToggle()` solo funciona asociado con el método `show()`.

4 Respecto a los eventos de ratón `mouseover()` y `mouseenter()`:

- a) Ambos se comportan de manera idéntica.
- b) Ambos son idénticos salvo cuando hay elementos anidados. `mouseover()` se ejecuta cuando se pasa de un hijo a un padre.
- c) Ambos son idénticos salvo cuando hay elementos anidados. `mouseenter()` se ejecuta cuando se pasa de un hijo a un padre.

5 Un evento `click()` en jQuery():

- a) Si se captura un `click()` no se puede capturar un `mousedown()`.
- b) Se ejecuta cuando sobre un objeto se presiona el botón izquierdo del ratón (diestros) y se suelta sobre el mismo elemento.
- c) Un `click()` es incompatible con capturar un `dbl-click()`.

6 Sobre algunos métodos de jQuery es falso que:

- a) `hover()` permite llamar a una función cuando el puntero entra en un elemento y a otra cuando sale.
- b) `toggle()` permite asociarle dos funciones. En los `click()` pares se ejecutará una y en los impares la otra.
- c) `bind()` permite asignar varios eventos a un elemento, pero de uno en uno, con tantas llamadas a `bind()` como eventos se quieran asignar.

5

Desarrollo de webs accesibles

OBJETIVOS DEL CAPÍTULO

- ✓ Reconocer la necesidad de diseñar webs accesibles.
- ✓ Identificar las principales pautas de accesibilidad al contenido.
- ✓ Analizar la accesibilidad de diferentes sitios web.
- ✓ Analizar los posibles errores según los puntos de verificación de prioridad.
- ✓ Verificar los niveles alcanzados mediante el uso de test externos y la visualización de la interfaz con diferentes navegadores y tecnologías.

5.1 CONCEPTO DE ACCESIBILIDAD

Hablar de Accesibilidad Web es hablar de un acceso universal a la Web, es hablar de que todo el mundo pueda acceder a la Web y desde cualquier contexto, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios.

Con esta idea de *accesibilidad* nació la Iniciativa de Accesibilidad Web, conocida como WAI (acrónimo de *Web Accessibility Initiative*). Se trata de una actividad desarrollada por el W3C⁹³. El objetivo de la WAI es facilitar el acceso de las personas con discapacidad, desarrollando pautas de accesibilidad, mejorando las herramientas para la evaluación y reparación de accesibilidad Web, llevando a cabo una labor educativa y de concienciación en relación a la importancia del diseño accesible de páginas Web y abriendo nuevos campos en accesibilidad a través de la investigación en este área.

Bajo el término accesibilidad radica hacer la Web más accesible para todos los usuarios independientemente de las circunstancias y los dispositivos utilizados a la hora de acceder a la información. Partiendo de esta idea, una página accesible lo será tanto para una persona con discapacidad como para cualquier otra persona que se encuentre bajo circunstancias externas que dificulten el acceso a la información (es el caso de ruidos externos, en situaciones donde nuestra atención visual y/o auditiva no esté disponible, pantallas con visibilidad reducida, etc.). En este sentido, diseñar considerando la accesibilidad no solo favorece el acceso a la información de personas con discapacidad sino que también, y bajo determinadas circunstancias y contextos, beneficia a personas con discapacidad temporal en uno u otro momento.

La Web ofrece a aquellas personas con discapacidad una oportunidad de acceder a la información y de interactuar. Hoy en día, la Web es un recurso muy importante para diferentes aspectos de la vida: la educación, el empleo, el gobierno, el comercio, la sanidad, el entretenimiento y muchos otros. Por ello, es muy importante que la Web sea accesible para así proporcionar un acceso equitativo y en igualdad de oportunidades a las personas con discapacidad. Una página Web accesible puede ayudar a personas con discapacidad a que participen más activamente en la sociedad.

Otra consideración importante, más orientada a instituciones y empresas, es que la accesibilidad web es un requisito establecido en algunos casos por leyes y políticas, lo cual la hace de obligado cumplimiento en muchas situaciones. Por ejemplo, la Ley de Impulso de la Sociedad de la Información (LISI), amplió la Disposición Adicional 5^a de la LSSICE y fijó el 31 de diciembre de 2008 para que todas las páginas, actualmente existentes o de nueva creación cumplan la nivel de adecuación AA de la Norma UNE 139803:2004 y la Ley 49/2007, de 26 de diciembre establece además el régimen de infracciones y sanciones en materia de Igualdad de Oportunidades, No Discriminación y Accesibilidad Universal de las personas con Discapacidad. Las multas oscilan entre los 301 euros y el millón de euros.

⁹³ <http://www.w3c.es/>. Ya se ha hecho referencia a la W3C en capítulos anteriores como consorcio de estandarización de todo lo relacionado con la web.

ACTIVIDADES 5.1



► Trate de ponerse en las diferentes situaciones que a continuación se le sugieren y trate de responder a las preguntas que se le formulan:

- a. Si cierra los ojos, ¿sería capaz de navegar a través de un sitio web?
- b. Si tuviera limitaciones auditivas, ¿sería capaz de navegar a través de un sitio web?
- c. Si no pudiera utilizar sus manos, ¿sería capaz de consultar un sitio web?

SOLUCIÓN

Si es ciego, si tiene dificultades en la visión o simplemente la luz ambiental le impide recibir la información de un dispositivo móvil o de una pantalla de ordenador no podrá recibir toda la información que se le transmite a través de esos dispositivos. El diseñador tiene que tener en cuenta estos detalles a la hora de diseñar sus sitios web. El usuario podrá utilizar herramientas que facilitarán el acceso a la información de personas con discapacidad, pero siempre y cuando esas páginas o sitios web hayan sido preparados para ello.

Las personas con dificultades auditivas deberían encontrarse con muy pocos problemas ante las interfaces web para el acceso a la información, ya que la mayoría están basadas en información visual. En cualquier caso, el diseñador de contenidos debe tener en consideración la codificación de los mensajes de alerta mediante sonidos, como puede ser un mensaje de error, y utilizar mensajes textuales simultáneos o equivalentes.

A una persona que tenga dificultades motrices le resultará complicado interactuar con herramientas de interacción tradicionales, como puede ser un ratón o un teclado, por ello se investiga y se desarrollan estilos de interacción diferentes.

5.2 EL CONSORCIO WORLD WIDE WEB (W3C)

Llegados ha este punto, aunque la W3C haya sido mencionada en numerosas ocasiones en capítulos anteriores, es importante describir explícitamente sus funciones aquí, con el fin de exponer claramente su importancia en el desarrollo web accesible. El Consorcio *World Wide Web* (W3C) es una asociación internacional formada por organizaciones, personal y público en general, que trabajan conjuntamente para desarrollar normas y estándares para la Web. La misión del W3C es:

“Guiar a la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web”.

El consorcio W3C desarrolla normas, estándares web y pautas, pero no es un organismo que vele por que se cumplan dichas normas, sino que ofrece recomendaciones sobre cómo deben usarse los diferentes formatos, y lenguajes web. Debido a su carácter independiente, las diferentes empresas que fabrican productos para la navegación en la Web, toman sus recomendaciones y las adaptan a los diferentes productos que publican.

Tim Berners-Lee, junto con otros, fue uno de los fundadores del consorcio dedicado a generar consenso en relación a las tecnologías Web. Berners-Lee, inventó la Web en 1989 mientras trabajaba en la Organización Europea de Investigación Nuclear (CERN), y ha sido el director del Consorcio desde su fundación en 1994.

La actividad del consorcio W3C tiene presentes diferentes líneas de actuación:

- Trabajar y potenciar la Web para todos. El valor social de la Web está en que permite actividades de comunicación, compartición de conocimiento y comercialización. El objetivo, por tanto, es que todas estas posibilidades estén al alcance de todo el mundo.
- Facilitar el acceso a la Web independientemente del dispositivo que se utilice y de la forma de interactuar (voz, gestos, formularios, etc.).
- Propiciar y defender la calidad y la confianza en los contenidos disponibles en la Web.
- Contribuir a la evolución de la propia Web, potenciando las posibilidades de la Web en tres dimensiones: colaboración (Web 2.0), semántica (Web 3.0) y ubicua (Web 4.0).

En la Web 2.0 se consolida el concepto de la web colaborativa y participativa, dejando atrás a los sitios tradicionales que pertenecían a la denominada Web 1.0. La Web 2.0 es una forma de entender Internet donde se promueve que la organización y el flujo de información dependan del comportamiento de las personas que acceden a ella, permitiéndose a éstas no solo un acceso mucho más fácil y centralizado a los contenidos, sino su propia participación tanto en la clasificación de los mismos como en su propia construcción, mediante herramientas cada vez más fáciles e intuitivas de usar. Ejemplos de sitios Web 2.0 son: Wikipedia o Flickr.

La Web 3.0, o Web semántica, es una extensión de la Web, con la que se le dota de mayor significado a los contenidos, con ello cualquier usuario en Internet debería poder encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida.

Antes de 2020 se prevé que evolucionemos hacia un nuevo concepto de web: la Web 4.0 o ubicua, aún no habiéndose desarrollado por completo la Web 3.0. La Web 4.0 se prevé que incrementará a través de la unión entre inteligencia humana e inteligencia artificial las ventajas ya existentes que nos proporciona la Web.

- Contribuir a la confianza en la Web, incorporando y desarrollando mecanismos que ofrezcan mecanismos de privacidad, seguridad, protección y cifrado de la información intercambiada a través de la Web.

ACTIVIDADES 5.2



- Visite la página web del Consorcio *World Wide Web* en España e identifique en qué trabaja, qué estándares desarrolla y cuáles de ellos están más relacionados con los contenidos de este libro.

SOLUCIÓN

El consorcio W3C desarrolla especificaciones técnicas y directrices a través de un proceso que ha sido diseñado para maximizar el consenso sobre el contenido de un informe técnico, de forma que se pueda asegurar la alta calidad técnica y editorial, así como obtener un mayor apoyo desde el W3C y desde la comunidad en general.

Los estándares y normas desarrolladas por el consorcio que están directamente ligadas con el módulo son las englobadas bajo el epígrafe Diseño y Aplicaciones Web.

5.3 PRINCIPIOS GENERALES DE DISEÑO ACCESIBLE

Con el fin de que un sitio web llegue al mayor número posible de usuarios, se ha de diseñar accesible a todos ellos, al margen de cualquier discapacidad que puedan tener. Para ello el consorcio W3C ha trabajado a lo largo del tiempo en identificar principios de diseño, a los que acompañan pautas que se verán en la próxima sección. Todas esas pautas se pueden organizar atendiendo a distintos principios de diseño, que son los que seguidamente serán presentados.

Los documentos o normas del consorcio W3C que deben considerarse para identificar estos principios de diseño son dos. Inicialmente, en 1999, las normas contenidas en la versión 1.0 de las guías de accesibilidad del contenido web (WCAG 1.0 – *Web Content Accessibility Guidelines*) identificaron catorce principios de diseño accesible. Estos principios fueron los siguientes:

1. Proporcionar alternativas para los contenidos visuales y auditivos. Se resalta la importancia de aportar textos equivalentes para los contenidos no textuales.
2. No basarse solo en el color ya que, si los textos y los gráficos no se entienden cuando se vean sin color, cuando los usuarios no tengan pantallas en color o utilicen dispositivos de salida no visuales, no recibirán información.
3. Utilizar marcadores y hojas de estilo y hacerlo apropiadamente⁹⁴.
4. Identificar el lenguaje natural usado, pues cuando el lenguaje natural no se identifica o se producen cambios en las abreviaturas, pueden ser indescifrables para los lectores de pantalla y los dispositivos Braille.
5. Crear tablas que se transformen correctamente, asegurarse que las tablas tienen los marcadores necesarios para transformarlas mediante los navegadores accesibles y otras aplicaciones de usuario.
6. Asegurarse de que las páginas que incorporen nuevas tecnologías se transformen correctamente: asegurarse de que las páginas son accesibles incluso cuando no se soportan las tecnologías más modernas o éstas están desconectadas.
7. Asegurar al usuario el control: asegurarse de que los objetos o páginas que se mueven, parpadean, se desplazan o se actualizan automáticamente, pueden ser detenidos o parados.
8. Asegurar la accesibilidad directa de las interfaces incrustadas.
9. Diseñar teniendo en cuenta diversos dispositivos (ratón, teclado, voz, etc.).
10. Utilizar soluciones provisionales: utilizar soluciones de accesibilidad provisionales de forma que las ayudas técnicas de los antiguos navegadores operen correctamente.
11. Utilizar las tecnologías y pautas W3C: dónde no sea posible utilizar una tecnología W3C, o usándola se obtengan materiales que no se transforman correctamente, proporcionar una versión alternativa del contenido que sea accesible.

⁹⁴ Como las vistas en el Capítulo 2.

12. Proporcionar información de contexto y orientación para ayudar a los usuarios a entender páginas o elementos complejos.
13. Proporcionar mecanismos claros y consistentes de navegación para incrementar la probabilidad de que una persona encuentre lo que está buscando en un sitio.
14. Asegurarse de que los documentos sean claros y simples para que puedan ser más fácilmente comprendidos.

Más recientemente, los principios anteriores han sido revisados en la versión 2.0 de las guías de accesibilidad de los contenidos para la Web (WCAG 2.0; 2009). En función de esta revisión, aquellos catorce principios han quedado reducidos y simplificados a los cuatro principios siguientes:

1. Los sitios web deberían ser fácil de percibir o perceptibles: la información y los componentes de la interfaz de usuario deben presentarse a los usuarios de la manera en que puedan percibirlos.
2. Los sitios web deberían ser fácil de operar u operables: los componentes de la interfaz de usuario y la navegación deben ser operables.
3. Un sitio web debería ser fácil de comprender o comprensible: la información y el manejo de la interfaz de usuario deben ser comprensibles.
4. Un sitio web debería ser robusto: el contenido debe ser lo suficientemente robusto como para confiarse en su interpretación por parte de una amplia variedad de agentes de usuario, incluidas las tecnologías que asisten a las personas con discapacidad.

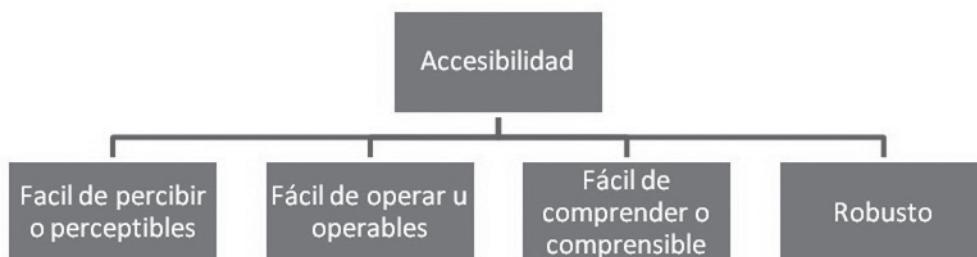


Figura 5.1. Principios de la accesibilidad según la WCAG 2.0

Ya sean los principios de diseño recogidos en el documento WCAG 1.0 o los cuatro principios aportados con la norma WCAG 2.0, y mostrados en la Figura 5.1, el Consorcio W3C utiliza dichos principios para organizar pautas, criterios y guías de estilo. Todos estos elementos facilitan dar respuesta no solo a qué hay que hacer para diseñar sitios web accesibles sino a saber cómo un diseñador se puede poner manos a la obra para lograrlo. Dichas pautas de accesibilidad se comentarán seguidamente.

ACTIVIDADES 5.3



► Aplicando los principios generales de diseño recogidos en esta sección, elige un sitio web, por ejemplo la de su equipo de fútbol favorito, el sitio web de su instituto o el de su ayuntamiento, e identifique cómo podría tener presentes dichos principios a la hora de diseñar un sitio web.

SOLUCIÓN

Los principios web se redactan de manera general y no son siempre fáciles de poner en práctica sin la suficiente experiencia. Los principios están recogidos a un nivel de abstracción muy grande. El hecho de que algo sea fácil de percibir, fácil de operar, fácil de comprender y robusto exige saber qué hacer y cómo hacerlo. Necesitamos algo más de ayuda que los principios para mejorar el desarrollo de web accesibles (eso se verá en las siguientes secciones).

En cualquier caso, si revisa cualquiera de las páginas sugeridas podrá observar que en dichas páginas hay imágenes, gráficos, animaciones, vídeos, color, texto, títulos y páginas enfatizadas, vínculos, botones, etc. Todos esos elementos serán susceptibles de ser el origen de problemas de accesibilidad.

5.4 PAUTAS DE ACCESIBILIDAD AL CONTENIDO EN LA WEB

Los documentos denominados Pautas de Accesibilidad al Contenido en la Web (incluidos en los mismos documentos que los principios recogidos en la Sección 5.3, es decir las guías WCAG 1.0 y 2.0) explican cómo hacer que el contenido Web sea accesible para personas con discapacidad. Dichas pautas han sido elaboradas por el grupo de trabajo denominado WAI (Iniciativa de Accesibilidad Web) del W3C. Es importante destacar que el término *contenido web* normalmente hace referencia a la información contenida en una página Web o en una aplicación Web, incluyendo texto, imágenes, formularios, sonido, etc., y en este documento lo utilizaremos con ese mismo significado.

Las guías WCAG, como comentamos en la Sección 5.3, son una parte de un conjunto de pautas de accesibilidad, que incluyen las Pautas de Accesibilidad para Herramientas de Autor (ATAG) y las Pautas de Accesibilidad para Agentes de Usuario (UAAG), que también las mantiene y gestiona el Consorcio W3C. Estas otras pautas no serán tratadas en este libro. En lo que resta de capítulo se tratará en profundidad las pautas contenidas en el documento WCAG 2.0.

Las Pautas de Accesibilidad de Contenido Web 2.0, que están organizadas bajo los cuatro principios comentados en la sección anterior, cubren un amplio espectro de recomendaciones para hacer el contenido web más accesible. Seguir estas pautas hará el contenido accesible para un mayor número de personas con discapacidad, las que incluyen ceguera o visión deficiente, sordera y pérdida de audición, deficiencias de aprendizaje, limitaciones cognitivas, movilidad reducida, deficiencias del lenguaje, fotosensitividad y las combinaciones de todas estas. Pero, aunque un sitio web aparentemente no vaya dirigido a personas con discapacidad, seguir estas pautas puede hacer que el contenido web sea más fácil de utilizar por personas en general. Por ello, se insiste en la conveniencia de que todo diseñador web las conozca y utilice.

Las pautas 2.0 se han redactado como sentencias que se pueden comprobar y no son específicas utilizando ninguna tecnología. Las pautas sirven para evaluar diseños, pero también para guiar acerca de cómo satisfacer los principios de diseño accesible que se vieron en la sección anterior.

Se recogen, seguidamente, las distintas pautas 2.0 organizadas en función de los principios de diseño:

Tabla 5.1 Principios y pautas de accesibilidad 2.0

Principios de diseño accesible	Pautas de accesibilidad 2.0
Perceptibilidad	<ul style="list-style-type: none">• Pauta 1.1 Proporcionar alternativas textuales para todo contenido no textual, de manera que pueda modificarse para ajustarse a las necesidades de las personas, como por ejemplo en una letra mayor, braille, voz, símbolos o un lenguaje más simple.• Pauta 1.2 Proporcionar alternativas sincronizadas para contenidos multimedia sincronizados dependientes del tiempo.• Pauta 1.3 Crear contenidos que puedan presentarse de diversas maneras (como por ejemplo una composición más simple) sin perder la información ni su estructura.• Pauta 1.4 Hacer más fácil para los usuarios ver y oír el contenido, incluyendo la separación entre primer plano y fondo.
Operatividad	<ul style="list-style-type: none">• Pauta 2.1 Hacer que toda funcionalidad esté disponible a través del teclado.• Pauta 2.2 Proporcionar a los usuarios con discapacidades el tiempo suficiente para leer y usar un contenido.• Pauta 2.3 No diseñar un contenido de manera que se sepa que puede causar ataques.• Pauta 2.4 Proporcionar medios que sirvan de ayuda a los usuarios con discapacidades a la hora de navegar, localizar contenido y determinar dónde se encuentran.
Comprendibilidad	<ul style="list-style-type: none">• Pauta 3.1 Hacer el contenido textual legible y comprensible.• Pauta 3.2 Crear páginas web cuya apariencia y operatividad sean predecibles.• Pauta 3.3 Ayudar a los usuarios a evitar y corregir errores.
Robustez	<ul style="list-style-type: none">• Pauta 4.1 Maximizar la compatibilidad con agentes de usuario actuales y futuros, incluyendo tecnologías asistivas.

En la Sección 5.5 se repasarán estas pautas de accesibilidad y se les asociarán criterios y técnicas para lograr implementar mecanismos que permitan su consecución.

ACTIVIDADES 5.4



En la actividad anterior ha elegido la página web y el sitio que quería revisar y observar. A partir de este momento utilizaremos un sitio web concreto. Será el correspondiente al sitio web de la Fundación CTIC, una institución que representa el W3C en España. Su referencia web es la siguiente: <http://fundacionctic.org>.

- Revise este sitio web y compruebe el nivel de exigencia en materia de accesibilidad que se autoimpone, y las referencias al concepto de accesibilidad que aparecen en él. Para la revisión tenga en cuenta las pautas 2.0 vistas en la sección.

SOLUCIÓN

En el sitio web considerado se muestra información de actividad y descriptiva de la institución CTIC. Esa institución está comprometida con el logro de la accesibilidad plena a todos los contenidos de su sitio web. Por ese motivo, está aplicando tecnologías estándar siguiendo las recomendaciones del W3C y respetamos las pautas de accesibilidad propuestas por la Iniciativa de Accesibilidad Web (WAI). El principal objetivo es conseguir que todos los contenidos y servicios facilitados a través de su sitio web resulten accesibles para cualquier persona, independientemente de cualquier limitación de entorno, conexión o personal que pudiera tener.

Para el diseño de su sitio web se han teniendo en cuenta las Pautas de Accesibilidad al Contenido Web (WCAG) en su versión 2.0, aplicando para ello los criterios de adecuación o exigencia de nivel doble A.

El alcance de esta declaración de conformidad se amplía a todo el sitio web de CTIC. Por ello, es un buen lugar donde ver ejemplos de qué considerar en materia de diseño accesible y cómo lograrlo. Las tecnologías usadas de forma compatible con la accesibilidad en este sitio web son: HTML, CSS y Javascript

5.5 CRITERIOS PARA SATISFACER LOS REQUISITOS DEFINIDOS EN LAS WCAG

El documento de las Pautas de Accesibilidad al Contenido Web 2.0 se ha diseñado para cubrir las necesidades de quienes necesiten un estándar técnico estable para lograr accesibilidad en los contenidos ofrecidos a través de un sitio web. Sin embargo, hay otros documentos, denominados documentos de apoyo, que se basan también en el documento de las pautas 2.0 y cumplen otros importantes propósitos adicionales. Los documentos de apoyo se componen, a su vez, de otros documentos que asisten en:

- **El cumplimiento de las pautas 2.0.** Una lista concisa y personalizable que incluye todas las pautas, criterios de éxito y técnicas que los autores pueden emplear para desarrollar y evaluar contenido web.
- **La puesta en práctica de las pautas 2.0.** Una guía para comprender e implementar las pautas 2.0. Se trata de un breve documento de comprensión para cada pauta y criterio de éxito perteneciente a las pautas así como de tópicos clave.
- **Las técnicas para satisfacer las pautas 2.0.** Una colección de técnicas y errores conocidos, cada una en un documento independiente, que incluye la descripción de la misma, ejemplos, código y pruebas.
- **La comprensión de las propias pautas 2.0.** Diagrama y descripción de cómo se relacionan y vinculan entre sí los documentos técnicos.

En los documentos anteriores, para cada pauta, se proporcionan los criterios de éxito verificables que permiten emplear las pautas 2.0 en aquellas situaciones en las que existan requisitos y necesidad de comprobación de conformidad de cara a la especificación de un diseño, una compra, el cumplimiento de una regulación o un acuerdo contractual. Con el fin de cumplir con los requisitos de los diferentes grupos y situaciones, se han definido tres niveles de adecuación: A (el más bajo), AA y AAA (el más alto).

A continuación se mostrarán las pautas y los criterios asociados a las pautas de accesibilidad al contenido web 2.0. Esta información es importante ya que cada vez es más habitual que los sitios web presenten la información de diferentes formas. Por este motivo, un diseñador de contenido web puede ocurrir que tenga que considerar distintos criterios en sus desarrollos. La Tabla 5.2 muestra los criterios asociados a cada pauta descrita en la sección anterior.

Tabla 5.2 Pautas y criterios ligados a la perceptibilidad

Pauta	Criterios a tener en cuenta (Nivel de adecuación)
1 Alternativas textuales	En un sitio web podemos encontrar diferentes formas de presentar la información y ofrecer mecanismos para interactuar, pero se ha de tener presente que si queremos lograr que el sitio web sea accesible todo contenido no textual que se presente al usuario debe contar con una alternativa textual que logre un propósito equivalente, excepto determinados supuestos (consultar el documento WCAG 2.0 guidelines) (A)
2 Contenido multimedia dependiente del tiempo	En un sitio web puedes incluir audio y vídeo, pero ese tipo de contenido puede dar lugar a problemas de accesibilidad para determinados colectivos y esos problemas podrán responder a distintos tipos de niveles de adecuación: <ul style="list-style-type: none"> • Solo audio y solo vídeo (pregrabado) (A) • Subtítulos (pregrabados) (A) • Audiodescripción o alternativa multimedia (pregrabada) (A) • Subtítulos (directo) (AA) • Audiodescripción (pregrabada) (AA) • Lengua de signos (pregrabada) (AAA) • Audiodescripción extendida (pregrabada) (AAA) • Alternativa multimedia (pregrabada) (AAA) • Solo audio (directo) (AAA)
3 Adaptabilidad	<ul style="list-style-type: none"> • Información y relaciones (A) • Secuencia significativa (A) • Características sensoriales (A)
4 Distingurable	<ul style="list-style-type: none"> • Empleo del color (A) • Empleo del Audio (A) • Contraste (mínimo) (AA) • Variar el tamaño del texto (AA) • Imágenes de texto (AA) • Contraste (mejorado) (AAA) • Fondo de audio bajo o inexistente (AAA) • Presentación visual (AAA) • Imágenes de texto (sin excepción) (AAA)

Más adelante, en este mismo capítulo, se recogerán recomendaciones y sugerencias más concretas para el logro en el cumplimiento de las pautas de accesibilidad web.

ACTIVIDADES 5.5



► Localice en el sitio web del Consorcio de la W3C (<http://www.w3c.es/>) las limitaciones asociadas al criterio ligado a la pauta 1.1. Alternativas textuales.

SOLUCIÓN

Los supuestos que están exentos del cumplimiento de ofrecer una alternativa textual son los siguientes:

- a. Controles, entrada de datos: si el contenido no textual es un control o acepta datos de entrada del usuario, entonces debe tener un nombre que describa su propósito.
- b. Contenido multimedia dependiente del tiempo: si el contenido no textual es contenido multimedia dependiente del tiempo, entonces el texto proporciona al menos una descripción identificativa del contenido no textual.
- c. Prueba: si el contenido no textual es una prueba o ejercicio que pudiera resultar inválido al presentarse como texto, entonces el texto alternativo proporciona al menos una descripción identificativa del contenido no textual.
- d. Experiencia sensorial: si el contenido ha sido creado principalmente para proporcionar una experiencia sensorial específica, entonces el texto proporciona al menos una descripción identificativa del contenido no textual.
- e. CAPTCHA: si el propósito del contenido no textual es confirmar si al contenido está accediendo un humano y no un ordenador, entonces los textos alternativos identifican y describen el propósito del contenido no textual, y se proporcionan maneras alternativas de CAPTCHA con emisiones dirigidas a distintos sentidos que se ajusten a distintas discapacidades.
- f. Decoración, formato, invisible: si el contenido no textual es pura decoración, se emplea exclusivamente por una cuestión de formato visual o no se presenta a los usuarios, entonces se ha implementado de manera que pueda ser ignorado por las tecnologías asistivas.

Algunos de los elementos anteriores aparecen en la página web de la Fundación CTIC y, efectivamente, no se aporta información alternativa o textual equivalente para esos datos.

En la Tabla 5.3 se recogen las pautas y los criterios asociados al principio de operatividad. Esta información es útil, por ejemplo, si como diseñador, se quiere lograr facilidad de operación en un sitio web. Para ello se debe tener en cuenta que si se ofrece información o actividades que únicamente son accesibles a través del teclado, habrá personas que pueden tener dificultades para acceder a ella o les resultará imposible.

Tabla 5.3 Pautas y criterios a considerar relacionados con la operatividad

Pauta	Criterio (Nivel de adecuación)
1 Accesible a través del teclado	<ul style="list-style-type: none"> • Teclado (A) • Teclado no bloqueado (A) • Teclado (sin excepción) (AAA)
2 Tiempo suficiente	<ul style="list-style-type: none"> • Límite de tiempo ajustable (A) • Pausar, detener, ocultar (A) • Sin tiempo (AAA) • Interrupciones (AAA) • Reautenticación (AAA)
3 Efectos visuales	<ul style="list-style-type: none"> • Tres destellos o por debajo del umbral (A) • Tres destellos (AAA)
4 Navegable	<ul style="list-style-type: none"> • Saltar bloques (A) • Página titulada (A) • Orden de foco (A) • Propósito de los enlaces (en su contexto) (A) • Múltiples medios (AA) • Encabezados y etiquetas (AA) • Foco visible (AA) • Ubicación (AAA) • Propósito de un enlace (vínculo solo) (AAA) • Encabezados de sección (AAA)

Un sitio web que ofrezca facilidades de comprensión desde el punto de vista de la accesibilidad tendrá presentes las pautas de legibilidad, predecibilidad y ofrecerá ayuda a la entrada de datos. Cada una de esas pautas podrá considerar diferentes criterios para su logro. Esas pautas y criterios se recogen en la Tabla 5.4.

Tabla 5.4 Pautas y criterios asociados a la facilidad de comprensión

Pauta	Criterio (Nivel de adecuación)
1 Legible	<ul style="list-style-type: none"> • Idioma de la página (A) • Idioma de partes (AA) • Palabras inusuales (AAA) • Abreviaturas (AAA) • Nivel de lectura (AAA) • Pronunciación (AAA)
2 Predecible	<ul style="list-style-type: none"> • Con foco (A) • Cambios imprevistos (A) • Navegación consistente (AA) • Identificación consistente (AA) • Solicitud de cambio (AAA)
3 Ayuda a la entrada de datos	<ul style="list-style-type: none"> • Identificación de errores (A) • Instrucciones o etiquetas (A) • Sugerencia tras error (AA) • Prevención de errores (legales, financieros, de datos) (AA) • Ayuda (AAA) • Prevención de errores (todo error) (AAA)

El último de los principios de accesibilidad es que el sitio web sea robusto y para ello debe cuidarse el lenguaje utilizado en su confección y tener especialmente presente que sea compatible con sus usuarios potenciales. Los criterios ligados al principio de robustez se han recogido en la Tabla 5.5.

Tabla 5.5 Pautas y criterios asociados al principio de robustez

Pauta	Criterio (Nivel de adecuación)
Compatible	<ul style="list-style-type: none">• Análisis (A)• Nombre, rol, valor (A)

ACTIVIDADES 5.6



➤ Revise el sitio web de la Fundación CTIC y cargue dicho sitio habilitando y deshabilitando las opciones de bloqueo de ventanas emergentes, carga de imágenes o activar javascript. También puede visualizar y navegar a través del sitio cargando o no las páginas con los colores originales o sin color.

La mayoría de los navegadores disponibles permitirán activar y desactivar dichas opciones.

SOLUCIÓN

Esta actividad posibilita interactuar con un sitio web, acceder a la información que éste proporciona y ser conscientes de cómo influye la carencia de contenidos y efectos de decoración en la información que éste ofrece. Por ejemplo, ver la página web sin colores, sin imágenes o sin vídeos, puede hacer que haya información que no pueda ser consultada o que toda la información o facilidades de interacción no estén disponibles.

Esta carencia en el acceso a la información no se presentará en el sitio web de la Fundación CTIC, pero sí que puede estar presente en otros muchos sitios web menos preocupados en cuestiones de accesibilidad web.

5.6 PRIORIDADES. PUNTOS DE VERIFICACIÓN. NIVELES DE ADECUACIÓN

En las secciones anteriores se han identificado principios, pautas y criterios que se tienen que tener en cuenta cuando se desarrollan sitios web accesibles. Tener en cuenta dichos principios, pautas y criterios se va a traducir en escribir nuestras páginas y sitios web de una forma u otra. En esta sección, y utilizando los principios y las pautas como elementos de organización, se recogen una serie de recomendaciones que todo diseñador debería tener presentes a la hora de diseñar sitios web accesibles. Estas recomendaciones se recogen atendiendo a que esos criterios deben ser logrados y verificar que así se ha hecho. Es decir, las siguientes secciones matizan más concretamente el grado de consecución de los criterios de accesibilidad.

5.6.1 PERCEPTEBILIDAD

El contenido web debe estar disponible y ser perceptible para los sentidos – vista, audición, y/o tacto. Ello implica tener en cuenta distintas pautas y criterios y atender a distintas recomendaciones.

Pauta 1.1. Alternativas textuales. Ofrezca alternativas en forma de texto para todo el contenido no textual

La Web es fundamentalmente información, en ocasiones esa información se puede presentar utilizando una imagen, un gráfico o un vídeo. En esas ocasiones se tendrá que garantizar que todo contenido no textual tenga y disponga de su equivalente textual.

Criterios de éxito	Nivel	Recomendaciones
1.1.1 Contenido no textual	A	<ul style="list-style-type: none">Todas las imágenes, botones de imagen de los formularios y las zonas activas de los mapas de imagen, tendrán un texto alternativo adecuado.Las imágenes que no transmitan contenidos, sean decorativas o con el contenido ya presente como texto se ofrecerán con el texto alternativo vacío (<code>alt=""</code>) o aplicadas como fondos de imagen CSS. Todas las imágenes enlazadas contarán con un texto descriptivo alternativo.El contenido equivalente alternativo para las imágenes complejas se ofrecerá en una página (enlazada o referenciada mediante <code>longdesc</code>) aparte.Los botones de los formularios tendrán nombres (<code>value</code>) descriptivos.Los elementos de los formularios tendrán etiquetas textuales (<code>label</code>) asociadas o, si éstas no pueden utilizarse, un título (<code>title</code>) descriptivo.Los elementos multimedia incrustados (<code>embedded</code>) se identificarán mediante textos accesibles.Los marcos (frames) tendrán un título apropiado.

Pauta 1.2. Contenido dependiente del tiempo: ofrezca alternativas para los contenidos que dependan del tiempo

Si un sitio web contiene animaciones o breves presentaciones incluidas en su interior se deberá garantizar que dichas animaciones sean accesibles para todos sus visitantes. Seguidamente se recogen una serie de recomendaciones relacionadas con estos escenarios.

Criterios de éxito	Nivel	Recomendaciones
1.2.1. Solo audio y solo vídeo pregrabado	A	<ul style="list-style-type: none"> • Se ofrecerá una transcripción descriptiva (incluyendo todas las pistas e indicadores visuales y auditivos) para el audio grabado (no en directo) basado en web (<i>podcast</i> de audio, archivos <i>mp3</i>, etc.). • Se ofrecerá una descripción auditiva o textual para los vídeos grabados (no en directo) sin audio basados en web (por ejemplo, vídeos que no incluyen pistas de audio).
1.2.2 Subtítulos (Pregrabados)	A	<ul style="list-style-type: none"> • Se ofrecerán subtítulos para los vídeos grabados (no en directo) basados en web (vídeos de YouTube, etc.).
1.2.3. Audio-descripciones o Contenidos "media" alternativos (Pregrabados)	A	<ul style="list-style-type: none"> • Se ofrecerá una transcripción o audio descripción de los vídeos basados en web grabados (no en directo).
1.2.4. Subtitulado (En directo)	AA	<ul style="list-style-type: none"> • Se ofrecerán subtítulos sincronizados con el audio para todo el contenido multimedia ofrecido en directo (emisiones solo audio, web cast, videoconferencias, animaciones Flash, etc.).
1.2.5. Audio descripción (Pregrabado)	AA	<ul style="list-style-type: none"> • Se ofrecerán audio descripciones para todo el contenido de vídeo. Nota: solo será necesario si el vídeo transmite contenido visual que no está disponible por defecto en la pista de audio.
1.2.6. Lengua de signos (Pregrabada)	AAA	<ul style="list-style-type: none"> • Se ofrecerá un vídeo en lengua de signos para todo el contenido "media" que contenga audio.
1.2.7. Audio descripción extendida (Pregrabada)	AAA	<ul style="list-style-type: none"> • Cuando una pista de audio descripción no se pueda añadir al vídeo debido a la sincronización del audio (por ejemplo, no existen pausas en el audio), se proporcionarán una versión alternativa del vídeo con pausas que permitan las descripciones de audio.
1.2.8. Alternativas multimedia (Pregrabado)	AAA	<ul style="list-style-type: none"> • Se ofrecerá una transcripción descriptiva para todos los medios pregrabados que contengan una pista de vídeo.
1.2.9. Solo audio (directo)	AAA	<ul style="list-style-type: none"> • Se ofrecerá una transcripción descriptiva (por ejemplo, el guión de una presentación en vivo de audio) para todos los contenidos en directo que contengan audio.

Pauta 1.3. Adaptabilidad: cree contenido que pueda presentarse de diferentes maneras (por ejemplo, un diseño simplificado) sin perder la información o estructura

Criterios de éxito	Nivel	Recomendaciones
1.3.1 Información y relaciones	A	<ul style="list-style-type: none"> El marcado semántico se usará para designar los encabezados (<code><h1></code>), listas (<code></code>, <code></code>, and <code><dl></code>), texto especial o enfatizado (<code></code>, <code><code></code>, <code><abbr></code>, <code><blockquote></code>, por ejemplo), etc. El marcado semántico deberá usarse apropiadamente. Las tablas se usarán para marcar los datos tabulados. Las celdas de datos (<code><td></code>) se asociarán con sus encabezados (<code><th></code>) donde sea necesario. Los títulos de las tablas (<code>caption</code>) y sus resúmenes (<code>summary</code>) se usarán de forma apropiada. Las etiquetas (<code>label</code>) textuales se asociarán con sus campos (<code>input</code>) correspondientes en los formularios. Los elementos de los formularios que estén relacionados se agruparán mediante <code>fieldset/legend</code>.
1.3.2 Secuencia significativa	A	<ul style="list-style-type: none"> El orden de navegación y lectura (determinado por el orden en el código fuente) será lógico e intuitivo.
1.3.3 Características sensoriales	A	<ul style="list-style-type: none"> Las instrucciones no dependerán de la forma, tamaño o ubicación visual (por ejemplo, "Haga clic en el ícono cuadrado para continuar" o "Las instrucciones están en la columna de la derecha"). Las instrucciones no dependerán del sonido (por ejemplo, "Un sonido beep le indica que puede continuar").

Pauta 1.4. Distinguible: facilite a los usuarios el ver y escuchar el contenido, incluyendo la separación entre el primer plano y el fondo

Criterio de éxito	Nivel	Recomendaciones
1.4.1 Empleo del color	A	<ul style="list-style-type: none"> No use el color como el único método para transmitir el contenido o distinguir elementos visuales. Los enlaces deben distinguirse de los elementos y texto que les rodean. Si utiliza el color para diferenciar los enlaces, use una forma adicional para distinguirlos. (por ejemplo, se subrayan cuando reciben el foco).
1.4.2 Empleo del audio	A	<ul style="list-style-type: none"> Se debe ofrecer un mecanismo para poder parar, pausar, silenciar o ajustar el volumen de cualquier sonido que se reproduzca automáticamente en la página más de tres segundos.

1.4.3 Contraste (mínimo)	AA	<p>El texto o las imágenes de texto deben tener una relación de contraste de al menos 4.5:1, excepto en los siguientes casos:</p> <ul style="list-style-type: none"> • En los textos grandes (de más de 18 puntos o 14 puntos en negrita) y las imágenes de texto grandes la relación de contraste debe ser de al menos 3:1. • En los textos, o las imágenes de texto, que forman parte de un componente de la interfaz de usuario inactivo, que son meramente decorativos, que no son visibles o que forman parte de una imagen cuyo significado es visual, no tienen un requisito mínimo de contraste. • Los textos que forman parte de un logotipo o de una marca comercial no tiene un requisito mínimo de contraste.
1.4.4 Variar el tamaño del texto	AA	<ul style="list-style-type: none"> • La página deberá ser legible y funcional cuando se doble el tamaño del texto.
1.4.5 Imágenes de texto	AA	<ul style="list-style-type: none"> • Si la misma representación visual puede realizarse usando solo texto, no deben usarse imágenes para representar ese texto.
1.4.6 Contraste (aumentado)	AAA	<ul style="list-style-type: none"> • El texto o las imágenes de texto deben tener una relación de contraste de al menos 7:1. • Los textos grandes (de más de 18 puntos o 14 puntos en negrita) deben tener una relación de contraste de al menos 4.5:1.
1.4.7 Bajo o sin sonido de fondo	AAA	<ul style="list-style-type: none"> • Compruebe que no hay o existe un ruido de fondo muy bajo que permita distinguir fácilmente las conversaciones.
1.4.8 Presentación visual	AAA	<p>Para bloques de texto de más de una frase de longitud:</p> <ul style="list-style-type: none"> • No habrá más de 80 caracteres de ancho. • No estarán justificados a ambos lados (alineados los márgenes izquierdo y derecho). • Tendrán un interlineado (de al menos la mitad de la altura del texto) y espacio entre párrafos (1.5 veces la medida del interlineado) adecuado. • Tendrán especificados un color de primer plano y fondo. Estos se pueden aplicar a elementos específicos de la página o en su totalidad utilizando CSS (y, por tanto, heredados por el resto de elementos). • No aparecerá desplazamiento horizontal cuando se doble el tamaño del texto.
1.4.9 Imágenes de texto (sin excepción)	AAA	<ul style="list-style-type: none"> • Solo se usarán imágenes de texto para decorar cuando no transmitan información o cuando la información no pueda presentarse de ninguna otra manera (por ejemplo, cuando el texto forme parte del logotipo de una empresa).

5.6.2 OPERATIVIDAD

Los formularios, controles, navegación y otros elementos de la interfaz deben permitir la interacción de los usuarios, independientemente de las características de estos.

Pauta 2.1. Accesibilidad a través del teclado: permita que toda la funcionalidad esté disponible usando el teclado

Criterios de éxito	Nivel	Recomendaciones
2.1.1 Teclado	A	<ul style="list-style-type: none"> Todas las funciones de las páginas deberán estar disponibles utilizando el teclado, excepto aquellas que de forma conocida no pueden realizarse con el teclado (por ejemplo, un dibujo a mano alzada). Los atajos de teclado y accesskeys (que normalmente deberían evitarse) no deben entrar en conflicto con las presentes en el navegador y/o lector de pantalla.
2.1.2 Teclado no bloqueado	A	<ul style="list-style-type: none"> El foco del teclado no deberá estar bloqueado o fijado en un elemento concreto de la página. El usuario deberá poder moverse por todos los elementos navegables de la página utilizando únicamente el teclado.
2.1.3 Teclado (Sin excepción)	AAA	<ul style="list-style-type: none"> Toda la funcionalidad de las páginas deberán estar disponibles utilizando el teclado.

Pauta 2.2. Tiempo suficiente: ofrezca a los usuarios el tiempo suficiente para que puedan leer y utilizar el contenido

Criterios de éxito	Nivel	Recomendaciones
2.2.1 Tiempo ajustable	A	<ul style="list-style-type: none"> Si una página o aplicación tiene un límite de tiempo para realizar una tarea deberá ofrecer la opción de apagar, ajustar o aumentar ese límite de tiempo. No es un requisito para eventos en tiempo real (por ejemplo, una subasta) donde el límite de tiempo es absolutamente necesario, o si el plazo de tiempo es de más de 20 horas.
2.2.2 Pausar, parar, ocultar	A	<ul style="list-style-type: none"> Todo movimiento automático, parpadeo o desplazamiento de más de tres segundos deberá poderse pausar, parar u ocultar por el usuario. El movimiento, parpadeo, o desplazamiento podrá usarse para llamar la atención del usuario o destacar un contenido si dura menos de tres segundos. El contenido actualizado automáticamente (por ejemplo, una página recargada o redireccionada automáticamente, un ticker de noticias, la actualización de un campo mediante AJAX, un aviso, etc.) deberá poder ser pausado, parado u ocultado por el usuario o el usuario deberá poder controlar manualmente los tiempos de actualización.
2.2.3 Sin tiempo	AAA	<ul style="list-style-type: none"> El contenido y funcionalidad no tendrá limitaciones de tiempo.
2.2.4 Interrupciones	AAA	<ul style="list-style-type: none"> Las interrupciones (alertas, actualizaciones de las páginas, etc.) deberán poder ser pospuestas o canceladas por el usuario.
2.2.5 Reautenticación	AAA	<ul style="list-style-type: none"> Si la autenticación en una sesión termina (expira), el usuario podrá reautenticarse y continuar con su actividad sin perder ningún dato de la página actual.

Pauta 2.3. Efectos visuales: no diseñe los contenidos de tal forma que puedan provocar ataques o convulsiones

Criterios de éxito	Nivel	Recomendaciones
2.3.1 Tres destellos o por debajo del umbral	A	<ul style="list-style-type: none"> • No deberá crear contenidos que destellen más de tres veces por segundo a menos que el parpadeo sea lo suficientemente pequeño, los destellos sean de bajo contraste y no contengan demasiado rojo.
2.3.2 Tres destellos	AAA	<ul style="list-style-type: none"> • No deberá crear contenidos que destellen más de tres veces por segundo.

Pauta 2.4. Navegable: ofrezca métodos que ayuden al usuario a navegar, encontrar el contenido y determinar dónde se encuentra

Criterios de éxito	Nivel	Recomendaciones
2.4.1 Saltar bloques	A	<ul style="list-style-type: none"> • Se ofrecerá un enlace para saltar la navegación y otros elementos que se repitan en todas las páginas. • Si una página cuenta con una estructura adecuada de encabezados, puede considerarse una técnica suficiente en lugar de un enlace del tipo "Ir al contenido principal". Tenga en cuenta que la navegación por encabezados todavía no está soportada en todos los navegadores. • Si una página utiliza un conjunto de marcos (<i>frameset</i>) y los marcos (<i>frame</i>) están apropiadamente titulados, puede considerarse una técnica suficiente para acceder directamente a cada marco individual.
2.4.2 Página titulada	A	<ul style="list-style-type: none"> • La página web deberá tener un título descriptivo e informativo de la misma.
2.4.3 Orden de foco	A	<ul style="list-style-type: none"> • El orden de la navegación por los enlaces, elementos de los formularios, etc., deberá ser lógico e intuitivo.
2.4.4 Propósito de los enlaces (en su contexto)	A	<ul style="list-style-type: none"> • Siempre que no sean ambiguos para los usuarios en general, los enlaces (o botones de imagen en un formulario, o zonas activas en un mapa de imagen) serán lo suficientemente descriptivos como para identificar su propósito (objetivo) directamente desde el texto enlazado o, en su caso, desde el enlace en su contexto (por ejemplo, en los párrafos que lo rodean, elementos de una lista, celdas o encabezados en una tabla, etc.). • Los enlaces (o botones de imagen en un formulario) con el mismo destino deberían tener las mismas descripciones (ser consistentes, según el criterio de éxito 3.2.4), pero los enlaces con diferentes propósitos y destinos deberían tener diferentes descripciones.

2.4.5 Múltiples medios	AA	<ul style="list-style-type: none"> Se deben ofrecer múltiples formas para encontrar otras páginas web en el sitio – al menos dos de las siguientes: una lista de páginas relacionadas, tabla de contenidos, mapa web, búsqueda en el sitio, o un listado de todas las páginas web.
2.4.6 Encabezados y etiquetas	AA	<ul style="list-style-type: none"> Los encabezados (<code><h></code>) de las páginas y las etiquetas (<code><label></code>) para los controles interactivos de los formularios deberán ser informativos. Evite el duplicar los encabezados (por ejemplo, "Más detalles") y las etiquetas de texto (por ejemplo, "primer nombre") a menos que la estructura ofrezca una diferenciación adecuada entre ellas.
2.4.7 Foco visible	AA	<ul style="list-style-type: none"> Compruebe que es visualmente evidente el elemento que tiene el foco actual del teclado (por ejemplo, si se mueve con el tabulador por la página, puede ver dónde se encuentra).
2.4.8 Ubicación	AAA	<ul style="list-style-type: none"> Si la página web forma parte de una secuencia de páginas o está dentro de un sitio con una estructura compleja, deberá indicar la ubicación de la página actual, por ejemplo, a través de las migas de pan (breadcrumbs) o especificando el paso actual en la secuencia (por ejemplo, "Paso 2 de 5 – dirección de envío").
2.4.9 Propósito de los enlaces (enlaces sin contexto)	AAA	<ul style="list-style-type: none"> Siempre que no sean ambiguos para los usuarios en general, los enlaces (o botones de imagen en un formulario, o zonas activas en un mapa de imagen) serán lo suficientemente descriptivos como para identificar su propósito (objetivo) directamente desde el texto enlazado. No deberán existir enlaces (o botones de imagen en un formulario) con el mismo texto que vinculen a lugares diferentes (por ejemplo, "Lea más").
2.4.10 Encabezados de sección	AAA	<ul style="list-style-type: none"> Además de proporcionar un documento con la estructura global del sitio, cada una de las secciones de contenido deberán ser designadas mediante encabezados (títulos), donde sea oportuno.

5.6.3 COMPRENSIBILIDAD

El contenido y la interfaz deben poder entenderse fácilmente y ser semánticamente ricos a los usuarios, independientemente de las características de estos.

Pauta 3.1. Legibilidad: cree contenidos legibles y fáciles de entender

Criterios de éxito	Nivel	Recomendaciones
3.1.1 Idioma de la página	A	<ul style="list-style-type: none"> El idioma principal de la página deberá estar identificado utilizando el atributo <code>lang</code> de HTML (por ejemplo, <code><HTML lang="es"></code>).
3.1.2 Idioma de partes	AA	<ul style="list-style-type: none"> Si algunas secciones tienen contenidos en un idioma diferente al principal, éste deberá estar identificado utilizando el atributo <code>lang</code> (por ejemplo, <code><blockquote lang="en"></code>) cuando sea apropiado. Existen algunas excepciones: nombres propios, términos técnicos, palabras o frases en un lenguaje indeterminado o inventado, locuciones propias de la lengua (vernaculares) que se entienden dentro del contexto (por ejemplo, locuciones latinas en español).
3.1.3 Palabras inusuales	AAA	<ul style="list-style-type: none"> Las palabras que puedan ser ambiguas, desconocidas o usadas de una forma muy específica, deberán definirse través de un texto adyacente, una lista de definiciones, un glosario, o de cualquier otro método.
3.1.4 Abreviaturas	AAA	<ul style="list-style-type: none"> La explicación para las abreviaturas se realizará, usando el elemento <code><abbr></code> o enlazando a un glosario de términos, la primera vez que se utilicen en el contenido.
3.1.5 Nivel de lectura	AAA	<ul style="list-style-type: none"> Una alternativa para hacer los contenidos más comprensibles es suponer que aquellos que sean más avanzados puedan ser razonablemente leídos por una persona con aproximadamente 9 años de educación primaria.
3.1.6 Pronunciación	AAA	<ul style="list-style-type: none"> Si la pronunciación de una palabra es vital para comprenderla, su pronunciación se mostrará seguida de dicha palabra o mediante un enlace a un glosario.

Pauta 3.2. Predecible: cree páginas web que se muestren y funcionen de forma previsible

Criterios de éxito	Nivel	Recomendaciones
3.2.1 Con foco	A	<ul style="list-style-type: none"> Cuando un elemento reciba el foco no se deberá iniciar un cambio en la página que confunda o desorienta al usuario.
3.2.2 Cambios imprevistos	A	<ul style="list-style-type: none"> Deberá advertir al usuario con antelación de los cambios, imprevistos o automáticos, en la configuración de cualquier elemento de la interfaz que causen una modificación en la página.
3.2.3 Navegación consistente	AA	<ul style="list-style-type: none"> Los enlaces de navegación que se repiten en las páginas web no deberían modificar su orden al navegar por el sitio.
3.2.4 Identificación consistente	AA	<ul style="list-style-type: none"> Los elementos que tienen la misma funcionalidad a través de múltiples páginas web deberán identificarse de manera consistente. Por ejemplo, un campo de búsqueda en la parte superior de la página deberá etiquetarse siempre de la misma forma.
3.2.5 Solicitud de cambio	AAA	<ul style="list-style-type: none"> Los cambios sustanciales de las páginas, la aparición de ventanas emergentes (<i>popups</i>), los cambios no controlados del foco del teclado, o cualquier otro cambio que podría confundir o desorientar al usuario deberán ser iniciados por éste. Alternativamente, siempre se le deberá ofrecer al usuario una opción para desactivar dichos cambios.

Pauta 3.3. Asistencia en la introducción de datos: ayude a los usuarios a evitar y corregir los errores

Criterios de éxito	Nivel	Recomendaciones
3.3.1 Identificación de errores	A	<ul style="list-style-type: none"> Ofrezca información al usuario sobre los campos obligatorios de un formulario, o aquellos que necesitan un formato, valor o longitud específica, utilizando el elemento <label> (si éste no está disponible ponga la información en el atributo de título title del elemento). Si se usa la validación de datos de los formularios (del lado del cliente o del servidor), ofrezca la información sobre los errores y avisos de forma eficiente, intuitiva y accesible. Los errores deben estar claramente identificados, ofrecer un acceso rápido al elemento problemático, permitir que el usuario pueda fácilmente solucionar el error y reenviar los datos del formulario.
3.3.2 Etiquetas o instrucciones	A	<ul style="list-style-type: none"> Se deberán proporcionar las suficientes etiquetas, avisos e instrucciones necesarios para los elementos interactivos. Use para ello instrucciones, ejemplos, posicione adecuadamente las etiquetas (<i>label</i>) y agrupe e identifique los campos con <i>fieldsets/legends</i>
3.3.3 Sugerencias de error	AA	<ul style="list-style-type: none"> Si se detecta un error al introducir un dato (mediante la validación en el lado del cliente o en el del servidor), deberá proporcionar sugerencias para solucionar el problema de forma oportuna y accesible.
3.3.4 Prevención de errores (Legales, financieros, de datos)	AA	<ul style="list-style-type: none"> Si el usuario puede modificar o eliminar datos de carácter legal, financiero o de prueba, estas acciones deberán ser reversibles, verificadas o comprobadas.
3.3.5 Ayuda	AAA	<ul style="list-style-type: none"> Si el usuario puede enviar, cambiar o eliminar información, la información deberá poder volver a estar disponible, y/o las acciones realizadas ser verificadas o confirmadas.
3.3.6 Prevención de errores (todos)	AAA	<ul style="list-style-type: none"> Si el usuario puede enviar información, el envío deberá poder ser reversible, verificado o confirmado.

5.6.4 ROBUSTEZ

El contenido debe ser lo suficientemente consistente y fiable como para permitir su uso con una amplia variedad de agentes de usuario, ayudas técnicas y preparado para las tecnologías venideras.

Pauta 4.1. Compatible: mejore la compatibilidad con los agentes de usuario actuales y futuros, incluidas las ayudas técnicas

Criterios de éxito	Nivel	Recomendaciones
4.1.1 Análisis	A	<ul style="list-style-type: none"> Se deberán evitar los errores de sintaxis de HTML/XHTML. El código puede comprobarse, analizarse y validarse a través de http://validator.w3.org/
4.1.2 Nombre, rol, valor	A	<ul style="list-style-type: none"> Deberá utilizar el marcado de tal forma que se facilite la accesibilidad. Esto incluye el seguir las especificaciones oficiales de HTML/XHTML, utilizando la gramática formal de forma apropiada.

ACTIVIDADES 5.7



➤ Teniendo en cuenta los principios, pautas, criterios y niveles de adecuación presentados en las secciones anteriores, y disponiendo de la información accesible a través de la Web en <http://www.w3.org/TR/WCAG/>, sobre las guías de accesibilidad al contenido web, consulte la información complementaria a cada una de las pautas y criterios. Dicha información permitirá comprender mejor las pautas, los criterios y cómo cumplir con dichos criterios.

5.7 MÉTODOS PARA REALIZAR REVISIONES PRELIMINARES Y EVALUACIONES DE ADECUACIÓN O CONFORMIDAD DE DOCUMENTOS WEB

Los puntos de verificación recogidos con las tablas anteriores permiten a un diseñador web abordar dos actividades:

- Por un lado, guiar el proceso de desarrollo de sitios web. Es decir, saber si el sitio web que se desarrolla cumple con los criterios de accesibilidad que se han propuesto.
- Por otro, evaluar diseños web atendiendo a criterios de accesibilidad. Evidentemente, un diseñador experto en evaluar la accesibilidad web de los sitios que desarrolla es también experto en saber si otros diseños web ajenos cumplen o no esas pautas de accesibilidad.

En la siguiente sección se identificarán herramientas que facilitan la evaluación de sitios web y la comprobación del cumplimiento de las recomendaciones de un diseño web accesible. Sin embargo, antes de ver esas herramientas es necesario establecer un proceso de evaluación (pasos a seguir) que faciliten el análisis y valoración de una web acorde a su accesibilidad, ayudando al diseñador a saber cómo evaluar la accesibilidad.

Una propuesta de pasos a seguir como proceso de evaluación de la accesibilidad de un sitio web es la siguiente:

1 Seleccionar una muestra representativa de diferentes páginas del sitio web. Entre ellas estará la página principal y otras páginas que se consideren especialmente importantes en función de la temática del sitio web que se esté diseñando y evaluando.

2 Evaluar las páginas con navegadores gráficos con diferentes configuraciones. Se deberán visualizar las páginas elegidas con diferentes navegadores, resoluciones de pantalla, y configuraciones (carga de imágenes, habilitación de sonidos, javascript, etc.)

3 Usar un navegador de voz o un navegador en modo texto. Las personas con discapacidad visual utilizan herramientas que leen las páginas web. Los lectores de pantalla (*screen readers* en inglés) son un software que permite la utilización del sistema operativo y las distintas aplicaciones mediante el empleo de un sintetizador de voz que “lee y explica” lo que se visualiza en la pantalla, lo que supone una ayuda para las personas con graves problemas de visión o completamente ciegas. Ejemplos de estos lectores son: *BrowseAloud*⁹⁵, *JAWS*⁹⁶, *CLiCk Speak*⁹⁷, y *Dolphin Hal*⁹⁸, entre otros.

4 Utilizar dos herramientas generales de evaluación de accesibilidad. Dichas herramientas, aunque no tendrán capacidad para identificar todos los problemas de accesibilidad e incluso pueden informarnos de falsos errores (como se comentará en la Sección 5.8) sí resultan de gran ayuda para revisar y comprobar el código asociado a las páginas evaluadas.

5 Resumir los resultados: muchas de las herramientas utilizadas para abordar el punto anterior permiten generar informes, en cualquier caso no estará de más, recopilar de una forma ordenada y consistente aquellos problemas que se hayan identificado, cómo fueron encontrados y las medidas que se hayan tomado para subsanar dichos inconvenientes.

ACTIVIDADES 5.8



- Siguiendo la propuesta metodológica recogida en esta sección, evalúa la conformidad del documento de la página web de la Fundación CTIC con las pautas de accesibilidad al contenido web.

SOLUCIÓN

- Utilizaremos la página principal como página representativa del sitio web. Si hubiera otras páginas que recogieran contenido especialmente sensible deberían contemplarse también en las actividades de evaluación.
- Veremos más adelante que existen sitios y páginas web con los que podemos conseguir capturas de pantalla un sitio web y percibir cómo se visualizan dichas páginas, sin necesidad de tener instalados distintos navegadores. Sin embargo, en esta tarea se debe trabajar con tres navegadores instalados en local: Firefox, Explorer y Chrome. Las Figuras 5.2, 5.3 y 5.4 muestran capturas del sitio web en diferentes navegadores.

⁹⁵ <http://www.browsealoud.com/>

⁹⁶ <http://www.freedomscientific.com/products/fs/jaws-product-page.asp>

⁹⁷ <http://clickspeak.clcwORLD.net/>

⁹⁸ <http://www.yourdolphin.co.uk/productdetail.asp?id=5>



Figura 5.2. Página web de la Fundación CTIC en MS Explorer



Figura 5.3. Página web de la Fundación CTIC en Firefox



Figura 5.4. Página web de la Fundación CTIC en Chrome

En todos los navegadores mencionados es posible habilitar y deshabilitar la carga de imágenes, javascripts, colores, etc. Habilitando y deshabilitando dichas características de un sitio web es posible comprobar si hay información que se transmite utilizando esos mecanismos y se pierde al no estar disponibles.

En el caso de la página web de la Fundación CTIC no hay pérdida de información. Si se accede al sitio web desde los tres navegadores se observa que ha sido diseñado de tal forma que trata de cumplir con la mayoría de pautas y recomendaciones para el logro de una web accesible.

Otras herramientas, como las mencionadas anteriormente, permiten completar el proceso de evaluación de accesibilidad de un sitio web comprobando si es posible navegar y acceder al contenido de dicho sitio web, utilizando herramientas que leen dicho contenido. En la sección se han sugerido distintas herramientas que permiten realizar dicha actividad. De casi todas ellas existen versiones *trial*. Por ejemplo, en la página web de la herramienta BrowseAloud (<http://www.browsealoud.com/>) puede descargarse una versión gratuita de la herramienta y utilizarse para comprobar cómo podría acceder al sitio web de la Fundación CTIC una persona con ceguera.

Complementaremos los puntos adicionales sugeridos para llevar a cabo la evaluación en la próxima tarea, una vez hayamos presentado herramientas adicionales que soportan el análisis de accesibilidad de sitios web.

5.8 HERRAMIENTAS DE ANÁLISIS DE ACCESIBILIDAD WEB

Existen herramientas que permiten identificar de forma automática problemas de accesibilidad. Pero hay que ser conscientes de que las herramientas de validación automática no son suficientes para asegurar que un sitio Web es 100% accesible. Comprobar la accesibilidad es un proceso tan complejo que a día de hoy no puede ser automatizado en su totalidad. Hay muchas herramientas que, con la intención de automatizar en mayor grado la evaluación de la accesibilidad, dan *falsos positivos* (considerar como error algo que no lo es) o no detectan algunos errores que el usuario debe revisar manualmente.

Para cubrir las limitaciones que presentan estas herramientas de análisis y evaluación automáticas se recomienda complementar su análisis con evaluaciones manuales, que permitan tratar los falsos positivos. Muchas de las herramientas disponibles se ofrecen *on line* y otras se pueden utilizar desde distintos navegadores, en las siguientes secciones se mencionan las herramientas más reseñables.

5.8.1 SOFTWARE Y HERRAMIENTAS ON LINE

En función de las recomendaciones recopiladas en este capítulo, se puede observar que muchas de ellas se traducen de una u otra manera en la escritura de código HTML. Por ello, la evaluación de la accesibilidad de un sitio web puede pasar, en primera instancia, por analizar las páginas atendiendo al código HTML.

Un primer conjunto de herramientas de análisis de accesibilidad serían aquellas que validan código HTML comprobando si las páginas están bien formadas y son válidas. La validez que comprueban es una validez gramatical. Ejemplo de estas herramientas son los validadores de HTML y de CSS de la W3C (algunas de ellas ya comentadas en el Capítulo 2).

- **Validador HTML de W3C⁹⁹:** es un validador *on line* de validación de código que comprueba la conformidad de páginas web respecto a las gramáticas y otros estándares de la W3C.
- **Validador de CSS de W3C¹⁰⁰:** es una herramienta *on line* que permite comprobar hojas de estilo frente a las especificaciones de la W3C.

Junto a las herramientas anteriores existen otras, también *on line*, que permiten evaluar la accesibilidad atendiendo a las recomendaciones WCAG 1.0, WCAG 2.0 y, en su caso, atendiendo a la normativa americana en materia de accesibilidad: Section 508. En este sentido, cabe destacar las siguientes herramientas:

- **TAW¹⁰¹** (Test de Accesibilidad Web): es una herramienta *on line* que permite la evaluación automática de accesibilidad. Con ella es posible evaluar sitios web atendiendo a las recomendaciones WCAG 1.0, WCAG 2.0, y mobileOk.
- **HERA¹⁰²**: es una herramienta *on line* desarrollada por la Fundación Sidar, con la que puede evaluarse la accesibilidad atendiendo a las recomendaciones WCAG 1.0.

ACTIVIDADES 5.9



- Se continúa con la evaluación de la página web de la Fundación CTIC. Para ello utilice la herramienta TAW para analizar sus problemas de accesibilidad.

SOLUCIÓN

Visitaremos el sitio web <http://www.tawdis.net> (la Figura 5.5 muestra una captura de esta página. Posteriormente, solicitaremos la evaluación de la página <http://fundacionctic.org> (la Figura 5.6 muestra una captura del informe obtenido).



Figura 5.5. Página principal de la herramienta TAW

⁹⁹ <http://validator.w3.org>

¹⁰⁰ <http://jigsaw.w3.org/css-validator/>

¹⁰¹ <http://www.tawdis.net>

¹⁰² <http://www.sidar.org/hera/>

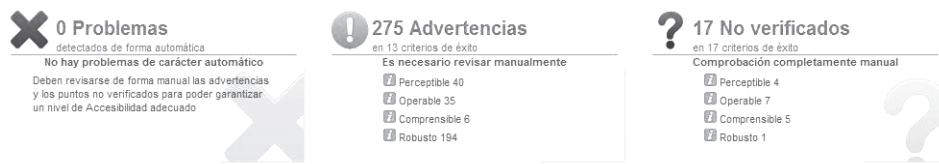


Figura 5.6. Resumen del informe obtenido con la herramienta TAW

Los errores y posibles errores reportados deben ser revisados, la intención será comprobar si realmente se trata de errores o no.

El informe elaborado por la herramienta TAW puede visualizarse en diferentes formatos: resumen, vista marcada, detalle o listado. La vista detalle es una de las más completas formas de revisar los resultados obtenidos, ya que además de mostrar los errores y posibles errores te ofrece enlaces a sugerencias para tratar dichas situaciones.

En el emitido por la herramienta utilizada no se identifica ningún error relacionado con accesibilidad y con los principios asociados a la misma. Si que existen diferentes situaciones que podrían ser errores y de las que se informa en el informe mostrado en la imagen.

5.8.2 CHEQUEO DE LA ACCESIBILIDAD WEB DESDE DIFERENTES NAVEGADORES

Algunos navegadores web facilitan el proceso de evaluación manual de la accesibilidad gracias a la disponibilidad de extensiones o complementos que permiten, por ejemplo, cambiar la resolución con rapidez, hacer comprobaciones de los colores utilizados en un sitio web, etc. Entre esas extensiones destacan las siguientes (algunas de estas han sido comentadas también en el Capítulo 2):

- **Web Developer**: es una extensión para Mozilla Firefox que añade una barra de herramientas con varias funciones de utilidad para los desarrolladores web. Esta barra está enfocada hacia el desarrollador web en general, aunque también incluye funciones útiles para la evaluación de la accesibilidad.
- **Firefox Accessibility Extension**: es una extensión para Mozilla Firefox que añade una barra de herramientas que incluye opciones que facilita la navegación por los contenidos a los usuarios con discapacidad y también permite realizar comprobaciones de accesibilidad.
- **Web Accessibility Toolbar**: es un *plugin* para Internet Explorer que ha sido desarrollado para facilitar la evaluación manual de la accesibilidad de las páginas web.
- **Firebug**: es una extensión permite a los desarrolladores modificar directamente el código fuente HTML, CSS, JavaScript, etc. contenido en la página web. También permite ver el código del documento de forma dinámica, según es generado o modificado por los *scripts*.
- **Fangs**: es una extensión muestra el contenido de la página emulando un lector de pantalla. La página web se convierte en una página solo texto en la que se detalla tanto el contenido del documento como los mensajes propios de los lectores de pantalla (identificando enlaces, imágenes, encabezados, listas, tablas, etc.).
- **HMTL Validator Tidy**: es una extensión para Mozilla que agrega un validador HTML dentro de Firefox. Muestra el número de errores de cualquier página HTML en la barra de estado mientras se navega, además muestra los errores de código al seleccionar la opción **Ver código fuente**.

5.8.3 CHEQUEO DE LA ACCESIBILIDAD WEB DESDE DISPOSITIVOS MÓVILES

Además de las herramientas comentadas en las secciones anteriores, hoy en día hay multitud de dispositivos para acceder a la información, por ejemplo los dispositivos móviles. Si se desea comprobar aquellos sitios web desarrollados específicamente para este tipo de dispositivos se pueden utilizar otras herramientas como son las siguientes:

- **TAW¹⁰³** (Test de Accesibilidad Web): ya ha sido comentada en la sección anterior y que incorpora facilidades para comprobar los diseños elaborados para dispositivos móviles.
- **El comprobador para móviles de W3C¹⁰⁴**: con la que es posible comprobar las buenas prácticas en los diseños realizados para dispositivos móviles siguiendo las recomendaciones web móvil 1.0.
- **mobiReady¹⁰⁵**: permite evaluar un sitio web para móviles utilizando las mejores prácticas y estándares de la industria.

5.9 CONCLUSIONES

En este capítulo se han recogido las definiciones y consideraciones relacionadas con el concepto de accesibilidad en la Web. Junto a su definición se han identificado las pautas de accesibilidad del contenido web, que están organizadas en cuatro principios: facilidad de percepción, facilidad de operación, facilidad de comprensión y robustez.

A partir de esos principios se han identificado las mencionadas pautas y para cada una de ellas hay distintos criterios. Finalmente, se ha proporcionado documentación adicional para comprender y utilizar toda esta información. Principios, pautas y criterios han sido elaborados por la Iniciativa para la Accesibilidad de la Web (WAI) del consorcio W3C.

En la parte final del capítulo se propone un método y se ha dado referencia de distintas herramientas que dan soporte a actividades de evaluación de sitios web considerando aspectos de accesibilidad.

¹⁰³ <http://www.tawdis.net>

¹⁰⁴ <http://validator.w3.org/mobile/>

¹⁰⁵ <http://ready.mobi>



RESUMEN DEL CAPÍTULO



En este capítulo se han recopilado los principios, pautas, criterios y sugerencias que deben tenerse en cuenta para facilitar el diseño de sitios web accesibles. Fundamentalmente, se ha hecho referencia a la versión más reciente disponible de las guías de accesibilidad para el contenido web (WCAG 2.0). Estas guías han sido elaboradas por la WAI, que es un grupo de trabajo dentro del consorcio W3C. En la parte final del capítulo se hace referencia a distintas herramientas que dan soporte a la evaluación semiautomática del cumplimiento de las pautas de accesibilidad para el contenido web.



EJERCICIOS PROPUESTOS



- **1.** Elabore un sitio web con un determinado propósito que cuente con una página principal en el que aparezca un logo, haya un menú de navegación, un cuerpo principal de la página que incluya algunas imágenes, una tabla y algunos enlaces a sitios web de contenido multimedia.
- **2.** Evalúe la página web anterior utilizando la herramienta TAW.
- **3.** Evalúe la página anterior pero ahora utilizando la herramienta JAWS.
- **4.** Identifique las soluciones propuestas en el capítulo para que sean accesibles los siguientes elementos que pueden aparecer en un sitio web:
 - Un logo.
 - Una tabla.
 - Un gráfico estadístico.
 - Un vídeo.
 - Un menú de navegación.
- **5.** Identifique, instale y utilice distintos complementos ofrecidos para lograr facilidades relacionadas con accesibilidad web, desarrollados para el navegador que utilice de manera habitual.



TEST DE CONOCIMIENTOS



1 Si todo contenido de una web tiene que tener una alternativa textual, eso quiere decir que un sitio web donde se ha utilizado únicamente texto es lo más adecuado para garantizar un alto nivel de accesibilidad:

- a)** Sí, una alternativa textual sería siempre completamente accesible.
- b)** No, la Web es en muchas ocasiones diseño, y el diseño no tiene que estar reñido con la accesibilidad.
- c)** Sí, cuando se trata de sitios web institucionales.

2 Atendiendo a su nivel de abstracción, tomando el criterio de mayor a menor, ¿cuál sería el orden correcto de las recomendaciones dadas en este capítulo para lograr accesibilidad?

- a)** Principios, pautas, criterios y recomendaciones.
- b)** Pautas, recomendaciones y criterios.
- c)** Pautas y criterios.

3 En el ámbito de la accesibilidad web, el nivel de adecuación es:

- a)** Es el nivel de exigencia que se establece para el cumplimiento de las pautas de accesibilidad.
- b)** Es el nivel de cumplimiento de los principios de accesibilidad para la Web.
- c)** Es el nivel alcanzado por el sitio web para sus visitantes.

4 La evaluación de la accesibilidad es siempre:

- a)** Objetiva.
- b)** Subjetiva.
- c)** Automática.

5 El nivel de adecuación de sitios web institucionales debería ser:

- a)** A.
- b)** AA.
- c)** AAA.

6 La nivel de accesibilidad alcanzado por un sitio web lo determina únicamente:

- a)** El código HTML.
- b)** El usuario que utilice el sitio.
- c)** Las herramientas utilizadas para su evaluación.

6

Implementación de la usabilidad en la Web. Diseño amigable

OBJETIVOS DEL CAPÍTULO

- ✓ Analizar la usabilidad de diferentes documentos web.
- ✓ Valorar la importancia del uso de estándares en la creación de documentos web.
- ✓ Modificar la interfaz web para adecuarla al objetivo que persigue y a los usuarios a los que va dirigido.
- ✓ Verificar la facilidad de navegación de un documento web mediante distintos periféricos.
- ✓ Analizar diferentes técnicas para verificar la usabilidad de un documento web.
- ✓ Verificar la usabilidad de la interfaz web creada en diferentes navegadores y tecnologías.

En el capítulo anterior se ha considerado un criterio de calidad, como fue el de la accesibilidad, a la hora de desarrollar sitios web. En este capítulo se completarán las exigencias de calidad de un sitio web considerando factores de calidad adicionales.

La visión que se ofrece en este capítulo introduce el concepto de interfaz web amigable, usable y que presenta un alto grado de usabilidad. A lo largo de este capítulo se definen todas esas características y se recopilan recomendaciones y principios de diseño para lograrlas en cualquier diseño de un sitio web. En la parte final aparecerán referencias a herramientas que dan soporte a la evaluación y validación del grado de calidad alcanzado.

Se pueden avanzar, antes de comenzar este capítulo, que serán tres las principales actividades generales que cualquier diseñador deberá considerar a la hora de crear interfaces web amigables. En primer lugar, el diseñador debe conocer al usuario potencial de su sitio web y las tareas que éste desea abordar utilizando ese sitio. En segundo lugar, el diseñador tendrá que realizar diferentes prototipos y barajar distintas alternativas de diseño del sitio que desea ofrecer. Dichos prototipos deberán evaluarse, atendiendo a distintos criterios que se precisarán precisando a lo largo del capítulo. Por último, el diseñador deberá mejorar los prototipos que se vaya elaborando, trabajando de una manera iterativa, es decir, por etapas y apostando por una u otra alternativa barajada en los prototipos. Estas tres actividades mencionadas son las que habitualmente se conocen en la literatura como Diseño Centrado en el Usuario.

6.1 CONCEPTO DE USABILIDAD

Inicialmente, cualquier sitio web (extendible a cualquier producto software) que vaya a ser utilizado por usuarios deberá tratar de cubrir las necesidades y exigencias de esos usuarios. Ese equilibrio, entre exigencias de los usuarios y las características de las que hace gala un sitio web, es lo que se conoce como calidad de un producto software.

Desde el punto de vista del usuario, cuando se quiere hacer referencia a que un sitio web es de calidad, se dice que dicho producto es amigable o que hace gala de un alto grado de usabilidad.

La usabilidad es un término que aunque no forma parte del diccionario de la Real Academia Española (RAE), es bastante habitual en el ámbito de la informática y la tecnología. El concepto proviene de la palabra inglesa *usability*, y hace referencia a la facilidad con que un usuario puede utilizar una herramienta fabricada por otras personas con el fin de alcanzar ciertos objetivos.

La usabilidad está vinculada, por tanto, a la simpleza, la facilidad, la comodidad y la practicidad. En otras palabras, el concepto tiene relación con la eficacia percibida de un objeto y la posibilidad de aprovechar todo su potencial. La usabilidad sostiene que si el sistema en cuestión no puede ser utilizado por el usuario en todo su potencial, no es útil y, por tanto, es deficiente. El grado de usabilidad se mide a partir de diferentes técnicas de análisis de la usabilidad, que serán introducidas en la siguiente sección.

Además de la definición recogida en el DRAE, una entidad internacional, la Organización Internacional para la Estandarización (ISO), ha propuesto dos definiciones para el concepto de usabilidad que merecen ser tenidas en cuenta. Las definiciones son las recogidas en distintas normas ISO: la 9241-11:1998 y la 9126-1:2001

La norma ISO/IEC 9126-1:2001, define la usabilidad como: “la capacidad de un software para ser **comprendido, aprendido, usado** y ser **atractivo** por el usuario, en condiciones específicas de uso”.

En esta definición se hace énfasis en los atributos internos y externos del producto (sitio web en nuestro caso), los cuales contribuyen a su usabilidad, funcionalidad y eficiencia. La usabilidad depende no solo del producto sino también del usuario. Por ello un producto no es en ningún caso intrínsecamente usable, solo tendrá la capacidad de ser usado en un contexto particular y por usuarios concretos. La usabilidad no puede ser valorada estudiando un producto software de manera aislada.

Paralelamente, en la norma ISO 9241:1998, se define la usabilidad de la siguiente forma: “La usabilidad es la **efectividad, eficiencia y satisfacción** con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso concreto”.

La definición anterior está centrada en el concepto de calidad en tanto el cuanto el producto es usado, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios particulares con efectividad.



Figura 6.1. Ejemplos de poca usabilidad en la vida real

En la Figura 6.1 se muestran distintos contraejemplos a lo que se ha venido definiendo como usabilidad. Se puede ver una caja de cerillas excesivamente pequeñas, una regadera que no permite un uso adecuado o un dispositivo de telefonía móvil cuya forma de interacción no parece la más adecuada para su manejo eficiente y eficaz. En el ámbito del desarrollo para la Web la usabilidad aportaría esas mismas características, identificadas en los estándares, y que se deben tener presentes a la hora de diseñar sitios web.

ACTIVIDADES 6.1



- Discuta sobre la relación, similitudes y diferencias que existen entre las distintas definiciones de usabilidad recogidas en esta sección.

SOLUCIÓN

Las dos definiciones de usabilidad procedentes de los estándares mencionados (ISO/IEC 9126-1 e ISO 9241-11) parecen distintas a primera vista, pero no lo son tanto. Fundamentalmente, presentan el concepto de usabilidad desde dos puntos de vista: producto y proceso, respectivamente.

Con la norma ISO/IEC 9126-1 podemos evaluar el producto software e incluso sin necesidad de consultar a usuarios finales, determinar si el producto software será fácil de entender, fácil de recordar o fácil de utilizar. Dicha evaluación podría hacerse por expertos o diseñadores utilizando su experiencia.

Sin embargo, la norma ISO 9241-11 define la usabilidad atendiendo al proceso, es decir, solo será posible saber si algo es efectivo, eficiente o satisfactorio si, utilizándose por usuario, logran sus objetivos, los logran en tiempo y forma y, además, lo hacen de manera satisfactoria.

Lo que resulta indudable es que existe una influencia entre las características internas de un producto software y las prestaciones que éste puede proveer a sus usuarios.

6.2 ANÁLISIS DE LA USABILIDAD. TÉCNICAS

Un aspecto importante para lograr incrementar el grado de usabilidad de un sitio web pasa por identificar qué necesitan los usuarios potenciales de dichos sitios web, y saber cuándo el diseñador las ha tenido en cuenta. Para ello se han propuesto e identificado diferentes técnicas de análisis de la usabilidad. Dichas técnicas pueden agruparse en distintos grupos. Estos grupos serían los siguientes:

- Las *técnicas de sondeo o indagación*. En este grupo se incluirían todas aquellas técnicas basadas en la realización de entrevistas, cuestionarios, y todas aquellas variantes en las que se pregunta a usuarios reales qué buscan o necesitan de un sitio web. En este tipo de técnicas se necesitan usuarios potenciales o reales del sitio web, ya que es a quienes se preguntará qué necesitan y si el sitio que han venido utilizando, o que se les ha presentado, ofrece alguna limitación o cubre todas sus necesidades.
- Las *técnicas de inspección*. En este otro grupo de técnicas se incluyen aquellas otras actividades donde no son necesarios los usuarios finales, pero sí el acceso a expertos, guías de estilo, heurísticas, o experiencia documentada con la que poder revisar y analizar el sitio web que se ha desarrollado o que se está desarrollando. Los ejemplos más representativos de este tipo de técnicas son las *evaluaciones heurísticas* y los *recorridos cognitivos*.
- Las *técnicas de prueba con usuarios*. Se incluirían en este último grupo de técnicas un conjunto de actividades para probar las ventajas o limitaciones de un sitio web en las que se requiere la participación de usuarios. Es decir, en esas actividades de prueba se precisa de usuarios reales para confirmar si el producto se adapta a sus necesidades o no, tomando nota de cómo se comportan los usuarios cuando utilizan ese sitio web. Para poner en práctica esta técnica de prueba se puede o no disponer del sitio web final, ya que se pueden utilizar prototipos más o menos elaborados. La intención, con estos últimos, es obtener *feedback*¹⁰⁶ del usuario lo antes posible y poder ofrecer un producto final que ofrezca garantías de que el producto será amigable.

Fruto de poner en práctica las técnicas anteriores se pueden identificar objetivos, tipos de usuarios y barreras que los usuarios identifican en los desarrollos. Sin embargo, para elaborar prototipos dirigidos a mejorar la usabilidad de un sitio web, es interesante seguir las recomendaciones que expertos en usabilidad siguen, ya que es la perspectiva más práctica que se puede tener en cuenta. En la siguiente sección se detallarán estas recomendaciones.

¹⁰⁶ Término anglosajón que hace referencia a una realimentación de ideas. A un usuario se le enseña un sitio web para que él lo vea y diga lo que se le ocurre para mejorar o para resaltar que están bien. Las apreciaciones del usuario es el *feedback* que obtiene el diseñador.

ACTIVIDADES 6.2



► Relacione las definiciones de usabilidad y la caracterización propuesta en los estándares internacionales con las técnicas de análisis de la usabilidad comentadas en este apartado.

SOLUCIÓN

En el apartado 6.1 dedicada a las definiciones de usabilidad hemos identificado dos orientaciones o puntos de vista: la definición orientada al producto y las orientadas al proceso.

En las técnicas clasificadas se identifican actividades de evaluación y de valoración de productos software en las que se requiere de usuarios potenciales o reales del producto software bajo desarrollo y otras técnicas en las que no es necesario recurrir a esos usuarios.

Por lo tanto, se propone la siguiente correspondencia entre las definiciones y las técnicas de análisis:

- a. Las técnicas de sondeo e indagación, pueden ser de utilidad tanto para evaluar el la usabilidad de un sitio web desde un punto de vista de producto como de proceso.
- b. Las técnicas de inspección no requieren de usuarios y pueden ligarse a un punto de vista de producto del concepto de usabilidad.
- c. Por último, la definición de proceso del concepto de usabilidad está más relacionada con técnicas de evaluación basadas en pruebas con usuarios.

6.3 PRINCIPIOS PARA CONSEGUIR WEBS AMIGABLES

Para lograr que los sitios web sean amigables y presenten, por tanto, un alto grado de usabilidad se necesita disponer de suficiente experiencia o documentación en principios de diseño para la Web, es decir, conocer buenas prácticas o soluciones habituales a problemas que surgen una y otra vez en el desarrollo de productos para la Web. De alguna manera, el diseñador de sitios web debe conocer los problemas que ya se han encontrado otros diseñadores y sus soluciones.

En este sentido, hay que resaltar que seguir recomendaciones o buenas prácticas de diseño no supone, en ningún caso, limitar las posibilidades creativas del autor o diseñador de sitios web. Por contra, tener presentes estas soluciones documentadas o buenas prácticas facilita que los usuarios de los sitios web diseñados entiendan, aprendan y utilicen los productos diseñados de una forma más eficiente, eficaz y satisfactoria.

Entre los expertos (llamados gurús en el argot) en el ámbito de la usabilidad, cuyos principios se mencionarán seguidamente, están Jakob Nielsen y Bruce Tognazzini (Figura 6.2). Ambos son consultores de usabilidad y se han hecho un hueco a nivel mundial en el terreno de la usabilidad y en el diseño de productos interactivos para la Web. Así, Nielsen ha publicado diferentes libros y propuestas relacionadas con el concepto de usabilidad y es uno de los pioneros en el uso de la denominación “Ingeniería de la usabilidad”.



Figura 6.2. Jakob Nielsen y Bruce Tognazzini

Los diez principios de diseño propuestos por Jakob Nielsen, para el diseño de productos software con alto grado de usabilidad, son los siguientes:

- **Visibilidad del estado del sistema:** el producto software o sitio web debe siempre mantener informado a los usuarios de lo que ocurre, con un correcto feedback en un tiempo razonable.
- **Correspondencia entre los contenidos del sitio web y el mundo real:** el sitio web debe hablar el lenguaje de los usuarios con palabras, frases y conceptos familiares. Es decir, el contenido debe seguir las convenciones del mundo real y el diseñador de sitios web debe ser capaz de mostrar la información de forma natural y lógica.
- **Control y libertad del usuario:** los usuarios frecuentemente eligen opciones por error, por eso siempre debe ofrecerse a los usuarios un punto de salida a un lugar seguro, por ejemplo deshacer las acciones realizadas o volver a la página principal de un sitio web. Para ello el diseñador debe indicar una salida clara a esas situaciones no deseadas sin necesidad de pasar por diálogos extensos o poco claros.
- **Consistencia y estándares:** los usuarios no deberían tener que adivinar que las diferentes palabras, situaciones o acciones significan lo mismo. Los colores, los tipos de fuentes, la distribución de los contenidos a lo largo de las distintas secciones y páginas de un sitio web deben ser homogéneas a lo largo de todo el sitio web.
- **Evitar errores:** el diseñador debe tener en cuenta que un diseño cuidado que previene de problemas al usuario es mejor que unos buenos mensajes de error.
- **Reconocimiento:** el diseñador de un sitio web debe ofrecer objetos, acciones y opciones claramente visibles e identificables. El usuario no debería tener que recordar información de unas zonas de un sitio web a otras. Si para hacer determinadas tareas es necesario saber unas instrucciones de uso, entonces éstas deben estar visibles o ser fácilmente recuperables.
- **Flexible y eficiente:** el diseño de un sitio web debe permitir ser utilizado por un rango amplio de usuarios. Además debe cuidarse dicho diseño en términos de no imponer retrasos a usuarios avanzados ni impedir que usuarios novatos no puedan interactuar con relativa facilidad.
- **Diseño minimalista:** cualquier contenido que aparezca en un sitio web debería estar justificado, ya sean imágenes, vídeos, texto, multimedia, enlaces, etc.
- **Reconocer, diagnosticar y recuperarse de los errores:** con el fin de ayudar a los usuarios, los mensajes de error deben estar escritos en lenguaje sencillo, indicar el problema de forma precisa e indicar también una solución.

- **Ayuda y documentación:** el sitio web, si fuera necesario para realizar determinadas tareas, facilitará documentación o asistencia. La información debe ser fácil de encontrar, está dirigida a las tareas de los usuarios, lista los pasos concretos para hacer algo y es breve.

Los principios de diseño anteriores no son los únicos que pueden proponerse. Otros autores han propuesto otras colecciones de principios de diseño amigable. Bruce Tognazzini propone una colección más amplia, y en algunos casos algo más precisa, de principios. Tognazzini, es también consultor de usabilidad y junto con Jakob Nielsen y Donald Norman crearon la consultora Norman Nielsen Group. Además, mantiene un sitio web dedicado a la difusión y diseminación del concepto de usabilidad (*askTog.com*). Los principios que Tognazzini propone son los siguientes:

- ✓ Anticipación, el sitio web debe anticiparse a las necesidades del usuario.
- ✓ Autonomía, los usuarios deben tener el control sobre el sitio web. Los usuarios sienten que controlan un sitio web si conocen su situación en un entorno abordable y no infinito.
- ✓ Los colores han de utilizarse con precaución para no dificultar el acceso a los usuarios con problemas de distinción de colores.
- ✓ Consistencia, las aplicaciones deben ser consistentes con las expectativas de los usuarios, es decir, con su aprendizaje previo.
- ✓ Eficiencia del usuario, los sitios web se deben centrar en la productividad del usuario, no en la del propio sitio web. Por ejemplo, en ocasiones tareas con mayor número de pasos son más rápidas de realizar para una persona que otras tareas con menos pasos, pero más complejas.
- ✓ Reversibilidad, un sitio web ha de permitir deshacer las acciones realizadas.
- ✓ Ley de Fitts. Esta ley indica que el tiempo para alcanzar un objetivo con el ratón está en función de la distancia y el tamaño del objetivo. A menor distancia y mayor tamaño más facilidad para usar un mecanismo de interacción.
- ✓ Reducción del tiempo de latencia. Siempre se debe tratar de optimizar el tiempo de espera del usuario, permitiendo la realización de otras tareas mientras se completa la previa e informando al usuario del tiempo pendiente para la finalización de la tarea.
- ✓ Aprendizaje, los sitios web deben requerir un mínimo proceso de aprendizaje y deben poder ser utilizados desde el primer momento.
- ✓ El uso adecuado de metáforas facilita el aprendizaje de un sitio web, pero un uso inadecuado de estas puede dificultar enormemente el aprendizaje.
- ✓ La protección del trabajo de los usuarios es algo prioritario, se debe asegurar que los usuarios nunca pierden su trabajo como consecuencia de un error.
- ✓ Legibilidad, el color de los textos debe contrastar con el del fondo, y el tamaño de fuente debe ser suficientemente grande.
- ✓ Seguimiento de las acciones del usuario. Conociendo y almacenando información sobre su comportamiento previo se ha de permitir al usuario realizar operaciones frecuentes de manera más rápida.

- ✓ Interfaz visible. Se deben evitar elementos invisibles de navegación que han de ser inferidos por los usuarios, menús desplegables, indicaciones ocultas, etc.
- ✓ Teniendo presentes los principios de Nielsen y Tognazzini se dispondrá de criterio para realizar diferentes diseños para un sitio web. Aunque es necesario insistir en que los principios presentados en esta sección no suponen limitación a la capacidad creativa de un diseñador.

ACTIVIDADES 6.3



➤ Identifique, en sitios web de reconocido prestigio y distinta temática, cómo se han llevado a la práctica y se han tenido en cuenta los principios propuestas por Nielsen y Tognazzini.

Los sitios web que se estudiarán son:

- a. Comercial: <http://www.amazon.es>
- b. Institucional: <http://www.defensordepueblo.es>
- c. Buscador: <http://www.google.es>

SOLUCIÓN

Consultando las referencias anteriores ejercitaremos también nuestra capacidad de observación. Podremos observar cómo hay una serie de elementos que aparecen una y otra vez en los diseños de sitios web, por ejemplo: un logo, una dirección significativa, un título significativo, unos elementos de navegación, un pie de página en el que se ofrecerá información complementaria.

Igualmente habrá localizaciones en una página web que tendrán propósitos determinados. Por ejemplo, el logo y el título de la página web siempre aparecen en la parte superior, formularios de búsqueda aparecen en la parte superior de la página, en el centro o a la derecha de la misma, los elementos de navegación principales aparecen en el lado izquierdo (en vertical) o en la parte superior (en horizontal), los enlaces de navegación internos suelen aparecer en la parte izquierda de la página, mientras que los enlaces externos o la publicidad suele incluirse en la parte derecha de la misma. Además, hay secciones del tipo "acerca de" o "contacta con nosotros", que son muy habituales.

6.4 IDENTIFICACIÓN DEL OBJETIVO DE LA WEB

Los sitios web que se pueden encontrar en la Web pueden ser de muy diversos tipos y temáticas. Estos tipos de sitios web pueden atender a diferentes criterios, pero uno de los principales elementos de organización de los sitios web es su objetivo.

Inicialmente, uno de los objetivos más destacados y demandados para la Web fue el meramente informativo. La Web informativa tiene como principal pretensión la de difundir y distribuir información. En este tipo de sitios web la accesibilidad es una característica y una exigencia fundamental.

Progresivamente, y fruto de las posibilidades para dar soporte a actividades comerciales en la Web, los sitios web con fines comerciales se han extendido rápidamente. Muchos sitios web, en la actualidad, surgen para promocionar y difundir instituciones y empresas con ánimo de lucro. Es decir, su finalidad es fundamentalmente económica y no solo informativa. Su audiencia puede estar formada por clientes, inversores, empleados (ya sean cualquiera de estos colectivos reales o potenciales) e incluso la competencia y los medios de comunicación. Los sitios web comerciales pueden organizarse en webs corporativas, es decir, sitios web que informan sobre la empresa y promocionales, destinadas a publicitar productos.

Otra actividad a la que da soporte la Web es el entretenimiento y el ocio. En este tipo de sitios web, en el fondo, suele haber una finalidad también económica. Este tipo de sitios web no son sitios fáciles de crear ni de mantener y a veces siguen reglas propias de diseño; puesto que en ellos es más importante sorprender al usuario con innovaciones, que dar soporte a una actividad productiva. Por ello, algunos de los principios que defendían los expertos vistos en la sección anterior (Nielsen o Tognazzini) para el diseño de sitios web no siempre son considerados, por ejemplo, los diseños minimalistas.

Un grupo de sitios web de bastante utilidad son aquellos sitios que nos permiten buscar información en la Web. Su propósito es facilitar la localización de la información y permiten que el usuario sepa dónde buscar lo que necesita o desea o simplemente saber si sobre dicha información existe constancia en la Web.

Otros sitios web que tienen cabida en la Web son los de propósito artístico. Estos sitios son un medio de expresión artística de su creador o creadores. Este tipo de sitios suele saltarse todas las convenciones y las únicas normas a aplicar son las que el propio artista o artistas deseen.

En la web también hay sitios web personales Al igual que los anteriores, son un medio de expresión de su creador o creadores. Sus objetivos y su audiencia pueden ser de lo más variopinto. Dentro de este grupo se pueden incluir todo tipo de webs, desde colecciones de fotos de la familia hasta publicaciones o trabajos de investigación de primer orden.

Fruto de la evolución de la Web, concretamente con la Web 2.0, muchos sitios web que inicialmente eran personales y donde el único responsable de los contenidos era su propietario, se han sustituido por sitios web donde la Web se convierte en plataforma, y donde la propiedad y la responsabilidad de los contenidos se distribuye entre muchos usuarios. Se estaría hablando en ese caso de sitios web colaborativos.

En definitiva, se ha visto en secciones anteriores que existen distintos principios de diseño que se deben considerar a la hora de desarrollar un sitio web, pero que en algunos casos el diseñador contará con libertad total para llevar a cabo esta labor. En cualquier caso, en la mayoría de las ocasiones los principios sugeridos son tan generales que pueden considerarse independientemente del objetivo del sitio web.

ACTIVIDADES 6.4



- Busque y localice sitios y páginas en la Web que se correspondan con los distintos tipos de sitios web identificados en esta sección. Aproveche para comparar la presencia o ausencia de los principios de diseño de Nielsen y Tognazzini en los diseños ofrecidos por dichas páginas.

6.5 TIPOS DE USUARIO. NECESIDADES

Independientemente de las características, objetivos y del potencial de un sitio web, será el usuario final el que mejor determinará el grado de usabilidad, y de calidad en general, que dicho sitio le ofrece.

En general, lo que el usuario busca en un sitio web es cubrir sus necesidades, realizando un conjunto de tareas, y nadie mejor que él mismo sabe qué necesita hacer y si el sitio web se lo permite. Estas necesidades pueden traducirse, atendiendo al concepto de usabilidad, en que el usuario es capaz de realizar tareas útiles para él en el menor tiempo posible, utilizando un número de recursos adecuados y de manera satisfactoria. El problema con los sitios web, respecto a otros productos software tradicionales, es que los usuarios potenciales o visitantes del sitio web pueden ser muy diversos. Cualquier persona puede visitar un sitio web ofrecido de manera pública, por ello las características de usabilidad y accesibilidad logradas por el sitio web son especialmente importantes.

Atendiendo al tipo de usuario y a las necesidades que éste puede precisar, se pueden identificar diferentes tipos de sitios web:

- Los sitios web públicos para usuarios en general, que serían aquellos sitios web dirigidos al público en general, es decir a un tipo de usuario sin restricciones de acceso a los contenidos y a las facilidades ofrecidas.
- Los sitios web públicos, pero donde el usuario tiene que registrarse para acceder de manera plena a los contenidos y facilidades allí ofrecidas por el sitio web. En ellos, el usuario debe identificarse para acceder al sitio web y en función de ello los contenidos que se le muestran pueden variar. Por ejemplo, sitios web de este tipo pueden ser aquellos donde esté soportada la actividad de proveedores de una empresa determinada, la actividad comercial de los clientes, o de sus trabajadores. Obviamente, cada uno de estos colectivos tendrá unos objetivos cuando accede al sitio web de dicha institución.
- El tercer tipo de sitios web y de usuarios serán los sitios web privados. En este otro grupo de sitios web el acceso está restringido a los usuarios de una empresa, organización o institución, normalmente funcionan dentro de redes privadas, aunque no siempre es así. El registro en estas web por parte de nuevos usuarios estará gestionado por la institución que mantiene el sitio web.

ACTIVIDADES 6.5



- Busque, localice y estudie distintos sitios web en los que se dé soporte a los distintos tipos de usuarios y formas de acceso identificados en la presente sección.

6.6 BARRERAS IDENTIFICADAS POR LOS USUARIOS E INFORMACIÓN FÁCILMENTE ACCESIBLE

Desde el capítulo primero de este libro se vienen identificando distintos elementos característicos de un sitio web. Cada uno de estos elementos pueden incorporar barreras a los usuarios que se acercan a cualquier sitio web. En este capítulo se han recogido distintos principios de diseño que tienen que ver con esos elementos de cualquier web.

Recordando, los elementos que estarán presentes en todos los sitios web son: el contenido, la estructura y la navegación, el diseño visual, la funcionalidad y la interactividad ofrecida por el sitio web. Todos esos elementos deben ser diseñados de manera consistente y cuidada, porque todos ellos, en función de cómo hayan sido diseñados, pueden introducir barreras a las actividades que el usuario desee realizar. En ocasiones estas barreras las introduce el diseño ofrecido del sitio web, pero en otras las barreras pueden venir dadas por las capacidades y características del usuario. La Figura 6.3 ilustra el comportamiento de los usuarios ante posibles barreras.



Figura 6.3. La usabilidad supone eliminar barreras a la actividad del usuario

Respecto al contenido que aporta el sitio web que se esté diseñando debe tenerse en cuenta que no solo se puede considerar el texto, sino también las animaciones, los sonidos o los vídeos incluidos. Todos ellos deben ser claros, concisos y apropiados considerando los usuarios a los que vaya dirigido.

Un aspecto clave es el relacionado con la estructura y la navegación a través del sitio web. En este sentido la organización de los contenidos, la presentación de la información atendiendo a su importancia y las facilidades para moverse a través del sitio web son esenciales. Las características principales que se deben tratar de lograr como diseñador de un sitio web es la de que la estructura y la navegación sean consistentes, intuitivas y transparentes. Esto quiere decir, que a lo largo del sitio web interacciones o actividades de navegación similares proporcionarán resultados idénticos, además serán fáciles de identificar y de llevarse a cabo. La disponibilidad de estas características de navegación propiciará que el usuario se mueva rápidamente por el sitio web.

Como se ha comentado anteriormente, hay sitios web en los que los aspectos de diseño visual adquieren una importancia relevante, es el caso de los sitios web de entretenimiento, ocio o artísticos. Pero las soluciones de diseño utilizadas en ellos no deberían introducir impedimentos a la hora de acceder a los contenidos o a la funcionalidad que en ellos se ofrece. Una página que debe cuidarse especialmente es la página principal de un sitio web, su diseño será la tarjeta de presentación de dicho sitio web.

En cualquier caso, cualquier visitante que se acerca a un sitio web, lo hace porque desea cubrir unos objetivos y los aspectos de funcionalidad son esenciales. Una buena funcionalidad significa que el sitio web trabaja bien; se carga rápidamente, los enlaces funcionan y que la información y facilidades que aporta el sitio web son relevantes para su audiencia y no se ven afectados por el navegador utilizado o por el dispositivo, la conexión o las propias características del usuario visitante.

Relacionado con la funcionalidad, la interactividad con el sitio es también importante. Progresivamente, en la Web, el usuario recibe información, pero también aporta datos, acciones y recursos. En muchos casos la finalidad de los sitios web es económica y los aspectos de interactividad son esenciales para dar soporte a dicha finalidad. El tratamiento de la información facilitada por los usuarios, en algunas ocasiones es delicado, por tratarse de información personal y sensible, y el tratamiento de los errores que se produzcan interactuando con el sitio web también debe tenerse en cuenta a la hora de desarrollar un sitio web.

ACTIVIDADES 6.6



- Revise el siguiente sitio web: <http://www.webpagesthatsuck.com/the-worst-web-page-in-the-world/>. En él se recopilan enlaces a sitios web diseñados de manera pobre. Analice y estudie, utilizando los principios propuestos por Nielsen y Tognazzini, por qué dichos sitios web están mal diseñados y qué barreras introducen a sus visitantes.

Para la realización de esta tarea se debe fijar la atención en:

- a. Aspectos de legibilidad.
- b. Agrupamiento significativo.
- c. Uso del color.
- d. *Feedback* (ayuda del sistema al usuario, información sobre dónde está).
- e. Etiquetado de enlaces.
- f. Títulos asignados a las páginas, concisión.
- g. Número de acciones para realizar las tareas.
- h. Densidad de la información.
- i. Control por parte del usuario.
- j. Mensajes de error.
- k. Protección ante errores.
- l. Consistencia y compatibilidad.

6.7 VELOCIDAD DE CONEXIÓN

Internet forma parte del día a día de mucha gente. Es fuente de investigación para algunos, ocio y trabajo para otros. La gran red de computadoras se expande a gran velocidad. Desde el año de su creación, en la década de los 60, hasta hoy, fueron desarrolladas diversas maneras de conectarse a la red. Por eso, es importante saber cuáles son las posibilidades de conexión, sus ventajas, desventajas y cómo funcionan. Hoy en día coexisten diferentes tipos de conexión que ofrecen diferentes características. Seguidamente se recogen estos tipos de conexión y las velocidades que ofrecen.

- **Conexión por línea telefónica.** También llamada dial-up, es el método de conexión más antiguo y era el único utilizado cuando Internet daba sus primeros pasos. El acceso es realizado por el usuario mediante un módem y una línea telefónica convencional. Este tipo de conexión es cada vez menos usada, ya que la capacidad de transmisión de datos no supera los 56 kbps, lo que hace que la navegación sea muy lenta.
- **Conexión xDSL.** La conexión xDSL es suministrada por medio de la red telefónica convencional, pero difiere al acceso dial-up. El servicio xDSL funciona mediante la contratación de un proveedor de acceso, al igual que el dial-up, y es posible acceder a servicios con diversas velocidades. Por ejemplo, en el ADSL, la velocidad varía de 256 kbps a 8 mbps; en el ADSL2 o ADSL2+ va desde 256 kbps hasta 24 Mbps; en el VDSL puede llegar a una velocidad de 52 Mbps y en el VDSL2 hasta 100 Mbps. A pesar de la popularidad de ese tipo de acceso, no está disponible en todos lados. El xDSL tiene como desventaja que al tratarse de un servicio compartido, la navegación puede ser más lenta en horarios pico, cuando muchos usuarios utilizan el servicio simultáneamente.
- **Conexión por televisión por cable.** La conexión por cable es cada vez más popular y utiliza la misma infraestructura que la del servicio de cable contratado, lo que facilita la instalación. Muchos servicios de televisión por cable ofrecen en el paquete el acceso a Internet con distintas velocidades. Este tipo de acceso pone a disposición distintos tipos de velocidades. La velocidad de navegación, y el límite de las descargas y de los datos subidos, dependen del paquete que se contrate. Además, la velocidad no se ve afectada por la cantidad de usuarios u horarios en que se use el servicio. A diferencia del acceso xDSL, el usuario siempre tendrá la misma velocidad de acceso, en cualquier horario.
- **Conexión por satélite.** Este tipo de conexión requiere equipos específicos que suelen tener un costo muy elevado. En algunos casos, la antena es suministrada por el propio proveedor del servicio. Este tipo de acceso a Internet, cuenta con planes que ofrecen velocidades que varían desde los 512 kbs hasta los 2 Mbps. Una de las ventajas de la conexión por satélite es que el acceso no depende de la localización. De esta manera se tendrá acceso a Internet en cualquier lugar donde llegue la cobertura. Sin embargo, mientras más remoto sea el lugar donde nos encontremos, más potente será la señal.
- **Conexión por radio.** El acceso a Internet por radio es una manera de extender una conexión de banda ancha a algún lugar donde no se dispone del servicio. Ese punto puede ser desde una pequeña área restringida, como una oficina, hasta una ciudad completa. Para eso es necesario configurar una red sin cables. Están incluidos en esta modalidad el Wi-Fi y el Wi-Max. Una de las ventajas de la conexión por radio es la posibilidad de repartir el acceso y la garantía de la movilidad a los usuarios.

■ **Conexión 3G.** La conexión a Internet a través de los teléfonos celulares es cada vez mejor. La llegada de la tecnología 3G proporcionó banda ancha a los teléfonos celulares, y otorgó una velocidad de navegación con una considerable aceleración. La movilidad es una gran ventaja de los servicios de este tipo. En el caso de las redes GSM, la velocidad de transferencia puede alcanzar los 800 kbps. En el caso de las redes CDMA, la transferencia puede llegar a alcanzar una velocidad de hasta 2 Mbps.

La velocidad de conexión y por tanto de carga de un sitio web afecta a su usabilidad y condiciona el comportamiento de sus usuarios. Esta afirmación puede resultar obvia pero la cuestión que plantea no lo es: ¿De qué manera, en cifras, afecta la velocidad de carga de un sitio al comportamiento de sus usuarios que lo visitan?

Los experimentos realizados por *Bing* y *Google* han tratado de dar respuesta a dicha cuestión introduciendo unos tiempos adicionales de espera en determinados sitios web para ver cómo reaccionaban los usuarios que lo visitaban. Los resultados fueron muy indicativos y significativos. Los datos revelan que una demora de 1000 ms puede causar una caída del interés general del usuario por un sitio de más del 2%. Estos datos, extrapolados en términos de rendimiento, suponen una pérdida muy importante para las grandes empresas que ven como un significativo porcentaje de sus ganancias se escapa por una mera cuestión técnica que provoca la impaciencia de los usuarios.

ACTIVIDADES 6.7



- Evalúe los tiempos de descarga del sitio web correspondiente a la Universidad de Castilla-La Mancha (<http://www.uclm.es>) utilizando los servicios facilitados en el sitio web cuya dirección se adjunta: <http://www.websiteoptimization.com/services/analyze/>.
- El uso de este servicio es muy simple, solo se necesita facilitar la referencia del sitio web que se desea analizar y solicitar que se lleven a cabo los cálculos.

SOLUCIÓN

Tiempos de descarga de la página web correspondiente a la Universidad de Castilla-La Mancha:

Tasa de conexión	Tiempo de descarga
14,4 K	619,26 segundos
28.8 K	313,33 segundos
33.6 K	269,63 segundos
56 K	164,74 segundos
ISDN 128 K	55,58 segundos
T1 1,44 Mbps	11,58 segundos

6.8 IMPORTANCIA DEL USO DE ESTÁNDARES EXTERNOS

Las pautas de usabilidad, las buenas prácticas y los estándares de facto, es decir, aquellos estándares y normas propuestas desde el ámbito de instituciones y organizaciones, han demostrado ser un mecanismo muy práctico para desarrollar sitios web usables, accesibles y útiles para los usuarios.

En esos compendios de buenas prácticas los diseñadores pueden encontrar soluciones más concretas y más fáciles de poner en práctica que los principios de Jakob Nielsen y Bruce Tognazzini. En este sentido, en la Tabla 6.1, se identifican algunos ejemplos de guías o manuales para el desarrollo web elaborados por los gobiernos de Chile, Uruguay, Colombia, Estados Unidos, Reino Unido o Costa Rica. Estas guías son ejemplos de buenas prácticas en el diseño.

Tabla 6.1 Lista de algunas directrices gubernamentales orientadas al desarrollo de sitios estatales

País	Guía o Manual
Chile	Guía para desarrollo de sitios web. Guía web v2.0. http://www.guiaweb.gob.cl
Colombia	Directrices de usabilidad para sitios web del Estado colombiano. http://www.gobiernoenlinea.gov.co/web/guest/forums/-/message_boards/message/578799
Estados Unidos	Research-Based Web Design and Usability Guidelines. http://usability.gov/guidelines/index.html
México	Guía de Desarrollo de Sitios Web de la Administración Pública Federal. http://www.sip.gob.mx/indexbded.html?option=com_content&task=view&id=37&Itemid=118
Reino Unido	Guidelines for UK Gobernment Websites. http://coi.gov.uk/webguidelines
Uruguay	Guía para el diseño e implementación de Portales Estatales. http://www.agesic.gub.uy/innovaportal/v/561/1/agesic/Gu%C3%A3Apara-el-dise%C3%B1o-e-implementaci%C3%B3n-de-Portales-Estatales.html?menuderecho=2
Costa Rica	Guía para desarrollo de sitios web. http://www.gobiernofacil.go.cr/e-gob/gobiernodigital/documentos/Guia%20para%20el%20Desarrollo%20de%20Sitios%20Web%202.0%20-%20Gobierno%20de%20Costa%20Rica.pdf

No obstante, si bien seguir las recomendaciones recogidas en los documentos citados previamente puede hacer más eficiente la interacción del usuario con un sitio web, no necesariamente resulta suficiente para garantizar el mejor nivel de usabilidad. En este contexto, es decir, cuando aun habiendo tenido en cuenta recomendaciones y buenas prácticas en los diseños, se pueden evaluar los sitios web desarrollados utilizando las distintas técnicas comentadas en la Sección 6.2.

Con la identificación y documentación de buenas prácticas, y la confección de los documentos recopilados en la Tabla 6-1, el objetivo es ir más allá en materia de logros de la usabilidad, es decir, proveer al diseñador de sitios web de herramientas que le faciliten su trabajo a la hora de lograr usabilidad en sus desarrollos.

ACTIVIDADES 6.8



- ▶ Elija y revise uno de los documentos recogidos en la Tabla 6.1. Preste especial atención a cómo se organiza la información que documentan y al nivel de detalle al que se documentan las buenas prácticas y las recomendaciones.
- ▶ Diferencie la información recogida en esos documentos de los principios sugeridos por Jakob Nielsen y Bruce Tognazzini.

SOLUCIÓN

La información documentada en las guías y manuales identificados en la Tabla 6.1 son guías de estilo y éstas se documentan a un nivel de abstracción más bajo que el utilizado para recoger principios de diseño. En este sentido estas guías son más objetivas y tienen una aplicación más directa.

6.9 NAVEGACIÓN FÁCILMENTE RECORDADA FRENTES A NAVEGACIÓN REDESCUBIERTA

Los elementos de navegación son los que permiten al usuario moverse por un sitio web y encontrar la información que busca. Con un sistema de navegación consistente y predecible se ayudará al usuario a hacerse una idea de la organización del sitio web.

La Web es navegación. La interacción más básica de cualquier usuario es pulsar sobre los enlaces o vínculos para moverse a través de la información en un espacio que abarca cientos de millones de documentos.

Una interfaz de navegación estará bien diseñada y, por tanto, será fácil de recordar y ofrecerá soporte para la realización de actividades al usuario, si fue diseñada para dar respuesta a distintas preguntas básicas. Entre esas preguntas están las siguientes:

- El diseño de toda página web debe responder al usuario la pregunta *¿dónde estoy?*: La Web es un medio de enormes dimensiones que aporta muy pocas pistas sobre la ubicación de la información. Debido a la naturaleza de la Web, un usuario puede pasar de un sitio web a otro sin apenas darse cuenta. Por ello es absolutamente necesario que el usuario sepa en todo momento dónde está. Esta necesidad está presente tanto en relación con la Web, como en relación con la estructura de un sitio web concreto.
- Una segunda pregunta que debe responderse al usuario visitante de un sitio web es *¿quién me está contando todo esto que encuentro en este sitio?*: Esta es una cuestión que se da por supuesta demasiadas veces, pero que no siempre se logra. Siempre hay que decirle al usuario, a través de buenas prácticas de diseño, quién ha creado la página. Esta información proporcionará mayor confianza en los contenidos recogidos en el sitio web.
- En tercer lugar, el usuario debe ser capaz de, alcanzada una página web, *¿qué puede encontrar allí?*. Todos los documentos necesitan títulos claros que capturen su atención. El título del documento es normalmente el primer elemento que los navegadores visualizan. En páginas con muchos gráficos, el título puede ser lo único que un usuario vea desde el principio, mientras se carga el resto de la página. Además, el título de una página será la frase que guardará el usuario en favoritos y le va a servir para identificar la página. Un título ambiguo o que cree confusión no ayudará en nada a los usuarios cuando pretendan recordar en qué página encontró determinada información.
- Las preguntas *¿dónde he estado?* y *¿qué he visitado ya?*, son otras preguntas que el usuario debiera poder responder sin esfuerzo. El usuario puede haber visitado otras partes del sitio, hay que evitar que se pase el rato dando vueltas en círculo porque no puede distinguir qué partes ha visitado y cuáles no. Esta información además puede ayudar al usuario a comprender la estructura del sitio. Los Navegadores incorporan unas características que ayudan al usuario a saber dónde han estado y por donde han venido: el botón atrás, y la lista de páginas visitadas (historial), y la carpeta de favoritos. Lamentablemente, son muchos los diseñadores que “secuestran” las barras de herramientas de los navegadores, privando a los usuarios de una herramienta primordial. ¿Cómo se sentiría usted si al abrir un documento de Word desparecieran todas sus barras de herramientas?
- Es importante saber dónde está el usuario, pero también, *¿dónde puedo ir de donde esté?* Esta pregunta parece fácil de responder con los elementos de navegación y cualquier otro enlace que haya en el documento, pero, claro está, siempre que estén claramente identificados como tales.
- Por último, la pregunta *¿cuándo?* también resulta interesante al usuario en muchas ocasiones. En la Web hay millones de páginas no actualizadas desde hace años y el diseñador debe ayudar a sus visitantes a saber si su información es actual o no, incorporando fechas o referencias que le permitan saber lo actualizada que está la información que el usuario está visualizando.

ACTIVIDADES 6.9



- Analice la página principal de www.amazon.es (una captura es mostrada en la Figura 6.4) identifique cómo se da respuesta a las preguntas mencionadas con anterioridad en esta sección.

Figura 6.4. Página principal de Amazon en España

SOLUCIÓN

El sitio web cuenta con título y logo, situado en la parte superior izquierda. Cuenta con un menú de navegación, que facilita al usuario saber qué puede encontrar y a qué secciones puede ir. Además, cuenta con distintas secciones en la página principal destinada a destacar novedades, ofertas, productos demandados, etc. En la parte inferior de la página existe referencia al *copyright* de los contenidos. Una vez el usuario empieza a navegar por el sitio web, el menú de navegación se ajusta en función de la sección que el usuario esté visitando, mostrando secciones específicas de la sección visitada.

6.10 FACILIDAD DE NAVEGACIÓN EN LA WEB

Los sitios web, en líneas generales, deben ser atractivos y satisfacer el interés de los usuarios que los visitan, pues de ello dependerá que compren, en unos casos, y que regresen en otros. El reto, a la hora de diseñar sitios web, es que esos sitios sean amigables, intuitivos, con aplicaciones sencillas, rápidas para encontrar lo que se busca y para ofrecer sus servicios y presenten facilidades de uso. El uso en la Web se traduce en actividades de navegación realizadas por los usuarios que la visitan. Para lograr esos objetivos de facilidad de navegación se deben tener presentes algunas recomendaciones, entre ellas cabe destacar las siguientes:

- Ofrecer enlaces “Volver”: el sitio y las páginas web deberán incluir enlaces “volver”. El usuario puede que no sepa, en un determinado momento, a dónde regresará al hacer clic en un enlace o vínculo. A la hora de incluir enlaces y botones en las páginas web es importante asegurarse de que dichos enlaces y las etiquetas que en ellos se incluyan describan y anticipen al usuario hacia dónde va a “volver” por ejemplo “volver al inicio”.
- Una buena práctica de diseño de sitios web es lograr que el menú de navegación esté siempre visible. Una sugerencia de diseño relacionada con la navegación es asegurarse que en los diseños se muestre siempre al usuario el menú más importante del sitio. El hecho de que el usuario pueda acceder al menú principal de navegación desde cualquier parte del sitio en el que se encuentre le evoca al usuario la sensación de tener el control sobre sus movimientos a través del sitio web. Si la estructura y la navegación se ha pensado de manera adecuada el usuario debería ser capaz de acceder a la información que busca en un sitio web utilizando no más de tres clics de navegación.
- Otro *feedback* (información, realimentación) que un sitio web debe facilitar al usuario que lo visita es la identificación de la situación actual en el sitio. El diseño de un sitio web deberá ofrecer información de dónde se encuentra el usuario en todo momento. Por ejemplo, si el usuario se encuentra en un sitio web en la sección que tenga la dirección url siguiente: <http://www.midominio.com/contacto>, el usuario debería recibir *feedback* adicional de que se encuentra en la sección “Contacto” de ese sitio web.
- Mencionar también que las facilidades de navegación tienen relación con la velocidad de conexión y de carga de los sitios web. Se debe evitar diseñar sitios pesados, que impidan la rapidez de descarga. Si una página tarda mucho tiempo en cargar lo habitual es que el usuario la abandone. Para ello se deben cuidar y optimizar las imágenes, los gráficos y las animaciones incluidas en el sitio web.

Algunos ejemplos de menús y elementos de navegación con diferentes presentación y formato se recogen en la Figura 6.5.

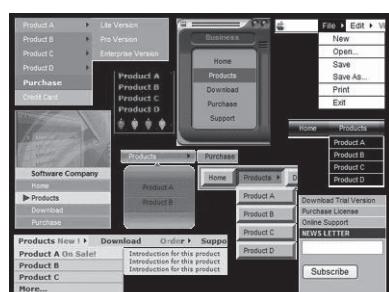


Figura 6.5. Distintos ejemplos para soportar la navegación en sitios y páginas web

ACTIVIDADES 6.10



- Diseñe e identifique el menú principal de navegación de una web para un restaurante. Para ello identifique las principales secciones y la información más relevante que se ofrecerá en dicho sitio web.

6.11 VERIFICACIÓN DE LA USABILIDAD EN DIFERENTES NAVEGADORES Y TECNOLOGÍAS

Tal y como se comentó en los capítulos 2, 3 y 4 es necesario asegurarse que todos los desarrollos de sitios web que se hagan se muestren de la misma manera en la mayor cantidad posible de navegadores. Esto es obligado ya que los usuarios potenciales de un sitio web es todo el mundo y cualquiera puede acceder al sitio web desde distintos navegadores y utilizando distintas tecnologías y dispositivos.

Aunque las páginas de un sitio web cumplan con los principios y buenas prácticas documentadas es recomendable que, de manera adicional, se compruebe que las páginas de un sitio web se visualizan correctamente en los navegadores más utilizados.

Una solución para probar los sitios web en distintos navegadores es instalarlos en local. Es la menos cómoda pero la más segura. Sin embargo, hay otras posibilidades, por ejemplo, usar distintos sitios web que realizan capturas de pantalla de un sitio web en diferentes navegadores y sistemas operativos. Esto evita al diseñador tener que instalar varios navegadores. Sin embargo, el inconveniente es que no se pueden probar, en tiempo real, las modificaciones que se hagan. Ejemplos de este tipo de sitios web son los siguientes:

- Browsershot¹⁰⁷ (gratuito).
- Browsecam¹⁰⁸ (comercial).

ACTIVIDADES 6.11



- Utilice Browsershot o Browsecam para conseguir capturas de visualización del sitio web correspondiente a la institución del Defensor del Pueblo (<http://www.defensordelpueblo.es>) en distintos sistemas operativos, navegadores y plataformas. Analice los resultados obtenidos.

¹⁰⁷ <http://browsershots.org/>

¹⁰⁸ <http://www.browsecam.com/>

6.12 HERRAMIENTAS Y TEST DE VERIFICACIÓN

La evaluación y verificación de la usabilidad es una labor menos objetiva que las actividades de evaluación de la accesibilidad vistas en el Capítulo 5. En el caso de la accesibilidad hay muchas herramientas que inspeccionan el código HTML de las páginas que conforman un sitio web. Pero en el caso de la usabilidad el escenario es distinto, ya que directamente del código de un sitio o página web no puede evaluarse la usabilidad en toda su dimensión. Como se mencionó a lo largo del presente capítulo la decisión última sobre la usabilidad de un sitio web esatará en sus usuarios.

En cualquier caso, existen herramientas que dan soporte a actividades de evaluación de la usabilidad de sitios web y que permiten identificar problemas en páginas y sitios web. Las herramientas que permiten evaluar o estimar la usabilidad tratan de dar soporte a las actividades de análisis de la usabilidad comentadas en el apartado 6.2 y muchas de ellas se basan en tomar muestras del comportamiento de los usuarios ante un sitio web, para identificar dónde pulsan, qué secciones o páginas visitan, si realizan sus tareas o si éstas quedan incompletas. Con las herramientas referidas en esta sección se pueden soportar actividades de evaluación de la usabilidad de un sitio web y descubrir qué barreras u obstáculos encuentran los usuarios en ellas.

Las herramientas de evaluación de la usabilidad se pueden agrupar atendiendo a muchos criterios. Seguidamente se identifican cuatro grupos de herramientas:

- En un primer grupo se pueden encontrar las herramientas basadas en *mapas de calor*. En estos mapas se destacan, con colores más intensos, áreas o zonas de un sitio web que reciben más pulsaciones por parte de los usuarios que visitan el sitio web. Se incluirían en esta sección aquellas herramientas que permiten asignar diferentes intensidades a distintas partes o localizaciones de una página web en función del número de clics que han realizado los visitantes de esos sitios web. Ejemplo de este tipo de herramientas son *ClickHeat*¹⁰⁹, *CrazyEgg*¹¹⁰ y *Clickdensity*¹¹¹.

Hay herramientas alternativas, a las mencionadas anteriormente, que permiten generar esos mapas de calor atendiendo no al número de clics sino a dónde mira el usuario cuando visita dicho sitio web. La empresa Tobii facilita material que da soporte a este tipo de estudios conocidos con el nombre de *eye-tracking*.

- Otro grupo de herramientas son las basadas en la grabación de pantallas, es decir, aquellas herramientas que soportan el seguimiento de los visitantes de un sitio web y crean, paralelamente, un vídeo a modo de log de la actividad realizada por el usuario. Ejemplos de este tipo de herramientas es *Navflow*¹¹².
- Simulación de usuarios. Las herramientas de simulación de usuarios permiten recrear la utilización de un sitio web por usuarios artificiales permitiendo obtener resultados parecidos a los que se podrían obtener mediante las pruebas tradicionales de usabilidad contando con usuarios reales. Un ejemplo de estas herramientas es *Selenium IDE*¹¹³, un plugin para Firefox para realizar pruebas de sitios web. Es decir *Selenium* es una

¹⁰⁹ <http://www.labsmedia.com/clickheat/index.html>

¹¹⁰ <http://www.crazyegg.com/>

¹¹¹ <http://www.clickdensity.com/>

¹¹² <http://navflow.com/>

¹¹³ <http://seleniumhq.org/projects/ide/>

herramienta que permite especificar mediante scripts comportamientos o acciones de un usuario ficticio y poder evaluar cuál serían los resultados de esos comportamientos.

- Obtener *feedback* y comentarios del usuario. Con este tipo de herramientas se puede recibir información y comentarios por parte de los usuarios visitantes a un sitio web. En general, en los sitios web se puede reservar una página o espacio para que el usuario pueda enviar comentarios o sugerencias sobre el contenido disponible en el sitio web. Esos comentarios pueden enviarse utilizando una cuenta de correo u ofreciendo un formulario destinado a dicho fin.

ACTIVIDADES 6.12



- Visite la página web de la herramienta CrazyEgg (<http://www.crazyegg.com/>) y describa cómo funciona y cómo podemos conseguir instalar todo lo necesario para usar dicha herramienta y conseguir mapas de calor de un sitio web sobre el que tengamos derechos.

6.13 CONCLUSIONES

El desarrollo de interfaces web amigables y usables supone un verdadero reto para todo desarrollador. El éxito en esta tarea depende de la capacidad del diseñador de un sitio web para responder a distintas preguntas. Entre esas preguntas están las siguientes: ¿quién utilizará mi sitio web?, ¿para qué lo utilizará? y, ¿qué espera encontrar en él? Una vez disponga de dichas respuestas la tarea del diseñador pasará por elaborar prototipos y valorar diferentes alternativas de diseño hasta encontrar una que tenga en cuenta las respuestas anteriores.

Como se ha tratado de resaltar a lo largo de este capítulo, la creación de sitios web con interfaz amigable es una labor que depende de la experiencia del diseñador. Para transmitir esa experiencia dentro de la comunidad de diseñadores, este capítulo ha hecho referencia a dos conceptos. El primero de esos conceptos ha sido el relacionado con los principios de diseño, en este sentido se ha hecho referencia a los principios propuestos por Jakob Nielsen y Bruce Tognazzini. El segundo grupo de documentos son estándares de facto y guías de estilo, de las que también se han identificado buenas prácticas y recomendaciones.

Para evaluar la calidad, desde el punto de vista de la usabilidad de un sitio web, se han identificado distintas técnicas y herramientas para dar soporte a distintas actividades de evaluación. Las herramientas identificadas no se basan en revisar el código asociado a una página web, como era el caso de las herramientas y técnicas principales identificadas para evaluar la accesibilidad de un sitio web. Por el contrario, para evaluar la usabilidad hay que contrastar que el usuario encuentra o puede encontrar problemas al utilizar un sitio web y, por tanto, los principales mecanismos de evaluación de la usabilidad pasan por hacer pruebas con usuarios o recurrir a expertos y hacer evaluaciones empíricas.



RESUMEN DEL CAPÍTULO



En este capítulo se han caracterizado los conceptos de interfaz amigable y de usabilidad. Para ello se ha hecho referencia a distintos estándares internacionales elaborados por la Organización Internacional de estandarización. Junto a la accesibilidad, la usabilidad es otro factor importante que un diseñador de sitios web debe tener presente cuando diseña sitios web. La usabilidad ofrece dos puntos de vista: producto y proceso, e influye en cómo el diseñador ofrece los contenidos presentes en una web, estructura la web, elabora un diseño visual para ella y determina cómo interactuar y la funcionalidad que en ella ofrece.

Este capítulo recomienda que para crear sitios web amigables se deben poner en práctica técnicas de Diseño Centradas en el Usuario, y para validar los diseños realizados se deben utilizar técnicas de análisis de la usabilidad. Algunas de estas técnicas están soportadas por herramientas software.



EJERCICIOS PROPUESTOS



- 1. Diseñe, confeccionando dos prototipos, un sitio web que informe sobre el horario de autobuses urbanos que haya en su ciudad. El sitio web deberá presentar, al menos, los siguientes elementos:
 - Una página principal con un logo, un título, un eslogan y una barra de navegación.
 - La página principal presentará el servicio de autobuses de manera general, recogiendo sus principales características. Por ejemplo, informará de la dirección su dirección, teléfono y fax de contacto, informará sobre las distintas líneas de autobuses y del horario en el que se presta el servicio.
 - Confeccione también una entrada en el menú de navegación principal para cada línea de autobuses y en la página asociada describa su recorrido concreto.
 - Incluya en la ventana principal enlaces a otras web relacionadas, como pudieran ser la web de su ayuntamiento o de su comunidad autónoma.
- 2. Partiendo de los prototipos confeccionados, identifique y justifique la elección del diseño más prometedor. Utilice para ello los principios de diseño propuestos por Nielsen y Tognazzini.
- 3. Implemente el diseño realizado y desarrolle el prototipo elegido eligiendo colores, fuentes y, en definitiva, características más concretas de su sitio web.
- 4. Visualice tu implementación en varios navegadores y depura posibles incompatibilidades o problemas ligados a su correcta visualización en ellos.
- 5. Utilice los servicios facilitados por *Clickdensity*¹¹⁴ para identificar qué secciones son más visitadas en el sitio web que ha construido.

¹¹⁴ <http://www.clickdensity.com/>



TEST DE CONOCIMIENTOS



1 Si atendemos a la usabilidad de un producto software y consideramos el producto en sí, ¿qué estándar o norma internacional define de manera más adecuada ese punto de vista para la usabilidad?

- a) La norma ISO/IEC 9126-1.
- b) La norma ISO 9241-11.
- c) La definición de usabilidad no puede atender a ese punto de vista.

2 Las técnicas de evaluación de la usabilidad en la que no necesitaríamos la presencia de usuarios para llevarlas a cabo serían:

- a) Las que se agrupan bajo actividades de investigación.
- b) Las que se agrupan bajo actividades de inspección.
- c) Las que se agrupan bajo actividades de prueba con usuarios.

3 Los mapas de calor son:

- a) Una representación gráfica de una página web en la que se resaltan con diferentes colores unas zonas concretas de la página, basándose en unos determinados criterios (por ejemplo, la visualización o el pulsado con el ratón).
- b) Representaciones gráficas en las que se resaltan y enfatizan las novedades de un sitio web.
- c) Representaciones gráficas de un sitio web en el que se describe su estructura y organización.

4 La última palabra en materia de logro de usabilidad de un producto software la tienen:

- a) Los expertos en usabilidad.
- b) Los usuarios y visitantes de las páginas web.
- c) Los responsables de un sitio web.

5 ¿Qué criterios de juicio se tienen en cuenta a la hora de revisar un sitio web para identificar posibles problemas o barreras en su uso?

- a) La organización de su código fuente.
- b) Su compatibilidad con los distintos navegadores y tecnologías.
- c) Su contenido, diseño visual, estructura y navegación, funcionalidad e interactividad.

Índice Alfabético

Símbolos

<audio>, 108, 164
<bgsound>, 107
<embed>, 107
<iframe>, 118
, 101
<object>, 110, 117
<source>, 108, 116
<video>, 116, 164

A

Absolute, 80
Accesibilidad, 170
ActionScript, 121
Adobe Edge, 127
Adobe Flash Professional, 122
Agrupamientos, 52
Algoritmo lzw, 100
Anidado común, 53
Anidamiento de selectores adyacentes, 56
Anidamiento de selectores hijos, 54
Anidamientos, 53
Animaciones, 120, 145, 150
append(), 140
ATAG, 175
Atributos, 60
Atributos border, 65
Atributos de enlaces, 73
Atributos de fondo, 72
Atributos de fuentes, 72
Atributos del contenido, 67
Atributos de listas, 73
Atributos de posición, 62
Atributos de párrafos, 72
Atributos margin, 63
Atributos padding, 64
Atributos visibilidad, 73

AtubeCatcher, 115

Audio, 104
autor, 99
avi, 114

B

bind(), 137
Browsercam, 220
Browsershot, 220
Buenas prácticas, 51, 57, 91, 215

C

Calidad de un producto software, 202
Calidad visual, 100
Callbacks, 142
Cellsea Free Web Photo Editor, 103
Centímetros, 61
Clickdensity, 221
ClickHeat, 221
Contenido web, 175
Cooties (Blue Dojo), 127
Copyleft, 97
Copyright, 97
CrazyEgg, 221
Creative Commons, 97
css(), 140
CSS, 44
CSS3, 126, 127, 128
Códec, 105, 114

D

Derechos compensatorios, 97
Derechos de autor, 96, 99
Derechos de remuneración, 96
Derechos económicos, 96
Derechos exclusivos, 96
Derechos morales, 96

Detectar errores, 89
 Diseño Centrado en el Usuario, 202
 DivX, 114
 DOM, 135
 DoSize, 104

E

Efectos de desvanecimiento, 147
 Efectos de plegado, 149
 Efectos visuales, 145
 em, 61
 eps (archivo postscript), 101
 Especificidad, 83, 84
 Eventos de ratón, 155
 Eventos de teclado, 160
 Eventos principales de ventana, 162
 Everystockphoto, 98
 Eye-tracking, 221

F

Fauxto, 103
 Fixed, 82
 fla, 101, 121
 Flickr, 98
 Formatos de imagen, 100
 Free Studio, 115

G

GIF Construction Set Professional, 121
 GIF (Graphic Image File Format), 100
 GifMake, 121
 Gifs animados, 120
 GIMP (GNU Image Manipulation Program), 102, 121

H

H.264, 114
 Height, 67
 Hera. Véase
 Herramientas de conversión, 106
 Herramientas de optimización de imágenes, 104
 Herramientas de validación, 89
 Herramientas de validación automática, 194
 Herramientas para validar, 88

HTML5, 106, 126, 134
 Hype, 127

I

Image Optimization, 104
 ImageTool, 103
 Imágenes vectoriales, 101
 Inherit, 82
 innerWidth(), 142
 Intellectual property, 99
 Interfaz web amigable, 202
 ISO 9241\1998, 203
 ISO/IEC 9126-1\2001, 202

J

JavaScript, 111, 126
 JPEG (acrónimo de, Joint Photographic Experts Group), 100
 jQuery, 123, 134

L

Licencia de software, 97
 Licencias Coloriuris, 97

M

Mapa de bits, 101
 Mapas de calor, 221
 Milímetros, 61
 Miro Video Converter, 115
 mkv, 114
 Modelo de cajas, 60
 Motor de renderizado, 128
 mov, 114
 mp3, 105
 mp4, 115

N

Navflow, 221

O

offset(), 142
 Ogg, 105
 ogv, 114
 outerWidth(), 142

P

Pautas 2.0, 176
 Pautas de accesibilidad 2.0, 176
 Pautas de Accesibilidad al Contenido en la Web, 175
 Pautas de Accesibilidad para Agentes de Usuario (UAAG), 175
 Pautas de Accesibilidad para Herramientas de Autor (ATAG), 175
 pdf, 101
 Photoshop, 102
 Picas, 61
 Picasion, 121
 Picfindr, 98
 Pixlr, 102
 Pixsy, 98
 Plantillas CSS3, 90
 player, 111
 PNG (Portable Network Graphics), 101
 Precedencia de estilos, 79, 83
 prepend(), 140
 Propiedad intelectual, 96, 99
 ps, 101
 Pulgadas, 61
 Puntos, 61

R

Real Audio, 105
 Recomendaciones de optimización de imágenes, 103
 Regla @import, 87
 Relative, 81
 Resolución de imagen, 103

S

Selectores, 45
 Selectores basados en clases, 47
 Selectores basados en etiquetas, 45
 Selectores basados en identificadores, 49
 Selenium IDE, 221
 Sencha, 127
 Shockwave, 122
 Static, 80
 Superposición de cajas, 79
 svg, 101
 swf, 101, 114, 121

SWF Easy, 122

Swiffy, 127

T

Taw, 195
 Técnicas de inspección, 204
 Técnicas de prueba con usuarios, 204
 Técnicas de sondeo o indagación, 204
 text(), 142
 toggle(), 139
 Tratamiento digital de imágenes, 102

U

UAAG, 175
 unbind(), 137
 Unidades de medida, 61
 Usabilidad, 202

V

Vídeo, 114
 Vincular hojas de estilo, 85
 Vincular hojas de estilo en cascada externa, 86
 vqf, 105

W

W3C, 44, 171
 WAI, 170
 Wallaby, 127
 wav, 105
 WCAG 1.0, 173
 WCAG 2.0, 174
 WebKit, 128
 webm, 114
 Width, 67
 wma, 105
 wmf, 101

X

Xvid, 114

Z

z-index, 79



La presente obra está principalmente dirigida a los estudiantes del Ciclo Formativo **Desarrollo de Aplicaciones Web** de Grado Superior, en concreto para el módulo profesional **Diseño de Interfaces Web**.

Los contenidos incluidos en este libro abarcan los conceptos básicos del diseño de interfaces web, que van desde los conceptos básicos del desarrollo web, las hojas de estilo CSS, el manejo de recursos multimedia, hasta la programación de animaciones con jQuery. Todo ello enmarcado dentro de las pautas y criterios que definen la usabilidad y accesibilidad de las interfaces.

Los capítulos incluyen actividades resueltas y ejemplos con el propósito de facilitar la asimilación de los conocimientos tratados. De esta manera, se pretende que el estudiante asimile la teoría desde una perspectiva práctica.

Así mismo, se incorporan test de conocimientos y ejercicios propuestos con la finalidad de comprobar que los objetivos de cada capítulo se han asimilado correctamente.

Además, reúne los recursos necesarios para incrementar la didáctica del libro, tales como un glosario con los términos informáticos necesarios, bibliografía y documentos para ampliación de los conocimientos.

 En la página web de **Ra-Ma** (www.ra-ma.es) se encuentra disponible el material de apoyo y complementario.

