UD2. HOJAS DE ESTILO

RA2. Crea interfaces web homogéneos definiendo y aplicando estilos.

ÍNDICE

2.1. Introducción

2.2. Ventajas de usar CSS

2.3. ¿Cómo funciona CSS?

2.4. Estructura de las reglas CSS

ÍNDICE

2.5. Tres maneras de añadir estilos CSS a un HTML

2.6. La cascada de CSS

2.7. Tipos de elementos: bloque y en línea

2.8. Modelo de cajas

ÍNDICE

2.9. Selectores

2.10. Unidades de medida en CSS

2.11. Posicionamiento

2.12. CSS con tablas y listas

2.13. Primeros pasos con SASS

2.1. Introducción

Las hojas de estilo en cascada (CSS, Cascading Style Sheets) son un estándar W3C que define la presentación de los documentos Web.

Lo más importante es que con CSS se mantienen las instrucciones de presentación separadas del contenido del documento HTML.

2.2. Ventajas de usar CSS

2.2. VENTAJAS DE USAR CSS

- Mayor control en el diseño de las páginas
- Menos trabajo
- Documentos más pequeños
- Documentos mucho más estructurados
- El HTML de presentación está en vías de desaparecer.
- Tiene buen soporte

2.3. ¿CÓMO FUNCIONA CSS?

2.3. ¿CÓMO FUNCIONA CSS?

1. Con HTML se crea la estructura de la página web.
2. Luego hay que escribir las reglas de estilo para definir el aspecto ideal de todos los elementos.
3. Hay que vincular los estilos al documento.

¿Qué crees que hace el siguiente código?

```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

2.4. ESTRUCTURA DE LAS REGLAS CSS

Todas las reglas CSS siguen misma estructura:

```
<selector> {
  <propiedad> : <valor>;
  <propiedad> : <valor>;
  <propiedad> : <valor>;
  ...
}
```

El conjunto de **‹propiedad› : ‹valor›;** también se llama **declaración.**

Hay que destacar que cuando a una propiedad como *font-family* se le asignan dos posibles valores estos van separados por comas (,) y a la hora de establecer uno de ellos intentará siempre hacerlo de izquierda a derecha.

2.5. TRES MANERAS DE AÑADIR ESTILOS CSS A UN HTML

2.5. TRES MANERAS DE AÑADIR ESTILOS CSS A UN HTML

Existen tres maneras de añadir CSS sl HTML:

1. Como atributo
2. Usando una etiqueta *style*
3. Usando un fichero *.css* externo

Como atributo (estilo en línea)

Se puede añadir información de estilo a un elemento concreto empleando el atributo *style* en la etiqueta del elemento HTML.

```
1   <body style="background-color:lightblue;">
```

Los estilos en línea contaminan el documento con información de presentación.

Debe emplearse con moderación, solo ocasionalmente, para ignorar reglas de nivel más alto.

Usando la etiqueta **style** (estilos incrustados)

Se puede incrustar un bloque de estilo en la parte superior (en la cabecera o *head)* del documento HTML empleando el elemento *style*.

```
<style>
body {
  background-color: lightblue;
}
</style>
```

Usando un fichero .css externo (estilos importados o externos)

Es el modo más potente de utilizar

Se reúnen todas las reglas de estilo en un documento de texto independiente y se crean vínculos a ese documento desde todas las páginas de un sitio.

De este modo, se pueden hacer cambios en el estilo de un modo homogéneo en todo el sitio editando la informción del estilo en un único documento.

Para hacerlo se emplea con el elemento link en el documento HTML.

Usando un fichero .css externo (estilos importados o externos)

index.html

```html
<head>
  <link rel="stylesheet" href="mystyle.css">
</head>
```

mystyle.css

```css
body {
  background-color: lightblue;
}
```

2.6. LA CASCADA DE CSS

La “cascada” de las hojas de estilo se refiere a lo que ocurre si varias fuentes de información de estilo quieren dar formato al mismo elemento de una página.

Cuando un navegador encuentra un elemento par el cual hay varias declaraciones de estilo, las ordena de la siguiente manera:

1. Según el origen de la hoja de estilo.
2. Según la especificidad de los selectores.
3. Orden de la regla para poder determinar cuál aplicar.

Según el origen de la hoja de estilo.

Los navegadores otorgan un peso distinto a las hojas de estilo que, ordenadas de menor a mayor peso, son:

1. Hojas de estilo del navegador: Son los que tiene el navegador por defecto.
2. Hojas de estilo del usuario: Son las que se pueden aplicar desde una extensión de navegador como Stylus.
3. Hojas de estilo del desarrollador.
4. Hojas de estilo del usuario con el uso de *!important*

.

Según la especificidad de los selectores

```
CSS

1   strong {color: red;}
2   h1 strong {color: blue;}
```

Cuanto más específico sea el selector se le dará más peso para ignorar las declaraciones en conflicto.

Según el orden

Cuando una hoja de estilo contiene varias reglas en conflicto de igual peso, sólo se tendrá en cuenta la que está en último lugar.

CSS

```
1   h1 {color: green;}
2   h1 {color: blue;}
3   h1 {color: red;}
```

¿Qué ocurre si añadimos varios estilos al mismo elemento?

Los estilos se procesan en cascada sobrescribiéndose unos a otros. De manera muy simplificada ordenamos de *mayor prioridad a menor* prioridad de la siguiente manera:

1. Importancia: Usando la palabra clave ! important
2. Origen
3. Especificidad:
    a. Estilos en línea.
    b. Seleccionando por ID
    c. Seleccionando por clase
    d. Seleccionando por elemento (o pseudo elementos)
    e. Seleccionando con el selector universal
4. Orden de aparición: El último prevalece.

2.7. Tipos de elementos: bloque y en línea

Un **elemento de bloque** en CSS siempre genera saltos de línea, antes y después de él.

Un **elemento en línea** aparecen en el flujo de la línea y sólo pasarán a otra línea si no tienen espacio.

En HTML, los párrafos (**p**), las cabeceras (**h1**), listas (**ol, ul, dl**) y contenedores (**div**) son los elementos de bloque más comunes, mientras que, el texto enfatizado (**em**), las anclas (**a**) y los elementos **span** son los elementos más comunes.

En CSS podrás indicarle al navegador cómo quieres que se vea en el documento empleando para ello los atributos **block** e **inline** de la propiedad **display**, independientemente de que un elemento sea de bloque o en línea.

La propiedad display

¿Qué diferencia hay entre elementos **block** y los elementos **inline**?

Por defecto, cada elemento HTML tiene un tipo de representación concreto. Como norma general, los elementos que se utilizan dentro de un párrafo, son de tipo **inline,** mientras que los que se utilizan para agrupar, son de tipo **block.** La propiedad *display* de CSS permite modificar el comportamiento de un elemento HTML.

- *block*: Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
- *inline*: Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora *width* o *height.*

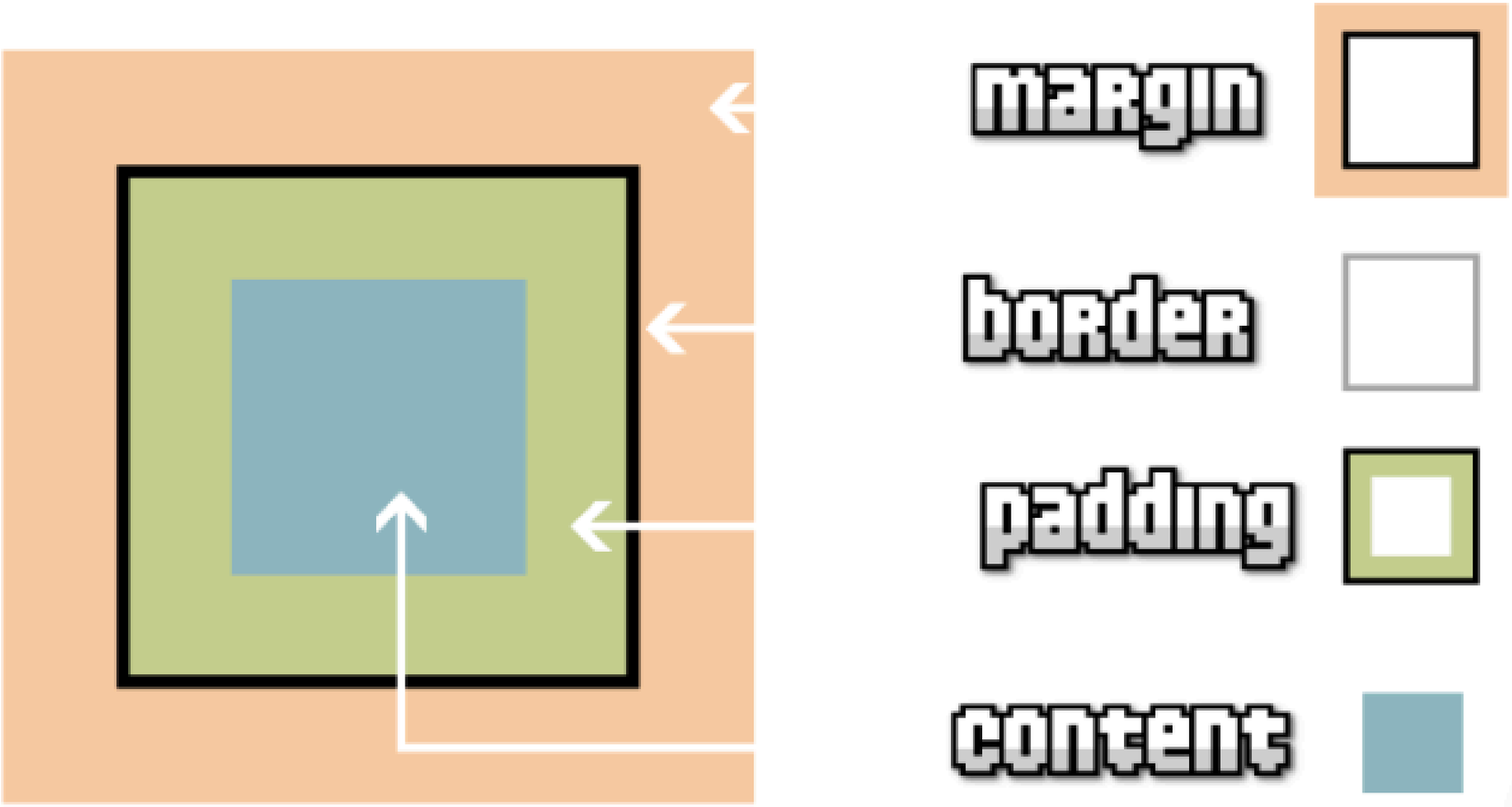
2.8. Modelo de cajas

INTRODUCCIÓN

El modelo de cajas es un concepto fundamental para compreder el funcionamiento de las hojas de estilos.
Según este modelo, todos los elementos de una página web generan una caja rectangular alrededor llamada “caja del elemento”

En estas cajas se pueden configurar propiedades como bordes, márgenes y fondos (entre otras). las cajas también se pueden emplear para posicionar los elementos y diseñar la página.

Las cajas de elementos, están hechas de cuatro componentes principales:



- Contenido (**content**), en azul, es la parte interior del elemento, excluyendo el relleno.
- Relleno (**padding**), en verde, es el espacio que rodea al contenido.
- Margen (**margin**), en naranja, es el espacio que rodea al borde, la parte más externa del elemento.
- Borde (**border**), en negro, es el límite que separa el interior del exterior del elemento.

A destacar:

- El relleno, los bordes y los márgenes son opcionales, por lo que, si ajustas a cero los valores se eliminarán de la caja.
- Cualquier color o imagen que apliques de fondo al elemento se extenderá por el relleno
- Los bordes se generan con propiedades de estilo que especifican su estilo, por ejemplo, sólido, grosor y color. Cuando el borde tiene huecos, el color o imagen de fondo aparecerá a través de los huecos.

A destacar:

- Los márgenes siempre son transparentes (el color del elemento padre se verá a través de ellos).
- Cuando definas el largo de un elemento estás definiendo el largo del área del contenido (los largos de relleno, de borde y de márgenes se sumarían a esta cantidad).
- Puedes cambiar el estilo de lados superior, derecho, inferior e izquierdo de una caja de un elemento por separado.

Modelo de cajas

CONTENIDO

El tamaño del contenido se define de la siguiente manera:

CSS

```
1   div {width:100px; height:200px; }
```

RELLENO

Podemos definir el relleno. El orden de los valores son: **top** (arriba), **right** (derecha), **bottom** (debajo) y **left** (izquierda), siempre en el sentido de las agujas del reloj y comenzando en las 12.

Establece un relleno para todos los elementos ***div***:

CSS

```
1   div {padding: 3px 20px 3px 20px;}
```

RELLENO

Al omitir un valor, asume que el valor del relleno a la izquierda es el mismo que el de la derecha:

```
CSS

1   div {padding: 3px 20px 3px; }
```

RELLENO

Al omitir dos valores, asume que el primer valor corresponde al relleno por encima y por debajo del área de contenido y, el segundo valor corresponde al relleno a la derecha y a la izquierda:

CSS

```
1   div {padding: 3px 20px; }
```

RELLENO

Al omitir tres valores, asume que ese valor es el mismo par todos.

CSS

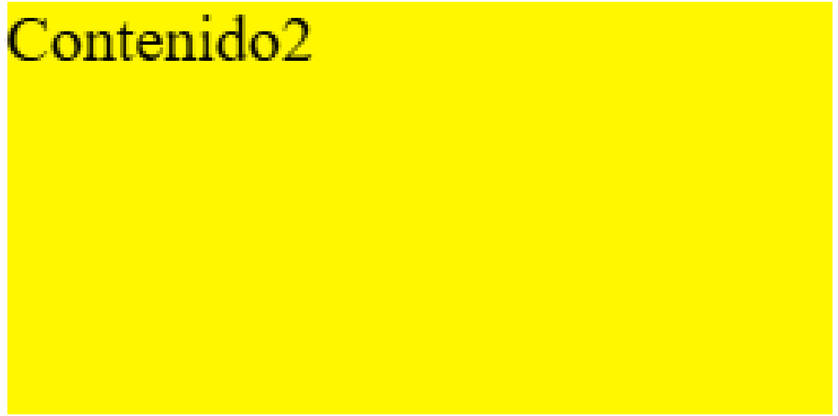
```
1   div {padding: 3px; }
```

RELLENO

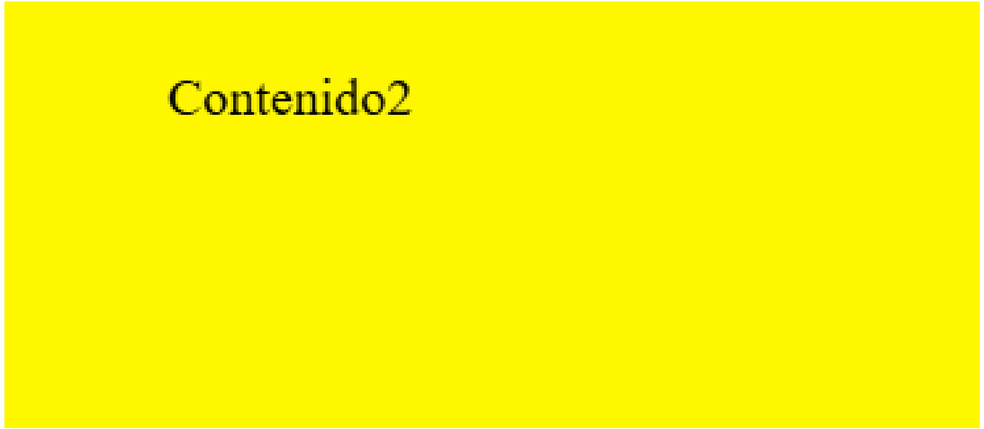
Otras características interesantes del relleno son:

- El valor del relleno se sumará al de width ya definido en el elemento.
- Su color el el mismo que al del área de contenido.
- El relleno nunca se “colapsa”.

¿ El background afecta al *padding*?



padding: 0px;



padding: 20px 50px 10px;

BORDE

Los bordes funcionan de la misma manera que el relleno, siguiendo un orden: superior, derecho, inferior, izquierdo.

Podemos configurar el estilo del borde, su grosor y su color.

La propiedad *border-style* se utiliza para configurar el estilo del borde.

```
CSS

1   div {border-style: solid dashed dotted double; }
```

BORDE

La propiedad *border-style* puede tener los valores:

- none: No define ninguna frontera.
- dotted: Define un borde de puntos
- solid: Define un borde continuo
- groove: Define un borde ranurado 3D. El efecto depende del valor del color del borde.
- inset: Define un borde 3D insertado. El efecto depende del valor del borde.
- hidden: Define un borde oculto

BORDE

- inherit: Toma la propiedad del elemento padre.
- hidden: Define un borde oculto
- dashed: Define un borde de guiones
- double: Define un borde doble
- ridge: Define un borde estriado 3D. El efecto depende del valor del borde.
- outset: Define un borde inicial 3D. El efecto depende del valor del color del borde.

BORDE

La propiedad *border-width* configuramos el grosor del borde y los valores que puede tomar esta propiedad son:

- thin: delgado
- medium: medio
- thick: grueso
- inherit
- valor concreto en píxeles

Si no se especifica esta propiedad tomará medium como valor por defecto

BORDE

Un ejemplo de la propiedad *border-width* es:

CSS

```
div {
  border-style: solid;
  border-width: thin medium thick 12px;
}
```

BORDE

La propiedad *border-color* nos permite configurar el color del borde:

```
CSS

div {
  border-style: solid;
  border-width: 4px;
  border-color: #333 #red rgb(0,0,255) #0044AC;
}
```

BORDE

La propiedad *border* une las propiedades “border” vistas anteriormente. Ésta se emplea cuando se quieren configurar los cuatro lados por igual.
También tenemos las propiedades:

- *border-top*
- *border-right*
- *border-bottom*
- *border-left*

BORDE

Por ejemplo:

```
CSS

h1 {border: .5em solid blue; }
h2 {border-left: solid blue .5em; }
h3 {border-right: solid .5em; }
```

MARGEN

Los márgenes *top* y *bottom* de dos elementos que van seguidos se “colapsan”. Es decir, se asume como margen entre ambos elementos el mayor de ellos.

¿Qué ocurre cuando tenemos dos elementos un h1 y un h2 colocados uno a continuación del otro?

CSS

```
1   h1 {margin: 10px 20px 10px 20px; }
2   h2 {margin: 20px; }
```

MARGEN

En h1 caso el margen superior e inferior es de 10 px. En h2 es de 20px. El espacio resultante entre los elementos será de 20px.

Por el contrario, si fuesen dos elementos colocados “uno al lado del otro”, deberíamos tener en cuenta que los márgenes *right* y *left* no se colapsan, sino que se suman.

¿Qué diferencia hay entre un *margin* y un *padding*?

El *margin* es un relleno externo al elemento mientras que el *padding* es un relleno interno de un elemento.

2.9. Selectores

Existen tres maneras básicas de referenciar elementos HTML desde CSS.

- Por etiqueta o elemento
- Por clase
- Por identificador

Aunque hay otras maneras más avanzadas de referenciar elementos HTML:

- Mediante combinators
- Por atributo
- Mediante pseudoselectores. Existen dos
  - Pseudoclases
  - Pseudoelementos

SELECTORES DE ETIQUETA O ELEMENTO

Son aquellos que se aplican a un elemento (etiqueta) del lenguaje HTML. Por ejemplo:

CSS

```
1  p {
2    font-family: verdana;
3    font-size: 20px;
4  }
```

HTML

```
1  <p>Esto es un párrafo con estilo</p>
```

SELECTORES DE ETIQUETA O ELEMENTO

CSS

```
h1 {color: blue;}
h2 {color: blue;}
p {color: blue;}
```

CSS

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

SELECTORES DE ETIQUETA O ELEMENTO

Existe un selector de elementos “universal” representado pr el asterisco (*)

```
CSS

1   * {color: grey;}
```

Esta regla pone en gris todos los elementos del documento que no tengan especificado otros color.

SELECTORES DE CLASE

En HTML se emplea el atributo **class** para identificar distintos elementos como parte de un grupo conceptual. Así, los elementos de una clase pueden modificarse con una única regla de estilo.

CSS

```
.bienvenida {
  color: white;
  text-align: center;
}
```

HTML

```
<p class="bienvenida">Bienvenida con estilo</p>
```

SELECTORES DE CLASE

También se puede hacer que un elemento pertenezca a más de una clase separando sus nombres de clase con espacios.

CSS

```
<p class="textocabecera especial">Hoy tenemos grandes rebajas.</p>
```

HTML

```
<h1 class="especial">Atencion!</h1>
<p class="especial">Hoy tenemos grandes rebajas.</p>
```

SELECTORES DE CLASE

Para escribir una regla de estilo que se aplique a todos los elementos de una determinada clase hay que escribir un }punto seguido del nombre de la clase. Por ejemplo:
.especial{color:green;}

SELECTORES DE CLASE

¿Qué propiedades debemos utilizar para que en el siguiente ejemplo *h1* sea de color rojo y *p* de color verde?

HTML

```html
<h1 class="especial">Atencion!</h1>
<p class="especial">Hoy tenemos grandes rebajas.</p>
```

CSS

```css
.especial {color: green;}
h1.especial{color: red;}
```

SELECTORES DE IDENTIFICADOR

Los elementos HTML disponen de un atributo llamado identificador (**id**), que tiene como finalidad identificar al elemento de manera excluyente.

Se recomienda que el valor de **id** sea un nombre que caracterice de forma breve y esquemática al elemento y que, además, sea fácilmente reconocible por el programador.

Para escribir una regla de estilo que se aplique a un determinado identificador hay que escribir el símbolo de almohadilla (**#**) seguido del nombre del identificador.

SELECTORES DE IDENTIFICADOR

HTML

```html
<p id="bienvenida">Bienvenida con estilo</p>
```

CSS

```css
#bienvenida {
  color: black;
  text-align: center;
}
```

SELECTORES DE IDENTIFICADOR

¿Pueden dos etiquetas HTML tener el mismo id?

Los identificadores deben ser únicos para identificar de manera unequívoca el elemento. Si tenemos varios elementos que necesitan un tratamiento similar, emplearemos el atributo **class**.

SELECTOR DE ATRIBUTO

El selector ***[attribute="value"]*** se utiliza para seleccionar elementos con un atributo y valor especificado.

```
a[target="_blank"] {
    background-color: yellow;
}
```

El elemento a que contiene el atributo target=_blank" aparece con el fondo en amarillo.

SELECTOR DE ATRIBUTO

El selector ***[attribute~="value"]*** se utiliza para seleccionar elementos con un valor de atributo que contiene una palabra especifica.

```
[title~="flor"]{
    border: 5px solid red;
}
```

```
<img src="./imagenes/flores 1.jpeg" width="150" height="113" title="flor de color violeta">
<img src="./imagenes/flores2.jpg" width="224" height="162"  title="flor">
<img src="./imagenes/arbol.jpg" width="223" height="325 title="arbol">
```

SELECTOR DE ATRIBUTO

El selector ***[attribute*="value"]*** se utiliza para seleccionar elementos con el atributo especificado, cuyo valor comienza con el valor especificado.

```
[class*="te"]{
    background-color: yellow;
}
```

```
<p class="primer_test">El primer párrafo</p>
<p class="segundo">El segundo párrafo</p>
<p class="mi test">El tercer párrafo</p>
<p class="mi-test">El cuarto párrafo</p>
```

SELECTOR DE ATRIBUTO

¿Qué diferencia hay entre el selector de atributo [attribute~="value"] **y el selector** [attribute*="value"] **?**

- [attribute*="value"] indica que el atributo *attribute* contiene el texto *value.*
- *[attributo~="value"]* nos permitirá seleccionar los elementos HTML que tengan un atributo con un alista de palabras separadas por espacios, donde una de ellas es el texto que hemos escrito a continuación. Se trata de una versión más restrictiva que el comparador *=.

Selectores de atributos

COMBINADORES CSS

Existen otras formas para seleccionar elementos dependiendo de la estructura del documento HTML denominados combinadores CSS.

Nombre	Símbolo	Ejemplo	Significado
Agrupación de selectores	,	p, a, div { }	Se aplican a varios elementos
Selector descendiente	Espacio	#page div { }	Se aplican estilos a elementos dentro de otros
Selector hijo	>	#page > div { }	Se aplican estilos a elementos hijos directos
Selector hermano adyacente	+	div + div { }	Se aplican estilos a elementos que siguen a otros.
Selector hermano general	~	div ~ div { }	Se aplican estilos a elementos del mismo nivel.
Selector universal	*	#page * { }	Se aplican estilos a todos los elementos

PSEUDOSELECTORES

Se emplean para aplicar estilos a elementos especiales como: los vínculos visitados, la primera línea de un párrafo o su primera letra.

Hay dos tipos de pseudoselectores:

- Pseudoclases
- Pseudoelementos

Pseudoclases

Clasifican los elementos basándose en características que van más allá de su nombre, atributos o contenido.

La mayoria de las pseudoclases afectan a elementos ancla (**a**), pudiendo definir un estilo diferente en función del comportamiento del enlace, por ejemplo:

- Si todavía no lo ha visitado (**:link**)
- Si ya lo ha visitado (**:visited**)
- Mientras el ratón pasa por encima (**:hover**)
- Justo cuando se pulsa el ratón sobre él (**:active**)

Pseudoclases

Para emplear una pseudoclase se escribe la misma a continuación del selector separándola de éste por un símbolo de dos puntos (**:**).

CSS

```
1   a:link {color: red;}
2   a:visited {color: blue;}
3   a:hover {color: fuchsia;}
4   a:active {color: maroon;}
```

Pseudoclases

¿Qué acción realizan estos pseudoclases?

```
CSS

a:link {color: red; text-decoration: none;}
a:visited {color: blue; text-decoration: none;}
a:hover {color: red; text-decoration: underline;}
```

Quitan el subrayado de los hiperenlaces y hacer que aparezca sólo cuando pasamos el puntero por encima.

Pseudoclases

Las pseudoclases ancla deben aparecer siempre en un determinado orden:

- :link
- :visited
- :hover
- :active

Pseudoclases

Existen otras pseudoclases que se emplean menos:

- **:focus** hace referencia a los elementos que tienen el foco, como ocurre, por ejemplo en elementos de un formulario.

CSS

```
1   input:focus {background-color: yellow;}
```

Pseudoclases

- **:first-child** hace referencia el primer hijo de un elemento padre.

```
CSS

ul li:first-child {font-weight: bold;}
```

Se aplica el estilo al primer elemento de una lista desordenada

Pseudoclases

- **:lang** (idioma) hace referencia al idioma en el que está un determinado elemento.

```
1   p:lang(en) {color:red}
```

Se aplica el estilo a cualquier párrafo que esté escrito en inglés.

Pseudoclases CSS

Pseudoelementos

Los pseudoelementos suelen ser partes de un elemento ya existente, como por ejemplo:

- después de su primera línea (**:first-line**)
- después de su primera letra (**:first-letter**)
- elementos sin concretar en la estructura del documento porque dependen de la estructura del documento (**:before** y **:after**).

Pseudoelementos CSS

Pseudoelementos

p:first-line {letter-spacing: 6pt;} /* añadimos espacio extra a la primera línea del texto de cada párrafo */

p.definicion:first-letter {font-size: 300%; color: red;} /* modificamos el estilo (tamaño y color) de la primera letra de los párrafos pertenecientes a la clase "definicion" */

p.incompleto:after { content: " continuará ..."; } /* añadimos el texto " continuará ..." al final de cada párrafo perteneciente a la clase "incompleto" */

LA PALABRA !important

!important se utiliza para dar prioridad a ciertas reglas. Cualquier definición de estilo que vaya acompañada de un **!important** tendrá prioridad sobre cualquier otra.

¿Qué diferencia hay entre estos dos códigos CSS?

```
1   #principal {
2     width: 800 px;
3     width: 1200px;
4   }
```

```
1   #principal {
2     width: 800px !important;
3     width: 1200px;
4   }
```

LA PALABRA !important

En el primero el ancho de elemento que tiene el identificador *principal* es de 1200px; mientras que en el segundo ancho del elemento que tiene el identificador *principal* el de 800px (no se aplica la regla de la cascada de CSS).

El uso de **!important** en CSS es algo que debe hacerse con cuidado, ya que aunque puede ser útil en ciertos casos, también puede generar problemas a largo plazo ya que rompe la cascada de CSS y hace más complicado el mantenimiento del código CSS.

2.10. Unidades de medida en CSS

2.10. UNIDADES DE MEDIIDA EN CSS

Código	Unidad	Descripción
mm	Milímetros	Medidas métricas
cm	Centímetros	
in	Pulgadas	
pt	Punto	Unidad tradicional de medida equivalente a 1/72 pulgadas
px	Pica	Unidad tradicional de medida equivalente a 12 puntos (o ⅙ pulgadas)
px	Píxel	Las unidades de píxel son relativas a la resolución del monitor
em	Em	Medida relativa respecto al tamaño de la letra que por defecto mostrará el navegador
ex	Ex	Unidad de medida relativa que es la altura de la letra minúscula "x"
%	Porcentaje	El valor porcentual se refiere al tamaño por defecto aplicado en el navegador

La mejor solución de trabajo es configurar un tamaño por defecto en forma de porcentaje, y para cada elemento de la página configurar el tamaño con la unidad **em**.

Algunos ejemplos

CSS

```
1   body { font-size: 15px; }
```

El tamaño del texto variará según la resolución del monitor.

```
</> CSS

1   body { font-size: 200%; }
```

Se configura el tamaño del texto con respecto al tamaño configurado en el navegador.
Si el navegador visualiza, por defecto, fuentes de 14 píxeles, un valor del 200% corresponde a 28 píxeles.

```
1   h1 { font-size: 2.5em; }
```

Establece el tamaño del texto de las cabeceras del primer nivel en un tamaño dos veces y medio mayor que el tamaño de la fuente mostrada por defecto en el navegador.

Unidades CSS

2.11. Posicionamiento

INTRODUCCIÓN

CSS utiliza el flotado y el posicionamiento par tener el máximo control sobre el lugar que ocupa cada elemento en una página web, sus condiciones de visibilidad y “flotabilidad”, así como controlar el manejo de capas.

INTRODUCCIÓN

Propiedades CSS para controlar el posicionamiento

display	visibility
float	z-index
clear	overflow
position	clip-path

INTRODUCCIÓN

Cuando hablamos de que los objetos de una página siguen el “flujo normal” del documento, queremos indicar que la forma en la que se disponen en la ventana del navegador coincide con el lugar que ocupan en el documento escrito (en el código HTML), donde el orden de lectura es de arriba a abajo y de izquierda a derecha.

Flotando y posicionando con CSS conseguimos que los elementos abandonen su flujo normal. De esta forma, un elemento que este en el documento escrito más abajo que otro en el documento puede verse en el navegador por encima de él.

DISPLAY

La propiedad ***display*** permite al documento interpretar de otra forma los elementos de tipo bloque y los elementos de tipo línea.

Algunos valores que puede tomar la propiedad ***display*** son:

- inline
- block
- inline-block
- flex

display: inline;

Al aplicar esta propiedad los elementos cumplen las siguientes caracteristicas:

1. Los elementos se alienan uno detrás de otro (en horizontal)
2. Su tamaño o dimensiones se adaptan al de su contenido
3. Aunque uses *width* o *height*, se ignora y no cambia de tamaño

Los elementos ** o ** se comportan como elementos en línea, mientras que los elementos * * por defecto se comportan como elementos en bloque.

display: block;

Al aplicar esta propiedad los elementos cumplen las siguientes caracteristicas:

1. Los elementos se apilan uno encima de otro (en vertical)
2. Por defecto, usa todo el ancho (en horizontal) disponible
3. Aunque uses *width* o *height*, obedece y cambia de tamaño del elemento

Los elementos * * por defecto se comportan como elementos en bloque.

display: inline-block;

Al aplicar esta propiedad se consigue un elemento que funcione como si fuera *inline*, pero obedeciendo a las propiedades *width* y *height*, las cuales se ignoran si estamos utilizando un *display: inline*.

La propiedad display

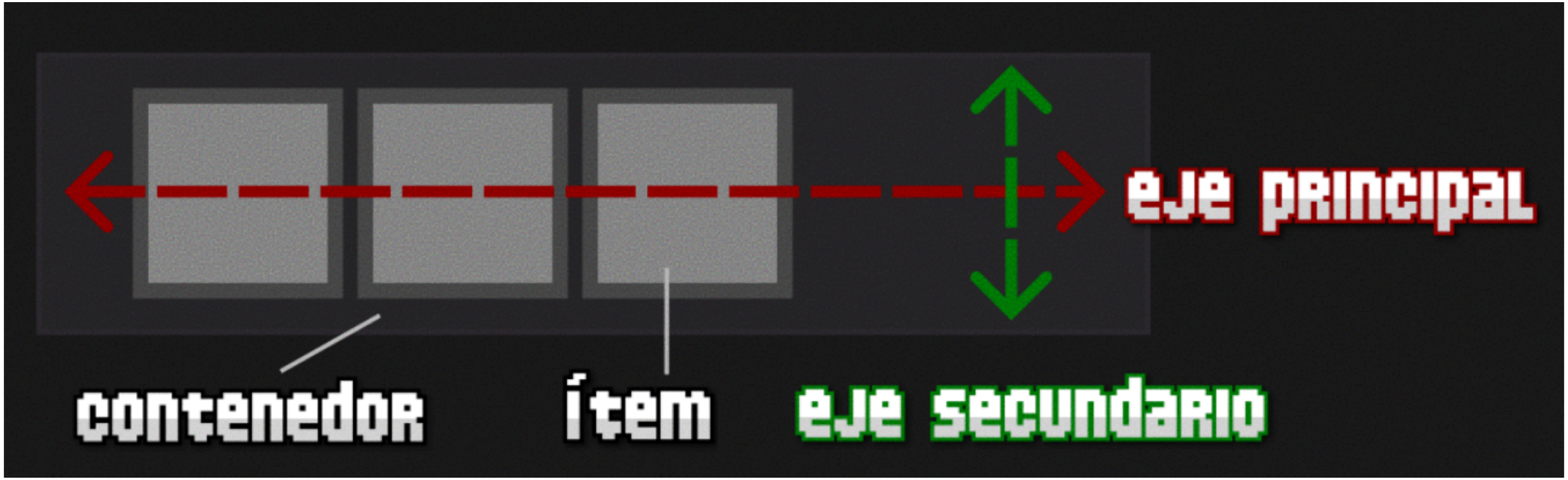
display: flex;

Flex es un sistema de elementos flexibles en la que los elementos HTML se adaptan y colocan automáticamente y es mucho más fácil personalizar los diseños de una página web.

Flex está diseñado para los siguientes escenarios:

- Crear estructuras simples de una sola dimensión
- Estructuras flexibles que no queremos que sean de un tamaño específico (adaptables)
- Crear diseños complejos enfocándote específicamente sólo en la estructura “actual”

display: flex;



display: flex;

Para activar el modo **flex**, se utiliza sobre el elemento contenedor la propiedad **display:flex;**

El contenedor se comporta como un bloque y todos los elementos se disponen sobre la misma línea.

Se puede modificar la dirección y el comportamiento de los ítems a lo largo del eje principal del contenedor con la propiedad **flex-direction** y sus valores: **row, row-reverse, column, column-reverse**.

display: flex;

La propiedad **flex-wrap** se puede modificar el comportamiento padre, de tal manera que si un elemento no cabe en el contenedor flex, permite que los elementos pasen a la línea siguiente. Los valores que puede adoptar son: **nowrap, wrap, wrap-reverse**.

Las propiedades **row-gap** y **column-gap** nos permiten establecer el tamaño de los “huecos” entre ítems de un contenedor flex. En ambas propiedades se ha de especificar el valor deseado entre filas o columnas.

display: flex;

La propiedad **row-gap** se utiliza para determinar el espacio entre filas, mientras que la propiedad **column-gap** nos indica el tamaño entre columnas.

📌 como **flex** es un sistema que trabaja en una sola dimensión, sólo una de las dos propiedades tendrá efecto:

- Si *flex-direction: column*, podrás utilizar *row-gap*
- Si *flex-direction: row*, podrás utilizar *column-gap*

La propiedad **gap** se utiliza para inidcar de una sola vez los valores para *row-gap* y *column-gap*.

Propiedades de alineación con flex

La propiedad **justify-content** se usa para alinear ítems del eje principal (horizontal).

Los valores que puede tomar esta propiedad son:

- **start**: alinea los ítems al inicio del eje secundario
- **end**: alinea los ítems al final del eje secundario
- **center**: alinea los ítems al centro del eje secundario
- **strech**: alinea los estirándolos de modo que cubran desde el inicio hasta el final del contenedor
- **baseline**: alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor

Propiedades de alineación con flex

La propiedad **align-items** sirve para alinear los ítems del eje secundario (vertical).

Los valores que puede tomar esta propiedad son:

- **start**: agrupa los ítems al inicio del eje principal
- **end**: agrupa los ítems al final del eje principal
- **center**: agrupa los ítems al centro del ejer principal
- **space-between**: distribuye los ítems dejando espacio entre ellos
- **space-around**: distribuye los ítems dejando espacio alrededor de ellos
- **space-evenly**: distribuye como *space-around*, pero con un espacio exactamente igual alrededor de ellos.

Propiedades de alineación con flex

La propiedad **align-content**, es un caso particular de **align-items.** Nos servirá cuando estemos trabajando con un contenedor flex multilinea creado con flex-wrap.

Los contendores multilinea son un tipo de contenedor en el que, cuando los ítems no caben en el ancho disponible, el eje principal se divide en múltiples líneas.

Propiedades de alineación con flex

La propiedad **align-content** sirve para alinear cada una de las líneas del contenedor multilínea.

Los valores que puede tomar esta propiedad son:

- **start**: agrupa los ítems al inicio del eje principal
- **end**: agrupa los ítems al final del eje principal
- **center**: agrupa los ítems al centro del ejer principal
- **space-between**: distribuye los ítems desde el inicio hasta el final
- **space-around**: distribuye los ítems dejando el mismo espacio a los lados de cada uno
- **stretch**: estira los ítems para ocupar de forma equitativa todo el espacio

Propiedades de alineación con flex

La propiedad **align-self** actúa igual que align-items, sin embargo es la primera propiedad de flex que se utiliza sobre un ítem hijo específico y no sobre el elemento padre contenedor.

La propiedad **align-self** nos permite cambiar el comportamiento de align-items y sobreescribirlo con comportamientos específicos para ítems concretos que no que queremos que se comporten igual que el resto.

Propiedades de alineación con flex

La propiedad **align-self** puede tomar los siguientes valores:

- **start**: alinea los ítems al inicio del contenedor
- **end**: alinea los ítems al final del eje contenedor
- **center**: alinea los ítems al centro del contenedor
- **strech:** alinea los ítems estirándolos al tamaño del contenedor
- **baseline:** alinea los ítems en el contenedor según la base de los ítems

Orden de los elementos

La propiedad **order** nos permite modificar y establecer un orden los elementos mediante números.

Por defecto, todos los elementos hijos de un contenedor flex tienen establecido un order por defecto al valor 0. Si indicamos una propiedad order con un valor numérico diferente, irá recolocando los ítems según dicho número, colocando antes los elementos con un número order más pequeño (incluso valores negativos) y después los elementos con números más altos.

Elementos flexibles en CSS

Existen una serie de propiedades muy interesantes para dotar de interacción y flexibilidad a los elementos que forman parte de un contenedor flex.

Las propiedades flexibles que se aplican sobre los elementos hijos son:

- flex-basis
- flex-grow
- flex-shrink

Propiedad *flex-basis*

La propiedad **flex-basis**, define el tamaño inicial por defecto de los ítems antes de aplicarle repartir su espacio. Se suele aplicar un tamaño específico (unidades, porcentajes, etc.), pero también se puede aplicar la palabra clave **content** que ajusta automáticamente el tamaño al contenido del elemento.

Propiedad *flex-grow*

La propiedad **flex-grow**, actúa en situaciones donde:

- Hay un flex-basis definido
- Los ítems cubren el tamaño total del contenedor flex padre.

En esas situaciones, la propiedad **flex-grow** indica el factor de crecimiento de los ítems en el caso de que no tengan un ancho o alto específico.

Propiedad *flex-shrink*

La propiedad **flex-shrink**, actúa en situaciones donde:

- Hay un flex-basis definido
- Los ítems no cubren el tamaño total del contenedor flex padre.

En esas situaciones, la propiedad **flex-grow** indica el factor de decrecimiento de los ítems en el caso de que no tengan un ancho o alto específico.

Existe una propiedad **flex** que sirve de atajo para estres tres propiedades indicadas en los ítems hijos. La nomenclatura es:

flex: ‹flex-grow› ‹flex-shrink› ‹flex-basis›

flex: 1 3 35%

FLOAT

La propiedad **float** sirve para mover una caja a la izquierda o a la derecha hasta que su borde exterior toque el borde de la caja que lo contiene o toque otra caja flotante.

Para que un elemento pueda flotar deber definido implícita o explicítamente su tamaño.

Las cajas flotantes no se encuentran en el “flujo normal” del documento por lo que las cajas que siguen el flujo normal se comportan como si las flotantes no estuviesen ahí. Es decir, no dejan el hueco de donde debería estar la caja flotante en su flujo normal, pero si se adaptan cuando se encuentran un elemento flotante.

FLOAT

La propiedad **float** puede tener los siguientes valores:

- **none** hará que el objeto no sea flotante.
- **left** hace que el elemento flote a la izquierda.
- **right** hace que el elemento flote a la derecha.
- **inherit** hará que el elemento tome el valor de esta propiedad de su elemento padre.

Primer ejemplo float

FLOAT

Flotar contenedor a la derecha

Flotar contenedor a la izquierda

Flotar dos contenedores a la izquierda

Flotar todos contenedores a la izquierda

Flotar todos contenedores a la izquierda y que se queden dentro del contenedor padre

FLOAT

Bloque contenedor demasiado estrecho

¿Qué ocurre si cambiamos el orden de las cajas?

FLOAT

Propiedad clear

La propiedad **clear** se encarga de impedir elementos flotantes en la zona indicada. Los valores de esta propiedad son:

- **left** indica que el elemento comienza por debajo de cualquier otro elemento del bloque al que pertenece que estuviese flotando a la izquierda.
- **right** funciona como el left pero en este caso el elemento deberá estar flotando a la derecha.
- **both** mueve hacia abajo el elemento hasta que esté limpio de elementos flotantes a ambos lados.
- **none** permite elementos flotantes a ambos lados. Es el valor por defecto.
- **inherit** indica que heredará el valor de la propiedad clear de su elemento padre

POSITION

La propiedad **position** se puede utilizar para modificar la posición donde aparecerá un elemento.

Esta propiedad admite cinco valores:

- static
- relative
- absolute
- fixed
- sticky

POSITION

Si utilizamos un modo de posicionamiento diferente al estático (absolute, fixed, sticky o relative), podemos utilizar una serie de propiedades par modificar la posición de un elementos.

Estas propiedades son las siguientes:

- **top**: empuja el elemento una distancia desde la parte superior hacia el inferior.
- **bottom**: empuja el elemento una distancia desde la parte inferior hacia la superior
- **left:** empuja el elemento una distancia desde la parte izquierda hacia la derecha.
- **right:** empuja el elemento una distancia desde la parte derecha hacia la izquierda.
- **inset:** propiedad de atajo par utilizar todas las anteriores.

Estas propiedades, lo único que hacen es colocar o fijar el elemento en el punto concreto indicado, de modo que si indicas *left:0;* significa que quieres que el elemento esté colocado 0 píxeles desde la izquierda. Puede contener valores negativos, invirtiendo la dirección.

La propiedad position

position: static;

La propiedad **position: static;** permite colocar los elementos según el flujo normal. Es el valor que asumirá por defecto en todos los elementos HTML.

position: relative;

La propiedad **position: relative;** permite dejar el elemento exactamente donde está. Un elemento posicionado de esta forma se puede cambiar desde su punto de partida estableciendo para ello una distancia vertical y/o horizontal.

Posicionamiento relativo

position: absolute;

La propiedad **position: absolute;** permite abandonar el flujo normal del haciendo que el elemento no ocupe ningún espacio de forma que el resto de elementos del flujo normal actuarán como si el elemento no estuviese allí. El modo de determinar el origen de coordenadas de nuestro elemento será el siguiente:

- Si el elemento posicionado de modo absoluto no está contenido dentro de ningún otro, su origen de coordenadas se mide respecto a la esquina superior izquierda del *body* (contenedor principal).

position: absolute;

- Si el elemento posicionado de modo absoluto está contenido dentro de otro elemento, el origen de coordenadas del elemento se calculan con respecto a la posición de la esquina superior izquierda del elemento que lo contiene.

Posicionamiento absoluto

position: fixed;

La propiedad **position: fixed** funciona de forma parecida al posicionamiento fixed pero posiciona según va apareciendo en el HTML y una vez posicionado se queda fijo en la parte superior de la página.

Posicionamiento fijo

position: sticky;

La propiedad **position: stcky** funciona de forma parecida al posicionamiento absoluto peor posiciona con respecto a la ventana del navegador apareciendo en la misma posición aunque el usuario se desplace por la página con las barras de desplazamiento.

Posicionamiento sticky

2.12. CSS con tablas y listas

CAMBIAR APARIENCIA DE LAS TABLAS

Para controlar la presentación de tablas tenemos las propiedades:

- caption-side
- table-layout
- border-collapse
- border-spacing
- empty-celles
- display

Propiedad caption-side

La propiedad **caption-side** sirve para indicar dónde se pone el título de la tabla.

Puede tener los valores:

- **top**
- **bottom**
- **left**
- **right**

Propiedad table-layout

La propiedad **table-layout** permite definir el modo en el que el navegador dibujará la tabla ya que puede hacerse de dos formas. Los valores que admite son:

- **fixed** dibuja la tabla basándose en las medidas establecidas en el código fuente. Con este valor se consigue que el sistema trabaja más rápido.
- **auto** dibuja la tabla basándose en el contenido de sus celdas. Es el valor por defecto.

Propiedad border-collapse

La propiedad **border-collapse** permite establecer el modo en el que se dibujan los bordes de las tablas:

- **separate** (separados), cada celda está rodeada por su borde haciendo el efecto de un borde con una línea doble.
- **collapse** (juntos) las celdas contiguas comparten sus bordes.
- **inherit** que permite heredar el valor de border-collapse que tenga su elemento padre.

Propiedad border-spacing

La propiedad **border-spacing** permite establecer la separación entre celdas contiguas. Para hacerlo se indica el valor del espaciado horizontal seguido del valor del espaciado vertical. Si se escribe un único valor, la separación horizontal y vertical serán iguales.

Propiedad empty-cells

La propiedad **border-spacing** soluciona la carencia que, al no dibujar las celdas que estaban vacías, obligaba a poner un espacio en blanco usando el carácter . Los valores que admite son:

- **show** que permite mostrar los bordes y fondos como en las celdas con contenido.
- **hide** que permite ocultar los bordes y fondos de las celdas vacías.
- **inherit** que permite heredar el valor de empty-cells que tenga su elemento padre.

CAMBIAR APARIENCIA DE LISTAS

Las listas son un elemento muy utilizado en las páginas web.
Hoy en día, su empleo no está limitado a la simple enumeración de elementos en el contenido, también se utilizan para crear barras de navegación verticales y horizontales.

Para dar formato a las listas tenemos, entre otras, las propiedades:

- list-style-type
- list-style-image
- list-style-position
- list-style

Propiedad list-style-type

La propiedad **list-style-type** permite elegir el marcador visual de la lista asignando a la propiedad uno de los siguientes valores:

- **none**: eliminar el marcador
- **square**: cuadrado
- **disc**: círculo
- **circle**: circunferencia
- **lower-roman**: números romanos en minúscula
- **lower-alpha**: letras en minúscula.
- **upper-alpha**: letras en mayúscula.

Propiedad list-style-image

La propiedad **list-style-image** permite especificar una imagen como un marcador. Para ello deberemos indicar la dirección o URL donde se encuentra la imagen.

Cuando se usa esta propiedad conviene declarar también la propiedad list-style-type en prevención de un fallo en la localización de la imagen.

Propiedad list-style-position

La propiedad **list-style-position** establece la posición del marcador de los elementos de la lista. Se puede colocar el marcador dentro del área del contenido con lo que las líneas de este elemento estarán alineadas por la izquierda (incluida la que lleva el marcador) o, se puede colocar fuera del área del contenido (como en una sangría francesa).

Los valores que permiten posicionar el marcador son:

- **inside**: dentro
- **outside**: fuera

Propiedad list-style

La propiedad **list-style** permite configurar las listas estableciendo, de forma abreviada y en cualquier orden, el valor de una o más de las propiedades individuales anteriores.

MENÚ DE NAVEGACIÓN CON LISTAS