CSS

Sitio:	Aula Virtual do IES de Teis	Impreso por:	Deivid Durán Durán
Curso:	Linguaxes de Marcas e Sistemas de Xestión de Información 2021-22 (DAM-A)	Data:	Venres, 24 de Xuño de 2022, 00:16
Libro:	CSS		

Táboa de contidos

CSS

- ***C**ascading **S**tyle **S**heets* **(CSS) u Hojas de Estilo en Cascada**
- Lenguaje de estilos utilizado para describir la **presentación** de documentos HTML, XHTML o XML en la pantalla, en la impresión en papel o en distintos medios
- Se trata de un estándar especificado por el **W3C**.
- Fue propuesto por el noruego **Håkon Wium Lie** en **1994** mientras trabajaba con Tim Berners Lee en el CERN.
- **La primera versión estandarizada** se publica en **1996.**
  - Se han desarrollado varias especificaciones de distintos niveles: CSS1, CSS2.1, CSS3.
  - Los primeros borradores de CSS3 se publicaron en 1999
  - Últimamente ya no se publican versiones numeradas completas, sino **versiones diferenciadas por módulos** también llamadas ***snapshots***
- La colaboración entre HTML y CSS permite **la separación de responsabilidades**: HTML se ocupa de la descripción del documento y su estructura y CSS, por su parte, de la presentación del mismo

Hakon Wium Lie - Opera de Franco Folini- bajo CC BY-SA 2.0

2. Sintaxis CSS

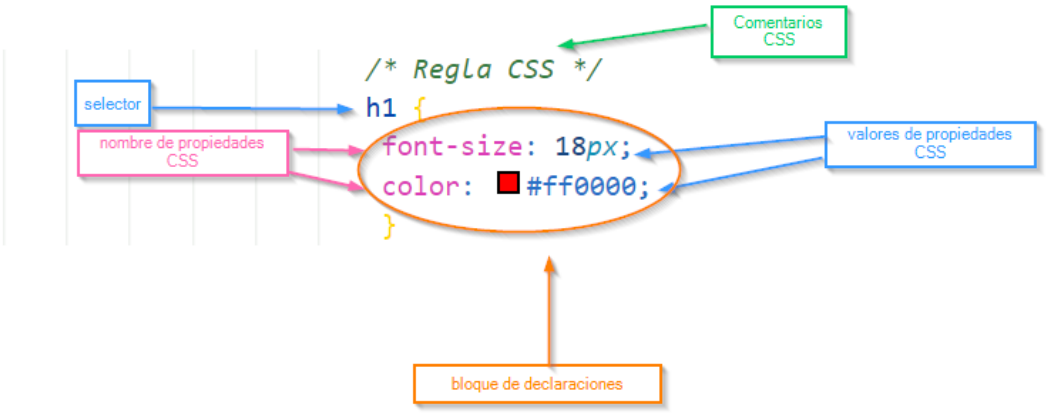
Las hojas de estilo CSS están compuestas de una o varias **reglas CSS.**

Las reglas CSS están compuestas a su vez de:

- **Uno o varios selectores:** Indican a qué elementos HTML se les aplicarán las propiedades CSS.
- **Un bloque de una o varias declaraciones CSS** encerrado entre llaves {}
  - Cada declaración CSS consta de **un nombre de propiedad CSS y un valor** para esa propiedad. El nombre y el valor van **separados por dos puntos** :
  - Las declaraciones CSS dentro del mismo bloque se separan por punto y coma. Para la última declaración el punto y coma no es obligatorio, pero es una buena práctica.

Es posible añadir comentarios dentro de /* */

Por ejemplo, la siguiente imagen muestra un regla CSS indica que para todos los encabezados de primer nivel ha de mostrarse un tamaño de letra de 18 píxeles y el color de la letra es rojo. El color se expresa en este caso como un número hexadecimal.



3. CSS y HTML: ¿Cómo aplicar estilos CSS a un documento HTML?

Existen 3 modos de añadir estilos CSS a un documento HTML:

Inline

- A través del atributo ***style*** de cualquier elemento HTML. Esta opción no se recomienda, pues no permite separar la estructura del documento de su presentación. Además, su mantenimiento es muy costoso.

```
<h1 style="font-size: 18px;color:#ff0000;" id="t1">Título 1</h1>
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Estilos inline</title>
</head>
<body>
    <h1 style="font-size: 18px;color:#ff0000;">Título 1</h1>
</body>
</html>
```

Hoja de estilos interna

A través de la etiqueta <style> dentro del elemento <head>, tal y como hemos hecho con los elementos table, td y th para mostrar los bordes del elemento <table>. Para las mismas propiedades del ejemplo anterior el uso de una hoja de estilos interna sería el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Estilos interna</title>
    <style>
        h1 {
            font-size: 18px;
            color: #ff0000;
        }
    </style>
</head>

<body>
    <h1>Título 1</h1>
</body>

</html>
```

Hoja de estilos externa

Se especifican las reglas CSS en un documento con extensión .css . Se establece la vinculación entre el documento HTML y la hoja de estilos externa a través de un elemento <link> que deberá situarse dentro del elemento <head>. <link> utiliza en este caso los siguientes atributos:

- **href:** indica la URL del fichero externo con extensión CSS
- **rel:** indica el tipo de relación con el fichero especificado en href.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Estilos externa</title>
    <link rel="stylesheet" href="estilos.css">
</head>

<body>
    <h1>Título 1</h1>
</body>

</html>
```

El contenido del fichero **estilos.css** sería el siguiente:

```css
h1 {
  font-size: 18px;
  color: #ff0000;
}

```

En los tres casos el resultado de la presentación de la página en un navegador debería ser el mismo:

Título 1

4. Selectores básicos de CSS

Selectores de etiquetas HTML

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

Todos los encabezados h1 tendrán letra de color azul.

Selectores de clase

Es posible aplicar propiedades CSS solo a algunas etiquetas de HTML y no a todas las del mismo tipo. Esto es posible a través del atributo de HTML *class*.

```
    <h1 class="destacado">Título de clase</h1>
    <p class="destacado">Primer párrafo</p>
    <p>Segundo párrafo</p>
    <p>Tercer párrafo</p>
```

En la hoja de estilos deberemos preceder el nombre de la clase con un punto:

```
.destacado {
    background-color: lightgrey;
}
```

En este caso se le estaría aplicando un color de fondo solo a los elementos con el atributo class="destacado". Se aconseja que se utilice un nombre que haga alusión al propósito del estilo y no tanto a las propiedades CSS que cambia.

Si quisiéramos que la regla solo se aplique a los párrafos con clase *destacado*, se utiliza la concatenación del elemento y la clase separada por un punto (sin espacios):

```
p.destacado {
    background-color: lightgrey;
}
```

El atributo *class* puede tener varias palabras separadas por espacios en cuyo caso se aplicarán las propiedades de los selectores de tantas clases como palabras contenga el atributo *class*.

Selectores de id

Este tipo de selector usa el atributo universal id de HTML. Recordemos que su valor debe ser único en todo el documento, por lo que este tipo de selector **solo se aplicará a un único elemento**.

```
<section id="login">

</section>
```

En la hoja de estilos hay que preceder el valor del atributo id con el símbolo de almohadilla #

```
#login{
      color: #0000ff;
}
```

Selector universal: *

Selecciona todos los elementos HTML de la página

```
* {
    background-color: yellow;
}
```

Agrupación de selectores

Si se aplican las mismas propiedades y valores a diferentes selectores es posible agruparlos separando los selectores por comas.

```
h1, h2, .destacado, #login{
    color: red;
}
```

Se aplicará a todos los encabezados de primer y segundo orden, a todos los elementos con la clase destacado y al elemento cuyo valor del atributo id sea login.

4.1. Combinación de selectores

Es posible combinar varios selectores CSS.

Se pueden consultar ejemplos de los diferentes tipos aquí.

Selector de descendientes

Selectores separados por un espacio. Se indica que el selector que ocupa la posición n+1 es un descendiente, hijo, nieto, bisnieto, etc., del selector que ocupa la posición n.

```
div p {
  background-color: yellow;
}
```

Se aplicará a todos elementos p con un ancestro div.

Selectores de hijo directo

Se separan por >

```
div > p {
  background-color: yellow;
}
```

Se aplica solo a los párrafos hijos de un elemento div. No se aplicará a los párrafos que sean nietos, bisnietos, etc.

Selectores de hermanos adyacentes (uno justo a continuación del otro)

Se separan por el símbolo +.

```
div + p {
  background-color: yellow;
}
```

Afectará a los párrafos que sigan inmediatamente a un elemento div

Selectores de hermanos en general

Se separan por símbolo ~ (Desde el teclado: Alt Gr + 4 y barra espaciadora)

```
div ~ p {
  background-color: yellow;
}
```

Afectará a los párrafos que sigan un elemento div en el mismo nivel de anidación.
(no hace falta que vayan uno a continuación del otro)

4.2. Selectores de pseudo-clase

Se usan para definir un estado específico de un elemento. Por ejemplo:

- Cuando el ratón se mueve sobre un elemento HTML
- Cuando un enlace aún no ha sido visitado
- Cuando un elemento tine el foco (la atención del teclado)

La pseudo-clase se añade al selector (o combinación de selectores) añadiendo dos puntos (sin espacios) y el nombre de la pseudo clase:

```
selector:pseudo-class {
  propiedad: valor;
}
```

Es muy habitual su uso para los enlaces:

```
/* enlace no visitado */
a:link {
  color: #FF0000;
}

/* enlace visitado */
a:visited {
  color: #00FF00;
}

/* ratón sobre enlace */
a:hover {
  color: #FF00FF;
}

/* enlace en el momento del clic sobre el mismo */
a:active {
  color: #0000FF;
}
```

Se recomienda usar **en este mismo orden.**

En la siguiente tabla, destaco un subconjunto de pseudo-clases:

Pseudo-clase	Ejemplo	Descripción
:first-child	p:first-child	Selecciona los elementos p que sean el primer hijo directo de un grupo de hermanos
:first-of-type	p:first-of-type	Selecciona el primer elemento p de un grupo de hermanos
:last-child	p:last-child	Selecciona los elementos p que sean el último hijo directo de un grupo de hermanos
:last-of-type	p:last-of-type	Selecciona el último elemento de tipo p de un grupo de hermanos
:nth-child(n)	p:nth-child(2)	Selecciona uno o más elementos en función de su posición entre un grupo de hermanos. n puede ser: -un número -una palabra clave (even para par y odd para impar) p:nth-child(even) seleccionará los elementos p que sean hijos pares de un grupo de hermanos. -una fórmula, por ejemplo, 3n para los múltiplos de 3. El índice del primer hijo es el 1. El ejemplo selecciona el elemento de tipo p que es el 2º hijo de entre un grupo de hermanos
:nth-of-type(n)	p:nth-of-type(2)	Igual que en la anterior, pero en función de su tipo y no su posición. El ejemplo selecciona el 2º elemento de tipo p de entre un grupo de hermanos
:not(selector)	:not(p)	Representa elementos que no coinciden con una lista de selectores. El ejemplo seleccionará todos los elementos que no sean párrafos

Una lista de todas las pseudo-clases disponibles se puede consultar en cualquiera de estos recursos:

- https://www.w3schools.com/css/css_pseudo_classes.asp
- https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes

4.3. Selectores de atributo

Permiten seleccionar elementos en función de si tienen un determinado atributo o filtrar por su valor.

```
selector[atributo]{

propiedad: valor;

}
```

Selectores de atributos	Ejemplo	Descripción ejemplo
[atributo]	[target]	Selecciona los elementos que tienen un atributo *target*
selector[atributo]	a[target]	Selecciona los elementos a que tengan un atributo *target*
selector[atributo="valor"]	a[href="#t1"]	Selecciona los elementos a con el atributo href exactamente igual al valor *#t1*
selector[atributo^="valor"]	a[href^="https://"]	Selecciona los elementos a con atributo href que comienza con la cadena de texto *https://*
selector[atributo$="valor"]	a[href$=".org"]	Selecciona los elementos a con el atributo href y este termina con la cadena de texto *.org*
selector[atributo*="valor"]	a[href*="aulavirtual"]	Selecciona los elementos a donde el atributo href contiene la palabra *aulavirtual*

4.4. Especificidad de selectores CSS

Reglas en conflicto y especificidad

Podría llegar a darse el caso en una hoja de estilos extensa de que existiesen reglas CSS que tuviesen valores diferentes para la misma propiedad CSS y se intentasen aplicar al mismo elemento.

En esos casos se utilizan unas reglas de precedencia de selectores. En general prevalecen los estilos más específicos sobre los más generales. Por orden de prioridad descendiente:

1. Propiedades acompañadas de !important. Estas propiedades tienen la máxima prioridad. Sin embargo, su uso está desaconsejado salvo que sea imprescindible. Un ejemplo de este tipo de regla podría ser
   ```
   table td    { height: 50px !important; }
   ```
2. Estilos *inline*
3. Selectores de id
4. Selectores de clase, de atributos y de pseudo-clases
5. Selectores de etiquetas HTML y de pseudo-elementos: Los pseudo-elementos permiten añadir estilos a una parte concreta del documento y tienen una sintaxis similar a las pseudo-clases, salvo que con un par de dos puntos . Por ejemplo: `::first-letter` aplica estilos a la primera letra de la primera línea un elemento de bloque. Se pueden consultar más pseudo-elementos aquí.

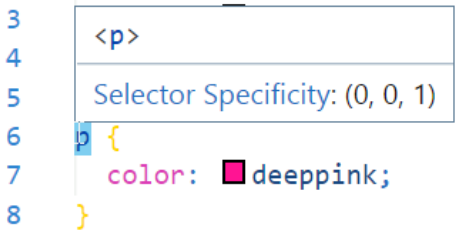
Si existiese una combinación de selectores, ha de calcularse la **especificidad** de la regla. Para ello se le otorga una puntuación numérica a cada selector:

Tipo de selector	Especificidad
Estilos inline (con atributo *style*)	1000
Selectores de id	100
Selectores de clase, de atributos y de pseudo-clases	10
Selectores de etiquetas HTML y de pseudo-elementos	1

El selector universal (*), los combinadores (+, >, ~, " ") y el selector de pseudo-clase :not() no tienen efecto en el cálculo de la especificidad. Sin embargo, los selectores que van dentro del paréntesis de :not() sí tienen efecto.

Se puede encontrar un ejemplo del cálculo de la especificidad aquí.

Visual Studio Code también os muestra una ayuda para calcular la especificidad de una regla si posáis el ratón sobre los selectores:



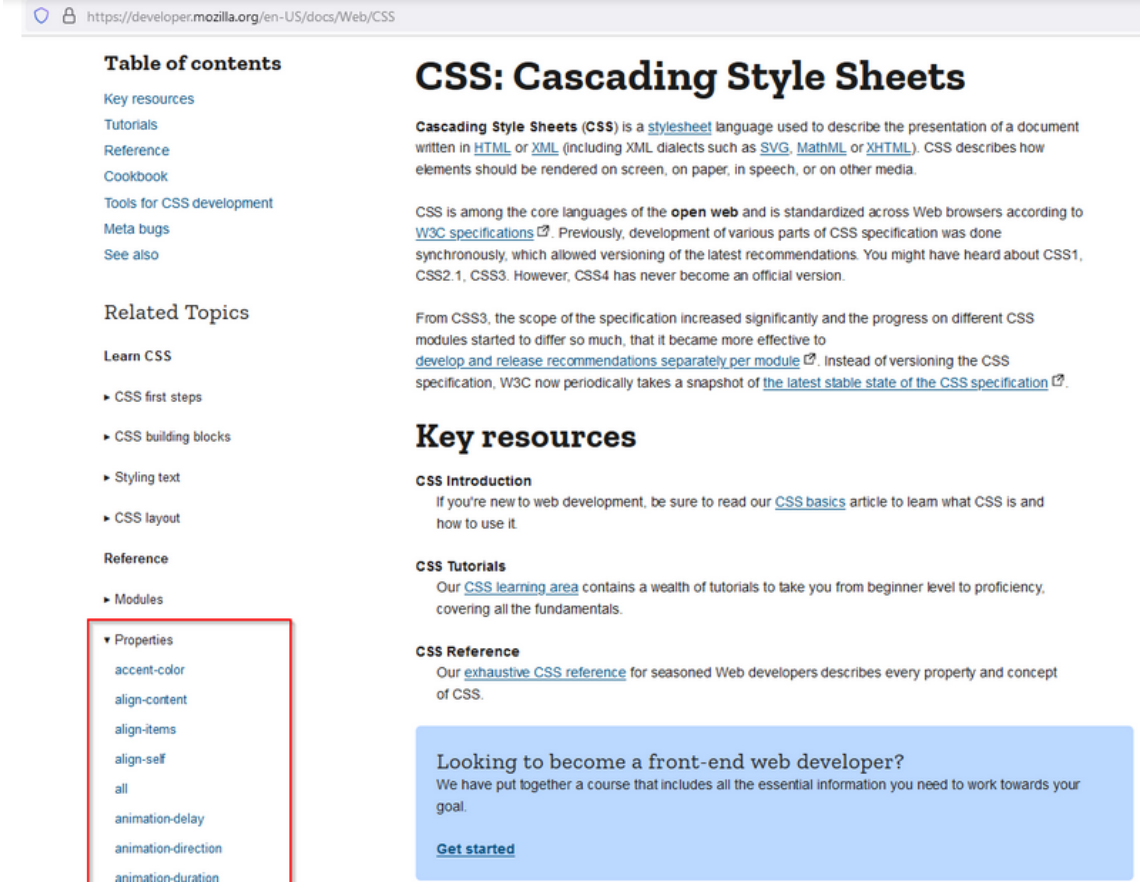
Os muestra entre paréntesis 3 valores separados por comas: El primero es para el número de selectores de valor 100, el segundo selectores de valor 10 y el tercero el número de selectores de valor 1.
A igual número de especificidad, se aplica la última propiedad asignada por orden en la hoja de estilos. La última en aparecer será la más específica.

Herencia

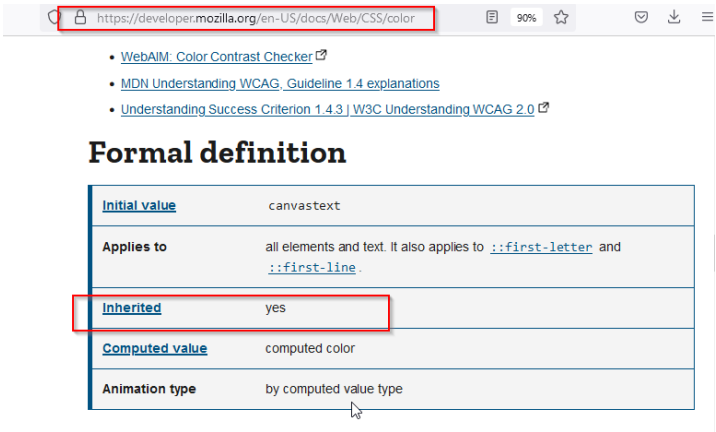
Algunos valores de las propiedades CSS que se han establecido para los elementos padre se heredan en los elementos hijo, pero otras no. El color (`color`) y el tipo de letra (`font-family`) se heredan a menos que se les haya aplicado un color y un tipo de letra diferentes directamente a un elemento hijo. Un ejemplo de herencia de `color` se puede ver aquí.

Sin embargo, la propiedad `width` no se hereda.

Si se hereda o no se puede consultar **para cada propiedad** en https://developer.mozilla.org/en-US/docs/Web/CSS, en el menú **Properties**, se selecciona la propiedad que uno busca:



La información relacionada con la herencia se puede encontrar en la tabla de **Formal definition,** en este caso para la propiedad `color`**:**



5. Unidades en CSS

En CSS es posible utilizar unidades absolutas y unidades relativas.

Unidades absolutas

Su valor no depende del valor de otros elementos. Es el caso, por ejemplo, de **cm**, **mm**, **in** (pulgadas), etc.

La unidad **px** (píxeles) suele incluirse entre las unidades absolutas aunque dependiendo de la densidad de píxeles del dispositivo, la equivalencia entre píxeles físicos y píxeles de CSS puede variar.

Normalmente, de entre las unidades absolutas, es más habitual usar píxeles para representación en pantalla y cm, mm para impresiones.

Unidades relativas

Su valor depende de otros elementos. La siguiente tabla recoge las unidades que veremos en el módulo:

Unidad	Relativa a
em	Tamaño de letra del **elemento padre**, en el caso de propiedades tipográficas como font-size. Un tamaño de fuente de 2em equivale a 2 veces el tamaño de fuente del elemento padre. En el caso de otras propiedades, como `width`, se trata del tamaño de la fuente del propio elemento.
rem	Tamaño de la letra del **elemento raíz**. En HTML el elementor raíz será **html**. 2rem será 2 veces el tamaño del elemento raíz.
vw	1% del ancho de la ventana gráfica. 20vw equivale al 20% del ancho visible del navegador.
vh	1% de la altura de la ventana gráfica. 30vh equivale al 30% de la altura visible del navegador.

Se recomienda usar medidas relativas por su mayor capacidad de adaptación a diferentes tipos de dispositivos y de estados de la ventana del navegador.

Aquí podemos consultar cómo se comportan las unidades px, vw y em.

Porcentajes

Los porcentajes son unidades relativas **al elemento padre**.
Se pueden consultar un par de ejemplos aquí.

Recursos:

https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Values_and_units
https://www.w3schools.com/css/css_units.asp

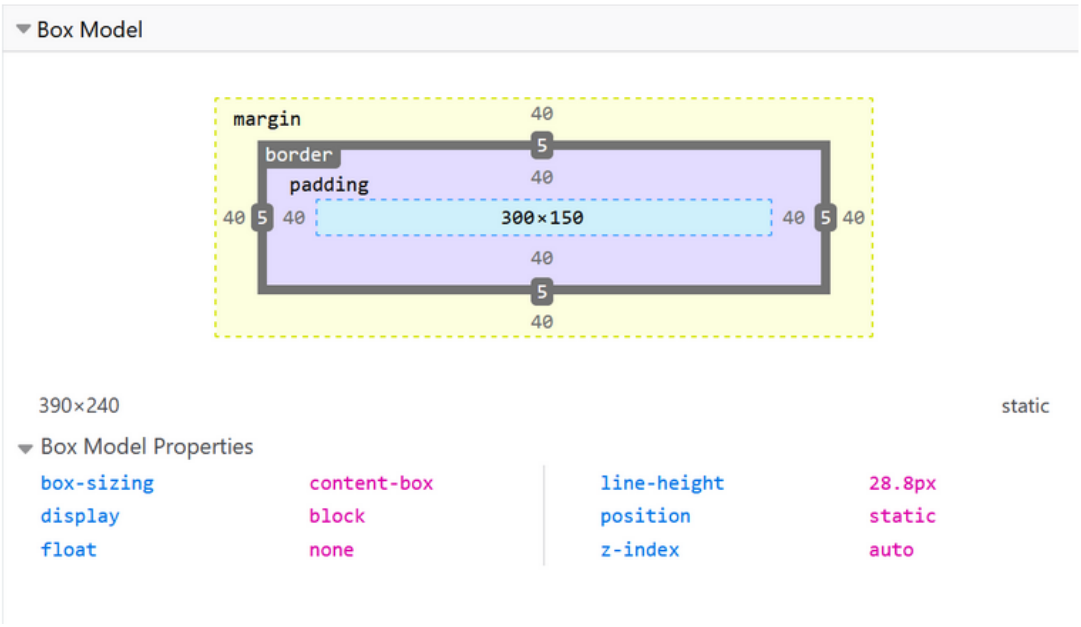
6. Modelo de caja

MODELO DE CAJA ESTÁNDAR

Los elementos en bloque se muestran en CSS utilizando el modelo de caja estándar. Este divide en 4 partes la representación de un elemento:

- El **contenido de la caja** (o *content box*): El área donde se muestra el contenido, cuyo tamaño puede cambiarse utilizando propiedades como `width` y `height`.
- El **relleno de la caja** (o *padding box*): El relleno es espacio alrededor del contenido; es posible controlar su tamaño usando la propiedad `padding` y otras propiedades relacionadas.
- El **borde de la caja** (o *border box*): El borde de la caja envuelve el contenido y el de relleno. Es posible controlar su tamaño y estilo utilizando la propiedad `border` y otras propiedades relacionadas.
- El **margen de la caja** (o *margin box*): El margen es la capa más externa. Envuelve el contenido, el relleno y el borde como espacio en blanco entre la caja y otros elementos. Es posible controlar su tamaño usando la propiedad `margin` y otras propiedades relacionadas.

Esto se puede ver gráficamente en las herramientas del desarrollador de Firefox, en la pestaña "**Disposición**":



El espacio que ocupa la caja usando el modelo de **cajas estándar** será **en realidad la suma del contenido, relleno y borde**: Por ejemplo, en la imagen anterior:
Ancho real de la caja: 300 (contenido) +40x2 (padding) + 5x2 (borde) = 390px
Alto real de la caja: 150 (contenido) + 40x2 (padding) + 5x2 (borde) = 240px

Por defecto, los navegadores usan el modelo de cajas estándar.

MODELO DE CAJA ALTERNATIVO

Cuando se utiliza este modelo alternativo se establece ***width*** y ***height*** se establece el ancho y alto de la caja visibles en la página y no del elemento contenido. Ya no será necesario sumar el relleno y el borde para obtener las dimensiones reales. Consecuentemente, el tamaño del contenido se verá reducido.
Para conseguir este efecto habría que aplicar la propiedad:

```
box-sizing: border-box;
```

Aquí se explica cómo se debe añadir dependiendo de si se aplicará a un elemento o a todos.

Aquí podéis ver un ejemplo para comparar ambos modelos.

6.1. Propiedades CSS aplicables al modelo de caja

	Propiedades CSS	Función	Ejemplos		
Relleno	padding	Versión abreviada o shorthand de relleno con 1, 2, 3 ó 4 medidas*	padding: 5%;	padding: 10px 5px;	padding: 2px 3px 4px;
	Existen versiones para cada dirección: padding-top, padding-bottom, padding-left, padding-right	Permiten establecer de forma individual el relleno para una dirección en concreto	padding-top: 1vh;	padding-right: 2px;	padding-bottom: 3%;
Margen	margin	Versión abreviada o shorthand de margen con 1, 2, 3 ó 4 medidas*. El valor **auto** establece los márgenes horizontales de forma automática por el navegador. Se aprecia cuando el elemento tiene un ancho definido.	margin: 2%	margin: 0 auto;	margin: 2px 3px 4px;
	Existen versiones para cada dirección: margin-top, margin-bottom, margin-left, margin-right	Permiten establecer de forma individual el margen para una dirección en concreto.	margin-top: 1vh;	margin-right: 2px;	margin-bottom: 3%;
Borde	border-width. Existen versiones para cada dirección	Establece la anchura del borde. Permite usar 1, 2, 3 ó 4 medidas*	border-width: 5px;	border-width: thick;	border-top-width: 5px;
	borde-style. Existen versiones para cada dirección	Establece el tipo de borde. Permite usar 1, 2, 3 ó 4 medidas*	border-style: double;	border-style: dashed;	border-left-style: dotted;
	border-color. Existen versiones para cada dirección	Establece el color del borde. Permite usar 1, 2, 3 ó 4 medidas*	border-color: #00ff00;	border-color: green;	border-bottom-color: blue;
	border. Existen versiones para cada dirección	Versión abreviada o shorthand de: border-width border-style (obligatorio) y border-color	border: 2px dotted red;	border-top: dashed;	border-left: 1px solid red;
	border-radius	Permite redondear las esquinas del borde. Permite usar 1, 2, 3 ó 4 medidas con significado diferente al de relleno y margen. Consultar en el enlace.	border-radius: 10px 15px	border-radius: 15px 50px 30px;	border-radius: 10px;

Los porcentajes en relleno y margen se calculan con respecto a la **anchura del elemento padre**.

*Las propiedades señaladas que permiten expresarse con 1, 2, 3 ó 4 medidas se aplica lo siguiente:

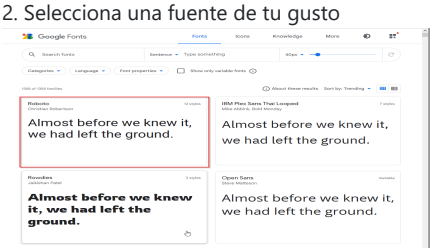
Número de medidas utilizadas	Explicación	Ejemplo
4	El primer valor se aplica arriba, el segundo a la derecha, el tercero abajo y el cuarto a la izquierda (giro en el sentido del reloj)	border-width: 5px 10px 15px 20px;
3	El primer valor se aplica arriba, el segundo valor al eje horizontal (derecha e izquierda) y el tercer valor, abajo	margin: -20px 5px 10;
2	El primer valor se aplica al eje vertical (arriba y abajo) y el segundo valor al eje horizontal	padding: 10px 5px;
1	Se aplica el mismo valor en las 4 direcciones	margin: 10%;

7. Propiedades básicas

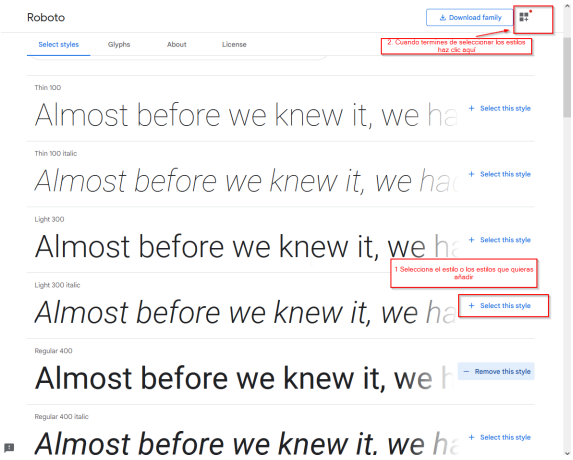
Además de las propiedades CSS que ya hemos visto, podemos destacar como propiedades básicas las siguientes:

Propiedades de tipo de letra o fuente	
Propiedad CSS	**Breve descripción**
color	Indica el color del texto.
font-size	Indica el tamaño de la fuente. Puede ser un tamaño absoluto, relativo, porcentaje o un conjunto de valores predefinidos: small, medium, large, etc.
font-family	Establece la familia a la que pertenece la fuente. Si el nombre de una fuente tiene espacios se utilizan comillas. El valor es el nombre de la familia fuente. Pueden utilizarse varios nombres separados por comas, en caso de que una fuente no sea soportada por un navegador. Se recomienda terminar con nombres genéricos de familias como "serif", "sans-serif", etc.
font-weight	Define el grosor de los caracteres. Los valores que puede tomar son: normal, bold, bolder, lighter, 100, 200, ..., 900. 400 se considera normal.
font-style	Determina si la fuente es normal o cursiva. El estilo oblique es similar al cursiva. Los valores posibles son: normal, italic, oblique.
font-variant	Determina si la fuente es normal o mayúsculas pequeñas. Los valores que puede tomar son: normal , small-caps
line-height	El alto de una línea y por tanto, el espaciado entre líneas.
font	Versión abreviada o shorthand de otras propiedades de font en el orden que se indica a continuación: font-style, font-variant, font-weight, font-size/[line-height], font-family. Los valores han de estar separados por espacios. Solo son obligatorios font-size y font-family.

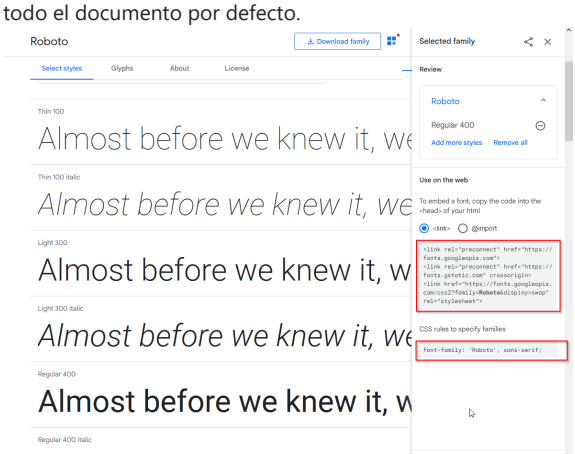
Es posible utilizar las fuentes de Google o Google Fonts. Para ello:
1. Visita la página de Google Fonts
2. Selecciona una fuente de tu gusto



3. Selecciona uno o varios estilos de esa fuente. Cuando termines, haz clic en el icono de la parte superior derecha.



4. Utilizaremos la fuente a través del elemenot <link>. Para ello, han de copiarse y pegarse las etiquetas señaladas bajo <link> dentro de la etiqueta <head> del documento HTML.
A continuación, ha de añadirse en la hoja des estilos la propiedad *font-family* dentro del selector deseado. Normalmente será *body* si queremos que se aplique a todo el documento por defecto.



5. Existe otra forma de importar las fuentes de Google Fonts a través de la regla @import, que permite importar el contenido de una hoja de estilos CSS en la hoja de estilos actual, pero nosotros no la usaremos.

Propiedades relacionadas con texto	
Propiedad CSS	**Breve descripción**
text-decoration	Versión abreviada o shorthand de otras propiedades. Establece si el texto está subrayado, sobrerayado o tachado y permite especificar el color o el estilo de línea. Los valores que puede tomar text-decoration-line son: none, underline, overline, line-through.
text-align	Indica la alineación del texto. Los valores que puede tomar son: left, right, center o justify
text-indent	Determina la tabulación de la primera línea. Los valores que toma son una longitud, en unidades CSS, o un porcentaje de la establecida.
text-transform	Nos permite transformar el texto, haciendo que tenga la primera letra en mayúsculas de todas las palabras, todo en mayúsculas o minúsculas. Los valores que puede tomar son: capitalize, uppercase, lowercase o none
vertical-align	Establece la alineación vertical del texto.

Propiedades relacionadas con el fondo o background	
Propiedad CSS	**Breve descripción**
background-color	Indica el color de fondo del elemento.
background-image	Permite colocar una imagen de fondo del elemento.
background-repeat	Indica si ha de repetirse la imagen de fondo y, en ese caso, si debe ser horizontal o verticalmente.
background-attachment	Especifica si la imagen ha de permanecer fija o realizar un scroll. Los valores que pueden tomar son: scroll o fixed.
background-position	Establece la posición inicial de la imagen de fondo. Por defecto, en la esquina superior izquierda. Las medidas pueden expresarse en porcentaje, tamaño, o valores predefinidos [top, center, bottom] [left, center, rigth]. Primero se expresa la componente horizontal y luego la vertical. Consultar enlace para más información.
background	Versión abreviada o shorthand de otras propiedades de background anteriores.

8. Propieades de maquetación y posicionamiento

La propiedad display

La propiedad display especifica cómo se va a tratar visualmente un elemento html de forma **externa** (con respecto a sus elementos hermanos) e **interna** (con respecto a sus hijos):

- **block**: el elemento se tratará como un elemento en bloque
- **inline**: el elemento se tratará como un elemento en línea
- **inline-block**: podrá aplicarse un width y un height y se respetan el relleno y el margen al elemento (esto no ocurre con un display:inline). Además, seguirá el flujo normal de los elementos, sin crear una nueva línea por sí mismo y permitiendo que nuevos elementos sigan a continuación.
- **none**: el elemento desaparece de la página. No se respeta el espacio que ocuparía el elemento y es como si no existiese. En el caso de que se quisiese que se respetase su espacio se puede usar otra propiedad CSS: visibility: hidden. Se puede ver una comparativa de ambos valores y propiedades en el enlace.

Se puede ver una comparativa de los 3 primeros valores aquí

Con respecto a su visualización **interna** existen más valores relacionados con la disposición en una tabla (table, table-row, table-cell, etc.) o en una lista (list-item) y la disposición de elementos en una sola dimensión (**flex**) (en una fila o en una columna) y **grid** (en filas y columnas).

8.1. Posicionamiento básico

La propiedad CSS position

Para el posicionamiento básico existe la propiedad *position* que puede tomar 5 posibles valores: static, relative, absolute, fixed y sticky.

Se dice que **un elemento está posicionado si su valor es diferente de static**, el valor por defecto.

Valor de position	Breve descripción
static	Es el valor por defecto. El elemento se posicionará según el flujo de la colocación de los elementos html y de si se tratan de elementos en bloque o en línea
relative	El elemento se posicionará con respecto a su posición natural. Las propiedades **top** y **bottom** especifican el desplazamiento vertical desde su posición original empezando desde el borde superior e inferior respectivamente: Un desplazamiento top: 10px indicará que se desplazará 10 px hacia abajo desde su posición normal. Las propiedades **right** y **left** especifican su desplazamiento horizontal comenzando por el borde derecho e izquierdo respectivamente. Estas 4 propiedades pueden usar unidades absolutas o relativas, porcentajes con respecto a la altura/anchura del elemento contenedor padre. Se pueden usar valores negativos y solo tienen efecto en elementos posicionados. No tendrán efecto en elementos con position: static.
absolute	El elemento se posicionará con respecto a su ancestro más próximo posicionado (con position diferente de static o con respecto a body si no hubiese un ancestro posicionado más próximo). Se usan las propiedades **top**, **bottom**, **right** y **left** con respecto a ese ancestro más próximo posicionado o con respecto a body si fuese el caso.
fixed	El elemento se posicionará con respecto a la ventana del navegador con las propiedades **top**, **bottom**, **right** y **left**.
sticky	Es un tipo de posicionamiento híbrido de los posicionamientos relativo y fijo. Un elemento con posicionamiento sticky es tratado como un elemento posicionado relativamente hasta que cruza un límite especificado por las propiedades de posicionamiento **top**, **bottom**, **right** y **left**. Este límite se alcanza haciendo scroll o desplazamiento normalmente en vertical. Cuando se llega a un punto determinado del desplazamiento se trata como con posicionamiento fixed. Un uso de este tipo de posicionamiento se puede ver en los menús que permanecen en la parte superior de la página, aún cuando el usuario se desplaza verticalmente hacia abajo. Se trata de un valor experimental y no siempre funciona. Depende de las propiedades de su ancestro más cercano y sus propiedad `overflow`

Para las propiedades o valores en estado experimental, cada navegador puede establecer unos prefijos. De esta forma se puden experimentar cambios o adaptaciones sin colisionar con otras implementaciones de otros navegadores o con la versión final de las propiedades o sus valores. Por ejemplo:

```
position: -webkit-sticky; /* Safari */
```

```
position:   sticky;
```

Cada navegador usa un prefijo:

- Mozilla usa `-moz-`
- Chrome/Opera/Safari usan `-webkit-`
- Microsoft usa `-ms-`

La propiedad CSS z-index

z-index: Indica mediante un entero (positivo o negativo) el posicionamiento de los elementos en caso de solapamiento en el eje Z (profundidad). El de mayor índice se situará en el frente. Solo tiene impacto si se aplica a elementos posicionados (cuyo valor de position es distinto de static). A igual índice, el último en el html se mostrará al frente. Su valor por defecto es **auto**, que equivale a 0.

Sin z-index, los elementos se mostrarán apilados en este orden:

1. El fondo y los bordes del elemento raiz
2. Bloques descendientes en el flujo normal, en orden de aparición (en HTML)
3. Elementos posicionados descendentemente, en orden de aparición (en HTML)

Fuente y ejemplo:
https://developer.mozilla.org/es/docs/Web/CSS/CSS_Positioning/Understanding_z_index/Stacking_without_z-index

9. FlexBox

El modelo Flexbox permite distribuir en una sola dimensión (en eje vertical u horizontal) los elementos hijos directos de **un elemento padre contenedor**, que llamaremos **contenedor flex.**

Los hijos directos de un contenedor flex se convierten automáticamente **en flex items.**

La dirección en la que se dispondrán los elementos se llamará el **eje principal.** El otro eje, perpendicular al principal, será el **eje cruzado o transversal.**

En el elemento contenedor deberá tener la propiedad **display:flex**

Propiedades CSS para trabajar con el modelo Flexbox

Propiedades en el contenedor padre:

	Propiedad CSS	Descripción	Valores
Contenedor flex	flex-direction	Establece la dirección en la que se dispondrán los elementos: • *row*: filas • *column*: columnas • Si se usan las versiones con *reverse*, los elementos se sitúan en el orden inverso al establecido en el documento HTML El valor por defecto es **row**.	row|column|row-reverse|column-reverse
	flex-wrap	Establece el comportamiento en caso de que los hijos de un elemento flexbox desborden el contenedor: • *wrap* crea una nueva fila/columna para acomodar los hijos. • *wrap-reverse* también crea una nueva fila/columna para acomodar los hijos, pero en orden invertido • *nowrap* no se crea una nueva fila/columna. Valor por defecto: **nowrap**	wrap|nowrap|wrap-reverse
	flex-flow	Versión abreviada o *shorthand* de flex-direction y flex-wrap. Pueden indicarse solo flex-direction, solo flex-wrap o ambas. (Si solo se ofrece un valor, el otro será el valor por defecto).	
	Alineación de los hijos establecida en el contenedor padre		
	Propiedad CSS	**Descripción**	**Valores**
	align-items	Controla dónde se ubican los elementos flexibles en el **eje transversal o cruzado**. • *stretch*: ensancha todos los elementos flexibles para rellenar el elemento contenedor en el eje cruzado • *center*: centrado en el eje cruzado • *flex-start y flex-end*: alinean los elementos al inicio o al final del eje cruzado • *baseline*: los hijos se desplazan en el eje cruzado para alinear la línea base (donde comienza su contenido) Valor por defecto: **stretch**	flex-start|flex-end|center|stretch|baseline
	justify-content	Controla dónde se ubican los elementos flexibles sobre el **eje principal**. • *space-around*: distribuye todos los elementos de manera uniforme sobre el eje principal y deja un poco de espacio en cada extremo • *space-between*: similar a space-around, pero sin espacio en los extremos • *space-evenly*: se reparte el espacio de forma uniforme entre los extremos y entre elementos hijo Valor por defecto: **flex-start**	flex-start|flex-end|center|space-between|space-around|space-evenly
	align-content	Ajusta las líneas dentro de un contenedor flex cuando se permite wrap y hay espacio extra en el eje transversal o cruzado	flex-start|flex-end|center|space-between|space-around|space-evenly

Propiedades CSS en los flex items

	Propiedad CSS	Descripción	Valores
Flex items	order	Especifica el orden utilizado para disponer los elementos en el contenedor en orden ascendente según el valor. Se permiten números negativos. El valor por defecto es **0**.	
	align-self	Alinea a los hijos flex en el eje transversal o cruzado, sobreescribiendo el valor de align-items.	flex-start|flex-end|center|stretch|baseline

	flex-grow	Es un valor sin unidades y especifica la cantidad de espacio disponible sobrante sobre el eje principal que ocupará cada elemento hijo flex. Mismo valor para todos los hijos indica que todos ocuparán igual cantidad de espacio libre. Si un hijo tiene un valor 2 y otro hijo tiene el valor 1, el primero ocupará el doble de espacio libre disponible que el segundo. Valor por defecto: **0**	número
	flex-shrink	Es un valor sin unidades y especifica el factor de contracción de un flex item cuando el tamaño de los hijos desborda el tamaño del contenedor. Cuanto mayor sea su valor, más podrá encoger el flex item con respecto a sus hermanos. No se permiten números negativos. Valor por defecto: **1**	número
	flex-basis	El valor de tamaño del contenido (con modelo de caja estándar) **inicial,** siempre que el tamaño del contenedor lo permita. Si hubiese una contradicción entre width (para rows)/height (para column) y flex-basis, tiene mayor prioridad flex-basis. Por defecto, **auto**.	• unidades absolutas o relativas o porcentajes relativos al tamaño del contenedor padre (width para row, height para column) • *auto*: dependerá del width y height establecidos • *content*: dimensionamiento automático, basado en el contenido del elemento flexible.
	flex	Propiedad abreviada o shorthand para flex-grow, flex-shrink y flex-basis. Los valores iniciales son respectivamente **0, 1, auto.** Es posible indicar solo 1 valor, 2 valores o los 3. Si se omite alguna propiedad, sus valores se aplicarán de la siguiente forma: • Si se omite flex-grow -> su valor es 1 • Si se omite flex-shrink -> su valor es 1 • Si se omite flex-basis -> su valor es 0%	/* Un valor, número sin unidades: flex-grow */ flex : 2; /* 2, 1, 0%*/ /* Un valor, width/height: flex-basis */ flex: 30px; /* Dos valores: flex-grow | flex-basis */ flex: 1 30px; /* flex-shrink sería 1 */ /* Dos valores: flex-grow | flex-shrink */ flex: 2 2; /* Tres valores: flex-grow | flex-shrink | flex-basis */ flex: 2 2 10%;

Un resumen de todas estas propiedades con ejemplos, se puede encontrar aquí

Recursos

- https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Flexbox
- https://www.w3schools.com/css/css3_flexbox.asp

Diferencia entre flex: 1; y flex-grow: 1;

La clave está en la diferencia de flex-basis.

	flex-grow	flex-shrink	flex-basis	Descripción
flex-grow:1;	1	1 (valor por defecto)	auto (valor por defecto)	**auto** usará las propiedades width (si flex-direction: row) o height (flex-direction: column). Si no están establecidas width o height, se calculará el tamaño del contenido de forma automática. Equivaldría a flex-basis:content; Para determinar el espacio que usará cada item, primero se asigna el contenido a cada hijo que le corresponda y después se reparte el espacio libre del padre, si lo hubiese, en función de flex-grow. Se conoce como reparto **flex relativo.**
flex: 1;	1	1 (valor si se omite con flex)	0% (valor si se omite con flex)	Se establece un valor inicial, antes de que el espacio libre del padre se distribuya, en cero. Se ignora el espacio que le correspondería a cada hijo flex y **se reparte todo el espacio** del padre como si estuviese todo disponible. Los hijos crecen en función de flex-grow. Se conoce como **reparto flex absoluto.**

Se observa un ejemplo en la documentación, del que he sacado la siguiente imagen:

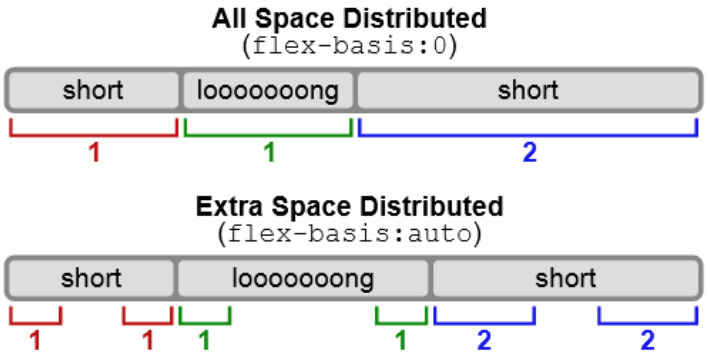


***Figure 7** A diagram showing the difference between "absolute" flex (starting from a basis of zero) and "relative" flex (starting from a basis of the item's content size). The three items have flex factors of '1', '1', and '2', respectively: notice that the item with a flex factor of '2' grows twice as fast as the others.*

En la primera distribución "All space distributed", se usa:

- flex: 1; para el primer short: Esto equivale a flex: 1 1 0%;
- flex: 1; para looooong. Esto equivale a flex: 1 1 0%;
- flex: 2; para el segundo short. Esto equivale a flex: 2 1 0%;

Se observa que el segundo hijo con *short* ocupa el doble que sus dos hermanos. Se hace un **reparto flex absoluto.**

Justo debajo, se usa la distribución "Extra space distributed":

- flex-grow: 1; para el primer short: Esto equivale a flex: 1 1 auto;
- flex-grow: 1; para looooong. Esto equivale a flex: 1 1 auto;
- flex-grow: 2; para el segundo short. Esto equivale a flex: 2 1 auto;

En este caso, en primer lugar, se conserva el espacio necesario para que cada hijo muestre su contenido y, en segundo lugar, se distribuye el espacio sobrante en el padre en función de flex-grow.

Se observa que:

- el segundo hijo con *short* recibe el doble de espacio sobrante que el primer hijo con *short* y, como su contenido es el mismo, su ancho también es el doble.
- el segundo hijo *short* recibe el doble de espacio sobrante que el elemento con looooong, pero la diferencia entre la anchura total que ocupa looooong y el segundo short no es exactamente el doble, pues entra en juego el contenido propio de cada item.

Se hace un **reparto flex relativo.**