



**Y**

**CSS3**

INTRODUCCION

Le damos la bienvenida a este **CURSO DE HTML Y CSS**. El curso que acaba de comenzar en nuestro centro está realizado según las directrices pedagógicas que más se adecuan a la enseñanza de la informática. Le damos las gracias por elegirnos.

La evolución que ha seguido Internet y las posibilidades que ofrecen las nuevas versiones de los navegadores ofrecen unas oportunidades de diseño de páginas web más funcionales y potentes por lo que se hace necesario conocer los estándares de programación más actuales.

La profunda renovación que ha supuesto la evolución de **HTML** hace imprescindible conocer las aportaciones de las nuevas versiones para adaptar los diseños de las páginas web a los formatos más modernos.

Por otro lado, las hojas de estilo **CSS**, imprescindibles en las últimas especificaciones de **HTML**, suponen un avance verdadero en lo que respecta al diseño, las representaciones gráficas y la apariencia de las páginas web con nuevas aportaciones que les dan un formato más actualizado.

Por todo esto, y aunque no sabemos sus expectativas a la hora de comenzar este curso, estamos seguros de que al final se sentirá satisfecho al comprobar que es capaz de crear páginas web de carácter profesional con tan sólo **HTML** y hojas de estilo **CSS**.

**FRAY LUIS CENTRO ACADÉMICO** pretende conseguir con este **CURSO DE HTML Y CSS** los siguientes objetivos:

OBJETIVOS DEL CURSO DE HTML Y HOJAS DE ESTILO CSS.

* Conocer las etiquetas y sus funciones del lenguaje **HTML**.
* Conocer el funcionamiento de las hojas de estilo **CSS** y sus parámetros.
* Ser capaces de crear una página web desde cero utilizando las últimas especificaciones de **HTML** y **CSS**.
* Aplicar **HTML** y **CSS** conjuntamente para dar formato profesional a una página web basada en estándares.
* Crear páginas web de carácter profesional siguiendo las directrices y reglas del World Wide Web Consortium, (W3C), y validadas por él, garantizando por lo tanto la calidad del desarrollo y diseño de la página web.
* Aprender a crear páginas web más o menos sofisticadas, más interesantes, más atractivas y más útiles.

METODOLOGIA

El **CURSO DE HTML Y CSS** que tiene entre sus manos está realizado según las directrices pedagógicas que más se adecuan a la enseñanza de la informática y recoge la experiencia de más de 25 años de enseñanza de nuestro centro. Efectivamente; la metodología que seguimos en nuestros cursos de informática presentan las siguientes características pedagógicas:

* **ENSEÑANZA PERSONALIZADA**:

Cada alumno puede seguir el ritmo de aprendizaje que considere más oportuno, adecuándose a su propia capacidad de concentración, de comprensión, de rapidez lectora, de fijación de conceptos, del nivel motivacional, del cansancio, etc.

Por tanto, la velocidad en el avance dependerá de cada alumno dado que el mismo se producirá con la realización correcta de todos los ejercicios propuestos.

Es por esto que cada uno de nuestros profesores atiende personalmente a cada alumno, adecuándose a las peculiaridades particulares y profesionales de cada alumno.

* **ENSEÑANZA EMINENTEMENTE PRACTICA**:

Cada uno de nuestros alumnos dispone de un ordenador personal de alta gama, conectado en red, impresoras láser a color y escáner, para realizar las actividades que se le van proponiendo.

El alumno, desde la primera hora de clase, está trabajando y experimentando con **HTLM Y CSS**.

* **DIFICULTAD CRECIENTE**:

La explicación avanza en pequeños pasos de dificultad creciente.

La comprensión conseguida se confirma mediante la ejercitación en el propio ordenador, permitiendo aprender desde el error.

* **ENSEÑANZA INTERACTIVA**:

El aprendizaje exige constante participación activa del alumno, dado que se necesita la ATENCIÓN-CONCENTRACIÓN del mismo para poder avanzar en los contenidos, ya que las diferentes cuestiones sólo se pueden resolver si se presta suficiente atención a los contenidos que se proponen.

* **ENSEÑANZA MOTIVADORA**:

La enseñanza personalizada, eminentemente práctica, interactuando constantemente con el ordenador, además de la comprobación inmediata del éxito en las tareas que se proponen, ejerce como animador hacia el esfuerzo siguiente.

Teniendo en cuenta estas características pedagógicas de nuestro método, nos permitimos hacerle las siguientes:

Recomendaciones pedagógicas para un aprendizaje eficaz

* El orden en que se han editado los temas de cada bloque tiene una importancia pedagógica, al menos para un primer estudio de los mismos. La razón es que normalmente en cada tema se utilizan conceptos que se apoyan en contenidos explicados en los temas precedentes.
* Cada unidad didáctica, cada tema, guarda una estructura lógica interna. Por tanto, es recomendable estudiarla en el orden que se propone, puesto que los contenidos que se van explicando fundamentan los que a continuación aparecerán y viceversa.
* Es **muy importante realizar correctamente todos y cada uno de los ejercicios propuestos**. Nuestra experiencia confirma que los alumnos que se esfuerzan en realizar bien los ejercicios, comprenden y retienen con mayor facilidad los contenidos didácticos.
* El manual del curso está expresamente realizado pensando en usted y en sus ganas de aprender. **Léalo detenidamente**. En él está explicado, muy resumidamente, todo lo necesario para realizar los ejercicios que se le proponen. A veces, es conveniente realizar una especie de resumen-esquema que le ayudará a recordar con rapidez lo que ya ha estudiado.
* **Si tiene alguna duda, no pase adelante**. Resuélvala antes de continuar. Pregunte a su profesor. ¡Para eso está!
* El buen resultado del curso no guarda relación directa con la velocidad a la que son recorridos los temas. No se debe pretender abarcar demasiado espacio en poco tiempo. Cada contenido requiere su tiempo para asentarse convenientemente. Es tontería avanzar sin aprenderse bien los contenidos. Más adelante tendrá que volver a repasar y esto le hará perder tiempo.
* **¡MUY IMPORTANTE!** Para obtener el certificado o diploma de estudios expedido por el centro, correspondiente a este curso, será necesario demostrar que ha adquirido los conocimientos necesarios mediante una prueba o Control de Aprendizaje personal así como presentar realizados correctamente todos los ejercicios del temario.

Estamos seguros de que siguiendo estas recomendaciones dominará de una manera sencilla y cómoda todos los contenidos de éste **CURSO DE HTML Y CSS** que ahora comienza.

CAPÍTULO 1: CONCEPTOS PREVIOS

INTRODUCCIÓN

Este curso está pensado para todas aquellas personas que deseen desarrollar, crear y publicar páginas web de carácter profesional, más o menos sofisticadas, consiguiendo sitios web más interesantes, atractivos y útiles, utilizando para ello, la última tecnología del lenguaje **HTML** y algunas utilidades de apoyo que forman lo que podríamos llamar la “familia **HTML**”.

A la hora de elaborar este manual hemos procurado, como en el resto de los cursos de Fray Luis Centro Académico, fomentar la participación del alumno con numerosos ejemplos y ejercicios prácticos, pues consideramos que es la mejor manera de aprender cualquier utilidad informática.

Para ello, durante todo este curso y como tema principal vamos a crear una página Web sobre el turismo en España y en Valladolid, aunque también deberá crear otras páginas en los ejercicios de cada lección.

Es muy importante que siga paso a paso las instrucciones de este manual y que realice todos los ejercicios correctamente puesto que, cada lección y cada ejercicio, está basado en los anteriores.

Elementos necesarios para crear una página web.

Las páginas Web se crean utilizando un lenguaje llamado **HTML** complementado con unas hojas de estilo **CSS**.

Una página **HTML** y sus correspondientes estilos **CSS**, se crean en un editor de texto y luego se carga en un navegador. Por lo tanto los medios imprescindibles para crear páginas Web son esos:

**•Un editor de texto**: Vale cualquier editor de texto, nosotros utilizaremos el bloc de notas incluido gratuitamente en Windows. Como norma general no se pueden crear archivos **HTML** en procesadores de texto aunque algunos, como Word, ofrecen la posibilidad de guardar los ficheros como "Sólo texto" para que no introduzca órdenes de formateo, que pueden provocar errores al cargarlo en el navegador.

También existen programas que facilitan la escritura de código **HTML** y **CSS**, entre otros: Notepad++, Sublime Text, Home Site, UltraEdit, Arachnophilia, Quanta, BlueGriffon, Aloha Editor, Rendera, etc, etc. (Aunque los editores de **HTML** faciliten la tarea de escribir código **HTML**, recomendamos que para empezar y sobretodo aprender, utilice el bloc de notas. Cuando tenga más experiencia podrá utilizar el editor que más le convenza).

**•Un navegador del Web**: Este permite visualizar la página web y comprobar que todos los elementos están situados correctamente y que cumplen su función. En la actualidad existen los siguientes navegadores que entienden las últimas especificaciones de **HTML**: IE, Firefox, Chrome, Safari y Opera, todos ellos en sus últimas versiones. Nosotros utilizaremos preferentemente Firefox.

HISTORIA DEL HTML

El **HTML** (Hyper Text Markup Language), es el lenguaje utilizado por los navegadores para mostrar las páginas web a los usuarios. Este lenguaje nos permite utilizar textos, sonidos e imágenes para combinarlos y mostrarlos en la web a nuestro gusto. Además nos permite la introducción de referencias a otras páginas por medio de los enlaces hipertexto.

El **HTML** se creó con objetivos divulgativos. Al no preverse el enorme desarrollo que llegaría a tener internet con sus propiedades multimedia y sus diversos usos, el **HTML** no se creó con la suficiente planificación por lo que se han tenido que ir realizando modificaciones y mejoras a lo largo del tiempo.

El W3C, (World Wide Web Consortium), es el organismo que ha ido realizando esas modificaciones y mejoras. Es quien indica las especificaciones que tiene que tener el lenguaje **HTML** para que los navegadores puedan entenderlo y los desarrolladores de páginas web sepan a qué instrucciones atenerse.

En 1998, el W3C finalizó la especificación **HTML 4.0** que es la especificación en la que se basan la gran mayoría de las páginas web que podemos ver hoy en día.

El W3C siguió realizando estándares y creo un lenguaje llamado **XML**, con reglas más estrictas que el **HTML**, y recomendando que se dejara de utilizar este, ya que así se desarrollarían webs más consistentes y con menos errores.

Con el mismo objetivo, en el año 2000, el W3C reformuló el **HTML** para adaptarlo a las reglas más estrictas de **XML**. Nació así el **XHTML** que no fue muy aceptado por los desarrolladores de webs que siguieron programando en **HTML 4.0.**

El desarrollo de la versión 2.0 del **XHTML** que pretendía ser un punto de no retorno en el desarrollo de páginas web, conllevó que no fuera compatible con todo lo que existía hasta la fecha incluyendo al **HTML**.

Esta ausencia de compatibilidad y diversos problemas de tipo técnico, supuso un gran descontento entre los fabricantes de los navegadores, que en 2004 crearon un grupo disidente del W3C llamado WHATWG, (Web Hypertext Application Technology Working Group), formado por Firefox, Opera y Safari y a los que se les unió Google Chrome. Su objetivo consistió en crear un lenguaje **HTML** mejor, con nuevas aplicaciones para la web, pero manteniendo la compatibilidad con los navegadores existentes.

Durante unos años, W3C y WHATWG, trabajaron en paralelo hasta que en 2006 decidieron unir sus fuerzas y trabajar conjuntamente abandonándose el **XHTML 2.0** y naciendo la especificación **HTML5** que, a fecha de elaboración de este curso, aún no se ha terminado de fijar completamente. (Se ha anunciado que estará totalmente terminada para el 2022).

Por último, a la vez que la elaboración del estándar **HTML5**, se está reformulando este para adaptarlo a las reglas del **XML**, con el nombre de **XHTML5**.

Toda esta intensa historia tiene unas consecuencias importantes para los desarrolladores de páginas web y para este curso:

* Gracias a que el W3C tomó el liderazgo del desarrollo de **HTML5** tenemos la garantía de que se trata de un estándar gratuito que no depende de una o varias empresas.
* El hecho de que algunas empresas fabricantes de navegadores no participaran en un primer momento en el grupo WHATWG hace que no todos los navegadores actuales soporten todas las posibilidades de **HTML5** lo que habrá que tener en cuenta para que las páginas web que desarrollemos se visualicen en todos los navegadores lo mejor posible. Siempre que sea posible haremos referencia a la compatibilidad de los distintos navegadores con las últimas utilidades implementadas en **HTML5**.
* El **HTML5** no es un estándar terminado por lo que habrá nuevas posibilidades que aún no están implementadas y características actuales que en un futuro podrán desaparecer.
* Todas las características que estudiaremos en este curso están actualizadas a **HTML5** habiéndose eliminado aquellas de **HTML 4.0** que no se conservan en esta nueva implementación.
* A lo largo de este curso intentaremos utilizar la sintaxis estricta para asegurar la compatibilidad con **XHTML**.

HOJAS DE ESTILO CSS

Hasta **HTML 4.0**, la especificación de este lenguaje incluía órdenes destinadas al contenido de una página web así como al formato aplicado a dicho contenido. En **HTML5** se separa el contenido de la presentación dejando este último en manos de las **HOJAS DE ESTILO CSS**, (Cascading Style Sheets).

Así que las **HOJAS DE ESTILO CSS**, aunque no forman parte de **HTML5**, se convierten en elementos inseparables del **HTML**, y ya no es posible abordar su estudio por separado. Es por ello que a lo largo de este curso iremos alternando partes de **HTML** propiamente dicho y partes de **CSS**.

Las **hojas de Estilo** **CSS** es un lenguaje en evolución que también se identifica mediante números de versión, y la última es **CSS3**. Como **CSS3** ha ido evolucionando en paralelo a **HTML5**, muchos de sus componentes suelen considerarse como **HTML5**, pero es importante conocer que son diferentes, aunque sí están íntimamente ligados, como ya hemos comentado e iremos comprobando.

Resumiendo, las **HOJAS DE ESTILO CSS** son órdenes de formato, color y fuente del texto, color de fondo, bordes, etc, etc, que el lenguaje **HTML** y el navegador entiende y que permiten a los desarrolladores el control total sobre el estilo y el formato de las páginas web.

El desarrollo de las **HOJAS DE ESTILO CSS** se hizo necesario para dar más potencia al lenguaje **HTML** a la hora de aplicar formatos al documento y para facilitar la lectura y el depurado del código **HTML** que se hacía muy pesado al tener mezclado las órdenes de contenido, formateo y estilo de la página web.

Por último, tenemos que nombrar al **DHTML**, es posible que en alguna ocasión oiga hablar de él o del **HTML dinámico** como también se le conoce. Es preciso que conozca que no es otro lenguaje de creación de páginas web diferente. Más bien se trata de un concepto de programación web que engloba varias herramientas como son: **HTML**, **CSS** y lenguajes de Script como **JavaScript**, **VisualBasic Script**, **PHP** y otros que son objetos de otros cursos diferentes. (Pregunte a su profesor por ellos).

CAPÍTULO 2 - COMENZANDO A TRABAJAR

Ha llegado el momento de empezar a trabajar con **HTML**. Para familiarizarnos con él vamos a abrir una página Web ya creada que servirá de base para todo el curso.

NUESTRA PRIMERA PÁGINA

El nombre de la página Web que vamos a abrir es **turismo.html** (Lo encontrará en su carpeta de prácticas).

***Es el momento de realizar la práctica 2.1.***

LAS ETIQUETAS

El uso de las etiquetas es el principio esencial del lenguaje **HTML**. Hay dos tipos de etiquetas: de inicio y de cierre. En las etiquetas de inicio simplemente se escribe la orden, **<***orden***>**, mientras que en las etiquetas de cierre se escribe la orden precedida de una barra, **</***orden***>.** Por regla general, a toda etiqueta de apertura le corresponde una etiqueta de cierre. Pero también puede haber etiquetas sin cierre.

La porción de texto situada entre las etiquetas de inicio y de cierre se ve afectada por la orden de las etiquetas. Por ejemplo, la etiqueta **<h1>** y su correspondiente **</h1>** determinan que el texto situado entre ellas, “**Turismo en España**”, es un título de nivel 1. Al conjunto de etiqueta de apertura, etiqueta de cierre y texto u objeto incluido entre ellas, le denominamos **ELEMENTO**.

Un ejemplo de etiqueta sin cierre, también llamadas etiquetas únicas o vacías, es **<img>** cuyo significado veremos un poco más adelante.

A un **ELEMENTO** se le pueden aplicar varias etiquetas. Diremos entonces que las etiquetas están anidadas o enlazadas.

***Realice la práctica 2.2.***

PARTES DE UN DOCUMENTO HTML MÍNIMO

Un documento **HTML** tiene que tener como mínimo las siguientes partes:

**•EL DOCTYPE**: Indica al navegador a que versión de **HTML** se ajusta el documento.

Todo documento **HTML** debe comenzar, en su primera línea, por un doctype. En **HTML5** es muy sencillo:

**<!DOCTYPE html>**

El doctype le dice al navegador qué tipo de código fuente, **HTML** y **XHTML**, debe interpretar y representar. Lo usará para representar la página según los estándares de W3C por lo que se deberá visualizar de manera idéntica en todos los navegadores.

El doctype no es obligatorio, pero si es una buena práctica. En su ausencia, al no saber a qué reglas atenerse, el navegador se acoge a unas reglas genéricas propias de cada navegador, por lo que puede suceder, que el mismo código, se vea diferente en los distintos navegadores.

Por si acaso se encuentra con él en alguna página web que desee investigar su código, el doctype de **HTML 4.01** es:

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "**[**http://www.w3.org/TR/html4/loose.dtd**](http://www.w3.org/TR/html4/loose.dtd)**">**

Y en **XHTML 1.0**:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict //EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

Por último el doctype no debe ser considerado como una etiqueta **HTML**, sino como una declaración del lenguaje que estamos utilizando, por lo que no es necesario utilizar una etiqueta de cierre.

**•LA ETIQUETA <html>**: Indica al navegador que se trata de un documento **HTML**. Es el elemento raíz del documento. Se declara justo debajo de la declaración del doctype.

***Es el momento de realizar la práctica 2.3.***

Note que la etiqueta **<html>** puede incluir, como en este caso, el atributo **lang=”es”** que le dice al navegador que el documento que va a procesar está en español. Además, está información es muy valiosa para los motores de búsqueda cómo Google. Veremos más sobre los atributos de las etiquetas en un apartado posterior.

**•EL ENCABEZADO**: comprendido entre las etiquetas **<head>** y **</head>**, que se colocan justo después de la etiqueta **<html>** y contiene el encabezado del documento con una serie de información relativa al propio documento. El encabezado no se muestra en pantalla pero es de gran utilidad para el navegador, para los motores de búsqueda y para el desarrollador. Entre la información que irá en el encabezado del documento **HTML** está:

* El título del documento entre las etiquetas **<title> … </title>**, que explicamos en el siguiente apartado.
* Las declaraciones de las hojas de estilo **CSS**.
* Funciones o vínculo a archivos **JavaScript**.
* Información dirigida a los motores de búsqueda para mejorar el posicionamiento de la página.
* Información dirigida a los navegadores.
* Meta-etiquetas que indican la codificación, la descripción de la página, las palabras clave asociadas a la página, el nombre del autor, menciones de copyright, etc, etc.

Dentro del encabezado debemos destacar dos elementos que no deben faltar:

**A.- El Título del documento**.

Como ya hemos visto la etiqueta **<title>Turismo en España</title>** asigna un título a la página que se visualiza en la barra de título de la ventana del navegador y/o en las pestañas abiertas por el mismo. Igualmente es el título con que se guarda la página cuando la agregamos a favoritos.

Tiene especial importancia en lo que respecta al posicionamiento en buscadores ya que es en lo primero que se fijan en las búsquedas e indexaciones, por lo que habrá que procurar que el título de la página sea atractivo y sintético reflejando la temática de nuestro sitio web.

No se puede tener más de un elemento **<title>** en un documento **HTML**.

**B.- La codificación del documento**:

Indica al navegador el juego de caracteres, (charset), a utilizar. Ejemplos de codificación son: ASCII, Windows-1252, iso-, etc.

Habitualmente se utilizan los juegos de caracteres utf-8, iso-8859-1 e iso-8859-15.

Forma parte de las buenas prácticas del diseño de páginas web definir explícitamente la codificación del documento.

En **HTML5** se utiliza la etiqueta: **<meta charset=“utf-8” />** para indicar el juego de caracteres que vamos a utilizar en nuestra página web.

***Realice ahora la práctica 2.4.***

**•EL CUERPO**: comprendido entre las etiquetas <**body**> y <**/body**> y que es donde se sitúa el contenido de la página web que verá el usuario en su navegador.

Concluyendo: En cualquier código de cualquier página Web creada con este lenguaje, la primera etiqueta, a parte del **doctype**, será <**html**> y la última <**/html**>.

El texto escrito en el encabezado no se visualiza cuando se abre la página pero no por eso deja de tener importancia ya que el texto del encabezado situado además entre las etiquetas de título (<**title**> y <**/title**>) es el nombre que adoptará nuestra página cuando se guarde en la carpeta de favoritos y por el que lo encontrarán los buscadores. Por esta razón el título debe ser breve y descriptivo.

Dentro del cuerpo de la página está todo lo que queremos que aparezca en pantalla, texto, imágenes, videos, etc, etc…

LOS TÍTULOS

El texto normal se escribe simplemente entre las etiquetas <**body**> y <**/body**>. Pero observe que si queremos que el texto tenga una título deberemos escribir ese título entre las etiquetas <**h1**> y <**/h1**>, <**h2**> y <**/h2**>, etc. (Hasta el número seis). El número es indicador del tamaño, así el número uno corresponde al título de mayor tamaño o importancia y el seis al de menor tamaño.

Observe que no debe utilizar estas etiquetas de título para variar el tamaño de los caracteres, para eso utilizaremos las hojas de estilo **CSS**.

En nuestra página el título "**Turismo en España**" es el principal y por eso se encuentra entre las etiquetas <**h1**> y <**/h1**>. El título "**Introducción**" por ser de menor importancia se encuentra entre las etiquetas <**h2**> y <**/h2**>.

Tenga en cuenta que las etiquetas de título:

* Por defecto escriben el texto en negrita.
* Por defecto incluyen una línea vacía entre el contenido de la etiqueta y el resto del documento.
* Si desea conseguir caracteres más grandes o más pequeños que los que **HTML** proporciona por defecto, habrá que utilizar hojas de estilo **CSS**.
* Las que aparecen al principio de la página web son tenidas muy en cuenta en los motores de búsqueda e influyen en la clasificación y recomendación de los sitios web.

LOS PÁRRAFOS

El texto normal de la página, en el bloc de notas, está escrito en líneas tabuladas pero observe que éstas no se corresponden con las de la página cargada en el navegador porque los saltos de línea y las tabulaciones no son identificados en el lenguaje **HTML**. Aunque, si observa bien, los saltos de línea que aparecen en el archivo de código HTML se sustituyen por un espacio en la visualización de la página web en el navegador. ¡Compruébelo!

Si se quiere comenzar un nuevo párrafo separado por una línea, dicho párrafo deberá estar precedido por la etiqueta <**p**> y su correspondiente de cierre.

Si lo que se desea es insertar un salto de línea utilizaremos la etiqueta <**br**> que no tiene una correspondiente de cierre (De ahí que utilicemos la etiqueta de la siguiente manera <**br /**> que explicaremos más adelante.)

Al utilizar la etiqueta <**p**>, por defecto, cada párrafo está precedido y seguido del espacio correspondiente a un salto de línea para marcar la separación con el contenido anterior y posterior. Este espacio puede modificarse mediante las hojas de estilo **CSS**.

Si se repite la etiqueta <**p**>, una detrás de otra, no se insertan líneas en blanco. Podría utilizar para ello la etiqueta <**br /**> pero recomendamos, también para este caso, hacerlo mediante las hojas de estilo **CSS**.

***Es hora de prácticar. Realice la práctica 2.5.***

ATRIBUTOS DE ETIQUETA

En ocasiones es preciso completar una etiqueta con instrucciones más precisas. Para ello se utilizan los atributos de etiqueta. El atributo se inserta en la etiqueta, entre la palabra de instrucción y el signo “>” final.

La sintaxis completa de una etiqueta con un atributo es:

**<etiqueta atributo=”valor”> … </etiqueta>**

El atributo contiene siempre un valor, que se indica mediante el signo igual (=) seguido del valor escrito entre comillas. El respeto estricto a la sintaxis exige que no exista espacio antes o después del signo igual.

En nuestra primera página web hemos incluido un atributo en la etiqueta <**html**>:

**<html lang="es">**

para indicar que el idioma utilizado en nuestra página web es el español.

Es posible utilizar varios atributos simultáneos, separados por espacio, en una misma etiqueta:

**<etiqueta atributo1=”valor” atributo2=”valor” > … </etiqueta>**

USO CORRECTO DE LAS ETIQUETAS

Antes de meternos en más profundidades conviene apuntar algunas reglas simples que será preciso respetar a la hora de escribir etiquetas en **HTML5**:

* En **HTML5** las etiquetas no distinguen entre mayúsculas y minúsculas, así que se pueden escribir de las dos maneras. Hay desarrolladores que utilizan la escritura de las etiquetas en mayúscula para distinguirlas del resto del documento, pero el uso generalizado es escribirlas en minúsculas.
* Por regla general se exige que toda etiqueta abierta se cierre.
* Las etiquetas deben enlazarse correctamente. Cuando se aplican varias etiquetas a un elemento, el orden de estas es esencial. La primera etiqueta de cierre debe corresponder con la última etiqueta de apertura que no haya sido cerrada como se ve en el siguiente ejemplo:
  + Correcto: **<a><b><c> contenido <c><b><a>**
  + Incorrecto: **<a><b><c> contenido <a><c><b>**
* Los valores de los atributos deben escribirse siempre entre comillas.

SINTÁXIS XHTML

Cómo comentamos al principio de este curso las páginas escritas en **XHTML** se pueden ver también en los navegadores actuales ya que las etiquetas y atributos de **XHTML** y **HTML** siguen siendo las mismas, pero la sintaxis del código **XHTML** es más estricta.

Las diferencias más importantes entre **XHTML** y **HTML** son estas:

* En **XHTML**, todas las etiquetas deben escribirse en minúsculas.
* **XHTML** exige que todas las etiquetas de apertura lleven su correspondiente etiqueta de cierre. Todas las etiquetas **XHTML** deben tener cierre, aun en el caso de etiquetas especiales que técnicamente no necesitan etiquetas de apertura y cierre. De ahí que, en la etiqueta <**br**>, que hemos utilizado para insertar un salto de línea, hayamos utilizado una sintaxis especial de autocierre para hacer nuestra página web compatible con **XHTML**. Recuerde que una etiqueta que se cierra ella misma tiene este aspecto (con un espacio y la barra inclinada antes del signo ">"): <**br /**>
* **XHTML** exige que el anidamiento de etiquetas (su inclusión unas dentro de otras), siga unas reglas estrictas.

Puesto que **HTML5** admite etiquetas **XHTML**, como ya hemos comentado, en este curso, utilizaremos código ajustado a **XHTML** para asegurar la compatibilidad hacia atrás, ayudar a que nuestros diseños sean compatibles con los navegadores y dispositivos móviles actuales, y también porque así es más fácil la resolución de errores cuando aparecen.

INSERCIÓN DE SEPARADORES

Otra etiqueta utilizada en nuestra página web es <**hr /**> que no necesita etiqueta de cierre porque sólo sirve para separar el contenido, o indicar un cambio de tema, en una página **HTML**. Añade un separador horizontal tan ancho como la pantalla y embutido en el fondo (no abuse de los separadores en sus páginas porque, en exceso, dan un aspecto fraccionado a la pantalla).

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 2.***

CAPÍTULO 3 – EMPEZANDO A DAR FORMATO

VALIDACIÓN DEL CÓDIGO HTML

Como ya hemos comentado, W3C es una organización sin ánimo de lucro cuyo objetivo consiste en orientar la evolución de la Web. Ofrece directrices y reglas para especificaciones como **HTML** y **XHTML**. Es posible comprobar la validad del código de nuestras páginas web utilizando el servicio online gratuito de validación del W3C.

***Para comprobar como se valida el código realice ahora la práctica 3.1***

Observe que la validación nos dice si el código está bien, si faltan etiquetas o no están bien escritas, pero no si la página es bonita.

El proceso de validación puede ser útil para resolver errores, ya que nos facilita la detección de cualquier problema de sintaxis. Tener un código conforme es uno de los pasos que nos garantizará que nuestras páginas web se van a ver en el mayor número posible de navegadores y dispositivos.

La validación de páginas del W3C es el aspecto más tangible de un diseño web conforme a estándares, pero hay también otras ventajas importantes al crear páginas bien estructuradas:

* Menos código: el uso de **HTML** y **CSS** nos permite crear páginas similares con menos líneas de código, es decir, menos trabajo a la hora de desarrollarlas y más rápida visualización en el navegador.
* Mantenimiento más sencillo: tener menos código supone un mantenimiento más sencillo, y esto beneficia tanto al que escribe el código como a cualquier otra persona que después deba encargarse de su mantenimiento y revisión.
* Accesibilidad: los documentos creados de forma semántica, es decir, que utilizan la etiqueta **HTML** más adecuada a su función en cada momento, pueden facilitar la lectura y navegación a personas con discapacidad visual y el visitante de la web encuentra la información más fácilmente.
* Optimización para motores de búsqueda: Las páginas web que incluyen secciones claras e identificadas de forma lógica, tanto a nivel de código como de contenidos, facilitan la labor a los motores de búsqueda a la hora de indexarlas y clasificarlas.
* Compatibilidad: los sitios web que independizan la estructura con respecto al estilo se adaptan mucho más fácilmente a dispositivos móviles y formatos de pantalla variados. **CSS** nos permite aplicar hojas de estilo alternativas que optimizan la presentación en pantalla dependiendo del dispositivo utilizado.
* Demostración de su propia capacidad y profesionalidad a la hora de realizar un código de calidad, conforme a los estándares de **HTML5**.

NEGRITA Y CURSIVA

Los dos métodos fundamentales para resaltar texto son los caracteres en negrita y en cursiva. Para que una parte del texto aparezca en negrita hay que situar ese texto entre las etiquetas **<b>** y **</b>** (Bold).

Para que el texto aparezca en letra cursiva hay que situarlo entre las etiquetas **<i>** y **</i>** (Italic).

También disponemos de la etiqueta <**strong**> … <**/strong**> que escribe texto en negrita en los navegadores. Esta etiqueta se diferencia de <**b**> en que si bien el efecto visual es el mismo, la interpretación es muy distinta en los programas de sintetización de voz que leen las páginas web para personas con deficiencias visuales, consiguiendo una entonación de voz distinta o un nivel sonoro más elevado.

Por otro lado, tenemos que hablar también de la etiqueta <**em**> … <**/em**>, (énfasis), que también escribe el texto en cursiva en los navegadores pero se diferencia de la etiqueta <**i**> en que tiene por objeto poner de relieve el texto que la contiene.

***Veálo en funcionamiento estas etiquetas realizando la práctica 3.3***

SUBINDICES Y SUPERINDICES

En ocasiones es necesario escribir algunos caracteres como exponentes o subíndices, por ejemplo para fórmulas matemáticas.

La etiqueta <**sup**> … <**/sup**> escribe el contenido en exponente y la etiqueta <**sub**> … <**/sub**> en subíndice.

TEXTO TACHADO

La etiqueta <**del**> … </**del**>, (delete), permite marcar un elemento de texto como tachado. Un ejemplo podría ser la actualización de un precio en una web comercial.

TEXTO PREFORMATEADO

La etiqueta <**pre**> … <**/pre**> permite visualizar el texto tal y como se ha codificado en el editor de texto. Los espacios, tabulaciones y saltos de línea se respetan tal cual en la pantalla y el texto se visualizará con un tipo de letra de paso fijo, normalmente Courier.

Como podrá haber observado la etiqueta <**pre**> se puede utilizar para mostrar texto con un formato inusual, o algún tipo de código informático.

DIRECCIÓN DEL TEXTO

Ahora una curiosidad. Algunos sistemas de escritura, como el árabe, se escriben de derecha a izquierda. Para ello podemos utilizar la etiqueta <**bdo**> … <**/bdo**>.

Esta etiqueta tiene dos atributos:

* dir=”ltr” (left to rigth) para el sentido de escritura de izquierda a derecha.
* dir=”rtl” (right to left) para el sentido de derecha a izquierda.

CITAS

La etiqueta <**blockquote**> y <**/blockquote**> se utiliza para destacar una cita textual dentro de un texto general. ¡Veámoslo!

Observe que la cita se visualiza con un salto de línea antes y otro después, igual que con los párrafos. El texto que contienen estas etiquetas aparece con márgenes a ambos lados de manera que destaca sobre el resto de la página.

Existe también la etiqueta <**q**> … </**q**> para las citas pero esta visualiza el texto entre comillas, no muestra espacio antes ni después, ni tampoco los márgenes a ambos lados.

***Es el momento de realizar la práctica 3.4***

COMENTARIOS NO VISIBLES

Puede que alguna vez necesite poner un comentario o nota en su programación de código **HTML** y no quiere que se visualice en su página Web. Para ello utilizaremos las etiqueta **<!--** que se cierra con **-- >**.

Por ejemplo: <!-- Este texto no se verá en mi página Web -->

Todo lo que escribamos entre estas etiquetas no se visualizará en la pantalla del navegador pero si en el archivo de código fuente. Los comentarios pueden ocupar varias líneas y pueden ir en cualquier parte de la página web, tanto en el <**head**> como en el <**body**>.

El uso de comentarios que documenten el código, es una de las buenas prácticas de todo buen diseñador de páginas web.

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 3.***

CAPÍTULO 4 – EMPEZANDO CON HOJAS

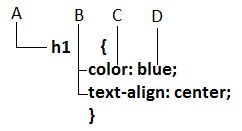
DE ESTILO CSS

Como ya hemos comentado, con **HTML5**, se establece una separación entre el contenido de la web y su formato o presentación. **HTML** se encarga solamente de la estructura del documento, y las hojas de estilo **CSS** se encargan de todo el aspecto relacionado con el formato visual.

En el capítulo anterior hemos utilizado algunas etiquetas para dar formato como cursiva o negrita. Efectivamente las hemos utilizado para iniciarnos en el aprendizaje del **HTML**, pero lo normal será utilizar hojas de estilo **CSS** y no esas etiquetas.

Las hojas de estilo pues, abarcan todo aquello relacionado con la presentación del documento **HTML** como el color, la alineación, el tamaño, los bordes, los fondos, etc, etc… Veamos qué son y cómo utilizarlas.

LA SINTAXIS DE CSS

 En la siguiente imagen puede observar una regla de **CSS**:

En ella podemos distinguir las siguientes partes:

A.- **Selector**. Se trata de los elementos o etiquetas del código **HTML** que se verán afectados por las órdenes de las hojas de estilo. En este caso todos los títulos de nivel 1 (**h1**). Tenga en cuenta que los selectores en una regla **CSS** se escriben simplemente con el nombre de la etiqueta sin los signos **<** y **>**.

Es posible aplicar el mismo efecto de estilo a distintos selectores. En este caso, cada selector implicado se escribe separado por una coma. Por ejemplo, la siguiente regla de estilo afectaría a los selectores h1, h2 y h3:

**h1, h2, h3 {**

**Color: blue;**

**Text-aling: center;**

**}**

B.- **Declaración**. Se refiere al conjunto propiedad-valor. Por ejemplo la declaración “**color: blue**” podríamos interpretarla como “poner en color azul”. En esta regla hay dos declaraciones. Es decir, que una regla de estilo puede incluir varias declaraciones para un mismo selector. Todas las declaraciones terminan con un punto y coma obligatoriamente.

C.- **Propiedad**. Indica el elemento de presentación que se ve implicado; por ejemplo “**color**” para el color, “**text-align**” para la alineación… La propiedad está separada de su valor mediante dos puntos. Para facilitar la legibilidad del código es buena costumbre dejar un espacio entre los dos puntos y el valor.

D.- **Valor**. Especifica una palabra clave, un porcentaje o un tamaño dependiendo de la propiedad a la que se refiera.

Toda la regla de estilo está contenida entre dos llaves **{** y **}**. Si nos olvidamos cualquiera de las dos llaves, la regla de estilo no será tenida en cuenta por el navegador.

Las hojas de estilo no distinguen entre mayúsculas o minúsculas aunque lo habitual es utilizar siempre minúsculas.

Concretando, la regla de estilo de la imagen le dice al navegador que todos los títulos de nivel 1 de la página web se escriban en color azul y con el texto centrado.

INCORPORACIÓN DE LA HOJA DE ESTILO A LA WEB

Existen 4 maneras diferentes de utilizar las hojas de estilo CSS en los archivos HTML:

* **Integradas en una etiqueta HTML.** En este caso se indicaría la declaración como si fuera un atributo más de la etiqueta. Por ejemplo:

**<h1 style=”color: blue;”>Título de nivel 1</h1>**

Aunque puede resultar útil en alguna ocasión, y en virtud del principio de separación entre contenido y presentación, lo habitual es evitar esta forma de utilizar los estilos.

* **Internas al documento HTML.** Las reglas de estilo se sitúan agrupadas en el encabezado del documento dentro las etiquetas <**head**> y <**/head**>. La sintaxis es la siguiente:

**<style type=”text/css”>**

**Las diferentes reglas de estilo…**

**</style>**

* **Externas al documento HTML.** La estudiaremos más adelante.
* **Importadas.** La estudiaremos más adelante.

***Aprenda a aplicar hojas de estilo realizando la práctica 4.1***

TIPOS DE SELECTORES CSS

Hasta ahora, en nuestras reglas de estilo, hemos utilizado selectores de etiqueta o selectores de tipo.

Se habrá dado cuenta que utilizando selectores de tipo, estamos limitados, puesto que se aplica el estilo a todas las apariciones de una etiqueta en el documento, pero no podemos aplicar un estilo a alguna cosa que no coincide exactamente con una etiqueta. Por ejemplo, a un trozo de texto, como aplicamos en su momento la negrita y cursiva. O asignar un color a una sola palabra. O asignar un formato sólo a una de las etiquetas y no a TODAS las que aparecen en el documento.

Para ello utilizaremos otros tipos de selectores:

A.- **Selectores de clase**. La declaración de clase se realiza escribiendo un punto seguido del nombre que queremos dar a la clase y seguido de la declaración de estilo escrita entre llaves, siguiendo este formato:

**.nombre\_de\_la\_clase { declaraciones de estilo }**

Por ejemplo:

**.azul {color: blue;}**

Para indicar que se aplique la clase se utiliza como si fuera un atributo de la etiqueta a la que queremos aplicar el estilo, de la siguiente manera:

**<etiqueta class=”nombre\_de\_la\_clase”>**

Hay que tener en cuenta que:

* El punto del selector de clase no se utiliza a la hora de llamar a la clase **class.**
* Una misma clase se puede invocar varias veces en un mismo documento **HTML**.
* Un nombre de clase puede estar formado por letras, cifras, el guión y el carácter de subrayado. El primer carácter no puede ser un número, un guión o un carácter de subrayado.

***Vamos a aplicarlo realizando la práctica 4.2***

B.- **Selectores de Identificador**. El selector **id**, también llamado identificador **id**, permite aplicar una hoja de estilo, de la misma forma que el selector **class**, aunque sólo se podrá utilizar una única vez en el documento **HTML**. La declaración de identificador se realiza escribiendo una almohadilla (**#**) seguida del nombre que queremos dar al identificador **id** y seguido de la declaración de estilo escrita entre llaves. Siguiendo este formato:

**#nombre\_del\_identificador { declaraciones de estilo }**

Por ejemplo:

**#azul {color: blue;}**

Esta selección sólo se podrá utilizar una única vez en el documento.

Para indicar que se aplique el identificador, se utiliza como si fuera un atributo de la etiqueta a la que queremos aplicar el estilo, de la siguiente manera:

**<etiqueta id=”nombre\_del\_identificador”>**

Hay que tener en cuenta que:

* Un identificador **id** no puede figurar más de una vez en el documento **HTML** en caso contrario, ¡el resto del código será incorrecto!
* Un documento puede contener varios identificadores **id** con nombres diferentes aunque sólo pueden utilizarse una única vez.
* Habitualmente, el identificador **id**, se utiliza para elementos **HTML** generales o estructurales que aprenderemos más adelante como <**section**>, <**article**>, etc, etc.
* Es posible mezclar declaraciones de estilo **class** e **id**.
* La almohadilla (**#**) del selector de identificador no se utiliza a la hora de llamar al **id**.
* Un nombre de identificador puede estar formado por letras, cifras, el guión y el carácter de subrayado. El primer carácter no puede ser un número, un guión o un carácter de subrayado.
* **Id** es más apropiado que **class** para usarlo en referencias con **JavaScript**.

Por último debemos de mencionar al asterisco (**\***) como **selector universal** que podemos utilizar para representar a cualquier tipo de elemento **HTML**.

COMENTARIOS EN CSS

Al igual que en **HTML**, en una hoja de estilo **CSS**, podemos y debemos insertar comentarios que documenten el código escrito para facilitar posteriores modificaciones y/o cambios.

En una hoja de estilo **CSS**, un comentario comienza con los caracteres de apertura barra oblicua y asterisco (**/\***) y termina con los caracteres asterisco y barra oblicua (**\*/**), siguiendo la siguiente forma:

**/\* Así se escribe un comentario en CSS \*/**

UNIDADES DE MEDIDA EN HOJAS DE ESTILO CSS

Las hojas de estilo **CSS** permiten utilizar numerosas unidades de medida de longitud distinguiendo entre:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UNIDADES ABSOLUTAS** | | | **UNIDADES RELATIVAS** | | |
| Especifican un tamaño fijo y están ancladas a alguna medida física. | | | Especifican un tamaño en relación a otra medida. | | |
| **UNIDAD** | **NOMBRE** | **EJEMPLO** | **UNIDAD** | **DESCRIPCIÓN** | **EJEMPLO** |
| pt | Punto. Utilizado en imprenta. | 48pt | em | Unidad relativa que se basa en el tamaño del tipo de letra del elemento padre. | 1.8em |
| pc | Pica | 4.5pc | rem | Unidad relativa que se basa en el tamaño del tipo de letra por defecto de la página. | 1.8rem |
| mm | Milímetro | 60mm | ex | Unidad relativa a la altura de la letra “x” minúscula en el elemento seleccionado. | 1.3ex |
| cm | Centímetro | 6cm | % | Porcentaje. También se basa en el tamaño de letra del elemento padre. | 80% |
| in | Pulgada (inch) | 0.1in | vw | Unidad relativa igual al 1% de la anchura de la ventana del navegador. | 8vw |
|  |  |  | vh | Unidad relativa igual al 1% de la altura de la ventana del navegador. | 3.0vh |
|  |  |  | vmin | Unidad relativa igual a la más pequeña de “vw” o “vh”. | 2vmin |
|  |  |  | vmax | Unidad relativa igual a la más grande de “vw” o “vh”. | 2vmax |
| px | Se trata de una medida absoluta. Un píxel es la parte más pequeña de una imagen. Algunos autores la consideran una medida relativa debido a que el tamaño del piexel depende de la resolución de pantalla. De hecho, hasta la especificación CSS2.1, se considerá una medida relativa. | | | | 220px |

Profundizaremos en algunas de estas unidades de medida a lo largo del curso.

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 4.***

CAPÍTULO 5 - LISTAS Y ENLACES EN HTML

CREACION DE LISTAS

Las listas son una forma muy eficaz de estructurar muchos contenidos. Podemos distinguir 4 tipos de listas:

A.- **Listas no numeradas o no ordenadas**. Se trata de una lista en la que cambiando el orden de los diferentes elementos no cambia el significado del documento. Los diferentes navegadores presentan los elementos de la lista con viñetas determinadas por ellos.

Las listas no numeradas se indican con las etiquetas <**ul**> … <**/ul**>. Cada elemento interior de la lista se indica con las etiquetas <**li**> … <**/li**>. La estructura de una lista no numerada es:

**<ul>**

**<li>Primer elemento de la lista</li>**

**<li>Segundo elemento de la lista</li>**

**<li>Tercer elemento de la lista</li>**

**<li>Cuarto elemento de la lista</li>**

**</ul>**

***Veamos cómo aplicarlas realizando la práctica 5.1***

B.- **Listas ordenadas**. Si la lista esta ordenada, es decir que lleva números por delante, habrá que poner <**ol**> antes de escribir la lista y <**/ol**> cuando se termine, y seguiremos poniendo <**li**> al empezar cada línea de la lista, así:

**<ol>**

**<li> primer elemento de la lista<\li>**

**<li> segundo elemento de la lista<\li>**

**<li> etc. </li>**

**</ol>**

Hay que tener en cuenta que:

* El atributo **start** permite comenzar la numeración en una cifra distinta al 1.
* El atributo **reversed** permite realizar una numeración descendente pero, por el momento, no está implementado en ningún navegador.
* El atributo **Type** permite remplazar las cifras por letras o números romanos aunque para eso lo lógico es utilizar las hojas de estilo CSS. La sintaxis es:

**<ol type=”1|a|A|i|I”>**

Siendo: **1** para números decimales (1, 2, 3, 4, …). Es lo que aparece por defecto.

**a** para listas ordenadas alfabéticamente en minúsculas (a, b, c, d, …).

**A** para listas ordenadas alfabéticamente en mayúsculas (A, B, C, …).

**i** para listas con números romanos en minúsculas (i, ii, iii, iv, …).

**I** para listas con números romanos en mayúsculas (I, II, III, IV…).

***Veamos cómo aplicar las listas ordenadas realizando la práctica 5.2***

C.- **Listas anidadas**. Podemos incluir listas dentro de otras listas con distintos niveles. Esto es lo que queríamos hacer en el caso de los hoteles de Benidorm. Por un lado queríamos una lista de nº de estrellas hoteleras y por otro una lista de hoteles para cada categoría de estrellas.

***¡Veamos cómo quedaría realizando la práctica 5.3!***

D.- **Listas de Definición**. Se trata de la típica lista de glosarios que hay al final de muchos libros. Esta lista presenta una lista de términos cada uno seguido de su definición o descripción. En este caso utilizaremos las etiquetas, <**dl**> antes de empezar la lista y <**/dl**> cuando la termine. Además para definir el elemento de la lista usaremos las etiquetas <**dt**> … <**/dt**>, y las etiquetas <**dd**> … <**/dd**>, para describir el elemento de la lista.

***Veamos un ejemplo realizando la práctica 5.4.***

LOS ENLACES. EL HIPERTEXTO.

Los enlaces son la esencia del lenguaje **HTML** y la web. Y es lo que le confiere la riqueza que se ha denominado: hipertexto. La etiqueta para los enlaces es <a> … </a>, cuya sintaxis básica es:

**<a href=”**destino del enlace**”>** texto del enlace **</a>**

El destino del enlace puede ser:

* Un lugar de la página en curso.
* Otra página del sitio Web.
* Algún lugar de otra página del sitio web.
* Una página de otro sitio existente en la web.
* Una dirección de correo electrónico.
* Un archivo para descargar.

Veamos cada uno de estos tipos de enlaces más detenidamente.

ENLACES DENTRO DE LA MISMA PÁGINA

Es normal ver en las páginas muy extensas un pequeño índice donde al hacer clic en una determinada opción lleva al usuario a una posición concreta o más avanzada de la página para ahorrar tiempo. Para ello se utiliza la técnica de las anclas o puntos de fijación.

Este enlace al interior de la página se realiza en dos tiempos:

La declaración del ancla: Sirve para definir un lugar en la página para poder hacer un enlace hacia ella. Por su puesto, el nombre del ancla no se visualiza en el navegador. La sintaxis de la declaración del ancla es:

**<a id=”**etiqueta o nombre del ancla**”> </a>**

El enlace hacia el ancla. Para crear el enlace, basta con escribir en el atributo **href** el nombre del ancla directamente precedido del signo almohadilla (**#**), de la siguiente forma:

**<a href=”#**nombre de etiqueta donde saltará**”>** mensaje que se desee **</a>**

***Realicemos un ejemplo con la práctica 5.5***

Observe que los enlaces se visualizan, por defecto, subrayados y en color azul para el enlace no visitado, en color púrpura, para el enlace visitado y en color rojo para el enlace activo.

El subrayado es una convención que se utiliza desde los comienzos del **HTML**. Para no desconcertar al navegante poco experimentado, seguiremos con esta norma, pero desde el punto de vista estético, una web que contenga muchos enlaces puede resultar fea. Veremos más adelante como quitar el subrayado usando hojas de estilo.

ENLACES CON OTRA PÁGINA DE NUESTRO SITIO WEB

Lo habitual es que un sitio web esté compuesto por varias páginas Web, una principal y otras secundarias que a su vez pueden estar relacionadas entre sí. En nuestro caso podemos considerar que estamos realizando un sitio web dedicado al turismo en España, con varias páginas diferentes y cuya página web principal es “**turismo.html**”.

Si el sito web es muy grande, puede suceder que, los diferentes archivos, se sitúen en varias carpetas para mantener el sitio bien estructurado. De ser así, tendríamos que tenerlo en cuenta para indicar correctamente la ruta donde se encuentra el archivo al que queremos enlazar.

Para enlazar nuestra página principal con otra situada en la misma carpeta de nuestro sitio web utilizaremos el código:

**<a href=”**nombre de la página**.**html**”>** mensaje que desee **</a>**

Siendo “**nombre de la página.html**” el nombre con el que se grabó el archivo de código de la página a donde deseamos ir.

Si la página con la que queremos enlazar no se encuentra en la misma carpeta habría que indicarle la ruta donde se encuentra de la siguiente manera:

**<a href=”**ruta decarpetas/nombre de la página**.**html**”>** mensaje que desee **</a>**

A la hora de realizar enlaces a otras páginas web es importante respetar las mayúsculas y minúsculas en los nombres de los archivos y carpetas así como tener en cuenta los espacios para buscar la compatibilidad con otros sistemas operativos diferentes a Windows.

ENLACE A UN LUGAR CONCRETO DE OTRA PÁGINA DEL SITIO WEB

También es posible realizar un enlace hacía un ancla situada en otra página. La sintaxis sería:

**<a href=”**ruta de carpetas**/**nombre de la página**.**html**#**ancla**”>**mensaje que desee**</a>**

Recuerde que previamente en esa página donde deseemos ir tendríamos que haber puesto la orden correspondiente al ancla de destino:

**<a id=”**ancla**”> </a>**

***Veamos el funcionamiento realizando la práctica 5.6:***

ENLACES A PÁGINAS DE CUALQUIER SITIO WEB

Si lo que queremos es enlazar con páginas Web de otros servidores de cualquier parte del mundo, lo único que deseamos saber es su dirección completa. La sintaxis:

**<a href=”**dirección completa de la página**”>** mensaje que desee **</a>**

Nótese que hay que incluir la dirección completa, incluido el texto del protocolo http://.

Por ejemplo si queremos enlazar con la web de nuestro centro la orden completa sería:

**<a href=”**http://www.academiafrayluis.es**”>Visite nuestra web** **</a>**

ENLACES A UNA DIRECCIÓN E-MAIL.

Podemos permitir a los visitantes de nuestro sitio web que contacten con nosotros a través del correo electrónico. Para ello realizaremos un enlace a una dirección de correo siguiendo el siguiente formato:

**<a href=”mailto:**dirección correo electrónico**”>** mensaje que desee **</a>**

***Veamos un ejemplo realizando la práctica 5.7***

Como ha podido comprobar, la activación del enlace hacia una dirección mail, abre una ventana de la aplicación de correo electrónico configurada por defecto en el puesto del visitante.

ENLACES PARA DESCARGAR UN ARCHIVO

Para descargar un archivo utilizamos la misma etiqueta <**a**> indicando la ruta y el nombre del archivo que se quiere descargar.

***Veámoslo con un ejemplo realizando la práctica 5.8.***

Si en el ordenador del visitante no hay ninguna aplicación instalada para leer archivo en formato .pdf, el navegador abre una ventana que invita a descargar el archivo.

ATRIBUTOS DE LA ETIQUETA DE ENLACES

La etiqueta <**a**> tiene varios atributos:

A.- **Href**. Indica la dirección del destino del enlace. Ya lo hemos estudiado ampliamente.

B.- **Hrflang**. Indica el idioma del documento destino si es diferente al del documento origen. Solo se usa si el atributo **href** está presente.

C.- **Ping**. Indica una lista de direcciones url que reciben una notificación cuando el usuario sigue el enlace. Es muy útil para realizar estadísticas de un sitio web.

D.- **Media**. Especifica que el contenido de la dirección url de destino está diseñado para dispositivos especiales, como el iPhone, el braille o sintetizadores de voz. Solo se usa si el atributo **href** está presente.

E.- **Rel**. Especifica la relación entre el documento de origen y el documento destino del enlace. Solo se usa si el atributo **href** está presente.

F. **Target**. Especifica donde abrir, en qué ventana, los documentos vinculados. Solo se usa si el atributo **href** está presente. Tiene las siguientes posibilidades:

* **Target=”\_selft”**. La página de destino se abre en la misma ventana que la página de origen del enlace. Es lo que se hace por defecto.
* **Target=”\_top”**. La página de destino del enlace se abre en la misma ventana pero ocupará la totalidad de la ventana mostrada. (Con lo que sabemos hasta ahora de **HTML**, el resultado de este valor es exactamente el mismo que para la opción **\_selft**).
* **Target=”\_blank”**. La página de destino del enlace se abre en una nueva instancia o en una nueva pestaña del navegador.

G.- **Type**. Indica al navegador el tipo MIME del archivo del enlace, por ejemplo si se trata de un archivo de sonido o una imagen. Solo se usa si el atributo **href** está presente.

DESCRIPCIÓN DEL ENLACE

Existen una serie de atributos, llamados globales, que se pueden utilizar prácticamente en todas las etiquetas de **HTML**. Uno de ellos es el atributo title que permite mostrar un texto explicativo cuando se pasa el puntero del ratón por encima de algún elemento de la web. Es una opción muy útil para las facilidades a los internautas invidentes.

La sintaxis es:

**<a href=”t-deport.htm” title=”Ir a la web de Turismo Deportivo”>Turismo Deportivo</a>**

***Veamos un ejemplo realizando la práctica 5.9.***

CAPÍTULO 6 – TIPOGRAFÍA, TEXTO

Y LISTAS EN CSS

USO DE FUENTES Y TIPOS DE LETRA EN EL DISEÑO WEB

Al diseñar páginas web, podemos dar formato al texto de una forma parecida a como se hace en un tratamiento de textos o un programa de diseño en cualquier ordenador. El problema es que cuando indicamos que se use una fuente, (tipo de letra), concreta, esa fuente ha de estar instalada en el ordenador o el dispositivo del usuario que visita nuestra página. Si el ordenador o dispositivo no dispone de esa fuente, el navegador la sustituye por otra.

Puesto que es imposible saber qué fuentes están instaladas en los ordenadores de los usuarios, y debido a la capacidad que tienen los navegadores para sustituir unos tipos de letra por otros, es posible que los resultados obtenidos en ciertos equipos no se correspondan con el diseño que hemos previsto.

Una opción es utilizar fuentes que sabemos que podemos encontrar en la mayoría de los ordenadores, pero son tan pocas las fuentes de las cuales podemos tener la certeza de que se encuentran en prácticamente todos los ordenadores, que los buenos diseñadores se sienten muy limitados.

Teniendo en cuenta las plataformas Mac y Windows y los ordenadores más antiguos, las fuentes que podemos utilizar con más confianza en la Web son:

|  |  |
| --- | --- |
| * Arial | * Verdana |
| * Georgia | * Times New Roman |
| * Courier | * Trebuchet |
| * Lucida | * Tahoma |
| * Impact |  |

Las fuentes Courier e Impact se usan poco porque, aunque estén disponibles de forma general, su apariencia es muy peculiar y limita mucho su uso de forma habitual.

Sin solucionar totalmente el problema, las Hojas de Estilo **CSS**, aportan mayor diversidad en la elección de la tipografía mediante el uso de una "pila" de fuentes, ("font stack" en inglés). Se trata de indicar al navegador una lista de varios tipos de letra, alternativos entre si, que puede utilizar para mostrar en pantalla.

La sintaxis del estilo es:

**font-family: nombre de la fuente o familia de la fuente;**

Por ejemplo:

**font-family: Arial, Helvética, Georgia, Sans-serif;**

En este caso, hemos indicado varias fuentes separadas por comas, para que el navegador utilice la primera fuente, Arial, si está instalada en el sistema del usuario. En caso contrario se pasa a la siguiente de la lista y así sucesivamente.

La última fuente es un tipo genérico, una familia de fuentes, la familia de fuentes Sans-serif, que indica al navegador que emplee cualquier fuente Sans-serif que encuentre.

Si no se encuentra en el sistema del usuario, ninguna de las fuentes indicadas, el navegador utilizará la fuente por defecto que es lo que pasa en los ejercicios que hemos realizado hasta ahora en este curso ya que no hemos indicado aún ninguna fuente.

Por defecto, los navegadores más habituales emplean la fuente Times New Roman, de 14 píxeles y color negro.

Si el nombre de la fuente incluye espacios, como “Courier New”, se debe escribir entre comillas.

En el nombre de las fuentes no es necesario distinguir entre mayúsculas y minúsculas.

Las familias de fuentes más habituales son:

* **Serif**. Tienen terminaciones al final de sus trazos y un espaciamiento proporcional entre las letras. Ej: Times New Roman, Bodoni, Georgia, Garamond…
* **Sans-Serif**. Tienen terminaciones planas, sin ensanchamiento y un espaciamiento proporcional entre las letras: Ej: Arial, Verdana, Helvética, Trebuchet…
* **Cursive**. Familia de fuentes que se muestran parecido a una escritura manuscrita. Ej: Script, Adobe Poética…
* **Fantasy**. Fuentes de fantasía o decorativas. Ej: Wingdings…
* **Monospace**. Todas las letras tienen las mismas dimensiones como si se tratara de una máquina de escribir manual. Habitualmente se utilizan para programar código informático por su facilidad de lectura en este ámbito. Ej: Courier, Courier New…

Como ya hemos comentado, al indicar una familia de fuentes, el navegador escogerá por sí mismo la fuente disponible en el ordenador del usuario que se adapte a los criterios del nombre genérico. Por ello es recomendable especificar un nombre de fuente genérico tras realizar una enumeración de nombres de fuente para que la visualización respete el tipo de fuente deseado por encima de la fuente por defecto del navegador.

***Pero… ¡pasemos a la acción realizando la práctica 6.1!***

ITALICA O CURSIVA

Habíamos utilizado la etiqueta <**i**> para resaltar algún texto en cursiva. Y habíamos comentado que aunque existía dicha etiqueta era mejor utilizar las hojas de estilo. Esta es la sintaxis para escribir texto en cursiva:

**font-style: italic;**

Otros valores para esta propiedad son:

* **oblique:** inclina hacia la derecha los caracteres. El resultado es parecido a la letra cursiva.
* **Normal:** retoma la escritura normal

NEGRITA

Esta propiedad permite escribir texto en negrita con muchas más posibilidades, aunque poco perceptibles, que las de la etiqueta <**b**>. La sintaxis es:

**font-weight: bold;**

Otros valores para esta propiedad son:

* **Bolder:** acentúa la negrita respecto al elemento padre.
* **Lighter:** disminuye la negrita respecto al elemento padre.
* **Un valor numérico entre 100 y 900.** Los valores disponibles son: 100, 200, 300, 400, 500, 600, 700, 800 y 900. El 500 se corresponde con el valor de letra normal. El 700 con la negrita y el 900 con la negrita más llamativa.
* **Normal:** retoma la escritura normal.

VERSALES

Esta propiedad permite visualizar el texto en mayúsculas versales, (pequeñas mayúsculas). La sintaxis es:

**Font-variant: small-caps**

Otro valor para esta propiedad es:

* **Normal:** retoma la escritura normal.

COMO NOMBRAR LOS COLORES EN HOJAS DE ESTILO CSS

En temas anteriores hemos utilizado los nombres de colores para utilizarlos en un estilo. Pero los colores en las hojas de estilo **CSS** se pueden indicar de varias maneras:

La notación hexadecimal clásica, como #FFCC00, es la notación más utilizada. Con esta notación podemos reproducir prácticamente cualquier color. En la notación hexadecimal, el código se divide en tres pares de dígitos de tal manera que si tenemos un código como #AABBCC:

* AA= Es un número en hexadecimal de dos dígitos donde se le indica cuanta cantidad de rojo se desea que tenga el color.

(Un número Hexadecimal tiene 16 valores posibles que son: 0 1 2 3 4 5 6 7 8 9 A B C D E F, es decir el 0 será el valor mínimo y la F será su valor máximo, pero al poder poner en éste caso dos dígitos hay más combinaciones donde el valor mínimo será el 00 y el máximo será FF).

* BB= Lo mismo que el anterior pero para la cantidad de verde.
* CC= Igual pero para el color azul.

Veamos cómo haríamos los colores más primarios utilizando la notación hexadecimal:

* #FF0000: Sería el color rojo ya que BB y CC están a 0 mientras que AA está a su valor máximo: FF.
* #00FF00: Sería el color verde por estar en este caso AA=00, BB=FF (su valor máximo) y CC=0.
* #0000FF: Sería el color Azul.

Como se puede suponer, con la combinación de todos los valores de AA, BB y CC podemos generar todos los colores que usted imagine. Para que tenga como referencia:

* #FFFFFF dará el color blanco.
* #000000 dará el color negro.
* #FFFF00 dará el color amarillo.

Si desea hacer tonalidades oscuras deberá de reducir el número de color a oscurecer dejando los otros dos sin tocar. Veamos, si tiene un verde #00FF00 y lo quiere poner más oscuro reduzca su valor por ejemplo a #009900 y observe, si aún lo quiere más oscuro pruebe a poner #004400.

Si lo que quiere es que ese color sea más suave o difuminado, (que no más claro), deberá probar a poner los otros dos colores con números más altos, (claros), que el que tengan en ese momento, pero con un número igual.

Si usamos la combinación #FF7070 obtendremos un rosa ya que como podemos observar el BB y el CC han aumentado su valor a 70 por lo que el rojo puro FF se va suavizando.

Ya sólo es cuestión de tiempo para buscar el color que desee y probar hasta conseguirlo, con un poco de práctica podrá fácilmente usar unos colores lo suficientemente atractivos para sus páginas WEB.

Aunque habitualmente utilizaremos la notación hexadecimal clásica, otras formas de asignar los colores en **CSS** son:

* Las palabras clave que ya conoce como blue, Green, yellow, etc, etc.
* La notación hexadecimal abreviada como #fd3. Esta notación permite ahorrar algunos caracteres. Cada cifra está implícitamente duplicada de tal manera que #fd3 se corresponde con la notación clásica #ffdd33. Como es lógico con esta notación abreviada nunca se podría indicar un color como #cfe4f5.
* La notación decimal, por ejemplo, rgb(0,0,255). El código rgb se codificaría con un número comprendido entre 0 y 255. Es el equivalente decimal a la notación hexadecimal.
* La notación en porcentaje, por ejemplo; rgb(25%, 50%, 0%). El valor 0% significa la ausencia del color y 100% su valor máximo.
* La notación RGBa de la siguiente manera: rgba(0,0,0,0). En este caso las tres primeras cantidades se corresponden con la notación RGB vista anteriormente y la cuarta cantidad, el último valor indica el grado de opacidad o de transparencia, definido entre 0 y 1.
* La notación HSL (Tonalidad, Saturación, Luminosidad). La notación HSL consiste en tres valores. El primero se expresa en grados, de 0º a 359º. Se corresponde con un color de la rueda cromática: rojo (0º), amarillo (60º), verde (120º), cian (180º), azul (240º) y magenta (300º). El segundo y tercer valor se expresan en porcentaje y describen respectivamente la saturación y la luminosidad. Por ejemplo: hsl(0, 100%, 50%) sería la forma de indicar el color rojo.
* La notación HSLa, que agrega un valor comprendido entre 0 y 1 para la transparencia o la opacidad. Por ejemplo, hsl(0, 100%, 50%, 0.5) indicaría un color rojo semitransparente.

***Pero vayamos a verlo en la práctica realizando la práctica 6.3.***

TAMAÑO DE LA LETRA

La propiedad **CSS** que controla el tamaño del texto es “**font-size**”.

Cómo hemos visto en páginas anteriores podemos indicar el valor de la propiedad “**font-size**” con varias unidades de medida y formas diferentes:

* **Tamaño absoluto**: se trata de una serie de palabras clave que indican tamaños predefinidos de letra. Los tamaños nominados escalan de acuerdo con las preferencias del usuario con respecto a la letra. Los valores posibles son: xx-small, x-small, small, medium, large, x-large y xx-large.
* **Longitud**: Es un número seguido de un indicador del sistema de medida (cm, mm, in, pt, o pc) o bien de unidades relativas (em, ex, o px).
* **Porcentaje**: Un entero seguido del signo de porcentaje (%). El valor es un tanto por ciento del tamaño de letra del objeto padre.
* **Tamaño relativo**: una serie de palabras clave que se interpretan como relativas al tamaño de letra del objeto padre. Los valores posibles son larger y smaller.

La elección de la unidad de medida para el tamaño de la letra en una página web es importante y no es tan fácil como parece. La principal dificultad en este sentido tiene que ver con la resolución de la pantalla y las distintas maneras en que, a lo largo del tiempo, los navegadores han resuelto la presentación de los textos. También tenemos que tener en cuenta los usuarios que navegan con teléfonos y dispositivos móviles.

Podemos indicar el tamaño de la letra en píxeles y puntos de la siguiente manera:

**p {**

**font-size: 12px;**

**}**

o

**p {**

**font-size: 12pt;**

**}**

Sin embargo, aunque nos parezca práctico al estar acostumbrados a su uso, por ejemplo en Word, no se recomienda utilizar estas medidas en el diseño de páginas web ya que:

* los puntos son una unidad de medida absoluta pensadas para la imprenta y que no tiene una equivalencia exacta en la pantalla.
* Los píxeles son la unidad de medida que se suele utilizar en los programas de diseño gráfico por ordenador. Los tamaños y resoluciones de pantalla se expresan en píxeles pero todos los navegadores ofrecen la posibilidad de cambiar el tamaño del texto lo que deja sin control al diseñador web.

Una forma de resolver el problema del cambio de tamaño de los textos que causa el uso de píxeles consiste en aplicar otras unidades de medida. Estableceremos un cambio de tamaño fiable combinando el porcentaje y la unidad "**em**".

***Veámoslo en la práctica realizando la práctica 6.4.***

SUBRAYADOS

Esta propiedad permite subrayar de diversas maneras el texto. Su sintaxis es:

**text-decoration: underline;**

y los valores posibles son:

* **underline**: subraya el texto por debajo.
* **overline**: subraya el texto por encima.
* **line-through**: traza una línea sobre el texto. Equivale al tachado.
* **none**: quita cualquier subrayado indicado anteriormente.

***Practiquemos con un ejemplo realizando la práctica 6.5.***

Sobre esta propiedad también debemos tener en cuenta:

* Se pueden definir varios valores, separados por un espacio, en una misma declaración. Ej: **text-decoration: underline overline**;
* El valor “**line-through**” es particularmente útil en webs comerciales para destacar una revisión de precios. Recuerde que la etiqueta <**del**> tiene el mismo efecto.
* El principal uso que podemos dar al valor “**none**” es aplicarlo a la etiqueta de enlace <**a**> para suprimir el subrayado que aparece por defecto.

TRANSFORMACIÓN

Esta propiedad muestra un texto en mayúsculas o minúsculas, independientemente de cómo esté escrito en el código fuente. Su sintaxis es:

**text-transform: capitalize;**

y los valores posibles son:

* **capitalize**: escribe la primera letra de cada palabra en mayúsculas.
* **uppercase**: escribe todas las letras en mayúsculas.
* **lowercase**: escribe todas las letras en minúsculas.
* **none**: deja las letras sin modificación alguna.

Esta propiedad es muy útil para dar formato a los datos destinados a una base de datos.

INDENTACIÓN

Esta propiedad realiza una sangría de primera línea. Su sintaxis es:

**text-indent: valor preciso (20px) o valor relativo (20%);**

ESPACIO ENTRE LETRAS

Esta propiedad varía el espacio entre caracteres. Si no se abusa de su uso, esta propiedad permite mejorar la legibilidad del texto. Su sintaxis es:

**letter-spacing: valor de longitud (8px) / normal;**

A tener en cuenta:

* Los valores relativos, por ejemplo un porcentaje, no están admitidos en esta propiedad.
* Un valor positivo agrega espacio entre las letras. Un valor negativo disminuye el espacio entre las letras.
* La alineación a ambos márgenes del texto exige al navegador separar el espacio entre letras. Por lógica, si indicamos un valor “**letter-spacing**” de **0,** impediremos la alineación del texto.

ESPACIO ENTRE PALABRAS

Esta propiedad varía el espacio entre palabras. Si no se abusa de su uso, esta propiedad permite mejorar la legibilidad del texto. Su sintaxis es:

**word-spacing: valor de longitud (8px) / normal;**

A tener en cuenta:

* Los valores relativos, por ejemplo un porcentaje, no están admitidos en esta propiedad.
* Un valor positivo agrega espacio entre las palabras. Un valor negativo disminuye el espacio entre las palabras.
* La alineación a ambos márgenes del texto exige al navegador separar el espacio entre letras. Por lógica, si indicamos un valor “**word-spacing**” de **0,** impediremos la alineación del texto.

***Veamos un ejemplo práctico de estas propiedades CSS realizando la práctica 6.6.***

INTERLINEADO

Esta propiedad modifica el interlineado del texto. Su sintaxis es:

**line-height: número / valor / porcentaje / normal;**

A tener en cuenta:

* Un número multiplica el interlineado por ese número. Por ejemplo 2 indica un interlineado doble.
* Un valor de longitud indica un interlineado sin tener en cuenta el tipo de fuente. Ej: 16px.
* Un porcentaje determina el interlineado en función del interlineado normal. Ej: 200% indica interlineado doble.
* Si indicamos un valor negativo se disminuirá el interlineado normal y se puede hacer ilegible el texto.
* El valor normal corresponde al interlineado por defecto.

ESPACIOS VACÍOS

Esta propiedad permite controlar los espacios vacíos. Su sintaxis es:

**white-space: pre / nowrap / normal;**

A tener en cuenta:

* El valor **pre** preserva los espacios múltiples y los saltos de línea.
* El valor **nowrap** impide los saltos de línea automáticos y muestra el texto de forma horizontal, sea cual sea el espacio disponible en el navegador.
* El valor **normal** deja que el navegador gestione a su criterio los saltos de línea y los espacios múltiples.

***Para ver el efecto de esta propiedad realice la práctica 6.7.***

ALINEACIÓN HORIZONTAL

Ya hemos utilizado anteriormente esta propiedad. Permite alinear el texto. Su sintaxis es:

**Text-align: left / right / center / justify;**

ALINEACIÓN VERTICAL

Esta propiedad indica la alineación vertical del texto y cualquier otro objeto. Su sintaxis es:

**vertical-align: baseline / sub / super / top / middle / bottom / valor / porcentaje;**

A tener en cuenta:

* Los valores **baseline, sup** y **super** indican la alineación vertical respecto a la línea de texto:
  + **baseline** alinea el texto a la parte inferior de la línea.
  + **sub** escribe el texto por debajo de la línea base.
  + **super** escribe el texto por encima de la línea base
* Los valores **top, middle** y **bottom** alinean el texto respecto al elemento padre.
* Un valor de longitud positivo alinea el texto por encima de la línea base. Un valor negativo lo hace por debajo. Ej: vertical-align: 1.7em;
* Un porcentaje positivo alinea el texto por encima de la línea base. Un porcentaje negativo lo hace por debajo. Ej: vertical-align: -10%;

DIRECCIÓN DEL TEXTO

Esta propiedad permite cambiar la dirección del texto. Su sintaxis es:

**direction: ltr (left to right) / rtl (right to left);**

AÑADIR SOMBRA AL TEXTO

Es posible añadir sobra al texto con la propiedad “text-shadow”. Su sintaxis es:

**text-shadow: x y z color;**

A tener en cuenta:

* **X** es el margen de sombra hacia la derecha.
* **Y** es el margen de la sombra hacia abajo.
* **Z** es la intensidad de degradado o de desenfocado. Es opcional y su valor por defecto 0.
* **Color** es el color de la sombra.
* Es posible utilizar valores negativos para los parámetros x e y. En este caso, la sombra se desplaza hacia la izquierda y hacía arriba

DECLARACIÓN ABREVIADA DE ESTILOS DE FUENTE

La declaración abreviada de estilos de fuente permite agrupar casi todas las propiedades de estilo de fuente en una única declaración siguiendo el siguiente ejemplo:

**font: italic bold small-caps 24pt/1.5 Arial, sans-serif;**

Que se corresponde con:

font-style: italic;

font-weight: bold;

font-variant: small-caps;

font-size: 24pt;

line-height: 1.5;

font-family: Arial, sans-serif;

A tener en cuenta:

* Se puede concluir que la ganancia de tiempo, al tener que teclear menos texto, es importante.
* Los distintos atributos deben informarse seguidos y separados por un espacio.
* No importa el orden de declaración de los distintos atributos. No es obligatorio definir todas las propiedades. Como es lógico, para aquellas propiedades no definidas, el navegador utilizará los valores por defecto.
* Como se puede observar la notación correspondiente al interlineado es algo particular: se indica el valor del tamaño de letra seguido de barra seguido del valor del interlineado.

TIPOS DE LETRA PERSONALIZADOS

Como hemos comentado, los tipos de letras utilizados en las páginas web dependen de las tipografías instaladas en el ordenador del internauta. Sin embargo con la propiedad “**@font-face**” incorporada en **CSS3** es posible importar y utilizar cualquier tipografía personalizada incluso aunque no esté instalada en el ordenador del visitante. De esta manera ahora es posible añadir una gran variedad de tipos de letra y tener la certeza de que nuestras páginas se podrán ver tal y como lo pretendemos.

Añadir tipos de letra personalizada se realiza en dos fases: primero tenemos que descargar el tipo de letra mediante **@font-face** y luego aplicarla como ya sabemos mediante la propiedad “**font-family**”. La sintaxis de **@font-face** es:

**@font-face {**

**font-family: “nombre\_de\_la\_fuente”;**

**src; url(’ruta\_del\_archivo\_de\_la\_fuente’);**

**}**

A tener en cuenta:

* **font-family: “nombre\_de\_la\_fuente”** indica el nombre que se va a utilizar en página web.
* **src: url(‘ruta\_del\_archivo\_de\_la\_fuente’)** indica la dirección completa, si es necesaria, y el nombre del archivo que contiene el tipo de letra.

***Veamos un ejemplo realizando la práctica 6.8.***

Por último comentar que existen tipografías proporcionadas por servicios de empresas como los de **Google Font Api** que están perfectamente adaptadas a la visualización de cualquier ordenador. Para aplicarlas a nuestros diseños web basta con seguir las instrucciones indicadas por la correspondiente empresa. (Si quiere conocer alguno de estos servicios busque en Google la palabra “**webfont**”).

AJUSTAR LA ALTURA DE LOS CARACTERES

A veces tipos de letra distintos pueden parecer de distinto tamaño aunque realmente hayamos indicado que sean iguales. Para solucionar este problema podemos aplicar la propiedad “Font-size-ajust” que permite obtener caracteres siempre del mismo tamaño, tanto para las mayúsculas como para las minúsculas, fijando un valor de coeficiente de aspecto. Su sintaxis es:

**Font-size-ajust: valor;**

Por ejemplo: **Font-size-ajust: 0.465;**

El valor de ajuste hay que definirlo por tanteo pero empieza a haber páginas web dedicadas a esta propiedad que nos pueden facilitar la tarea.

TIPOS DE MARCADORES PARA LISTAS

Esta propiedad permite cambiar la apariencia del marcador o estilo de numeración de una lista. Su sintaxis es:

**list-style-type: disc / circle / square / decimal / decimal-leading-zero / upper-roman / lower-roman / upper-alpha / lower-alpha;**

A tener en cuenta:

* El valor **disc** muestra un círculo relleno.
* El valor **circle** muestra un círculo vacío.
* El valor **square** muestra un cuadrado.
* El valor **decimal** muestra números decimales: 1, 2, 3, etc.
* El valor **decimal-leading-zero** muestra números decimales precedidos de 0: 01, 02, 03, etc.
* El valor **upper-roman** muestra números romanos en mayúscula: I, II, III, IV, etc.
* El valor **lower-roman** muestra números romanos en minúscula: i, ii, iii, iv, etc.
* El valor **upper-alpha** muestra letras en mayúscula: A, B, C, etc.
* El valor **lower-alpha** muestra un letras en minúscula: a, b, c, etc

POSICIÓN DE LAS LISTAS

Esta propiedad determina la posición de la segunda línea y siguientes de un elemento de lista respecto al marcador. Su sintaxis es:

**list-style-position: outside / inside;**

A tener en cuenta:

* El valor **outside** es el valor por defecto y coincide con la presentación normal de las listas.
* El valor **inside** presenta un desplazamiento solamente para la primera línea del elemento de la lista. Las demás líneas del elemento se alinean con el marcador.

INDICAR UNA IMAGEN COMO MARCADOR DE LISTAS

Esta propiedad permite utilizar una imagen como marcador de una lista. Su sintaxis es:

**list-style-image: url(ruta\_del\_archivo\_de\_imagen) / none;**

***Veamos como funcionan estas propiedades realizando la práctica 6.9.***

DECLARACIÓN ABREVIADA PARA ESTILOS DE LISTAS

La declaración abreviada de estilos de listas permite agrupar los valores correspondientes a las listas en una única declaración siguiendo el siguiente ejemplo:

**list-style: url(mimarcador.png) circle inside;**

A tener en cuenta:

* Los valores se reagrupan y separan por espacios.
* Ningún valor es obligatorio, y es posible definir un solo valor.
* Hemos puesto el valor “**circle**” para indicar que si el navegador no encuentra la imagen en la ruta especificada, muestre un marcador de tipo círculo.

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 6.***

CAPÍTULO 7 – USO DE IMÁGENES Y FONDOS

CON HTML y CSS

FORMATOS DE IMAGEN PARA LA WEB

Existen tres formatos de imagen más utilizados en la creación de páginas web. Debemos conocer sus características básicas para emplear el más adecuado a nuestras necesidades. Son:

* Formato GIF (Graphics Interchange Format). Fue el primero que se utilizó en la web.
  + Tiene un peso reducido.
  + Sólo permite un máximo de 256 colores.
  + Permite construir animaciones, lo que se conoce como GIFs animados.
  + No es bueno para fotos pero si para logotipos, botones, puntos, barras y elementos gráficos que no precisen de una paleta de colores muy extensa.
* Formato JPEG (Joint Photographic Expert Group). Es el formato más extendido en el mundo de la fotografía digital.
  + Permite hasta 16,7 millones de colores.
  + Una tasa de comprensión de entre el 15% y el 20% generará una imagen de calidad satisfactoria con un tamaño de archivo reducido.
  + Es un formato excelente para las fotografías aunque puede generar deformaciones en los degradados de color, las esquinas y los bordes.
* Formato PNG (Portable Network Graphic). Se trata de un formato concebido especialmente para uso de imágenes en páginas Web que trata de reunir todas las ventajas de GIF y JPEG. Los profesionales lo están adoptando mayoritariamente sobre todo para fotografías de pequeño tamaño.
  + Soporta 16,7 millones de colores.
  + Su tasa de comprensión de entre 5% y el 25% se realiza sin pérdida de calidad.
  + Permite realizar transparencias de hasta 256 colores.

Hay un nuevo formato de imagen, el WebP, desarrollado por Google y que seguramente gane importancia en los próximos años.

A la hora de utilizar imágenes en la creación de páginas web se recomienda disminuir el tamaño del archivo a una dimensión razonable para mejorar el tiempo de descarga.

INSERTAR UNA IMAGEN CON HTML

Para insertar una imagen en una página web utilizamos la etiqueta **HTML** <**img**>. Su formato es el siguiente:

**<img src=”ruta\_del\_archivo\_de\_imagen” />**

Tenga en cuenta que esta etiqueta no tiene su correspondiente de cierre. Recuerde que nosotros utilizamos la notación **XHTML** aunque no sea necesaria.

***Veamos un ejemplo realizando la práctica 7.1.***

La etiqueta <**img**> tiene el atributo: **alt=”texto explicativo”** que permite incluir una breve descripción de la imagen. Este atributo es muy útil para las personas que utilizan sistemas de lectura de pantalla para navegar por la Web, ya que permite convertir a voz el texto indicado como valor. También sirve para dar un contenido al espacio vacío que se genera cuando, por cualquier razón, la imagen no se puede representar (porque el enlace está roto o el archivo tiene un formato incorrecto, por ejemplo).

Además el contenido del atributo **alt** es usado en los motores de búsqueda para alimentar su base de datos de imágenes.

Otros atributos de la etiqueta <**img**> son “**height**” y “**width**”. Estos dos atributos indican al navegador las dimensiones, en píxeles, que tendrá la imagen en pantalla, ancho y alto respectivamente, que no necesariamente tienen que coincidir con las dimensiones originales.

La definición de estos atributos hace que el navegador reserve un lugar para la imagen antes de descargarla completamente, mientras continúa con la composición de la página y la visualización del texto de la misma. Aunque estos atributos son opcionales, están considerados como esenciales por parte de los diseñadores profesionales.

INSERTAR UN ENLACE SOBRE UNA IMAGEN CON HTML

Los enlaces desde una imagen se realizan simplemente insertando la imagen entre las etiquetas de enlace <**a**> … <**/a**>.

***Veamos un ejemplo realizando la práctica 7.2.***

APLICAR UN COLOR DE FONDO CON CSS

La propiedad **background-color** permite asignar un color de fondo a un elemento. Su sintaxis es:

**background-color: notación de color / transparent;**

A tener en cuenta:

* La notación de color se realiza como ya hemos comentado en el caso de la propiedad “**color**”.
* El valor **transparent** aplicado al fondo de la página muestra el valor por defecto del navegador.

***Compruébelo realizando la práctica 7.3.***

UTILIZAR UNA IMAGEN COMO FONDO CON CSS

La propiedad **background-image** permite asignar una imagen de fondo a un elemento. Su sintaxis es:

**background-image: url(ruta\_archivo\_imagen) / none;**

A tener en cuenta:

* El valor **none** no muestra ninguna imagen de fondo.
* Por defecto la imagen se coloca en la esquina superior izquierda del elemento y se repite tanto vertical como horizontalmente, es decir se muestra como un mosaico.

***Veamos un ejemplo realizando la práctica 7.4.***

Cómo ha podido comprobar nada impide definir un color de fondo y a la vez una imagen de fondo. Jugar con esta posibilidad nos permite crear efectos más o menos creativos. Además, el fondo de color se mostrará en el caso de que no se encuentre la imagen en la ruta indicada.

REPETICIÓN DE LA IMAGEN CON CSS

Con la propiedad **background-repeat** podemos elegir la repetición de la imagen de fondo en el caso de que no queramos que se muestre como un mosaico. Su sintaxis es:

**background-repeat: repeat / repeat-x / repeat-y / no-repeat;**

A tener en cuenta:

* Lógicamente para utilizar esta propiedad previamente tenemos que haber definido una imagen de fondo.
* El valor **repeat** repite la imagen horizontal y verticalmente consiguiéndose el mismo resultado de mosaico de la imagen.
* El valor **repeat-x** repite la imagen horizontalmente.
* El valor **repeat-y** repite la imagen verticalmente.
* El valor **no-repeat** muestra la imagen una sola vez, sin repetirla.

***Veamos un ejemplo realizando la práctica 7.5.***

POSICIONAMIENTO DE LA IMAGEN CON CSS

Hemos dejado la imagen anterior en la esquina superior izquierda de la pantalla, sería estupendo poder colocarla en cualquier parte. Esto es lo que hace la propiedad **background-position**. Su sintaxis es:

**background-position: longitudes en px / porcentajes / left center right top center bottom;**

Por ejemplo: **background-position: 80px 120px;**

**background-position: 30% 55%;**

**background-position: center top;**

A tener en cuenta:

* Lógicamente para utilizar esta propiedad previamente tenemos que haber definido una imagen de fondo.
* También es lógico que utilicemos esta propiedad cuando la imagen no está repetida.
* El primer valor indica la posición respecto al borde izquierdo y el segundo valor respecto al borde superior. Ambos valores tienen que estar separados por un espacio.
* Es posible asignar un único valor que será recogido como el valor horizontal. En este caso el segundo valor, o valor horizontal se definirá de forma automática al valor por defecto que es **center** o **50%**.
* Es posible combinar los valores en porcentaje y en longitud como: **background-position: 150px 40%;**
* Los valores **left**, **center** o **right** determinan la posición horizontal. Mientras que los valores **top**, **center** y **bottom** determinan la posición vertical.
* También se aceptan valores negativos.

FIJAR LA IMAGEN DE FONDO CON CSS

La propiedad **background-attachment** permite fijar una imagen de fondo para que no se mueva con el resto de la página. Su sintaxis es:

**background-attachment: scroll / fixed;**

A tener en cuenta:

* El valor **scroll** permite a la imagen de fondo deslizarse con el contenido de la página.
* El valor **fixed** deja la imagen de fondo fija mientras se desliza el contenido de la página.

***Veámos estas dos propiedades en la práctica realizando la práctica 7.6.***

DECLARACIÓN ABREVIADA DE PROPIEDADES DE FONDO EN CSS

Al igual en que en los estilos de tipo de letra, es posible declarar todas las propiedades correspondientes al fondo de la página web de manera abreviada en una sola línea. Así por ejemplo, la declaración:

**Background: url(mimarcador.npg) blue no-repeat right center fixed;**

Se corresponde con: **background-image: url(mimarcador.npg);**

**Background-color: blue;**

**Background-repeat: no-repeat;**

**Background-position: right center;**

FONDOS MÚLTIPLES CON CSS

Es posible visualizar varias imágenes de fondo en una declaración “**background**”. Un ejemplo sería:

**Background: url(imagen1.png) left top no-repeat,**

**url(imagen2.png) right top no-repeat,**

**url(imagen3.png) right bottom no-repeat;**

DEGRADADOS DE COLOR CON CSS

Aunque a la hora de escribir este manual los degradados de color no están totalmente definidos en las especificaciones de **HTML5** y **CSS3** ya casi todos los navegadores los contemplan. Para aplicar degradados en Firefox utilizaremos las siguientes declaraciones:

A.- Para un degradado lineal:

**Background: -moz-linear-gradient (punto\_de\_origen, color\_inicial, color\_final);**

A tener en cuenta:

* **Linear** indica un degradado lineal.
* **Punto\_de\_origen** se puede escoger entre **top**, **right**, **bottom** o **left** o un par de valores, separados por un espacio, elaborados a partir de **top**, **right**, **bottom**, o **left**. Por ejemplo: **bottom** **right**.
* **Color\_inicial** indica el color de inicio del degradado.
* **Color\_final** indica el color de fin del degradado.

B.- Para un degradado circular:

**Background: -moz-radial-gradient(posición, forma, color\_inicial, color\_final);**

A tener en cuenta:

* **Radial** indica un degradado circular.
* **Posición** es el punto central del gradiente radial. Es una palabra clave a escoger entre **top**, **right**, **bottom**, **left**, center o un par de valores, separados por un espacio, elaborados a partir de **top**, **right**, **bottom**, **left** o **center**. Por ejemplo: **center** **center**.
* **Forma**, la forma del gradiente radial, **circle** o **ellipse**.
* **Color\_inicial** indica el color de inicio del degradado.
* **Color\_final** indica el color de fin del degradado.

***Veámoslo realizando la práctica 7.7.***

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 7.***

CAPÍTULO 8 – ETIQUETAS SEMÁNTICAS Y DE ORGANIZACIÓN EN HTML

INTRODUCCIÓN

Las etiquetas semánticas y de organización nos van a servir para estructurar de una manera lógica nuestra página web y para describir algunos elementos de la página. Con ello facilitaremos que los motores de búsqueda encuentren mejor la información que transmitimos y mejoramos la legibilidad del código al hacerlo más coherente facilitando el reconocimiento de sus distintas secciones.

Estas etiquetas no tienen ninguna influencia sobre la presentación y visualización de la página. Solo se utilizan para mejorar la estructura.

El concepto de web semántica es una idea original y muy defendida por Tim Berners-Lee el inventor de la web.

Si piensa en las páginas web que podemos ver hoy en día en la red, todas tienen una estructura muy semejante, en la que podemos distinguir los siguientes elementos:

1.- Un encabezado que suele contener un logotipo, un banner, el nombre del sitio web, una caja de búsqueda…

2.- Un menú de navegación para pasar a las diferentes secciones del sitio web.

3.- El contenido propiamente dicho de la página web.

4.- Una parte con publicidad u otra información relacionada con la página web.

5.- Un pie de página que suele contener, el copyright, la empresa autora de la web, indicaciones legales, etc, etc.

Para explicarlo visualmente, la mayoría de las páginas web que visitamos en la actualidad suelen tener alguna de estas formas:

A)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

B)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

C)

Encabezado de página

Navegación

Contenido de la Página

Pie de página.

Información

adicional

Para indicar a **HTML** la estructura lógica de nuestra página web utilizamos las etiquetas semánticas y de organización.

ETIQUETAS SEMANTICAS DE ORGANIZACIÓN:

HEADER

La etiqueta <**header**> indica al navegador la cabecera de nuestra página web. En ella generalmente podemos encontrar el nombre del sitio web, una imagen, un logotipo, un elemento h1-h6, un formulario de búsqueda, etc, etc.

***Veámoslo en la práctica 8.1.***

NAV

La etiqueta <**nav**> indica al navegador una sección de enlaces o links de navegación.

***En nuestra página web de ejemplo podríamos utilizarla como en la práctica 8.2.***

SECTION

La etiqueta <**section**> se utiliza para indicar una sección general o principal dentro de un documento. El ejemplo más claro para diferenciar una sección sería el capítulo de un libro. Dentro de <**section**> podemos incluir subsecciones y si utilizamos **h1-h6**, podemos estructurar mejor toda la página creando jerarquías en el contenido, algo muy favorable para el buen posicionamiento web.

***Apliquemos esta etiqueta a nuestra página realizando la práctica 8.3.***

ASIDE

La etiqueta <**aside**> especifica una sección de la página que abarca un contenido tangencialmente relacionado con el contenido que lo rodea, por lo que se le puede considerar un contenido independiente. Se puede utilizar para barras laterales, elementos publicitarios, grupos de enlaces de navegación u otro contenido que consideremos separado pero relacionado con el contenido principal de la página.

En nuestra web de práctica crearemos una barra lateral con cierto contenido que no es la información principal de nuestra web. ***¡Veámoslo realizando la práctica 8.4!***

Al igual que en las etiquetas semánticas y de organización precedentes <**aside**> solamente especifica que el texto que abarca es una información adicional o paralela a la ofrecida en el bloque principal de la web. Aunque pretendemos que aparezca en una barra lateral, como esta etiqueta no da formato, el texto sigue apareciendo debajo del resto de la página web. Seguimos esperando a las posibilidades de **CSS** para conseguir la presentación que deseamos.

FOOTER

La etiqueta <**footer**> indica el pie de una página web o una sección de ella. En el pie habitualmente se muestra información acerca de la página que tiene poco que ver con su contenido, como el autor, el copyright, el año, etc, etc…

***En nuestra web de ejemplo está claro lo que pondremos como pie de página… Para incluir el <footer> en nuestra web realizamos la práctica 8.5.***

ARTICLE

La etiqueta <**article**> El elemento de artículo representa un componente de una página que consiste en un texto autónomo en un documento, página, aplicación, o sitio web con la intención de que pueda ser reutilizado y repetido. Habitualmente se utiliza en los post de los foros, las entradas de un blog, un comentario escrito por un usuario, etc, etc.

***En nuestra web de práctica aplicaríamos <article> realizando la práctica 8.6.***

Tenemos que tener en cuenta que nada impide utilizar etiquetas <**nav**> en los <**footer**>, <**header**> dentro de <**section**>, <**header**> dentro de <**article**>, etc, etc, siempre que nos ayuden a tener más clarificado el código **HTML**. Por ejemplo se puede incluir más de un elemento de cabecera de página <**header**>, podemos tener una cabecera de página para todo el sitio web y también una cabecera para una sección concreta. Sin embargo, no se debe abusar de las etiquetas de organización ya que se corre el peligro de “ensuciar” el código más que organizarlo.

DIV

¿Y qué pasa si en algún momento necesitamos hacer una división o crear una sección dentro del contenido de la página web que no coincida exactamente con la cabecera, el pie, <**section**>, <**aside**>, <**nav**> o <**article**>? En esos casos tendremos que utilizar la etiqueta <**div**>.

Podemos considerar a la etiqueta <**div**> como un contenedor genérico que nos permitirá crear un bloque de contenido totalmente independiente. Por sí solo, la etiqueta <**div**> no dice nada. Se utiliza para agrupar contenido en bloque y darle formato mediante el uso de atributos **ID** o identificadores de clase y **CSS**.

En una página web podemos tener todos los elementos <**div**> que consideremos necesarios. De hecho, hasta la llegada de **HTML5** se utilizaba esta etiqueta para crear el armazón visual de la página web.

***Para ver un ejemplo de utilización de la etiqueta <div> realice la práctica 8.7.***

OTRAS ETIQUETAS SEMÁNTICAS

<**abbr**>. Indica una abreviatura, por ejemplo S.A. o IVA. Además sirve para indicar a los programas de síntesis vocal que no traten de leer la palabra tal y como está escrita, sino deletreándola.

<**address**>. Indica una dirección de contacto del autor o propietario de un documento o artículo. No debe usarse para describir una dirección postal a menos que sea parte de la información de contacto. Es uno de los elementos típicos de <**footer**>. Su contenido se visualiza por defecto en cursiva y con letra más pequeña.

<**cite**>. Indica el título de una obra. Por ejemplo: un libro, una canción, una película... Se visualiza en cursiva.

<**code**>. Señala una sintaxis o un código informático. Se visualiza en un tipo de letra de paso fijo.

<**samp**>. Pone de relieve un texto de ejemplo. Se visualiza en un tipo de letra de paso fijo.

<**dfn**>. Indica la definición de un término. Se visualiza en cursiva.

<**kbd**>. Indica al usuario las letras del teclado que tiene que pulsar. Se visualiza en un tipo de letra de paso fijo.

<**var**>. Indica que contiene una variable. Se visualiza en cursiva.

<**strong**>. Ya vista en la página 21.

<**em**>. También vista en la página 21.

<**hgroup**>. Indica que se trata de un grupo de títulos <hx>.

<**hr**>. Ya vista en la página 14.

<**time**>. Define una fecha o una hora. El atributo opcional “datetime” indica la fecha y la hora siguiendo la representación numérica internacional de la norma ISO8601.

<**mark**>. Remarca parte del texto que se muestra subrayado sobre un fondo amarillo.

<**meter**>. Define una medida. Se usa para medidas con valores máximos y mínimos conocidos como por ejemplo uso del disco. Solo es reconocida por Chrome y Opera. Esta etiqueta tiene seis atributos:

* **value**: el valor del dato sobre la escala.
* **min**: el valor mínimo posible.
* **low**: el valor mínimo esperado.
* **high**: el valor máximo esperado.
* **max**: el valor máximo posible.
* **optimum**: el valor máximo ideal.

<**progress**>. Representa el progreso de una tarea. Se utiliza conjuntamente con JavaScript. Tiene dos atributos:

* **max**: Especifica que cantidad de trabajo requiere la tarea total.
* **value**: Especifica la cantidad de tarea que se ha completado.

<**figure**>. Sirve para agrupar elementos tales como imágenes y videos.

<**figcaption**>. Usada de forma conjunta con la etiqueta <**figure**> proporciona una leyenda a los elementos agrupados.

<**details**>. Indica detalles o contenido accesorio. Actualmente solo es soportada por chrome.

<**summary**>. Usada conjuntamente con la etiqueta <**details**>, proporciona un resumen de los elementos accesorios. Actualmente solo es soportada por chrome.

CATEGORÍAS DE CONTENIDO EN HTML

Llegados a este punto de nuestro curso es conveniente hacer una reflexión sobre qué tipo de contenido podemos encontrar en una página web. Esto nos ayudará a entender o clarificar cómo se organiza la información y cómo la entienden los navegadores.

Podemos dividir el contenido de una página web en varias categorías:

* **Metadatos**: Establece el comportamiento del resto del contenido de la página. Un ejemplo de metadatos es la etiqueta <**meta charset**> que hemos utilizado para indicar el juego de caracteres que utilizamos en nuestra página web. Los metadatos se ubican en la sección <**head**> del documento. Veremos más sobre los metadatos más adelante.
* **Flujo**: Son los elementos que constituyen el contenido de una página web. Normalmente será texto o un archivo insertado, como una imagen o un video.
* **Secciones**: Se trata de un subconjunto del contenido de flujo. A las secciones corresponden algunas etiquetas que hemos estudiado en este tema: <**section**>, <**article**>, <**nav**>, <**aside**>, <**header**>, <**footer**>.
* **Títulos**: Se trata de los titulares de la página web, las etiquetas <**h1**>, <**h2**> y sucesivos, además de la etiqueta <**hgroup**>. El contenido de los titulares forma parte del contenido de flujo.
* **Textos**: Engloba todo el texto de la página web. Es un subconjunto del contenido de flujo.
* **Contenidos embebidos**: Son aquellos que se importan de otros recursos dentro de la página, como imágenes, videos, archivos para descargar…
* **Contenidos interactivos**: La etiqueta <**a**> y otros elementos, como los formularios que veremos más adelante, y que permiten interactuar con el usuario de la página web.

ELEMENTOS EN BLOQUE Y ELEMENTOS EN LÍNEA

Otro concepto que debemos tener claro para ser más eficientes en nuestros diseños web es la distinción entre elementos de bloque y elementos de línea.

Las etiquetas de renderizado en bloque ocupan por defecto toda la longitud de la ventana del navegador o del elemento que las contiene. Se sitúan unas debajo de otras y generalmente el navegador inserta automáticamente un espacio encima y debajo del contenido de la etiqueta.

Así por ejemplo el código:

**<p>Párrafo 1</p><p>Párrafo 2</p>**

aunque está escrito en una sola línea, en el navegador se visualiza en dos líneas puesto que la etiqueta <**p**> es una etiqueta de tipo bloque. Cada párrafo va a ocupar una línea.

Esto es lo que nos ha sucedido en la última modificación del archivo “**turismo.htm**” que nos muestra primero la imagen y debajo de ella el título centrado de la página. De la misma manera, el resto de contenido del archivo, va situándose uno debajo del otro en toda la web.

Algunas etiquetas de bloque son: <**p**>, <**hx**>, <**hr**>, <**ol**>, <**ul**>, <**dl**>, <**div**>, <**header**>, <**footer**>, <**section**>, <**article**>, <**nav**>, etc, etc…

Por el contrario las etiquetas en línea se sitúan siempre uno al lado del otro permaneciendo en el mismo flujo de texto.

Así por ejemplo el código:

**texto en <b>negrita</b> y en <i>cursiva</i>**

se escribe en una sola línea sin interrumpir el flujo del texto.

Algunas etiquetas en línea son: <**b**>, <**i**>, <**br**>, <**a**>, <**img**>, <**span**>…

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 8.***

CAPÍTULO 9.- PROPIEDADES CSS DE CAJA

Y VISUALIZACIÓN

HOJAS DE ESTILO EXTERNAS, HERENCIA Y HOJAS EN CASCADA

**HOJAS DE ESTILO EXTERNAS**.

En temas anteriores hablamos de las formas de incorporar las hojas de estilo **CSS** a nuestras páginas web. Hasta ahora hemos utilizado estilos incorporados en el mismo archivo de código **HTML**, situados dentro de la sección de cabecera <**head**> de la página. Esta modalidad recibe el nombre de hoja de estilos interna.

Además de hojas de estilo internas, también existen las hojas de estilos externas y los estilos aplicados directamente “**en línea**”, en la propia etiqueta **HTML**.

Una hoja de estilo externa es un documento independiente con la extensión .**css**. Cuando utilizamos un archivo externo, en él únicamente se definen estilos **CSS** y en el de código **HTML** tenemos que insertar un enlace a dicho archivo.

La ventaja de utilizar hojas de estilo externas es que una misma hoja de estilo se puede utilizar para distintos archivos de código **HTML**, mientras que en las hojas de estilo internas, las reglas **CSS** solo se aplican al documento en donde residen.

Por ejemplo, si tenemos un sitio web con 14 páginas y utilizamos hojas de estilo internas, debemos crear una hoja de estilo dentro de cada una de esas 14 páginas. Si tuviéramos que hacer un cambio cualquiera en los estilos tendríamos que modificar todas y cada una de las hojas lo que supone una pérdida de tiempo muy considerable y una posible fuente de errores y equivocaciones. Es lo que hemos tenido que hacer hasta ahora con nuestras webs de prácticas referidas al turismo.

Otra de las ventajas es que este método de utilizar hojas de estilo externas para dar formato a una página web respeta al máximo las prescripciones del **W3C** de separación entre contenido y presentación de las que tantas veces hemos hablado en este curso.

**NOCIÓN DE CASCADA**

Nada impide que para un mismo archivo de código **HTML** utilicemos hojas de estilo en línea, internas y externas a la vez. En este caso se podría dar algún tipo de conflicto u orden de estilo contradictorio entre los distintos formatos que indiquemos en ellas que queremos aplicar. Si fuera así se aplicaría la **NOCIÓN DE CASCADA**, (recuerde que **CSS** es Cascading Style Sheets), u orden de prioridad que de menor a mayor es:

* Propiedades de formato por defecto del navegador.
* Hojas de estilo externas.
* Hojas de estilo internas.
* Hojas de estilo en línea.

Por tanto la regla de prioridad, para visualizar el documento en el navegador, consiste en utilizar la regla de estilo más próxima a la etiqueta **HTML**.

**NOCIÓN DE HERENCIA**

Un último concepto teórico que debemos dominar llegados a este punto, es el de **HERENCIA**. La herencia en **CSS** es el mecanismo mediante el cual determinadas propiedades de un elemento padre se transmiten a sus hijos. Por ejemplo, si aplicamos un tamaño de fuente a la etiqueta <**section**> todo el texto, de todas las etiquetas contenidas dentro de <**section**> heredan ese tamaño de fuente. Podríamos decir por tanto que una etiqueta <**p**> contenida dentro de <**section**> es hija de <**section**>. O al revés que <**section**> es el contenedor padre de <**p**>.

Para entender el porqué de la herencia basta con pensar que pasaría si no existiera… ¡Entonces tendríamos que especificar individualmente todo el formato, familia de fuente, tamaño, color, etc, etc, para todos y cada uno de los elementos de nuestro código **HTML**.!

Pero es hora de ponernos manos a la obra. ***Procederemos a crear una hoja de estilo externa realizando la práctica 9.1.***

HOJAS DE ESTILO DE RESET

Muchos diseñadores web utilizan “archivos de reset de **CSS**”. Se trata de una hoja de estilo externa que “pone a cero” todos los valores de formato aplicados por defecto por los diferentes navegadores.

Efectivamente cada navegador web aplica sus propias reglas para mostrar los elementos **HTML** en pantalla. Esto puede significar que, a pesar de que las diferencias sean mínimas, los diseños web que hayamos creado no se vean igual en distintos navegadores. Por ejemplo, un navegador determinado puede dejar un margen de 10px para un elemento <**h1**> y otro, exagerando un poco, de 15px.

Una manera de homogeneizar las presentaciones en pantalla es crear un archivo **CSS** de reset que elimine los estilos por defecto aplicados a los elementos **HTML** utilizados con más frecuencia. Con este reseteo de valores podemos conseguir una base fiable y coherente sobre la cual empezar nuestro diseño.

Muchos diseñadores experimentados se preparan sus propias hojas de estilo de reset. No obstante existen en internet numerosas distribuciones gratuitas. Nosotros, para ilustrar su uso, utilizaremos el de Eric Mayer que puede consultar en su web <http://meyerweb.com> y que está disponible en su carpeta de prácticas.

***Para ver el resultado de aplicar una hoja de estilos de reset realice la práctica 9.2.***

CONCEPTO DE CAJA O BLOQUE

Otra noción importante que debemos dominar antes de empezar a dar formato a nuestra web es el concepto de caja o bloque. Su dominio nos permitirá situar bien todos los elementos en la web.

Muchas etiquetas **HTML** son tratadas por los navegadores como un elemento de caja o bloque o contenedor, que con todos estos nombres nos lo podemos encontrar. Está constituido por:

* Un contenido
* Un margen interior (padding)
* Un borde (border)
* Un margen exterior (margin)

Margin= margen externo

Border = borde

Padding = margen interno

CONTENIDO

Gráficamente:

Comprendidos los componentes de una caja o contenedor hay que tener en cuenta que a la anchura y/o altura del contenido hay que sumar la anchura del borde y los márgenes para así obtener las medidas finales del contenedor.

Visto con un ejemplo: si tenemos una caja con una anchura del contenido de 200px, un margen interior de 20px, una anchura de borde de 10px y un margen exterior de 20px, la anchura final de todo el contenedor es de 300px (De izquierda a derecha: 20px+10px+20px+200px+20px+10px+20px = 300px).

Las hojas de estilo **CSS** proporcionan al diseñador un control total sobre la altura y anchura del contenido, los márgenes internos y externos y los bordes de un contenedor como vamos a ir viendo.

PROPIEDADES DE ANCHURA Y ALTURA

Ha llegado el momento de ir dando formato a nuestra página web de prueba empezando desde cero.

Las propiedades **width** y **height** fijan respectivamente la anchura y la altura del contenido de un elemento caja sin márgenes ni bordes. Su sintaxis es:

**width: auto / valor / porcentaje;**

y

**height: auto /valor / porcentaje;**

A tener en cuenta:

* El valor **auto** ajusta automáticamente la dimensión del contenido.
* El valor de longitud especifica un valor fijo.
* El porcentaje especifica un valor relativo.

***Veámoslo en la práctica realizando la práctica 9.3.***

MÁRGENES EXTERNOS

La propiedad **margin** permite definir el margen exterior del elemento caja. Su sintaxis es:

**margin: auto / valor / porcentaje;** o,

**margin-top: auto / valor / porcentaje;** o,

**margin-right: auto / valor / porcentaje;** o,

**margin-bottom: auto / valor / porcentaje;**  o,

**magin-left: auto / valor / porcentaje;**

A tener en cuenta:

* El valor **auto** deja que el navegador ajuste automáticamente los márgenes externos.
* Un valor de longitud especifica un valor fijo para los márgenes exteriores.
* Un porcentaje define la longitud respecto a un elemento padre.
* **Margin** especifica los cuatro márgenes pero podemos especificar el margen exterior de un solo lado del contenedor usando respectivamente margin-top, margin-right, margin-bottom y margin-left.
* No obstante también utilizando **margin** podemos especificar de forma abreviada el margen exterior de uno o varios lados del contenedor de la siguiente forma:
  + **margin: 10px;** Si se indica una única cantidad se aplica a los cuatro lados.
  + **margin: 10px 10px;** Si se indican dos cantidades la primera se aplica a los lados superior e inferior y la segunda a los márgenes laterales derecho e izquierdo.
  + **margin: 10px 10px 10px;** Si se indican tres valores el primero se aplica al lado superior, el segundo a los márgenes laterales y el tercero al lado inferior.
  + **margin:** **10px 10px 10px 10px;** Indicando cuatro cantidades se aplican a los cuatro márgenes exteriores en el sentido de las agujas del reloj partiendo del lado superior.
* Se pueden utilizar cantidades negativas.

***¡Practiquémoslo realizando la práctica 9.4!***

¿ANCHURA FIJA O VARIABLE PARA NUESTRA WEB?

Una de las primeras cosas que hay que decidir a la hora de iniciar el desarrollo de una página web es decidir que anchura va a tener. Existen dos maneras de diseño atendiendo a la anchura de la web:

* **Web de Anchura fija**. En estas páginas todos los elementos se incluyen dentro de un contenedor que tiene un ancho exacto. En nuestro caso vamos a utilizar como contenedor la misma etiqueta <**body**> y como ancho 960 píxeles. Este diseño permite ubicar con precisión todos los elementos de la página. Es la estructura que emplearemos
* **Web de anchura variable**. Están pensadas para adaptarse a la anchura de la ventana del navegador en cualquier momento. Es muy útil para diseños web en los que se prevé que el usuario acceda a la web por dispositivos con resoluciones de pantalla variadas: tablets, dispositivos móviles, etc, etc.

Como hemos comentado en esta práctica nosotros emplearemos una estructura de anchura fija de 960 píxeles.

GROSOR DEL BORDE

La propiedad **border-width** permite definir el grosor de los cuatro bordes simultánea o individualmente. Su sintaxis es:

**border-width: valor / thin / medium / thick;** o,

**border-top-width: valor / thin / medium / thick;** o,

**border-right-width: valor / thin / medium / thick;** o,

**border-bottom-width: valor / thin / medium / thick;** o,

**border-left-width: valor / thin / medium / thick;**

A tener en cuenta:

* El grosor no se muestra si no se ha definido **border-style**.
* El valor **thin** indica un borde fino.
* Un valor **medium** indica un borde medio.
* El valor **thick** ordenada un borde grueso.
* La interpretación de **thin**, **médium** y **thick** puede variar de un navegador a otro.
* **Border-width** especifica los cuatro bordes pero podemos especificar el borde de un solo lado como ya sabemos por el caso de los márgenes exteriores.
* También podemos especificar de forma abreviada el borde de uno o varios lados del contenedor de la siguiente forma:
  + Si se indica un grosor se aplica a los cuatro bordes.
  + Si se indican dos grosores el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres grosores el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro grosores se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

***¡Practiquémoslo realizando la práctica 9.5.!***

ESTILO DEL BORDE

La propiedad **border-style** permite definir el grosor de los cuatro bordes simultánea o individualmente. Su sintaxis es:

**border-style: solid / dashed / dotted / double / groove / ridge / inset / outset / hidden / none;**

o,

**border-top-style: “** o,

**border-right-style:**  **“** o,

**border-bottom-style: “** o,

**border-left-style: “**

A tener en cuenta:

* El el estilo del borde no se muestra si no se ha definido **border-width**.
* Los diferentes estilos del borde son:

|  |  |  |
| --- | --- | --- |
| Solid | Un trazo liso | **solid** |
| Dashed | Guiones | **dashed** |
| Dotted | Pequeños puntos | **dotted** |
| Double | Trazos lisos dobles | **double** |
| Groove | Efecto 3D de relieve incrustado en la página. | **groove** |
| Ridge | Efecto 3D de relieve saliendo de la página. | **ridge** |
| Inset | Bordes entrantes, incrustados en la página | **inset** |
| Outset | Bordes salientes, biselados en la página | **outset** |
| Hidden | Sin borde. Influye sobre el borde adyacente. | **hidden** |
| None | Sin borde. | **none** |

* **Border-style** especifica el estilo de los cuatro bordes pero podemos especificar el de un solo lado como ya sabemos.
* También podemos especificar de forma abreviada el estilo del borde de uno o varios lados del contenedor de la siguiente forma:
  + Si se indica un estilo se aplica a los cuatro bordes.
  + Si se indican dos estilos el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres estilos el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro estilos se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

***¡Veamos su funcionamiento realizando la práctica 9.6!***

COLOR DEL BORDE

La propiedad **border-color** permite definir el color de los cuatro bordes de un contenedor simultánea o individualmente. Su sintaxis es:

**border-color: nombre / notación / transparent;** o,

**border-top-color: nombre / notación / transparent;** o,

**border-right-color: nombre / notación / transparent;** o,

**border-bottom-color: nombre / notación / transparent;** o,

**border-left-color: nombre / notación / transparent;**

A tener en cuenta:

* El color no se puede aplicar si no se ha definido **border-style** y **border-width**.
* El valor indicado en **border-color** define el color del borde de los cuatro lados del contenedor.
* Podemos especificar el color del borde de un solo lado con las propiedades **border-top-color, border-right-color, border-bottom-color y border-left-color**.
* También podemos especificar de forma abreviada el borde de uno o varios lados del contenedor con la propiedad **border-color** de la siguiente forma:
  + Si se indica un solo color se aplicará a los cuatro bordes.
  + Si se indican dos colores el primero se aplica a los bordes superior e inferior y el segundo a los bordes laterales derecho e izquierdo.
  + Si se indican tres colores el primero se aplica al borde superior, el segundo a los bordes laterales y el tercero al borde inferior.
  + Indicando cuatro colores se aplican a los cuatro bordes en el sentido de las agujas del reloj partiendo del lado superior.

ESCRITURA ABREVIADA DEL BORDE

Podemos escribir todos los valores de los bordes de caja de forma abreviada utilizando la propiedad “**border**”. Un ejemplo de su sintaxis es:

**border: red double 5px;**

A tener en cuenta:

* Esta forma solo vale en el caso de que los cuatro bordes sean idénticos.
* No importa el orden de los diferentes valores. Eso sí, tiene que ir separadas por espacio.
* Para asegurar la compatibilidad óptima entre los distintos navegadores, se recomienda informar las tres propiedades.

Semejante a “**border**” existe también la propiedad “**outline**” que genera un borde exterior pero no ocupa espacio y no es necesariamente rectangular. Además “**outline-offset**” permite establecer una distancia entre el borde interior y el exterior.

***¡Veámoslo en la práctica 9.7.!***

FORMATEANDO EL ELEMENTO <HEADER>

 Vamos a abordar ahora el formato del elemento <**header**>. Dejaremos la foto a la izquierda y pondremos el título de la página centrado a su izquierda con un fondo degradado rojo que recuerde a España. En la siguiente imagen podemos hacernos una idea del resultado final.

SELECTORES JERÁRQUICOS

A parte de los selectores que ya conocemos: de clase, de etiqueta, de identificador, existen otros muchos que iremos explicando a lo largo del curso. Ahora vamos a trabajar con selectores jerárquicos.

Los selectores jerárquicos son aquellos que aprovechan la circunstancia de que unos elementos **HTML** están anidados dentro de otros. Podríamos decir entonces, que unos elementos son hijos de otros. Así en el código que acabamos de escribir la etiqueta <**h1**> es hija de <**div**> pues <**h1**> está dentro de <**div**>. Por la misma razón, <**div**> sería el elemento padre de <**h1**>.

Siguiendo con este ejemplo podemos considerar tres selectores jerárquicos:

|  |  |  |
| --- | --- | --- |
| Div h1 | Selector descendente | Aplicaríamos los estilos a todos los <**h1**> que estén dentro de <**div**>. |
| Div > h1 | Selector hijos | Aplicaríamos los estilos a todos los <**h1**> que sean hijos directos de <**div**>. |
| Div + h1 | Selector hermano adyacente | Aplicaríamos los estilos a cualquier elemento <**h1**> precedido inmediatamente del elemento <**div**> |

***Para ver el funcionamiento de todo esto realice ahora la práctica 9.8.***

FLOTAR

La propiedad **float** permite retirar un elemento de tipo caja del flujo normal para situarlo lo más a la derecha o lo más a la izquierda posible dentro de su elemento padre. Su sintaxis es:

**float: right / left / none;**

A tener en cuenta:

* El valor **right** alinea a la derecha el elemento indicado, empujando a los demás elementos a la izquierda.
* El valor **left** alinea a la izquierda el elemento indicado, empujando a los demás elementos a la derecha.
* El valor **none** deja la gestión al navegador.

***¡Veamos el resultado realizando la práctica 9.9.!***

MÁRGENES INTERNOS

Tenemos que centrar el texto del título verticalmente. Para ello podríamos aplicar la propiedad “**vertical-aling**” que ya vimos en páginas anteriores pero desgraciadamente esta propiedad no funciona en todos los contenedores por lo que, en este caso, tendremos que utilizar un pequeño truco aprovechándonos de los márgenes internos.

La propiedad **padding** permite definir el margen interior del elemento caja. Su sintaxis es:

**padding: auto / valor / porcentaje;** o,

**padding-top: auto / valor / porcentaje;** o,

**padding-right: auto / valor / porcentaje;** o,

**padding-bottom: auto / valor / porcentaje;** o,

**padding-left: auto / valor / porcentaje;**

A tener en cuenta:

* El valor **auto** deja que el navegador ajuste automáticamente los márgenes internos.
* Un valor de longitud especifica un valor fijo para los márgenes interiores.
* Un porcentaje define la longitud respecto a un elemento padre.
* **Padding** especifica los cuatro márgenes pero podemos especificar el margen interior de un solo lado del contenedor usando respectivamente padding-top, padding-right, padding-bottom y padding-left.
* No obstante también utilizando **padding** podemos especificar de forma abreviada el margen interior de uno o varios lados del contenedor de la siguiente forma:
  + **padding: 10px;** Si se indica una única cantidad se aplica a los cuatro lados.
  + **padding: 10px 10px;** Si se indican dos cantidades la primera se aplica a los lados superior e inferior y la segunda a los márgenes laterales derecho e izquierdo.
  + **padding: 10px 10px 10px;** Si se indican tres valores el primero se aplica al lado superior, el segundo a los márgenes laterales y el tercero al lado inferior.
  + **padding:** **10px 10px 10px 10px;** Indicando cuatro cantidades se aplican a los cuatro márgenes interiores en el sentido de las agujas del reloj partiendo del lado superior.

***¡Apliquémoslo realizando la práctica 9.10.!***

BORDES REDONDEADOS

La propiedad **border-radius** permite formatear contenedores con bordes redondeados. Su sintaxis es:

**border-radius: x y;**

A tener en cuenta:

* **X** e **Y** pueden ser un valor o un porcentaje.
* Los valores de **X** e **Y** determinan los radios horizontal y vertical de un cuarto de elipse, lo cual provocará la curvatura de la esquina.
* Es posible indicar un único valor. En ese caso el valor de **X** será igual al valor de **Y**.
* Como en otras propiedades **border-radius** especifica las cuatro esquinas pero podemos especificar el borde redondeado de una sola esquina del contenedor usando respectivamente border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius y border-top-left-radius.
* También es posible definir el borde redondeado para cada esquina a partir de una escritura abreviada que se lee, como de costumbre, según el sentido de las agujas del reloj comenzando por la parte superior.

***Para poner los bordes redondeados de la parte derecha de nuestro contenedor <div> realice la práctica 9.11.***

Con referencia a los bordes simplemente dejaremos aquí anotado que existe la propiedad **border-image** que permite asignar una imagen como borde de un contenedor.

EFECTO SOMBREADO SOBRE UN ELEMENTO CAJA

La propiedad **box-shadow** añade sombra a un elemento de caja. Su sintaxis es:

**box-shadow: x y z color inset;**

A tener en cuenta:

* **X** es el margen de sombra hacia la derecha.
* **Y** es el margen de sombra hacia abajo.
* **Z** es la intensidad de degradado o desenfocado. (Opcional con valor por defecto 0).
* **Color** es el color de la sombra.
* **Inset.** Indica que la sombra sea interna.
* Los parámetros **X** e **Y** admiten valores negativos. En este caso la sombra se desplaza hacia la izquierda y hacia arriba.
* Es importante tener en cuenta que la sombra se aplica sobre la caja del elemento, no sobre su borde por lo que la sombra no afecta al tamaño de la caja.

***¡Vayamos a la práctica y pongamos sombra al contenedor <div> del <header> realizando la práctica 9.12!***

FORMATEANDO EL ELEMENTO <NAV>

 En nuestra zona de navegación utilizaremos un fondo negro sobre el colocaremos los enlaces tal y como se ve en la siguiente imagen.

***Para ello realice la práctica 9.13.***

DESPEJAR

La propiedad “**clear**” anula el efecto introducido por la propiedad “**float**”. Su sintaxis es:

**clear: right / left / both /none;**

A tener en cuenta:

* **Right** anula los elementos flotantes a la derecha.
* **Left** anula los elementos flotantes a la izquierda.
* **both** anula los elementos flotantes a ambos lados.
* **None** anula los elementos flotantes.

***¡Apliquémoslo realizando la práctica 9.14.!***

VISUALIZAR ELEMENTOS EN BLOQUE O EN LÍNEA

Vamos a dar formato a los elementos <**li**> correspondientes a los enlaces de <**nav**>.

Los cuatro enlaces que tenemos están uno debajo de otro porque son elementos en bloque que como ya sabemos ocupan todo el espacio horizontalmente. Para que aparezcan en una sola línea uno detrás de otro debemos indicar que se comporten como elementos en línea. Esto lo podemos hacer con la propiedad “**display**”. Su sintaxis es:

**display: block / inline / none;**

A tener en cuenta:

* **Block** redefine un elemento en línea como un elemento en bloque.
* **Inline** redefine un elemento en bloque como un elemento en línea.
* **None** define un elemento que no se va a mostrar y por lo tanto se retira del documento y de la visualización de la página. Es como si no existiera dicho elemento.
* Esta propiedad es muy rica en posibilidades de presentación y se utiliza a menudo en hojas de estilo **CSS** complejas.

Si observa la hoja de estilos de reset verá que muchos elementos en bloque se les pone en línea o viceversa. ¡Compruébelo abriendo la hoja de estilos “**reset.css**”!

***Para efectuar los cambios de formato en los elementos <li> realice la práctica 9.15.***

FORMATEANDO EL ELEMENTO <SECTION>

 Vamos a maquetar el cuerpo principal de nuestra web que está indicado bajo la etiqueta <**section**>. Muchas de las propiedades de **CSS** que manejaremos ya las hemos aprendido pero veremos alguna nueva. Al terminar este proceso la web tendrá un aspecto parecido al de la siguiente imagen:

***Abordemos el formateo realizando la práctica 9.16.***

FORMATEANDO LOS ELEMENTOS <ARTICLE>

Los elementos <**article**> los vamos a colocar horizontalmente como si se tratara de celdas de una tabla. Para ello vamos a utilizar un valor de la propiedad “**display**” que no comentamos en su momento y que estudiaremos más adelante.

***Para ello realice ahora la práctica 9.17.***

DAR FORMATO AL ELEMENTO <ASIDE>

Aunque el elemento <**aside**> puede situarse en cualquier parte de una página web, en la mayoría de los casos se utiliza para las barras laterales, como es nuestro caso.

***Realice la práctica 9.18.***

FORMATEANDO EL ELEMENTO <FOOTER>

Con todo lo que hemos practicado hasta ahora dar formato al pie de página va a resultarnos muy sencillo. **Realice para ello la práctica 9.19.**

Con esta última práctica hemos terminado de dar formato a nuestra página web. Aún podemos realizar algunas mejoras y cambios que iremos viendo en lo que queda de curso.

***Por ahora para terminar de dominar esta parte del tema realice la primera ejercitación del capítulo 9.***

POSICIONAMIENTO DE ELEMENTOS HTML CON CSS

Las hojas de estilo **CSS** proporcionan las herramientas adecuadas para posicionar con una precisión de pixel, cualquier elemento de un documento **HTML**. Aplicando la propiedad “**position**”, un elemento puede posicionarse de cuatro formas:

A.- **Posicionamiento estático**: Es el posicionamiento normal del elemento. La sintaxis es:

**position: static;**

A tener en cuenta:

* El diseñador no tiene el control.
* El elemento no puede posicionarse ni reposicionarse.
* La visibilidad, propiedad que veremos más adelante, no puede modificarse.

B.- **Posicionamiento relativo**: Es el posicionamiento de un elemento respecto a su posición normal o estática. Su sintaxis es:

**position: relative; left: valor ó %;**

**Top: valor ó %;**

**Right: valor ó %;**

**Bottom: valor ó %;**

A tener en cuenta:

* El elemento permanece en el flujo de los datos aunque está, en cierta medida, descentrado respecto a su posición normal.
* La posición se define mediante las coordenadas (**x,y**) donde:
  + **X** es la distancia respecto al borde izquierdo del elemento padre o de la ventana del navegador (eje horizontal). De esta manera:
    - “**left**” determina la distancia entre la parte izquierda del elemento y el borde izquierdo de la página.
    - “**right**” determina la distancia entre la parte derecha del elemento y el borde derecho de la página.
  + **Y** es la distancia respecto al borde superior del elemento padre o de la ventana del navegador (eje vertical). De este modo:
    - “**top**” determina la distancia entre el borde superior del elemento y el borde superior de la página.
    - “**bottom**” determina la distancia entre el borde inferior del elemento y el borde inferior de la página.
* Basta con indicar una sola especificación en el eje horizontal (“**left**” o “**right**”) y otra sobre el eje vertical (“**top**” o “**bottom**”).

C.- **Posicionamiento absoluto**: El posicionamiento absoluto crea un elemento independiente del resto del documento. Su sintaxis es:

**position: absolute; left: valor ó %;**

**Top: valor ó %;**

**Right: valor ó %;**

**Bottom: valor ó %;**

A tener en cuenta:

* El elemento se retira del flujo normal y se posiciona en el lugar exacto definido por el diseñador.
* La posición se define mediante las coordenadas (**x,y**) donde:
  + **X** es la distancia respecto al borde izquierdo del elemento padre o de la ventana del navegador (eje horizontal). De esta manera:
    - “**left**” determina la distancia entre la parte izquierda del elemento y el borde izquierdo de la página.
    - “**right**” determina la distancia entre la parte derecha del elemento y el borde derecho de la página.
  + **Y** es la distancia respecto al borde superior del elemento padre o de la ventana del navegador (eje vertical). De este modo:
    - “**top**” determina la distancia entre el borde superior del elemento y el borde superior de la página.
    - “**bottom**” determina la distancia entre el borde inferior del elemento y el borde inferior de la página.
* Basta con indicar una sola especificación en el eje horizontal (“**left**” o “**right**”) y otra sobre el eje vertical (“**top**” o “**bottom**”).
* El posicionamiento absoluto tiene el riesgo de no mostrar correctamente la página en resoluciones de pantalla diferentes a la utilizada en tiempo de diseño puesto que los elementos que se hayan posicionado mediante esta propiedad pueden superponerse sobre los elementos del flujo normal de la página.
* Conviene recordar que los navegadores agregan por defecto un margen al cuerpo de la página **HTML** y que este margen por defecto varía de un navegador a otro. Por ello es prudente, e caso de usar posicionamiento absoluto, especificar los márgenes de la etiqueta <**body**>.

D.- **Posicionamiento fijo**: El posicionamiento fijo crea un elemento independiente del que se puede definir su posición exacta en relación a la ventana del navegador. Su sintaxis es:

**position: fixed;**

***Para ver el funcionamiento de esta propiedad realice la práctica 9.20.***

SUPERPOSICIÓN

La propiedad “**z-index**” permite posicionar los elementos unos por encima o por debajo de otros: Su sintaxis es:

**z-index: nº entero positivo;**

A tener en cuenta:

* Si un elemento tiene la propiedad “**z-index: 2;**” aparecerá por delante de otro elemento que tenga la propiedad “**z-index: 1;**”.
* Esta propiedad solo funciona con un posicionamiento absoluto de los elementos.

***Para ilustrar esta propiedad realice la práctica 9.21.***

DESBORDAMIENTO

La propiedad “**overflow**” determina lo que debe hacer el navegador cuando un elemento es más grande que el elemento padre que lo contiene. Su sintaxis es:

**overflow: hidden / scroll / visible / auto;**

A tener en cuenta:

* Con el valor “**hidden**”, la parte que desborda se oculta, sin posibilidad de verla ni acceder a ella.
* Con el valor “**scroll**”, la parte que desborda se oculta, pero aparecen barras de desplazamiento para acceder a ella.
* Con el valor “**visible**”, la parte que desborda se muestra, ignorando las especificaciones del elemento padre que lo contiene.
* Con el valor “**auto**”, se deja la gestión al navegador.

***¡Veamos un ejemplo realizando la práctica 9.22.!***

RECORTE

La propiedad “**clip**” determina la parte visible del elemento, generalmente una imagen. Su sintaxis es:

**clip: rect (sup\_izquierda sup\_derecha inf\_derecha inf\_izquierda / auto;**

A tener en cuenta:

* Las coordenadas del rectángulo recortado vienen dadas por las esquinas superior izquierda, superior derecha, inferior derecha e inferior izquierda respecto de la imagen inicial.
* Se recomienda incluir el elemento que se quiere recortar dentro de una etiqueta <**div**> o <**span**>.
* Solamente funciona con un posicionamiento absoluto del elemento.

***Para ver su funcionamiento realice la práctica 9.23.***

VISIBILIDAD

La propiedad “**visibility**” determina si un elemento se está visible u oculto. Su sintaxis es:

**visibility: visible / hidden;**

A tener en cuenta:

* El valor “**visible**” muestra el elemento.
* El valor “**hidden**” oculta el elemento.
* Si se establece el valor “**hidden**” el elemento ocupa todavía su hueco en el documento y en consecuencia se muestra un rectángulo en blanco donde estuviera situado el elemento. Es decir se conserva el formato de página a pesar de la ausencia, aparente, del elemento, de tal manera que los demás elementos de la página conservan su posición relativa inicial.
* No confundir el valor “**hidden**” con la propiedad “**display: none;**” que elimina por completo el elemento de la página y no se reserva sitio alguno.

***Para ver el efecto que produce esta propiedad realice la práctica 9.24.***

CURSOR DEL RATÓN

La propiedad “**cursor**” permite modificar el cursor del ratón. Su sintaxis es:

**cursor: palabra clave;**

Las distintas palabras clave son:

|  |  |
| --- | --- |
| URL | Especifica un archivo donde se encuentra la imagen que se desea usar como cursor. El archivo de imagen especificado en la URL debe tener el formato .cur o .ani. **Nota:** Se pueden indicar varios cursores separados por comas. En este caso especifique siempre un cursor genérico al final de la lista por si ninguno de los cursores definidos por URL se pueden utilizar. |
| auto | El navegador establece el cursor según el contexto. Es la opción predeterminada. |
| crosshair | Cursor en forma de cruz. |
| default | Cursor por defecto del sistema operativo. |
| e-resize | Cursor apuntando hacia el este. |
| help | El cursor indica que hay ayuda disponible. A menudo se muestra con un signo de interrogación. |
| move | El cursor indica un objeto que se puede trasladar. |
| n-resize | Cursor apuntando hacia el norte. |
| ne-resize | Cursor apuntando hacia el noreste. |
| nw-resize | Cursor apuntando hacia el noroeste. |
| pointer | El cursor presenta un dedo que indica un enlace. |
| progress | El cursor indica que el programa está ocupado, en curso, pero se puede seguir trabajando. Normalmente se muestra una flecha con un reloj de arena. |
| s-resize | Cursor apuntando hacia el sur. |
| se-resize | Cursor apuntando hacia el sureste. |
| sw-resize | Cursor apuntando hacia el suroeste. |
| text | El cursor indica que es posible escribir o seleccionar texto. |
| w-resize | Cursor apuntando hacia el oeste. |
| wait | El cursor indica que el programa está ocupado y debemos esperar. Suele representarse con un reloj de arena o un círculo dando vueltas. |
| not-allowed | El cursor muestra una señal de prohibido. |
| no-drop | El cursor tiene forma de mano con un dedo desplegado y una señal de prohibido. |
| col-resize | Cursor compuesto por dos trazos verticales con una flecha a cada lado. |
| row-resize | Cursor compuesto por trazos horizontales. |

***Para ver el funcionamiento de esta propiedad realice la práctica 9.25.***

COLUMNAS PERIODÍSTICAS

La propiedad “**column-count**” permite presentar contenido en columnas de estilo periodístico. Su sintaxis es:

**-moz-column-count: valor entero que determina el nº de columnas.**

**-moz-column-width: valor del ancho de cada columna. (Es opcional).**

**-moz-column-gap: valor del espacio entre columnas.**

**-moz-column-rule: Define el borde entre las columnas.**

***Para ver su funcionamiento realice la práctica 9.26.***

CAPÍTULO 10.- PSEUDOCLASES EN CSS

¿QUÉ ES UNA PSEUDOCLASE?

Hasta el momento, para declarar los estilos en las hojas de estilo, hemos utilizado sobre todo selectores de etiqueta y en pocas ocasiones selectores de clase, (si es necesario recuerde estos conceptos explicados en el capítulo 4), pero **CSS** nos permite aplicar estilos a elementos de un documento **HTML** que no podemos especificar a través de una etiqueta o de una clase como, por ejemplo, un determinado enlace o la primera letra de un párrafo. Para referenciar esos elementos más específicos, que luego indicaremos, utilizamos pseudoclases.

Una pseudoclase permite tener en cuenta diferentes condiciones o eventos a la hora de definir un estilo para una etiqueta **HTML**.

La sintaxis de las psuedoclases es:

**Selector:pseudoclases {**

**…**

**declaraciones de estilo**

**…**

**}**

Las pseudoclases también se pueden utilizar con las clases. En ese caso su sintaxis sería:

**Selector.clase:pseudoclase {**

**…**

**declaraciones de estilo**

**…**

**}**

¡Veamos y practiquemos con las pseudoclases!

PSEUDOCLASES DE ENLACE

Los enlaces que indicamos con la etiqueta <**a**> pueden tener diferentes estados y podemos asignar estilos diferentes, dependiendo del estado en que se encuentre, utilizando las siguientes pseudoclases:

**A.-** **a:link**. Permite definir la apariencia de un enlace que aún no se ha seleccionado o visitado.

**B.-** **a:visited**. Permite definir la apariencia de enlaces que dirigen a páginas que el usuario ya ha visitado. ***¡Veamos un ejemplo práctico realizando la práctica 10.1!***

**C.-** **a:hover**. Permite definir la apariencia de un enlace en el momento que se pasa el ratón sobre él. ***¡Veamos cómo funciona realizando la práctica 10.2!***

**D.-** **a:active**. Permite definir la apariencia de un enlace en el momento preciso en que se hace clic sobre él. ***Un ejemplo de su uso podría ser como en la práctica 10.3…***

A tener en cuenta:

* La pseudoclase **:active** se utiliza, generalmente, de forma conjunta con **:visited**, **:hover**, y **:link**.
* Para obtener un resultado correcto es indispensable enunciar las apariencias para los enlaces en el orden siguiente: **:link**, **:visited**, **:hover** y **:active**.
* La pseudoclase **:hover** puede utilizarse con otras etiquetas **HTML** además de con los enlaces como ya hemos visto al utilizarlo con la etiqueta <**li**>.

Llegados a este punto, es necesario hacer una reflexión sobre el subrayado de los enlaces. Hasta ahora, en todas nuestras páginas de ejemplos y ejercicios hemos eliminado todos los subrayados de enlaces con la propiedad “**text-decoration: none**”. Debemos tener cuidado con esta práctica ya que los usuarios de internet están acostumbrados al subrayado azul de los enlaces de las páginas web y saben que pueden hacer clic en ellos. Si quitamos dicho subrayado existe la posibilidad de confundir a los usuarios y disminuir la usabilidad de la web. El uso de las pseudoclases de enlace ayudarán en este cometido.

***Realice ahora la primera parte de la ejercitación del tema 10.***

PSEUDOCLASES DE PÁRRAFO

**A.-** **:first-letter**. Permite definir la apariencia de la primera (y solo de la primera) letra de un párrafo. Se logra un efecto de letra capital. ***¡Veamos un ejemplo realizando la práctica 10.4!***

**B.-** **:first-line**. Permite definir la apariencia de la primera (y solo de la primera) línea de un párrafo. ***¡Vayamos al ejemplo realizando la práctica 10.5!***

PSEUDOCLASES PARA INSERTAR CONTENIDO

**A.-** **:before**. Asociada a la propiedad “**content**”, permite insertar un contenido antes de un elemento. ***¡Vayamos a la práctica realizando la 10.6!***

**B.-** **:after**. Asociada a la propiedad “**content**”, permite insertar un contenido después de un elemento. ***¿Lo probamos realizando la práctica 10.7?***

PSEUDOCLASES DE SELECCIÓN

Además de las pseudoclases que hemos utilizado en los pasos anteriores existen otras pseudoclases que nos permiten indicar elementos **HTML** dependiendo de la posición que ocupen dentro de la página web, padres o hijos, y en relación con sus elementos hermanos. Recordar el concepto de “**herencia**” que aprendimos en temas anteriores nos ayudará a entender estas pseudoclases. Las llamadas pseudoclases de selección son:

|  |  |
| --- | --- |
| **:root** | Representa el elemento raíz o el elemento en el nivel más alto del documento. Por ejemplo, en **HTML5**, la etiqueta <**html**>. |
| **:empty** | Corresponde a los elementos vacíos y que no tienen hijos. |
| **:only-child** | Devuelve el hijo único. Este elemento no tiene elementos hermanos. |
| **:first-child** | Selecciona el primer elemento hijo. |
| **:last-child** | Selecciona el último elemento hijo. |
| **:nth-child(n)** | Designa e enésimo elemento hijo, donde n es un número o bien las palabras clave “**even**” (pares) y “**odd**” (impares). |
| **:nth-last-child(n)** | Selecciona el enésimo hijo a partir del último elemento. |
| **:only-of-type** | Devuelve el único elemento de este tipo. |
| **:first-of-type** | Representa el primer elemento de este tipo. |
| **:last-of-type** | Representa el último elemento de este tipo. |
| **:nth-of-type(n)** | Designa al enésimo elemento de este tipo, donde n es un número o bien las palabras claves “**even**” (pares) y “**odd**” (impares). |
| **:nth-last-of-type(n)** | Selecciona el enésimo elemento de este tipo partiendo del último elemento. |
| **not()** | Selecciona todos los elementos excepto el indicado entre paréntesis. |

***¡Veamos algunos ejemplos en la práctica 10.8!***

SELECTORES DE ATRIBUTOS

Ya conocemos que las etiquetas en **HTML** pueden tener atributos tales como: “**alt**”, “**title**”, “**href**”, etc… Pues bien los llamados “**selectores de atributo**”, aprovechan dichos atributos para convertirlos en selectores **CSS** y así poder aplicar un estilo a los elementos que tienen dichos atributos. Esto nos librará de poner clases e identificadores a muchas etiquetas **HTML**.

Los selectores de atributos son:

|  |  |
| --- | --- |
| **elemento[atributo]** | Los estilos se aplicarán a todas las etiquetas que tengan ese atributo. |
| **elemento[atributo=”valor”]** | Los estilos se aplicarán a todas las etiquetas que tengan ese atributo con ese “**valor**”. |
| **elemento[atributo~=”valor”]** | Los estilos se aplicarán a las etiquetas que tengan ese atributo y que uno de sus valores separados por espacios sea el “**valor**”. |
| **elemento[atributo|=”valor”]** | Selecciona los elementos que tienen establecido un atributo llamado “**atributo**” que contenga una lista de valores separados por guiones, pero que comienza por “**valor**”. |
| **elemento[atributo^=”valor”]** | Los estilos se aplicarán a todos los elementos que tengan ese atributo y cuyo valor comienza por la cadena de texto indicada en “**valor**”. |
| **elemento[atributo$=”valor”]** | Los estilos se aplicarán a todos los elementos que tengan ese atributo y cuyo valor termina por la cadena de texto indicada en “**valor**”. |
| **elemento[atributo\*=”valor”]** | Los estilos se aplicarán a todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada en “**valor**”. |

***Para ver un ejemplo de este tipo de selectores realice la práctica 10.9***

***A continuación realice la segunda parte de la ejercitación del capítulo 10.***

CAPÍTULO 11.- FORMULARIOS CON HTML

Los formularios ocupan un lugar relevante en el diseño y explotación de una aplicación o un sitio web. Nos sirven para recibir de vuelta la información que proviene directamente de los visitantes de nuestra web.

**HTML5** ha supuesto enormes mejoras en el tratamiento de los formularios convirtiendo en sencillas tareas que antes necesitaban de lenguajes de programación como **JavaScript**.

El propósito último de los formularios es el procesamiento automático de los datos escritos por los usuarios. Para esta tarea será preciso usar lenguajes de programación de gestión de bases de datos como **PHP** o **MySQL** que estudiará más adelante en nuestro curso de “**Creación y Edición de Páginas Web**”.

FORMULARIOS: DECLARAR UN FORMULARIO

La etiqueta <**form**> no muestra nada en la web. Tiene como única función indicar al navegador que a continuación se va a mostrar un formulario. Engloba el resto de etiquetas y elementos que constituyen el formulario.

Los atributos de la etiqueta <**form**> son:

* **Name**. Asigna un nombre al formulario.
* **Action**. Le indica al navegador la acción que vamos a realizar con los datos introducidos en el formulario. Para el **W3C** la presencia de este atributo es obligatoria. Pueden ser:
  + Llamar a un programa de procesamiento de datos como veremos en el curso del lenguaje **PHP**. Por ejemplo la etiqueta podría ser como: **<form action=”http://servidor/datos.php”>**.
  + Llamar al programa cliente de correo instalado en el ordenador del usuario para enviar los datos recogidos en el formulario a un correo electrónico. En este caso la etiqueta quedaría como: **<form action=”mailto:info@academiafrayluis.es”>**.
  + Procesar internamente los datos del formulario mediante **JavaScript**. En este caso el atributo “**action**” quedaría vacío como: **<form action=””>**.
* **Enctype**. Especifica el formato informático, (mime type), en que se transmitirán los datos del formulario.
* **Method**. Especifica el método de transmisión de los datos recogidos en el formulario. Puede ser **GET** o **POST**. En la práctica se utiliza más este último.

***¡Vayamos a la práctica realizando la 11.1!***

FORMULARIOS: CAMPO DE TEXTO

La etiqueta <**input**> genera un campo de texto que permite introducir datos tanto alfabéticos como numéricos. No tiene etiqueta de cierre aunque se acepta la sintaxis de **XHTML** <**input /**>.

Los atributos de la etiqueta <**input**> son:

* **Type**. Especifica el tipo de dato que se va a almacenar en el campo de texto. Este atributo tiene bastantes valores que estudiaremos en este capítulo.
* **Name**. Asigna un nombre único para este elemento.
* **Size**. Indica el número de caracteres visibles del campo de texto y por tanto la longitud del campo. Su valor por defecto es 20 pero, aunque no se vean, el usuario puede introducir tantos caracteres como quiera.
* **Maxlength**. Determina el número máximo de caracteres que el usuario puede introducir en el campo.
* **Value**. Indica el valor por defecto del campo texto. Aparece como valor del campo de texto cuando se carga la página.
* **Readonly**. Indica que el usuario no puede modificar el valor por defecto definido en el atributo anterior.
* **Placeholder**. Permite incluir una sugerencia acerca del valor que se debe introducir en el campo. Esta sugerencia aparece de color gris en el campo tras cargar la página y desaparece cuando el usuario sitúa el foco en el elemento afectado.
* **Autofocus**. Sitúa el foco sobre el campo una vez cargada la página.
* **Required**. Convierte al campo en obligatorio para poder enviar los datos recogidos en el formulario. El navegador muestra un bocadillo indicando que el campo se debe rellenar.
* **Pattern**. Se utiliza junto con JavaScript.
* **Height**. Indica la altura del campo en píxeles o porcentaje. Se puede reemplazar por la propiedad **CSS** que ya conocemos.
* **Width**. Indica la anchura del campo en píxeles o porcentaje. Se puede reemplazar por la propiedad **CSS** que ya conocemos.
* **Autocomplete**. Especifica si se activa o no la función autocompletar en el campo. Cuando esta función está activa el navegador recuerda lo introducido en el mismo campo anteriormente y lo ofrece mediante un menú desplegable ahorrando al usuario el tener que escribirlo de nuevo.

FORMULARIOS: ETIQUETADO DE LOS CAMPOS.

La etiqueta <**label**> asocia a un campo de un formulario un título. El funcionamiento de esta etiqueta es el siguiente:

**A.-** En primer lugar el texto asignado como título de un campo de formulario debe situarse entre las etiquetas <**label**> … <**/label**>. Por ejemplo:

**<label>Nombre</label>**

**B.-** Luego hay que asociar esta etiqueta <**label**> a su correspondiente control. Para ello el campo del formulario debe definirse mediante un identificador de tipo “**id**”.

**<label>Nombre</label>**

**<input type=”text” id=”nombre1”>**

**C.-** El atributo “**for=”…”**” dentro de la etiqueta <**label**> permite asociar directamente la etiqueta al campo haciendo referencia a ese identificador.

**<label for=”nombre1”>Nombre</label>**

**<input type=”text” id=”nombre1”>**

¡Vamos a crear nuestro primer campo del formulario! ***Para ello realice la práctica 11.2.***

CAMPO DE TEXTO PARA DIRECCION DE CORREO ELECTRÓNICO

Utilizando el atributo “**type=”email”**” en la etiqueta <**input**> podemos pedir la dirección de correo electrónico al usuario de nuestra web. El navegador se encarga de comprobar o validar que el usuario introduzca correctamente, con su formato correspondiente, una dirección de correo electrónico. ***¡Veámoslo en la práctica 11.3!***

CAMPO DE TEXTO PARA DIRECCION WEB

Si lo que queremos es pedir una dirección web utilizaremos el atributo “**type=”url”**”. El navegador también se encarga de validar que la dirección web introducida sea correcta***. ¡Vayamos a la práctica 11.4!***

BOTON DE ENVÍO

Hasta ahora hemos recogido los datos de nuestro visitante de nombre, apellidos, e-mail y web ¿Pero cómo le decimos al navegador que nos envíe los datos o que realice la acción que hemos indicado en la etiqueta <**form**>? Para eso tenemos el atributo “**type=”submit”**”. ***¡Veámoslo en la práctica 11.5!***

CAMPO DE TEXTO DE BÚSQUEDA

Es muy habitual, en muchas de las páginas web que visitamos, encontrarnos con campos de búsqueda. Para ello podemos utilizar el atributo “**type=”search”**” ***¡Tal como lo puede comprobar en la práctica 11.6!***

CAMPOS DE TRANSFERENCIA DE ARCHIVOS

La etiqueta <**input type=file**> permite transferir un archivo desde el ordenador del usuario que visita nuestra web hasta otro ordenador de tipo servidor. ***¡Vamos a probarlo realizando la práctica 11.7!***

A tener en cuenta:

* Por supuesto pueden utilizarse los atributos habituales de la etiqueta <**input**>, pero hay que reseñar que el atributo “**maxlength**” permite fijar el tamaño máximo del archivo a transferir.
* Como hemos visto en la declaración de la etiqueta <**form**>, es preciso usar los atributos “**method=”post”**” y “**enctype=”multipart/form-data”**” según el formato correcto del archivo.
* Existe también el atributo “**accept**” que permite limitar la transferencia a ciertos tipos de archivos concretos.
* Observe que esta etiqueta no sirve más que para seleccionar el archivo que se quiere enviar. Para enviarlo realmente se necesita un programa en **PHP** u otro lenguaje de programación.

CAMPOS DE CONTRASEÑA

Se trata de un simple campo de texto, <**input type=”password”**>, en donde el texto introducido se sustituye, en la visualización, por puntos o asteriscos.

Este campo de contraseña no protege los datos ya que se transmiten sin cifrar. Únicamente protege de las personas que pudieran estar mirando la pantalla en el momento de escribir la contraseña. ***¡Veamos su funcionamiento en la práctica 11.8!***

CAMPOS OCULTOS

Este tipo de campo, <**input type=”hidden”**>, permite capturar datos que no estarán visibles al visitante de la página. Se utilizan junto con programas en **JavaScript** o **PHP** entre otros. Si escribimos su código veremos que no aparecen en la web.

***¡Compruébelo realizando la práctica 11.9!***

BOTON DE ANULACION

Este campo, <**input type=”reset”**>, genera un botón que borra todo lo que hayamos escrito en los diferentes campos del formulario. ***Como siempre, veámoslo en la práctica 11.10.***

Como en otros casos, el texto “**Restablecer**” que muestra el botón, lo selecciona automáticamente el navegador, pero es posible cambiarlo utilizando el atributo “**value**” de la etiqueta <**input**>.

CAMPO DE TEXTO CON FORMATO NUMÉRICO

La etiqueta <**input type=”number”**> genera un campo de tipo numérico semejante a los controles numéricos típicos de Windows. Tiene el siguiente formato:

**<input type="number" name="q" min="2" max="10" step="2" value="2" />**

Los atributos:

* **Min**. Indica el valor mínimo del contador.
* **Max**. Indica el valor máximo del contador.
* **Step**. Determina el incremento del contador cada vez que se hace clic en las flechitas.
* **Value**. Indica el valor inicial del contador.

Esta etiqueta solo es reconocida por ahora en el navegador “**Opera**”. El resto de navegadores no la tienen en cuenta y simplemente incluyen un campo de texto normal y corriente que debe completarse manualmente.

CAMPO DE TEXTO CON FORMATO DE FECHA

Al igual que en el caso anterior las etiquetas <**input**>, “**type=”date”**”, “**type=”month”**”, “**type=”week”**” y “**type=”time”**” solo son reconocidas por el navegador “**Opera**” y muestran los típicos recuadros de fechas para seleccionar un día determinado que podemos encontrar por ejemplo en las páginas webs de hoteles o reservas de vuelos.

CAMPO DE TEXTO DE COLOR

Y la etiqueta <**input type=”color”**> permitirá escoger un color. A día de hoy no está aún implementada en ningún navegador.

CAMPO DE RANGO

La etiqueta <**input type=”range”**> permite crear controles deslizantes iguales a los de Windows. Tiene el siguiente formato:

**<input type="range" name=”deslizate" min="1" max="5" step="1" value="3" />**

Los atributos:

* **Min**. Indica el valor mínimo del contador.
* **Max**. Indica el valor máximo del contador.
* **Step**. Determina el paso de avance al deslizar la pestaña.
* **Value**. Indica el valor inicial del contador.

***Para ver su funcionamiento realice la práctica 11.11.***

AREA DE TEXTO

En ocasiones es necesario proveer de un campo con suficiente espacio para que el usuario pueda escribir con más libertad. Es el caso de campos para comentarios, sugerencias, etc. En estos casos se utiliza la etiqueta <**textarea**> con su correspondiente cierre <**/textarea**>.

Los atributos de la etiqueta <**textarea**> son:

* **Name**. Asigna un nombre para este elemento.
* **Maxlength**. Determina el número máximo de caracteres que el usuario puede introducir en el área de texto.
* **Cols**. Define el número de caracteres visibles en anchura para el área de texto. Puede reemplazarse por la propiedad **CSS** “**width**”.
* **Rows**. Define el número de caracteres visibles en altura para el área de texto. Puede reemplazarse por la propiedad **CSS** “**height**”.
* **Readonly**. Indica que el usuario no puede modificar el valor por defecto definido en el área de texto.
* **Placeholder**. Permite incluir una sugerencia acerca del valor que se debe introducir en el campo. Esta sugerencia aparece de color gris en el campo tras cargar la página y desaparece cuando el usuario sitúa el foco en el elemento afectado.
* **Autofocus**. Sitúa el foco sobre el campo una vez cargada la página.
* **Required**. Convierte al campo en obligatorio.
* **Wrap**. Especifica la forma en que se gestionan los saltos automáticos de la línea de texto a la hora de enviar el contenido. Con **wrap=”hard”** se inserta un carácter de salto de línea junto con el texto. Con **wrap=”soft”**, que es el valor por defecto, no se inserta ningún carácter de salto de línea a la hora de transmitir la información.

Es posible indicar un valor por defecto en el área de texto pero no mediante el atributo “**value**” como en otros tipos de campos sino directamente escribiendo el texto entre las etiquetas <**textarea**>.

***¡Veámoslo en la práctica 11.12!***

BOTONES DE SELECCIÓN ÚNICA – BOTONES DE OPCIÓN

También conocidos como botones de radio se pueden mostrar mediante la etiqueta <**input type=”radio”**>. Tienen la particularidad de que son mutuamente excluyentes entre sí. Dicho de otro modo solo puede estar uno seleccionado. Tiene los siguientes atributos:

* **Name**. Asigna un nombre para este elemento. Es obligatorio. Debe ser igual para todos los botones de radio.
* **Checked**. Permite preseleccionar un botón de radio.
* **Value**. Permite atribuir un valor a cada botón de radio lo que permitirá realizar algún procesamiento posterior.

***¡Veamos cómo funcionan realizando la práctica 11.13!***

BOTONES DE SELECCIÓN MÚLTIPLE – CASILLAS DE VERIFICACIÓN

Si lo que queremos es seleccionar varias opciones a la vez utilizaremos los campos de tipo **checkbox**, <**input type=”checkbox”**>, conocidas en el entorno Windows como “casillas de verificación”. Tiene los siguientes atributos:

* **Name**. Asigna un nombre para este elemento. Es obligatorio. No es necesario que el nombre sea el mismo para todas las casillas. Si más adelante queremos utilizar las casillas en un programa de **PHP** o **JavaScript** tendremos que dar un nombre diferente a cada casilla.
* **Checked**. Permite preseleccionar una casilla.
* **Value**. Permite atribuir un valor a cada casilla lo que permitirá realizar algún procesamiento posterior.

***¡Veamos su funcionamiento realizando la práctica 11.14!***

LISTA DESPLEGABLE

La etiqueta <**select**> indica al navegador el uso de una lista desplegable. Los elementos de la lista se introducen mediante etiquetas <**option**>. Sus atributos son los siguientes:

* **Name**. Define un nombre para la lista desplegable.
* **Size**. Define el nº de elementos de la lista que serán visibles.
* **Multiple**. Habitualmente, el usuario no puede seleccionar más de un elemento de la lista desplegable. Este atributo permite seleccionar más de un elemento a la vez manteniendo pulsada la tecla “**Ctrl**”.
* **Selected**. En una lista desplegable, por defecto aparece seleccionado el primer elemento. Podemos conseguir que sea otro indicando este atributo en la etiqueta <**option**> correspondiente.
* **Value**. Permite atribuir un valor a cada elemento de la lista para realizar algún procesamiento posterior. Si no se utiliza este atributo se recogerá el texto situado tras la etiqueta <**option**>.

***¡Practiquémoslo realizando la práctica 11.15!***

CAMPO DE TEXTO CON LISTA DE SUGERENCIAS

La etiqueta <**datalist**>, agregada a un campo de texto, muestra una lista de sugerencias para facilitar la introducción del texto al usuario. El efecto es el mismo que cuando realizamos una búsqueda en Google. Además, a medida que el usuario escribe letras en el campo de texto, las sugerencias van reduciéndose a aquellas que contienen las letras que el usuario escribe.

La etiqueta <**datalist**> contiene una lista de etiquetas <**option**> con las sugerencias. Estas etiquetas <**option**> deben poseer un atributo “**value**”. ***¡Veámoslo en la práctica 11.16!***

Observe que la etiqueta <**datalist**> está ligada al campo de texto del formulario mediante un identificador “**id**” que se asocia con el atributo “**list**” de aquél.

BOTON DE ACCIÓN – BUTTON

Llegados a este punto nos parece oportuno hacer referencia a la etiqueta <**button**>. Esta etiqueta muestra un botón que al hacer clic sobre él permite ejecutar una acción específica generalmente mediante código **JavaScript**.

Al tener una etiqueta de apertura y otra de cierre permite incluir texto, imágenes u otro contenido **HTML**.

Esta etiqueta no tiene por qué estar anidada a un formulario pudiéndose usar en múltiples contextos.

Sus atributos son los siguientes:

* **Name**. Define un nombre para el botón.
* **Type**. Este atributo admite tres argumentos:
  + **Button:** se utiliza cuando el botón inicia un script. Es el valor por defecto del atributo en la mayoría de los navegadores.
  + **Submit:** para enviar el formulario (en caso de utilizar la etiqueta dentro de un elemento formulario).
  + **Reset:** para reinicializar el formulario (en caso de utilizar la etiqueta dentro de un elemento formulario).
* **Disabled**. Permite desactivar el botón cuando se carga la página. En este caso el botón aparece sombreado en gris. Es posible activarlo (enabled) con un script mediante una acción del usuario, por ejemplo después de haber aceptado las condiciones de uso.

***Un ejemplo del uso de <button> podría ser el que puede ver en la práctica 11.17:***

ORGANIZACIÓN DE CAMPOS DE FORMULARIO

En el caso de formularios largos y complejos, en ocasiones resulta útil reagrupar gráficamente ciertos elementos para organizar la página de forma lógica. Las etiquetas <**fieldset**> y <**legend**> permiten mejorar sensiblemente la ergonomía y la usabilidad de los formularios.

La etiqueta <**fieldset**> engloba los campos de formulario que usted determine. Estos campos se muestran en la pantalla rodeados por un borde.

La etiqueta <**legend**> situada justo detrás de la etiqueta <**fieldset**>, permite agregar una leyenda a modo de título de los campos que hemos agrupado.

***Aplique estas etiquetas a nuestra web realizando la práctica 11.18.***

***Para terminar de dominar este tema realice ahora la ejercitación del capítulo 11.***

CAPÍTULO 12.- TABLAS EN HTML y CSS

Las tablas en **HTML** utilizan los mismos conceptos de filas, columnas, cabeceras y títulos que se utilizan en cualquier otro entorno de publicación de documentos.

Hasta hace poco las tablas en **HTML** se utilizaban para maquetar o estructurar la disposición de las páginas web colocando los textos, las imágenes y demás contenidos dentro de celdas de tablas que a su vez estaban dentro de otras tablas y así sucesivamente hasta conseguir la presentación web deseada. Ahora con **CSS**, esta técnica ya no se utiliza y las tablas vuelven a su función: presentar la informacion, especialmente la numérica.

CREACIÓN DE UNA TABLA

Para crear tablas en **HTML** se utilizan básicamente estas tres etiquetas: <**table**> que indica el comienzo de una tabla, que se compone de filas <**tr**>, que contienen celdas <**td**>. Los pasos a seguir son:

* Se declara la apertura de la tabla con la etiqueta <**table**>.
* Esta tabla se componen de una primera fila, introducida por la etiqueta <**tr**>.
* Esta fila se compone de una primera celda, indicada con la etiqueta <**td**>.
* Escribimos el contenido de la primera celda.
* Se cierra la primera celda con la etiqueta <**/td**>.
* Abrimos la segunda celda y sucesivas con <**td**> y <**/td**>.
* Indicamos la el fin de la primera fila con la etiqueta <**/tr**>.
* Comenzamos la segunda fila con una nueva etiqueta <**tr**>.
* Especificamos las celdas contenidas en esta fila con sus correspondientes <**td**> y <**/td**>.
* Indicamos el fin de la segunda fila con la etiqueta <**/tr**>.
* Por último, después de todas las filas y celdas, cerramos la tabla con <**/table**>.

***¡Veamoslo en la práctica 12.1!***

Observe que al definir una tabla, se debe pensar en primer lugar en las filas que la forman y a continuación en las columnas. El motivo es que **HTML** procesa primero las filas y por eso las etiquetas <**tr**> aparecen antes que las etiquetas <**td**>.

Para dar formato a las tablas utilizaremos las propiedades **CSS** como ya conocemos. ***Compruébelo realizando la práctica 12.2.***

CELDAS DE LA TABLA

Las celdas pueden contener cualquier elemento **HTML**, bien sea un texto, imágenes, enlaces, fondos o incluso otras tablas. Y por supuesto, pueden tener su propio formato.

***Compruébelo realizando la práctica 12.3.***

COMBINAR CELDAS

Es posible combinar celdas tanto horizontal como verticalmente:

* Para **COMBINAR CELDAS HORIZONTALMENTE** aplicaremos a la etiqueta **HTML** <**td**>, el atributo “**colspan=”x”**”, donde “**x**” indica el nº de columnas a combinar. ***¡Veámoslo en un ejemplo realizando la práctica 12.4!***
* De igual forma **PARA COMBINAR VARIAS CELDAS EN VERTICAL** utilizaremos en la etiqueta **HTML** <**td**>, el atributo “**rowspan=”x”**”, donde “**x**” indica el nº de filas a combinar.

Resumiendo, para no tener lío, tenga en cuenta que en **HTML** todas las tablas deben disponer de una estructura regular, es decir, todas las filas de una tabla **HTML** deben tener el mismo número de filas y columnas. Para mostrar menos columnas en una fila se fusionan mediante el atributo “**colspan**”. Y si se quiere fusionar filas utilizaremos el atributo “**rowspan**”.

CELDAS DE ENCABEZADO

En una tabla podemos indicar al navegador las celdas que constituyen el encabezado o título de cada columna mediante la etiqueta <**th**> y su correspondiente de cierre <**/th**>.

Esta etiqueta es de bastante utilidad para los lectores de pantalla usados por personas invidentes. ***¡Veamos su funcionamiento realizando la prática 2.5!***

LEYENDA DE LA TABLA

La etiquea <**caption**> asocia un título o leyenda a una tabla. Debe situarse justo después de la etiqueta de apertura <**table**> y debe aparecer una sola vez en la tabla. ¡Apliquémoslo en la práctica 12.6!

Por defecto, la etiqueta <**caption**> muestra la leyenda de la tabla de forma centrada y en la parte superior de la tabla. Cada tabla solo puede contener una etiqueta <**caption**>.

Como siempre podemos cambiar esto aplicando estilos **CSS** con la propiedad “**caption-side**” que tiene dos valores: “**top**” y “**bottom**”.

¿Sería capaz de conseguir que la leyenda de la tabla aparezca debajo de la tabla? ¡Pruébelo! Cuando lo haya probado vuelva a poner la leyenda encima de la tabla.

AGRUPAR COLUMNAS

La etiqueta <**colgroup**> se usa para agrupar **columnas** con el objetivo de aplicar el mismo formato al conjunto de etiquetas del grupo mediante hojas de estilo **CSS**.

El atributo “**span=”x”**” determina el nº de columnas agrupadas. Por defecto el valor de “**x**” es 1.

La etiqueta <**colgroup**> se posiciona justo tras la etiqueta <**table**> o justo tras la etiqueta <**caption**> si se está usando, y delante de cualquier etiqueta <**tr**> o <**td**>. ***¡Veamos un ejemplo realizando la práctica 12.7!***

Semejante a la etiqueta <**colgroup**>, **HTML** también define la etiqueta <**col**> que se utiliza para asignar las mismas propiedades a varias columnas de forma simultánea. La diferencia es que la etiqueta <**col**> no agrupa columnas, sino que sólo asigna estilos comunes a varias columnas.

También tiene el atributo “**span=”x”**” que determina el nº de columnas a las que se les asignarán las propiedades.

Esta etiqueta no tiene su correspondiente de cierre, por lo que utillizaremos la notación de **XHTML**: <**col /**>. Se puede utilizar dentro o fuera de la etiqueta <**colgroup**>. ***Veamos un ejemplo realizando la práctica 12.8.***

ESTRUCTURA LÓGICA DE UNA TABLA

Además de filas y celdas de datos, en tablas más complejas, podemos indicar una sección de cabecera de tabla, una sección de pie de tabla y varias secciones de datos. **HTML** proporciona las siguientes etiquetas para estructurar el contenido de una tabla de una forma lógica:

* <**thead**> indica o agrupa a las celdas consideradas como cabecera de tabla.
* <**tbody**> indica o agrupa el cuerpo de la tabla, es decir el conjunto de datos de la misma.
* <**tfoot**> indica o agrupa a las celdas consideradas como pie de tabla.

Cada tabla puede contener solamente una cabecera y un pie, pero puede incluir un número ilimitado de secciones de datos.

Si en una tabla definimos una cabecera y/o un pie, las etiquetas <**thead**> y/o <**tfoot**> deben colocarse inmediatamente antes que cualquier etiqueta <**tbody**> de modo que, al dibujar la tabla, el navegador pueda preveer el pie de página antes de la recepción de las filas con las celdas de datos. Si la etiqueta <**tfoot**> aparece después de un elemento <**tbody**>, la página no se considera válida.

Los navegadores pueden usar estas etiquetas para habilitar el desplazamiento del cuerpo de la tabla independientemente de la cabecera y el pie. Además, al imprimir una tabla grande que abarque varias páginas, estos elementos pueden permitir que la cabecera y pie de la tabla se imprima en la parte superior e inferior de cada página.

Por otra parte la etiqueta <**tbody**> es indispensable para acceder al contenido de una tabla usando **JavaScript** por lo que conviene acostumbrarnos a incluirla en todas las tablas que creemos por muy sencillas que sean.

PROPIEDADES CSS PARA APLICAR A TABLAS

Además de la propiedad ya vista, “**caption-side**”, **CSS** dispone de las siguientes propiedades para modificar el aspecto de las tablas:

**Border-collapse**

La propiedad “**Border-collapse**” indica como muestra el navegador los bordes en las tablas. Su sintaxis es:

**border-collapse: collapse / separate;**

A tener en cuenta:

* **Collapse** fusiona los bordes de las celdas adyacentes y muestra un solo borde.
* **Separate** muestra los bordes de las celdas de la tabla de manera separada, es decir, fuerza a que cada celda muestre sus cuatro bordes. Es la opción por defecto.

Generalmente, los diseñadores prefieren el modelo collapse porque es el más parecido a como tradicionalmente conocemos las tablas en Word.

En realidad los bordes de la tabla están separados pero con un espacio entre los bordes de 0px. De ahí que se muestren los bordes con una anchura doble, ya que en realidad se trata de dos bordes iguales sin separación entre ellos.

**Border-spacing**

La propiedad “**Border-spacing**” indica al navegador el espacio que tiene que dejar entre los bordes de las celdas. Su sintaxis es:

**border-spacing: px / pt / %;**

A tener en cuenta:

* Si solamente se indica como valor una medida, se asigna ese valor como separación horizontal y vertical. Si se indican dos medidas, la primera es la separación horizontal y la segunda es la separación vertical entre los bordes de las celdas.
* Lógicamente esta propiedad solo funciona si previamente hemos indicado el valor “**separate**” de la propiedad “**border-collapse**”.

**Empty-cells**

La propiedad “**empty-cells**” determina el comportamiento del navegador cuando se encuentra con una celda vacía en una tabla. Su sintaxis es:

**empty-cells: show / hide;**

A tener en cuenta:

* El valor **Show** muestra la celda vacía. Los bordes y el fondo también son visibles.
* El valor **Hide** no muestra la celda vacía ni los bordes, ni el fondo.
* Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es “**separate**”.

***¡Comprobémoslo con un ejemplo realizando la práctica 12.9!***

**Table-layout**

La propiedad “table-layout” indica el algoritmo que debe utilizar el navegador para mostrar la tabla. Su sintaxis es:

**table-layout: fixed / auto;**

A tener en cuenta:

* El valor **fixed** le indica al navegador que calcule una anchura igual y proporcional para las celdas de la tabla.
* Con el valor **auto** la anchura de las columnas se adapta a su contenido. El navegador debe disponer de todos los contenidos de la tabla entera para poder aplicar un cálculo complejo que determina la anchura de cada columna. Por lo tanto la carga de la página es más lenta.

CAPÍTULO 13.- MAS SOBRE ASPECTOS VISUALES Y ORGANIZATIVOS DE LA WEB.

MÁS SOBRE LA PROPIEDAD CSS DISPLAY

Hemos utilizado anteriormente la propiedad **CSS** “**display**” con alguno de sus valores. Ahora que ya sabemos mucho de **HTML** y **CSS** podemos profundizar más en ella para entender mejor su funcionamiento, conocer todo su alcance, su potencial para el diseño y el posicionamiento de los diferentes elementos en nuestra web y todos sus valores.

En **HTML** todos los elementos son una caja que contiene otros elementos y sobre los que podemos aplicar unos estilos **CSS**. En apartados anteriores aprendimos como calcular la dimensión total de una caja teniendo en cuenta su anchura, su “**padding**” y su “**margin**” y como colocarla en la web indicando que se comportara como un elemento en línea, en bloque o como las celdas de una tabla.

Pues bien, la propiedad “**display**” define el **tipo de caja** que genera un elemento **HTML,** el posicionamiento y espacio que ocupa y el comportamiento con respecto a los elementos adyacentes. Su sintaxis es:

**display: inline / block / list-item / run-in / inline-block / table / inline-table / table-row-group**

**/ table-header-group / table-footer-group / table-row / table-column-group**

**/ table-clomn / table-cell / table-caption / flex / inline-flex / none;**

A tener en cuenta:

* **Display: block**; Indica que el elemento se comporte como un bloque. En este caso:
  + El elemento no permitirá la ubicación de elementos adyacentes.
  + Si no se especifican sus dimensiones, el elemento ocupará todo el ancho disponible dentro de su contenedor. Su altura se determinará por su contenido.
  + Si se especifica un ancho, el elemento ocupará solo la extensión determinada pero sin permitir elementos adyacentes.
* **Display: inline**; Indica que el elemento **HTML** se comporte como un elemento en línea. En este caso:
  + El elemento permitirá la ubicación de elementos adyacentes.
  + No se podrá especificar las dimensiones del elemento, ya que como elementos en línea, sus dimensiones son determinadas por el contenido.
* **Display: none**; Indica al elemento **HTML** que no genere ninguna caja con lo que el elemento simplemente desaparece igual que si estuviera ausente del código fuente. En este caso:
  + El elemento no ocupa ningún espacio.
  + Los elementos adyacentes ignoran su presencia y ocupan su espacio.
  + Sus elementos descendientes, en caso de que los tenga, también desaparecen.
* **Display: inline-block**; Indica al elemento **HTML** que se comporte como un elemento en bloque pero permitiendo elementos adyacentes como si fuera un elemento en línea. En este caso:
  + Se les puede asignar las propiedades de los elementos de texto.
  + Se puede determinar sus dimensiones como los elementos en bloque.
  + Si no se especifica su anchura, ocupan todo el espacio disponible como los elementos en bloque.
  + La posición que ocupan estos elementos, es similar a los elementos en línea, lo que quiere decir que la parte inferior del elemento se apoyará en la base de la línea.
* **Display: list-item**; Indica al elemento **HTML** que se comporte como un ítem de lista, es decir, que se comporte exactamente como si se tratara de un elemento <**li**>. En este caso:
  + No permite la presencia de elementos adyacentes como cualquier <**li**>.
  + Pueden especificarse todas sus dimensiones.
  + A este elemento se le pueden aplicar todas las propiedades **CSS** de listas que ya conocemos: “**list-style-type**”, “**list-style-image**”, etc, etc…
* **Display: run-in**; Indica al elemento **HTML** que se comporte de forma diferente dependiendo de los elementos adyacentes. Cuando un elemento cuya propiedad “**display**” tiene el valor “**run-in**” va seguido de un elemento cuya propiedad “**display**” tiene como valor “**block**”, el elemento se comporta como si el valor fuera “**inline**” y se ubica **dentro** del elemento con “**display: block**”. Para todos los demás casos, el elemento marcado con este valor **se comporta como un elemento de bloque**.

Este valor, en la actualidad, solo es reconocido por algunos navegadores como **Ópera** y **Chrome**. Quizás sea por esta circunstancia el que no sea muy utilizado.

* **Display: table**; Indica al elemento **HTML** que se comporte como una tabla.
* **Display: table-row**; Indica al elemento **HTML** que se comporte como una fila de una tabla.
* **Display: table-row-group**; Indica al elemento **HTML** que se comporte como un grupo de filas de una tabla.
* **Display: table-cell**; Indica al elemento **HTML** que se comporte como las celdas de una tabla.
* **Display: Inline-table**; Indica al elemento **HTML** que se comporte como una tabla pero que se mantenga como un elemento en línea, sin salto de línea anterior ni posterior.
* **Display: table-column**; Indica al elemento **HTML** que se comporte como una columna de una tabla.
* **Display: table-column-group**; Indica al elemento **HTML** que se comporte como un grupo de columnas de una tabla.
* **Display: table-header-group**; Indica al elemento **HTML** que se comporte como un grupo de filas de encabezado de tabla.
* **Display: table-footer-group**; Indica al elemento **HTML** que se comporte como un grupo de filas de pie de tabla.
* **Display: table-caption**; Indica al elemento **HTML** que se comporte como si se tratara del título de una tabla.

Todos estos valores de la propiedad “**display**” nos permiten manejar distintos elementos de nuestra web y conseguir que se coloquen como si de una tabla se tratara. Volvemos a recordar aquí que la recomendación es usar las tablas de **HTML** con el propósito de mostrar datos tabulados a semejanza de una hoja de cálculo. Mientras que estos valores de la propiedad “**display**” se utilizarán para mejorar la presentación de los distintos elementos dentro de la página.

* **Display: flex**; Crea un elemento **HTML** flexbox de bloque. Profundizaremos más en este modelo de caja proximamente.
* **Display: Inline-flex**; Crea un elemento **HTML** flexbox en línea. Profundizaremos más en este modelo de caja proximamente.

Para ilustrar el funcionamiento de esta potente propiedad practicaremos sobre una web sencilla que nos permita distinguir sus diferentes valores. ***Para ello realice la práctica 13.1. y la primera ejercitación del capítulo 13.***

COMPATIBILIDAD DE LOS NAVEGADORES

Hasta ahora hemos venido trabajando con el navegador Firefox. Pero este no es el único naveagador que pueden utilizar los visitantes de nuestra web. ¿No se ha preguntado cómo se ven en otros navegadores las páginas web que hemos ido creando en nuestras prácticas?

Aunque hemos hablado brevemente sobre ello a lo largo del curso, en este apartado vamos a profundizar sobre la compatibilidad de los diferentes navegadores con la normativa **HTML5** y **CSS3**. Esta tecnología punta sólo se tiene en cuenta en las últimas versiones de los navegadores. Por tanto, un problema con el que se encuentra el diseñador web, es que no todos los navegadores soportan con la misma eficacia las especificaciones de la **W3C** y debe tener en cuenta esta realidad al crear sus webs.

En la actualidad un diseñador web se puede encontrar, entre otros, con los siguientes navegadores:

* **Internet Explorer**. Aplica **HTML5** y **CSS3** a partir de su versión 9. Pero con una implementación muy pobre. Además, un problema añadido es que un buen número de internautas aún utiliza versiones mucho más antiguas, como Internet Explorer 6, que para nada soportan estas nuevas especificaciones. Utiliza el motor de renderizado “**Trident**”.
* **Firefox**. Es uno de los navegadores preferidos por los diseñadores web por sus extensiones que facilitan el desarrollo de aplicaciones web. Es además uno de los que más han integrado **HTML5** y **CSS3**. Utiliza el motor de renderizado “**Gecko**”. (A la hora de realizar este manual ha anunciado que adoptará, en colaboración con la empresa Samsung, un nuevo motor de renderizado llamado “**Servo**”.)
* **Chrome**. Es ya el tercer navegador más usado en la red. Es moderno, dinámico y soporta prácticamente todas las especificaciones **HTML5** y **CSS3**. Utiliza el motor de renderizado “**Webkit**”. (A la hora de realizar este manual Chrome ha anunciado que abandona el motor de renderizado “**Webkit**” y adoptará uno nuevo llamado “**Blink**”.)
* **Safari**. Es el navegador por defecto de los equipos Mac aunque también está disponible para Windows. Está a la última en lo que respecta a la integración de **HTML5** y **CSS3**. También utiliza el motor de renderizado “**Webkit**”.
* **Opera**. Es poco utilizado pero es el más innovador y soporta en su totalidad todas las especificaciones **HTML5** y **CSS3**. En un principio utilizaba el motor de renderizado “**Presto**” pero posteriormente se cambió a “Webkit”. (A la hora de realizar este manual también ha anunciado que se pasará a “**Blink**”.)
* **IOS Safari**. Es el navegador móvil más compatible con **HTML5** y **CSS3**. Utiliza el motor de renderizado “**Webkit**”.
* **Opera Mini**. Se presenta en dos versiones, Opera Mini y Opera Mobile. Es probablemente uno de los navegadores para móviles más utilizado en el mundo. Como su hermano “mayor” soporta **HTML5** y **CSS3** en su totalidad. Utiliza el motor de renderizado “**Presto**”.
* **Android Browser**. Es bastante compatible con **HTML5** y **CSS3**. Utiliza el motor de renderizado “**Webkit**”.

Algunos de estos navegadores se adelantan a las nuevas funcionalidades definidas por el **W3C** y para que puedan entender las nuevas etiquetas **HTML5** o propiedades **CSS3** necesitan de los llamados prefijos privativos. Así lo hemos hecho en el caso de aplicar los degradagos en la propiedad “**Background**” en los que aplicábamos el prefijo “**-moz**” a los valores del degradado. Esto hace que sólo reconozcan correctamente esta propiedad **CSS** aquellos navegadores cuyo motor de renderizado sea “**Gecko**” pero no otros***. ¡Veámoslo realizando la práctica 13.2!***

Para navegadores **webkit** la sintaxis es:

**background: -webkit-gradient(linear, origen, fin, from(color), to(color));**

A tener en cuenta:

* **Linear**. Indica un degradado lineal.
* **Origen**. Es el punto inicial del degradado. Está determinado por dos valores separados por un espacio. Se puede indicar números, porcentajes o las palabras claves **top**, **right**, **bottom** o **left**. Por ejemplo: **left top**.
* **Fin**. Es el punto final del degradado. Está determinado por dos valores separados por un espacio. Se puede indicar números, porcentajes o las palabras claves **top**, **right**, **bottom** o **left**. Por ejemplo: **right top**.
* **From(color)**. Indica el color de inicio del degradado.
* **To(color)**. Indica el color de fin del degradado.

Para Internet Explorer la sintaxis es parecida a la de Firefox pero con el prefijo “**-ms**”:

En el caso de los degradados circulares, la sintaxis para “**Webkit**” es:

**background: -webkit-gradient(radial, origen, radio1, fin, radio2, from(color), to(color));**

A tener en cuenta:

* **Radial**. Indica un degradado circular.
* **Origen**. Es el punto inicial del degradado, en este caso del círculo interior. Está determinado por dos valores separados por un espacio. Se puede indicar números, porcentajes o las palabras claves **top**, **right**, **bottom**, **left** y **center**. Por ejemplo: **center center**.
* **Radio1**. Es el radio del círculo interior.
* **Fin**. Es el punto final del degradado. Está determinado por dos valores separados por un espacio. Se puede indicar números, porcentajes o las palabras claves **top**, **right**, **bottom**, **left** y **center**.
* **Radio2**. Es el radio del círuclo exterior.
* **From(color)**. Indica el color de inicio del degradado, es decir del círculo interior.
* **To(color)**. Indica el color de fin del degradado, es decir del círculo exterior.

Y para Internet Explorer la sintaxis del degradado radial es igual que la de Firefox sólo que cambiando el prefijo “**-moz**” por el prefijo “**-ms**”.

Y por último, ¿Cómo será la sintaxis de los degradados para el navegador Opera? Pues si, ha acertado. Igual que para Firefox pero con el prefijo “**-o**”.

***Para terminar de dominar el tema realice ahora la 2ª ejercitación del capítulo 13.***

SOPORTE PARA ELEMENTOS HTML5 EN NAVEGADORES ANTIGUOS

En apartados anteriores de este tema hemos vislumbrado otro problema para los desarrolladores web más modernos… ¿Qué pasa con mis webs creadas con **HTML5** y **CSS3** cuando se visualizan en navegadores antiguos, como el caso que hemos comentado de Internet Explorer 6, que no entienden estas implementaciones?

Cuando un navegador se encuentra con un elemento desconocido simplemente lo ignora y no lo muestra o lo muestra de una manera diferente, por ejemplo, considerando un elemento en bloque como un elemento en línea. Esto significaría que nuestra página web desarrollada en **HTML5** se mostraría de una forma totalmente diferente y deformada.

Una forma de conseguir que los navegadores más antiguos apliquen adecuadamente los estilos **CSS3** y los elementos **HTML5** es utilizar la librería JavaScript **MODERNIZR**. Se trata de una utilidad pensada para detectar funcionalidades dentro de cualquier navegador y añadirles el soporte necesario para poder utilizar los nuevos objetos y estilos si no están soportados.

Para añadir la librería **MODERNIZR** a una web basta con añadir el siguiente código en el <**head**> de la página justo debajo de la línea que llama a la hoja de estilos **CSS** externa:

**<script src=”modernizr.js” type=”text/javascript”></script>**

Para más información y para ver si existen versiones más recientes de **MODERNIZR** visite su web: [**www.modernizr.com**](http://www.modernizr.com).

Desde la página web anterior, se puede descargar esta librería que tiene dos versiones, una completa, llamada versión de desarrollo que contiene todas las características de **HTML5** y **CSS3** y otra, llamada versión de producción donde se puede elegir la característica a compatibilizar.

***Detengámonos brevemente para comprobar el funcionamiento de esta librería realizando la práctica 13.3.***

Para tener una visión completa y actualizada de la compatibilidad de los diferentes navegadores con las recomendaciones **HTML5** y **CSS3** de la **W3C** puede visitar la web: [**www.caniuse.com**](http://www.caniuse.com).

Otra utilidad que le invitamos a conocer es **BOILERPLATE**, (**html5boilerplate.com**). Se trata de una plantilla que pretende simplificar los comienzos de un proyecto web incluyendo el código necesario optimizado para solucionar, entre otros, estos problemas de compatibilidad.

ORGANIZACIÓN DE UN SITIO WEB

Como hemos podido comprobar a medida que hemos ido avanzando en el curso hemos ido completando un ejemplo de sitio web dedicado al turismo en España. Un sitio web se compone de varias páginas web que a su vez contienen varios archivos de estilos, de imágenes, etc, etc.

Hasta ahora hemos ido guardando todos los archivos necesarios para nuestro sitio en una sola carpeta pero las buenas prácticas indican que se deben ordenar en varias carpetas.

Cada creador puede utilizar las carpetas que le parezcan oportunas. Nosotros proponemos, para nuestro ejemplo, la siguiente estructura:

* Todos los archivos “**.htm**” los dejamos en la carpeta principal en este caso su carpeta de prácticas.
* Los archivos para descargar, tenemos un archivo “**.pdf**” que utilizamos al principio, los colocaríamos en la carpeta “**descargas**”.
* Los archivos de estilos “.css” los almacenaríamos en la carpeta “**estilos**”.
* Los archivos correspondientes a tipos de letra los almacenaríamos en la carpeta “**fuentes**”.
* Los archivos de imágines los almacenaríamos en la carpeta “**imágenes**”.
* Las librerías los almacenaríamos en la carpeta “**librerias**”.

Hay que tener en cuenta que en todos y cada uno en los enlaces de nuestro sitio web hay que indicar la ruta correcta a cada archivo siguiendo los ejemplos “**href=estilos/reset.css**” o “**imagenes/turismovacaciones.jpg**”.

***Realice ahora la tercera ejercitación del tema 13 para dejar totalmente dominado este apartado.***

EL MODELO DE CAJA FLEXBOX

Eje

Principal

Eje Transversal

***Elementos distribuidos horizontalmente***

***Elementos distribuidos verticalmente***

Eje Principal

Eje

Transversal

Al finalizar el apartado correspondiente a la propiedad “**display**”, en este mismo capítulo, anunciábamos que explicaríamos más adelante sus valores “**flex**” e “**inline-flex**”. ¡Vamos a ello!

Hasta ahora los diferentes elementos se mostraban en el navegador siguiendo el orden en que aparecían en el documento **HTML**. Este orden lo alterábamos un poquito utilizando, entre otras, las propiedades **CSS** “**float**”, “**clear**” y “**position**”. Pues bien, el modelo flexible de caja, modifica radicalmente este planteamiento permitiendo colocar los diferentes elementos de un documento **HTML** alterando el orden de forma dinámica, apareciendo en el navegador, antes o después, con independencia de cómo estén en el archivo del código fuente. También permite adaptar el tamaño para rellenar el espacio disponible en cualquier dispositivo de visualización. Una caja flexible expande sus elementos para llenar el espacio libre disponible, o los reduce para que no haya desbordamientos.

Este modelo de caja es tan novedoso que, a la hora de escribir este manual, aún hay pocos navegadores que lo implementen, pero es de esperar que en el futuro se utilice másivamente. Además la notación aquí utilizada no es la recomendación definitiva del **W3C** aunque no se preveen cambios significativos. No obstante recomendamos que se esté atento por si hubiera algún cambio.

Para entender bien el funcionamiento del modelo de caja flexible debemos conocer su terminología:

* **Contenedor flexible o Flex container**. Es la caja contenedora o padre de los elementos flexibles. La mayoría de las propiedades FlexBox se aplican al contenedor flexible o principal.
* **Flex ítems**. Son los hijos del flex container y sobre los que se aplicará el modelo de caja flexible.
* **Ejes**. En cada flex container se definen dos ejes, (Ver gráfico en página anterior):
  + El **eje principal** a lo largo del cual se ordenan los elementos flexibles que puede ser horizontal, en filas, o vertical, en columnas.
  + El **eje transversal** que es el eje perpendicular al eje principal.

Veremos que una propiedad **CSS** define la dirección del eje principal, en línea o en columna y otras propiedades nos permitirán alinear los flex ítems teniendo en cuenta el eje principal y/o el eje transversal. También podremos cambiar el sentido de aparición de los elementos flexibles.

* **Líneas.** Los ítems pueden ser distribuidos en una sóla línea o en varias dentro del contenedor.

***Veamos un ejemplo realizando la práctica 13.4.***

La sintaxis de esta propiedad es:

**-webkit-flex-direction: row / row-reverse / column / column-reverse;**

A tener en cuenta:

* **Row**. Indica que el eje principal sea el horizontal y la dirección de izquierda a derecha. Como hemos visto, es el valor que se aplica por defecto si no utilizamos esta propiedad.
* **Row-reverse**. Indica que el eje principal sea el horizontal y la dirección de derecha a izquierda.
* **Column**. Indica que el eje principal sea el vertical y la dirección de arriba abajo.
* **Column-reverse**. Indica que el eje principal sea el vertical y la dirección de abajo arriba.

***Realice la práctica 13.5.***

La propiedad “**flex-wrap**” define si los flex ítems se colocan dentro de su contenedor en una o varias líneas. Su sintáxis es:

**-webkit-flex-wrap: nowrap / wrap / wrap-reverse;**

A tener en cuenta:

* **Nowrap**. Indica que los elementos flexibles no se coloquen en varias líneas. Tal como acabamos de comprobar en chrome, es el valor que se aplica por defecto si no utilizamos esta propiedad.
* **Wrap**. Indica que los elementos flexibles se coloquen en varias líneas.
* **Wrap-reverse**. Indica que los elementos flexibles se coloquen en varias líneas en orden inverso.

***¡Veámoslo en la práctica 13.6!***

Existen varias propiedades que nos permiten alinear los elementos flexibles siguiendo el eje principal y/o el eje transversal. ¡Vamos a ver una a una!

La propiedad “**align-items**” alínea los hijos FlexBox a lo largo del eje transversal. Su sintaxis es:

**-webkit-align-items: flex-start / baseline / flex-end / center / stretch;**

A tener en cuenta:

* **Flex-start o baseline**. Indica que el inicio de todos los elementos flexibles esté a ras del inicio del eje transversal.
* **Flex-end**: Indica que el final de todos los elementos flexibles esté a ras de la parte inferior del eje transversal.
* **Center**. Indica que el centro de todos los elementos flexibles esté a ras de la parte central del eje transversal.
* **Stretch**. Indica que todos los elementos flexibles se expandan para llenar toda la longitud del eje transversal. Esta es la opción por defecto si no utilizamos esta propiedad. Por lo tanto, el resultado de este valor, es lo que en estos momentos nos muestra chrome. Muy importante caer en la cuenta de que, utilizando este valor, todos los elementos flexibles ocupan toda la altura de su contenedor padre, sin importar su contenido y sin importar que cambie la anchura y la altura por cambios en el tamaño de la ventana, lo que facilita enormemente el diseño.

***Práctiquelo realizando la práctica 13.7.***

La propiedad “**justify-content**” especifica cómo se disponen los elementos sobre el eje principal, en término de lo que sucede con el espacio blanco sobrante entre hijos. Su sintaxis es:

**-webkit-justify-content: flex-start / flex-end / center / space-between / space-around;**

A tener en cuenta:

* **Flex-start**. Agrupa todos los hijos FlexBox al principio del eje principal, con todo el espacio al final.
* **Flex-end**. Agrupa todos los hijos FlexBox al final del eje principal, con el espacio todo al principio.
* **Center**. Agrupa todos los hijos FlexBox en el centro del eje principal, con el espacio repartido por igual a cada extremo.
* **Space-around**. Reparte todo el espacio entre los elementos hijos y en el exterior de todos ellos en cantidades iguales.
* **Space-between**. Tiene un efecto similar a “**space-around**”, pero sin espacio asignado a cada extremo de la serie de hijos FlexBox, es decir, los hijos de ambos extremos se ajustan a cada lado del contendor y se reparte el espacio exterior que hay entre los elementos hijos centrales.

***¡Veámoslo en la práctica 13.8!***

La propiedad “**align-content**” define como se alinean las líneas formadas por los elementos flexibles dentro del contenedor flexible cuando hay espacio adicional en el eje transversal. Sólo en el caso de que estos se distribuyan en varias líneas, es decir, tengan el valor “**wrap**” en la propiedad “**flex-wrap**” o “**flex-flow**”, (como sucede en el ejemplo que estamos empleando para nuestra práctica). Su sintaxis es:

**-webkit-align-content: flex-start / flex-end / center / space-between / space-around / stretch;**

A tener en cuenta:

* **Flex-start**. Agrupa todos los hijos FlexBox al principio del eje transversal, con todo el espacio al final.
* **Flex-end**. Agrupa todos los hijos FlexBox al final del eje transversal, con tod el espacio al principio.
* **Center**. Agrupa todos los hijos FlexBox en el centro del eje transversal, con el espacio repartido por igual a cada extremo.
* **Space-around**. Reparte todo el espacio entre los elementos hijos y en el exterior de todos ellos en cantidades iguales.
* **Space-between**. Tiene un efecto similar a “**space-around**”, pero sin espacio asignado a cada extremo de la serie de hijos FlexBox, es decir, los hijos de ambos extremos se ajustan a cada lado del contendor y se reparte el espacio exterior que hay entre los elementos hijos centrales.
* **Stretch**. Es el valor por defecto. Muestra los elementos flexibles alineados por el eje transversal.

***¡Veámoslo en la práctica 13.9!***

La propiedad “**order**” especifica en que orden se dibujan los elementos flexibles en su contenedor con independencia de cómo aparezcan en el **HTML**. Su sintaxis es:

**-webkit-order: nº entero;**

A tener en cuenta:

* Los elementos se disponen en orden ascendente según su valor de “**order**”. Cuanto mayor sea el valor más tarde aparecerá el elemento hijo.
* Por defecto, todos los hijos FlexBox tienen el valor “order” 0.
* Los elementos con el mismo valor en “**order**” se presentan en el orden en que aparecen en el código fuente.
* Esta propiedad admite números negativos.

La propiedad “**flex-grow**” establece el factor de crecimiento flex. Se asigna a los elementos flexibles e indica cuánto crecerá este elemento en relación a los demás cuando haya espacio libre.

La propiedad “**flex-shrink**” establece el factor de encogimiento flex. Se asigna a los elementos flexibles e indica cuanto decrecerá un elemento respecto a los demás cuando no haya espacio libre.

La propiedad “**flex-basis**” estalece el ancho base inicial del elemento flex. Acepta los mismos valores que la propiedad “**width**” y su funcionamiento es similar, salvo en caso de asignar el valor “**auto**”.

La propiedad “**flex**” es la forma acortada de “**flex-grow**”, “**flex-shrink**” y “**flex-basis**”.

***¡Veámoslo en funcionamiento realizando la práctica 13.10!***

MAPAS DE IMÁGENES

Anteriormente aprendimos a enlazar a otra página pulsando sobre una imagen. Un mapa de imágenes en **HTML** es una imagen permite enlazar a varias páginas diferentes en función de la parte de la imagen donde se haga el clic del ratón.

Aunque el uso de mapas de imágenes es una práctica cada vez más en desuso en las webs acutales por los problemas que plantean a la hora del posicionamiento en los motores de búsqueda y de visibilidad en las web móviles, no deja de ser una utilidad más de **HTML** y por eso lo estudiaremos al menos brevemente.

Existen varias formas de utilizar mapas. Nosotros trabajaremos con el método **CSIM** (*Client Side Image Map*) ya que es el más usado y por que se ejecuta en el lado del cliente no necesitando hacer llamadas al servidor web incrementando el tiempo de respuesta de la web.

Para realizar un mapa necesitamos:

* Una imagen en la que indicaremos las zonas donde se puede hacer clic mediante unas coordenadas en píxeles. Estas zonas se conocen en **HTML** como “**áreas**” o “**zonas activas**”.
* La etiqueta <**img**> en la que indicaremos que se trata de una imagen de mapa
* Las etiquetas propias del mapa y los enlaces a otras páginas.

***¡Veámoslo realizando la práctica 13.11!***

La etiqueta <**map**> indica que comienza un mapa y le asigna un nombre mediante el atributo “**id**” que se corresponde con el nombre indicado en el atributo “**usemap**” añadido en la etiqueta <**img**>.

La etiqueta <**area**> determina las zonas activas de la imagen. Su sintaxis es:

**<area shape=”forma” coords=”coordenadas” href=”destino” alt=”comentario” />**

A tener en cuenta:

* **Shape**. Determina la forma en la que indicamos las coordenadas de las zonas activas en la imagen del mapa. Puede ser:
  + **Rect**. La zona activa es un rectángulo que viene indicado por las coordenadas en píxeles de la esquina superior izquerda e inferior derecha del rectángulo. (**coords=”x, y, x1, y1”**).
  + **Circle**. La zona activa es un círculo que viene indicado por las coordenadas en píxeles del punto central y la longitud en píxeles del radio. (**coords=”x, y, r”**).
  + **Polygon**. La zona activa es un polígono que viene indicado por las coordenadas en píxeles de tantos puntos como vértices tenga el polígono. (**coords=”x, y, x1, y1,x2, y2,x3, y3, x4, y4”**).
  + **Default**. La zona activa es el resto de la imagen, la parte a la que no se le ha asignado ninguna otra zona activa con “**rect**”, “**circle**” o “**poligon**”.
* **Coords**. Especifica las coordenadas que permiten al navegador construir la forma geomética de la zona activa.
* **Href**. Indica el archivo o la web de destino asociada a la zona activa indicada en “**coords**”. Es opcional porque puede ser que el diseñador desee que alguna zona activa no incluya ninguna acción.
* **Alt**. Permite agregar un comentario. También es opcional.

Existen programas que ayudan a realizar mapas de imágenes como MapThis o Xmap y, por su puesto, Dreamweaver.

***Para terminar de dominar bien esta parte realice la cuarta ejercitación del capítulo 13***

OPACIDAD Y TRANSPARENCIA CON CSS

La propiedad “**opacity**” permite indicar la opacidad o transparencia de un elemento. Su sintaxis es:

**opacity: Valor comprendido entre 0 y 1;**

A tener en cuenta:

* Con el valor “**1**”, la opacidad es completa y la transparencia nula. Con “**0**” la opacidad es nula y el elemento completamente transparente.
* Se aplica al elemento y a todos sus descendientes.
* El elemento al que se le aplica la propiedad “**opacity**” es transparente y deja entrever el elemento que se encuentra situado por debajo de él.
* Esta propiedad es entendida por todos los navegadores.

***¡Veamos un ejemplo realizando la práctica 13.12!***

***Para terminar de dominar este amplio capítulo realice ahora su quinta y última ejercitación.***

CAPÍTULO 14.- ANIMACIONES Y MULTIMEDIA

CON HTML Y CSS

Las últimas especificaciones de CSS nos permiten aplicar animaciones a los distintos elementos de nuestra página web. En este capítulo veremos las diferentes propiedades CSS y etiquetas HTML con las que podremos transformar la visibilidad de nuestras webs aplicándoles transformaciones y animaciones a sus elementos así como añadiéndoles sonido y vídeo.

TRANSFORMACIONES

La propiedad “**transform**” permite aplicar distintas transformaciones: rotar, escalar, mover e inclinar o sesgar, en 2d y 3d, sobre un elemento. Su sintaxis es:

**transform: función(valor, valor) función(valor, valor);**

donde “**función**” es siempre una de las siguientes palabras claves:

**rotate(ángulo) / rotateX(ángulo) / rotateY(ángulo)**

**scale(númeroX, númeroY) / scaleX(número) / scaleY(número)**

**skew(ángulo) / skewX(ángulo) / skewY(ángulo)**

**translate(númeroX, númeroY) / translate(número) / translate(número)**

**matrix(número, número, número, número, número, número)**

A tener en cuenta:

* Esta propiedad es aceptada tanto por los elementos **HTML** en línea como los de bloque.
* Solo afecta a los elementos hasta su borde. Los márgenes se excluyen al aplicar las transformaciones.
* Las transformaciones no afectan al flujo del documento. Si un elemento se escala o se rota, para el resto no existen tales transformaciones, se comportan como si no hubiese cambiado su tamaño o su visualización.
* La propiedad “**transform**” está implementada en todos los navegadores y sólo necesita prefijo privativo para los navegadores basados en el motor “**webkit**”.
* **El valor rotate(ángulo)**, gira el elemento respecto a su posición original.
  + El valor del ángulo puede definirse en grados sexagesimales (deg), radianes (rad) o grados centesimales (grad).
  + Los valores positivos efectúan la rotación del elemento en el sentido de las agujas del reloj y los valores negativos lo hacen en sentido inverso.
  + Cuando se define un único valor, éste se aplica a los ejes X e Y. Con dos valores, se aplican respectivamente al eje de las X y al de las Y.
  + El punto de rotación coincide con el centro del elemento **HTML** que se gira.
  + **rotateX(ángulo)**, gira el elemento sólo en el eje X.
  + **rotateY(ángulo)**, gira el elemento sólo en el eje Y.

***¡Veamos un ejemplo realizando la práctica 14.1!***

* **El valor scale(númeroX, númeroY)**, permite aumentar o reducir el tamaño del elemento original.
  + Los ejes horizontal y vertical se controlan independientemente, por lo que podemos escalar el elemento de distinta manera en los dos ejes.
  + El “**número**” indica la proporción en la que se deber aumentar o reducir. Con el valor 1 el elemento queda como está. Los valores mayores a 1 realizan un zoom acercándose y los decimales menores a 1, un zoom alejándose.
  + Cuando se define un único valor, éste se aplica a los ejes X e Y. Con dos valores, se aplican respectivamente al eje de las X y al de las Y.
  + El punto de rotación coincide con el centro del elemento **HTML** que se gira.
  + **scaleX(número)**, escala el elemento sólo en el eje X.
  + **scaleY(número)**, escala el elemento sólo en el eje Y.

***¡Veamos un ejemplo realizando la práctica 14.2!***

* **El valor skew(ángulo, ángulo)** hace un sesgado del elemento original convirtiéndole en un romboide. Para ello se inclinan los lados horizontal y vertical en los ángulos indicados.
  + Los lados horizontal y vertical se controlan independientemente, por lo que podemos sesgar el elemento de distinta manera en los dos lados.
  + El “**ángulo**” indica la inclinación de los lados.
  + Cuando se define un único valor en “**ángulo**”, éste se aplica a los lados verticales y horizontales. Con dos valores, se aplican respectivamente al la vertical y al horizontal.
  + El valor del ángulo puede definirse en grados sexagesimales (deg), radianes (rad) o grados centesimales (grad).
  + **skewX(ángulo)**, inclina sólo los lados verticales.
  + **skewY(ángulo)**, inclina sólo los lados horizontales.

***¡Veamos un ejemplo realizando la práctica 14.3!***

* **El valor translate(número, número)**, desplaza el elemento hacia donde indican los parámetros.
  + El primer “**número**” es la translación que se produce en horizontal hacia la derecha. (Eje X). Si la cantidad indicada es negativa la translación es horizontal hacia la izquierda.
  + El segundo “**número**” indica la translación del elemento verticalmente, hacia abajo. (Eje Y). Si la cantidad indicada es negativa la translación es vertical hacia arriba.
  + El valor de “**número**” puede definirse en px, %, em, in, mm o cm.
  + Cuando se define un único valor en “**número**”, éste se aplica a los ejes X e Y. Con dos valores, se aplican respectivamente al desplazamiento horizontal y al vertical.
  + **translateX(número)**, desplaza el elemento a lo largo del eje X (horizontal).
  + **translateY(número)**, desplaza el elemento a lo largo del eje Y (vertical).

***¡Veámoslo en la práctica 14.4!!***

* **El valor matrix(número ...)** Combina todos los valores anteriores. Utiliza 6 parámetros que se obtienen transformando las coordenadas de los vértices en una matriz matemática. La fórmula es algo complicada por eso solo nombraremos que existe y no profundizaremos más en él puesto que tampoco se utiliza masivamente.

La propiedad “**transform-origin**” indica cual será el punto de origen desde el que se transforma el elemento. Piense en los valores que hemos estudiado en la propiedad “**transform**”, todos, las rotaciones, los escalamientos, los sesgos y los traslados se han realizado tomando como punto de referencia el centro del elemento que hemos transformado. Esta propiedad, pues, permite cambiar ese punto. Su sintaxis es:

**transform-origin: coordenadaX coordenadaY;**

A tener en cuenta:

* **CoordenadaX**, indica la coordenada X (horizontal) del origen. Puede expresarse en % o una medida o las palabras claves “**left**”, “**center**” y “**right**”.
* **CoordenadaY**, indica la coordenada Y (vertical) del origen. Puede expresarse en % o una medida o las palabras claves “**top**”, “**center**” y “**bottom**”.

***Para terminar de dominar bien esta parte realice la 1ª ejercitación del capítulo 14***

TRANSFORM EN 3D

En las propiedades anteriores nos manejábamos en 2D, en el plano de la pantalla. Recientemente **CSS3** ha dado el salto a la 3ª dimensión con las siguientes propiedades. Como aún no son definitivas solamente las enunciaremos para conocer su existencia.

* **Transform-style:** Especifica cómo se representan los elementos anidados en el espacio 3D. Admite los valores: “**flat**” y “**preserve**”.
  + **Flat**: Los hijos del elemento al que se le declara no conservan su posición en la 3D. Desaparecen de la vista.
  + **Preserve-3D**: Los hijos del elemento al que se le declara conservan sus propiedades y se mantienen visibles.
* **Perspective.** Aplica la 3ª dimensión o profundidad. Ajusta el punto de vista con el que se muestra el elemento. Cuanto más bajo es el valor más profundidad otorga. Con un valor de 0 el resultado es que no hay efecto 3D.
* **Perspective-origin.** Establece la posición X e Y desde la cual el espectador parece estar mirando a los hijos del elemento.
* **Backface-visibility.** Controla si se muestra o no la cara posterior del elemento al transformarlo. Tiene dos valores “**visible**” y “**hidden**”.

¿Se atreve a probar su funcionamiento?

TRANSICIONES con CSS3

En **CSS3** una transición es un cambio progresivo de un valor de una propiedad a otro valor de la misma propiedad. Por ejemplo para la propiedad “color” pasar del valor “verde” al valor “amarillo”. O para la propiedad “width” pasar del valor 20px., al valor 100px. Al hablar de cambio “progresivo” significamos que el cambio de un valor a otro se realiza pasando por sus valores intermedios provocando un efecto de animación.

Las transiciones no se pueden aplicar a todas las propiedades CSS, solo a aquellas cuyos valores tengan un componente numérico. Al final de este apartado incluiremos un listado de las propiedades a las que se puede aplicar.

A continuación veremos las 4 propiedades CSS que nos permiten realizar transiciones.

La propiedad “**transition-property**” indica las propiedades a las que se se aplicarán las transiciones, por ejemplo, como ya hemos dicho, el color o la anchura de un elemento. Su sintaxis es:

**transition-property: none / all / propiedad, propiedad, … ;**

A tener en cuenta:

* **None**. Indica que no se aplican las transiciones a ninguna propiedad. Es el valor por defecto.
* **All**. Indica que las transiciones se aplicarán a todas las propiedades que puedan admitirlas.
* **Propiedad**. Se trata del nombre de la propiedad o propiedades a las que queremos aplicarles las transiciones. Si hay más de una deben ir serparadas por coma.

***¡Vayamos a la práctica 14.5!***

La propiedad “**transition-duration**” indica el tiempo que tardará en realizarse la transición. Su sintaxis es:

**transition-duration: tiempo;**

A tener en cuenta:

* **Tiempo**. Indica el tiempo de duración de la transición en segundos “**s**” o milisegundos “**ms**”.
* Si en “**tiempo**” indicamos un valor negativo se computa como 0 y por lo tanto no se produce la transición
* Si en la propiedad “**transition-property**” se indican varias propiedades a cambiar se puede declarar un tiempo de transición para cada una.

***¡Veámoslo en la práctica 14.6!***

Con las dos propiedades anteriores bastaría para que la transición funcionara pero tenemos otras propiedades…

La propiedad “**transition-timing-function**” determina los cambios de velocidad mientras se ejecuta la transición en el tiempo estipulado. Su sintaxis es:

**transition-timing-function: linear / ease / ease-in / ease-out / ease-in-out / steps (num) / steps (num, start) / steps (num, end) / cubic-bezier (num, num, num, num);**

A tener en cuenta:

* **linear**. La velocidad de la transición se mantiene constante durante toda su duración.
* **Ease.** Es el valor por defecto. La transición se efectúa rápida al principio y se ralentiza al terminar.
* **Ease-in.** La transición se efectúa lenta al principio y coge su máxima velocidad al ir llegando a su fin.
* **Ease-out.** La transición es rápida al principio y va disminuyendo la velocidad al finalizar.
* **Ease-in-out.** La trasición va lenta al principio, se vuelve rápida a la mitad y acaba lenta.
* **Steps (num).** La transición no es líneal sino a saltos. “**num**” son los saltos que se producen en la transición.
* **Steps(num, start).** Indica el número de saltos que se realiza al inicio de cada fracción de tiempo.
* **Steps(num, end).** Indica el número de saltos que se realiza al final de cada fracción de tiempo.
* **Cubic-bezier.** Definimos la velocidad de la transición en base a una curva cúbica de Bezier. Para ello incorporamos cuatro valores “**num**” que serán números decimales entre el 0 y el 1.

***¡Vemos con la práctica 14.7 cómo se efectúan las transiciones utilizando timing-function!***

La propiedad “**transition-delay**” indica el retraso de tiempo que transcurre entre la acción que desencadena la transición y su comienzo. Su sintaxis es:

**transition-delay: tiempo;**

A tener en cuenta:

* **Tiempo**. Indica el tiempo en que se retrasará el inicio de la transición en segundos “**s**” o milisegundos “**ms**”.
* **Tiempo,** admite valores de tiempo negativos.

***¡Veamos cómo afecta esta propiedad realizando la práctica 14.8!***

Las propiedades de transición se pueden indicar de forma acortada o resumida con la siguiente sintáxis:

**transition: property duration timing-function delay;**

A tener en cuenta:

* Se puede utilizar la forma acortada para declarar varias propiedades a cambiar con igual o distintos parámetros en cada una de las transiciones. En este caso se declaran las distintas propiedades en el orden indicado separadas por comas de la siguiente forma: “**transition: color 3s linear 0s, width 1s ease 1s;**”.

***Para terminar de dominar bien esta parte realice la 2ª ejercitación del capítulo 14***

ANIMACIONES EN CSS

Hasta ahora, en este capítulo, hemos visto las propiedades de transformación y transición. Estas inician sus cambios después de la intervención del usuario, casi siempre con la pseudoclase “**:hover**”. Pero **CSS3** provee de una serie de propiedades que permiten que se realicen animaciones sin la intervención del usuario es decir que se activan automáticamente al cargar la página web en el navegador.

Otra diferencia entre las animaciones y las propiedades “**transform**” y “**transition**” es que estas tienen un estado inicial y uno final, mientras que en aquellas permiten alterar la línea de tiempo y desarrollo de los cambios en función de las necesidades o gustos del desarrollador.

Las animaciones en **CSS3** se crean utilizando la técnica del “**Fotograma Clave**”. Los fotogramas clave son los que marcan el estado del elemento animado al principio y al final de la animación y en las posiciones intermedias en las que el elemento tenga que variar de forma o propiedad.

Además de los fotogramas clave disponemos de una serie de propiedades, semejantes a las de transiciones, que dan nombre a la animación, marcan la duración, los retrasos, etc…

Los fotogramas claves se crean con la regla “**@Keyframes**”. Su sintaxis es:

**@keyframes “nombre\_animación” {**

**0% {propiedad1: valor; propiedad2: valor; …}**

**…**

**100% {propiedad1: valor; propiedad2: valor; …}**

**}**

A tener en cuenta:

* **Nombre\_animación**, es el nombre que asignaremos a la animación mediante la propiedad “**animation-name**”. Este nombre es necesario para poder aplicar el resto de propiedades de la animación.
* Dentro de la declaración, entre llaves, pondremos los diversos fotogramas clave los cuales son unas “subreglas” **CSS**, (en realidad cada fotograma clave es en si una regla **CSS**), compuestas por:
  + **Selector**. Indicados en tantos por ciento, posicionan el elemento en un punto concreto de la animación exponiendo las propiedades que el elemento debe tener en ese punto de la animación. El primero debe ser siempre el 0% que indica el principio de la animación, y el último el 100% que indica el final. Entre medias se pueden colocar los fotogramas claves que se desee indicando su tanto por ciento.
  + **Declaraciones**. Dentro de las llaves de los fotogramas pondremos las propiedades **CSS** que tenga el elemento en ese punto concreto de la animación. Lo normal es poner las mismas propiedades en varios fotogramas, pero con diferentes valores.
* Es obligatorio poner al menos dos fotogramas clave, el inicio y el final de la animación.
* El paso de un fotograma clave al siguiente se realiza de forma progresiva, siempre que esto sea posible. Sólo las propiedades cuyo valor esté basado en un número, (números, medidas, porcentajes, colores), pueden variar progresivamente. Las propiedades cuyos valores no son números, (palabras clave, nombres, urls, etc…), son ignoradas y no se pude hacer ningún tipo de cambio en ellas.
* Para mover un elemento dentro de la página, éste debe estar posicionado, es decir, tiene que tener la propiedad “**position: absolute;**” o “**position: relative;**” ya que sino no se le pueden aplicar las propiedades de posicionamiento (left, right, top, bottom), que son las que cambian al elemento de sitio.

Veamos ahora las propiedades de animación imprescindibles para realizar una animación simple:

La propiedad “**animation-name**” indica a qué elemento debe aplicarse la animación y asigna un nombre a la animación. Su sintaxis es:

**animation-name: “nombre\_animación”;**

A tener en cuenta:

* **Nombre\_animación**, es el nombre que usaremos en la regla @keyframes.

La propiedad “**animation-duration**” indica el tiempo que va a durar la animación. Su sintaxis es:

**animation-duration: “tiempo”;**

A tener en cuenta:

* **Tiempo**, Indica el tiempo de duración de la animación en segundos “**s**” o milisegundos “**ms**”.
* Esta propiedad es obligatoria ya que toda animación tiene siempre una duración en el tiempo.

***¡Y ahora vayamos a la práctica 14.9!***

La propiedad “**animation-iteration-count**” indica las veces que se repetirá la animación. Su sintaxis es:

**animation-iteration-count: “num” / infinite;**

A tener en cuenta:

* **Num**, es un número entero positivo que Indica el número de veces que se repetirá la animación. Su valor por defecto es 1 por lo que si no ponemos esta propiedad la animación se ejecutará una vez.
* **Infinite,** indica que la animación se repetirá indefinidamente.

***¡Veámoslo en la práctica 14.10!***

La propiedad “**animation-direction**” indica si la animación al llegar al final acaba o si debe volver en sentido inverso. Su sintaxis es:

**animation-direction: normal / alternate;**

A tener en cuenta:

* **Normal**, es el valor por defecto, la animación, una vez concluido el tiempo que se le ha marcado finaliza.
* **alternate,** indica que una vez terminada la animación se repitá en sentido inverso, volviendo a reproducirse la secuencia de atrás a adelante. (Con lo cual la duración de la animación será el doble).

***¡Veámoslo en la práctica 14.11!***

La propiedad “**animation-timing-function**” determina los cambios de velocidad mientras se ejecuta la animación en el tiempo estipulado. Es semejante a la propiedad “**transition-timing-function**” que ya conocemos. Su sintaxis es:

**animation-timing-function: linear / ease / ease-in / ease-out / ease-in-out /**

**cubic-bezier (num, num, num, num);**

A tener en cuenta:

* **linear**. La velocidad de la animación se mantiene constante durante toda su duración.
* **Ease.** Es el valor por defecto. La animación va lenta al principio para volverse rápida en el medio y terminar mucho más lenta.
* **Ease-in.** La animación se efectúa lenta al principio y coge su máxima velocidad al ir llegando a su fin.
* **Ease-out.** La animación es rápida al principio y va disminuyendo la velocidad al finalizar.
* **Ease-in-out.** La animación va lenta al principio, se vuelve rápida a la mitad y acaba lenta.
* **Cubic-bezier.** Definimos la velocidad de la animación en base a una curva cúbica de Bezier. Para ello incorporamos cuatro valores “**num**” que serán números decimales entre el 0 y el 1.

La propiedad “**animation-delay**” indica el tiempo que tarda la animación en empezar después de cargarse la página. Su sintaxis es:

**animation-delay: tiempo;**

A tener en cuenta:

* **Tiempo**. Indica el tiempo en que se retrasará el inicio de la animación en segundos “**s**” o milisegundos “**ms**”.
* **Tiempo,** admite valores de tiempo negativos.

La propiedad “**animation-play-state**” indica si la animación está ejecutándose o en pausa. (Se utiliza junto con JavaScript para que el usuario, mediante un botón, pueda pausar o poner en marcha la animación). Su sintaxis es:

**animation-play-state: paused / running;**

A tener en cuenta:

* **Paused**. Indica que la animación está en pausa.
* **Running.** Indica que la animación se está ejecutando. Este es el valor por defecto.

Al igual que en otros casos, las propiedades de animación se pueden indicar de forma acortada o resumida con la siguiente sintáxis:

**animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-direction;**

A tener en cuenta:

* Las propiedades “**animation-name**” y “**animation-duration**”, que son obligatorias, deben declararse en primer lugar y por ese orden. Las propiedades que no son imprescindibles u obligatorias pueden cambiar de orden e incluso omitirse.

***Para terminar de dominar bien esta parte realice la 3ª ejercitación del capítulo 14***

INSERTAR ARCHIVOS DE AUDIO CON HTML

Insertar un archivo de audio en una web se realiza de forma muy sencilla mediante la etiqueta <**audio**>. Su sintaxis básica es:

**<audio src=”archivo\_de\_sonido”> … </audio>**

Esta etiqueta tiene los siguientes atributos:

* **Src**. Define la ruta del archivo de sonido. Lógicamente es obligatorio.
* **Controls.** Muestra los controles del reproductor de audio. Incluye las funcionalidades de reproducción, pausa, avance y volumen. Si no incluimos este atributo no veremos ningún efecto en pantalla, ni podremos reproducir el sonido a menos que indiquemos el atributo “**autoplay**”.
* **Autoplay.** Indica que el archivo de sonido se reproduzca automáticamente tras la carga de la página.
* **Loop.** Especifica que el archivo de sonido se repita constantemente.
* **Preload.** Indica al navegador que debe descargar el archivo de audio durante la carga de la página de modo que esté disponible para una reproducción inmediata una vez la solicite el usuario. Puede tomar los siguientes valores:
  + **Preload=”none”.** Sin carga previa.
  + **Preload=”metadata”.** Carga previa de los metadatos asociados al archivo de sonido.
  + **Preload=”auto”.** Carga previa automática.
  + El atributo “**preload**” se ignorará si el atributo “**autoplay**” está activado.

**El Problema de los formatos de sonido**

El problema que nos encontramos a la hora de añadir audio a la web es que no todos los navegadores soportan, debido a problemas de patentes, todos los formatos de sonido que existen en la actualidad. Los más habituales son: Ogg, Mp3, Acc y Wav.

A la hora de redactar este manual Google Chrome y Firefox soportan todos los formatos. Safari soporta mp3, wav y acc e Internet Explorer soporta mp3 y aac.

Para asegurarnos que nuestro sonido se puede oir en todos los navegadores la solución más sencilla es utilizar un programa de conversión de audio como por ejemplo “**Audacity**”.

La etiqueta <**source**> permite resolver la problemática de los distintos formatos de sonido. Se utiliza para especificar varios tipos de archivos de audio. Cada navegador escogerá el formato que mejor convenga o el que pueda reproducir. Su formato es:

**<audio controls>**

**<source src=”música.ogg”>**

**<source src=”música.mp3”>**

**<source src=”música.acc”>**

**Su navegador no soporta la etiqueta audio.**

**</audio>**

El navegador reproducidará el primer archivo que entienda. Los otros son ignorados.

La etiqueta <**source**> tiene dos atributos:

* **Src**. Define la ruta del archivo de sonido. Lógicamente es obligatorio.
* **Type**. Define el tipo MIME del contenido. Sirve para acelerar el proceso de carga. Puede valer:
  + Type=”audio/oog”.
  + Type=”audio/mpeg”.
  + Type=”audio/acc”.

***Para terminar de dominar bien esta parte realice la 4ª ejercitación del capítulo 14***

INSERTAR ARCHIVOS DE VIDEO CON HTML

Al igual que el sonido, insertar un archivo de vídeo en una web se realiza de forma muy sencilla mediante la etiqueta <**video**>. Su sintaxis básica es:

**<video src=”archivo\_de\_video”> … </video>**

Esta etiqueta tiene los siguientes atributos:

* **Src**. Define la ruta del archivo de vídeo. Lógicamente es obligatorio.
* **Width**. Indica la anchura del vídeo.
* **Height**. Determina la altura del vídeo. Al igual que con la etiqueta <**img**> se puede definir de forma explícita las dimensiones del vídeo. En otro caso, el elemento se muestra por defecto con la altura y la anchura propias del vídeo. Si especifica una dimensión, pero no la otra, el navegador ajustará automáticamente el tamaño de la dimensión que no se haya especificado para preservar la proporción de aspecto del vídeo.
* **Poster**. Permite especificar una imagen que el navegador usará mientras está descargando el vídeo o hasta que el usuario inicie su reproducción. Si no se especifica este atributo, se inserta la primera imagen del vídeo en su lugar.
* **Controls.** Muestra los controles del reproductor de vídeo. Incluye las funcionalidades de reproducción, pausa, avance y volumen. . Si no incluimos este atributo veremos el primer fotograma del video pero no podremos reproducirlo a menos que indiquemos el atributo “**autoplay**”.
* **Autoplay.** Indica que el archivo de vídeo se reproduzca automáticamente tras la carga de la página.
* **Loop.** Especifica que el archivo de vídeo se repita constantemente.
* **Preload.** Indica al navegador que debe descargar el archivo de vídeo durante la carga de la página de modo que esté disponible para una reproducción inmediata una vez la solicite el usuario. Puede tomar los siguientes valores:
  + **Preload=”none”.** Sin carga previa.
  + **Preload=”metadata”.** Carga previa de los metadatos asociados al archivo de vídeo.
  + **Preload=”auto”.** Carga previa automática.
  + El atributo “**preload**” se ignorará si el atributo “**autoplay**” está activado.

**El Problema de los formatos de vídeo**

Al igual que en el caso del sonido con el vídeo también tenemos el problema de los distintos formatos que soportan los navegadores. En la actualidad podemos encontrar los siguientes: Ogv, H.264, y WepM.

A la hora de redactar este manual Google Chrome y Firefox soportan todos los formatos. Safari e Internet Explorer soporta todos aunque instalando los plugin adecuados.

Aunque prácticamente las últimas versiones de los navegadores soportan todos los formatos de audio y vídeos debemos tener en cuenta que muchos usuarios no tienen actualizados sus navegadores y que muchas personas utilizan tecnologías de navegación móviles. Para estar completamente al día de los codecs de sonido y audio soportados por los diferentes navegadores de todos los tipos recomendamos visitar las webs “HTML5\_audio” y “HTML5\_video” de la Wikipedia en su versión inglesa.

También podemos utilizar un programa de conversión de video como “Miro Video Converter”, “Prism Video Converter” o la extensión de Firefox “Firefogg”, para asegurarnos que nuestro video se puede ver en todos los navegadores.

Además, como en el caso del audio, podemos utilizar la etiqueta <**source**>.

***Para terminar de dominar bien esta parte realice la 5ª ejercitación del capítulo 14***

CAPÍTULO 15.- OTRAS TECNOLOGÍAS EN HTML5. CANVAS, GEOLOCALIZACIÓN…

A lo largo de todo este curso hemos ido aprendiendo el lenguaje **HTML5** y su compañero inseparable **CSS3**, pero el mundo **HTML5** es una plataforma abierta que no se queda aquí. El **W3C** define 8 tecnologías que lo componen aunque, a la hora de elaborar este manual, aún se estén desarrollando:



* **Semántica.** Es el conjunto de todas las etiquetas que hemos ido estudiando a lo largo de este curso. Es la parte central y frontal del **HTML5**.
* **Offline y Almacenamiento local.** Se trata de servicios para que las aplicaciones **HTML5** sigan funcionando incluso aunque no se disponga de conexión a internet. Se trata del Web storage, local storage, Bases de Datos indexadas y Apis de ficheros.



* **Acceso al dispositivo.** Se trata de aplicaciones que permiten el acceso al dispositivo donde está navegando el usuario. Acceso a la cámara, al micrófono. El más avanzado es la geolocalización.
* **Conectividad** Se trata de aplicaciones que permiten una conectividad más eficiente para disponer de comunicaciones en tiempo real, chats, juegos más rápidos y en general mejor comunicación con el servidor. Son los llamados Web sockets.



* **Multimedia.** El Audio y Vídeo que ya conocemos de forma nativa en el navegador.
* **3d, Gráficos y Efectos:** Se trata de tecnologías como: Canvas, WebGL, SVG y las propiedades para 3D de CSS3, que permiten realizar webs impresionantes efectos visuales representados de forma nativa en el navegador. Por ejemplo para el desarrollo de juegos en 3D.



* **Optimización e integración.** Tecnologías como Web Workers y XMLHttpRequest que se encargan de que el navegador se integre con el dispositivo lo que permitirá navegar muchísimo más rápido.



* **CSS3.** Los estilos y efectos que ya conocemos y que mejoran las webs sin sacrificar su rendimiento y estructura semántica.

En este tema vamos a tratar brevemente algunas de estas tecnologías para tener un conocimiento global de la especificación **HTML5**. No profundizaremos en ellas puesto que en todas se utiliza **JavaScript** para su funcionamiento y este es objeto de otro curso de **Fray Luis Centro Académico**.

CANVAS

“**Canvas**” se podría traducir por “lienzo”. Se trata de un espacio de dibujo, un lienzo en la web, para dibujar. En dicho espacio podemos dibujar mediante scripts de **JavaScript**.

La etiqueta <**canvas**> tiene la siguiente sintáxis:

**<canvas width=”200px” height=”200px”> … </canvas>**

A tener en cuenta:

* Los atributos “**width**” y “**height**” definen las dimensiones del lienzo de dibujo. Si no se especifican el área de dibujo o lienzo tendrá una dimensión de 300px de largo y 150px de alto.
* Es posible incluir varias etiquetas <**canvas**> en la misma página. El identificador “**id**” puede resultar útil en este caso.
* Es posible aplicarle estilos de la misma forma que a cualquier otra imagen de la web.
* Por defecto la zona de dibujo que crea esta etiqueta es transparente por lo que, al menos en la fase de diseño, es recomendable agregar un estilo de borde simple de 1px.
* Las especificaciones de **HTML5** preveen en un futuro diseños en 3D.
* Esta etiqueta es reconocida por todos los navegadores.

Esto es todo en lo que respecta a canvas en **HTML**. A partir de aquí el control pasa a a **JavaScript** para realizar los dibujos y diseños. Se pueden realizar líneas y figuras geométricas con color, distintos grosores, añadir imágenes, texto, sobreados y transformaciones.

***Pero… ¡Veamos un ejemplo sencillo realizando la práctica 15.1!***

A modo de apunte, otras funciones de dibujo en **JavaScrip** son:

* **lineTo(x, y)**. Dibuja una línea recta. Los argumentos “**x**” e “**y**” son las coordenadas del extremo final de la recta. El punto de partida depende de los caminos diseñados anteriormente, dado que el último punto de un camino es el punto de partida para el siguiente. No obstante el punto de partida puede modificarse con la función “**moveTo**”.
* **moveTo(x, y)**. Indica un nuevo punto de partida para el siguiente dibujo a realizar.
* **clearRect(x, y, width, height)**. Borra la zona especificada haciéndola completamente transparente.
* **stroke()**. Como hemos comprobado con los rectángulos se aplican directamente el diseño, color, etc. Sin embargo, para las demás formas, líneas, etc, etc es como si estuviéramos usando un puntero sin tinta. Para que se vea la línea, es necesario ejecutar después de la función correspondiente, la función “stroke()”.
* **lineWidth=valor**. Define el grosor de la línea. El valor por defecto es 1px.
* **lineCap=valor**. Determina como se diseñan los extremos de cada línea. Existen tres valores:
  + **butt.** La línea termina exactamente en el lugar definido por el código. Es el valor por defecto.
  + **round.** Agrega medio círculo a cada extremo de la línea.
  + **square.** Agrega un borde cuadrado a cada extremo de la línea.
* **arc(x, y, radio, anguloInicial, anguloFinal, sentidoInverso).** “**x**” e “**y**” son las coordenadas del centro del círculo. “**radio**” es el radio del mismo. Los parámetros “**anguloInicial**” y “**anguloFinal**” definen los puntos de partida y de llegada del arco en radianes, medidos a partir del eje X. El parámetro “**sentidoInverso**” puede ser “**false**”, dibuja el arco en el sentido de las agujas del reloj, o “**true**” que dibuja el arco en el sentido inverso a las agujas del reloj.
* **quadraticCurveTo.** Dibuja formas complejas a través de fórmulas matemáticas cuadráticas.
* **bezierCurveTo.** Dibuja formas complejas a través de fórmulas matemáticas de curvas de Bézier

***Para terminar de dominar bien esta parte realice la 1ª ejercitación del capítulo 15***

SVG

Además de **Canvas**, **SVG**, (Scalable Vector Graphics), es otra posibilidad para crear gráficos web interactivos y que **HTML5** la reconoce de manera nativa. **SVG** está basado en el lenguaje **XML** que es muy parecido al **HTML** por lo que las instrucciones también se indican con etiquetas.

La etiqueta <**svg**> tiene la siguiente sintáxis:

**<svg width=”200px” height=”200px”> … </svg>**

A tener en cuenta:

* Los atributos “**width**” y “**height**” definen las dimensiones del dibujo.

***¡Pero veamos un ejemplo realizando la práctica 15.2!***

Las principales diferencias entre **Canvas** y **SVG** son:

|  |  |
| --- | --- |
| **CANVAS** | **SVG** |
| Orientado a Pixel. Manipula píxeles. | Basado en el modelo de objetos. Manipula elementos como las mismas etiquetas **HTML**. |
| En una web, es un elemento individual, similar en su comportamiento a la etiqueta <**img**>. | En una web, cada elemento gráfico **SVG**, se puede tratar individualmente como si fuera una etiqueta independiente. |
| Para modificar un gráfico tenemos que utilizar **JavaScript**. | Cada elemento gráfico se puede modificar con etiquetas y/o estilos **CSS**. Se permite a los lenguajes de script el acceso a todos los elementos, propiedades y atributos que componen un documento **SVG**. |

|  |  |
| --- | --- |
| El modelo evento/interacción con el usuario es rudimentario exclusivamente a nivel del elemento **canvas**; las interacciones se deben programar manualmente a partir de las coordenadas del ratón. | Existe la posibilidad de asignar eventos a los distintos elementos de un documento **SVG** tales como al posicionar el ratón o al hacer clic, etc, etc… |
| Modo inmediato. **Canvas** no tiene consciencia de las formas que se han dibujado en el lienzo. Lo único que permanece es el mapa de bits resultante. | Modo retenido. Los elementos del dibujo **SVG** permanecen dentro de un modelo en memoria. |
|  | No pierde calidad si se hace zoom o si se redimensiona. |
|  | Se puede realizar el dibujo con un programa de diseño gráfico como **Illustrator** y pasarlo al código **HTML** bien, copiando y pegando, bien mediante llamadas al archivo externo **.SVG** utilizando las etiquetas <**object**>, <**embed**> e <**img**>. |
|  | Cuando el elemento <**svg**> aparece en un documento **HTML5**, funciona igual que un bloque “**inline**”. |

WEB STORAGE

Cuando navegamos por internet hay muchas páginas que “recuerdan” que ya les hemos visitado. Esta memoria se consigue a través de las tan famosas “cookies”, pequeños pedacitos de información que viajan en cada petición entre el cliente y el servidor web. La utilización de cookies ya no es considerada una buena solución ya que:

* Aumentan el peso de cada petición al servidor.
* Tienen una limitación de 4Kb de espacio disponible.
* No todos los visitantes de nuestra web pueden tener las cookies habilitadas ya que muchos usuarios desconfían de la utilidad que muchas empresas dan a dichas cookies.

**Web Storage**, especificación incluida de forma nativa en **HTML5**, permite utilizar **JavaScript** para guardar información en los navegadores de nuestros usuarios que se elimine al finalizar la sesión, “**SessionStorage**”, o que se guarde de manera permanente en su disco duro, “**LocalStorage**”.

Otras tecnologías de **HTML5** que permiten que las aplicaciones almacenen datos en los dispositivos del cliente son: **Web SQL Database**, **Indexed Database** y **File Access**.

Hay varias razones por las que puede ser recomendable utilizar el almacenamiento en el cliente:

* Permite que una aplicación web funcione cuando el usuario no está conectado, sincronizando los datos cuando vuelve a establecer conexión.
* Aumenta el rendimiento ya que puede mostrar una gran cantidad de datos en la web en el momento en que el usuario hace clic en el sitio, sin tener que esperar a que vuelvan a descargarse.
* Se programa de manera más sencilla con **JavaScript**.

**Web Storage**, al utilizar **JavaScript**, supera ampliamente el alcance de este curso por lo que sólo la incluimos a modo informativo.

WEB WORKERS

Uno de los problemas del lenguaje **JavaScript** es que no permite la ejecución de script en paralelo. El usuario lo nota porque en procesos largos y costosos el navegador se queda “congelado” durante unos instantes.

**Web Workers** es una especificación **HTML5** que permite ejecutar tareas de fondo en paralelo al programa **JavaScript** principal sin afectar al rendimiento de la página web.

WEBSOCKETS

**WebSockets** es una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva para actualizaciones de información entre el navegador del usuario y un servidor. Con esta  API, puede enviar mensajes a un servidor y  recibir  respuestas controladas por eventos sin tener que consultar al servidor para una respuesta. Muy útiles para aplicaciones web de indicadores bursátiles, plataformas web orientadas al trabajo colaborativo, sistemas de notificaciones, juegos on-line, o los conocidos chats.

DRAG AND DROP – ARRASTRAR Y SOLTAR

Esta es una de las nuevas características añadidas en **HTML5** que sirve para crear la capacidad de arrastrar y soltar un elemento de un lugar a otro de igual forma que se hace con los iconos de **Windows**. Hasta ahora, esta funcionalidad se intentaba realizar usando **JavaScript** y **Flash** pero en **HTML5** se implementa de forma nativa por lo que, sin necesidad de recurrir a complementos externos y con un poquito de JavaScript, podemos utilizarlo en nuestras webs.

La tarea de arrastrar y soltar en un navegador, está compuesta por dos partes básicas: los atributos **HTML** y la función **JavaScript**.

En cuanto a los elementos **HTML** se necesitan dos tipos: un elemento arrastrable y otro en el que se pueda soltar. Cada uno de ellos se especifica con una serie de atributos:

Atributos para el elemento arrastrable:

* **draggable**: Si es “**true**” el elemento se puede arrastrar, si es “**false**” o se omite, no se puede mover el elemento.
* “**ondragstart**”: Aquí se indicará la acción que se llevará acabo cuando se empiece a arrastar el elemento.
* “**ondrag**”: Aquí se indicará la acción que se produce mientras se arrastra el elemento.
* “**ondragend**”: Aquí se indicará la acción que se llevará acabo cuando se termine de arrastar el elemento, cuando se suelte.

Y para el elemento al que se pueden arrastrar:

* “**ondragenter**”: Aquí se indicará la acción que se llevará acabo cuando un elemento arrastrable entre dentro del elemento de destino.
* “**ondragover**”: Aquí se indicará la acción que se llevará acabo cuando un elemento arrastrable esté sobre el elemento de destino. En esta función es donde se indica qué elementos arrastrables se pueden soltar en ese lugar.
* “**ondragleave**”: Aquí se indicará la acción que se llevará acabo cuando un elemento arrastrable deje de estar encima del elemento donde se puede soltar.
* “**ondrop**”: Aquí se indicará la acción que se llevará acabo cuando se suelte un elemento arrastrable dentro del elemento de destino.

***¡Pero dejémonos de teoría y veamos un ejemplo práctico realizando la práctica 15.3!*** Comprobaremos que no es tan complicado como parece.

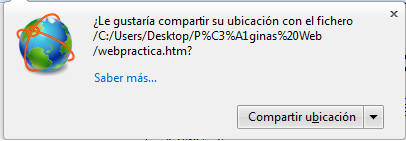
GEOLOCALIZACIÓN

La geolocalización es un procedimiento que permite conocer las coordenadas geográficas, longitud y latitud, de un usuario y posicionarlo sobre un mapa o un plano.

Como otros muchos de los servicios que estamos viendo en este tema, la geolocalización no es nueva en la red, la novedad reside en que esta funcionalidad esté incluida de forma nativa en **HTML5**.

Las aplicaciones de la geolocalización son muchas, una web comercial pude proporcionar los distribuidores más cercanos al usuario o las tiendas y/o hoteles, una red social puede indicar los amigos que se encuentran en los alrededores, etc, etc…

Para geolocalizar al usuario se utiliza su dirección IP, la de su red Wifi o el GPS de su teléfono móvil.

Puesto que obtener esta información del usuario puede generar problemas de protección de la privacidad este servicio de geolocalización debe suponer una opción voluntaria del usuario por lo que al utilizar este servicio aparece una ventana de aviso en el navegador del visitante pidiéndole autorización y la información obtenida se envía a través de internet de forma cifrada. Además los datos obtenidos deben expirar en dos semanas.

Como ya es habitual esta utilidad de geolocalización utiliza en gran medida **JavaScript**.

***Pero… ¡Veámoslo en la práctica 15.4!***

La característica **JavaScript** “**position.coords**” devuelve las coordenadas como hemos visto pero también permite recuperar otros valores como:

* **position.coords.longitude** envía la longitud de la posición actual.
* **position.coords.latitude** envía la latitud de la posición actual.
* **position.coords.altitude** devuelve la altitud de la posición actual.
* **position.coords.accuracy** indica la precisión de las coordenas.
* **position.coords.altitudeAccuracy** proporciona la precisión de la altitud.
* **position.coords.heading** proporciona la posición en grados respecto al norte.
* **position.coords.speed** corresponde con la velocidad del usuario respecto a su última posición.

***Para terminar de dominar bien esta parte realice la 2ª ejercitación del capítulo 15***

CAPÍTULO 16.- RESPONSIVE WEB DESIGN:

LOS MEDIA QUERIES Y

OTRAS TÉCNICAS DE DISEÑO WEB.

En el capítulo 9 de este curso hablamos de páginas web de anchuras fijas o variables. Hasta aquí hemos optado por diseñar con una anchura fija. En este capítulo abordaremos el diseño de webs con una anchura variable que se adapta al tamaño de la ventana que el usuario está utilizando. Esta técnica también se conoce como “**diseño líquido**” o más recientemente como “**Responsive Web Design**” en inglés o “**Diseño Web Adaptativo**” en español.

Con la masiva implantación de móviles y tabletas, cada una con una forma y tamaño de pantalla distinto, cada vez es mayor el número de usuarios que utilizan estos medios para navegar sin que se visualicen las páginas de forma correcta. Se hace pues necesario diseñar para los distintos tamaños de pantalla.

Así, las ventajas de utilizar el **Diseño Web Adaptativo** son:

* Mejora la experiencia de usuario viendo la web de manera más óptima en cada dispositivo para todos los usuarios. Por ejemplo, se evita que el usuario tenga hacer zoom y scroll en su pantalla del móvil para ver bien la web.
* Costes más bajos: ya no es necesario hacer un diseño para cada tipo de pantalla.
* Actualizaciones más eficientes.
* Búsquedas: sólo es necesaria una url, lo que evitará errores o redirecciones.
* Mejora en el posicionamiento de los buscadores.
* Continuidad. Como las webs se ven de forma parecida y correcta en cada dispositivo con el que el usuario se conecte, conseguiremos ofrecerle una experiencia ininterrumpida allá donde esté.

Para diseñar siguiendo un estilo “**Responsive Web Design**” utilizamos:

* Las meta tags: la etiqueta <**meta**>.
* Un diseño fluido o de web flexible abandonando las medidas fijas como píxeles o puntos y adaptando unidades relativas como porcentajes, rem o em.
* **CSS3 Media Queries** que permiten utilizar distintos estilos **CSS** basándose en las medidas del dispositivo donde se va a visualizar nuestra web.

¡Veamos cada uno de ellos!

LA ETIQUETA <META>

Ya aprendimos que en el área del <**head**> de una página web se contiene la información necesaria para la correcta visualización de la página por parte de los navegadores. Y que el contenido de dicha área no se muestra en la página web.

También vimos que en esta área se encuentran los metadatos que es una de las categorías de contenido en **HTML**. Estos metadatos, también conocidos como “**meta tags**”, se indican con las etiquetas <**meta**>, de la cuales ya hemos utilizado alguna como <**meta charset=”utf-8” /**>.

Los metadatos aportan información de varios tipos para diferentes aplicaciones que acceden a nuestra web, como buscadores y utilidades de diversa índole. Cada uno de estos sistemas procesa únicamente aquellas etiquetas <**meta**> que entiende e ignora el resto.

La etiqueta <**meta**> no tiene su correspondiente de cierre y su sintaxis habitual es:

**<meta name=”….” content=”….” />**

Los atributos de la etiqueta <**meta**> son:

* **name.** Asigna un nombre al metadato.
* **content**. Se utiliza para dar valores al metadato creado con “**name**”.

“**name**” y “**content**” se utilizan para indicar propiedades del documento. Cada una de esas propiedades es un metadato. Los valores de estas propiedades o metadatos no son una lista cerrada de palabras clave, aunque el atributo “**name**” suele utilizarse con una serie de valores preestablecidos. Algunos de los metadatos más utilizados son:

* + **application-name**: asigna un nombre al documento web.

Ej: <**meta name=”aplication-name” content=”Página de turismo en España” /**>.

* + **author**. Indica el nombre del creador de la página.

Ej: <**meta=”author” content=”Nombre\_del\_Alumno” /**>

* + **description**. Pequeña descripción de lo que contiene la página web. No debe contener más de 155 caracteres. Utilizado por los buscadores. Se puede acompañar con el atributo “**lang**” para poder ofrecer la descripción de la página para diferentes lenguajes.
  + **generator**. Indica con qué software se ha creado la página. Si la página ha sido creada directamente con **HTML**, como nuestro caso, no es necesario indicarlo.
  + **keywords**. Conjunto de palabras relevantes de la web. Cada palabra o grupo de palabras va separada de las demás por comas. En la actualidad los grandes buscadores han dejado de tenerla en cuenta, aunque muchos diseñadores la siguen implementando para buscadores minoritarios. También se puede utilizar conjuntamente con el atributo “**lang**” para ofrecer “**keywords**” en otros idiomas.
  + **copyright**. Indica el autor de la propiedad intelectual de los contenidos de la web.
  + **organization**. Indica la empresa a la que pertenece la página web.
  + **language**. Especifica el idioma de la web. (El nombre del idioma se indica en inglés). Ej: <**meta name=”language” content=”Spanish” /**>
  + **revisit-after**. Indica a los buscadores cada cuanto tiempo se actualiza la página para que vuelvan a visitarla e indexar los cambios. Lo normal es escribirlo en inglés: “15 days”, “1 month”, “2 months”, etc. Los buscadores más importantes ignoran este metadato. Ej: <**meta name=”revisit-after” content=”1 month” /**>
  + **robots**. Define el comportamiento que los robots de búsqueda deben tener con la página. Podemos incluir todos los valores que queramos separados por comas. Los posibles valores son:
    - **index**. Permite al robot indexar la página y por ello aparecerá en las hojas de resultados de los buscadores. Es el valor por defecto.
    - **noindex**. Indica al robot que no indexe la página.
    - **follow**. Permite al robot seguir los enlaces que hay en nuestra página. También es el valor por defecto junto con “**index**”. Por lo tanto no añadir este metadato es lo mismo que poner:

<**meta name=”robots” content=”index,follow” /**>

* + - **nofollow**. Indica al robot que no puede seguir los links de la página.
    - **noarchive**. Indica a los motores de búsqueda que no utilicen el enlace “En caché” de la página.
    - **nosnippet**. Indica que no se muestre la descripción de la página en los resultados de una búsqueda. (Esa pequeña descripción q ue sale en los resultados de búsqueda, por ejemplo de google, se llama “**snippet**”.)
    - **noimageindex**. Indica que la página no aparezca como la página de referencia de una imagen que se muestra en los resultados de la búsqueda de Google.
    - **noydir**. Indica que no se use la descripción Yahoo Directory, si la hay, como descripción de la página en los resultados de una búsqueda (Sólo para Yahoo).
  + **googlebot**. Es igual que “**robots**” pero sólo para Google.
* **charset**. Se utiliza, como ya sabemos, para especificar la codificación usada en nuestra página. Este elemento debe estar dentro de los primeros 512 bytes de la página.
* **http-equiv**. Este atributo de la etiqueta <**meta**> se utiliza para indicarle al servidor de la página el comportamiento o algunas de sus propiedades. Se trata de metadatos de control. Puede tener los siguientes valores:
  + **http-equiv=“content-type”.** Se utiliza como alternativa al atributo “**charset**” en versiones anteriores a **HTML5**.
  + **http-equiv=”refresh”**. Indica el tiempo para que se refresque la página. Hasta ahora era muy utilizado por ejemplo en las páginas de noticias. También sirve para redireccionar pasados unos segundos a otra página diferente. No es bien vista por los buscadores y el **W3C** recomienda no usarla, así que mejor no emplearla.

Ej: <**meta http-equiv=”refresh” content=”10;url=t-rural.htm” /**>

* + **http-equiv=”Expires”**. Indica cuando deja de ser válido el contenido de la web y obliga al navegador a volver a cachear la página. El valor de la fecha ha de estar en formato GMT. Ej: <**meta http-equiv=”expires” content=”Mon, 23 Sep 2013 12:00:00 GMT” /**>

También es necesario comentar que existen etiquetas meta para informar a los motores de búsqueda de Facebook y otras redes sociales semejantes a: <**meta property=”og:title” content=”Página de turismo” /**>

Por otro lado comentar aquí una curiosa iniciativa que tiene relación con el autor de la página web. Se trata de “**humanstxt.org**”, una web que promueve una web “más humana”. Visite su web y comprobará que en realidad se trata de una alternativa a la etiqueta <**meta name=”autor”/**>.

***Pero… ¡Dejémonos de teoría y realicemos la práctica 16.1!***

***Para asegurar estos conocimientos realice la primera ejercitación del capítulo 16***

EL METADATO VIEWPORT

Por su importancia para el Responsive Web Design y por lo tanto por la relevancia que tomará en los próximos años, nos interesa profundizar y trabajar un poco más con el metadato “**viewport**”.

**Viewport** es un metadato creado por Apple para facilitar el desarrollo de webs para sus dispositivos móviles. Actualmente se puede considerar un estándar utilizado para cualquier dispositivo móvil.

**Viewport** ajusta el contenido de la página al tamaño de dispositivos móviles. Hay que tener muy claro que el **viewport** no se corresponde con el tamaño real de la pantalla en píxeles sino al espacio útil donde se mostrará una página web. Por ejemplo, el primer iphone tenía 230px de ancho, pero su **viewport** simulaba un área de 980px. Lo que quiere decir que encajaba los 980px en los 230px. con lo que el contenido de la web se veía completo pero muy pequeño, obligando al usuario a hacer zoom para simplemente poder leer el texto. Y lo mismo sucede con el resto de dispositivos móviles cambiando las cantidades de **viewport** entre unos y otros.

Para solucionar esto podemos utilizar los siguientes valores en la etiqueta **meta name=”viewport”**:

* + **width**. Define el ancho, en píxeles, del **viewport**. Puede ser un entero positivo o la cadena de texto “**device-width**”. En este último caso se toma el ancho del dispositivo.

Ej: <**meta name=”viewport” content=”width=320”**>, o

<**meta name=”viewport” content=”width=device-width”**>

Para diseñar webs móviles, esta última opción es la mejor pues automáticamente nos va a arrancar la web con el ancho máximo que tenga el dispositivo, sea el que sea.

* + **eigth**. Define la altura, en píxeles, del **viewport**. Puede ser un entero positivo o la cadena de texto “**device-height**”.
  + **initial-scale**. Define el ratio entre el ancho del dispositivo y el tamaño del **viewport**. Puede ser un número positivo entre 0.0 y 10.0.
  + **maximum-scale**. Define el valor máximo del zoom; este debe ser mayor o igual a la **minimun-scale** o el comportamiento será indeterminado. Puede ser un número positivo entre 0.0 y 10.0.
  + **minimun-scale**. Define el valor mínimo del zoom; este debe ser menor o igual a la **maximun-scale** o el comportamiento será indeterminado. Puede ser un número positivo entre 0.0 y 10.0.

Ej: <**meta name=”viewport” content=”width=device-width, maximun-scale=1, minimun-scale=1 /**>.

Con esta declaración anularíamos el zoom del dispositivo móvil.

* + **user-scalable**. Si tiene el valor “**no**”, el usuario no podrá hacer zoom en la página. El valor por defecto es “**yes**”.

Ej: <**meta name=”viewport” content=”width=device-width, user-scalable=no” /**>.

Produce el mismo efecto que el ejemplo anterior.

***Pero… ¡Veámoslo en la práctica 16.2!***

Para diseñar nuestras web según Responsive Web Design habitualmente utilizaremos este metadato.

También debemos conocer que a la hora de escribir este manual el **W3C** está trabajando en la directiva **@viewport** que anulará la meta viewport transfiriéndola a **CSS**.

***Asegure estos conocimientos realizando la segunda ejercitación del capítulo 16***

DISEÑO WEB FLEXIBLE O FLUIDO

La segunda pata del **Responsive Web Design** es la web flexible o fluida. Para ello el secreto está en diseñar con medidas flexibles, utilizando por tanto unidades en porcentaje, rem o em y olvidando las medidas en px y cm. Siempre trabajando sobre **CSS**.

Esto supone un cambio de mentalidad en el diseñador. Hasta ahora hemos aprendido a diseñar webs para pantallas de ordenador, con unas medidas fijas. A partir de ahora tendremos que pensar en móvil, en varios tamaños de pantalla y en porcentajes. Es más, lo lógico, sería diseñar primero la página web para la pantalla del móvil para luego ir adaptando el diseño a pantallas de sobremesa.

Para diseñar una web flexible debemos de tener en mente estas 10 normas básicas:

* Para cambiar una web con medidas fijas a porcentajes debemos aplicar la fórmula básica del Responsive Web Design que es “**target / contexto**”, es decir **dividir lo que queremos modificar entre su contenedor**.
* Tenemos que sumar y restar para completar el 100% en las diferentes medidas.
* Aplicar también la flexibilidad a las imágenes, videos... para ello utilizaremos, entre otras, la propiedad “**max-width: 100%**”.
* Para la tipografía usar 100% en el body y “em” para el resto de la página. Aunque están surgiendo nuevas unidades como “rem” o “vw” que también pueden ser útiles.
* No olvidar la fórmula básica de las fuentes flexibles: **100% = 1em = 12pt. = 16px**.
* Tener en cuenta que no solamente hay pantallas pequeñas de dispositivos móviles. También hay muchos usuarios que navegan con grandes pantallas o incluso empiezan a proliferar las grandes televisiones con conexión a internet. Para ellos utilizar la propiedad **CSS** “**max-width**” para no pasarse de tamaño.
* Y al contrario, se puede utilizar la propiedad **CSS** “**min-width**” para que un elemento no se haga tan pequeño que no se vea bien.
* No utilizar posiciones absolutas. Si posicionamos un objeto por ejemplo 600px a la derecha, en una pantalla de 400px, dicho objeto quedaría fuera de ella.
* Para posicionar los elementos en la web utilizar la propiedad **CSS** “**display: inline-block**” junto con “**float**”.
* Solo utilizaremos las medidas fijas, por ejemplo, px, para establecer los límites, el tamaño mínimo o máximo, de un elemento.

***Pero… ¡Veámoslo realizando la práctica 16.3!***

LA PROPIEDAD BOX-SIZING

Incluimos esta propiedad **CSS** a estas alturas del curso por su importancia para el **Responsive Web Design**. **Box-sizing** indica al navegador cómo queremos que calcule el espacio total ocupado por un elemento incluyendo o no en el ancho total el valor del padding y del border. Su sintaxis es:

**box-sizing: content-box / border-box;**

A tener en cuenta:

* **content-box:** Es el valor por defecto. La anchura de caja se calcula como ya conocemos: width + padding + marging + border.
* **border-box**: Fuerza al navegador a incluir en el ancho de la caja los valores de las propiedades padding y border. Es decir el padding y el border pasan a formar parte de la anchura de la caja. La anchura de la caja se calcula: width + marging.
* A la hora de escribir este manual hay que utilizar los prefijos privativos con esta propiedad.

Decimos que esta propiedad es importante para el **Diseño Web Adaptativo** porque si establecemos un ancho de 100% para un elemento para que ocupe todo el ancho de su contenedor, al añadir su padding y border, necesariamente se desbordará. Y calcular la anchura del padding y border en porcentaje puede ser bastante complicado. Utilizar el valor “**border-box**” de la propiedad “**box-sizing**” nos simplifica enormemente los cálculos.

***Asegure estos conocimientos realizando la tercera ejercitación del capítulo 16***

LOS MEDIA QUERIES

La tercera pata del **Responsive Web Design** son los **Media Queries**.

Como hemos podido comprobar el diseño fluido a base de porcentajes está muy bien hasta que nos encontramos con anchos de pantalla realmente pequeños como por ejemplo la de un móvil. Llega un momento en que los elementos se descolocan porque no caben y puede acabar todo en un caos. En este caso es probable que tengamos que prescindir de ciertos elementos de la web o situarlos en un lugar diferente. Para ello utilizaremos los **Media Queries**.

Las hojas de estilo **CSS** permiten tener representaciones diferentes según el medio de salida: pantalla, impresora, lector braille, smartphone, etc, etc… Para ello disponemos del atributo “**media**” que se agrega en la declaración del estilo y nos permite diseñar hojas de estilo diferentes según los medios de salida.

La manera de indicar los estilos para cada medio de salida depende de la declaración de estilos:

* **1.- Declaración de estilos interna**. Las propiedades **CSS** se indican en la cabecera del documento **HTML** como en el siguiente ejemplo:

**<head>**

**<style type=”text/css” media=”print”>**

**/\* Reglas de estilo que se aplican al documento impreso \*/**

**</style>**

**<style type=”text/css” media=”screen”>**

**/\* Reglas de estilo que se aplican en la visualización en pantalla \*/**

**</style>**

**</head>**

* **2.- Declaración de estilos externa**. Las propiedades **CSS** se indican en un archivo diferente al del **HTML**, como en el siguiente ejemplo:

**<link rel=”stylesheet” type=”text/css” href=”impresora.css” media=”print”>**, o

**<link rel=”stylesheet” type=”text/css” href=”pantalla.css” media=”screen”>**.

Los diferentes medios de salida que entiende el parámetro “**media**” son:

|  |  |
| --- | --- |
| **media=”all”** | Los estilos definidos se aplican a todos los tipos de medio. Es el valor por defecto. |
| **media=”screen”** | Los estilos se aplican a la visualización en las pantallas. |
| **media=”print”** | Los estilos se aplican a la impresión en papel. |
| **media=”projection”** | Los estilos se aplican a webs vistas mediante proyectores. |
| **media=”handheld”** | Los estilos se aplican a la visualización en ordenadores de bolsillo o smartphones. |
| **media=”speech”** | Los estilos se aplican a sintetizadores de voz. |
| **media=”braille”** | Los estilos se aplican a los lectores braille. |
| **media=”embossed”** | Los estilos se aplican a las impresoras de perforación braille. |
| **Media=”tty”** | Los estilos se aplican cuando se utiliza un dispositivo con un tamaño fijo de carácter, como un teletipo. |
| **Media=”tv”** | Los estilos se aplican cuando se accede a una web desde una televisión. |

En un futuro, necesariamente, se incrementará esta lista, a medida que aparezcan nuevos dispositivos o medios en los que se puedan visualizar páginas webs. Por ejemplo: “**3d-glasses**”.

Llegados a este punto es preciso mencionar que debemos prestar una especial atención al **media=”print”**. Todos hemos comprobado en alguna ocasión que al imprimir una web, lo que hemos obtenido en papel, era bastante decepcionante: fondos negros que nos gastan mucha tinta, menús de navegación inútiles en papel, tipos de letra ilegibles, texto cortado porque no cabe en la página o papel echado a perder…

Por ello consideramos importante crear reglas de estilo solo para la versión impresa de nuestras webs que tengan en cuenta entre otras cosas:

* Eliminar los elementos inútiles o molestos a la hora de imprimir la página.
* Repasar el formato del contenido de texto pensando en su lectura en papel.
* Agregar información, habitualmente reservada a la visualización en pantalla.
* Controlar los fondos de colores o de imágenes.
* Controlar el color, el tipo y tamaño de la letra.
* Controlar los enlaces.
* Controlar las imágenes.
* Comprobar si es necesario el menú de navegación.
* Quitar las animaciones.
* Controlar si merece la pena que se impriman los formularios.
* Controlar los márgenes
* Etc, etc…

Debemos tener en cuenta que una página web en pantalla debe resultar atractiva al ojo mientras que una información en papel debe ser atractiva al espíritu. En una página impresa se busca riqueza de información o contenido.

A modo de enumeración, algunas propiedades **CSS** relacionadas con la impresión son:

|  |  |  |
| --- | --- | --- |
| **Propiedad** | **Utilidad** | **Valores** |
| Page-break-before | Fuerza un salto de página antes de algún elemento de la página. | Always / avoid / auto |
| Page-break-after | Fuerza un salto de página después de algún elemento de la página. | Always / avoid /auto |
| Page-break-inside | Evita el salto de página en el elemento designado | Always/ avoid /auto |
| Selector: @page {  /\* propiedades para imprimir \*/  }  @page :first {  /\* propiedades para la primera página \*/  } | Permite especificar diferentes propiedades para un medio impreso. Se debe definir al principio del **CSS**. Además de las propiedades habituales son propias de este selector:   * Size. * Size: portrait * Size: landscape * Y las pseudoclases: :first, :left, y :right para indicar primera página, páginas izquierdas o páginas derechas. |  |

* **3.- Declaración de estilos con la regla @media.**

Las reglas **@media**, también conocidas como **Media Queries**, implementadas en **CSS3**, nos permiten especificar el conjunto de reglas **CSS** que deben aplicarse a cada medio de salida pero condicionados a unos determinados criterios.

Las reglas **@media** son de verdadera utilidad para el **Responsive Web Design** ya que podemos especificar que el medio de salida sea la pantalla pero además de un determinado tamaño.

La sintaxis de la regla **@media** es:

**@media medio\_de\_salida and (criterio) {**

**/\* Reglas CSS que se desean aplicar cuando**

**se cumpla ese medio\_de\_salida y el (criterio) \*/**

**}**

A tener en cuenta:

* **Medio\_de\_salida**, puede ser cualquiera de los que hemos indicado más arriba: all, print, screen, etc, etc…
* **Criterio**, indica una propiedad del medio de salida que lo especifica. Estas propiedades pueden ser de los siguientes tipos:
  + **Aspect-ratio**, especifica el dispositivo por la relación de aspecto. Ej: 16:9. Las propiedades de este tipo son:
    - Aspect-ratio
    - Max-aspect-ratio
    - Min-aspect-ratio
    - Device-aspect-ratio
    - Max-device-aspect-ratio
    - Min-device-aspect-ratio
  + **Width** y **Height**, especifica el dispositivo por el ancho o el alto de su área de visualización. Las propiedades de este tipo son:
    - Device-height
    - Max-device-height.
    - Min-device-height.
    - Device-width.
    - Max-device-width.
    - Min-device-width.
    - Height.
    - Max-height.
    - Min-height.
    - Width.
    - Max-wdith.
    - Min-width.
  + **Orientation**, especifica el dispositivo si está en vertical u horizontal. Las propiedades de este tipo son:
    - Orientation portrait.
    - Orientation landscape.
  + **Resolution**, especifica el dispositivo por su densidad de píxeles. Las propiedades de este tipo son:
    - Resolution.
    - Max-resolution.
    - Min-resolution.
  + **Color, color-index** y **monochrome**, especifica el dispositivo de salida por la cantidad de colores que admite. Las propiedades de este tipo son:
    - Color
    - Max-color.
    - Min-color.
    - Color-index.
    - Max-color-index.
    - Min-color-index.
    - Monochrome
    - Max-monochrome.
    - Min-monochrome.

Aunque a primera vista pueda parecer lo mismo, hemos de distinguir entre la resolución de pantalla y el ancho de ventana del navegador. Para entenderlo piense que en un ordenador podemos tener una resolución de pantalla de 1980px de ancho pero si reducimos la ventana, el ancho de ventana del navegador es menos que la resolución de pantalla.

Los media queries actúan por defecto sobre la vista del navegador con la excepción de aquellos que especifican en su nombre la palabra “**device**” que actúan sobre toda la ventana del dispositivo.

Algunos ejemplos de sintaxis de la regla **@media** son:

* Para una pantalla de móvil de 320px:

**@media screen and (max-width: 320px) {**

**/\* Reglas CSS que se desean aplicar para este tamaño de pantalla \*/**

**}**

* Para una pantalla de ancho superior a 1.200px, la regla podía ser:

**@media screen and (min-width: 1200px) {**

**/\* Reglas CSS que se desean aplicar para este tamaño de pantalla \*/**

**}**

* Un poco más complicado… para maquetar para una famosa tablet en su posición horizontal podría ser:

**@media only screen and (min-device-width: 768px) and (max-device-width: 1024px) and (orientation: landscape) {**

**/\* Reglas CSS que se desean aplicar para este tamaño de pantalla \*/**

**}**

***Pero… ¡Pasemos a la práctica realizando la número 16.4!***

El problema que se nos plantea es la infinidad de tamaños de pantalla que hay en la actualidad y más teniendo en cuenta que pueden utilizarse en vertical u horizontal. Nosotros hemos tenido en cuenta una sola media querie en nuestra práctica. Intentar abarcar todos los dispositivos es una locura. Algunos autores recomiendan optimizar las webs para estos anchos de pantalla:

* 320px
* 480px
* 600px
* 768px
* 900px
* 1200px

Pero lo habitual es maquetar para tres tipos de pantalla:

* Móviles en vertical: donde la pantalla solo permite lectura y navegación muy simplificada y podemos incluirla entre 320px y 400px.
* Móviles en horizontal y tablets: donde ya se permite una visualización horizontal clásica con pantallas de 400px, 600px y hasta 800px.
* Dispositivos de sobremesa: donde ya contamos con webs normales mayores de 800px.

Es preferible utilizar la regla **@media** a la declaración de estilos externa ya que así mejoramos el rendimiento de la página al no tener que cargar archivos **CSS** adicionales para los diferentes dispositivos.

FONT-SIZE EN FUNCIÓN DEL TAMAÑO DE LA VENTANA.

Recientemente el **W3C** ha añadido algunas unidades nuevas para poder dar tamaño a los textos en función de las medidas de su caja contenedora inicial o del tamaño de la ventana. Con ello conseguimos que las fuentes vayan cambiando de tamaño en función del tamaño de la ventana. Estas unidades, conocidas como unidades viewport, son:

* **vw**: Igual al 1% de la anchura de la pantalla.
* **vh**: igual al 1% de la altura de la pantalla.
* **vmin**: Igual a la más pequeña de “vw” o “vh”.
* **vmax**: Igual a la más grande de “vw” o “vh”.

***¡Para ver el resultado realice la práctica 16.4!***

De estas tres unidades “**vw**” es la más factible para ser utilizada en el **Responsive Web Design**.

También debemos comentar que a la hora de realizar este manual, el **W3C** está trabajando un una nueva etiqueta **HTML5** llamada <**picture**> que permita mostrar imágenes diferentes en función del tipo de pantalla en que se visualice. Esto permitirá incluir imágenes con mayor o menor peso adaptándose a los diferentes dispositivos móviles y sacar un mayor partido de las nuevas pantallas de alta resolución “retina display”.

***Asegure estos conocimientos realizando la cuarta ejercitación del capítulo 16***

CAPITULO 17.- PUBLICACIÓN DE UNA WEB

Y OTROS TEMAS

CONTADORES DE VISITAS

Insertar un contador de vistas en nuestra web es muy sencillo. Contamos con dos opciones:

1) Utilizar el sistema de contadores del proveedor de alojamiento de la página web. Generalmente los proveedores de alojamiento ofrecen alguna posibilidad para insertarlos, en este caso, lo lógico es seguir sus instrucciones.

2) Utilizar un servicio de contadores gratuitos. Por ejemplo en <http://www.websmultimedia.com/contador-de-visitas-gratis> encontrará contadores gratis.

En los dos casos lo que hay que hacer es colocar el código **HTML** que proporciona el programa del contador. Habitualmente se trata de una etiqueta <**img**> cuyo **SRC** se enlaza con el programa contador.

***¡Realice ahora la práctica 17.1!***

AÑADIR UN FAVICON

Un “**favicon**”, también conocido como “icono de página” es una pequeña imagen que representa una página o un sitio web. Es como el logotipo de cada web. Se suele mostrar en la barra de direcciones del navegador y a la izquierda del nombre de la web en la barra de título o pestaña de la ventana del navegador.

El **favicon** también se utiliza para identificar las páginas en las listas de marcadores o favoritos y en el historial de páginas visitadas.

Para añadir un **favicon** se necesita una imagen de 16x16 ó 32x32 píxeles con formato .png, .gif o .ico y añadir en el <**head**> de la página el siguiente código:

**<link rel="icon" type="image/png" href=”n\_del\_icono.png" >**

***¡Veámoslo en la práctica 17.2!***

ATRIBUTOS GLOBALES DE HTML5

A lo largo de este curso, cuando hemos ido estudiando cada elemento **HTML**, hemos ido indicando sus atributos, los que le son propios. Hay una serie de atributos, llamados atributos globales, que están disponibles para todos los elementos **HTML** y que, aunque algunos ya los hemos utilizado, debería conocer. Son los siguientes:

|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **accesskey** | Especifica una tecla de acceso directo para activar/hacer focus en un elemento. |
| **class** | Especifica el nombre de la clase de un elemento para aplicar estilos css. |
| **contenteditable** | Especifica si el contenido de un elemento es editable o no. |
| **contextmenu** | Muestra  un menú contextual al hacer clic con el botón derecho del ratón. |
| **dir** | Especifica la dirección del texto contenido en un elemento. |
| **draggable** | Indica si un elemento es arrastrable o no. |
| **dropzone** | Especifica si los datos arrastrados son copiados, movidos o vinculados, cuando se dejan caer sobre un elemento. |
| **hidden** | Indica si el elemento es visible o no. |
| **id** | Identificador único del elemento en la página. |
| **lang** | Especifica el idioma del contenido de un elemento. |
| **spellcheck** | Especifica si el elemento debe tener su ortografía y gramática comprobada o no. |
| **style** | Aplica un estilo css al elemento. |
| **tabindex** | Especifica el orden de tabulación de un elemento. |
| **title** | Especifica información adicional acerca de un elemento. |
| **translate** | Indica si un elemento se traduce o no. (Es para evitar que código de programación o similares sean traducidos automáticamente con el resto de la página). |
| **data-\*** | Permite añadir atributos definidos por el usuario.  Por ejemplo: **<p data-calorias=”50”>kiwi</p>** |

ATRIBUTOS DE EVENTOS

Los eventos de **HTML** permiten a los desarrolladores web agregar interactividad entre el sitio web y el visitante ejecutando un programa, generalmente en **JavaScript**, cuando el visitante de la web realiza una acción como por ejemplo hacer doble clic sobre un elemento o pasar con el puntero del ratón sobre un elemento. Esto último lo hemos realizado con la pseudoclase “**:hover**” de **CSS** pero lo interesante de los atributos de eventos es que son mucho más numerosos y que podemos aplicarles verdaderos programas.

Hemos utilizado atributos de eventos como, “**ondragstart**”, “**ondragover**” u “**ondrop**”, cuando creamos el puzle en la web “**t-rural.htm**”. Como pudimos ver entonces, los atributos de eventos se indican como el resto de atributos de los elementos **HTML** siguiendo el formato:

**<etiqueta atributo\_de\_evento="código JavaScript">**

A modo de ejemplo en el siguiente código, definimos un párrafo que cambia el color de su texto a rojo cuando pasamos por encima con el puntero del ratón, y lo devuelve a negro cuando lo retiramos:

**<p onmouseover="this.style.color='red'" onmouseout="this.style.color='black'">**

**Este es el texto que cambiaría de color al pasar el ratón.**

**</p>**

Sacaremos el máximo partido a los atributos de eventos de **HTML** cuando dominemos **JavaScript** pero para que nos vayan sonando, estos son los atributos de eventos en **HTML5**:

|  |  |
| --- | --- |
| **Atributos de Eventos globales y de ventana.** | |
| **Atributo de Evento** | **Descripción** |
| **onafterprint** | Se ejecutará un script después de imprimirse el documento. |
| **onbeforeprint** | Se ejecutará un script antes de imprimir el documento. |
| **onbeforeunload** | Se ejecutará un script antes de que se cargue el documento. |
| **onerror** | Se ejecutará un script cuando se produzca un error. |
| **onhaschange** | Se ejecutará un script cuando el documento haya cambiado. |
| **onload** | Se ejecutará un script después de que la página termine de cargarse. |
| **onmessage** | Se ejecutará un script cuando se active el mensaje. |
| **onoffline** | Se ejecutará un script cuando el documento está desconectado. |
| **ononline** | Se ejecutará un script cuando el documento se ponga en línea. |
| **onpagehide** | Se ejecutará un script cuando la ventana está oculta. |
| **onpageshow** | Se ejecutará un script cuando la ventana se haga visible. |
| **onpopstate** | Se ejecutará un script cada vez que la entrada del historial cambie. |
| **onredo** | Se ejecutará un script cuando el documento realiza un “rehacer”. |
| **onresize** | Se ejecutará un script cuando se redimensiona la ventana del navegador. |
| **onstorage** | Se ejecutará un script cuando se actualiza el almacenamiento web. |
| **onundo** | Se ejecutará un script cuando el documento realiza un “deshacer”. |
| **onunload** | Se ejecutará un script cuando el usuario abandone el documento o la ventana del navegador se ha cerrado. |

|  |  |
| --- | --- |
| **Eventos de Formulario** | |
| **Atributo de Evento** | **Descripción** |
| **onblur** | Se ejecutará un script en el momento en que el elemento pierde el foco. |
| **onchange** | Se ejecutará un script cuando se cambia el valor del elemento. |
| **oncontextmenu** | Se ejecutará un script cuando se activa el menú contextual. |
| **onfocus** | Se ejecutará un script cuando el elemento obtiene el foco. |
| **onformchange** | Se ejecutará un script cuando un formulario cambie. |
| **onforminput** | Se ejecutará un script cuando el usuario escribe en un formulario. |
| **oninput** | Se ejecutará un script cuando un usuario escribe en un elemento. |
| **oninvalid** | Se ejecutará un script cuando un elemento no es válido. |
| **onselect** | Se ejecutará un script después de que un fragmento de texto haya sido seleccionado en un elemento. |
| **onsubmit** | Se ejecutará un script cuando se envía un formulario. |

|  |  |
| --- | --- |
| **Eventos de teclado** | |
| **Atributo de Evento** | **Descripción** |
| **onkeydown** | Se ejecutará un script cuando el usuario está presionando una tecla. |
| **onkeypress** | Se ejecutará un script cuando el usuario pulsa una tecla. |
| **onkeyup** | Se ejecutará un script cuando el usuario suelte una tecla. |
| **Eventos del mouse** | |
| **Atributo de Evento** | **Descripción** |
| **onclick** | Se ejecutará un script cuando se haga clic con el ratón en el elemento. |
| **ondblclick** | Se ejecutará un script cuando el ratón haga doble clic en el elemento. |
| **ondrag** | Se ejecutará un script cuando se arrastre un elemento. |
| **ondragend** | Se ejecutará un script al final de una operación de arrastre. |
| **ondragenter** | Se ejecutará un script cuando un elemento ha sido arrastrado a un destino válido. |
| **ondragleave** | Se ejecutará un script cuando un elemento deja un destino válido. |
| **ondragover** | Se ejecutará un script cuando un elemento está siendo arrastrado por un destino válido. |
| **ondragstart** | Se ejecutará un script en el inicio de una operación de arrastre. |
| **ondrop** | Se ejecutará un script cuando se suelte el elemento arrastrado. |
| **onmousedown** | Se ejecutará un script cuando se pulse un botón del ratón sobre un elemento. |
| **onmousemove** | Se ejecutará un script cuando el usuario mueve el ratón sobre un elemento. |
| **onmouseout** | Se ejecutará un script cuando el puntero del ratón se mueva fuera de un elemento. |
| **onmouseover** | Se ejecutará un script cuando el puntero del ratón se sitúa sobre un elemento. |
| **onmouseup** | Se ejecutará un script cuando se suelta el botón del ratón sobre un elemento. |
| **onmousewheel** | Se ejecutará un script cuando se gira la rueda del ratón. |
| **onscroll** | Se ejecutará un script cuando se desplaza la barra de desplazamiento de un elemento. |

|  |  |
| --- | --- |
| **Eventos con elementos Multimedia** | |
| **Atributo de Evento** | **Descripción** |
| **OnAbort** | Se ejecutará un script al abortar la ejecución multimedia. |
| **oncanplay** | Se ejecutará un script cuando el archivo multimedia este listo para comenzar a reproducirse. |
| **oncanplaythrough** | Se ejecutará un script cuando un archivo multimedia se reproduzca entero hasta el final sin detenerse para cargar partes del archivo. |
| **ondurationchange** | Se ejecutará un script cuando se cambie la longitud del multimedia. |
| **onemptiedv** | Se ejecutará un script cuando de repente el archivo multimedia no es accesible por errores de red o de carga… |
| **onended** | Se ejecutará un script cuando la reproducción llega al final. (Un evento útil para los mensajes como "gracias por escucharme"). |
| **onerror** | Se ejecutará un script cuando se produzca un error al cargar el archivo. |
| **onloadeddata** | Se ejecutará un script cuando se carguen los datos de archivos multimedia. |
| **onloadedmetadata** | Se ejecutará un script cuando se cargan los metadatos de un archivo multimedia como dimensiones, duración, estilo, autor... |
| **onloadstart** | Se ejecutará un script cuando el navegador empiece a cargar los datos del archivo multimedia. |
| **onpause** | Se ejecutará un script cuando la reproducción multimedia este en pausa ya sea por el usuario o mediante programación. |
| **onplay** | Se ejecutará un script cuando la reproducción multimedia esté lista para comenzar. |
| **onplaying** | Se ejecutará un script cuando realmente haya empezado la reproducción del archivo multimedia. |
| **onprogress** | Se ejecutará un script cuando el navegador este obteniendo los datos del archivo multimedia. |
| **onratechange** | Se ejecutará un script cada vez que se cambie la velocidad de reproducción como cuando un usuario cambia a un modo de cámara lenta o avance rápido. |
| **onreadystatechange** | Se ejecutará un script cada vez que el archivo multimedia cambia al estado “listo”. |
| **onseeked** | Se ejecutará un script cuando el atributo de búsqueda se establece en “false”. Indica que la búsqueda ha terminado |
| **onseeking** | Se ejecutará un script cuando el atributo de búsqueda se establece en “true”. Indica que la búsqueda está activa. |
| **onstalled** | Se ejecutará un script cuando el navegador no sea capaz de recuperar los datos de los archivos multimedia por cualquier motivo. |
| **onsuspend** | Se ejecutará un script cuando se detiene el recuperar los datos de los archivos multimedia por cualquier razón antes de que estén completamente cargados. |
| **ontimeupdate** | Se ejecutará un script cuando la posición de reproducción haya cambiado como cuando el usuario avanza rápidamente a un punto diferente del archivo multimedia. |
| **Onvolumechangev** | Se ejecutará un script cada vez que se cambie el volumen. Incluido poner el volumen en “silencio”. |
| **onwaiting** | Se ejecutará un script cuando la reproducción del multimedia se ha detenido pero se espera que reanude como cuando la reproducción hace una pausa para cargar más datos. |

SPRITES CSS

 **CSS Sprites** es una técnica de diseño que consiste en utilizar un solo archivo de imagen compuesto por todas las imágenes que utilizamos en la web. Esta única imagen se va posicionando haciendo uso de las propiedades “**background-position**” de **CSS** para mostrar sólo la imagen que nos interesa en cada parte de la web.

A la izquierda mostramos uno de los **Sprite CSS** utilizado por Google que lo ha ido cambiando, según sus necesidades, a lo largo de los años.

La principal ventaja de la técnica de los **CSS Sprites** es la mejora en la velocidad de descarga de la web pues no es lo mismo bajar 15 archivos de imágenes que una sola imagen que vamos posicionando. Además sólo se hace una petición al servidor en vez de 15.

Ultimamente, con la llegada del Responsive Web Desing y su tendencia hacia el “diseño flat”, el uso de los **Sprites CSS** se está cuestionando y los diseñadores web optan por mostrar iconos mediante fuentes tipográficas como las que puede ver en <http://css-tricks.com/flat-icons-icon-fonts/>

PUBLICAR NUESTRA PÁGINA WEB

Para publicar nuestra web en Internet necesitamos:

1.- Un dominio que puede ser de pago como [www.academiafrayluis.es](http://www.academiafrayluis.es) o gratuito, en este caso, lo normal es que sea un subdominio tal como [www.alumno1.academiafrayluis.es](http://www.alumno1.academiafrayluis.es). Si nuestra web es profesional lo lógico es adquirir un dominio propio que normalmente se paga mediante anualidades. Para practicar nosotros emplearemos un subdominio gratuito.

2.- Un proveedor de hosting donde alojar los archivos que componen nuestra web. Al igual que en el caso anterior, podemos utilizar un hosting de pago o uno gratuito. Lógicamente un hosting de pago tendrá más servicios y sobretodo más capacidad de almacenamiento. Para practicar nosotros utilizaremos el hosting gratuito de la empresa **HOSTINGER**.

3.- Un programa FTP que nos permite trasladar todos los archivos de la web desde nuestro disco duro local al del proveedor del hosting. Nosotros utilizaremos “**Filezilla**” que es uno de los más habituales en todos los proveedores hosting. Esta transferencia de archivos se realiza en **modo autentificado**, es decir, introduciendo un nombre de usuario y contraseña para evitar que otras personas puedan publicar en nuestro espacio web.

***¡Vayamos a la práctica realizando la número 17.3!***

RECOPILACIÓN DE BUENAS PRÁCTICAS CON HTML Y CSS

A modo de enumeración esta es una recopilación de algunas buenas prácticas:

Para **HTML**:

* Recordar cerrar las etiquetas.
* Nombrar las etiquetas siempre en minúsculas.
* Utilizar ficheros externos para **CSS** y **JavaScript**.
* Enlazar los **CSS** externos al principio del documento en su <**head**>.
* Enlazar los **JavaScript** externos al final del documento justo antes de la etiqueta <**/body**>.
* Ser cuidadoso con la compatibilidad de los elementos que usemos. Usar **www.caniuse.com**.
* Usar el atributo “**alt**” en todas las imágenes.
* Validar el código.
* Usar una “**chuletilla**” con las etiquetas **HTML5**.

Para **CSS**:

* Primero definir los estilos para etiquetas **HTML** como **header**, **footer**, **div**, **a**, **p**, etc.
* Después definir los estilos para indentificadores “**id**”.
* Por último definir los estilos para las clases “**class**”.
* Ordenar las reglas de estilos conforme van apareciendo en el **HTML**.
* Ordenar las propiedades en cada regla de estilo por orden alfabético.
* Utilizar tabulaciones en las propiedades **CSS** para facilitar su lectura.
* Especificar el alto y ancho de las imágnes.
* Definir un tamaño de fuente global en el body.
* Evitar el uso de muchos selectores descendentes como: **section div ul li a { … }**.
* Reducir el uso de selectores **CSS3** como **:first-child**, facilitan mucho la vida pero consumen muchos recursos.
* Aprovechar las ventajas en cascada de **CSS**. No es necesario escribir: **nav ul li a {Font-weight: 600;}**. Nos vale lo mismo con: **nav {Font-weight: 600;}**.
* Recordar poner las reglas de estilo de reset antes de cualquier otra regla de estilo.
* Comprobar el diseño en varios navegadores.
* Utilizar comentarios para separar las secciones de la hoja de estilos.
* Reduccir el número de archivos **CSS** de la página. Si es posible, combinar todos los archivos **CSS** individuales en un único archivo.
* Una vez que hemos asegurado el acceso completo y correcto a toda la información de la página independientemente del navegador utilizado por el usuario, utilizar las características más modernas de **CSS** para que se puedan disfrutar en los navegadores más modernos.
* Usar herramientas como **Firebug** en Firefox para el depurado de errores **CSS**.

***Asegure estos conocimientos realizando la ejercitación del capítulo 17***

***Demuestre todo lo que ha aprendido realizando la ejercitación final del curso.***