

DISEÑO DE INTERFACES WEB

UT3. Aplicar estilos CSS (I): Estilos básicos.



CFGs Desarrollo de Aplicaciones Web
CIFP Juan de Colonia
Curso 2020-2021

Tabla de contenido

1.	Conceptos básicos.....	3
1.1.	Sintaxis	3
1.1.1.	Comentarios.....	3
1.1.2.	Mayúsculas y minúsculas.....	3
1.1.3.	Identificadores.....	3
1.1.4.	Palabras clave.....	3
1.1.5.	Declaraciones	3
1.1.6.	Propiedades short-hand (o propiedades resumidas)	5
1.1.7.	Cómo interpretan los navegadores los errores	5
1.1.8.	Buenas prácticas de escritura de reglas CSS.....	5
1.2.	Selectores.....	6
1.2.1.	Selectores de tipo, universal, de clase y de identificador	6
1.2.2.	Selectores de atributo.....	6
1.2.3.	Pseudoclasas	7
1.2.4.	Pseudo-elementos (CSS1 y 2)	11
1.2.5.	Combinadores de selectores.....	11
1.2.6.	Especificidad de los selectores.....	12
1.3.	Aplicar CSS al HTML: Hojas de estilo en Cascada	13
1.3.1.	Estilos en línea	13
1.3.2.	Estilos incrustados	13
1.3.3.	Hojas de estilos externas	13
1.3.4.	Importar hojas de estilos con @import	14
1.3.5.	Herencia.....	15
1.3.6.	Cascada.....	17
2.	Dar estilo a los elementos	18
2.1.	Unidades.....	18
2.1.1.	Unidades absolutas.....	19
2.1.2.	Unidades relativas.....	19
2.2.	Cadenas	20
2.3.	URL.....	20
2.4.	Color	21
2.4.1.	Nombres de colores	21
2.4.2.	Valores de los colores: RGB, RGBA, HSL, HSLa	21
2.5.	Fuentes.....	23
2.5.1.	Fuentes web @font-face	24
2.6.	Texto.....	26

2.7.	Fondo.....	27
2.8.	Listas	28
2.9.	Tablas	29
3.	Modelo de formato visual	30
3.1.	El modelo de cajas (box model)	30
3.1.1.	Márgenes (margin)	30
3.1.2.	Relleno (padding)	32
3.1.3.	Bordes (border)	32
3.1.4.	Bordes redondeados (border-radius) CSS3.....	33
3.1.5.	Contenido.....	34
3.2.	Bloques de contención.....	34
3.2.1.	El bloque de contención inicial	34
3.2.2.	Tipos de elementos: elementos a nivel de bloque y elementos en línea	34
3.3.	La propiedad "display".....	35
3.4.	Esquema de posicionamiento	36
3.4.1.	Esquema de posicionamiento flotante	36
3.4.2.	Esquemas de posicionamiento de flujo normal y posicionamiento absoluto	40
3.5.	Efectos visuales	42
3.5.1.	Desbordamiento: overflow	42
3.5.2.	Visibilidad: visibility	43
4.	Estructuras de distribución (<i>layouts</i>).....	44
4.1.	Diseños de ancho fijo o fluidos.....	44
4.1.1.	Diseños de ancho fijo	44
4.1.2.	Diseños fluidos o líquidos.....	45
5.	Técnicas para crear columnas	46
5.1.	Crear columnas utilizando contenedores flotantes	46
5.2.	Crear columnas utilizando posicionamiento absoluto	47
5.3.	¿Qué técnica aplicar en los diseños fijo y fluido?	47
6.	Técnicas para crear barras de navegación.....	47
6.1.	Crear barras de navegación verticales.....	47
6.2.	Crear barras de navegación horizontales.....	48
7.	Referencias y bibliografía.....	49

1. Conceptos básicos

Las hojas de estilo en cascada o **CSS** (*Cascading Style Sheets*) permiten separar el contenido y el estilo en un sitio web.

Las **ventajas** de separar el contenido y el estilo son:

- Simplifica el **desarrollo y mantenimiento** de los sitios web.
- Permite **unificar los sitios web**: se pueden aplicar distintos estilos, con el mismo HTML.

1.1. Sintaxis

1.1.1. Comentarios

- Un comentario es el texto que esté contenido entre `/*` y `*/`.
- Un comentario puede ocupar varias líneas y puede aparecer en cualquier lugar.
- Los comentarios no pueden anidarse.

1.1.2. Mayúsculas y minúsculas

- Las hojas de estilo CSS **no diferencian mayúsculas/minúsculas** (*case-insensitive*) en los nombres de las propiedades CSS y de las palabras reservadas (ej. `sans-serif`). Sin embargo, se aconseja utilizar minúsculas.

Ejemplo: Las dos declaraciones siguientes son correctas.

```
color: red;  
COLOR: Red;
```

- **Sí se diferencian mayúsculas/minúsculas** en los nombres de los identificadores o de las clases.

1.1.3. Identificadores

Los identificadores (nombres de elementos HTML, clases, identificadores, etc.):

- Pueden contener solamente los caracteres [A-Za-z0-9], el guion alto (-), el guion bajo (_) y la apertura de exclamación (!; carácter U+00A1 (161)).
- No pueden comenzar con un número.
- Sólo algunos identificadores pueden comenzar con un guion: propiedades, valores, unidades, pseudo-clases, pseudo-elementos, reglas arroba.

1.1.4. Palabras clave

- Las palabras clave se forman como los identificadores.
- Las palabras clave no se deben poner entre comillas ("..." o '...').

Ejemplo: La primera declaración es correcta, pero la segunda no lo es.

```
color: red; /* Correcto: red es una palabra reservada */  
color: "red"; /* Incorrecto: "red" es una cadena de caracteres */
```

1.1.5. Declaraciones

Una hoja de estilo CSS consiste en una lista de declaraciones. Hay dos clases de declaraciones: **Reglas CSS** y **Reglas-arroba**.

Reglas CSS

Una **regla CSS** está formada por un selector y un bloque de declaraciones.

Selector

Un selector identifica los **elementos** afectados por la regla. Un selector puede ser el nombre de un elemento HTML, una clase, un identificador, etc. Un selector puede estar formado por varios selectores, **separados por comas**.

Bloque de declaraciones

Un bloque comienza con una llave de apertura ({) y termina con la correspondiente llave de cierre (}). Entre ambas llaves, debe haber una lista de cero o más declaraciones **separadas por punto y coma**.

Declaración

La declaración contiene las instrucciones que indican al navegador cómo mostrar los elementos seleccionados. Está formada por uno o varios pares **propiedad:valor** (separados por dos puntos).

Ejemplos:

```
h1 {  
    color: blue;  
}  
h2, h3 {  
    color: blue;  
    background-color: red;  
}
```

Reglas-arroba

Las reglas-arroba comienzan con el carácter arroba, '@', seguido inmediatamente por un identificador (ej. '@import' o '@media'). Después, puede ir una declaración seguida de punto y coma o un bloque de declaraciones.

Ejemplo: La regla '@import' termina en el punto y coma.

```
@import url("texto.css");
```

Ejemplo: La regla '@font-face' va seguida de un bloque de declaraciones.

```
@font-face {  
    font-family: "Vintage";  
    src: url("fuentes/vintage.ttf") format("truetype");  
}
```

1.1.6. Propiedades short-hand (o propiedades resumidas)

Algunas propiedades son propiedades resumidas que permiten especificar los valores de varias propiedades con una sola propiedad. Por ejemplo, la propiedad `font` es una propiedad resumida para definir a la vez `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height` y `font-family`. Cuando se omiten algunos valores en la fórmula resumida, a cada propiedad "ausente" se le asigna su valor por defecto.

Ejemplo: Las siguientes reglas son equivalentes:

```
h1 {
    font-weight: bold;
    font-size: 12px;
    line-height: 14px;
    font-family: Helvetica;
    font-variant: normal;
    font-style: normal;
}

h1 {
    font: 12px/14px Helvetica;
}
```

En la segunda regla, las propiedades `font-variant` y `font-style` no se han especificado por lo que toman sus valores por defecto.

1.1.7. Cómo interpretan los navegadores los errores

- **Si un navegador encuentra una regla cuya sintaxis no entiende**, ignora la declaración entera y continúa con la siguiente, independientemente de si el error se ha producido en el selector, en el atributo, en el valor o en todo.
- **Si se omite un punto y coma entre dos declaraciones**, ambas declaraciones son ignoradas.
- **Si un navegador encuentra una regla-arroba que no conoce** (ej. '@mio'), ignora todo lo que haya hasta el fin de la regla-arroba.

1.1.8. Buenas prácticas de escritura de reglas CSS

- Una declaración en cada línea.
- Terminar todas las declaraciones con punto y coma. El *punto y coma* se utiliza para separar declaraciones por lo que no sería necesario un punto y coma después de la última declaración. Sin embargo, se aconseja escribirlo por dos motivos: no olvidarlo cuando es necesario y facilitar que se puedan añadir declaraciones al bloque de declaraciones.

1.2. Selectores

1.2.1. Selectores de tipo, universal, de clase y de identificador

Selector	Descripción	Tipo de selector	Nivel CSS
*	cualquier elemento	selector universal	2
e	un elemento de tipo <e>	selector de tipo	1
e#miID	un elemento <e> con ID igual a "miID"	selector de identificador	1
e.clase	un elemento <e> cuya clase es "clase"	selector de clase	1

Ejemplo: Seleccionar...

Para seleccionar	Selector
todos los elementos	*
elementos de tipo <p>	p
elementos <p> con identificador "importante"	p#importante
elementos <p> con clase "importante"	p. importante

1.2.2. Selectores de atributo

Seleccionan los elementos que tengan un atributo, o a los que tengan un valor determinado para un atributo.

Selector	Descripción	Nivel CSS
e[miAtr]	<e> si posee el atributo "miAtr"	2
e[miAtr="v"]	<e> si posee el atributo "miAtr" y este tiene el valor "v"	2
e[miAtr~="v"]	<e> si posee el atributo "miAtr" que es una lista de valores separados por espacios, y uno es "v"	2
e[miAtr^="v"]	<e> si posee el atributo "miAtr" y este tiene un valor que comienza con "v"	3
e[miAtr\$="v"]	<e> si posee el atributo "miAtr" y este tiene un valor que termina con "v"	3
e[miAtr*="v"]	<e> si posee el atributo "miAtr" y este tiene un valor que contiene el texto "v"	3
e[miAtr ="v"]	<e> si posee el atributo "miAtr" que es una lista de valores separados por guiones, y el primer valor es "v"	2

Ejemplo: Seleccionar...

Para seleccionar	Selector
elementos con atributo placeholder	*[placeholder]
elementos input de tipo email	input[type="email"]
elementos cuyo atributo href empiece por http	*[href^="http"]
elementos cuyo atributo href termine con .com	*[href\$=".com"]
elementos cuyo atributo href contenga "elmundo"	*[href*="elmundo"]
elementos de la clase "def" (pueden ser de otras clases también)	*[class~="def"]
elementos cuyo idioma empiece por "es-" y a continuación tenga otro valor	*[lang ="es"]

1.2.3. Pseudoclases

Permiten **seleccionar elementos en función de su estado**. Las pseudoclases se definen con el nombre del elemento, el carácter ':' (dos puntos) y el nombre de la pseudoclase, sin dejar ningún espacio en blanco alrededor del carácter ':'.

Pseudoclases de enlaces

Hay dos pseudoclases que se aplican a enlaces. Los enlaces pueden estar en dos estados: **link** y **visited**.

Selector	Descripción	Nivel CSS
a:link	Enlaces no visitados	1
a:visited	Enlaces visitados	1

Pseudoclases que dependen de las acciones del usuario

Hay tres pseudoclases que se aplican a cualquier elemento (no sólo a los enlaces), dependiendo de las acciones del usuario: **active**: "se está haciendo clic sobre él", **hover**: "se está pasando el ratón sobre él" y **focus**: "recibe foco del teclado".

Selector	Descripción	Nivel CSS
e:active	<e> cuando es activado (tiempo entre que se pulsa un botón sobre él y se suelta)	1 y 2
e:hover	<e> cuando se posiciona el cursor del ratón sobre él, pero no se activa	1 y 2
e:focus	<e> cuando el foco del teclado está posicionado en él	1 y 2

Uso de pseudoclases en los enlaces:

- Cuando se aplican varios selectores sobre un mismo elemento se pueden producir colisiones. Por ejemplo, al pasar el ratón sobre un elemento visitado, se aplican las pseudoclases **:visited** y **:hover**. Por este motivo, es importante el orden en el que se definen las pseudoclases, que debe ser el siguiente: **link**, **visited**, **hover**, **active**.
- Una técnica habitual es desactivar el subrayado por defecto a **a:link** y **a:visited** y activarlo para **a:hover**, **a:focus** y **a:active**. Además, algunos diseñadores tienen a agrupar **link** (estado inicial) y **visited** por un lado y **hover**, **focus** y **active** por otro.
- Se recomienda utilizar un estilo **:focus** para los usuarios que usan el teclado para navegar a través de los enlaces, en lugar de hacerlo con el ratón. Es frecuente que se aplique el mismo estilo que para **:hover**, pero no es obligatorio.

Ejemplo:

```
a { text-decoration: none; }
a:link { color: #00008b; }
a:visited { color: #4682b4; }
a:hover, a:focus, a:active { text-decoration: underline; }
```

- Si se utilizan imágenes como enlaces (imágenes que están dentro de un enlace), se suele eliminar el borde:

```
a img { border: none; }
```

Pseudoclase target

Si en la URL de la página, hay una referencia a un elemento dentro de la página (ej. #parrafo), se puede seleccionar el elemento que tenga ese identificador.

Selector	Descripción	Nivel CSS
e:target	<e> cuando el URI de la página selecciona un elemento dentro de la página, y el elemento <e> es el elemento seleccionado (por medio de su id)	3

Ejemplo:

```
p:target { background-color: blue; }
```

Pseudoclase lang

Si se ha definido el idioma en un documento, se puede utilizar un selector que representa un elemento en función de su idioma.

Selector	Descripción	Nivel CSS
e:lang(es)	los elementos <e> si están en el idioma es	2

Pseudoclases de elementos de UI (*user interface*)

Selector	Descripción	Nivel CSS
e:enabled	un elemento del UI <e> que está activo	3
e:disabled	un elemento del UI <e> que está desactivado	3
e:checked	un elemento del UI <e> que está seleccionado (ej. un <i>radio-button</i> o <i>checkbox</i>)	3

Pseudoclases estructurales

Permiten seleccionar elementos dependiendo de su situación en el HTML.

Selector	Descripción	Nivel CSS
e:root	un elemento <e>, raíz del documento; en HTML 4, siempre es html	3
e:nth-child(n)	un elemento <e>, hijo número "n" de su padre	3
e:nth-last-child(n)	un elemento <e>, hijo número "n" de su padre, empezando a contar desde el último	3
e:nth-of-type(n)	un elemento <e>, hermano número "n" de este tipo	3
e:nth-last-of-type(n)	un elemento <e>, hermano número "n" de su tipo, empezando a contar desde el último	3
e:first-child	un elemento <e>, primer hijo de su padre.	2
e:last-child	un elemento <e>, último hijo de su padre.	3
e:first-of-type	un elemento <e>, primer hermano de su tipo	3
e:last-of-type	un elemento <e>, último hermano de su tipo	3
e:only-child	un elemento <e>, único hijo de su padre	3
e:only-of-type	un elemento <e>, único hermano de su tipo	3
e:empty	un elemento <e>, que no tiene hijos (ni siquiera texto)	3

Las pseudoclases que aparecen con una n entre paréntesis (n), pueden utilizarse para seleccionar uno o varios elementos. En los selectores anteriores aparece (n), pero en realidad podría ser (an + b):

e:nth-child(an + b).

- 'a' y 'b' deben ser números enteros, positivos, negativos o cero.
- Para seleccionar los elementos, n va tomando valores 0, 1, 2, ...

Hay que tener en cuenta que el primer hijo de un elemento es el 1.

Ejemplo: Elementos p que ocupan un lugar impar entre sus hermanos. Es decir, ocupan los lugares 1, 3, 5, ...

Entonces, $a=2$ y $b=1$ (ocupan la posición $2*0+1, 2*1+1, 2*2+1, \dots$)

```
p:nth-child(2n+1) /* Párrafos que ocupan una posición impar */
```

Otras características:

- `:nth-child()` puede tomar como argumentos 'odd' (equivalente a $2n+1$), y 'even' (equivalente a $2n$).

Ejemplo: Uso de los argumentos odd y even.

```
tr:nth-child(odd) /* Filas impares de una tabla */  
tr:nth-child(even) /* Filas pares de una tabla */
```

Ejemplo: b con valor negativo.

```
p:nth-child(10n-1) /* párrafos en la posición 9°, 19°, 29°... */
```

- Si ' a ' tiene valor 0, se pueden omitir.

```
p:nth-child(0n+5) /* Elemento  $p$ , 5° hijo de su padre */  
p:nth-child(5) /* lo mismo */
```

- Si ' a ' tiene valor 1, no es necesario escribir el 1 antes de la n .

```
p:nth-child(1n+0) /* todos los elementos  $p$  */  
p:nth-child(n+0) /* lo mismo */  
p:nth-child(n) /* lo mismo */  
p /* lo mismo */
```

- Si ' $b=0$ ', se puede omitir.

```
tr:nth-child(2n+0) /* filas pares de una tabla */  
tr:nth-child(2n) /* lo mismo */
```

Pseudoclase de negación (CSS3)

Selector	Descripción	Nivel CSS
e:not(s)	un elemento $\langle e \rangle$ que no es seleccionado por el selector simple " s "	3

Ejemplo: Seleccionar todos los elementos que no sean de tipo $\langle p \rangle$.

```
*:not(p)
```

1.2.4. Pseudo-elementos (CSS1 y 2)

Los pseudo-elementos **permiten acceder a parte de los elementos**, o incluso **crear contenido** que no existen en el documento original. Los pseudo-elementos se definen con el nombre del elemento, dos caracteres ':' (dos puntos) y el nombre del pseudoelemento, sin dejar ningún espacio en blanco alrededor de los caracteres '::'.

La notación :: se introduce en CSS3 para diferenciar pseudo-classes y pseudo-elementos. Sin embargo, en CSS1 y 2, estos pseudo-elementos se escribían con :. Por este motivo, los agentes de usuario (navegadores) aceptan ambas notaciones en los pseudo-elementos de nivel 1 y 2.

Selector	Descripción	Nivel CSS
e::first-line	primera línea de un elemento <e>	1
e::first-letter	primera letra de un elemento <e>	1
::before y ::after	Estos pseudo-elementos se utilizan para insertar contenido generado antes o después del contenido del elemento; para ello se utiliza la propiedad 'content'.	2

Ejemplo: Añadir el texto "(*EMPIEZA UN PÁRRAFO*)" al principio de todos los párrafos.

```
p::before { content: "(*EMPIEZA UN PÁRRAFO*)"; }
```

1.2.5. Combinadores de selectores

Combinadores de descendientes

Selector	Descripción	Nivel CSS
e f	un elemento <f> que es descendiente de un elemento <e>	1
e > f	un elemento <f> que es hijo de un elemento <e>	2

Combinadores de hermanos

Selector	Descripción	Nivel CSS
e + f	un elemento <f> precedido inmediatamente por un elemento <e>	2
e ~ f	un elemento <f> precedido por un elemento <e>	3

Ejemplo: Aplicar color rojo al texto que esté dentro de la etiqueta **em** que está dentro de **strong**, y que a su vez está dentro de **h1**.

```
h1 strong em {color:red}
```

1.2.6. Especificidad de los selectores

La especificidad de un selector se calcula del siguiente modo:

- a: número de **selectores de identificador** que contiene.
- b: número de **selectores de clase, pseudo-clase y de atributo** que contiene.
- c: número de **selectores de tipo de elemento y pseudo-elementos** que contiene.

Teniendo en cuenta las siguientes reglas:

- Ignorar el selector universal.
- Los selectores dentro de la pseudo-clase de negación se cuentan como cualquier otro, pero el selector de negación no se cuenta como una pseudo-clase.
- Los combinadores de selectores (como >, + ó el espacio en blanco), no afectan a la especificidad.

Concatenando los tres números (a-b-c) tenemos la especificidad.

Ejemplo: Calcular la especificidad de los siguientes selectores.

Selector	A	B	C	Especificidad
h1	0	0	1	001
.clase	0	1	0	010
#id	1	0	0	100
html > head + body ul#nav *.home a:link	1	2	5	125
html h1	0	0	2	002

Las especificidades se comparan comparando los tres componentes en orden: la especificidad con un valor A mayor es más específica; si los dos valores A son iguales, entonces la especificidad con un valor B mayor es más específica; si los dos valores B también son iguales, entonces la especificidad con un valor C más grande es más específica; si todos los valores son iguales, las dos especificidades son iguales.

1.3. Aplicar CSS al HTML: Hojas de estilo en Cascada

1.3.1. Estilos en línea

Los estilos en línea se definen en un elemento y sólo se aplican al elemento en el que se definen. Para ello se utiliza el atributo `style` en el elemento, como en el siguiente ejemplo:

```
<h1 style="color:blue; background-color:red">...</h1>
```

Notas

- En general, estos estilos se deben evitar, porque no tiene las ventajas de las hojas de estilo (facilitar el mantenimiento) y no llevan asociado el atributo `media`.
- No obstante, tienen algunas aplicaciones prácticas. Por ejemplo, los correos electrónicos en formato HTML. Los programas de correo no reconocen los estilos embebidos con la etiqueta `<style>` ni las hojas de estilo externas. Los diseñadores se ven forzados a incorporar estilos en línea para dar estilo a sus correos.

1.3.2. Estilos incrustados

Las hojas de estilo definidas en el elemento `style` en la cabecera `<head>` se denominan internas, embebidas o incrustadas. Se pueden utilizar si la página tiene un estilo único, independiente del resto de las páginas. Las reglas se definen en el elemento `style` en el `head`, como en el siguiente ejemplo:

```
<style media="all">
    h1 {color:blue; background-color:red; }
    h2 {color:blue; }
</style>
```

Atributos del elemento `style`

<code>media="dispositivo"</code>	Dispositivo de visualización en el que se aplica el estilo (opcional).
<code>type="text/css"</code>	Lenguaje que se utiliza en la hoja de estilo (obligatorio en HTML4.01 y XHTML 1.0/1.1; opcional en HTML5).

1.3.3. Hojas de estilos externas

Los estilos se pueden colocar directamente en archivos separados del HTML. Los archivos tendrán extensión `.css`. Este documento sólo debe tener reglas CSS y comentarios. Algunas ventajas de utilizar estos archivos son:

- En el diseño es posible **reutilizar estilos**.
- Es más fácil que el sitio web tenga un **diseño homogéneo**.
- Se facilita el **mantenimiento** del sitio web.
- **Rapidez en la descarga** de las páginas de un sitio: el archivo con los estilos se descarga sólo una vez, y luego se accede a él en la caché.

Para vincular una hoja de estilos externa al archivo HTML, se utiliza el elemento `link`, dentro del `head`, como en el siguiente ejemplo:

```
<link rel="stylesheet" href="miEstilo.css" media="screen">
```

Atributos del elemento `link`:

<code>rel="stylesheet"</code>	Relación entre el documento CSS y el documento enlazado (obligatorio).
<code>href="miEstilo.css"</code>	URL del documento enlazado (obligatorio).

* Ver los atributos `media` y `type` en el apartado anterior.

Nota: Se pueden utilizar varios elementos `link` para enlazar varias hojas de estilo.

1.3.4. Importar hojas de estilos con `@import`

La regla `'@import'` permite a los usuarios importar hojas de estilo desde otras hojas de estilo. Se pueden importar varias hojas de estilos. La palabra clave `'@import'` debe ir seguida por el URI de la hoja de estilo a incluir. También se permite una cadena; que será interpretada como si contuviera `url(...)` en torno a ella. Cualquier regla `'@import'` debe preceder a todas las otras reglas en una hoja de estilo. Los navegadores ignorarán toda regla `'@import'` que aparezca dentro de un bloque o después de otras reglas CSS.

Ejemplo: Importar varias hojas de estilo dentro del elemento `<style>`.

```
<style>
    @import url("texto.css") print; /* Para impresión */
    @import "imagenes.css" all;      /* Para todos los medios */
    @import "tablas.css";            /* Para todos los medios */
    ...
</style>
```

Ejemplo: En un documento CSS, las reglas `'@import'` aparecen al principio.

```
@import url("miEstilo.css");
...
```

Ejemplo: El navegador ignorará la segunda regla `@import`, porque está después de una declaración CSS.

```
@import url("texto.css") print;
h1 {color: red}
@import "imagenes.css"
```

Ejemplo: El navegador ignorará la `@import` porque aparece dentro de un bloque `@media`.

```
@media print {
    import url("texto.css");
}
```

1.3.5. Herencia¹

La herencia en CSS es el mecanismo mediante el cual ciertas propiedades de un elemento padre se transmiten a sus hijos.

Ejemplo: A continuación, vamos a presentar un ejemplo concreto donde estudiar la herencia.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Herencia</title>
    <style>
      html {
        font: 75% Verdana,sans-serif;
        background-color: blue;
        color: white;
      }
    </style>
  </head>
  <body>
    <h1>Título</h1>
    <p>Un párrafo de texto.</p>
  </body>
</html>
```

Qué propiedades son heredables

No todas las propiedades CSS se heredan, porque no tendría sentido. Por ejemplo, no sería lógico que el ancho de un elemento fuera heredable. En general, podemos decir que se heredan las propiedades relacionadas con el texto: fuente, tamaño, color, estilo...

No se heredan las propiedades que afectan a la "caja" donde está el elemento: bordes, márgenes, fondo, etc. Normalmente el sentido común dicta qué propiedades se heredan y cuáles no, pero para estar seguros debemos consultar las tablas de propiedades de la especificación CSS: <https://www.w3.org/Style/CSS/all-properties.en.html> (*Inherited: yes/no*).

En el ejemplo anterior...

Serían heredables las propiedades `font` y `color`. Sin embargo, la propiedad `background-color`, que se refiere a la caja, no es heredable. Esto nos puede sorprender porque el color de fondo de toda la página es azul. El motivo es que el valor por defecto del color de fondo es `transparent` (transparente).

¹ <http://mosaic.uoc.edu/ac/le/es/m6/ud2/>

Qué elementos heredan de sus padres las propiedades heredables

Todos los elementos de un documento HTML heredan de su padre las propiedades heredables, excepto el elemento raíz (html), que no tiene padre.

En el ejemplo de herencia anterior...

Para el <html>, font-size es el 75%. Pero, ¿el 75% de qué? Sería el 75% del tamaño heredado de la fuente. Como HTML no hereda (porque no tiene un elemento padre), sería el 75% del tamaño de la fuente por defecto.

Qué valores se heredan

Se heredan los valores computados. Por ejemplo, si el tamaño de una fuente está definido como "1.2em" para un <div>, el navegador, calcula cuál es el valor en píxeles de ese atributo, y es el que heredan sus hijos.

En el ejemplo anterior...

El tamaño de la fuente en el <h1> y en el <p>, ¿sería el 75% del tamaño de la fuente del <html>? No, porque en CSS se heredan valores computados. Es decir, se calcula cuál es el 75% del tamaño de la fuente por defecto, y ese resultado es el que heredan los elementos hijos. Por ejemplo, si el tamaño de la fuente por defecto es 16px, el 75% sería 12px, y ese sería el tamaño que tendrían html y los elementos que hereden este valor.

Pero, si visualizamos el documento, nos sorprenderá que el tamaño del <h1> es superior al tamaño del <p>. ¿Por qué, si ambos heredan el tamaño? El motivo se explica a continuación.

¿Qué tiene prioridad: la herencia o la definición de propiedades para ese elemento?

El hecho de que las propiedades se hereden no quiere decir que siempre se apliquen. Por ejemplo, si se define el color del texto para un elemento, será ese color el que se aplique, no el heredado.

En el ejemplo anterior...

El motivo es que existe una regla del navegador que da estilo a h1, y las reglas definidas tienen más prioridad que los valores heredados.

Forzar la herencia

Mediante la palabra clave `inherit` se puede forzar la herencia, incluso para propiedades que no se heredan normalmente (esto sólo es compatible con IE8 y posteriores).

En el ejemplo anterior...

Si hubiéramos incluido la siguiente regla, el tamaño de la fuente sería el mismo de su elemento padre.

```
h1 {      font-size: inherit; }
```

1.3.6. Cascada

CSS significa “hojas de estilo en cascada”. La cascada es el mecanismo que controla el resultado final cuando se aplican varias declaraciones CSS contradictorias. Para encontrar el valor para una combinación de elemento/propiedad, las aplicaciones de usuario aplican los siguientes criterios:

- Encontrar todas las declaraciones que se aplican al tipo de medio seleccionado.
- En caso de colisiones, se aplicará la que tenga más “importancia”.
- En caso de colisiones, se aplicará la que tenga un selector más “específico”.
- Si sigue habiendo colisiones, clasificar por el orden de declaración. Aplicar las últimas que se definen.

A continuación, se explicarán los conceptos de importancia, especificidad y orden.

Importancia

La importancia de una declaración depende de donde se haya especificado. El nivel de importancia (de menor a mayor), es el siguiente:

1. Hoja de estilos de agente de usuario (*user-agent-style*).
2. Declaraciones normales en hojas de estilo de usuario (*user-style*).
3. Declaraciones normales en hojas de estilo de autor (*author-style*).
4. Declaraciones importantes en hojas de estilo de autor.
5. Declaraciones importantes en hojas de estilo de usuario.

Una **hoja de estilos de agente de usuario** es la hoja de estilo integrada en el navegador. Cada navegador tiene sus propias reglas sobre cómo mostrar varios elementos de HTML si no se especifica ningún estilo.

Una **hoja de estilos de usuario** es una hoja de estilos que ha especificado el usuario. No todos los navegadores son compatibles con las hojas de usuario, pero pueden ser muy útiles para personas con discapacidades.

Una **hoja de estilos de autor** es una hoja de estilos proporcionada con un documento HTML.

Las **declaraciones normales** son las que utilizamos normalmente. Lo contrario son las **declaraciones importantes**, que son las declaraciones que van seguidas de una directiva **!important**.

Como vemos, las declaraciones importantes en la hoja de estilos del usuario tienen la máxima prioridad. Esto permite, por ejemplo, que el usuario pueda visualizar la página de algún modo concreto, con independencia de la hoja de estilos que haya definido el autor de la página.

Ejemplo: Declaración !important.

```
h1 {  
    font-size: 300% !important;  
}
```

Especificidad

- Las declaraciones en el atributo style tienen la máxima especificidad.
- Para el resto, se calcula la especificidad como se vio antes.

Orden

- Si dos declaraciones afectan al mismo elemento, tienen la misma importancia y la misma especificidad, se tiene en cuenta el orden en que aparecen. La declaración que aparece más tarde prevalece sobre las anteriores.

Ejemplo: En este ejemplo, el primer párrafo, que tiene identificador #special, tendrá fondo rojo, porque se le aplica la regla más específica. Sin embargo, el segundo párrafo, tendrá fondo cyan, ya que es la última que aparece en el documento.

```
<style>
  #special { background-color: red; }
  p { background-color: yellow;}
  p { background-color: cyan; }
</style>

....

<p id="special">Un párrafo de texto.</p>
<p>Un segundo párrafo de texto.</p>
```

Ejemplo: El texto marcado como strong dentro de h1, será rojo, mientras que el texto marcado como strong que no esté dentro de h1, será azul.

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

<h1>Una cabecera <strong>importante</strong> de tipo h1</h1>
```

2. Dar estilo a los elementos

2.1. Unidades

Las unidades de medida se utilizan, entre otras, para definir la altura, anchura y márgenes de los elementos, y para establecer el tamaño de la letra del texto. Las medidas se especifican con un signo – o + (opcional), un número y la unidad de medida, todo sin espacios en blanco (sólo se admite no escribir ninguna medida para el valor 0).

Ejemplo: Las siguientes son ejemplos de medidas de longitud.

```
12px    /* Correcto */
-1.2em  /* Correcto */
.2em    /* Correcto */
0        /* Correcto */
12       /* Incorrecto: falta la unidad de medida */
1 em    /* Incorrecto: hay un espacio entre el número y la unidad */
```

Las unidades de medida se clasifican en absolutas y relativas.

2.1.1. Unidades absolutas

Las unidades de medida absolutas son útiles solamente cuando las propiedades del medio físico de salida son conocidas.

- **in** Pulgadas (*inches*) (1 inch = 2,54 cm)
- **cm** Centímetros
- **mm** Milímetros
- **pt** Punto (1 pt = 1/72 pulgadas)
- **pc** Pica (1 pc = 12 puntos)

2.1.2. Unidades relativas

- **em**: La **unidad 'em'** es relativa respecto al tamaño de letra empleado. Su referencia es la altura de la letra 'M' mayúscula del tipo de fuente (font-family) utilizado en el elemento.
 - La unidad em es relativa al valor computado de la propiedad 'font-size' del elemento en el que se usa.
 - Sin embargo, si 'em' aparece en la propiedad 'font-size', será relativa al valor computado de la propiedad 'font-size' del elemento padre.
- **ex**: La **unidad 'ex'** es relativa al tamaño de letra empleado. Su referencia es la altura de la letra "x" minúscula. Además, esta unidad está definida también para aquellas fuentes que no contienen la letra "x".

Cuando se utilizan en el elemento raíz, <html>, las unidades **em** y **ex** se refieren al valor por defecto de la propiedad.

- **px**: La unidad '**px**' (píxeles) es relativa respecto a la pantalla del usuario (en el nivel 3 de CSS 1in=96px, el tamaño real de cada píxel en el monitor dependerá de la resolución en la que esté configurado y del escalado, por lo que las unidades de medida absolutas pueden no coincidir con su medida física ²).

Ejemplos con px y em:

```
h1 { line-height: 1.2em; } /* La altura de línea de los elementos h1
                             será un 20% mayor que el tamaño
                             de la fuente de los elementos h1
                             */

h1 { font-size: 1.2em; } /* La propiedad font-size de los
                           elementos h1 será un 20% mayor que
                           el tamaño de la fuente del
                           elemento padre */

p { font-size:11px;
    text-indent:2em; } /* El texto se indentará 22px */

div { font-size:15px; }
div p { font-size:1.2em; } /*Dentro de este elemento, el
                           texto será un 20% mayor que en el
                           resto del div */
```

² <https://www.mydevice.io/>

Porcentajes

Es también una medida relativa. El formato de un valor en porcentaje es un <número> seguido inmediatamente por '%'. Los valores expresados en porcentajes son siempre relativos a otra medida.

Para cada propiedad que admite porcentajes, el estándar define a qué valor se refiere el porcentaje: otra propiedad para el mismo elemento, una propiedad para un elemento antepasado u otro valor.

Ejemplo: El tamaño de la fuente será 10px; la altura de la línea será el 120% de 10px, es decir, 12px. Es lo mismo indicarlo con 120% o con 1.2em.

```
p { font-size: 10px }  
p { line-height: 120% } /* 120% de 'font-size' */  
p { line-height: 1.2em } /* Equivalente a la anterior */
```

Ejemplo: Establecer la anchura de los elementos <div>: el ancho del div#principal será el 60% del ancho del div#wrapper, es decir, 576px.

```
div#wrapper { width: 960px; }  
div#principal { width: 60%; }  
  
<div id="wrapper">  
  <div id="principal">  
  </div>  
</div>
```

2.2. Cadenas

Las cadenas pueden escribirse con comillas simples o dobles. Para incluir una comilla dentro de una cadena, se deben escapar con la barra invertida \.

2.3. URL

Las direcciones URL se especifican de la siguiente forma:

```
url(dirección)
```

La dirección puede ir entre comillas simples o dobles.

Cuando en la dirección URL aparecen paréntesis, espacios, comas o comillas, deben ir escapadas con una barra invertida (\): \(. \). \', \".

Las direcciones relativas se interpretan como relativas respecto al archivo donde aparecen. Por ejemplo, si se utilizan en un archivo de hoja de estilo, serán relativas a ese documento, no al documento HTML que utiliza la hoja.

Ejemplo: En la segunda regla, se supone que la imagen (mi_imagen.png) está en el mismo directorio que la hoja de estilos donde aparece.

```
background-image: url("http://www.example.com/mi_imagen.png");  
background-image: url("mi_imagen.png");
```

2.4. Color

Para especificar un color en las hojas de estilo, se puede utilizar un nombre de color predefinido, o un valor.

2.4.1. Nombres de colores

- CSS1 y CSS2 adoptaron los 16 nombres de colores estándar de HTML4.01.
- CSS2.1 añadió el color 17 (orange)

maroon #800000	red #ff0000	orange #ffa500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
	black #000000	silver #c0c0c0	gray #808080	

- CSS3 proporciona 147 colores. Estos colores, llamados también nombres de colores de X11, son los proporcionados originalmente con el sistema X Window de Unix. Una lista de estos colores: <http://www.w3.org/TR/2011/REC-css3-color-20110607/#svg-color>.

2.4.2. Valores de los colores: RGB, RGBA, HSL, HSLa

Colores RGB

- **#RRGGBB** RR, GG, BB: Números hexadecimales.
- **#RGB** Notación simplificada (#RGB = #RRGGBB).
- **rgb(R,G,B)** R, G, B: Números enteros entre 0 y 255, ambos incluidos.
- **rgb(r%,g%,b%)** r%, g% y b%: valores entre 0.0 y 100.0 que indican el porcentaje del cada color.

Colores RGBA (CSS3)

El color RGBA permite especificar un color y aplicarle un nivel de transparencia. La "a" de RGBA significa "*alpha*", que es un canal adicional que controla el nivel de transparencia. Los colores rgba() se definen de forma parecida a los rgb():

- **rgba(R,B,G, a),** R, G, B: Números enteros entre 0 y 255, ambos incluidos.
La a es un valor decimal, en la escala de 0.0 (completa transparencia) a 1.0 (completa opacidad).
- **rgba(r%,g%,b%, a)** r%, g% y b%: valores entre 0.0% y 100.0%
La a tiene el mismo valor que en el caso anterior.

Ejemplo: Distintos modos de indicar el color rojo.

```
em { color: rgb(255,0,0) } /* rango de enteros: 0 y 255 */
em { color: rgba(255,0,0,1) } /* lo mismo, con opacidad de 1*/
em { color: rgb(100%,0%,0%) } /* rango de reales: 0.0% - 100.0% */
em { color: rgba(100%,0%,0%,1) } /* lo mismo, con opacidad de 1*/
```

Ejemplo: Color rojo con distintos niveles de transparencia.

```
h1 { color: rgba(255, 0, 0, 1) } /* Rojo */
h1 { color: rgba(255, 0, 0, 0.5) } /* Rojo semi-transparente */
h1 { color: rgba(255, 0, 0, 0.25) } /* Rojo más transparente */
```

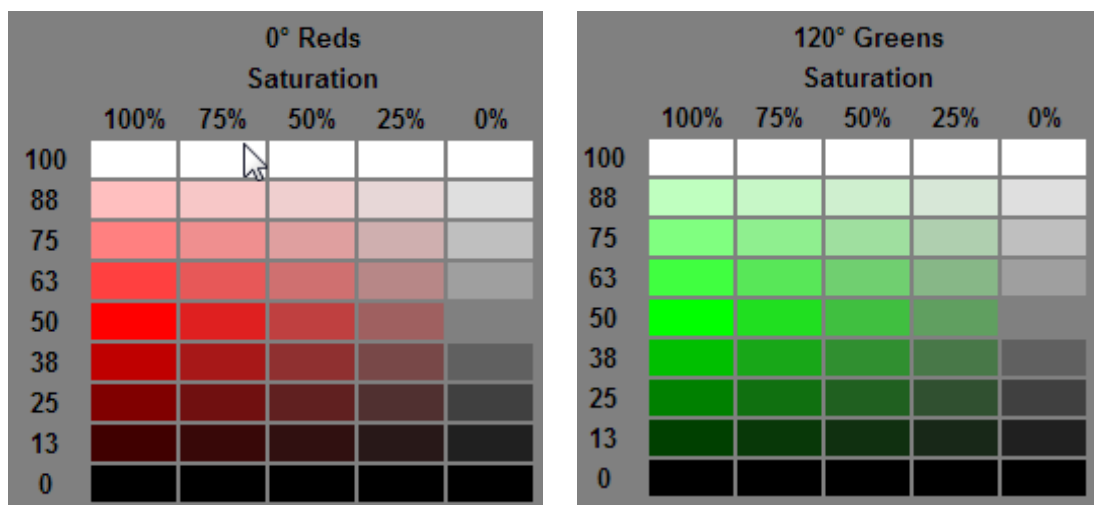
Colores HSL (CSS3)

CSS3 introduce la posibilidad de especificar los colores con sus valores HSL: H, tono (*hue*); S, saturación, L, luminosidad.

Los colores hsl se definen con **hsl(H, s%, l%)**:

- **H: tono, *hue***: Valor numérico en el círculo cromático, entre 0 y 360.
0=360 es rojo, verde 120 y azul 240.
- **s%: saturación**: 100.0% es total saturación; 0.0% es un tono de gris.
- **l%: luminosidad**: 0.0% es negro, 100.0% es blanco y 50% es normal.

Dos ejemplos de HSL, para los tonos rojo (H=0 o H=360) y verde (H=120) serían los siguientes (tomados de <https://www.w3.org/TR/css-color-3/#hsl-examples>):



Ejemplo: Distintos colores basados en el verde (120), variando la saturación y la luminosidad.

```
* { color: hsl(120, 100%, 50%) } /* lima */
* { color: hsl(120, 100%, 25%) } /* verde oscuro */
* { color: hsl(120, 100%, 75%) } /* verde claro */
* { color: hsl(120, 75%, 75%) } /* verde pastel */
```

Colores HSLA (CSS3)

La función hsla() permite especificar un color HSL y un nivel de transparencia (de igual modo que rgba).

Ejemplo: Color rojo con distintos niveles de transparencia.

```
* { color: hsl(0, 100%, 50%) } /* rojo */
* { color: hsla(0, 100%, 50%, 1) } /* lo mismo, con opacidad de 1 */
* { color: hsla(0, 100%, 50%, 0.5) } /* rojo semi-transparente */
```

Compatibilidad:

Algunos navegadores (ej. Internet Explorer 8 y anteriores) no soportan las funciones RGBa, HSL y HSLa. Una solución es proporcionar dos reglas CSS. La primera asigna un color en RGB que se aproxime al color que deseamos utilizar. La segunda asignará el color CSS3 que queremos utilizar. Los navegadores que no soportan las nuevas funciones ignoran la segunda regla, mientras que en el resto, la segunda regla sobrescribe la primera.

Ejemplo: Para utilizar un gris con transparencia.

```
h1 { color: rgb(120, 120, 120);  
      color: rgba(0, 0, 0, 50%); }
```

2.5. Fuentes

Propiedad	Descripción	Valores
font-family	Familias de fuentes	[[<nombre-familia> <familia-genérica>] [, <nombre-familia> <familia-genérica>]*]
font-style	Estilo de la fuente	[normal italic oblique]
font-variant	Convierte las minúsculas a mayúsculas pero mantienen un tamaño inferior a las mayúsculas	[normal small-caps]
font-weight	Intensidad de la fuente	[normal bold bolder lighter 100 200 300 400 500 600 700 800 900]
font-size	Tamaño de la fuente	[[xx-small x-small small medium large x-large xx-large] [larger smaller] <longitud> <porcentaje>]
font	Atajo para establecer el resto de propiedades sobre las fuentes a la vez	[[[<font-style> [<font-variant> [<font-weight>]? <font-size> [/ <line-height>]? <font-family>] caption icon menu message-box small-caption status-bar]

Familia de fuentes

font-family. Lista de familias de fuentes posibles separadas por comas. La lista está ordenada: primero se intentará utilizar la primera fuente; si no está disponible, se intentará la segunda, etc.

Fuentes genéricas:

- Se recomienda que la última fuente sea una fuente genérica.
- Las fuentes genéricas son: **serif**, **sans-serif**, **monospace**, **fantasy** y **cursive** (las dos últimas apenas se utilizan).
- Los nombres de las fuentes genéricas son palabras reservadas y no se deben escribir entre comillas. Se suelen escribir en minúsculas.

Fuentes no genéricas:

- Los nombres de las fuentes (no genéricas) se suelen escribir con la primera letra mayúscula. Se pueden escribir entre comillas o sin comillas, pero si contienen espacios o caracteres especiales, se recomienda escribirlas entre comillas.
- Se debe elegir una familia de fuentes que estén disponibles en Windows, Mac y Linux.

Estilo de la fuente

font-style. Este atributo puede tomar los valores **normal**, **italic** y **oblique**.

- El navegador buscará una fuente que sea de ese tipo.
- Si se especifica **italic** y no existe, se buscará una fuente **oblique**.

Variante de la fuente

font-variant. Este atributo puede tomar los valores **normal** y **small-caps** (versalles o versalitas).

- En la fuente **small-caps** el texto aparecerá en mayúsculas, pero en un tamaño algo más pequeño y con diferentes proporciones.

Grosor de la fuente

font-weight. Especifica el peso de la fuente.

- Los valores van de **100** a **900**, donde **100** es el menos grueso.
- El valor normal es **400** y **bold** es igual a **700**.

Tamaño de la fuente

font-size. Tamaño de la fuente. Puede tomar los valores:

- **xx-small, x-small, small, medium, large, x-large, xx-large:** Son tamaños absolutos, definidos por el navegador.
- **larger, smaller:** Son tamaños referidos al tamaño de la fuente del elemento padre. Por ejemplo, si el padre tenía small, larger indica que en este elemento el tamaño será medium.
- **Longitud y porcentaje.**

Propiedad abreviada

font. Permite definir las cinco propiedades anteriores y la propiedad **line-height**.

Los valores **caption, icon, menu, message-box, small-caption** y **status-bar** se corresponden a las fuentes utilizadas por el sistema para rotular ciertos elementos.

Ejemplo: Definir la fuente con la propiedad abreviada Font.

```
font: 12px/1.4em Arial; /* Correcto */  
font: 12px;             /* Incorrecto. font-family es obligatorio.*/
```

2.5.1. Fuentes web @font-face

Las fuentes web son las fuentes que pueden mostrarse en el navegador sin necesidad de que el usuario las tenga instaladas en su ordenador ya que se descargan automáticamente.

Sintaxis

La regla **@font-face** permite definir una nueva font-family si disponemos del archivo donde se está la fuente. Una vez definida, se puede utilizar como cualquier otra fuente. La sintaxis de la regla **@font-face** es la siguiente:

```
@font-face {  
    font-family: nombre_que_queremos_usar;  
    src: url("...") [, url("...")];  
    ...  
}
```

Donde:

- **src:** Ruta dónde se encuentra el fichero con la fuente. El fichero puede estar en otro sitio Web. Se pueden definir varias propiedades src o varias rutas en la misma src.

Ejemplo:

```
@font-face {  
    font-family: 'Vintage';  
    src: url("fuentes/vintage.ttf");  
}  
  
h1 { font-family: "Vintage", sans-serif;}
```

Compatibilidad de la regla @font-face³

Esta regla estaba incluida en CSS2, se eliminó en CSS2.1 y se ha incluido de nuevo en CCS3. Está disponible en algunos navegadores, como Internet Explorer 9, Safari 5, Firefox 7. Internet Explorer 8 y anteriores no soportan la regla @font-face. Por este motivo, es necesario utilizar fuentes alternativas.

Formatos de las fuentes y compatibilidad

Los formatos de fuentes disponibles son:

- **EOT** (*Embedded OpenType fonts*): Fuente desarrollada por Microsoft.
- **WOFF** (*Web Open Font Format*): Recomendado por la recomendación W3C WOFF File Format 1.0 de diciembre de 2012.
- **SVG fonts**. Fuentes definidas como formas SVG. El soporte para este tipo de fuentes se está retirando de los navegadores, porque esta característica se eliminó del SVG 2.0.
- **TTF** (*True Type Fonts*). Formato estándar de tipos de letras escalables utilizado por Apple y Microsoft.
- **OTF** (*OpenType Fonts*). Formato de tipos de letra escalables basado en TTF.
- **WOFF 2.0**. Formato WOFF más comprimido. Está siendo desarrollado por un grupo de trabajo de W3C.

Formatos y compatibilidad con los navegadores:

Fuente	IE	Firefox	Chrome	Safari	Opera
EOT	6+	-	-	-	-
WOFF	9+	3.6+	5+	5.1+	11.5+
TTF/OTF	9+ (soporte parcial)	3.5+	4+	3.1+	10.1+
WOFF 2.0	-	39+	36+	10	23+

³ La información sobre compatibilidades fue obtenida de <http://caniuse.com>

Qué formato de fuente elegir

Habría que comprobar que la fuente que hemos elegido es aceptada por la mayoría de los navegadores (vemos que el formato con más compatibilidad es WOFF). Podríamos utilizar una fuente de ese formato, o proporcionar dos ficheros de fuentes. Por ejemplo, uno .eot y otro .ttf, e incluirlos en la regla @font-face.

Ejemplo:

```
@font-face {  
    font-family: 'Vintage';  
    src: url('fuentes/vintage.eot'),  
         url('fuentes/vintage.ttf');  
}  
  
h1 { font-family: "Vintage", sans-serif;}
```

Obtener fuentes

Existen páginas web con fuentes que podemos descargar para nuestro sitio web. Una página con fuentes libres para uso comercial es <http://www.fontsquirrel.com/>. En esta página también se puede subir una fuente TTF y obtener las fuentes WOFF y WOFF2 equivalentes.

Utilizar Google Fonts / Google Fonts API

Google Fonts es un servicio de alojamiento de fuentes libres (<https://fonts.google.com/>). Las fuentes pueden utilizarse en nuestras páginas web sin necesidad de alojarlas en nuestro servidor. Esto tiene una ventaja: evitamos tráfico de red a nuestro servidor; y un inconveniente: la fuente puede no estar disponible.

En la página de Google nos indica cómo utilizarlas en nuestra página Web: en este caso, no escribimos la directiva @font-face en nuestro CSS, sino que Google nos proporciona un archivo con el CSS donde está definida.

Ejemplo:

```
<link href="https://fonts.googleapis.com/css?family=Shrikhand"  
rel="stylesheet">  
  
h1 { font-family: font-family: 'Shrikhand', cursive;}
```

2.6. Texto

Propiedad	Descripción	Valores
text-indent	Desplazamiento de la primera línea del texto	[<longitud> <porcentaje>]
text-align	Alineamiento del texto	[left right center justify]
text-decoration	Efectos de subrayado, tachado, parpadeo	[none [underline overline line-through blink]]
letter-spacing	Espacio entre caracteres	[normal <longitud>]
word-spacing	Espacio entre palabras	[normal <longitud>]
text-transform	Transformaciones del texto a mayúsculas/minúsculas	[capitalize uppercase lowercase none]
white-space	Comportamiento de los espacios dentro de los elementos	[normal pre nowrap pre-wrap pre-line]

text-align. Cómo se alineará el texto (o las imágenes) dentro de un elemento. Esta propiedad se asigna al elemento contenedor (párrafo, cabecera, div, etc.).

Alinear imágenes: Las imágenes son elementos en línea, lo que significa que pueden mezclarse con el texto en la misma línea. Y también significa que la alineación de imágenes básicas se controla a través de la misma propiedad: **text-align**. En el caso de las imágenes, el valor *justify* no tiene sentido.

letter-spacing. Si es una longitud, indica el espacio entre caracteres a mayores del espacio por defecto entre los caracteres (definido por la fuente o el agente de usuario).

word-spacing. Si es una longitud, este valor indica el espacio entre palabras a mayores del espacio entre las palabras por defecto (definido por la fuente o el agente de usuario).

white-space. Declara cómo son tratados los espacios en blanco y los saltos de línea dentro del elemento. Los valores posibles son:

- **pre.** Respeta los espacios y los saltos de línea. No ajusta el texto al espacio disponible.
- **pre-wrap.** Respeta los espacios y los saltos de línea. Ajusta el texto al espacio disponible.
- **nowrap.** Sustituye los espacios y los saltos de línea por un espacio. Escribe todo el texto en una línea. (como la etiqueta `<pre>`).
- **pre-line.** Sustituye los espacios consecutivos, por un espacio. Pero respeta los saltos de línea. Ajusta el texto al espacio disponible.
- **normal** (valor por defecto). Considera los espacios y los saltos de línea como espacios. Ajusta el texto al espacio disponible.

2.7. Fondo

Propiedad	Descripción	Valores
color	Color del primer plano	<code><color></code>
background-color	Color de fondo	<code>[<color> transparent]</code>
background-image	Imagen de fondo	<code>[url(...) none]</code>
background-repeat	Repetición de la imagen de fondo	<code>[repeat repeat-x repeat-y no-repeat]</code>
background-attachment	Desplazamiento de la imagen de fondo	<code>[scroll fixed]</code>
background-position	Posición de la imagen de fondo	<code>[[<porcentaje> <longitud>] left center right] [[<porcentaje> <longitud>] top center bottom] ?] [[left center right] [top center bottom]]</code>
background	Propiedades individuales relacionadas con el fondo	<code>[<background-color> <background-image> <background-repeat> <background-attachment> <background-position>]</code>

Las siguientes propiedades se utilizan cuando se quiere definir una imagen de fondo:

background-image. Imagen de fondo. Si se utiliza una ruta relativa, esta debe ser relativa al archivo donde se define esta propiedad (hoja de estilos o documento HTML).

background-repeat. Por defecto, una imagen de fondo rellena el elemento contenedor repitiendo la imagen horizontal y verticalmente cuantas veces sea necesario. Este comportamiento se puede modificar con la propiedad `background-repeat` que puede tomar cuatro valores:

- `repeat`: La imagen se repite horizontal y verticalmente (valor por defecto).
- `repeat-y`: La imagen se repite verticalmente, a lo largo del eje y (vertical).
- `repeat-x`: La imagen se repite horizontalmente, a lo largo del eje x (horizontal).
- `no-repeat`: La imagen no se repite, sólo se muestra una vez.

background-position. Si la imagen sólo aparece una vez, esta propiedad define su posición dentro del elemento contenedor: se indica primero la posición horizontal y luego la vertical. La posición puede tomar tres valores: *medidas fijas*, *porcentajes* o *palabras reservadas*.

- La posición horizontal se puede definir con: left, right, center.
- La posición vertical se puede definir con: top, bottom, center.

background-attachment. Determina cómo se visualiza la imagen de fondo cuando hacemos scroll:

- `scroll`: La imagen se desplaza con el texto.
- `fixed`: La imagen no se mueve.

Varias imágenes de fondo en CSS3

CSS3 permite definir varias imágenes de fondo.

```
#fondos{
    background: url(fondo3.png) bottom right no-repeat,
               url(fondo2.png) center no-repeat,
               url(fondo1.gif) center repeat;
    width:300px;
}
```

2.8. Listas

En los elementos **ol** y **ul** se pueden aplicar los siguientes estilos.

<code>list-style-type</code>	Estilo aplicable a los marcadores visuales de las listas	<code>[disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none]</code>
<code>list-style-image</code>	Imagen aplicable a los elementos de las listas	<code>[url("http://...") none]</code>
<code>list-style-position</code>	Posición dentro de la lista de los elementos marcadores de las listas	<code>[inside outside]</code>
<code>list-style</code>	Permite establecer el estilo de la lista, la imagen y/o la posición	<code>[<list-style-type> <list-style-position> <list-style-image>]</code>

list-style-type. Define qué símbolo (disco, letras, números, ninguno) se verán delante de los elementos de una lista (no se puede cambiar el color, tamaño, etc.). Los valores que puede tomar son:

- none: Elimina el símbolo.
- disc, circle, square: Para listas desordenadas .
- Resto de valores: para listas ordenadas.

@counter-style

CSS3 introduce nuevos elementos utilizando la regla `@counter-style`.

Ejemplo:

```
@counter-style circled-alpha {
  system: fixed;
  symbols: □ □ □ □ □ □ □ □ □ □ □ (M) □ □ □ □ □ □ □ □
           □ □ □ □ □;
  suffix: " ";
}

.items {
  list-style: circled-alpha;
}
```

list-style-image. Permite utilizar una imagen en lugar de los valores predeterminados para las listas desordenadas (*disc*, *circle*, *square*). La imagen de *list-style-image* no admite parámetros *width* y *height*, por lo que habrá que asegurarse de que la imagen tiene el tamaño adecuado.

2.9. Tablas

Las siguientes propiedades se aplican a la tabla (*table*):

Propiedad	Descripción	Valores
<code>caption-side</code>	Posición del título de la tabla	[<code>top</code> <code>bottom</code>]
<code>table-layout</code>	Control del algoritmo usado para el formato de las celdas, filas y columnas	[<code>auto</code> <code>fixed</code>]
<code>border-collapse</code>	Selección del modelo de los bordes	[<code>collapse</code> <code>separate</code>]
<code>border-spacing</code>	Espaciado entre los bordes de celdas adyacentes	<longitud> <longitud>?
<code>empty-cells</code>	Visibilidad de los bordes de celdas sin contenido	[<code>show</code> <code>hide</code>]

Para ver los bordes de la tabla habrá que utilizar la propiedad **border** de la tabla (*table*) o de la celda (*th* o *td*). También se pueden utilizar las propiedades *padding* y *margin* para estos elementos.

```
table {  
    margin: 20px;  
    border: 1px solid black;  
}
```

caption-side. Determina dónde se verá el título (<caption>), si existe: sobre la tabla (`top`) o debajo de la tabla (`bottom`).

border-collapse. Define cómo queremos que se vean los bordes de las celdas.

- `separate` (default): Hace que veamos las celdas adyacentes separadas
- `collapse`: Hace que los bordes de celdas adyacentes se unan, y se vean como uno solo.

border-spacing. Se establece el espacio que hay entre los bordes de las celdas. Esta propiedad sólo se aplica si los bordes de las celdas están separados (*border-collapse: separate*).

- Se indica primero el espacio horizontal (espacio-x) y después el vertical (espacio-y).
- Si sólo se indica un valor, se aplica a los dos espacios.

Ejemplo:

```
border-spacing: 5px; /* Establece el espaciado entre bordes a 5 */  
border-spacing: 5px 15px; /* espacio-x=5px, espacio-y=15px */
```

empty-cells. Permite definir cómo se visualizan las celdas vacías. Esta propiedad sólo se aplica si los bordes de las celdas están separados (*border-collapse: separate*).

- `show`: Se visualizan el background y los bordes.
- `hide`: No se visualiza nada.

Alinear tablas

La alineación de las tablas se realiza con los márgenes, como en los elementos de bloque. Por defecto, están alineadas a la izquierda.

Ejemplo: Centrar en su contenedor.

```
table {  
    margin: 0 auto;  
}
```

3. Modelo de formato visual

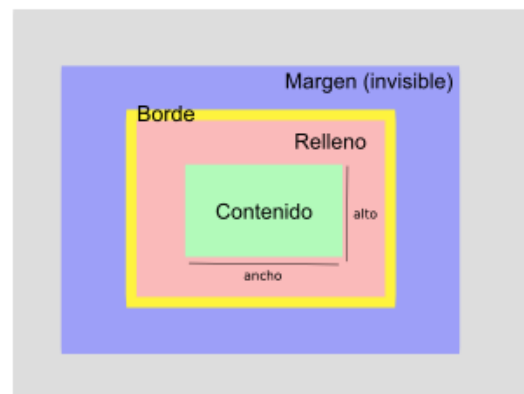
El objetivo de este apartado es describir cómo las aplicaciones de usuario procesan la estructura del documento para los medios visuales. Los medios visuales son los que proporcionan una salida visual, por ejemplo, la pantalla y la impresora.

3.1. El modelo de cajas (box model)

El navegador crea una "caja" para cada uno de los elementos html que componen una página web (<h1>, <p>, , <div>...). Mediante CSS se pueden configurar las características de esas "cajas". Este comportamiento de las páginas Web se conoce como "modelo de cajas".

Estas cajas se organizan de la siguiente forma:

- **Contenido**: Es lo que se quiere mostrar, texto, imagen, etc.
- **Relleno** (*padding*): Espacio que se puede crear alrededor del contenido.
- **Borde** (border): Línea alrededor del contenido y el relleno (si existe).
- **Margen** (margin): Espacio que se puede crear alrededor del borde. Separa esta caja de las cajas adyacentes o de la caja del elemento padre. Los márgenes son siempre transparentes.



Cómo se aplica el *background* a las cajas

- Las imágenes de fondo (background-image) y el color de fondo (background-color) están debajo del contenido y del relleno o padding.
- Si se definen un contenido, una imagen de fondo y un color de fondo, el orden de presentación es el siguiente: se muestra el contenido sobre la imagen de fondo y esta sobre el color del fondo. Es decir, el color de fondo se verá si el contenido y la imagen de fondo lo permiten: porque no ocupen completamente el espacio del contenido o porque tengan zonas transparentes.

3.1.1. Márgenes (margin)

Establecen la distancia entre el borde y las cajas adyacentes o de la caja del elemento padre.

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

Propiedad	Descripción	Valores
margin-top margin-right margin-bottom margin-left	Tamaño del margen superior, derecho, inferior e izquierdo	[<longitud> <porcentaje> auto]
margin	Ancho para varios márgenes individuales	[<longitud> <porcentaje> auto]{1,4}

Valores que se pueden utilizar para especificar los márgenes

- longitud: normalmente en píxeles o em
- porcentaje
- auto: Calcula automáticamente la distancia mínima según la relación con otros elementos.

Puede tomar 1, 2, 3 o 4 valores

- Si toma 1 valor, se aplica el mismo margen a los cuatro lados.
- Si toma 2 valores, se aplica el primer valor a los márgenes superior e inferior, y el segundo a los márgenes izquierdo y derecho.
- Si toma 3 valores, el primero se aplica al margen superior, el segundo a los márgenes izquierdo y derecho, y el tercero al margen inferior.
- Si toma 4 valores, se aplican a los márgenes superior, derecho, inferior e izquierdo (empezando arriba, y en el sentido de las agujas del reloj).

Ejemplo: Establecer los márgenes superior e inferior como 15px y los márgenes izquierdo y derecho como 30px.

```
margin-top: 15px;  
margin-right: 30px;  
margin-bottom: 15px;  
margin-left: 30px;
```

El ejemplo anterior se podía haber escrito así:

```
margin: 15px 30px;
```

Márgenes negativos

Es posible especificar valores negativos para los márgenes. Cuando se aplica un valor negativo, el contenido, el relleno y el borde de la caja se desplazan en dirección contraria de donde se colocarían con un valor positivo.

Fusión de márgenes verticales

Si se juntan los márgenes verticales de cajas adyacentes o anidadas, los márgenes se fusionan de forma automática para formar un solo margen. No puede haber entre ellos rellenos o bordes. Sólo se fusionan en el caso de cajas de bloques en el flujo normal. El nuevo margen será igual a la altura del margen más alto de los que se han fusionado.

3.1.2. Relleno (padding)

- Establece la distancia entre el contenido del elemento y el borde.
- Como margin, puede tomar 1, 2, 3 ó 4 valores.

Propiedad	Descripción	Valores
padding-top padding-right padding-bottom padding-left	Ancho del relleno superior, derecho, inferior e izquierdo	[<longitud> <porcentaje>]
padding	Tamaños para varios rellenos individuales	[<longitud> <porcentaje>] {1,4}

3.1.3. Bordes (border)

Se pueden definir tres propiedades: el ancho (border-width), el color (border-color) y el estilo (border-style).

Propiedad	Descripción	Valores
border-top-width border-right-width border-bottom-width border-left-width	Anchura del borde superior, derecho, inferior o izquierdo	[thin medium thick <longitud>]
border-width	Anchos de varios bordes individuales	[thin medium thick <longitud>] {1,4}
border-top-color border-right-color border-bottom-color border-left-color	Color del borde superior, derecho, inferior o izquierdo	[<color> transparent]
border-color	Colores de varios bordes individuales	[<color> transparent] {1,4}
border-top-style border-right-style border-bottom-style border-left-style	Estilo del borde superior, derecho, inferior o izquierdo	[none hidden dotted dashed solid double groove ridge inset outset]
border-style	Estilos de varios bordes individuales	[none hidden dotted dashed solid double groove ridge inset outset] {1,4}
border-top border-right border-bottom border-left	Ancho, estilo y el color para el borde superior, derecho, inferior o izquierdo	[<border-top-width> <border-top-style> <border-top-color>]
border	Ancho, el estilo y el color para los 4 bordes	[<border-top-width> <border-top-style> <border-top-color>]

border-color

- El valor inicial es el que tiene la propiedad color.

border-style

- Los bordes más utilizados en los diseños habituales son solid y dashed, seguidos de double y dotted.
- Los estilos none y hidden son idénticos visualmente, pero se diferencian en la forma que los navegadores resuelven los conflictos entre los bordes de las celdas adyacentes en las tablas.
- El valor por defecto de border-style es none.

Como margin y padding, las propiedades border-width, border-color y border-style pueden tomar 1, 2, 3 ó 4 valores.

3.1.4. Bordes redondeados (border-radius) CSS3

Se utiliza para hacer bordes redondeados en lugar de cuadrados.

Propiedad	Descripción	Valores
<code>border-top-left-radius</code> <code>border-top-right-radius</code> <code>border-bottom-left-radius</code> <code>border-bottom-right-radius</code>	Radio de los bordes superior-izquierdo, superior-derecho, inferior-izquierdo e inferior-derecho.	[<longitud> <porcentaje>] {1,2}
<code>border-radius</code>	Radio de los cuatro bordes redondeados.	[<longitud> <porcentaje>]{1,4}

`border-top-left-radius`, `border-top-right-radius`, `border-bottom-left-radius`, `border-bottom-right-radius`

Se pueden especificar uno o dos valores. El primero es el radio horizontal. El segundo es el radio vertical. Si no se especifica, toma como valor el radio vertical.

Ejemplo:

```
div {
    width: 100px;
    height: 100px;
    background-color: blue;
    border-top-right-radius: 20px;
}
```



Cada esquina se definen dos valores: el radio horizontal seguido del radio vertical.

```
div {
    width: 100px;
    height: 100px;
    background-color: blue;
    border-top-right-radius: 20px 40px;
}
```

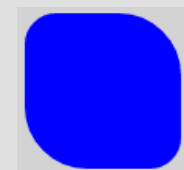


`border-radius`

- Las propiedad `border-radius` puede tomar 1, 2, 3 o 4 valores.
- Si se quieren especificar los radios horizontal y vertical, se escriben del siguiente modo: radio-horizontal/radio-vertical.
- Si se omite el radio vertical, toma el mismo valor que el radio horizontal.

Ejemplo: los bordes superior-izquierdo e inferior derecho tienen un radio de 20px, mientras que los bordes superior-derecho e inferior-izquierdo tienen un radio de 40px.

```
div {
    width: 100px;
    height: 100px;
    background-color: blue;
    border-radius: 20px 40px;
}
```



3.1.5. Contenido

Propiedad	Descripción	Valores
<code>width</code>	Ancho	[<code><longitud></code> <code><porcentaje></code> <code>auto</code>]
<code>min-width</code>	Ancho mínimo	[<code><longitud></code> <code><porcentaje></code>]
<code>max-width</code>	Ancho máximo	[<code><longitud></code> <code><porcentaje></code> <code>none</code>]
<code>height</code>	Alto	[<code><longitud></code> <code><porcentaje></code> <code>auto</code>]
<code>min-height</code>	Alto mínimo	[<code><longitud></code> <code><porcentaje></code>]
<code>max-height</code>	Alto máximo	[<code><longitud></code> <code><porcentaje></code> <code>none</code>]

Las propiedades `width` y `height` permiten establecer el ancho y el alto del contenido.

Las propiedades `min-width`, `min-height`, `max-width` y `max-height` permiten establecer unos valores mínimos y máximos para el ancho y el alto respectivamente.

Valores

- Longitud: No admite valores negativos.
- Porcentajes: Se calculan a partir de la anchura o altura de su elemento padre.
- `auto`: el navegador debe calcular automáticamente la anchura o altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

3.2. Bloques de contención

En general, una caja "establece" el bloque de contención para sus descendientes. El "bloque de contención de una caja" es la caja dentro de la cual se ha creado esa caja. Cada caja tiene una posición dada con respecto a su bloque de contención, pero lo puede desbordar.

3.2.1. El bloque de contención inicial

Es el que se genera para el elemento raíz (`html`).

Ancho del bloque de contención inicial:

- Se puede especificar con la propiedad `width`.
- El valor `"auto"` significa que la aplicación de usuario calcula el ancho inicial

Alto del bloque de contención inicial:

- Se puede especificar con la propiedad `height`.
- El valor `"auto"` significa que la aplicación de usuario calcula el alto inicial

Este bloque no puede ser posicionado o flotar. Las aplicaciones de usuario ignoran las propiedades `"position"` y `"float"` para el elemento raíz.

3.2.2. Tipos de elementos: elementos a nivel de bloque y elementos en línea

Elementos a nivel de bloque

- Los *elementos de bloque* empiezan en una línea nueva y ocupan todo el ancho del elemento contenedor. Cuando se redimensiona la ventana o un elemento contenedor, los elementos de bloque se ajustan al nuevo ancho.
- Sólo pueden aparecer dentro de otros elementos de bloque: no pueden insertarse dentro de elementos en línea.
- Pueden contener cajas de bloque o cajas en línea.

- Los siguientes valores de la propiedad *display* conforman un elemento a nivel de bloque: *block*, *list-item*, *run-in* (a veces) y *table*.
- Por ejemplo, son elementos de bloque: *h1*, *h2*, *h3*, *h4*, *h5*, *h6*, *p*, *div*, *li*, *ul*, *ol*, *table*.

Elementos a nivel de línea

- Los *elementos en línea* se visualizan en la posición en la que aparecen y ocupan sólo el espacio que necesitan. Cuando se redimensiona el navegador, el contenido se reajusta.
- Pueden aparecer tanto dentro de un elemento de bloque como dentro de un elemento en línea.
- Los siguientes valores de la propiedad *display* conforman un elemento en línea: *inline*, *inline-table* y *run-in* (a veces).
- Por ejemplo, son elementos en línea: *a*, *br*, *em*, *strong*, *sub*, *sup*.

3.3. La propiedad "display"


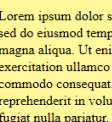

Todos los elementos tienen una propiedad *display* que determina el tipo de caja que se generará. El tipo de caja predeterminado es *inline*, pero la hoja de estilo del navegador establece otros estilos predeterminados. Por ejemplo, el estilo predeterminado para un elemento *div* es *block*.

MODELO DE FORMATO VISUAL		
Propiedad	Descripción	Valores
<i>display</i>	Comportamiento del contenedor	[<i>inline</i> <i>block</i> <i>list-item</i> <i>run-in</i> <i>inline-block</i> <i>table</i> <i>inline-table</i> <i>table-row-group</i> <i>table-header-group</i> <i>table-footer-group</i> <i>table-row</i> <i>table-column-group</i> <i>table-column</i> <i>table-cell</i> <i>table-caption</i> <i>none</i>]

Algunos de los valores más utilizados son:

- **block**: El elemento genera una caja de bloque
- **inline-block**: El elemento genera una caja de bloque, que fluye como una caja en línea, similar a un elemento reemplazado (*img* o *iframe*).
- **inline**: El elemento genera una caja en línea.
- **none**: El elemento no genera cajas en la estructura de formato. Los descendientes tampoco generan ninguna caja. El resto de elementos se visualizan como si no existiera este elemento.

Ejemplo: Aplicar la propiedad *display* a la imagen.

		
<i>display: inline (default)</i>	<i>display: none</i>	<i>display: block</i>

3.4. Esquema de posicionamiento

En CSS se definen tres esquemas de posicionamiento:⁴

- **Flujo normal:** El objeto se coloca en función del lugar que ocupa en el documento (esto significa que el lugar que ocupa en el árbol de renderización es similar al lugar que ocupa en el árbol de DOM) y se diseña de acuerdo con sus dimensiones y con el tipo de caja.
- **Flotante:** El objeto se diseña primero según el flujo normal y, posteriormente, se mueve hacia la derecha o hacia la izquierda todo lo posible.
- **Posicionamiento absoluto:** El objeto se coloca en el árbol de renderización de una forma diferente a la que se utiliza para colocarlo en el árbol de DOM. El diseño se define con exactitud independientemente del flujo normal. El objeto no participa en el flujo normal.

Las propiedades `position` y `float` determinan el esquema de posicionamiento.

- Si se utiliza **position**, con valores "**static**" o "**relative**", se genera un flujo normal.
- Si se utiliza **float**, se genera un esquema de posicionamiento flotante.
- Si se utiliza **position**, con valores "**absolute**" y "**fixed**", se produce un posicionamiento absoluto.

Nota: Si `display` vale `none`, se ignoran las propiedades `float` y `position`.

3.4.1. Esquema de posicionamiento flotante

Propiedades float y clear

La distribución de elementos en una página con elementos flotantes se controla con las propiedades `float` y `clear`.

<code>float</code>	Posicionamiento flotante	[<code>left</code> <code>right</code> <code>none</code>]
<code>clear</code>	Control de cajas adyacentes a los <code>float</code>	[<code>none</code> <code>left</code> <code>right</code> <code>both</code>]

float. Define el comportamiento del objeto que flota.

El objeto se coloca en su posición normal y después se sitúa todo a la izquierda o a la derecha que se puede en su línea (dentro de su elemento padre).

El resto de cajas fluye a su alrededor.

clear. Define qué lados de un elemento no pueden estar al lado de elementos flotantes.

- **left:** Impide que el lado izquierdo del elemento se sitúe junto a un elemento flotante.
- **right:** Impide que el lado derecho del elemento se sitúe junto a un elemento flotante.
- **both:** Impide que el elemento se sitúe junto a elementos flotantes.
- **none:** No hay restricciones sobre la posición de la caja respecto a elementos flotantes.

⁴ <http://www.html5rocks.com/es/tutorials/internals/howbrowserswork/#css>

Notas:

- Se debe *especificar el ancho de los elementos flotantes* (sólo se podría omitir en las imágenes porque el navegador lo calcula).
- Los elementos flotantes *se colocan cuando aparecen* en el flujo del documento. Por ello, siempre estarán dentro de la línea en la que aparecen o más abajo, *nunca más arriba*.
- Los elementos flotantes tienen que aparecer antes que los elementos que fluyen alrededor.
- Los elementos flotantes permanecen siempre dentro del área de contenido de su elemento padre (no invaden el padding).

Ejemplo: Imagen flotando a la derecha

```
img {  
    float: right;  
}  
  
p {  
    padding: 15px;  
    background-color: #FFF799;  
    border: 2px solid #6C4788;  
}  
  
<p>Lorem ipsum dolo... </p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Elementos en línea como elementos flotantes

Si se define la propiedad float para un elemento, cambia la propiedad display: *display=block*. Al comportarse como elementos de bloque, se pueden especificar los márgenes superior e inferior que normalmente no se tienen en cuenta para los elementos en línea.

Ejemplo: span flotando a la derecha.

```
span.aclaracion {  
    float: right;  
    width: 150px;  
    padding: 10px;  
    margin: 5px;  
    background-color: gray;  
    color: white;  
}
```

```
<p><span class="aclaracion">Texto latino</span>Lorem ipsum  
dolo... </p>
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Texto latino

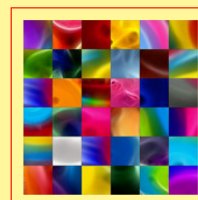
Relleno, bordes y márgenes del elemento flotante

Un elemento flotante conserva siempre sus márgenes. Es decir, flota toda la caja (contenido, rellenos, bordes y márgenes), y los márgenes no se colapsan con los superiores o inferiores.

Ejemplo: Se especifica relleno, borde y margen para la imagen.

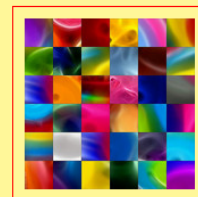
```
img {  
    float: right;  
    padding: 10px;  
    border: 1px solid red;  
    margin: 25px;  
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



El margen de la imagen se añade al margen del elemento contenedor. Si el margen superior fuera 0, el borde de la imagen (o la imagen, si no tiene borde) quedaría alineado con el resto del contenido del párrafo (`margin: 0 25px;`).

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Varios elementos flotando dentro de un contenedor

Supongamos que varios elementos dentro de un contenedor flotan a la izquierda. ¿Dónde se colocarán en este caso el resto de los elementos flotantes?

Cada uno se colocará todo a la izquierda que pueda, en la línea en la que está o más abajo, teniendo como límite el borde del elemento padre y otros elementos flotantes que ya estén colocados en ese lado, si existen.

¿Qué ocurre si un elemento flotante no cabe en la línea (porque se haya cambiado la ventana del navegador, por ejemplo)?

En este caso, se desplazará hacia abajo, hasta que encuentre una línea en la que quepa.

Ejemplo: Los estilos del ejemplo anterior se aplican al siguiente código HTML (varios flotan a la izquierda):

```
<p> 
  <span class="aclaacion">Texto latino</span>
  Lorem ipsum...</p>
```



Si se redimensiona la ventana del navegador, lo podríamos ver así:

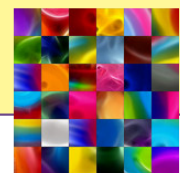


Elementos contenedores de elementos flotantes

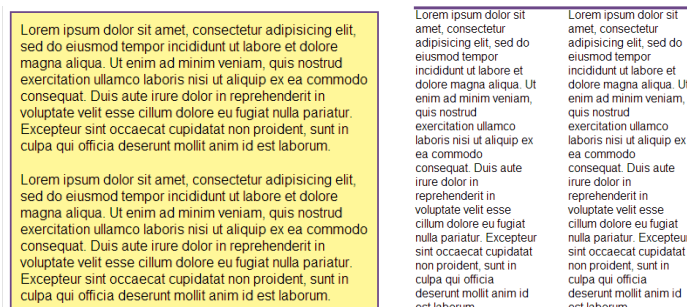
El elemento padre no tiene en cuenta a los elementos flotantes para calcular su altura.

Ejemplo:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Ejemplo: En la primera captura tenemos los dos div sin flotar. En la segunda captura hemos hecho flotar los dos div: observamos que del div general sólo se ven los bordes.



Hay varias técnicas con las que se puede conseguir este efecto.

- Una es hacer que el elemento contenedor sea flotante y darle un ancho del 100%.
- Otra solución frecuente es beneficiarnos del comportamiento de la propiedad `overflow`. Dando el valor `auto` o `hidden` a esta propiedad del contenedor, este tendrá en cuenta a los elementos flotantes para calcular su altura.

Ejemplo: En el ejemplo anterior podemos añadir la siguiente propiedad al elemento `<p>`.

```
p {
    float: left;
    ...
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Se podría conseguir un efecto parecido con la siguiente propiedad (en lugar de las anteriores):

```
p {
    overflow: auto;
    ...
}
```

3.4.2. Esquemas de posicionamiento de flujo normal y posicionamiento absoluto

Como se dijo en el punto 3.4, además del esquema flotante, hay dos esquemas de posicionamiento:

- **flujo normal:** Si `position` es `static` o `relative`.
- **posicionamiento absoluto:** Si `position` es `absolute` o `fixed`.

El tipo de esquema se define con la propiedad `position`, y la posición del elemento se define con las propiedades `top`, `bottom`, `left` y `right`.

position	Esquema de posicionamiento	[static relative absolute fixed]
top		
right	Desplazamiento de la caja (respecto al límite superior,	[<longitud> <porcentaje> auto]
bottom	derecho, inferior o izquierdo del contenedor)	
left		

La propiedad `position` puede tener cuatro valores:

- **static:** Posicionamiento normal o estático.
- **relative:** Posicionamiento relativo.
- **absolute:** Posicionamiento absoluto.
- **fixed:** Posicionamiento fijo.

Nota: Si *position = absolute* o *fixed*, se asigna: *float=none* y *display=block*.

Las propiedades *top*, *bottom*, *left* y *right*, indican el desplazamiento del elemento respecto a un elemento u objeto de referencia. Especifican el desplazamiento desde un borde de dicho elemento y hacia el centro del elemento.

- **top:** Desplazamiento respecto al borde superior: es un desplazamiento hacia abajo.
- **right:** Desplazamiento respecto al borde derecho: es un desplazamiento hacia la izquierda.
- **bottom:** Desplazamiento respecto al borde inferior: es un desplazamiento hacia arriba.
- **left:** Desplazamiento respecto al borde izquierdo: es un desplazamiento hacia la derecha.

Valores como porcentajes

Si el desplazamiento se indica en porcentaje, se refiere al porcentaje respecto a la anchura (propiedades *right* y *left*) o respecto a la altura (propiedades *top* y *bottom*) del elemento padre.

Valores negativos

Se pueden utilizar valores negativos, que mueven los elementos en dirección contraria a los valores positivos.

Flujo normal

Posicionamiento estático

En el tipo de posicionamiento por defecto En este posicionamiento, las propiedades *top*, *right*, *bottom* y *left* son ignoradas.

Posicionamiento relativo

Es una variante del posicionamiento estático. El navegador coloca todos los elementos de la página según el posicionamiento estático. Después, desplaza la caja **respecto a su posición original**, teniendo en cuenta los valores de las propiedades *top*, *bottom*, *left* y *right*. El resto de las cajas permanecen como si este elemento no se hubiera movido, y quedará vacío el espacio que ocuparía el elemento. Se pueden producir solapamientos entre cajas.

Posicionamiento absoluto

Posicionamiento absoluto

El navegador coloca el elemento en la posición que se determine con las propiedades *top*, *right*, *bottom* y *left*, que indican el desplazamiento **respecto a su primer elemento antecesor posicionado**.

Un elemento posicionado es un elemento para el que se ha asignado al atributo **position** un valor **distinto de static**, aunque no se haya desplazado, es decir, que puede aparecer en su posición original (ej. *position=relative*, pero no se modifican las propiedades de desplazamiento).

Si no hubiera ninguno, se posiciona respecto al elemento body. Ese elemento queda fuera del flujo de elementos de la página: el resto de elementos se colocan ignorando la posición de ese elemento: es como si no existiera. Se producen solapamientos.

Posicionamiento fijo

Es una variante del posicionamiento absoluto. Se colocan los elementos de la página según el posicionamiento absoluto, pero siempre **respecto a la ventana del navegador**. Después, su lugar en la pantalla es fijo, y no varía, aunque se haga *scroll*.

Superposición de elementos, z-index

Los métodos de posicionamiento pueden provocar la superposición de cajas. Por defecto, los elementos se apilan en el orden en que aparecen. Se puede cambiar este comportamiento con la propiedad *z-index*.

La propiedad *z-index* permite definir el nivel de profundidad de una caja. Su valor es un número entero: aunque se permiten números negativos, 0 se suele tomar como el valor más bajo. Cuanto más alto sea el valor, "más cerca" del usuario se muestra la caja. La propiedad *z-index* sólo tiene efecto sobre elementos posicionados (position: relative, absolute o fixed).

3.5. Efectos visuales

EFFECTOS VISUALES

Propiedad	Descripción	Valores
<code>overflow</code>	Comportamiento del contenido si se desborda en la caja	[<code>visible</code> <code>hidden</code> <code>scroll</code> <code>auto</code>]
<code>clip</code>	Especifica la región visible del elemento	[<code>rect</code> (<longitud>, <longitud>, <longitud>, <longitud>) <code>auto</code>]
<code>visibility</code>	Visibilidad de las cajas	[<code>visible</code> <code>hidden</code> <code>collapse</code>]

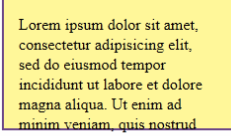
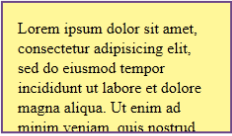
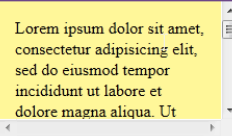
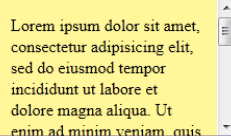
3.5.1. Desbordamiento: overflow

Generalmente, el contenido de una caja de bloque está dentro de los límites del contenido de la caja. En ciertos casos, el contenido se puede "desbordar". Esta propiedad especifica si el contenido de un elemento a nivel de bloque se recorta cuando desborda la caja del elemento.

Puede tomar los siguientes valores:

- **visible:** El contenido no se recorta, es decir, puede ser procesado fuera de la caja de bloque. Es el valor por defecto.
- **hidden:** El contenido se recorta y no se proporcionan barras de desplazamiento para acceder a la zona recortada.
- **scroll:** El contenido se recorta y, si la aplicación de usuario dispone de un mecanismo de desplazamiento (como las barras de desplazamiento), este mecanismo se muestra aunque el contenido quepa dentro de su contenedor.
- **auto:** Se permite que el navegador decida cómo manejar el overflow. En la mayoría de los casos, las barras de *scroll* se añaden sólo si son necesarias.

Ejemplo:

 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure	 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud	 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut	 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
<code>div { overflow: visible }</code>	<code>div { overflow: hidden }</code>	<code>div { overflow: scroll }</code>	<code>div { overflow: auto }</code>


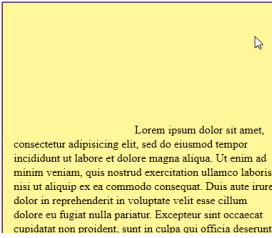
3.5.2. Visibilidad: visibility

La propiedad *visibility* se utiliza para ocultar elementos. A diferencia de *display:none*, el elemento es invisible, pero el espacio que ocupa se mantiene dejando un hueco.

Puede tomar los siguientes valores:

- **visible**: El elemento es visible. Es el valor por defecto
- **hidden**: El elemento no es visible. El resto de los elementos se visualizan como si el elemento fuera visible.
- **collapse**:
 - Si se aplica sobre filas, grupos filas, columnas o grupos de columnas de una tabla, esta propiedad hace que el elemento se colapse y el resto de los elementos se visualicen como si este elemento no estuviera en la pantalla.
 - Si se aplica sobre otros elementos, su efecto es idéntico a la propiedad *hidden*.

Ejemplo: Aplicar la propiedad *visibility* a la imagen.

 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt	
<code>img { visibility: visible (default) }</code>	<code>img { visibility: hidden (o collapse) }</code>

4. Estructuras de distribución (*layouts*)

Existe un número prácticamente ilimitado de formas de estructurar una página web. En este apartado, se verán sólo algunos de ellos.

4.1. Diseños de ancho fijo o fluidos

Los diseñadores utilizan varios enfoques para la distribución del contenido de una página Web; los principales son:

- **Diseños de ancho fijo.**
- **Diseños fluidos o líquidos.**
- **Diseños híbridos:** Contienen áreas de ancho fijo y áreas de ancho variable.

Los diseños se diferencian por las unidades de medida que utilizan para especificar las dimensiones horizontales de los elementos (ancho).

4.1.1. Diseños de ancho fijo

La página tiene siempre el mismo tamaño, independientemente de la ventana del navegador.

Características

- En este diseño, el ancho de los elementos se determina en píxeles.
- **Ancho de la página.** La mayoría de los sitios se diseñan para un ancho de 960 píxeles de ancho aproximadamente, para que se ajusten bien en pantallas con una resolución de 1024x768.

Uso

- Apropiado para páginas que se van a acceder desde un equipo de escritorio (no dispositivos móviles, *tablets*, etc.).
- **Inconveniente:** Si la ventana del navegador es más pequeña que la página, aparece la barra de *scroll* horizontal.

Ejemplo



4.1.2. Diseños fluidos o líquidos

Ajusta la página proporcionalmente cuando la ventana del navegador cambia de tamaño.

Características

- En este diseño, el ancho de los elementos se determina en porcentajes.
- El diseño fluido es uno de los pilares de la técnica de **diseño web adaptativo** (*responsive web design*).
- Este tipo de diseños está tomando cada vez más importancia. ¿Los motivos? La gran variedad de pantallas de distintos tamaños desde las que actualmente se accede a Internet: obviamente, no es posible crear un diseño fijo para cada tamaño de pantalla.

Un ejemplo: www.w3.org.

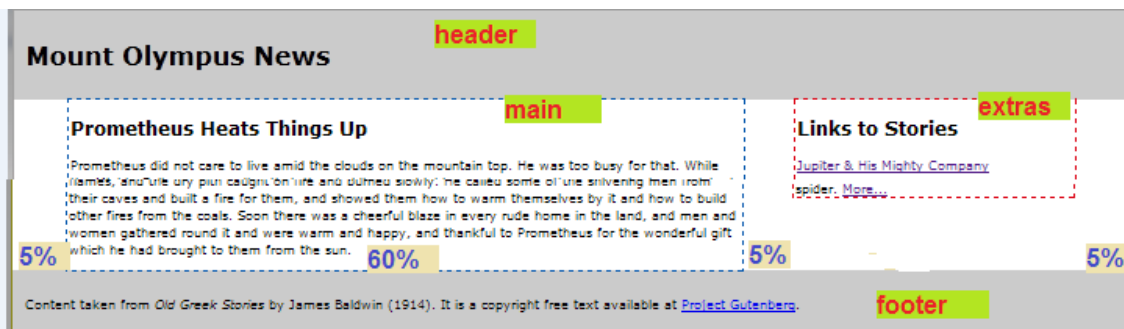
Ventajas

- Se adaptan a distintos medios.
- El texto ocupa toda la pantalla.
- No hay barras de *scroll* horizontales.

Inconvenientes

- Si la pantalla es muy grande, la línea de texto puede ser demasiado larga.
- Menos predecible.
- Más cálculos para determinar las medidas.

Ejemplo



5. Técnicas para crear columnas

5.1. Crear columnas utilizando contenedores flotantes

Este método consiste en utilizar la propiedad *float* para crear columnas. Es el método más utilizado. La ventaja de utilizar la propiedad *float* es que es más fácil que no existan solapamientos, aunque el ancho de las columnas sea fijo. La desventaja es que el diseño puede ser más complicado, ya que el orden en el que los elementos aparecen en el código influye en la visualización de la página.

Se pueden crear dos columnas dentro de un contenedor de distintos modos:

- Hacer flotar un **div** a un lado y dejar un margen al otro.
- Hacer flotar los dos **div**, ambos a la izquierda o a la derecha.
- Hacer flotar un **div** a la izquierda y el otro a la derecha (o al revés).

Para crear más columnas, se haría de forma parecida.

Ejemplo: Se crean dos *div* (de clase "columna") dentro de un *div*, #contenedor:

```
#contenedor {
  overflow: auto;
  width: 98%;
  background-color: #EEE;
  border: 2px solid #CCC;
  margin: 0 auto;
}

.columna {
  float: left;
  width: 46%;
  margin: 0 2% 1.2em;
}

<div id="contenedor">
  <div class="columna"><p><strong>Lorem ipsum dolor sit amet,</strong> ... </p>
</div>
  <div class="columna"><p><strong>Duis aute irure dolor</strong> in ... </p>
</div>
</div>
```

Se verá:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

- Es importante calcular bien los anchos de cada columna, teniendo en cuenta los rellenos, bordes y márgenes. Dependerán de la propiedad **box-sizing**: content-box (width/height incluye solo contenido) o border-box (incluye contenido, rellenos y bordes).

- Es frecuente mezclar porcentajes y **em** para crear diseños fluidos que se adapten al tamaño de la ventana del dispositivo. Algunos diseñadores utilizan **porcentajes** en distancias horizontales y **em** en las verticales, ya que hace referencia a las líneas de texto.

5.2. Crear columnas utilizando posicionamiento absoluto

El segundo modo de crear columnas es utilizar posicionamiento absoluto. La desventaja de utilizar este método es que no podemos posicionar un pie de página, a no ser que las columnas tengan un alto fijo. Además, con esta técnica, pueden existir solapamientos, si el ancho de las columnas es fijo.

5.3. ¿Qué técnica aplicar en los diseños fijo y fluido?

Ambas técnicas se pueden aplicar a diseño fijo o fluido. La diferencia está en la unidad de medida que utilicemos para los anchos de las columnas, así como los rellenos y márgenes:

- *Diseño fijo*: Las unidades de medida serán *píxeles*.
- *Diseño fluido*: Las unidades de medida serán *porcentajes*.

6. Técnicas para crear barras de navegación

Una barra de navegación normalmente consiste en una serie de enlaces agrupados en un área vertical u horizontal. La mayoría de los diseñadores Web actuales utilizan listas desordenadas para crear barras de navegación para sus sitios.

¿Por qué se utilizan listas desordenadas?

- Los enlaces de la barra de navegación son una colección de *elementos similares*, como los elementos de las listas.
- Si el navegador no es capaz de mostrar bien los estilos, la barra de navegación será una lista de enlaces operativa.
- Los elementos del submenú de una barra de navegación, que se pueden mostrar cuando se hace clic o se pasa sobre el elemento de menú del nivel principal, encuentran su paralelo exacto en los elementos de las listas anidadas.

6.1. Crear barras de navegación verticales

1. Crear la lista, normalmente dentro de un elemento nav.

```
<nav>
  <ul>
    <li><a href="inicio.html">Inicio</a></li>
    <li><a href="productos.html">Productos</a></li>
    <li><a href="servicios.html">Servicios</a></li>
    <li><a href="contacto.html">Contacto</a></li>
  </ul>
</nav>
```

2. Quitar las viñetas, y establecer los márgenes y el relleno a cero.

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

[Inicio](#)
[Productos](#)
[Servicios](#)
[Contacto](#)

3. Definir las anclas como elementos de bloque. Así se pueden establecer sus dimensiones, *padding*s, márgenes y otros efectos visuales.

```
ul li a {  
    display: block;  
    width: 10em;  
    /* más estilos */  
}
```



Inicio
Productos
Servicios
Contacto

6.2. Crear barras de navegación horizontales

1. y 2. Para crear una barra de navegación horizontal, primero creamos la lista y quitamos las viñetas, como en los pasos 1 y 2 de la creación de barras de navegación verticales.

3. A continuación, tenemos que hacer que los ítems de lista se coloquen uno al lado de otro. Esto se puede hacer de varios modos:

- Con float:left: Flotar cada ítem de la lista a la izquierda y definir las anclas como elementos de bloque.

```
ul li {  
    float: left;  
}  
  
ul li a {  
    display: block;  
    width: 10em;  
    /* más estilos */  
}
```

- Con display: inline-block: Se puede conseguir el mismo efecto utilizando display:inline-block en lugar de float:left.

```
ul li {  
    display: inline-block;  
}  
  
ul li a {  
    display: block;  
    width: 10em;  
    /* más estilos */  
}
```

- Con display: inline: También se puede utilizar display: inline. Este método hace más difícil controlar el espacio entre ítems de navegación porque el navegador calcula el espacio en blanco entre los ítems teniendo en cuenta el *font-size* del contenedor.

```
ul li {  
    display: inline;  
}
```

7. Referencias y bibliografía

REFERENCIAS WEB

CSS

- Validador CSS: <http://jigsaw.w3.org/css-validator/>.
- Guía de referencia CSS2.1, traducción al castellano: <http://www.w3c.es/Divulgacion/GuiasReferencia/CSS21/>
- Recomendación de W3C sobre el módulo de color en CSS3: <http://www.w3.org/TR/css3-color/>
- Información sobre CSS 3: <http://www.css3.info/preview/>
- W3SCHOOLS. Referencia CSS3: <http://www.w3schools.com/cssref/default.asp>
- Libros Web. Introducción a CSS: <http://www.librosweb.es/css/index.html>
- Libros Web. Referencia CSS 2.1: <http://www.librosweb.es/referencia/css/index.html>
- Libros Web. CSS avanzado: http://www.librosweb.es/css_avanzado/
- Herencia y cascada: <http://mosaic.uoc.edu/ac/le/es/m6/ud2/index.html>

Fuentes Web

- Fuentes: <https://fonts.google.com/>
- Fuentes Web: http://www.mclibre.org/consultar/htmlcss/css/css_fuentes_web.html

BIBLIOGRAFÍA

- CÓRCOLES TENDERO, J.E.; MONTERO SIMARRO, F. Diseño de Interfaces. Madrid: Ra-Ma, 2012. p. 43-93.
- NIEDERST ROBBINS, J.; Learning Web Design. O'Reilly, 2012.