

CSS

Hojas de Estilo en Cascada Cascading Style Sheets

CONTENIDO

INTRODUCCIÓN	5
Definición	5
ESTRUCTURA Y SINTAXIS DE CSS	6
Reglas CSS	6
Aplicación de reglas CSS	8
Prioridades y orden en CSS.....	9
Cascada, especificidad y herencia	10
MAQUETACIÓN CON CSS.....	14
Modelo de caja	14
Reglas básicas	15
Modelos de maquetación: Display.....	16
display: inline	16
display: inline-block	16
display: block	16
display: none	16
visibility: hidden	16
display por defecto de etiquetas html	17
display: table o inline-table	18
propiedad column-count y columns.....	19
Propiedad box-sizing: border-box	22
Colapso de Márgenes Verticales o «margin collapse»	24
SELECTORES CSS	25
Selectores básicos	26
Elementos	26
Identificadores	26
Clases.....	27
Selectores avanzados	28
Selectores avanzados basados en las relaciones de parentesco	30
• Selector descendiente (espacio)	30
• Selector hijo >	30
• Selector de hermano adyacente +	30

• Selector de hermano cualquiera ~	30
Selectores avanzados basados en la selección de atributos	32
Selectores Pseudoclase	33
Combinadores Lógicos	36
Selectores de Pseudoelemento	38
Propiedad content	38
PROPIEDADES CSS	41
Unidades absolutas	41
Unidades relativas: em, rem, ex, ch y %	42
Unidades flexibles (viewport)	46
Unidades de relativas: Porcentajes	49
Dimensiones: width, height, max-width, min-width, max-height, min-height	49
Márgenes, Bordes y Paddings	59
Estilos para el texto	60
Función calc()	61
Estilos para las listas	62
Propiedades de color	63
Formas de indicar un color	63
Función RGB	64
Formato HSL	65
Canales Alfa	66
Herramientas para seleccionar colores	67
Propiedad background-image	68
Gradientes o degradados	74
Propiedad opacity	76
Propiedades overflow y scroll	76
Propiedad overflow-wrap(CSS3) o word-wrap	80
Propiedad hyphens	80
PREFIJOS CSS EN LOS NAVEGADORES	82
Extensión Autoprefixer para Visual Studio Code	82
Herramienta Comprobar compatibilidad CSS en los navegadores	83
REGLAS CSS «AT RULES»	84
PROPIEDAD POSITION	85
Position static.	86
Position relative	87
Position absolute	87
Position sticky	88

Position fixed	89
Centrado vertical de elementos de bloque.....	89
ELEMENTOS FLOTANTES	90
Propiedades float.....	91
Propiedad clear.....	94
TIPOGRAFÍAS CSS.....	95
Características de una tipografía	95
Propiedades de las tipografías	97
La reglas @font-face y @import.....	99
Sitios para descargar tipografías para el diseño web	100
Fonts.com.....	100
Adobe Fonts	100
Cloud Typography	100
Fonts Squirrel.....	100
Font Library	100
MENÚS CSS	101
Menú horizontal CSS.....	101
Menú Vertical CSS.....	102
TRANSICIONES Y ANIMACIONES CSS	103
Transiciones	103
Animaciones.....	105
Regla @keyframes	107
TRANSFORMACIONES 2D	110
Funciones de translación	111
Funciones de rotación.....	112
Funciones de escalado	113
Efecto espejo con CSS	114
Funciones de deformación	114
Función matrix	114
Función perspective	116
Grados sexagesimales.....	116
La propiedad transform-origin o punto de origen	117
PROPIEDAD CURSOR	118
PROPIEDAD BORDER-RADIUS.....	121
PROPIEDAD BOX-SHADOW.....	126
ELEMENTOS DE MAQUETACIÓN: FLEX	129
Propiedades para los contenedores (padre).....	131

flex-direction	132
flex-wrap	133
flex-flow	135
justify-content	135
align-items	136
align-content	138
Diferencias entre align-items y align-content	139
Propiedades para los elementos hijos	140
order	140
flex-grow	140
flex-shrink	141
flex-basis	142
flex	144
align-self	145
Para practicar	145
CUSTOM PROPERTIES O PROPIEDADES PERSONALIZADAS	146
Definir una custom property o variable css	146
Utilizar una custom property (variable CSS)	146
Ámbito de las custom properties	147
PROPIEDAD Z-INDEX	148
Solapamiento de Cajas en CSS	148
Contexto de apilamiento	149
Interferencia de los contextos de apilamiento	151
ANEXO: SELECTORES CSS	153

INTRODUCCIÓN

DEFINICIÓN

CSS son las siglas de Cascading Style Sheets («hojas de estilo en cascada»), y es el lenguaje que se utiliza para definir el aspecto de las páginas HTML.

Se diseñó para separar los datos (contenidos en los documentos HTML) de las reglas de presentación (contenidos en los documentos CSS), ya que, con anterioridad a su aparición, las reglas de presentación formaban parte de las etiquetas HTML.

Esta separación proporciona una enorme versatilidad a la hora de presentar la información, pues un mismo documento HTML puede combinarse con diferentes hojas de estilo CSS para generar distintas presentaciones.

CSS (Cascading Style Sheets) es un mecanismo para agregar estilo (colores, tipos de letra, espaciados, etc.) a las páginas web, pero no es un lenguaje de marcas.

Al igual que HTML, es el consorcio W3C la entidad encargada del mantenimiento y la estandarización de CSS.

La primera versión CSS vio la luz en 1996 y desde entonces ha habido sucesivas modificaciones hasta llegar a la versión actual, denominada CSS3.

Hasta la versión 2.1, todas las anteriores se presentaban como una especificación única. Por cuestiones de organización, el W3C decidió que la especificación de la versión 3 se realizaría por módulos, por lo que ya no existe una única versión actual, sino módulos actualizados a la versión 3 y módulos en versión 2.1.

En la Tabla 3.1 se muestran las diferentes versiones y su año de publicación.

AÑO	VERSIÓN	OBSERVACIONES
1996	CSS1	El W3C ya no mantiene esta versión.
1998	CSS2	El W3C ya no mantiene esta versión.
2011	CSS2.1	Corrige errores de CSS2.
2012	CSS3	Esta versión está dividida en módulos, por lo que no existe una única fecha global de cambio de versión.

La última versión de CSS es la 3, y es posible que no lleguemos a ver ningún CSS4, ya que la construcción por módulos del nivel 3 la convierte en la versión de CSS definitiva.

ESTRUCTURA Y SINTAXIS DE CSS

REGLAS CSS

El funcionamiento de CSS consiste en definir unas reglas de presentación que se van a aplicar a un número indeterminado de elementos del documento HTML, al que están vinculadas. Por lo tanto, se necesitan dos herramientas básicas para crear y aplicar un estilo:

Selectores: son las herramientas que permiten seleccionar el elemento o elementos sobre los que aplicar las reglas.

Declaraciones: son las indicaciones para asignar, mediante pares de **propiedad**-**valor**, el aspecto deseado a los elementos determinados por el selector.

Los selectores y las declaraciones se agrupan en reglas.

- Una regla CSS se compone de un selector y de un bloque de declaración.
- Cada declaración está compuesta por uno o más pares **propiedad**/**valor**
- El bloque de declaración empieza y acaba con los signos de llaves { ... }
- Cada par propiedad/valor se separa por el signo punto y coma ;
- Cada propiedad debe separarse de su valor por el signo dos puntos :

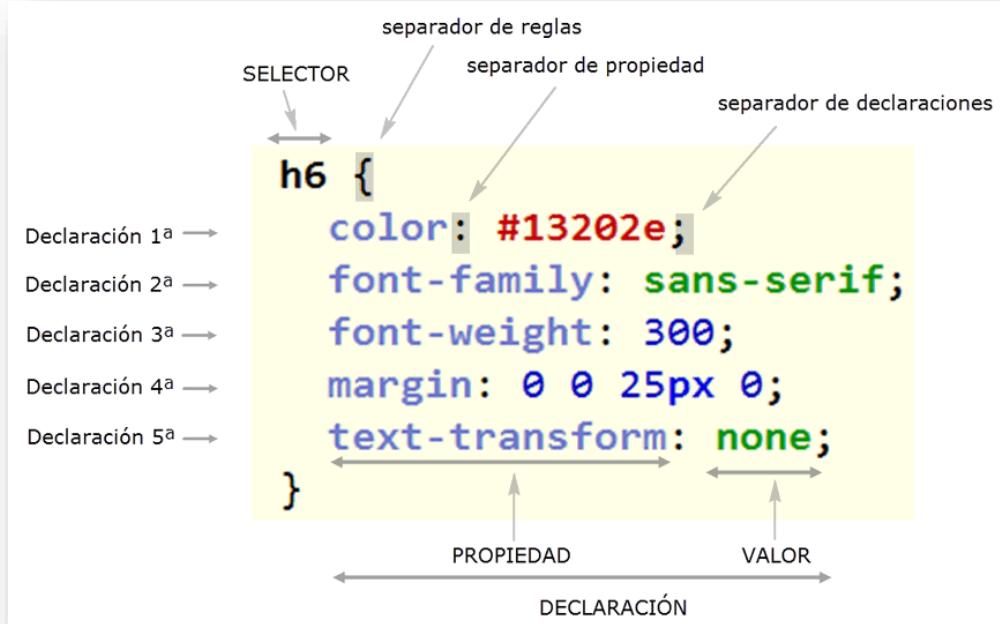


```

selector {
    propiedad: valor; — una regla por línea
    propiedad: valor;
    propiedad: valor; — terminación de regla
}

```

terminación de regla
opcional (al ser la última)



APLICACIÓN DE REGLAS CSS

Para que los estilos que se **implementan con CSS se reflejen** en el HTML, tenemos que encontrar una manera de enlazar el CSS con el HTML.

Esto se puede realizar de tres maneras distintas:

- **CSS EN LÍNEA:** *ATRIBUTO STYLE* contiene información de CSS que se aplica al elemento que contiene el atributo.

```
<p style="text-align:center;background-color:black;">
```

- **CSS INTERNO:** *ETIQUETA <STYLE>* contiene información de CSS que se aplica a todo el documento.

Este elemento HTML `<style></style>` solo puede ser insertado en el encabezado del documento `<head>`.

```
<head>
  <style>
    body {
      text-align: center;
      background-color: black;
      color: white;
    }
  </style>
</head>
```

- **CSS EXTERNO:** Las reglas se almacenan en un documento externo con extensión .css y se establece una relación entre el documento HTML y dicho fichero. Para ello utilizamos la etiqueta `<LINK>`

```
<link rel="stylesheet" type="text/css" href="carpeta/estilo.css" />
```

En ningún caso estas alternativas son incompatibles ni excluyentes, en un mismo documento HTML:

- Se puede añadir una referencia a una o más hojas de estilo externas,
- Incluir un elemento `<style>`
- Y, además, añadir un atributo `style` a un elemento.

Cada una de las opciones para agregar estilos tiene sus ventajas e inconvenientes, pero la opción más utilizada es la de crear un documento externo, ya que permite su reutilización en diferentes documentos.

Las otras opciones a veces son elegidas para realizar prototipos rápidos.

PRIORIDADES Y ORDEN EN CSS

El número de reglas que se puede aplicar a un documento HTML es ilimitado y estas pueden entrar en conflicto.

¿Qué regla va a elegir el navegador en caso de que dos o más apliquen al mismo elemento? La respuesta está en la «C» de CSS.

Pueden existir tres hojas de estilo distintas en una página:

1. La propia del navegador, que es la primera que se aplica por defecto;
2. La que proporciona el usuario (de un navegador), que se asocia al navegador, se aplica por defecto a todas las páginas visualizadas en ese navegador y es muy útil para personas con discapacidad.
3. Por último, están las hojas de estilo del diseñador, que se aplican al final y pueden ser más de una.

El orden en el que se aplican las hojas de estilo es el siguiente:



Figura 2.1 Orden en el que se aplican las diferentes hojas de estilos

La hoja de estilos del navegador se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML: tamaños de letra, decoración del texto, márgenes, etc. Esta hoja de estilos siempre se aplica a todas las páginas web, si un documento HTML no incluye ninguna hoja de estilos propia (hoja del diseñador).

user-agent stylesheet

Los navegadores vienen con una sorprendente cantidad de CSS por defecto, que llamamos user-agent stylesheets (hojas de estilo del agente de usuario). Estos estilos son la razón por la que, sin ningún tipo de CSS por nuestra parte, un `<h1>` es más grande que un `<h2>`, y por la que el `<body>` tiene un margen que muchas veces hay que eliminar.

El principal problema de las hojas de estilo de los navegadores **es que los valores que aplican por defecto son diferentes en cada navegador.**

Aunque todos los navegadores coinciden en algunos valores importantes (tipo de letra serif, color de letra negro, etc.) presentan diferencias en valores tan importantes como:

- los márgenes verticales (margin-bottom y margin-top) de los títulos de sección (`<h1>`, ... `<h6>`)
- la tabulación izquierda de los elementos de las listas (margin-left o padding-left según el navegador)

CASCADA, ESPECIFICIDAD Y HERENCIA

Herencia

La **herencia** provoca que los valores de algunas de las propiedades aplicadas a un elemento contenedor se apliquen también a los elementos contenidos por él. **No** todas las propiedades se ven afectadas por este comportamiento, ya que en algunos casos no tendría sentido. Es decir, que hay propiedades que no se propagan mediante la herencia. Las propiedades CSS que aplicamos se transmiten de forma automática a todos los elementos descendientes o dentro de un elemento contenedor. Esto es lo que llamamos **herencia**.

Por ejemplo, si aplicamos body {font-size:12px;}, se aplicaría este tamaño de letra a todos los elementos dentro de <body>.

Especificidad

La especificidad es un concepto que determina qué regla se aplicará cuando varias reglas compiten para dar formato al mismo elemento. La especificidad se basa en un sistema de puntuación que asigna valores numéricos a los selectores y determina cuál tiene más peso. El que tenga mayor puntuación será el que apliquen los navegadores.

Cascada

Los estilos CSS se aplican en cascada, según unas reglas fijas.

La cascada en CSS es el proceso mediante el cual se determina qué reglas se aplicarán a un elemento específico, teniendo en cuenta la especificidad, el orden de aparición, la importancia y el origen del estilo.

Los factores que participan en la decisión de qué estilo aplicar son los siguientes:

- **Origen e importancia:**
 1. Los estilos inherentes al navegador tienen menos prioridad
 2. Los estilos elegidos por el usuario, prioridad intermedia.
 3. Los creados por el diseñador de la página, que son los que se acaban imponiendo en caso de conflicto.
- **Dentro de los estilos proporcionados por el diseñador**, influye si están definidos:
 1. En línea (declaradas directamente en el HTML con el atributo style) (mayor prioridad)
 2. En un elemento <style> en el head del documento (prioridad intermedia)
 3. En un archivo externo mediante link (menor prioridad)
- **Nivel de especificidad:** cuanto más específico, mayor prioridad

Listado de selectores de mayor a menor especificidad

- **Selectores de ID (#example)**
- **Selectores de clase (.example), selectores de atributos ([type="radio"])** y **pseudo-clases (:hover)**

- **Selectores de tipo (h1) y pseudo-elementos (::before).**
Los selectores de ID tienen mayor especificidad que los de clase, y estos a su vez tienen más especificidad que los selectores de tipo (los que seleccionan elementos HTML) o universales (*).
Ejemplo: `#mild` tiene más especificidad que `.miClase`

- **Orden de aparición o posición del Selector:** Cuando dos selectores tienen la misma especificidad, se aplica el que aparece más abajo en el código.

- **CSS dispone de la declaración !important**

El modificador **!important**, que impide que el valor de una propiedad pueda ser sobrescrito. Lo usaríamos del siguiente modo:

```
p {font-size: 12px !important;}
```

Se debe evitar su uso porque dificulta el mantenimiento y la escalabilidad del código.

La herencia

La herencia es el mecanismo por el cual algunas propiedades de estilo aplicadas a un elemento se transmiten a sus descendientes.

Cuatro tipos de propiedades según su comportamiento con la herencia:

1. Heredada:

- La propiedad se transmite al elemento hijo a menos que se indique otro valor.

Ejemplo: **font-family, color, text-align.**

2. No Heredada:

- La propiedad no se transmite al elemento hijo, a menos que se use la palabra clave **inherit**.

Ejemplo: **border, margin, padding, width, height, background, float, clear position, top, right, bottom, left, z-index.**

3. Parcialmente Heredada:

- La propiedad solo se transmite a ciertos elementos, como aquellos que contienen texto.

Ejemplo: **font-size** en elementos de texto.

4. Heredada por Defecto:

- La propiedad se transmite al elemento **hijo si no tiene ningún valor definido**, pero puede ser sobreescrita con otro valor.

Ejemplo: **font-weight** en elementos de texto.

Estas reglas generales de CSS se pueden modificar mediante una serie de valores comunes a todas las propiedades que tienen efecto sobre cómo se realiza la herencia.

1. **inherit**. Activa la herencia, haciendo que la propiedad en la que se está asignando como valor herede la configuración del elemento padre.
2. **initial**. La palabra clave initial se utiliza para establecer el valor inicial de una propiedad, independientemente de las variables o estilos definidos en el documento. Sus valores de referencia son valores establecidos por el navegador.
3. **unset**. Este valor actúa como **inherit** si la propiedad es heredable y como **initial** si no lo es.

La especificidad en CSS se expresa **mediante cuatro valores**, que son:

1. **Número de identificadores (#id)**: Este valor representa la cantidad de identificadores en el selector.
2. **Número de clases, atributos y pseudoclases (.clase, [atributo], :hover, etc.)**: Representa la cantidad de clases, atributos y pseudoclases presentes en el selector.
3. **Número de elementos y pseudoelementos (div, ::before, ::after, etc.)**: Indica la cantidad de elementos y pseudoelementos en el selector.
4. **Importancia (!important)**: Este valor se aplica si se utiliza la declaración **!important** en una regla.

```
/* Ejemplo de especificidad */
#miId {
    color: red;
    /* Mayor especificidad */
}
.miClase {
    color: blue;
    /* Menor especificidad */
}

/* Ejemplo de herencia */
body {
    font-family: 'Arial', sans-serif;
    /* Propiedad heredada */
}

/* Ejemplo de no heredada */
div {
    border: 1px solid black;
    /* Propiedad no heredada */
}

/* Ejemplo de herencia parcial */
p {
    font-size: 16px;
    /* Propiedad heredada parcialmente, solo elementos de texto */
}

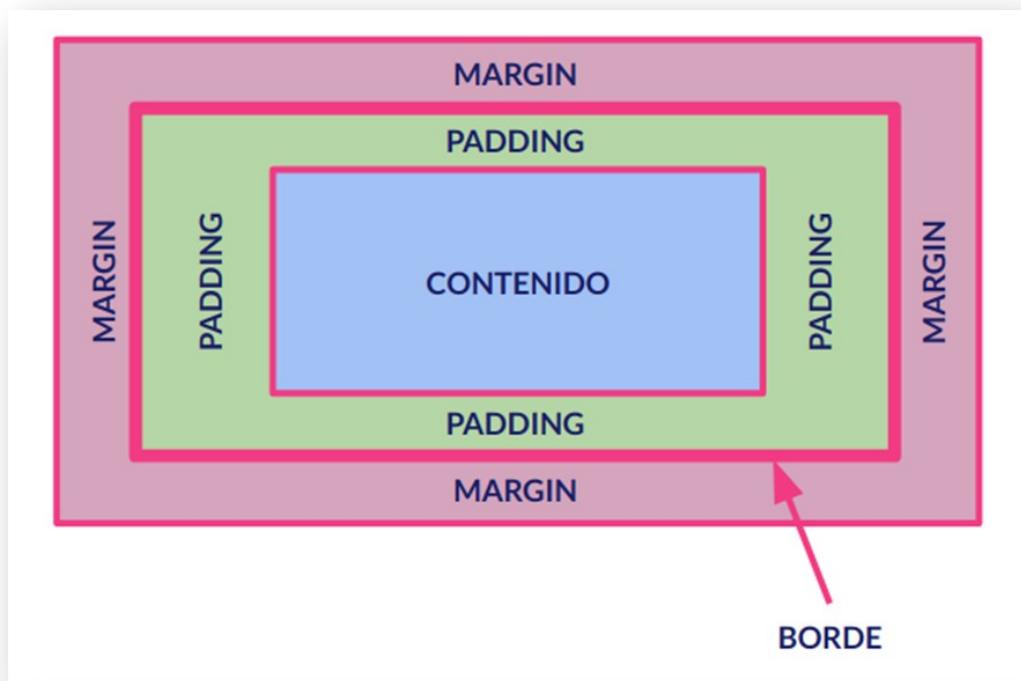
/* Ejemplo de herencia por defecto */
h1 {
    font-weight: bold;
    /* Propiedad heredada si el hijo no tiene valor establecido */
}
```

MAQUETACIÓN CON CSS

Todos los elementos de una página HTML son cajas, pero NO todas las cajas se comportan igual cuando las añadimos en un documento.

MODELO DE CAJA

- **Contenido:** Lo que se pretende mostrar.
- **Padding:** Distancia entre el contenido y el borde de la caja.
- **Borde:** Elemento que marca la división con el resto de los elementos o “cajas” de la página.
- **Margin:** Distancia de la “caja” con el resto de “cajas”.

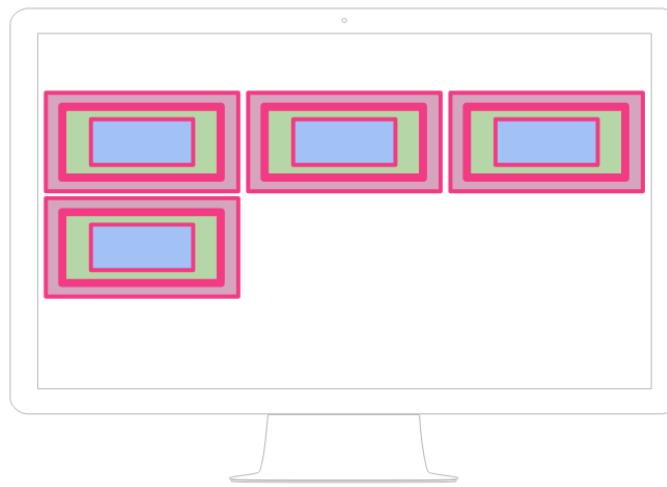


REGLAS BÁSICAS

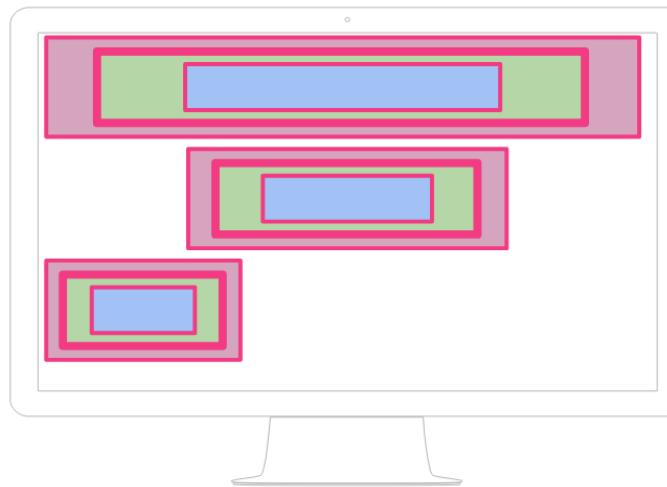
Los navegadores, lo único que hacen es mostrar los elementos de nuestra página HTML en el mismo **ORDEN** en el que los hemos escrito.

Siguen solo dos reglas básicas dependiendo de las propiedades de las cajas:

- Determinados tipos de caja se van poniendo unas detrás de otros mientras quepan en la pantalla. Cuando no caben las cajas pasan a la “siguiente línea” del navegador.



- Otros tipos de caja provocan un “salto de línea”.



Pese a que todos los elementos de mi página HTML son cajas lo cierto es que no todas las cajas se comportan igual cuando las añadimos a una página.

MODELOS DE MAQUETACIÓN: DISPLAY

El comportamiento viene determinado por la propiedad CSS **display**.

Cada etiqueta tiene un valor por defecto para esta propiedad, pero, para conseguir el diseño que queremos, podremos modificarlas si lo estimamos necesario. Los valores que puede tomar son muchos, pero los más usados son:

DISPLAY: INLINE

Elementos en línea

- Los elementos **inline** no rompen el flujo de la línea y se van colocando uno detrás de otro mientras quepan. Aceptan *margin* y *padding*, **pero solo se tienen en cuenta los valores horizontales de margin**.
- Ignoran *width* y *height* (salvo las etiquetas).

DISPLAY: INLINE-BLOCK

- Los elementos inline-block fluyen con el texto y demás elementos como si fueran elementos en-línea.
- Pero respetan el ancho, el alto y los márgenes verticales.
- Son lo mejor de los dos mundos.
- Los elementos **inline-block** funcionan exactamente como los elementos inline, **pero podemos asignarles width y height**.

DISPLAY:BLOCK

Elementos en bloque

Los elementos en bloque rompen el flujo de la línea y provocan “*un salto de línea*” tanto anterior como posterior. Por defecto, si no lo especificamos, **ocuparán todo el ancho del elemento que los contiene**.

DISPLAY: NONE

Elementos con valor none en la propiedad display

El atributo **none** en la propiedad display oculta el elemento, es decir que el elemento existe, pero no es mostrado. La propiedad **display:none** además de ocultar el elemento visualmente, **este no es dispuesto dentro la página, evitando así dejar el espacio del elemento sin ocupar**.

No dejan un espacio vacío (colapsa el espacio) aunque siguen en el código HTML.

VISIBILITY: HIDDEN

Es importante diferenciarlo de la propiedad **visibility:hidden** que únicamente oculta el elemento, pero éste es colocado dentro de la página, dejando así un espacio sin ocupar donde el elemento está alojado (**reserva el espacio**).

DISPLAY POR DEFECTO DE ETIQUETAS HTML

Lista de elementos de BLOQUE:

<address> Información de contacto.	<figure> <small>HTML5</small> Grupos contenido multimedia con una leyenda (ver <figcaption>).	<noscript> Contenido para ser usado si los scripts no son soportados o permitidos.
<article> <small>HTML5</small> Contenido de Artículo.	<footer> <small>HTML5</small> Sección o pie de página.	 Lista ordenada.
<aside> <small>HTML5</small> Contenido adicional.	<form> Formulario de entrada.	<output> <small>HTML5</small> Formulario de salida.
<audio> <small>HTML5</small> Reproductor de audio	<h1>, <h2>, <h3>, <h4>, <h5>, <h6> Niveles de cabecera 1-6.	<p> Párrafo.
<blockquote> Bloque de "cita".	<header> <small>HTML5</small> Sección o cabecera de página.	<pre> Texto preformatado.
<canvas> <small>HTML5</small> Dibujo canvas.	<hgroup> <small>HTML5</small> Grupos información de encabezado.	<section> <small>HTML5</small> Sección de una página web.
<dd> Descripción de definición.	<hr> Regla Horizontal (línea divisoria).	<table> Tabla.
<div> División de documento.	 Elemento de lista.	<tfoot> Pie de tabla.
<dl> Lista de definición.	<main> Engloba el único contenido central del documento.	 Lista no ordenada.
<fieldset> Etiqueta de conjunto de campos.	<nav> Contiene enlaces de navegación.	<video> <small>HTML5</small> Reproductor de vídeo.
<figcaption> <small>HTML5</small> Leyenda de figura.		

(F) MDN

Elementos de LÍNEA:

****, **<i>**, **<small>**, **<abbr>**, **<cite>**, **<code>**, **<dfn>**, ****, **<kbd>**, ****, **<time>**, **<var>**, **<a>**, **
**,
****, **<map>**, **<qp>**, **<script>**, ****, **<sub>**, **<sup>**, **<button>**, **<input>**, **<label>**, **<select>**,
<textarea>,

DISPLAY: TABLE O INLINE-TABLE

Permite asignar un comportamiento de tabla a los elementos que haya dentro (anidados a la etiqueta donde estás colocando el display: table). Es equivalente a la etiqueta TABLE tradicional del HTML.

Atributos básicos del modelo de tabla CSS:

display: table-row Actúa como un elemento fila, etiqueta TR.
display: table-cell Actúa como un elemento celda, etiqueta TD.
table-caption se comportan como la etiqueta CAPTION
table-column se comportan como la etiqueta COL
table-column-group se comportan como la etiqueta COLGROUP
table-footer-group se comportan como la etiqueta TFOOT
table-header-group comportamiento como la etiqueta THEAD
table-row-group comportamiento como la etiqueta TBODY

```
<div class="tabla">
    <div class="fila">
        <div class="col">1.1</div>
        <div class="col">1.2</div>
        <div class="col">1.3</div>
        <div class="col">1.4</div>
    </div>
    <div class="fila">
        <div class="col">2.1</div>
        <div class="col">2.2</div>
        <div class="col">2.3</div>
    </div>
    <div class="fila">
        <div class="col">3.1</div>
        <div class="col">3.2</div>
    </div>
</div>
```

```
.tabla {  
    display: table;  
}  
.fila {  
    display: table-row;  
}  
.col {  
    display: table-cell;  
    padding: 12px;  
    background: #ddd;  
    border: 1px solid chocolate;  
}
```

PROPIEDAD COLUMN-COUNT Y COLUMNS

Las propiedades **columns** y **column-count** en CSS se utilizan para controlar la presentación de un texto o contenido en columnas, lo que permite dividir el contenido en columnas más pequeñas.

column-count

La propiedad column-count se utiliza para especificar el número de columnas en las que se debe dividir el contenido de un elemento.

Por ejemplo, column-count: 3; divide el contenido en tres columnas. El contenido se reorganizará automáticamente en las columnas, y el ancho de cada columna se ajustará para ocupar de manera uniforme el espacio disponible dentro del elemento contenedor.

```
.contenedor {  
    width: 80%;  
    float: right;  
    column-count: 3;  
    margin-left: 5%;  
    margin-top: 30px;  
    text-align: justify;  
}
```

Columns

La propiedad **columns** es una forma más avanzada de controlar la presentación de columnas, ya que permite establecer tanto el número de columnas como el ancho específico de cada columna.

```
p {  
  columns: 2 8em;  
  float: left;  
  text-align: justify;  
}
```

Espacio entre columnas: **column-gap**

column-gap establece el tamaño del espacio entre las columnas de un elemento.

Borde entre columnas: **column-rule**:

column-rule-color, **column-rule-width** y **column-rule-style**

Admite tres valores, el color, el grosor y el estilo del borde, como en el caso de los bordes de los elementos.

```
p {  
  columns: 3 20px;  
  column-rule-color: red;  
  column-rule-width: 2px;  
  column-rule-style: dashed;  
  text-align: justify;  
}
```

```
.containerColumns {  
  /* Crea 6 columnas de 100px de ancho cada una */  
  columns: 6 100px;  
  /* Establece una línea roja de 1px entre las columnas */  
  column-rule: 1px dashed rgb(156, 20, 20);  
  /* Establece una separación entre columnas de 60px */  
  column-gap: 60px;  
  text-align: justify;  
}
```

Existen otros modelos de maquetación avanzada:

- **Flexbox**
- **Grid**

PROPIEDAD BOX-SIZING: BORDER-BOX

Propiedad box-sizing con atributo border-box

La maquetación web consiste en disponer estas cajas para que cada una ocupe el lugar que queremos al ser mostradas en nuestro navegador.

Para conseguir esto, nos encontramos que los elementos al ser representados en el navegador ocupan el siguiente espacio:

- **La altura del elemento:** altura del contenido + el padding + el borde.
- **La anchura del elemento:** anchura del contenido + el padding + el borde.

Es demasiado complejo en maquetaciones grandes, debemos sumar

Solución

El **ancho y alto real** que tomará un elemento al añadir border y padding:

- La propiedad **box-sizing** que por defecto tiene el valor **content-box**, modifica las propiedades del modelo de caja y con su valor **content-box**, modifica ancho y alto sumando los valores de border y padding al ancho y alto total del elemento.
- Podemos establecer la propiedad **box-sizing: border-box** y de esta manera no tendremos que echar cuentas con los *bordes* y los *padding*.

El tamaño que le demos al elemento será la suma de todo: contenido + padding + border

*** { box-sizing: border-box; }**

Si cambiamos el valor del atributo **box-sizing** a **border-box**, mediante este valor el tamaño de la caja establecido se mantiene, es decir, que el tamaño del padding y del border no serán sumados al tamaño total del elemento, lo cual, no hará crecer la caja más allá del tamaño especificado **excluyendo el margin** que ya no se añadirá al tamaño total de la caja.

Ojo: Esto permite mantener el tamaño de la caja, pero deja menos espacio para el contenido de esta.

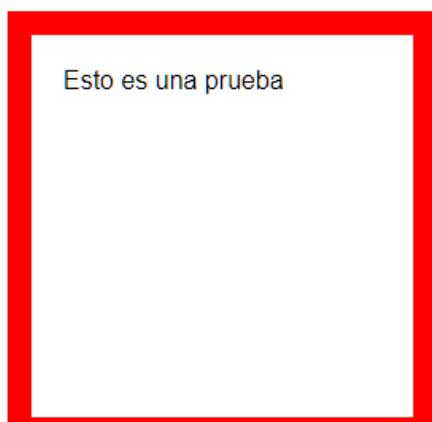
Muy importante, **si el elemento tiene márgenes, estos serán respetados** la propiedad **box-sizing: border-box** no los incluye dentro del tamaño establecido en la propiedad **width**.

Ejemplo propiedad `box-sizing: border-box`

Permite cambiar el comportamiento de este modelo de cajas por uno simplificado donde el tamaño del border y del padding están incluidos en el tamaño dado por propiedades como `width` o `height`.

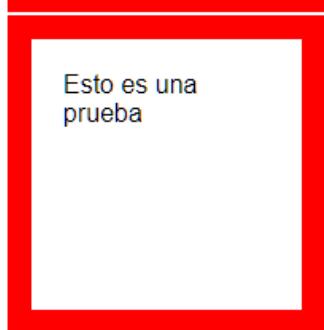
Tamaño `width` y `height` de 200px.

- Borde de 15px por cada lado.
- Relleno de 20px por cada lado.



Esto es una prueba

Observa que, en el caso de arriba, donde usamos `box-sizing: content-box` (modelo por defecto), los tamaños de borde y de relleno se suman al dado por anchos y altos, por lo que el tamaño total del elemento será de 270px.



Sin embargo, en el caso de abajo, `box-sizing: border-box`, los tamaños de borde y de relleno se restan al tamaño dado por anchos y altos, por lo que el tamaño total del elemento será de 200px, sin embargo, el tamaño disponible para el contenido es de 130px (200px - 15px - 15px - 20px - 20px).

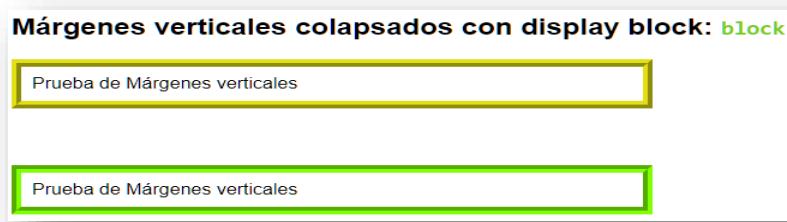
```
.normal {  
    width: 200px;  
    height: 200px;  
    border: 15px solid red;  
    padding: 20px;  
    margin: 2px;  
}  
.cajaFija {  
    width: 200px;  
    height: 200px;  
    border: 15px solid red;  
    padding: 20px;  
    box-sizing: border-box;  
    margin: 2px;  
}
```

COLAPSO DE MÁRGENES VERTICALES O «MARGIN COLLAPSE»

Cuando **dos elementos de bloque** HTML tienen márgenes verticales (inferior y superior) **que se tocan entre sí**, estos dos márgenes colapsan o fusionan en uno y dominará el más grande.

Imaginemos que tenemos dos elementos adyacentes verticalmente, y cada uno de ellos, tiene un margen de 60px definido. Lo normal es pensar que:

- el elemento de la arriba tiene 60px de margen inferior,
- mientras que el elemento de abajo tiene 60px de margen superior,
- al sumarse, predecimos que resultará un tamaño de 120px.
- Sin embargo, lo que ocurre es que se aplica el margin collapse: se refiere a que, en estos modelos de maquetación básica, **los márgenes de dos elementos se colapsan, ocupando el tamaño una sola vez.**



- En el ejemplo anterior el margen, ocupa sólo esos 60px, en lugar de 120px.

El colapso de margen ocurre en cuatro casos básicos:

1. Hermanos adyacentes.
2. Un contenedor padre y su primer elemento hijo.
3. Un contenedor padre y su último elemento hijo.
4. Bloques vacíos.

Excepciones: el colapso ocurre cuando los márgenes verticales aparecen contiguos y se tocan.

El colapso entre un contenedor y su primer elemento NO ocurre si hay algo que separe los dos márgenes; si el elemento contenedor presenta contenido inline antes del primer elemento hijo, por ejemplo, algo de texto, un borde o un padding.

Tenemos una idea equivocada sobre el funcionamiento de los márgenes. El margen sirve para aumentar la distancia entre hermanos. No está pensado para aumentar la distancia entre un hijo y el cuadro delimitador de su parente; para eso está el padding.

Puedes pensar en el padding/border como una especie de muro; si se sitúa entre dos márgenes, éstos no pueden colapsar, porque hay un muro en el camino, incluso 1px de padding interferirá con el colapso de los márgenes.

SELECTORES CSS

Para aplicar un estilo a una parte de un documento hay que hacer referencia a ésta.

Los selectores CSS son reglas o patrones que nos van a permitir seleccionar los distintos elementos de mi página web para poder modificar sus propiedades o estilos.

En las reglas de una hoja de estilo, esta referencia se recoge en **el selector**, que son los elementos, identificadores, clases, o grupos de los anteriores, que aparecen antes de la apertura de llave que engloba las declaraciones.

* es el **selector universal**. Selecciona todos los elementos de la página.

```
* {  
    font-family: "Courier New", Courier, monospace;  
}
```

, Por ejemplo **selector1, selector2** permite agrupar múltiples selectores para aplicar las mismas reglas de estilo a varios elementos diferentes. Esto facilita la aplicación de estilos idénticos a múltiples elementos sin tener que repetir las reglas CSS una y otra vez.

Hay dos tipos de selectores:

- **Selectores propiamente dichos:** Hacen referencia a elementos o atributos presentes en el marcado del documento, como pueden ser un párrafo, el segundo encabezado de una sección concreta, o una clase llamada **destacado**.
- **Pseudoselectores:** Hacen referencia a partes de un documento, pero que no existen en el marcado de este:
 1. Dependen del contexto:
Ejemplo: **la primera línea de un párrafo** dependerá del tamaño de fuente o de la resolución de pantalla
 2. Depende de la interacción del usuario
Ejemplo, si ha situado el ratón sobre un botón o si ha pinchado en un enlace.Como tales, no corresponden a ninguna línea de código HTML que pueda marcarse.

```
selector {  
    prop1: valor1;  
    prop2: valor2;  
    ...  
    propn: valorn;  
}
```

- **selector** hace referencia a la regla o patrón mediante cuya aplicación elegiremos uno o varios elementos de mi página web.
- **propX** son las propiedades que queremos modificar en los elementos seleccionados.
- **valorX** es el valor que daremos a cada una de las propiedades modificadas.

SELECTORES BÁSICOS

Hay tres clases de selectores básicos: Elementos, Identificadores y Clases

ELEMENTOS: Cualquiera de los elementos que hemos visto en HTML es susceptible de ser seleccionado y recibir un estilo, **siempre y cuando aparezca en el body del documento**, sea el propio **body**, o bien sea el elemento **html**.

Su sintaxis es: `elemento{...}`

Ejemplos:

```
html{
    color:#000;
    background-color:inherit;
    font-family:Georgia,"Times New Roman",Times,serif;
}
h1{
    font-weight:normal;
    font-size:140%;
    padding:0.4em;
}
p{
    margin:1em 50px;
}
```

IDENTIFICADORES: Corresponden a los **id** signados a los elementos de la página.

Su sintaxis es: `#identificador{...}`

Ejemplos:

```
#contenido-principal{
    color:#333;
    background-color:transparent;
    font-family:Verdana,Helvetica,sans-serif;
    font-size:0.8em;
    margin:1em;
}
#navegacion{
    display:inline;
    padding:5px;
}
#pie{
    font-size:90%;
    text-align:center;
}
```

CLASES: Corresponden a los atributos `class` incluidos en el código del documento.

Su sintaxis es: `.clase{...}`

Ejemplos:

```
.intro{  
    color:#FFF;  
    background-color:#333;  
    font-family:"Trebuchet MS",Verdana,Helvetica,sans-serif;  
    font-style:italic;  
    padding:0.75em;  
    border:1px dotted #000;  
}  
.error{  
    color:red;  
}
```

Se pueden combinar los selectores básicos para **hacerlos más específicos**. Por ejemplo:

`p.intro` seleccionaría sólo los párrafos que además tuviesen un atributo `class` con el valor `intro`;

`div#pie` sólo seleccionaría aquel elemento cuyo `id` es `pie` y que sea un `div`.

SELECTORES AVANZADOS

Los selectores avanzados permiten acotar las partes del documento a las que se va a aplicar un estilo.

Por ejemplo, hemos visto que aplicando un estilo a `<p>` modificaríamos el aspecto de todos los párrafos de una página. ¿Pero qué ocurre si queremos modificar exclusivamente el aspecto de los párrafos de la columna derecha de información adicional?

Un principiante añade clases y las emplea como selector básico, pero si queremos llegar a ser «buenos programadores», aplicamos los selectores avanzados.

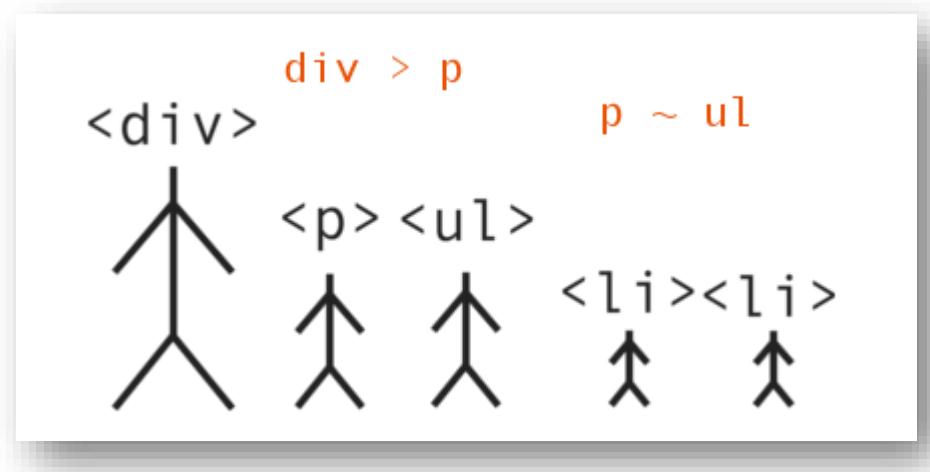
Hay dos tipos de selectores avanzados:

1. Los que se basan en relaciones de parentesco entre los nodos del documento
2. Los que se basan en la selección por medio de atributos.

NOTAS sobre el árbol del documento (DOM)

Si miramos un documento HTML, vemos elementos anidados dentro de otros:

1. Todos los elementos que están anidados en el nivel inmediatamente inferior de un elemento son *los hijos* de ese elemento, el cual a su vez recibe el nombre de *padre*.
2. Dos elementos que comparten el mismo parente son *hermanos*.
3. Todos los elementos anidados dentro de un primer nivel de un elemento se consideran *descendientes* de ese primero.
4. Y, por último, partiendo de un elemento, cada uno de los que lo contiene en algún grado son sus *ascendientes o ancestros*.



La descendencia se refiere al nivel en que se encuentra un elemento HTML con respecto a otro dentro de un documento HTML. El nivel más alto lo tiene la etiqueta <html> y <body>

```
<body>
  <section id='hijo-body'>
    <p>Ejemplo de descendencia</p>
    <span>Fragmento de texto</span>
    <ul>
      <li>Elemento 1</li>
      <li>Elemento 2</li>
    </ul>
  </section>
  <footer>Pie de página</footer>
</body>
```

Analizando la estructura que tienen los elementos html del ejemplo:

- El elemento **body** es padre del elemento **section** y del elemento **footer** pero **no** de los elementos **p** ni **span**.
- El elemento padre de los elementos **p** y **span** es **section** por lo tanto los elementos **p** y **span** son hermanos ya que están al mismo nivel uno del otro.
- Los elementos **footer** y **section** son hermanos.

Es importante notar la diferencia entre hijos y descendentes.

Dentro del elemento **section** del ejemplo:

- Sus **hijos** son los elementos que se encuentran un nivel directamente debajo de el mismo elemento **section**.
- Sus *descendientes* son todos los elementos que se encuentran dentro del elemento **section** sean los mismos hijos o los hijos de los hijos, etc. En el ejemplo los elementos **li** ya no son hijos del elemento **section** pero si son descendientes ya que son hijos de un hijo, que es el elemento **ul**.

SELECTORES AVANZADOS BASADOS EN LAS RELACIONES DE PARENTESCO

- **SELECTOR DESCENDIENTE (ESPACIO)**

elemento elemento

Selecciona cualquier elemento que sea descendiente de otro elemento padre

```
section p{  
    padding : 1rem;  
}
```

En el ejemplo anterior la regla css se aplica a todos los elementos **p** que desciendan de un elemento **section**.

- **SELECTOR HIJO >**

elemento > elemento

Este selector es parecido al anterior, solo que este selecciona a cualquier elemento que sea hijo directo de un elemento padre, no va más allá del primer nivel dentro del elemento.

```
section > p{  
    padding : 1rem;  
}
```

- **SELECTOR DE HERMANO ADYACENTE +**

elemento + elemento

Permite seleccionar un elemento específico que está inmediatamente después de otro elemento dado, siempre y cuando, estén al mismo nivel, es decir que sean hermanos.

```
p + p{  
    color: red;  
}
```

- **SELECTOR DE HERMANO CUALQUIERA ~**

elemento ~ elemento

Es similar al anterior, la diferencia está en que con este selector no necesariamente el elemento dado tiene que ser el siguiente inmediato puede haber entre ellos uno o más elementos, pero mientras sean hermanos y se encuentre después y no antes el selector será aplicado.

```
p ~ p{  
    color: red;  
}
```

Consideremos el siguiente fragmento de código:

```
<div>
    <h2>Título</h2>
    <p>Párrafo 1</p>
    <span>Texto cualquiera</span>
    <p>Párrafo 2</p>
    <p>Último párrafo</p>
</div>
<p>Párrafo 3</p>
```

Al emplear el selector de hermano adyacente `p + p{}` el elemento que se seleccionará:

```
<p>Último párrafo</p>
```

Al emplear el selector de hermano cualquiera `p ~ p{}` los elementos seleccionados serán:

```
<p>Párrafo 2</p>
<p>Último párrafo</p>
```

Ejemplo

```
#main {
    background-color: grey;
}

h1.especial {
    color: blue;
}

h2,
h3 {
    border: 1px solid orange;
}

li {
    color: red;
}

li li {
    color: green;
}

p > img {
    border: 5px solid black;
}

p ~ h3.hermanoCualquiera {
    background-color: pink;
}

p + h3 {
    background-color: blue;
}

img[alt="segunda"] {
    border: 5px solid red;
}
```

SELECTORES AVANZADOS BASADOS EN LA SELECCIÓN DE ATRIBUTOS

En CSS, estos atributos se indican rodeando de corchetes [] al atributo en cuestión.

Hay varias formas de utilizar los atributos CSS, algunas muy potentes y flexibles, basadas en un concepto llamado expresiones regulares en los que están inspirados (aunque no son estrictamente expresiones regulares):

Importante, los fragmentos de código para seleccionar atributos CSS son sensibles a mayúsculas y minúsculas (case sensitive), o lo que es lo mismo, sólo tendrían en cuenta uno de los dos casos.

Atributo	¿Cuándo se aplica el estilo al elemento?
Básicos	
[href]	Si tiene un atributo href .
[href="#top"]	Si tiene un atributo href y su valor es #top .
[class~="manzana"]	Si tiene un atributo class con una lista de valores y uno de ellos es manzana .
[lang = "es"]	Si tiene un atributo lang con una lista de valores, donde uno empieza por es- .
Avanzados	
[href^="https://"]	Si tiene un atributo href y su valor comienza por https:// .
[href\$=".pdf"]	Si tiene un atributo href y su valor termina por .pdf (un enlace a un PDF).
[href*="manzana"]	Si tiene un atributo href y su valor contiene manzana .

Algo importante, **es que estos fragmentos de código para seleccionar atributos CSS son sensibles a mayúsculas y minúsculas** (case sensitive), o lo que es lo mismo, sólo tendrían en cuenta uno de los dos casos.

Para evitar esto, podemos añadir una **i** antes del cierre] del atributo. De esta forma, no será sensible a mayúsculas y minúsculas, **ya que esa i hace referencia a «case insensitive»**

```
a[href$=".pdf" i] {  
    background:red;  
}
```

```

selector #id .clase [atributo] :pseudoclase ::pseudoelemento {
    propiedad : valor ;
    propiedad : valor
}

```

SELECTORES PSEUDOCASE

Las pseudoclasses se utilizan para hacer referencia a elementos HTML que tengan un comportamiento concreto.

Son palabras clave que se añaden a los selectores y que nos indican un ESTADO o POSICIÓN concreto de los elementos seleccionados.

Selectores de pseudoclase: Seleccionan elementos HTML en función de su estado o posición.

Pseudo-clases para enlaces	Descripción
:link	No visitado por el usuario
:visited	Visitado por el usuario
:hover	Modifica el estilo cuando el ratón pasa por encima
:active	Se activa cuando el usuario pulsa el elemento
:target	Se activa cuando se hace clic un enlace con un identificador de destino permitiendo aplicar estilos personalizados a ese elemento específico que se ha convertido en el destino de la navegación.

Pseudo-clase para seleccionar hermanos	Descripción
:first-child	Primer hijo
:last-child	Último hijo
:first-of-type	Primer hermano de su tipo
:last-of-type	Último hermano de su tipo
:only-child	Hijos únicos
:only-of-type	Únicos hermanos de su tipo
:empty	Elementos que no tienen hijos
:nth-child(n)	Enésimo elemento hijo
:nth-last-child(n)	Enésimo elemento hijo contando desde el último
:nth-of-type(n)	Enésimo hermano de su tipo
:nth-last-of-type(n)	Enésimo hermano de su tipo comenzando desde el último

Pseudo-clase para formularios	Descripción
:focus	Se activa cuando el elemento está seleccionado. Normalmente a los elementos <input> de los formulario
:focus-within	Representa contenedores que tengan algún <input> dentro y se activará cuando alguno de sus hijos obtenga el foco.
:enabled	Representa un elemento de la interfaz de usuario que se encuentra en estado activado.
:disabled	Representa un elemento de la interfaz de usuario que está desactivado.
:placeholder-shown	Representa cualquier elemento <input> o <textarea> que esté mostrando actualmente el texto <i>placeholder</i> .
:default	Coincide con uno o más elementos de interfaz de usuario que son los predeterminados entre un conjunto de elementos.

Pseudo-clase para formularios	Descripción
:checked	Se utiliza elementos como <checkbox> y <input type="radio"> activados.
:valid	Coincide con un elemento cuyo contenido es válido. Por ejemplo, un elemento de entrada con el tipo "email", "tel", "url" que contenga una dirección de correo válida, un teléfono o una URL
:invalid	Coincide con un elemento cuyo contenido no es válido. Por ejemplo, un elemento de entrada de tipo 'email' con un formato incorrecto.
:in-range	Se aplica a elementos con limitaciones de rango. Por ejemplo, <input type="range"> cuando el valor seleccionado está dentro del rango permitido.
:out-of-range	Se aplica a elementos con limitaciones de rango. Por ejemplo, <input type="range"> cuando el valor seleccionado está fuera del rango permitido.
:required	Coincide cuando se requiere un elemento de formulario. Elementos <input> o <textarea> con el atributo required.
:optional	Coincide cuando un elemento del formulario es opcional. se selecciona a los inputs o textareas que no contengan el atributo required.
:user-invalid	Representa un elemento con entrada incorrecta, pero sólo cuando el usuario ha interactuado con él.

:target

La pseudo-clase :target en CSS permite seleccionar un elemento específico en una página web cuando se hace clic en un enlace que apunta a él mediante el uso del símbolo "#" en la URL.

Por ejemplo, si tienes un enlace con el siguiente atributo href="#section1" en tu página web, puedes utilizar :target para dar estilo al elemento con el id "section1" **cuando ese enlace es activado:**

```
/* Selecciona un elemento con una ID que coincide con el fragmento de la URL actual */
```

```
:target {
    border: 2px solid black;
}

<a href="#seccion1">Soy un ancla </a>

<section id="seccion1">Ejemplo</section>
```

Puedes utilizar `:target` para cambiar cualquier propiedad de estilo, incluyendo el tamaño, posición, color, etc. Puedes también utilizarlo para mostrar u ocultar elementos de la página o para cambiar el contenido de un elemento.

Ten en cuenta que la pseudo-clase `:target` solo se activa cuando se hace clic en un enlace, no cuando se carga la página.

COMBINADORES LÓGICOS

¿Qué es un combinador lógico?

En CSS, cuenta con una serie de mecanismos para agrupar o combinar selectores de una forma potente y flexible, dentro de una categoría denominada combinadores lógicos.

Combinadores lógicos	
<code>:is()</code>	Seleccionamos varios elementos separándolos por comas y permite combinar con otros selectores.
<code>:where()</code>	Seleccionamos varios elementos separándolos por comas y permite combinar con otros selectores., pero con menor especificidad CSS.
<code>:has()</code>	Permite seleccionar <code>elementos padre</code> que tengan ciertas características en sus hijos.
<code>:not()</code>	Permite seleccionar elementos que no cumplan ciertas características.

```
.contenedor :is( h3,nav) {  
    /* Especificidad (0,2,0) */  
    color: rgb(0, 26, 255);  
}  
  
.contenedor :where(div, li) {  
    /* Especificidad (0,1,0) */  
    color: rgb(255, 0, 0);  
}
```

La pseudo-clase de negación `:not()` es muy útil, ya que permite seleccionar todos los elementos que no cumplen los criterios indicados en sus paréntesis.

```
p:not(.general) {  
    border: 1px solid rgb(156, 8, 8);  
    padding: 8px;  
    background: #FFF;  
}
```

Este código nos indica que todos los párrafos `<p>` que no pertenezcan a la clase `.general`, se les aplique el estilo especificado.

La pseudo-clase `:has()` admite dentro del paréntesis una lista de elementos como argumento y los estilos se aplican a un elemento en función de lo que le siga en el DOM.

Y en el argumento puedes incluir los combinadores de selectores (`>`, `+`, `~` ...) para seleccionar no sólo al padre o abuelo, sino también a hermanos precedentes. También puede incluir clases o id.

```
a:has(> img) {  
    text-decoration: none;  
}
```

En este caso, la propiedad `text-decoration: none` se aplica sobre el enlace `<a>`, sólo si en el interior del enlace existe una etiqueta ``. Este ejemplo podría ser muy útil para eliminar estilos sobre imágenes que son enlaces.

```
section:not(:has(h1, h2, h3, h4, h5, h6)) {  
    color: rgb(0, 26, 255);  
}
```

El selector anterior selecciona los `<section>` que no contienen ningún elemento de los indicados en el argumento (dentro del paréntesis). Basta con que tenga uno cualquiera de los indicados para que no se aplique la regla.

```
h2:has(+ h3) {  
    background: rgb(199, 25, 25);  
}
```

Selecciona todos los `<h2>` que tengan un `<h3>` inmediatamente después de ellos.

SELECTORES DE PSEUDOELEMENTO

Los **pseudoelementos** son selectores especiales que permiten seleccionar y dar estilo **una parte de un elemento HTML**. Además nos permiten añadir contenidos. Y además nos permiten añadir contenido.

La sintaxis de los pseudoelementos está precedida de dos puntos **dobles (::)** para diferenciarlos de las pseudoclases, las cuales sólo tienen **dos puntos (:)**

Pseudo-elemento	Descripción
::first-line	Primera línea de texto de un elemento
::first-letter	Primera letra de la primera línea de texto de un elemento
::before	Añade contenido al principio del elemento
::after	Añade contenido al final del elemento
::selection	Coge la porción del texto que se está seleccionando por el usuario

PROPIEDAD CONTENT

La propiedad **content** es una de las propiedades CSS más poderosas y a la vez más controvertidas. Esta propiedad **permite agregar contenido** generado en CSS a un elemento HTML sin modificar directamente el contenido del DOM, es decir, el contenido generado no existe en el documento HTML y **solo se muestra en la representación visual de la página**.

La propiedad **content** se utiliza exclusivamente en combinación con los pseudoelementos **::before** y **::after**

Como CSS es un lenguaje de hojas de estilos cuyo único propósito es controlar el aspecto o presentación de los contenidos, algunos diseñadores defienden que **no** es correcto generar nuevos contenidos mediante CSS.

```

p::first-letter{
    font-size:200%;
    border:2px outset;
    padding:0.2em;
    margin-right:0.1em;
}

p::first-line{
    text-decoration: underline;
    color: rgb(138,56,7);
}

p::selection{
    color: rgb(138,56,7);
}

```

```

<body>
    <header>
        <h3>Tabla de contenido</h3>
        <nav>
            <ol>
                <li><a href="#p1">¡Salta al primer párrafo!</a></li>
                <li><a href="#p2">¡Salta al segundo párrafo!</a></li>
            </ol>
        </nav>
    </header>
    <section>
        <h3>Probando primera línea y primera letra en un div</h3>
        <div>Esta es la primera línea de un div<br>
            <br>Yo soy la segunda línea del div.</div>
    </section>
    <section>
        <h3>Creando contenido</h3>
        <p id="p1">Puede elegir seleccionar
            <i>este párrafo</i>utilizando un fragmento de URL.
            ¡Haz clic en el enlace de arriba para probar!</p>
        <p id="p2">Este es <i>otro párrafo</i>, también accesible
            desde los enlaces de arriba.</p>
    </section>
</body>

```

```
p::target {
    background-color: gold;
}
/* Agrega un pseudo-elemento dentro del elemento de destino */
p::target::before {
    font: 70% sans-serif;
    content: "🐘🐘"; /*contenido añadido desde CSS*/
    color: limegreen;
    margin-right: .25em;
}
/* Estilo de elementos en cursiva dentro del elemento de destino */
p::target i {
    color: rgb(0, 26, 255);
}
div::selection
{
    background-color:rgb(151, 12, 12);
}
div::first-letter {
    font-size: 150%;
    border: 2px outset;
    padding: 0.4em;
    margin-right: 0.1em;
}
div::first-line {
    text-decoration: underline;
    color: rgb(145, 12, 105);
}
div::after {
    color:red;
    /*Creando contenido desde CSS*/
    content: "[texto añadido al final del div desde CSS]";
}
```

PROPIEDADES CSS

Hay muchísimas propiedades que podemos modificar, algunas son comunes a todas las etiquetas y otras propiedades sólo se pueden modificar en algunas etiquetas.

Podéis acceder a lista completa desde estos enlaces:

[Propiedades CSS - Lista completa](#)

Lista de propiedades referentes a los siguientes aspectos:

- ✓ **Unidades**
- ✓ **Dimensiones**
- ✓ **Márgenes, Bordes y Padding**
- ✓ **Texto**
- ✓ **Color**
- ✓ **Fondos**

UNIDADES ABSOLUTAS

Unidad	Significado	Medida aproximada
in	Pulgadas	1in = 25.4mm
cm	Centímetros	1cm = 10mm
pc	Picas	1pc = 4.23mm
mm	Milímetros	1mm = 1mm
pt	Puntos	1pt = 0.35mm
px	Píxels	1px = 0.26mm
Q	Cuarto de mm	1Q = 0.248mm

UNIDADES RELATIVAS: EM, REM, EX, CH Y %

Unidad	Medida aproximada	Ejemplo
em	Se utiliza para hacer referencia al tamaño actual de la fuente en ese elemento HTML. Por defecto, es un valor aproximado a 16px (salvo que se modifique por el usuario).	$1.5\text{em} * 16\text{px} = 24\text{px}$
ex	La medida ex está basada en la altura de la x minúscula , que es aproximadamente un poco más de la mitad de la fuente actual (depende de la tipografía utilizada).	$1\text{ex} \sim 0.5\text{em}$
ch	La unidad ch por su parte, equivale al tamaño de ancho del 0 de la fuente actual.	$1\text{ch} \sim 1 \text{ carácter}$
rem	En relación al tamaño de la letra (font-size) que tiene la etiqueta HTML o de la pseudoclase :root	
%	Relativa a herencia (concretamente, al elemento padre). El porcentaje se calcula teniendo en cuenta lo que ocupe el padre del elemento dentro del árbol DOM	$50\% = \text{mitad del parente}$



```
:root {  
    font-size: 22px;      /* Tamaño base */  
}  
h1 {  
    font-size: 2rem;     /* El doble del tamaño base: 44px */  
}  
h2 {  
    font-size: 1rem;     /* El mismo tamaño base: 22px */  
}
```

La pseudo-clase `:root` de CSS selecciona el elemento raíz de un árbol que representa el documento.

En HTML, `:root` representa el elemento `<html>` y es idéntico al selector `html`, excepto que su especificidad es mayor.

EM y REM

Las unidades `em` y `rem` son unidades CSS pensadas para trabajar con tipografía, aunque también se utilizan para especificar dimensiones en otros elementos HTML.

Tanto `em` como `rem` son considerados unidades relativas, pero ¿relativo a qué? o ¿respecto a qué? Son relativas al `font-size`.

Las medidas `em` y `rem` **se traducen** a valores de `pixel`. Por ejemplo si se establece `1em` o `1rem` como valor a un elemento HTML el navegador va a computar o convertir ese valor en un valor de píxel (`px`).

La mayoría de los navegadores tienen un estándar en el que el tamaño base de la fuente o `font-size` que asignan al texto **es de 16px**, excluyendo por supuesto a los encabezados `<h1>` a `<h6>` y el elemento `<small>` ya que el navegador les asigna otro tamaño de fuente.

MUY IMPORTANTE:

- Las unidades `em` para la propiedad `font-size` serán relativas al `font-size` del **elemento padre**.
- Pero las unidades `em` en otras propiedades (que no sean `font-size`) serán relativas al **font-size del elemento actual**.
- Las **unidades rem** siempre serán relativas al `font-size` del elemento raíz que es `html` en ese instante.

Las unidades **em** pueden usarse para mucho más que solo para establecer el font-size , pueden usarse en otras propiedades como **padding**, **margin**, **width**, **height**, **max-width**, etc.

Cuando las unidades em se usan en otras propiedades que no sea el font-size , el valor es relativo al propio font-size del elemento.

Ejemplo:

```
.parent {  
    font-size: 18px;  
}  
.child {  
    font-size: 1.5em; /* = 27px */  
    padding: 2em 1em;  
}
```

El **padding-top** y **padding-bottom** del elemento hijo .child será de 54px , si hacemos los cálculos: 2em x 27px = 54px.

El **padding-left** y **padding-right** del elemento hijo .child será de 27px , si hacemos los cálculos: 1em x 27px = 27px.

Un elemento HTML automáticamente **hereda su valor del font-size** de su elemento padre y esto puede llevar a un comportamiento no deseado en elementos anidados con valores **em**.

rem

La unidad rem, abreviatura de **root em**, se basa siempre en el valor del font-size del elemento raíz, que es el elemento <html>.

Los **valores en rem** se calculan multiplicando en base al valor del font-size del elemento root (<html>) del documento:

- Si el elemento <html> no tiene un font-size especificado, se utiliza el valor por defecto del navegador que es 16px .
- Si el font-size del <html> es 16px, 1rem sería igual a 16px **en cualquier parte del documento**.
- Al utilizar la unidad rem se ignoran los valores del font-size de los elementos padre y solo se tiene en cuenta el valor del font-size del <html>.

¿Cuándo usar em y rem?

El uso de las medidas em y rem van de la mano con los temas de responsive design, usabilidad y accesibilidad. En realidad no hay una unidad mejor, todo depende de las preferencias personales.

Cuándo y cómo utilizar em

Para la propiedades *border*, *padding* y *border-radius* en componentes como botones o inputs. El botón crece de manera proporcional al tamaño de fuente del texto.

Cuándo y cómo utilizar rem

Para la propiedad *font-size*, es recomendable usar [unidades rem para el texto](#) porque tiene importantes beneficios de accesibilidad, se adaptan a las preferencias del usuario, mientras que si utilizan píxeles puede sobrescribir estas preferencias.

Nota:

El tamaño de fuente base en los navegadores son 16px, puede ser cambiado (en las preferencias del navegador) por el usuario a cualquier valor entre 9px y 72px.

UNIDADES FLEXIBLES (VIEWPORT)

Existen unas unidades de "nueva generación" que resultan muy útiles, porque dependen del viewport (**región visible de la página web en el navegador**). Con estas unidades podemos hacer referencia a un porcentaje concreto del tamaño específico que tengamos en la ventana del navegador, independientemente de si es redimensionado o no.

Unidad	Significado	Medida aproximada
vw	viewport width	$1\text{vw} = 1\%$ ancho de navegador
vh	viewport height	$1\text{vh} = 1\%$ alto de navegador
vmin	viewport minimum	$1\text{vmin} = 1\%$ de alto o ancho (el mínimo)
vmax	viewport maximum	$1\text{vmax} = 1\%$ de alto o ancho (el máximo)

La unidad vw (viewport width)

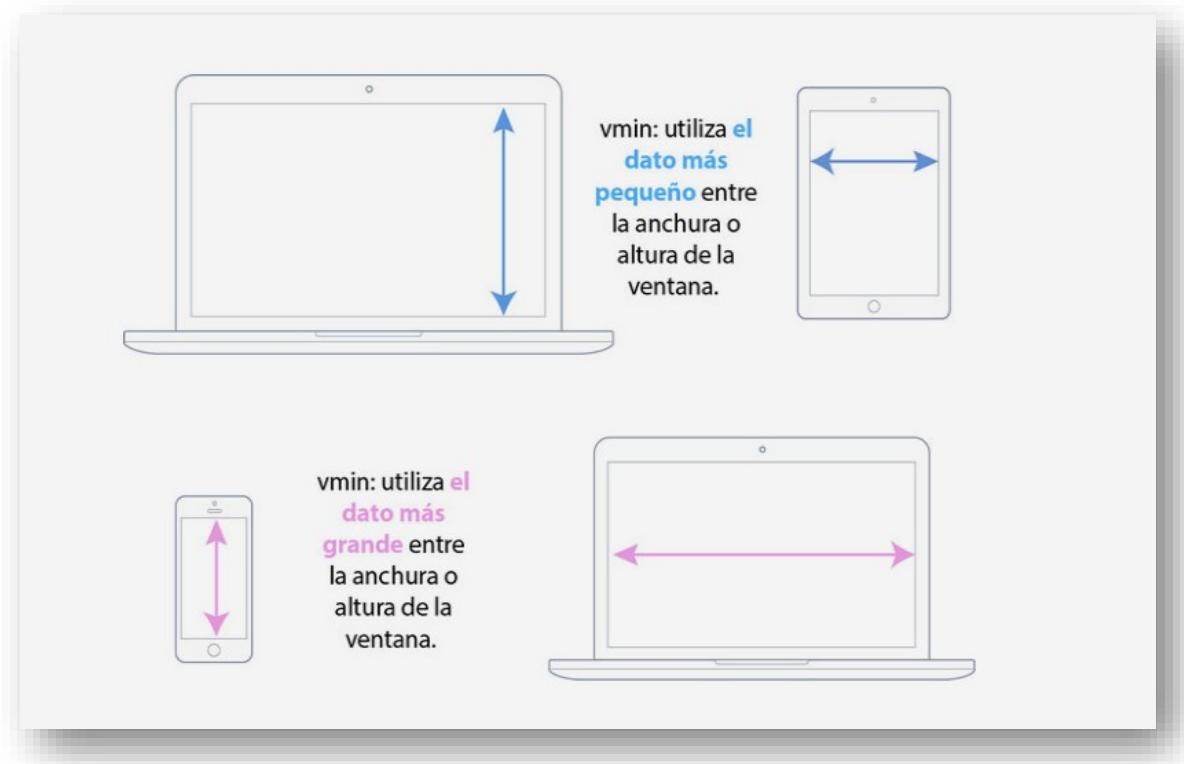
La unidad **vw** hace referencia al ancho del viewport (región visible de la página en el navegador). Por ejemplo, si definimos 50vw, estamos indicando un 50% del ancho actual del navegador. Esto nos permite utilizar tamaños que dependan de las dimensiones de la ventana del navegador.

La unidad vh (viewport height)

La unidad **vh** hace referencia de la misma forma al alto del viewport. Por ejemplo, mientras 50vw hace referencia al 50% del ancho del navegador, si indicamos 50vh estaremos haciendo referencia a la mitad (50%) del alto del navegador.

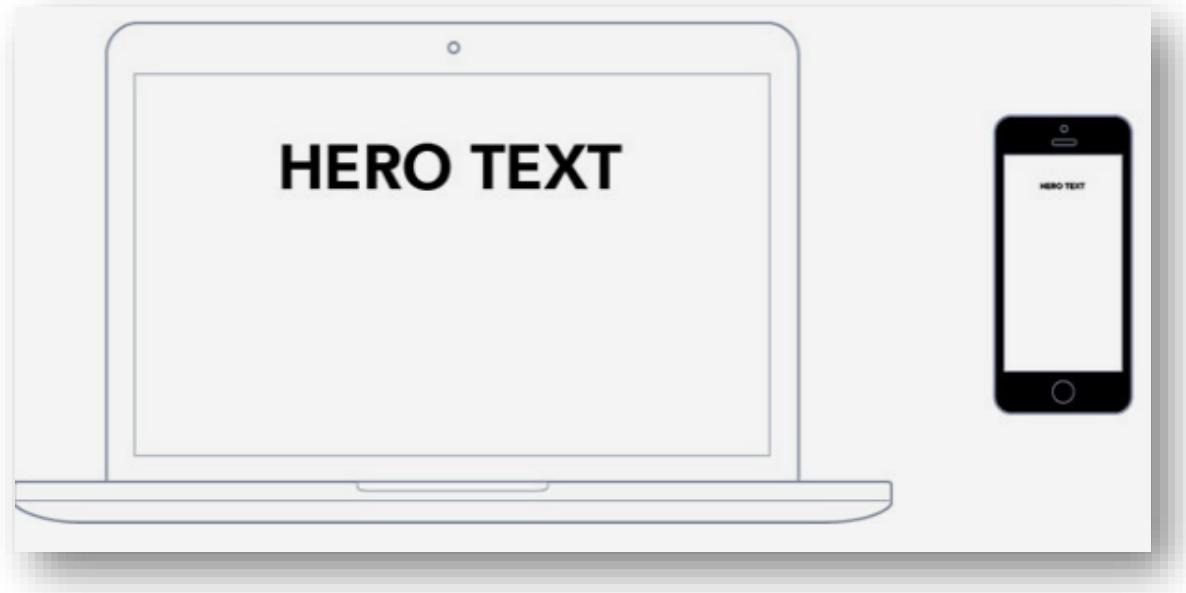
Las unidades vmin / vmax

Las unidades **vmin** y **vmax**. La unidad **vmin** (viewport minimum), simplemente hace referencia al valor más pequeño (mínimo) entre el tamaño de ancho y el tamaño de alto del viewport del navegador. La unidad **vmax** (viewport maximum) hace referencia al valor más grande (máximo) entre el tamaño de ancho y el tamaño de alto del viewport del navegador.



Mantener bajo control el texto

Se pueden utilizar las unidades vw para los textos de cabecera, pero hacerlo conlleva un problema importante: el texto que comienza en un tamaño razonable con las unidades vw, rápidamente se sale de control con tamaños relativamente grandes o pequeños de la ventana gráfica:



Una alternativa es establecer el font-size de la cabecera con vmin:

```
h1 {  
    font-size: 20vmin;  
    font-family: Avenir, sans-serif;  
    font-weight: 900;  
    text-align: center;  
}
```

Midiendo el tamaño de la letra en vmin, el tamaño del texto de cabecera no se hará más grande o más pequeño en un navegador totalmente expandido si se estrecha, ya que la unidad responde a la altura de la ventana gráfica.

Pero si la ventana se hace más estrecha de su anchura - en otras palabras, si la página se mueve en una orientación vertical - los textos se hacen más grandes o más pequeños.

UNIDADES DE RELATIVAS: PORCENTAJES

Cuando especificamos un porcentaje como valor en un elemento, este se calcula en base a la medida del elemento padre, y para que esto funcione, la medida del elemento padre directo debe estar especificada.

Si el elemento padre no tiene un tamaño específico establecido, es decir, si no tiene un ancho o alto definido explícitamente, entonces se aplicará el tamaño predeterminado de los elementos según su tipo (elemento de bloque, de línea...) y contenido.

DIMENSIONES: WIDTH, HEIGHT, MAX-WIDTH, MIN-WIDTH, MAX-HEIGHT, MIN-HEIGHT

Las propiedades `width` (ancho) y `height` (alto) en CSS se utilizan para establecer las dimensiones de un elemento HTML, es decir, el tamaño horizontal y vertical del elemento.

- **width** para la anchura de nuestro elemento.
- **height** establece la altura de los elementos **de bloque**.

Un ejemplo básico:

```
#first img {  
    width: 50%;  
}  
#second {  
    width: 600px;  
    ...;  
}
```

Propiedad width

La propiedad `width` se utiliza para establecer el ancho de un elemento HTML. Puede definirse con valores numéricos seguidos de una unidad de medida (por ejemplo, px, em, %, etc.) o con valores especiales como **auto**.

Particularidades:

- Si se establece un valor numérico con una unidad de medida, el ancho del elemento se fijará en esa cantidad específica. Por ejemplo, `width: 200px;` establece un ancho de 200 píxeles.
- Si se utiliza el **valor auto**, el ancho del elemento se calculará automáticamente basado en su contenido y las propiedades de los elementos padres. Esto puede ser útil para elementos como imágenes y elementos de bloque sin ancho definido, que tomarán el ancho de su contenido.

- La propiedad width **afecta solo al contenido del elemento, no incluye el padding, el borde o el margen**. Si se desea incluir estos valores en el tamaño total del elemento, se debe considerar el modelo de caja (box-sizing) y utilizar la propiedad box-sizing: border-box;.

Elementos a los que se aplica:

- La propiedad width se aplica a todos los elementos de bloque (display: block;) y elementos reemplazables (display: inline-block;), como <div>, <p>, , , <section>, entre otros.

Propiedad height

La propiedad *height* se utiliza para establecer la altura de un elemento HTML. Al igual que *width*, puede definirse con valores numéricos seguidos de una unidad de medida o con el valor especial auto.

Particularidades:

- Si se establece un valor numérico con una unidad de medida, la altura del elemento se fijará en esa cantidad específica. Por ejemplo, height: 100px; establece una altura de 100 píxeles.
- Al igual que con width, si se utiliza el **valor auto**, la altura del elemento se calculará automáticamente basado en su contenido y las propiedades de los elementos padres. Esto es útil para elementos sin altura definida que tomarán la altura de su contenido.
- La propiedad height **afecta solo al contenido del elemento**, y el padding, borde y margen no se incluyen en el tamaño total del elemento a menos que se use box-sizing: border-box;.

Elementos a los que se aplica:

- **La propiedad height se aplica principalmente a elementos de bloque (display: block;) y elementos reemplazables (display: inline-block;).**
- **En ciertos casos**, como con imágenes, también puede aplicarse a elementos de línea (display: inline;) que tienen una propiedad **line-height** establecida.
- La propiedad **line-height** se utiliza para establecer la altura de línea en un elemento. La altura de línea se refiere a la altura total ocupada por una línea de texto, desde la parte superior de la letra más alta hasta la parte inferior de la letra más baja.

```
p:first-child {  
    /*altura de línea será 1.5 veces el tamaño de fuente.*/  
    line-height: 1.5;  
}  
  
p:nth-child(2) {  
    /*altura de línea 150% del tamaño de la letra actual*/  
    line-height: 150%;  
}  
  
p:nth-child(3) {  
    /*altura de línea el valor del navegador*/  
    line-height: normal;  
}
```

Casos específicos en los que no tiene sentido o no es apropiado aplicar la propiedad height.

- Elementos de línea (display: inline;): Los elementos de línea, como el texto y los enlaces (, <a>, etc.), generalmente no tienen una altura definida ya que su altura se ajusta automáticamente al contenido que contienen. Por lo tanto, aplicar height a elementos de línea puede no tener ningún efecto visible.
- Elementos vacíos o con contenido muy pequeño: Si un elemento no tiene contenido o solo contiene espacios en blanco o caracteres de espacio, establecer una altura fija con height puede no ser visible o no tener ningún impacto en el diseño.

Es importante tener en cuenta que establecer valores fijos para width y height puede afectar la capacidad de respuesta adaptativa (responsive) de un diseño.

Por lo tanto, en la mayoría de los casos, es preferible utilizar unidades de medida relativas, como porcentajes (%), em, rem, para permitir que los elementos se adapten a diferentes tamaños de pantalla y dispositivos.

width: auto vs. width:100%

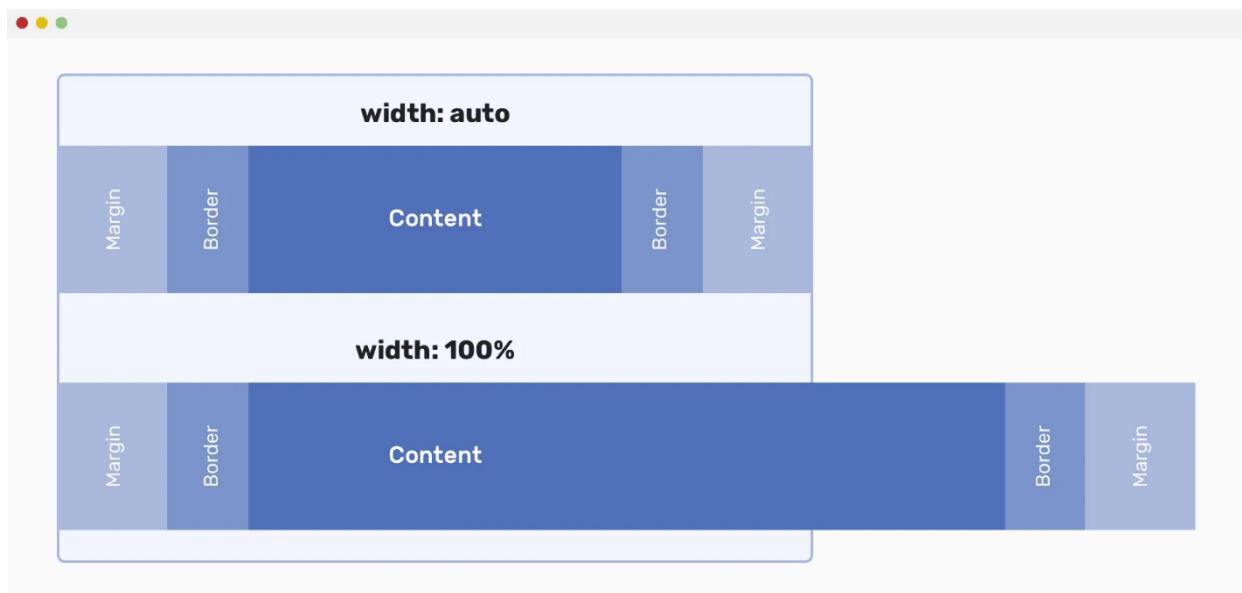
El ancho (width) inicial de los elementos HTML de bloque como <div> o <p> es auto (automático), esto hace que se expandan y ocupen todo el espacio horizontal disponible de su bloque contenedor.

La palabra clave **auto** usada en la propiedad width, significa que el navegador va a calcular automáticamente el ancho de un elemento en función del espacio que tenga disponible de su bloque contenedor sin llegar a desbordarse.

Si se especifica **width:100%** el ancho total del elemento será el 100% del bloque que lo contiene, es decir, el elemento hijo ocupa todo el ancho del elemento contenedor.

Sin embargo la propiedad width por defecto **no tiene en cuenta el margin, padding y border**, si alguna de estas propiedades tiene un valor distinto de cero, el tamaño final del elemento será superior al de su contenedor y como resultado el elemento hijo se sale de su contenedor padre.

A menos que añada la declaración **box-sizing: border-box** en cuyo caso solo afectan los márgenes al elemento hijo



height: auto vs. height:100%

Establecer **height: auto** en un elemento, permite que el navegador calcule automáticamente la altura del elemento basándose en el contenido interno de ese elemento.

Cuando se declara la propiedad CSS height en porcentaje, esta hace referencia a la altura del elemento con respecto a su elemento contenedor (padre).

Funcionamiento:

Si el elemento contenedor no tiene una altura establecida de manera explícita, la altura en porcentaje especificada se ignora y se reemplaza por el valor predeterminado de la propiedad height, que es auto. En este caso la altura depende del contenido del elemento.

Excepción:

El elemento contenedor **no tiene una altura establecida** de manera explícita pero la **propiedad position: absolute** está presente en el elemento, la altura en porcentaje **sí** se calculará con respecto al elemento contenedor.

La presencia de **position: absolute** permite que la altura en porcentaje se calcule en relación con el elemento contenedor, incluso cuando no se establece una altura explícita en dicho contenedor.

La altura se calculará en relación con la altura del contenedor primario más cercano que no tenga **position: static** (que es el valor predeterminado de position).

Ejemplo

Si se tiene un elemento dentro de un contenedor, es decir, una relación padre-hijo, el contenedor padre debe tener un alto (height) declarado, ya que si la caja contenedora **no tiene un alto definido no se podrá calcular el porcentaje.**

HTML

```
<div class="parent">
    <div class="child"></div>
</div>
```

CSS

```
.parent {
    width: 500px;
    height: 200px;
    background-color: rebeccapurple;
}
.child {
    width: 50%;
    height: 50%;
    background-color: skyblue;
}
```



.parent { height: 200px; }

Si se elimina la propiedad height del contenedor padre, ambas cajas desaparecen y, por lo tanto, el valor de height es cero. Esto sucede porque no tiene contenido el elemento.

La palabra clave auto usada en la propiedad height, significa que el navegador va a calcular el alto de un elemento en base al contenido de dicho elemento.

Al añadir texto al contenedor principal, ahora la altura por defecto es auto, `height: auto`, lo que quiere decir que la altura del elemento se calcula en base al contenido.

Si se añade más texto, la caja seguirá creciendo. Y esto lejos de ser un problema es algo útil, ya que si se pone una medida fija en la altura independientemente de la unidad de medida que uses, el texto se puede desbordar o se salirse del contenedor principal.

```
.parent {  
    width: 500px;  
    color: white;  
    /* height: 200px; */  
    background-color: rebeccapurple;  
}  
.child {  
    width: 50%;  
    /* height: 50%; */  
    color: black;  
    background-color: skyblue;  
}
```

Texto añadido al contenedor padre valor `height:auto`

Texto añadido al contenedor hijo,
que va creciendo según su
contenido

Hay que tener cuidado al usar la propiedad height, en muchas ocasiones es mejor dejar que el contenido determine el alto del elemento.

Si se necesita una altura determinada es mejor utilizar las propiedades min-height o padding-bottom.

Las propiedades max-width y min-width

Se utilizan para controlar el tamaño horizontal de un elemento en una página web. Aunque todas están relacionadas con el ancho de un elemento, cada una tiene un propósito y comportamiento específico.

max-width

La propiedad max-width se utiliza para establecer **un ancho máximo permitido para un elemento**. Define el ancho máximo que puede tener el elemento antes de que se detenga su crecimiento y comience a reducirse si el contenedor padre es más pequeño.

Comportamiento: Si el ancho real del elemento (ya sea establecido por un valor fijo de width o el tamaño del contenido) es menor o igual al valor especificado en max-width, el elemento se mostrará con ese ancho.

Si el ancho real es mayor al valor de max-width, el ancho se ajustará para asegurarse de que **no exceda** el valor establecido, es decir, impide que el valor de width sea más largo que el valor especificado en max-width.

min-width

La propiedad min-width se utiliza para establecer **un ancho mínimo permitido para un elemento**. Define el ancho mínimo que debe tener el elemento, incluso si el contenido es más pequeño.

Comportamiento: Si el ancho real del elemento (ya sea establecido por un valor fijo o el tamaño del contenido) es mayor o igual al valor especificado en min-width, el elemento se mostrará con ese ancho. Si el ancho real es menor al valor de min-width, el ancho se ajustará para asegurarse de que no sea menor al valor establecido.

html, body { height: 100% }

- El elemento <body> es un elemento de bloque.
- El <body> tiene la propiedad height inicialmente establecida en **auto**.
- Por lo tanto, la altura inicial de body es 0, a medida que se añaden elementos la altura del body irá creciendo.
- Ahora queremos que el elemento body ocupe todo el ancho y el alto de la pantalla, entonces vamos a asignar un alto del 100%

```
body { height: 100%;}
```

Sin embargo, esto no funciona...

- La razón es que el elemento <body> también **tiene un elemento padre que es <html>**
- El <html> pese a no tener declarado un valor para la propiedad height aparenta ocupar todo el alto de la ventana del navegador en todo momento, pero solo es eso, “una apariencia”, ya que su altura no es toda la ventana.
- La propiedad height por defecto del <html> es **auto** lo que significa que su altura crece según el contenido que haya en <body>, y en ausencia de contenido y/u otras propiedades que lo modifiquen la altura del elemento <html> es 0 (cero).

¿Te has preguntado si el elemento <html> tiene un parent?

- Según la especificación **el elemento raíz no tiene ningún parent**. Sin embargo, **si tiene un bloque contenedor**.
- Al asignar medidas en porcentajes al elemento html, este **se basa en el bloque de contención inicial, es decir el viewport**.
- Como lo dice expresamente la especificación:

Una altura declarada en porcentaje en el elemento raíz (html) es relativa al bloque inicial de contención (viewport).

```
html {  
    height: 100%;  
}  
body {  
    min-height: 100%;  
}
```

Esto permite al elemento `html` hacer referencia al viewport y que tenga un valor de altura igual al 100%.

Como el elemento `html` tiene establecido ya un valor de altura, el `min-height` asignado al elemento `body` le da una altura inicial que coincide con el elemento `html`.

Esto también permite que el `body` supere el alto del 100% si el contenido supera el tamaño de lo visible de la página.

Durante años esta fue la configuración de altura ideal para una página full responsive.

hay otra opción más moderna

```
body {  
    min-height: 100vh;  
}
```

En esta configuración usamos como unidad `vh` (viewport height) para permitir que el `body` establezca una altura mínima basada en la altura completa del viewport.

Si NO asignamos un valor de altura máxima al elemento `html`, asumirá el mismo valor de la altura del elemento `body`.

Por lo tanto, esta solución evita el desbordamiento del elemento `html` presente en la solución anterior y ambos elementos crecen con su contenido.

NOTA:

Puede que tu página tenga instantáneamente una barra de scroll horizontal.

Este problema surge cuando `cualquier elemento`, no solo los elementos `html` o `body`, se establecen con `width 100vw` (viewport width).

Las unidades de viewport **no tienen en cuenta los 10 píxeles aproximados que ocupa la barra de scroll vertical**.

Por lo tanto, cuando se activa la barra de scroll vertical, también se obtiene una barra de scroll horizontal.

Solución: si en un elemento estableces el ancho de página, elige el 100% sobre 100vw para evitar el scroll horizontal.

MÁRGENES, BORDES Y PADDINGS

Son propiedades para establecer las dimensiones de los elementos de la caja.

Para márgenes y paddings tenemos varias formas de hacerlo. Si tenemos en cuenta que A(Arriba)- D (Derecha) - AB (Abajo) - IZQ (Izquierda):

```
selector {  
    /* A -D - AB - IZQ */  
    margin: 20px 50px 20px 50px;  
  
    /* A y AB - DCHA e IZQDA */  
    margin: 20px 50px;  
  
    /* Todos */  
    margin: 50px;  
  
    /* O de manera individual */  
    margin-left: 10px;  
    margin-top: 10px;  
    margin-bottom: 10px;  
    margin-right: 10px;  
}
```

Para el padding sería exactamente lo mismo. Sólo tenemos que sustituir *margin* por *padding*.

En relación con el borde de un elemento tenemos también varias posibilidades. De igual manera lo vamos a ilustrar mediante un ejemplo:

<https://developer.mozilla.org/es/docs/Web/CSS/border-style>

```
/* De manera general */  
border: 1px solid black;  
  
/* Sólo el borde */  
border-color: black;  
  
/* Sólo la anchura del borde */  
border-width: 1px;  
  
/* Sólo el estilo de la línea del borde */  
/* posibles valores solid, dashed, dotted */  
border-style: solid;
```

ESTILOS PARA EL TEXTO

Hay multitud de propiedades para establecer los estilos del texto de mi página web. Algunas de las más destacables son:

```
/* Para establecer el tipo de fuente */
font-family: "Times New Roman", Times, serif;

/* Para establecer el tamaño de la fuente */
font-size: 2em;

/* Para establecer el grosor del tipo de letra */
/* Posibles valores: bold, bolder, lighter */
font-weight: bold;

/* Para establecer la alineación texto */
/* Posibles valores: center, left, right, justify */
text-align: center;

/* Para establecer la decoración de texto */
/* Posibles valores: underline, overline, none, line-through */
text-decoration: underline;

/* Para establecer tabulaciones */
text-indent: 10px;

/* Para transformar un texto todo a mayúscula o minúsculas */
/* Posibles valores: uppercase, lowercase, capitalize */
text-transform: uppercase;
```

FUNCTION CALC()

Con esta función se pueden especificar valores basándose en la aritmética, en lugar de utilizar números fijos.

Ejemplo	Nombre	Resultado
25% + 5px	Suma	Suma el 25% y 5px
2.5em - 5px	Resta	Diferencia entre 2.5em y 5px
25% * 2	Multiplicación	Producto del 25% y 2
25% / 2	División	División del 25% y 2

La función calc() de CSS3 nos permite realizar operaciones matemáticas sencillas como: sumar (+), restar (-), multiplicar (*) o dividir (/), y puede ser utilizada siempre que se trate de valores numéricos como: longitud, frecuencia, duración, ángulo o número.

```
section {  
    height: calc(100% - 2 * 10px);  
}
```

La función calc() hace posible que podemos sumar, restar, multiplicar o dividir:

- píxeles (px) con em
- porcentajes (%) con píxeles (px)
- vw y hw con píxeles (px)

Ya que una de las habilidades de calc() es mezclar unidades y otras funciones css.

```
.ejemplo {  
    padding: calc(1em + min(2em, 10px));  
    height: calc(max(100vh - 20px, 400px));  
}  
@media (min-width: 768px) {  
    .bloque  
    {  
        font-size: calc(16px + 1vw);  
    }  
}
```

ESTILOS PARA LAS LISTAS

Las propiedades CSS de las listas son las que nos permiten controlar los estilos de los marcadores y la posición de los elementos dentro de las listas.

Propiedad	Descripción	Valores
list-style-type	Estilo aplicable a los marcadores visuales de las listas	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none
list-style-image	Imagen aplicable a los elementos de las listas	url(«...») none
list-style-position	Posición del marcador dentro de la lista	inside outside
list-style	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type list-style-position list-style-image

```
.a {  
    list-style-image: url('/img/lapizP.png');  
    list-style-position: outside;  
}  
.b {  
    list-style-type: square;  
    list-style-position: outside;  
}  
.c {  
    list-style-type: upper-roman;  
    list-style-position: inside;  
}  
.d {  
    list-style: lower-alpha, inside;  
}
```

PROPIEDADES DE COLOR

Los primeros cambios de estilo que hacemos cuando aprendemos CSS es hacer variaciones en los colores de primer plano y de fondo de un documento HTML o de cualquiera de sus elementos o partes. Sin embargo, indicar el color específico no es una tarea fácil. Hay múltiples formas de definir un color en CSS, algunas más sencillas, otras más complejas.

Propiedades CSS que podemos utilizar para cambiar el color de texto y el color de fondo de un elemento HTML:

Propiedad	Valor	Significado
color	COLOR	Cambia el color del texto que está en el interior de un elemento.
background-color	COLOR	Cambia el color de fondo de un elemento.

FORMAS DE INDICAR UN COLOR

Esquema	Nombre	Descripción
red	Palabra clave predefinida	Establece un color mediante una palabra clave predefinida.
rgb()	Función RGB	Utiliza una función rgb() (rojo, verde, azul).
rgba()	Función RGB con canal alfa	Función rgb() o rgba() con un canal alfa (transparencia) añadido.
#rrggbbaa	Código RGB hexadecimal	Notación RGB abreviada en hexadecimal .
#rgb	Código RGB hexadecimal con canal alfa	Notación RGB abreviada en hexadecimal con un canal alfa añadido.
hsl()	Función HSL	Función hsl() (matiz de color, saturación y brillo)
hsla()	Función HSL con canal alfa	Función hsl() o hsla() con un canal alfa añadido.

FUNCIÓN RGB

RGB es una sigla formada por los términos de la lengua inglesa red, green y blue. El concepto se emplea para referirse al modelo cromático que consiste en representar distintos colores a partir de la mezcla de estos tres colores primarios.

El modelo RGB se basa en lo que se conoce como **síntesis aditiva de color**. Empleando la luminosidad del rojo, el verde y el azul en diferentes proporciones, se produce el resto de los colores.

PARA indicar en CSS un color utilizando la función RGB UTILIZAREMOS la función `rgb()`, escogiendo entre una de las siguientes variaciones:

Función RGB	Descripción
<code>rgb(r, g, b)</code>	Notación clásica: Valores numéricos o porcentajes separados por coma.
<code>rgb(r, g, b, a)</code>	Se añade al anterior un valor correspondiente al canal alfa , separado por coma.
<code>rgb(r g b)</code>	Notación moderna: Valores numéricos o porcentajes separados por espacio.
<code>rgb(r g b / a)</code>	Se añade al anterior un valor correspondiente al canal alfa, separado por /

Color RGB	Hexadecimal (RGB Abreviado)	Hex. abreviado	Palabra clave
<code>rgb(255, 0, 0)</code>	<code>#FF0000</code>	<code>#F00</code>	<code>red</code>
<code>rgb(0, 0, 0)</code>	<code>#000000</code>	<code>#000</code>	<code>black</code>
<code>rgb(0, 255, 255)</code>	<code>#00FFFF</code>	<code>#0FF</code>	<code>cyan</code>
<code>rgb(147, 112, 219)</code>	<code>#9370DB</code>	<code>#97D</code>	<code>mediumpurple</code>

EN cada uno de los casos anteriores, se deben indicar **los valores r, g y b**. Mezclando las cantidades de cada canal, se puede obtener prácticamente cualquier color.

Para especificar la cantidad de color de cada canal se puede hacer de dos formas:

- Como **números, desde el 0 al 255**
- Como **porcentajes, desde el 0% al 100%**

```
.colores
/* Notación clásica */
background-color: rgb(125, 80, 10);
background-color: rgb(55%, 25%, 75%);
/* Notación moderna */
background-color: rgb(125 80 10);
background-color: rgb(55% 25% 75%);
}
```

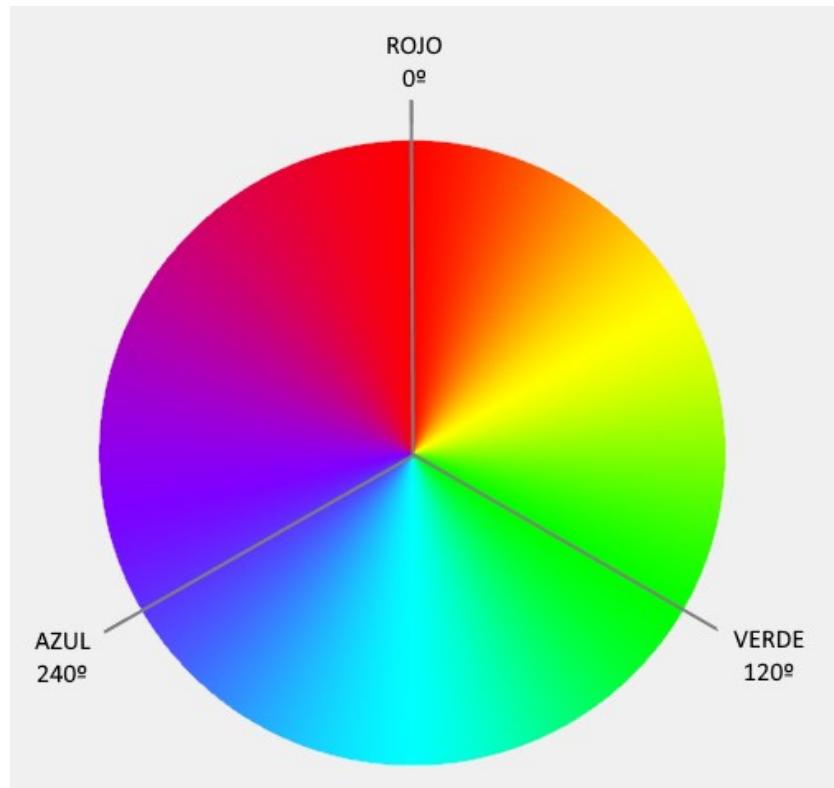
FORMATO HSL

Otra función para indicar colores en CSS es la función **hsl()**, ya que sus parámetros suelen ser mucho más intuitivos para la mayoría de los desarrolladores que otros como hexadecimal o RGB.

Función HSL	Descripción
hsl(h, s, l)	Notación clásica: Número de grados separados por coma.
hsl(h, s, l, a)	Se añade al anterior un valor correspondiente al canal alfa, separado por coma.
hsl(h s l)	Notación moderna: Número de grados separados por espacio.
hsl(h s l / a)	Se añade al anterior un valor correspondiente al canal alfa, separado por /

Las siglas HSL significan matiz de color (hue), saturación y luminosidad (brillo):

- El matiz de color, se trata de un valor ANGULAR 0 a 360.
- El porcentaje de saturación, un valor de 0% a 100%.
- El porcentaje de brillo del color, un valor de 0% a 100%.



CANALES ALFA

Sirven para indicar que un color que tenga cierto grado de transparencia, y de esta forma, refleje el contenido, color o imágenes que se encuentren detrás parcial o completamente.

Existe la posibilidad de utilizar los denominados canales alfa, que permiten establecer un porcentaje de transparencia parcial sobre un color. Estos se pueden establecer en los diferentes formatos, salvo en los colores con palabras clave.

Método	Descripción
<code>rgb(r, g, b, a)</code>	Notación clásica: Se añade un último parámetro con el porcentaje del canal alfa.
<code>rgb(r g b / a)</code>	Notación moderna: Se añade el porcentaje del canal alfa separado por un <code>/</code> .
<code>hsl(r, g, b, a)</code>	Notación clásica: Se añade un último parámetro con el porcentaje del canal alfa.
<code>hsl(r g b / a)</code>	Notación moderna: Se añade el porcentaje del canal alfa separado por un <code>/</code> .
<code>#rrggbbaa</code>	Se añade el porcentaje del canal alfa en hexadecimal como un nuevo par <code>aa</code> .
<code>#rgba</code>	Notación abreviada: igual al anterior, pero sin repetir cada par.

```
.canales-alfa {  
    /* Notación clásica */  
    background-color: rgb(0, 0, 0, 0.5);  
    /* Notación moderna */  
    background-color: rgb(0 0 0 / 50%);  
    /* Notación clásica */  
    background-color: hsl(180, 50%, 25%, 0.75);  
    /* Notación moderna */  
    background-color: hsl(180 50% 25% / 75%);  
    /* Notación abreviada */  
    background-color: #aa44bb77;  
    background-color: #a4b7;  
}
```

HERRAMIENTAS PARA SELECCIONAR COLORES

<https://hslpicker.com/#ffd900>

<https://color.adobe.com/es/create/color-wheel>

PROPIEDAD BACKGROUND-IMAGE

CSS proporciona la propiedad background-image, con la cual se pueden indicar imágenes de fondo o, incluso gradientes o degradados de varios colores. Para poner una imagen de fondo en cualquier contenedor (también en el body) utilizaremos la propiedad **background-image** y en el **valor de la propiedad**, el nombre de la imagen (o la dirección URL donde está alojada), siempre rodeada de la función `url()`.

Propiedad	Valor	Significado
background-image	none	No utiliza ninguna imagen de fondo.
background-image	<code>url("imagen.jpg")</code>	Usa la imagen indicada como fondo. Las comillas no son obligatorias.
background-image	Gradiente	Utiliza un gradiente de tipo lineal, radial o cónico

Una vez establecida una imagen de fondo con `background-image`, se puede personalizar la forma en la que se mostrará dicha imagen mediante propiedades:

Propiedad	Descripción
background-repeat	Establece el modo en el que se repite la imagen de fondo de ser muy pequeña.
background-attachment	Indica si la imagen de fondo permanece fija o se desplaza al hacer scroll.
background-position	Establece una posición para la imagen de fondo, de modo que podemos moverla de sitio.
background-clip	Modo de relleno de la imagen de fondo.
background-origin	Origen de la imagen de fondo si se utiliza background-clip .
background-size	Establece un tamaño diferente a la imagen de fondo.

La propiedad `background-repeat`

La propiedad `background-repeat` especifica si la imagen se repetirá horizontalmente (`repeat-x`), si se repetirá verticalmente (`repeat-y`), si lo hará en ambas direcciones (`repeat`) o en ninguna (`no-repeat`). Por defecto, si no se indica nada, esta propiedad está ajustada en `repeat`.

Valor	Significado
repeat	Repite la imagen de fondo horizontal y verticalmente.
repeat-x	Repite la imagen de fondo sólo horizontalmente (eje x).
repeat-y	Repite la imagen de fondo sólo verticalmente (eje y).
space	Repite la imagen y rellena con espacio los huecos.
round	Repite la imagen y amplia cada repetición para ajustar.
no-repeat	La imagen de fondo no se repite.

Los valores, **space** y **round**, que se repite el fondo. En el caso de que tengamos una imagen de fondo que se repita varias veces en mosaico, **space** evita que se corte la imagen, introduciendo un espacio entre las repeticiones individuales.

Sin embargo, **round** lo que hace es ajustar la imagen individual, de modo que la expande o contrae para ajustarla al espacio disponible. En ambos casos la repetición de los fondos nunca se mostrará cortada.

La propiedad background-attachment

La propiedad background-attachment especificará si la imagen de fondo seguirá el desplazamiento del usuario al hacer scroll por la página, es decir, si el usuario al hacer scroll y bajar para ver el contenido de la página, la imagen de fondo de desplazará hacia arriba siguiendo el flujo normal de una página. Este comportamiento se consigue con la opción scroll, que es la que viene establecida por defecto.

Si indicamos el valor fixed, la imagen de fondo se quedará fijada y no se moverá mientras el usuario se desplaza por la página, algo que puede ser útil en muchos escenarios.

background-attachment: fixed hace que una imagen de fondo se muestre fija al desplazar la ventana del navegador.

Valor	Significado
scroll	Cuando hacemos scroll la imagen de fondo se desplaza .
fixed	Cuando hacemos scroll, la imagen de fondo permanece fija .

La propiedad background-position

El punto 0,0 de la imagen es la esquina superior izquierda

La propiedad background-position permite desplazar la imagen en la zona especificada por X y por Y. Por defecto, esos valores son 0% 0%, y pueden especificarse tanto con unidades (porcentajes, pixels, etc.) como mediante palabras clave que representan zonas predefinidas (top, left, right, bottom y center).

Si sólo se especifica un valor, se tomará el desplazamiento para el eje x, mientras que el valor del eje Y será automáticamente establecido a center, que en porcentaje es 50%

Las palabras clave permitidas son equivalentes a algunos porcentajes significativos: top = 0%, left = 0%, center = 50%, bottom = 100%, right = 100%.

CSS permite mezclar porcentajes y palabras clave, como, por ejemplo:

```
background-position: 50% 2cm;  
background-position: center 2cm;  
background-position: center 10%;
```

Valor	Significado
posX	1 parámetro. Desplaza la imagen de fondo al punto (x, 50%). Se toma solo el desplazamiento en el eje X, para el eje Y se la asigna el 50% que es lo mismo que center sobre el eje vertical.
posX posY	2 parámetros. Desplaza la imagen de fondo al punto (x, y).

La propiedad background-size

Una propiedad muy interesante es background-size, la cual nos permite dar un tamaño a la imagen de fondo. Podemos ajustar tanto el tamaño de ancho como el de alto, e incluso tenemos algunas palabras clave predefinidas para obtener un resultado específico.

Por defecto, una imagen de fondo toma automáticamente el tamaño de la imagen (que podría ser demasiado grande). Para no tener que modificar la imagen original de forma manual, podemos utilizar esta propiedad y ajustarla mediante CSS:

Valor	Significado
size	1 parámetro. Aplica un de (<u>ancho</u> × <u>auto</u>) a la imagen de fondo. Mantiene la proporción.
size size	2 parámetros. Aplica un de (<u>ancho</u> × <u>alto</u>) a la imagen de fondo. Hay que vigilar la proporción.

```
background-image: url(..../img/transparentes/chocolate.png);  
background-size: 300px, 300px;  
background-repeat: no-repeat;
```

Con **background-size** se pueden utilizar los siguientes valores. Los valores **cover** y **contain**, sólo pueden indicarse en el caso de que se especifique un sólo parámetro como valor en la propiedad `background-size`.

Valor	Significado
auto	No escala la imagen. Utiliza el tamaño original. Es el valor por defecto.
size <i>unidad</i>	Indicamos el tamaño específico que queremos usar (<i>píxeles o porcentaje</i>).
cover	Escala el ancho de la imagen de fondo al ancho del elemento.
contain	Escala el alto de la imagen de fondo al alto del elemento.

La propiedad **background-clip** y propiedad **background-origin**

En CSS existen unas propiedades para indicar como cubrirá la imagen de fondo al elemento seleccionado para darle estilo.

La propiedad **background-clip** establece la forma en la que el color o la imagen de fondo cubrirá el elemento, mientras que la propiedad **background-origin** intenta posicionar el comienzo de la imagen de fondo, útil con imágenes no con fondos de color.

Propiedad	Valor	Significado
background-clip	border-box padding-box content-box	Modo de relleno de la imagen de fondo
background-origin	border-box padding-box content-box	Origen del modo de relleno del fondo

Valor	Significado
padding-box	La imagen o color de fondo cubrirá la zona del espaciado y contenido.
border-box	La imagen o color de fondo cubrirá la zona del borde, espaciado y contenido.
content-box	La imagen o color de fondo cubrirá sólo la zona del contenido.

Una buena forma de darse cuenta del funcionamiento de estas propiedades es establecer un borde grueso punteado. Usando **border-box** la imagen de fondo se extenderá en todo el elemento, incluyendo borde, espaciado y contenido. El valor **padding-box** extenderá la imagen de fondo sólo mediante el padding y el contenido y, por último, la propiedad **content-box** extenderá la imagen de fondo sólo en la zona del contenido.

Fondos múltiples

Propiedad	Descripción
background-image: url(imagen1), url(imagen2), ...	Establece varias imágenes de fondo en un elemento.

CSS permite establecer múltiples fondos separando por comas. Hay que tener en cuenta que la primera imagen establecida es la que permanecerá al frente, la siguiente imagen añadida aparecerá detrás de la primera y así con todas las imágenes que se vayan indicando. Si colocamos una imagen en formato .png o .webp, las cuales soportan transparencias nos permite realizar diferentes combinaciones

Al tener múltiples imágenes de fondo en la propiedad background-image, las propiedades derivadas de **background-*** pasan a ser múltiples también, ya que cada valor se aplica en orden a su respectiva imagen de fondo. En el caso de dejar sólo un valor, se aplicará a todas las imágenes de fondo:

```
.contenedor3 {
    background-image: url(..../img/galleta.png), url(..../img /leche.png);
    background-repeat: no-repeat, no-repeat;
    background-size: 30%,30%,50%,50%;
}
```

Atajo: background

Es posible establecer todas estas propiedades anteriores en una sola regla de CSS a modo de atajo, y así ahorrar mucho espacio en escribir las propiedades anteriores por separado. Si alguno de los valores no necesitamos indicarlo, simplemente lo omitimos.

El atajo se construye con la propiedad background y sigue la siguiente estructura:

Atajo	Orden
background	Color Image Position /size Repeat Attachment Origin Clip

El parámetro **/size** es opcional. De indicar el carácter **/** la propiedad background esperará el valor background-size a continuación.

```
.contenedor{  
    /* Atajo simple */  
    background: #fff url(imagen.jpg) top center repeat-x;  
  
    /* Atajo completo */  
    background: rgb(173, 255, 20) url(imagen.jpg) 0 0 / 150px space scroll  
padding-box padding-box;  
  
    /* Atajo múltiple */  
    background:  
        url(primer-plano.jpg) center center / cover,  
        rgb(20, 255, 98) url(imagen.jpg) 0 0 / 150px space scroll padding-box  
padding-box;  
}
```

GRADIENTES O DEGRADADOS

La propiedad background-image además de imágenes mediante la función url(), posee un mecanismo interesantísimo que permite establecer gradientes o degradados a partir de código.

Existen 3 funciones de gradientes: linear-gradient(), radial-gradient() y conic-gradient():

Función de gradiente	Significado
linear-gradient()	Define un gradiente lineal : en una dirección específica.
radial-gradient()	Define un gradiente radial : en forma de círculo o elipse.
conic-gradient()	Define un gradiente cónico : un cono visto superiormente.

<https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/linear-gradient>

<https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/radial-gradient>

<https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/conic-gradient>



```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="css/gradientes.css">
    <title>Gradientes</title>
</head>
<body>
    <h1>Degradados con CSS</h1>
    <h2>Degradados con Lineal</h2>
    <div class="degradado-css-lineal"></div>
    <h2>Degradados con Radial</h2>
    <div class="degradado-css-radial"></div>
    <h2>Degradados con Cónico</h2>
    <div class="degradado-css-conico"></div>
    <h2>Degradados con Imagen</h2>
    <div class="degradado-css-radial-imagen"></div>
</body>
</html>
```

```
h1 {
    text-align: center;
}
.degradado-css-lineal {
    background-image: linear-gradient(to right, #44ee50, #011604);
    height: 400px;
    margin-bottom: 20px;
}
.degradado-css-radial {
    background-image: radial-gradient(circle, #44ee50, #011604);
    height: 400px;
    margin-bottom: 20px;
}
.degradado-css-conico {
    background-image: conic-gradient(#64ff6f 90deg, #011604);
    height: 500px;
}
.degradado-css-radial-imagen{
    background-image: url(../img/transparentes/uvas.png), radial-gradient(circle, #44ee50, #011604);
    height: 600px;
    margin-bottom: 20px;
}
```

PROPIEDAD OPACITY

Para indicar la opacidad de un color o de una imagen podemos indicar tanto valores numéricos entre 0 (completamente transparente) y 1 (completamente visible), pasando por valores como 0.25 o 0.75. Si lo preferimos, podemos usar valores porcentuales entre 0% (completamente transparente) y 100% (completamente visible)

La propiedad **opacity** afecta al elemento indicado y a todos sus hijos. Se hereda.

Propiedad	Valor	Significado
opacity	número	Establece una transparencia con un valor del 0 al 1 (permite decimales).
opacity	porcentaje	Establece una transparencia con un valor del 0% al 100% .

```
.color-imagen-opacidad {  
    background-image:  
        url(../img/transparentes/uvas.png);  
    background-color: #ff0000;  
    color: #ffffff;  
    opacity: 0.5;  
    /* Equivalente a la anterior */  
    opacity: 50%;  
}  
img.opaca {  
    opacity: 40%;  
}
```

PROPIEDADES OVERFLOW Y SCROLL

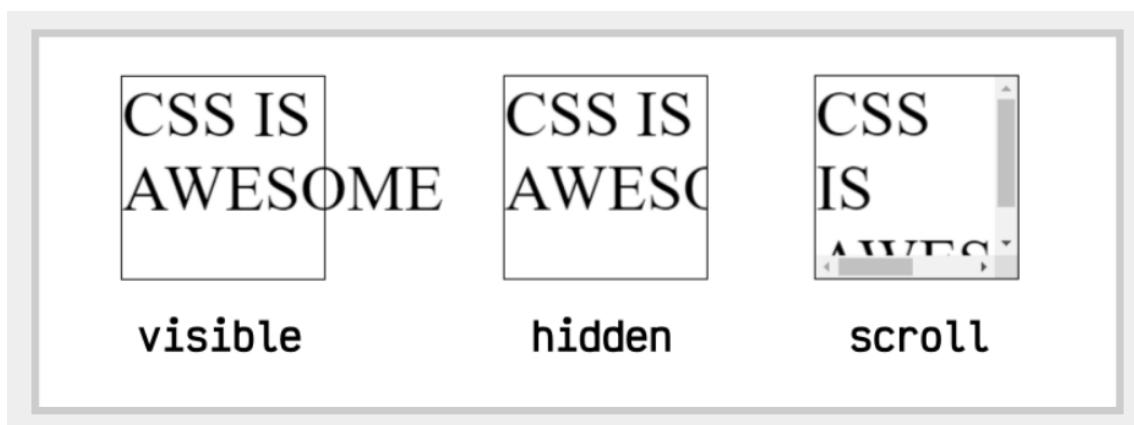
Mediante la propiedad **overflow** y sus derivadas **overflow-x** y **overflow-y**, podemos gestionar el desbordamiento de contenido en un elemento.

Por defecto, el navegador desborda el contenido del elemento, sin embargo, nosotros podemos decidir qué hacer.

La propiedad CSS **overflow** se puede utilizar junto con **scroll** para agregar barras de desplazamiento cuando sea necesario.

Valores de la propiedad overflow

Valor	Descripción
visible	No se controla el desbordamiento, el contenido simplemente se desborda. Valor por defecto.
hidden	Oculta el contenido desbordado y las barras de desplazamiento del elemento.
clip	Igual que hidden , pero no permite desplazamiento mediante programación.
scroll	Oculta el contenido desbordado y muestra barras de desplazamiento.
auto	El navegador decide. Generalmente, aplica scroll .



```
/* Diferentes valores de overflow */
/* Sin recortar, muestra el contenido desbordado */
overflow: visible;

/* Recorta y oculta el contenido desbordado */
overflow: hidden;

/* Muestra barras de desplazamiento siempre, incluso si no desborda */
overflow: scroll;

/* Muestra barras de desplazamiento, solo si el contenido desborda */
overflow: auto;
```

Propiedades básicas que tenemos en CSS para modificar el scroll de ratón:

Propiedad	Valores	Descripción
<code>scroll-behavior</code>	<code>auto</code> <code>smooth</code>	Aplicado sobre <code><html></code> , suaviza ciertos desplazamientos de scroll.
<code>overflow-x</code>	<code>visible</code> <code>hidden</code> <code>clip</code> <code>scroll</code> <code>auto</code>	Muestra/oculta la barra de desplazamiento horizontal del elemento.
<code>overflow-y</code>	<code>visible</code> <code>hidden</code> <code>clip</code> <code>scroll</code> <code>auto</code>	Muestra/oculta la barra de desplazamiento vertical del elemento.
<code>overflow</code>	<code>overflow-x</code> <code>overflow-y</code>	Propiedad de atajo para las dos anteriores.
<code>overflow</code>	<code>overflow</code>	Propiedad de atajo que aplica el valor <code>overflow</code> tanto a x como a y .

Valores de los atributos de overflow:

- `auto` depende de la hoja de estilos del navegador (agente de usuario). Firefox provee barras de desplazamiento si hay contenido excedente.
- `visible` (El contenido no es recortado, podría ser dibujado fuera de la caja contenedora), indicamos que todo el contenido sea visible, quepa o no en la caja contenedora.
- `scroll` : El contenido es recortado si no cabe en el contenedor y el navegador usa las barras de desplazamiento, se haya recortado contenido o no. Esto previene cualquier problema con las barras de desplazamiento apareciendo o desapareciendo en un entorno dinámico.

```
.contenedor {  
    /* Ancho mayor que el contenedor para desbordar horizontalmente */  
    width: 350px;  
    /* Altura mayor que el contenedor para desbordar verticalmente */  
    height: 200px;  
    /* Mostrar barras de desplazamiento horizontal solo si es necesario */  
    overflow-x: auto;  
    /* Mostrar siempre barras de desplazamiento vertical */  
    overflow-y: scroll;  
    /* Desplazamiento suave */  
    scroll-behavior: smooth;  
}
```

Podemos establecerlo para toda la página:

```
html {  
    scroll-behavior: smooth;  
}
```

PROPIEDAD OVERFLOW-WRAP(CSS3) O WORD-WRAP

La propiedad **word-wrap** se utiliza para controlar cómo se deben manejar las palabras largas que no caben en un contenedor. Su objetivo es evitar que las palabras desborden fuera del contenedor y afecten el diseño de la página.

Establece si está permitido **partir palabras**. Esta propiedad se aplica a los elementos en línea, estableciendo **si el navegador debe insertar saltos de línea dentro de una cadena que**, de otro modo, no se rompería, para evitar que el texto desborde su caja de línea.

La propiedad word-wrap tiene dos posibles valores:

- **normal**: Este es el valor por defecto. Las palabras largas no se dividirán y podrían desbordar del contenedor si no hay suficiente espacio para ellas.
- **break-word**: las palabras largas se dividirán en la medida de lo posible para ajustarse dentro del contenedor sin desbordar. Si una palabra es más larga que el ancho disponible en el contenedor, se dividirá en varias líneas para que todas las letras quepan dentro del contenedor.

Importante, a partir de la especificación CSS3, la propiedad **word-wrap ha sido reemplazada por la propiedad overflow-wrap**.

PROPIEDAD HYPHENS

Para elementos en bloque especifica cómo deben dividirse las palabras cuando el texto se ajusta a través de múltiples líneas.

Hyphens=guiones

Esta propiedad no es soportada por todos los navegadores.

Puede impedir la separación de sílabas por completo, usar guiones manualmente en puntos específicos del texto o dejar que el navegador inserte los guiones automáticamente donde corresponda según el idioma.

Las **reglas de separación** silábica son específicas del idioma. En HTML, el idioma es determinado por el atributo **lang** y los navegadores separarán únicamente si este atributo está presente y si existe un diccionario de separación silábica adecuado.

Valores:

- **none**: (valor por defecto) No se permiten divisiones de palabras con guiones.
- **manual**: Las divisiones de palabras con guiones solo se producirán en los puntos definidos mediante el uso de la etiqueta <wbr> en el código HTML.
- **auto**: Permite divisiones de palabras con guiones automáticos según las reglas del idioma.

```
.division_palabras {  
    /* Si olvidas establecer lang, no funcionará hyphens*/  
    width: 150px;  
    padding: 5px;  
    border: 1px blue solid;  
  
    -webkit-hyphens: auto;  
    -ms-hyphens: auto;  
    hyphens: auto;  
  
    /* Para navegadores más antiguos que no soporten hyphens */  
    overflow-wrap: break-word;  
}
```

supercalifragilisticoexpialidoso.
Electroencefalografista.
Lorem ipsum dolor
sit amet
consectetur
adipiscing elit.
Tenetur
consequuntur
temporibus
possimus laborum
reprehenderit
suscipit cum odit
repellendus fugit,
soluta eius
laboriosam a
exercitationem qui
illum asperiores
eveniet pariatur
dolores.

supercalifragilistico
expialidoso.
Electroencefalograf
ista. Lorem ipsum
dolor sit amet
consectetur
adipiscing elit.
Tenetur
consequuntur
temporibus
possimus laborum
reprehenderit
suscipit cum odit
repellendus fugit,
soluta eius
laboriosam a
exercitationem qui
illum asperiores
eveniet pariatur
dolores.

supercalifragili-
coexpialidoso.
Electroencefalo-
grafista. Lorem ip-
sum dolor sit amet
consectetur adipisi-
cing elit. Tenetur
consequuntur tem-
poribus possimus
laborum reprehen-
derit suscipit cum
odit repellendus fu-
git, soluta eius la-
boriosam a exerci-
tationem qui illum
asperiores eveniet
pariatur dolores.

overflow-wrap: break-word;

hyphens: auto;

PREFIJOS CSS EN LOS NAVEGADORES

Actualmente los navegadores tienen implementadas muchas de las nuevas características de CSS3 utilizando sus propias versiones de cada propiedad mediante prefijos.

Esto se hace así para evitar los posibles errores ocasionados por las primeras implementaciones que aún no son estables. Por ello, los navegadores proporcionan valores utilizando sus prefijos propios y una declaración sin prefijo.

Después de un tiempo, cuando las especificaciones son estables, se eliminarán las propiedades con prefijo. Los prefijos para los navegadores más comunes son los siguientes:

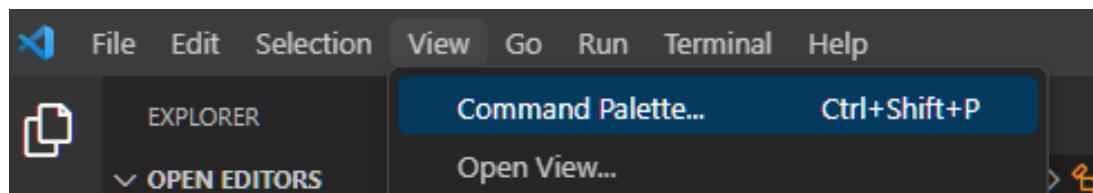
Prefijo	Navegador
-moz-	Firefox
-webkit-	Safari y Chrome
-o-	Opera
-ms-	Internet Explorer

EXTENSIÓN AUTOPREFIXER PARA VISUAL STUDIO CODE

Para ahorrar tiempo y facilitarnos la tarea de incluir los prefijos de las propiedades CSS que todavía no son estables podemos hacer uso de la extensión «Autoprefixer» en Visual Studio Code.

Una vez instalada la extensión, en **view /command palete** (o **Ctrl+shift+p**) escribimos «autoprefixer run» para añadir todos los prefijos necesarios en una hoja de estilos.

Ejemplo, añadiendo una propiedad no estable (**user-select: none;**) en una hoja de estilo CSS y viendo cómo se insertan sus correspondientes prefijos:



```
body
{
/* controla si el usuario puede seleccionar el texto */
user-select: none;
}
```

Ejecutamos **autoprefixer run** en Visual Studio Code y obtenemos:

```
body
{
/* controla si el usuario puede seleccionar el texto */
-webkit-user-select: none;
-moz-user-select: none;
-ms-user-select: none;
user-select: none;
}
```

HERRAMIENTA COMPROBAR COMPATIBILIDAD CSS EN LOS NAVEGADORES

Se puede conocer qué navegadores soportan una determinada propiedad CSS o un elemento HTML5 en la página:

<https://caniuse.com/>

REGLAS CSS «AT RULES»

En CSS existen las denominadas «at-rules» (reglas precedidas del carácter @), dependiendo de la regla en cuestión, puede tener una sintaxis u otra:

Una regla-at es una declaración CSS que comienza con el símbolo arroba, @, seguido por un identificador, e incluye todo el contenido hasta el siguiente punto y coma, o el siguiente bloque CSS, lo primero que encuentre.

Regla	Descripción
@import	Indica al motor de CSS que incluya una hoja de estilos externa
@media	Aplica los estilos si cumple las condiciones indicadas. Usado para responsive. Media Queries
@font-face	Indica una tipografía externa que el navegador debe descargar.
@keyframes	Describe el aspecto de los pasos intermedios en una secuencia de animación CSS

PROPIEDAD POSITION

En CSS, el flujo natural de los elementos se ve influenciado por la propiedad **display**, que determina cómo se presentará un elemento en el diseño de la página.

- **block (bloque):**

Los elementos de tipo bloque se colocan uno debajo del otro en el flujo vertical. Ocupan todo el ancho disponible y se extienden hasta el final del contenedor padre.

- **inline (en línea):**

Los elementos de tipo en línea aparecen en la misma línea, uno al lado del otro en el flujo horizontal. Solo ocupan el ancho necesario para mostrar su contenido, sin forzar un salto de línea.

- **inline-block (en línea-bloque):**

Combina características de elementos en línea y en bloque. Permite que los elementos se coloquen en la misma línea, pero se pueden aplicar propiedades de tamaño y margen horizontal como si fueran elementos de bloque.

- **none (ninguno):**

Oculta el elemento y lo elimina del flujo normal del documento.

El espacio que ocuparía se elimina, y otros elementos llenan el espacio.

Para un posicionamiento más exacto y concreto CSS nos proporciona la propiedad **position** cuyos valores van íntimamente asociados a las **propiedades CSS top, bottom, left, right y z-index**.

Las 4 primeras permiten desplazamientos respecto a un punto de referencia concreto, la última (z-index), permite trabajar con capas.

Los distintos valores que puede tomar esta propiedad son:

- **static:** Es el valor por defecto. El elemento sigue el flujo que le corresponde. No funcionan top, bottom, left, right o z-index.
- **relative:** Mueve el elemento de su posición dentro del flujo que le corresponde, utilizando top, bottom, left, right o z-index. Ejemplo, si se quiere bajar 20px, habría que indicar top: 20px;
- **absolute:** este tipo de posicionamiento coloca los elementos utilizando como punto de origen el primer contenedor padre con posicionamiento diferente a estático. Se le aplica top, bottom, right o z-index en relación con la primera etiqueta padre que no tenga posición estática.
- **fixed:** Se le aplica top, bottom, right o z-index en relación con documento, es decir, deja fijo el contenido. No atiende al scroll. La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador.
- **sticky:** («pegajoso»). Se comporta como *relative* hasta llegar a una posición de scroll y a partir de entonces *fixed*. Se utiliza para fijar los menús en la parte superior. Necesita sufijos CSS, no es soportado igual por todos los navegadores.

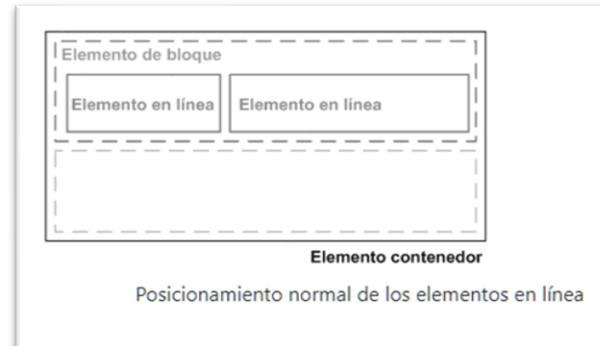
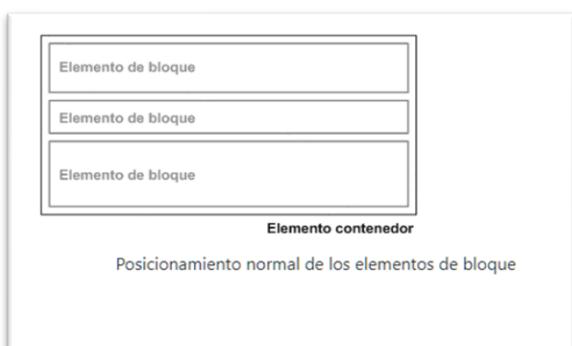
Propiedad	Valor	Significado
top:	auto size	Empuja el elemento una distancia desde la parte superior hacia el inferior.
bottom:	auto size	Empuja el elemento una distancia desde la parte inferior hacia la superior.
left:	auto size	Empuja el elemento una distancia desde la parte izquierda hacia la derecha.
right:	auto size	Empuja el elemento una distancia desde la parte derecha hacia la izquierda.
z-index:	auto número	Coloca un elemento en el eje de profundidad, más cerca o más lejos del usuario.

Debemos tener claras las propiedades top, bottom, left y right, que sirven para mover un elemento desde la orientación que su propio nombre indica hasta su extremo contrario.

Esto es, si utilizamos `left` e indicamos `20px`, estaremos indicando mover desde la izquierda 20 píxeles hacia la derecha.

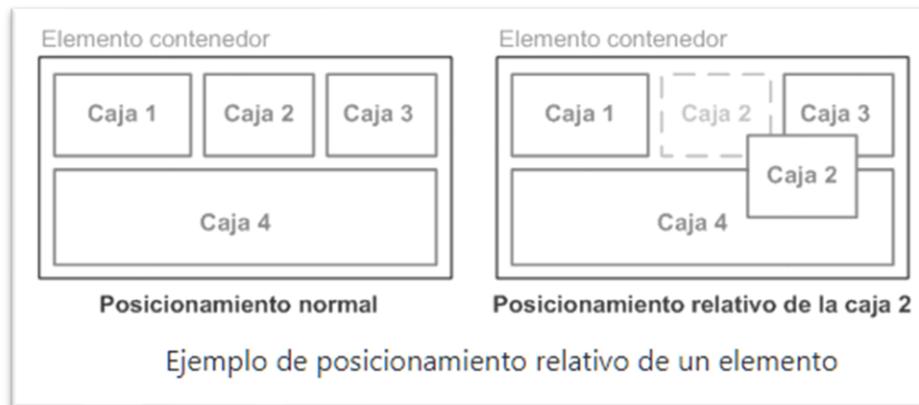
POSITION STATIC

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, sólo se tiene en cuenta si el elemento es de bloque o en inline-block sus propiedades width y height y su contenido.



POSITION RELATIVE

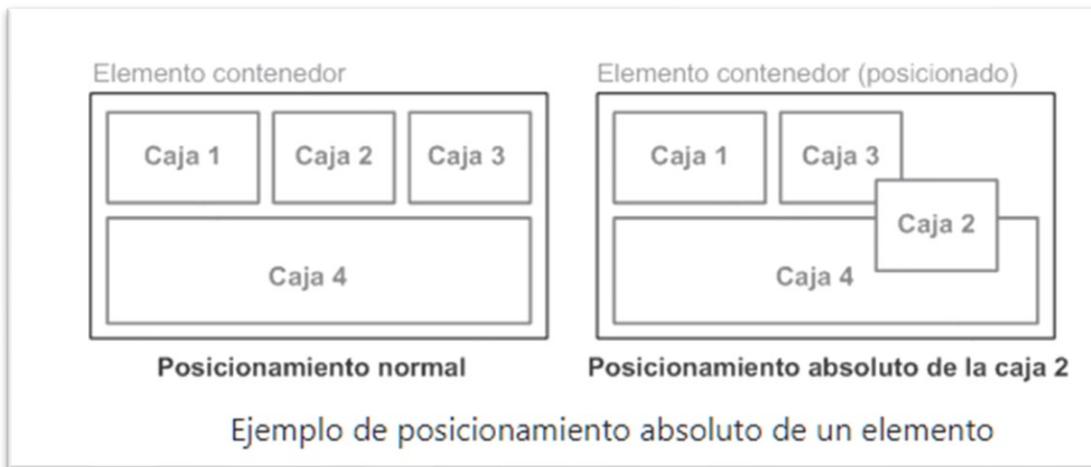
El desplazamiento relativo de una caja no afecta al resto de cajas adyacentes, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.



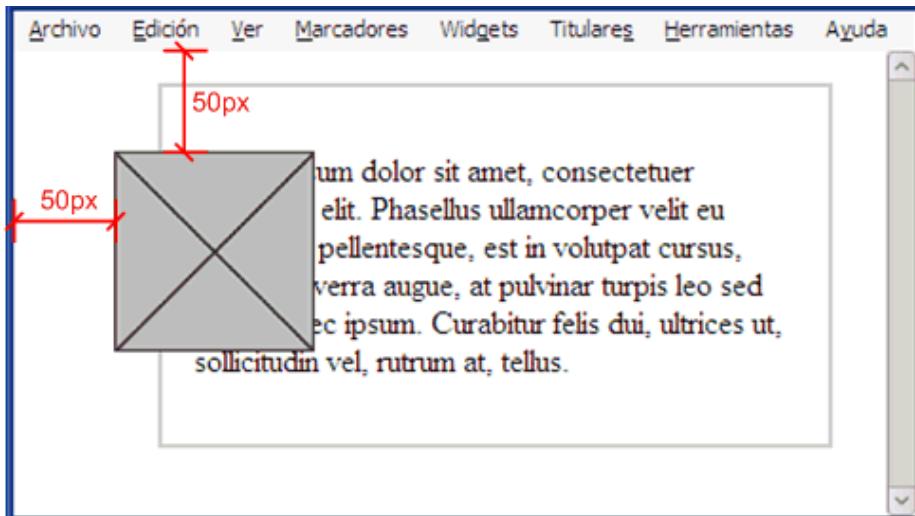
La caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página no modifican su posición, se producen solapamientos entre los contenidos de las cajas. Además, no ocupan el hueco dejado por la caja 2.

POSITION ABSOLUTE

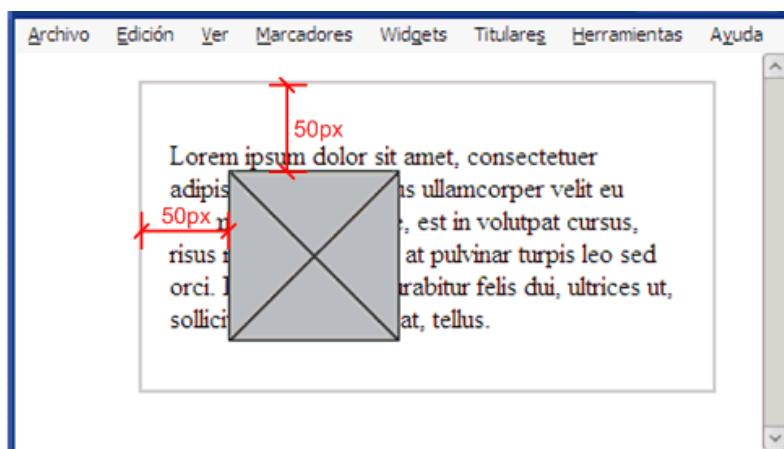
- El primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static` se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento `<body>` de la página.
- Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición
- Las cajas posicionadas de forma absoluta "salen del flujo normal de la página", lo que provoca que el resto de elementos de la página se muevan y en ocasiones, ocupen la posición original en la que se encontraba la caja.
- El posicionamiento absoluto de una caja es probable que se produzca solapamientos con otras cajas.



```
div img {  
    position: absolute;  
    top: 50px;  
    left: 50px;  
}
```



La referencia del posicionamiento absoluto es la ventana del navegador



La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

Si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible añadir al contenedor la propiedad **position: relative**, pero no es obligatorio desplazar el elemento contenedor respecto de su posición original.

POSITION STICKY

El posicionamiento sticky es un híbrido del posicionamiento estático y fixed. Cuando se hace scroll inicialmente parece que tenga un posicionamiento static. Al llegar con el scroll al lugar natural que ocuparía la caja, el posicionamiento pasa a transformarse en fixed.

POSITION FIXED

La caja salta del flujo natural del documento web y pasa a ocupar de forma invariable una determinada posición. Permanece fijo en la pantalla incluso al hacer scroll.

Si es necesario se superpone a otras cajas.

Del mismo modo que en el posicionamiento absoluto obliga al resto de cajas a desplazarse para ocupar la posición que queda vacía.

CENTRADO VERTICAL DE ELEMENTOS DE BLOQUE

Una vez ya sabemos usar la propiedad position podemos centrar verticalmente elementos de bloque. Nos encontraremos con dos casos:

- Cuando conocemos la altura del elemento.
- Cuando desconocemos la altura del elemento.

Si conocemos la altura del elemento y esta es por ejemplo 150px;

```
.contenedor {  
    position: relative;  
}  
.elemento_a_centrar {  
    height: 150px;  
    margin-top: -75px;  
    /* La mitad de la altura */  
    position: absolute;  
    top: 50%;  
}
```

Si desconocemos la altura del elemento:

```
.contenedor {  
    position: relative;  
}  
.elemento_a_centrar {  
    position: absolute;  
    top: 50%;  
    transform: translateY(-50%);  
}
```

ELEMENTOS FLOTANTES

Son aquellos que se le asigna la propiedad `float`. Esta propiedad “rompe” el flujo en bloque y los elementos flotan en línea.

Con la propiedad `float` puedes conseguir que los elementos que quieras, alteren su comportamiento y floten a la izquierda (`left`) o a la derecha (`right`).

Con el valor `none` (valor por defecto) eliminás esta característica de desplazamiento.

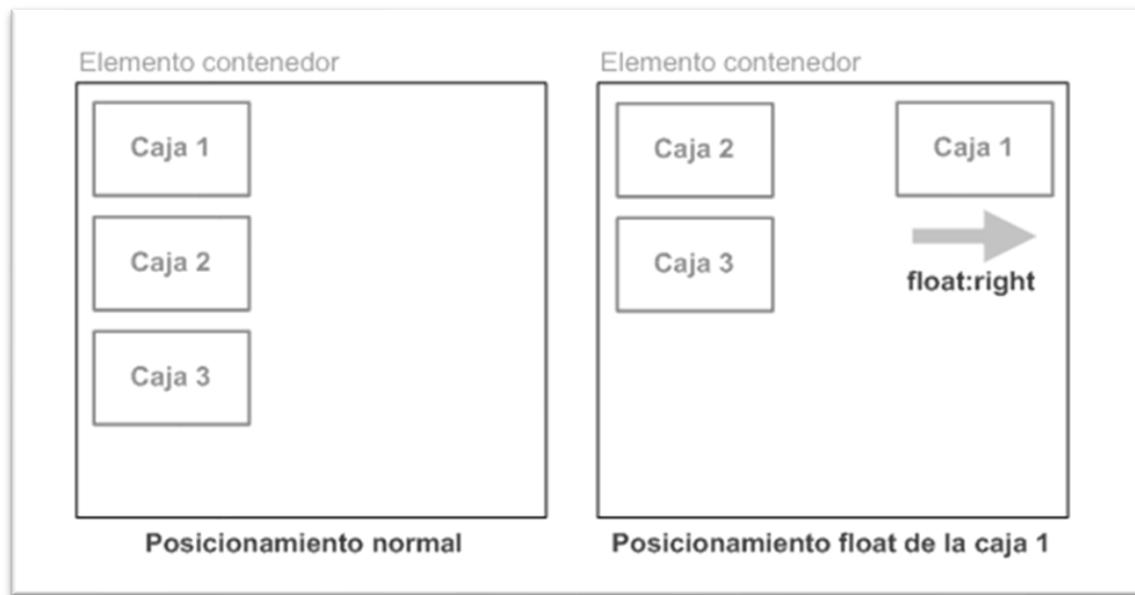
Propiedad	Valor	Significado
<code>float</code>	<code>none</code> <code>left</code> <code>right</code>	Cambia el flujo para que el elemento flote a la izquierda o derecha.
<code>clear</code>	<code>none</code> <code>left</code> <code>right</code> <code>both</code>	Impide que los elementos puedan flotar en la orientación indicada.

Los elementos flotantes no ocupan espacio en el flujo normal del documento, lo que significa que no afectan el tamaño o la posición de otros elementos en el contenedor.

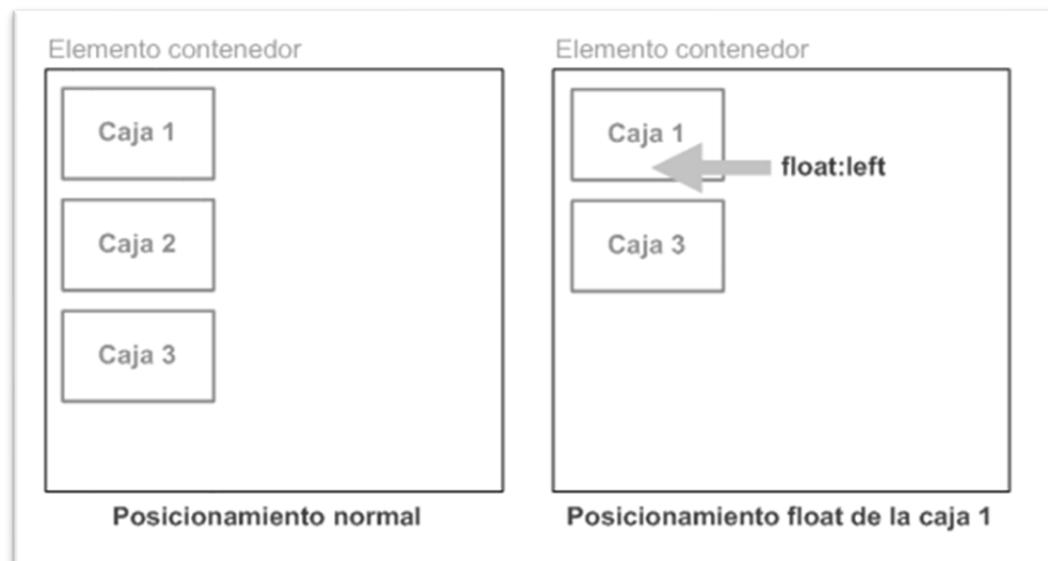
Observad que si están dentro de un contenedor, este no "crece" para ajustarse al tamaño de los elementos flotantes.

PROPIEDADES FLOAT

La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:



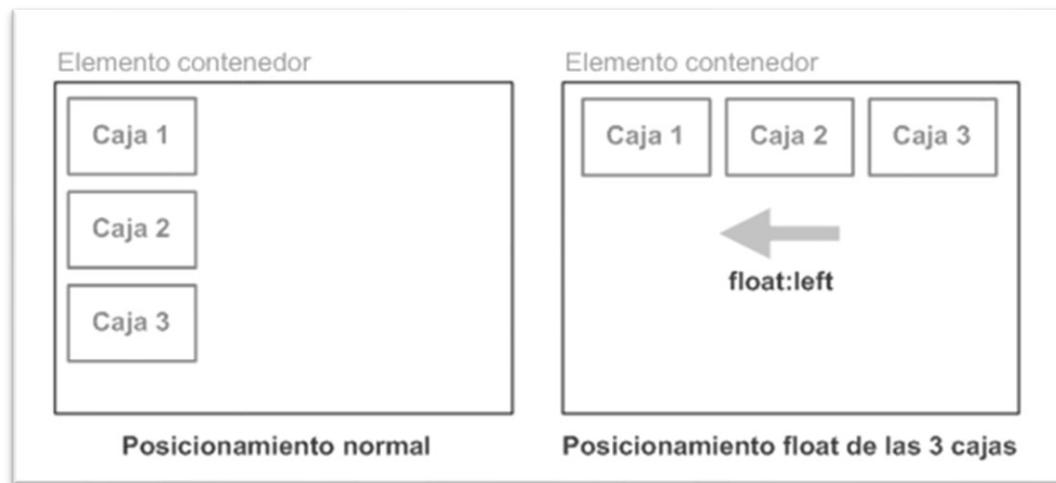
Ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:



La caja 1 es de tipo flotante, por lo que desaparece del flujo normal de la página y el resto de cajas ocupan su lugar. El resultado es que la **caja 2** ahora se muestra dónde estaba la caja 1 y la caja 3 se muestra dónde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. **El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.**

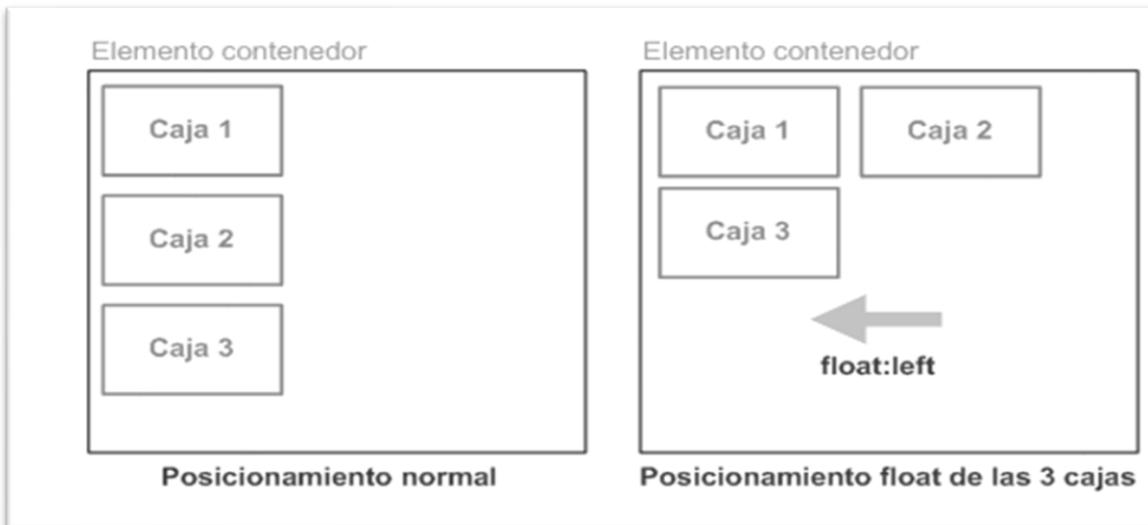
Ejemplo: se posicionan de forma flotante [hacia la izquierda las tres cajas](#)



En el ejemplo anterior, las cajas no se superponen entre sí porque **las cajas flotantes tienen en cuenta las otras cajas flotantes existentes.**

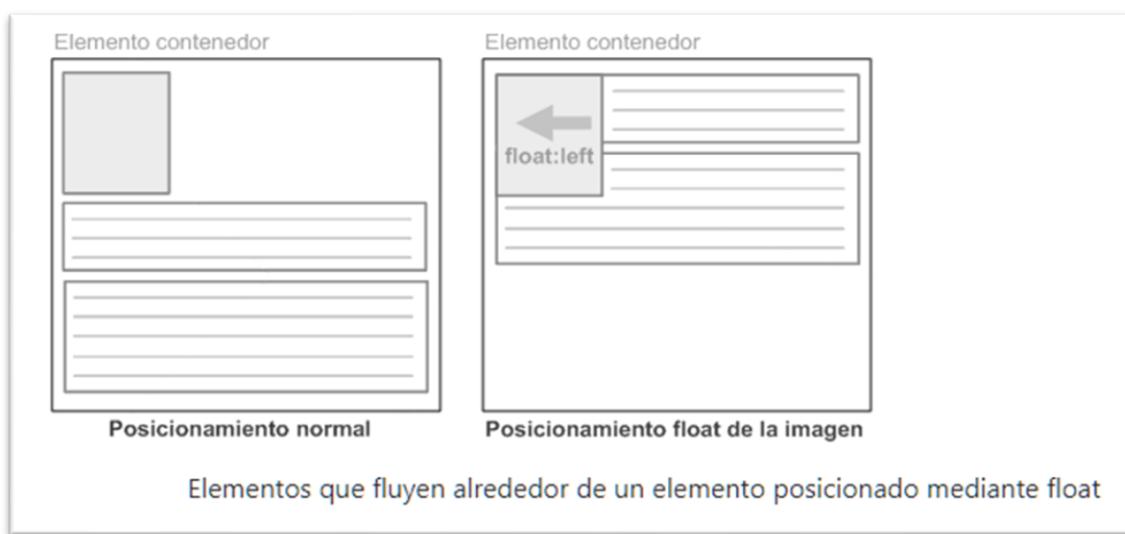
Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

[Si no existe sitio en la línea actual](#), la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:



Posicionamiento flotante cuando **no** hay espacio

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:



La regla CSS que se aplica en la imagen del ejemplo anterior es:

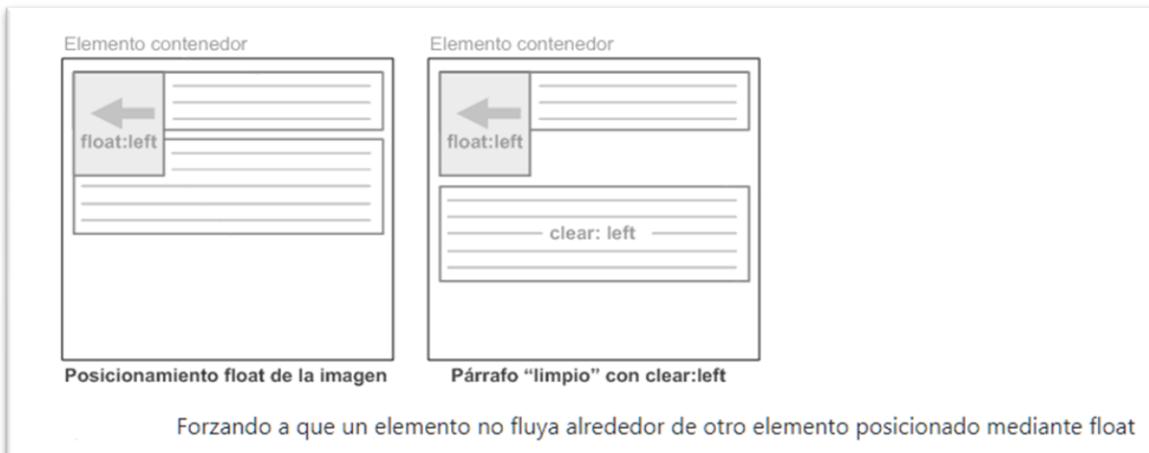
```
img {
    float: left;
}
```

Uno de los principales motivos para la creación del posicionamiento float fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float. De hecho, en muchas ocasiones queremos que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen, entonces utilizaremos la propiedad [clear](#).

PROPIEDAD CLEAR

La propiedad [clear](#) indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante.



Explicación de los valores de la propiedad [clear](#)

El valor [both](#) despeja los lados, izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

El valor [none](#) indica que el elemento no será movido de forma descendente para limpiar elementos flotantes que existan ni a la derecha ni a la izquierda.

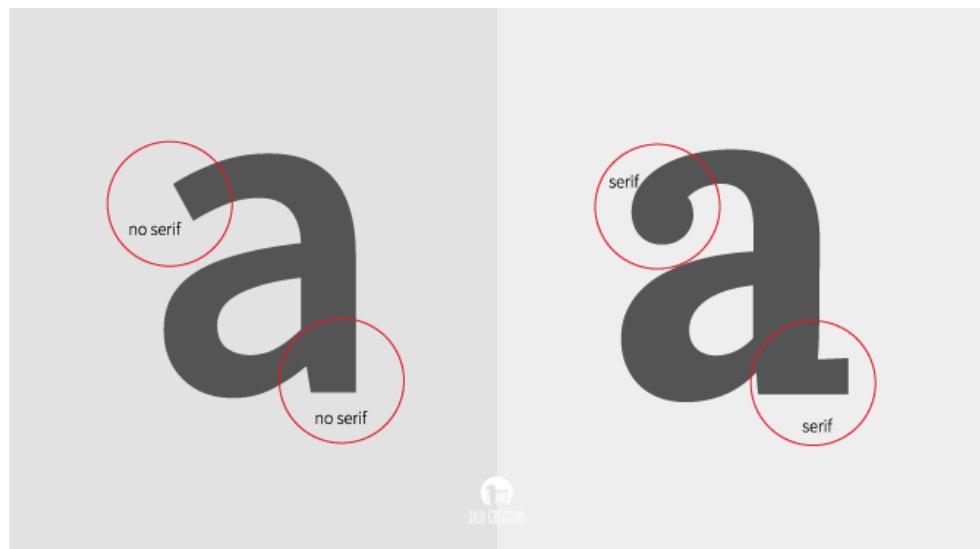
TIPOGRAFÍAS CSS

Las tipografías (también denominadas **fuentes**) son una parte muy importante del mundo de CSS y uno de los pilares del diseño web. La elección de una tipografía adecuada, su tamaño, color, espacio entre letras, interlineado y otras características pueden variar mucho, de forma consciente o inconsciente, la percepción en la que una persona interpreta o accede a los contenidos de una página.

CARACTERÍSTICAS DE UNA TIPOGRAFÍA

1. **Serif o Serifa o Gracia:** Las fuentes o tipografías que utilizan serif o gracia, son aquellas que incorporan unos pequeños adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Serif» (con serifa).
2. **Sans Serif o Paloseco:** Las fuentes o tipografías de paloseco o sans serif son las opuestas a la anterior: unas tipografías lisas, sin adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Sans Serif» (sin serifa).
3. **Monoespaciada:** Por otro lado, existe un tipo de tipografía denominada fuente monoespaciada, que se basa en que cada una de sus letras tienen exactamente el mismo ancho. Son muy útiles para tareas de programación o emuladores de terminal, donde se leen mejor líneas con estas características, ya que no queremos que una línea sea más corta dependiendo de su contenido.





Tradicionalmente, se han utilizado tipografías **con serifa en medios impresos** argumentando que dichos bordes ofrecen una mayor legibilidad que las tipografías de paloseco, ya que ayudan a reconocer más rápidamente las letras. En medios digitales, las **tipografías de paloseco** suelen ser más comunes puesto que dan un aspecto más limpio y ayudan a que se canse menos la vista del usuario.

mono
[] [] [] [] []

fuente no monoespaciada
(ancho de letra diferente)

mono
[] [] [] [] []

fuente monoespaciada
(siempre mismo ancho de letra)

PROPIEDADES DE LAS TIPOGRAFÍAS

Existe un amplio abanico de propiedades CSS para modificar las características básicas de las tipografías a utilizar. Las propiedades CSS más básicas para aplicar a cualquier tipo de tipografía:

Propiedad	Valor	Significado
<code>font-family</code>	Una fuente o una lista de fuentes	Indica el nombre de la fuente (o una lista de ellas).
<code>font-size</code>	<code>xx-small x-small small medium large x-large xx-large smaller larger</code> tamaño unidad	Indica el tamaño de la fuente.
<code>font-style</code>	normal italic (además de inclinarse, la forma de algunas letras cambia) oblique (la forma de la letra no cambia, simplemente se inclina)	Indica el estilo de la fuente.
<code>font-weight</code>	<u>Peso</u> normal bold (son los valores numéricos 400 y 700) lighter bolder (un valor inferior y superior respecto del elemento padre)	Indica el peso (grosor) de la fuente (100-900)

Podemos seleccionar tipografías concretas, especificar su tamaño, estilo o grosor.

`font-family: Georgia, "Times New Roman", serif;`

El navegador buscará primero Georgia. Si está instalada, el navegador mostrará esa fuente. A continuación, buscará Times New Roman. Si tampoco está disponible, recurrirá a mostrar la fuente genérica por defecto de la familia serif.

`font-weight: bold;`

`font-weight: 500;`

Esos grosores no se aprecian de la misma manera en cada tipo de letra.



Normalmente se incrementan los pesos de 100 en 100.

Problemas con las tipografías

1. Tipografías no instaladas

Las tipografías especificadas en CSS mediante la propiedad font-family **deben estar instaladas en el sistema donde se visualiza la página web. De lo contrario, no se visualizarán.**

```
p {  
    font-family: Vegur, Georgia, "Times New Roman", sans-serif;  
}
```

Un usuario con la tipografía Vegur instalada, vería sin problema el diseño con dicha tipografía, mientras que un usuario de Windows que no la tenga instalada en su sistema, la vería con la tipografía Georgia. De no tenerla instalada, la vería con Times New Roman. Esta última tipografía es una tipografía que viene instalada en los sistemas Windows, por lo que vería con esta tipografía si las anteriores fallan.

2. Tipografías entre sistemas

Por otro lado, imaginemos que ahora visita la web un usuario que utiliza Linux o Mac. Si no tiene las dos primeras tipografías, tampoco tendrá la tipografía Times New Roman, ya que es una tipografía que viene instalada por defecto en sistemas Windows, pero no en sistemas Linux o Mac, por lo que finalmente, utilizaría la tipografía segura «sans-serif» y cargaría una tipografía sin serifa.

Esto ha creado una cierta inconsistencia, ya que la tipografía se está viendo de forma diferente dependiendo del sistema, y lo que nos gustaría es que sea lo más consistente posible entre diferentes sistemas y se vea en todos con la tipografía elegida por el diseñador.

3. Tipografías con licencias

Muchas tipografías genéricas tienen derechos de autor y no se deberían utilizar en nuestra página web si no hemos pagado su licencia. De la misma forma, muchas tipografías no vienen instaladas de serie en el sistema operativo porque no tienen permiso ni licencia.

Las tipografías que vienen en sistemas Windows de serie (Times New Roman, Verdana, Tahoma, Trebuchet MS, ...) se verían correctamente en navegadores de usuarios con dicho sistema operativo, pero no ocurriría lo mismo en dispositivos con Linux o Mac. Y lo mismo con tablets o dispositivos móviles.

En definitiva, no es tan sencillo establecer una fuente específica para obtener el mismo resultado de diseño en todos los navegadores y sistemas disponibles hasta que llegó la [regla de CSS @font-face](#)

LA REGLAS @FONT-FACE Y @IMPORT

Permite descargar una fuente o tipografía de una página web, cargarla en el navegador y utilizarla en nuestras páginas, aunque no esté instalada en el sistema. Todo ello de forma transparente al usuario, sin que deba realizar ninguna acción.

La regla @font-face suele colocarse al principio del fichero CSS para avisar al navegador que vamos a utilizar una tipografía que es muy posible que no tenga instalada

Abrimos un bloque @font-face, establecemos su nombre mediante font-family y, opcionalmente, definimos sus características mediante propiedades como font-style o font-weight.

El factor clave viene a la hora de indicar la tipografía, que se hace mediante la propiedad **src (source)**

```
@font-face {  
    font-family: 'Open Sans';  
    font-style: normal;  
    font-weight: 400;  
    src:  
        local("Open Sans"),  
        url("/fonts/opensans.woff2") format("woff2"),  
        url("/fonts/opensans.woff") format("woff"),  
        url("/fonts/opensans.ttf") format("truetype"),  
        url("/fonts/opensans.otf") format("opentype");  
}
```

SITIOS PARA DESCARGAR TIPOGRAFÍAS PARA EL DISEÑO WEB

Google Fonts

Lo más común utilizar Google Fonts como repositorio proveedor de tipografías para utilizar en sitios web por varias razones:

Gratuitas: Disponen de un amplio catálogo de fuentes y tipografías libres y/o gratuitas.

Cómodo: Resulta muy sencillo su uso: Google nos proporciona un código y el resto lo hace él.

Existen dos formas de incluir la tipografía en nuestra página:

Mediante un fragmento de código HTML, marcando la **opción <link>**

Mediante un fragmento de código CSS, marcando la **opción @import**.

Si marcamos elegimos <link> es para añadirlo en al código HTML, antes de cerrar la etiqueta </head>:

Ejemplo del link ofrecido por Google fonts

```
<link href="https://fonts.googleapis.com/css2?family=Open+Sans&display=swap" rel="stylesheet">
```

Es un archivo CSS prefabricado de Google con reglas @font-face para cargar la tipografía Open Sans desde sus servidores.

Seleccionando @import en la página de Google Fonts, te da un código entre etiquetas <style> y </style>.

En ese caso, el fragmento de código iría en HTML, pero si lo quieres añadir en CSS, debes eliminar dichas etiquetas y la regla @import siempre debe ir al principio del fichero .css

```
/* Código CSS */  
@import  
url('https://fonts.googleapis.com/css2?family=Lato:wght@300;400&family=Roboto&display=swap');
```

Otros sitios para descargar tipografías gratis:

[Fonts.com](#)

[Adobe Fonts](#)

[Cloud Typography](#)

[Fonts Squirrel](#)

[Font Library](#)

MENÚS CSS

MENÚ HORIZONTAL CSS

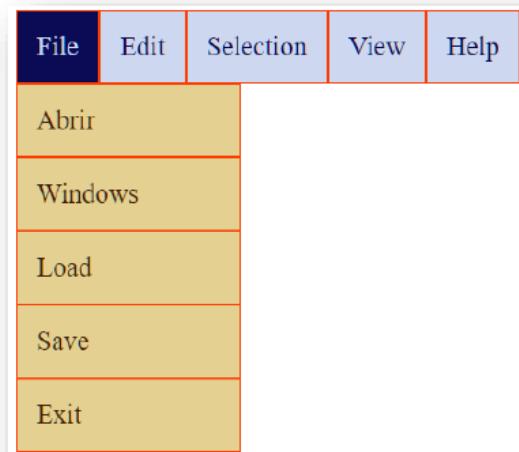
MENU HORIZONTAL CSS

```
nav ul, nav li {  
    margin: 0px;  
    padding: 0px;}
```

```
nav ul li {  
    float: left;  
    list-style-type: none;  
    position: relative;}
```

```
nav ul li ul {  
    display: none;  
    position: absolute;}
```

```
nav ul li:hover ul {  
    display: block;}
```



MENÚ VERTICAL CSS

MENU VERTICAL ACORDEON CSS

```
nav ul, nav li {  
    margin: 0px;  
    padding: 0px;}
```

```
nav ul li {  
    list-style-type: none;}
```

```
nav ul li ul {  
    display: none;}
```

```
nav ul li:hover ul {  
    display: block;}
```



TRANSICIONES Y ANIMACIONES CSS

TRANSICIONES

Las transiciones se basan en un principio muy básico: conseguir un efecto suavizado entre un estado inicial y un estado final al realizar una acción.

Propiedades	Descripción
transition-property	Propiedades CSS afectadas por la transición.
transition-duration	Tiempo de duración.
transition-timing-function	Ritmo de la transición.
transition-delay	Tiempo de retardo inicial.

Ritmo de la transición

Valor	Inicio	Transcurso	Final
ease	Lento	Rápido	Lento
linear	Normal	Normal	Normal
ease-in	Lento	Normal	Normal
ease-out	Normal	Normal	Lento
ease-in-out	Lento	Normal	Lento

```

a {
    background: #ddd;
    color: #222;
    padding: 8px;
}

a:hover {
    background: #fff;
    color: #666;
    padding: 25px;
    border: 1px solid #888;

    transition-property: all;
    transition-duration: 0.5s;
    transition-timing-function: linear;
    transition-delay: 2s;
}

/*Atajo transition */
/*transition: <property> <duration> <timing-function> <delay>
*/
a:hover {
    background: #fff;
    color: #666;
    padding: 25px;
    border: 1px solid #888;

    transition: all 0.5s linear 2s;
}

```

En el ejemplo, con **all** determinamos animar todas las propiedades que cambien:

- La propiedad **background** de color de fondo cambiará de #ddd a #fff
- La propiedad **color** de color de texto cambiará de #222 a #666
- La propiedad **padding** del tamaño del relleno cambiará de 8px a 25px
- La propiedad **border** cambiará de 0 a 1px solid #888.

Este último es un caso especial, puesto que cambia de estilos porque toma el estilo inicial por defecto, que es un **borde de 0px de grosor**. Cada una de estas transiciones se realizarán a un ritmo lineal, durante 0.5s de duración y con un retardo de 2s.

Transiciones de entrada y salida

La transición del ejemplo anterior se aplica sólo al mover el ratón sobre el elemento (transición de entrada). Sin embargo, si movemos el ratón fuera del enlace, no se produce transición, sino que realiza el cambio de forma brusca.

Esto ocurre porque le estamos diciendo que solo realice la transición cuando tenemos el ratón encima (:hover), en caso contrario no lo hará porque no hay definidas propiedades de transición.

Si movemos las propiedades de transición al primer bloque, se aplicarán tanto en las transiciones de entrada como en las transiciones de salida.

Por otro lado, si indicamos unas propiedades transition-* en a y otras en a:hover con diferentes valores, podremos conseguir (por ejemplo) duraciones diferentes.

ANIMACIONES

Las transiciones son una manera de suavizar un cambio de un estado inicial a un estado final. La idea de las animaciones CSS parten del mismo concepto, pero a diferencia de las transiciones, permiten añadir más estados aún. Así pues, con las animaciones podemos partir desde un estado inicial, a un estado posterior, a otro estado posterior, y así sucesivamente.

Para crear animaciones CSS es necesario realizar 2 pasos:

1. Utilizar la propiedad **animation** para indicar qué elemento HTML vamos a animar.
2. Definir mediante la regla **@keyframes** la animación en cuestión y sus estados (fotogramas clave).

La notación Kebab Case combina las palabras usando un guion - como nexo. Las letras estarán todas en minúsculas. Debido a su buen resultado visual y a su simpleza es el estándar que se usa en la creación de URLs.

Ejemplos:

La sintaxis de **contar palabras** en notación Kebab Case sería **contar-palabras**.

La sintaxis de **aumentar nivel dificultad** en notación Kebab Case sería **aumentar-nivel-dificultad**

Propiedades	Descripción	Valor
animation-name	Nombre de la animación a aplicar.	<i>nombre en notación Kebab Case</i>
animation-duration	Duración de la animación.	0 tiempo en segundos
animation-timing-function	Ritmo de la animación o transición	
animation-delay	Retardo en iniciar la animación.	0 tiempo en segundos
animation-iteration-count	Número de veces que se repetirá.	1 infinite número de repeticiones (no lleva unidad)
animation-direction	Dirección de la animación.	normal reverse alternate alternate-reverse
animation-fill-mode	Como se «completa» la animación.	none forwards backwards both
animation-play-state	Estado de la animación.	running paused

animation-direction indicamos el orden en el que se reproducirán los fotogramas, pudiendo escoger un valor entre los siguientes:

Valor	Significado
normal	Los fotogramas se reproducen en orden: desde el primero hasta el último.
reverse	Los fotogramas se reproducen en orden inverso: desde el final hasta el primero.
alternate	En iteraciones par, se reproducen como normal . En impares, como reverse .
alternate-reverse	En iteraciones par, se reproducen como reverse . En impares, como normal .

Mediante la propiedad **animation-fill-mode** podemos indicar que debe hacer la animación cuando no se está reproduciendo:

- El valor **none** una animación antes de arrancar y después de terminar (si no está establecida en repetición infinite) no tiene aplicados los estilos de la animación especificada.
- El valor **backwards** indica que la animación debe tener aplicados los estilos del fotograma inicial antes de empezar.
- El valor **forwards** indica que la animación debe tener aplicados los estilos del fotograma final al terminar.
- El valor **both** indica que debe aplicar los dos casos anteriores (backwards y forwards).

Atajo: Animaciones

CSS ofrece la posibilidad de resumir todas estas propiedades en una sola, para hacer nuestras hojas de estilos más compactas. El orden para los valores de la propiedad de atajo sería el siguiente:

```
div {  
    /* animation: <name> <duration> <timing-function> <delay>  
       <iteration-count> <direction> <fill-mode> <play-state> */  
  
    animation: change-color 5s linear 0.5s 4 normal forwards running;  
}
```

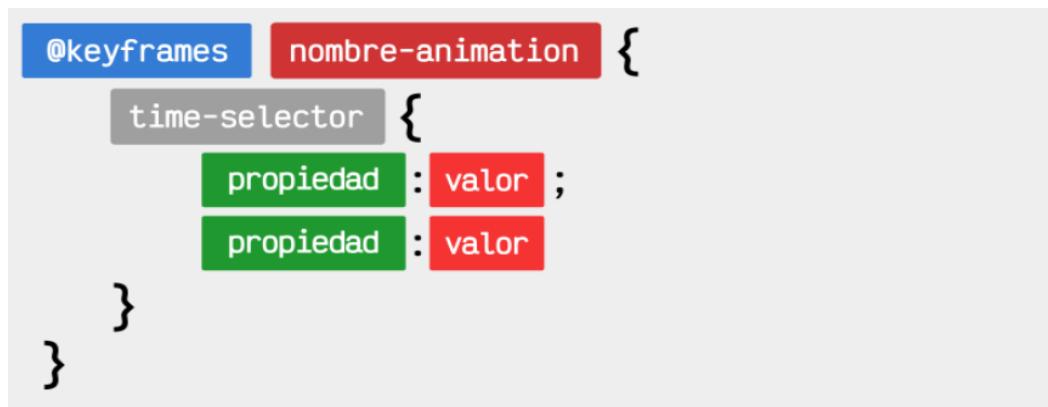
NOTA: Mucho cuidado al indicar los segundos en las propiedades [animation-duration](#) y/o [animation-delay](#). Si no indicamos la unidad (incluso con valores de cero), se interpretará como el valor de la propiedad [animation-iteration-count](#), que es el número de veces que se repite la animación (y no lleva unidad).

REGLA @KEYFRAMES

Una animación está formada por varios fotogramas, una secuencia de imágenes (30-60 fotogramas por segundo, por ejemplo) que mostradas una detrás de otra generan el efecto de movimiento que conocemos de una animación.

En CSS, los fotogramas se crean a partir de propiedades CSS, y no hace falta definir tantos fotogramas. Sólo crearemos [fotogramas clave](#) y el resto de fotogramas los [generará el navegador](#).

Para definir esos fotogramas clave, utilizaremos la [regla @keyframes](#), la cual es muy sencilla de utilizar. Se basa en el siguiente esquema:



Cada **time-selector** será un momento clave de cada uno de los fotogramas de nuestra animación, y ya veremos que se pueden definirse muchos en una misma animación.

Parte	Descripción
<code>@keyframes name</code>	Regla para darle un nombre y definir los fotogramas clave de una animación.
<code>from</code>	Fotograma clave inicial con los estilos CSS a aplicar. Equivalente a 0% .
<code>to</code>	Fotograma clave final con los estilos CSS a aplicar. Equivalente a 100% .
porcentaje	Porcentaje específico de la animación con los estilos CSS a aplicar. Permite decimales.

```
@keyframes change-color {  
    from { background: red; } /* Primer fotograma 0% */  
    to { background: green; } /* Segundo y último fotograma 100%*/  
}
```

No basta con definir la animación mediante `@keyframes`, también que hay que asociar la animación al elemento o etiqueta HTML al que queremos aplicársela:

```
.animated {  
    background: grey;  
    color: #FFF;  
    width: 150px;  
    height: 150px;  
    animation: change-color 2s ease 0s infinite;  
}
```

Selectores porcentuales

Los selectores **from** y **to** son muy similares a colocar **0%** y **100%**, pero podemos añadir nuevos fotogramas intermedios utilizando porcentajes

```

@keyframes change-color {
    0% {
        background: red;           /* Primer fotograma */
    }
    50% {
        background: yellow;      /* Segundo fotograma */
        width: 400px;
    }
    100% {
        background: green;       /* Último fotograma */
    }
}
.animated {
    background: grey;
    color: #FFF;
    width: 150px;
    height: 150px;
    animation: change-color 2s ease 0s infinite;
}

```

Propiedades CSS a las que se les pueden aplicar animaciones y/o transiciones:

- Propiedades de color: color, opacity
- Propiedades de columnas: column-width, column-count, column-gap, column-rule-color, column-rule-width.
- Propiedades de texto: letter-spacing, tab-size, text-indent, word-spacing.
- Propiedades de decoración de texto: text-decoration-color, text-shadow.
- Propiedades flex: flex-basis, flex-grow, flex-shrink, order.
- Propiedades de fondo y bordes: background-color, background-position, background-size, border-witdh, border-radius, border-image.
- Propiedades del box-model: box-shadow, margin, padding, max-height, min-height, height, max-width, min-width, width, visibility, vertical-align.
- Propiedades de posicionamiento: bottom, left, right, top, z-index.
- Propiedades de tipografía: font-weight, font-strech, font-size, line-height, font-size-adjust.
- Propiedades de interacción: outline-color, outline-width, outline-offset.
- Propiedades de transformación: transform, transform-origin, perspective, perspective-origin.
- Propiedades de forma: shape-outside, shape-margin, shape-image-threshold.

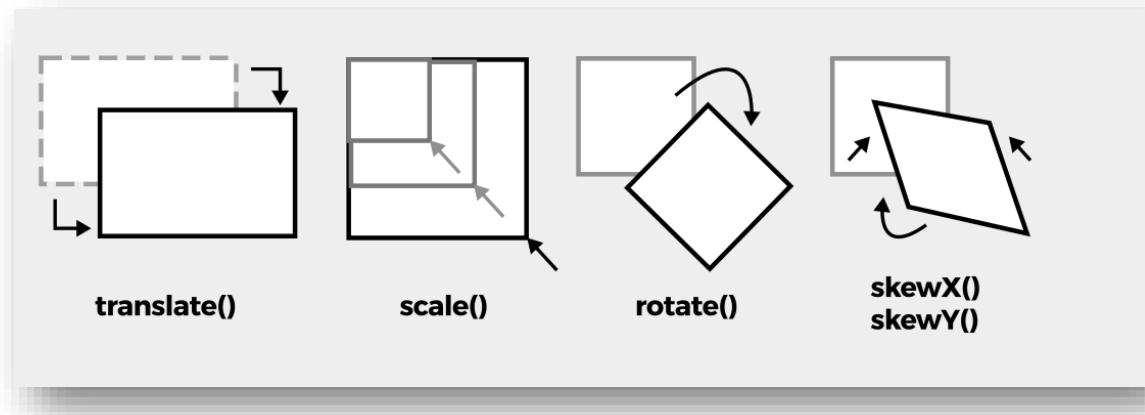
TRANSFORMACIONES 2D

Las transformaciones son una de las características de CSS más interesantes y potentes que se introducen en el lenguaje para convertir las hojas de estilo en un sistema capaz de realizar efectos visuales 2D y 3D. Con ellas podemos hacer cosas como mover elementos, rotarlos, aumentarlos o disminuirlos.

La propiedad transform

Estas transformaciones se pueden efectuar en CSS mediante la propiedad `transform` que permite recibir una función de transformación determinada, que será aplicada en el elemento HTML en cuestión seleccionado mediante CSS. Dicho elemento HTML se verá transformado visualmente.

Tipo de transformación	Descripción
Translación 2D	Desplaza un elemento en el eje X (<i>izquierda, derecha</i>) y/o en el eje Y (<i>arriba, abajo</i>)
Escalado 2D	Escala el elemento una determinada cantidad más grande o más pequeña. También se puede voltear.
Rotación 2D	Gira el elemento sobre su eje X o sobre su eje Y . También se puede girar sobre sí mismo.
Deformación 2D	Inclina el elemento sobre su eje X o sobre su eje Y .



Se pueden emplear múltiples transformaciones separándolas mediante espacio. En el siguiente ejemplo, aplicamos una función de rotación, una función de escalado y una función de traslación de forma simultánea:

```
.element {
    transform: rotate(5deg) scale(2) translate(25px, 150px);
}
```

FUNCIONES DE TRANSLACIÓN

Respecto al ámbito 2D, tenemos 3 funciones de translación: `translateX()` y `translateY()` para cada eje, y la propiedad de atajo `translate()`, que es una mezcla de las dos anteriores en una sola función de translación.

Funciones	Significado
<code>translateX(x)</code>	Traslada el elemento una distancia de <code>SIZE</code> <i>x</i> horizontalmente.
<code>translateY(y)</code>	Traslada el elemento una distancia de <code>SIZE</code> <i>y</i> verticalmente.
<code>translate(x, y)</code>	Propiedad de atajo de las dos anteriores.
<code>translate(x)</code>	Equivalente a <code>translate(x, 0)</code>

```
.element {
    transform: translateX(20px) translateY(-30px);
}

/* La transformación anterior es equivalente a esta (atajo) */
.element {
    transform: translate(20px, -30px);
}
```

La propiedad `translate`

En nuevas versiones de los navegadores, ya se soporta la propiedad individual `translate`, y no hace falta utilizarla dentro de la propiedad `transform`.

Propiedad	Valor	Significado
<code>translate</code>	<code>none</code>	No aplica desplazamiento. Valor por defecto.
<code>translate</code>	<code>SIZE</code>	Desplaza un elemento el tamaño especificado en el eje X .
<code>translate</code>	<code>SIZE SIZE</code>	Desplaza un elemento una cierta cantidad en el eje X y eje Y .

Se puede indicar 1 parámetro, 2 parámetros o 3 parámetros, dependiendo de los ejes que quieras desplazar. En el caso de indicar sólo un parámetro, los demás ejes serán 0px por defecto.

```
.element {
  translate: 50px;           /* Equivalente a translateX(50px) */
  translate: 50px 150px;     /* Equivalente a translate(50px, 150px) */
  translate: 0 150px;        /* Equivalente a translateY(150px) */
}
```

FUNCIONES DE ROTACIÓN

Las funciones de rotación son aquellas que realizan una transformación en la que giran un elemento respecto a un eje específico.

Funciones	Significado
<code>rotateX(x)</code>	Establece una rotación 2D en <code>ANGLE</code> x sólo para el eje horizontal X.
<code>rotateY(y)</code>	Establece una rotación 2D en <code>ANGLE</code> y sólo para el eje vertical Y.
<code>rotateZ(z)</code>	Establece una rotación 2D en <code>ANGLE</code> z sobre si mismo.
<code>rotate(z)</code>	Alias a la anterior.

```
.element {
  transform: rotateX(30deg) rotateY(20deg);
}

.element {
  transform: rotateZ(5deg);
}
```

La propiedad rotate

En nuevas versiones de los navegadores, ya se soporta la propiedad individual `rotate`, y no hace falta utilizarla dentro de la propiedad `transform`.

Propiedad	Valor	Significado
<code>rotate</code>	<code>none</code>	No aplica rotación. Valor por defecto.
<code>rotate</code>	<code>ANGLE</code>	Rota el elemento sobre si mismo. Equivalente a <code>rotateZ()</code> .
<code>rotate</code>	<code>AXIS ANGLE</code>	Rota el elemento sobre el eje (<code>X</code> , <code>Y</code> o <code>Z</code>) indicado.
<code>rotate</code>	<code>NUMBER NUMBER NUMBER ANGLE</code>	Indica un vector de rotaciones con el ángulo indicado.

```

.element {
  rotate: 45deg;           /* Equivale a transform: rotateZ(45deg); */

  rotate: x 45deg;         /* Equivale a transform: rotateX(45deg); */
  rotate: y 120deg;        /* Equivale a transform: rotateY(120deg); */

  rotate: 0 0 1 45deg;     /* Equivale a transform: rotateZ(45deg); */
  rotate: 1 0 0 15deg;    /* Equivale a transform: rotateX(15deg); */
  rotate: 0 1 1 5deg;      /* Equivale a transform: rotateY(5deg) rotateZ(5deg); */
}

```

FUNCIONES DE ESCALADO

Las funciones de escalado son aquellas que realizan una transformación en la que aumentan o reducen el tamaño de un elemento. Para ello, las utilizaremos en el interior de la propiedad CSS transform

Funciones	Significado
<code>scaleX(fx)</code>	Reescalas el elemento un número de <code>NUMBER</code> f_x veces (sólo en horizontal).
<code>scaleY(fy)</code>	Reescalas el elemento un número de <code>NUMBER</code> f_y veces (sólo en vertical).
<code>scale(fx, fy)</code>	Propiedad de atajo de las dos anteriores (escalamiento simétrico).
<code>scale(fx)</code>	Equivalente al anterior: <code>scale(fx, fx)</code> .

La **propiedad transform: scale(2, 0.5)** realiza una transformación de escalamiento del elemento, ampliéndolo al doble de su tamaño original en el eje X (horizontal) y a la mitad en el eje Y (vertical).

La propiedad scale

En nuevas versiones de los navegadores, ya se soporta la propiedad individual scale, y **no hace falta utilizarla dentro de la propiedad transform.**

Propiedad	Valor	Significado
<code>scale</code>	<code>none</code>	No aplica escalamiento. Valor por defecto.
<code>scale</code>	<code>NUMBER</code>	Aplica el factor de escala simétrico al eje X/Y. Igual a <code>scale: x x 1</code> .
<code>scale</code>	<code>NUMBER NUMBER</code>	Aplica los factores de escala al eje X y eje Y. Igual a <code>scale: x y 1</code> .
<code>scale</code>	<code>NUMBER NUMBER NUMBER</code>	Aplica un factor de escala a cada eje. Igual a <code>scale: x y z</code> .

EFFECTO ESPEJO CON CSS

Con la función de escalado de CSS se puede hacer un efecto «mirror» y darle la vuelta a una imagen, por ejemplo, de forma muy sencilla. Basta con utilizar la función scale(-1) con valor negativo. Si 1 representa a la imagen tal cual está, -1 es la imagen invertida.

```
.image {  
    transform: scaleX(-1); /* Imagen espejo en horizontal */  
    transform: scaleY(-1); /* Imagen espejo en vertical (boca abajo) */  
  
    transform: scale(-1); /* Equivalente a las dos anteriores a la vez */  
}
```

FUNCIONES DE DEFORMACIÓN

Las funciones de deformación establecen un ángulo para torcer, tumbar o inclinar un elemento en 2D

Funciones	Significado
<code>skewX(xdeg)</code>	Establece un ángulo de <code>ANGLE</code> <code>xdeg</code> para una deformación 2D respecto al eje X.
<code>skewY(ydeg)</code>	Establece un ángulo de <code>ANGLE</code> <code>ydeg</code> para una deformación 2D respecto al eje Y.

```
.element {  
    transform: skewX(-5deg);  
    transform: skewY(25deg);  
  
    transform: skewX(5deg) skewY(15deg);  
}
```

FUNCIÓN MATRIX

Es una función de transformación 2D que permite aplicar una transformación de matriz personalizada a un elemento HTML. Esta función toma seis valores numéricos que representan los elementos de una matriz de transformación de 2D.

Matrix() combina todos los métodos de transformación 2D en uno.

La sintaxis básica de la función matrix() es la siguiente:

```
transform: matrix(a, b, c, d, e, f);
```

Donde:

- **a** representa la escala horizontal
- **b** representa la inclinación horizontal
- **c** representa la inclinación vertical
- **d** representa la escala vertical
- **e** representa la traslación horizontal
- **f** representa la traslación vertical

El método matrix () toma seis **parámetros sin indicar unidad**, que contienen funciones matemáticas, lo que le permite **rotar, escalar, mover y sesgar**(deformar) elementos HTML.

Los parámetros son los siguientes:

```
matrix ( scaleX (), skewY (), skewX (), scaleY (), translateX (), translateY () )
```

La matriz de transformación "unidad", es una matriz de transformación que no cambia la forma, el tamaño o la posición de ninguna manera se escribiría como:

```
transform: matrix(1, 0, 0, 1, 0, 0)
```

Los números de las posiciones scaleX y scaleY posiciones son ambos 1 (que no están cambiando el tamaño), y skewY y skewX son 0 (no estamos sesgar el objeto en absoluto), y finalmente, translateX y translateY son 0, así que no estamos moviendo el cualquier lugar objeto.

Estos valores se utilizan para crear una matriz de transformación de 2D que se aplica a un elemento HTML.

Por ejemplo, para escalar un elemento en el eje X por un factor de 1.5 y trasladarlo horizontalmente en 50 píxeles, se podría utilizar la siguiente regla de CSS:

```
transform: matrix(1.5, 0, 0, 1, 50, 0);
```

También es posible utilizar la función matrix() en combinación con otras funciones de transformación 2D, como translate(), rotate() y scale(), para crear transformaciones más complejas y precisas.

Es importante tener en cuenta que la función matrix() es una función de transformación 2D y no se puede utilizar para realizar transformaciones 3D. Para eso, se debe utilizar la función matrix3d().

FUNCIÓN PERSPECTIVE

Define una transformación que establece la distancia entre el usuario y el plano Z, la perspectiva desde la que estaría el espectador si el plano bidimensional fuera tridimensional.

Da a un elemento 2D un espacio 3D, afectando a la distancia entre el plano Z y el usuario.

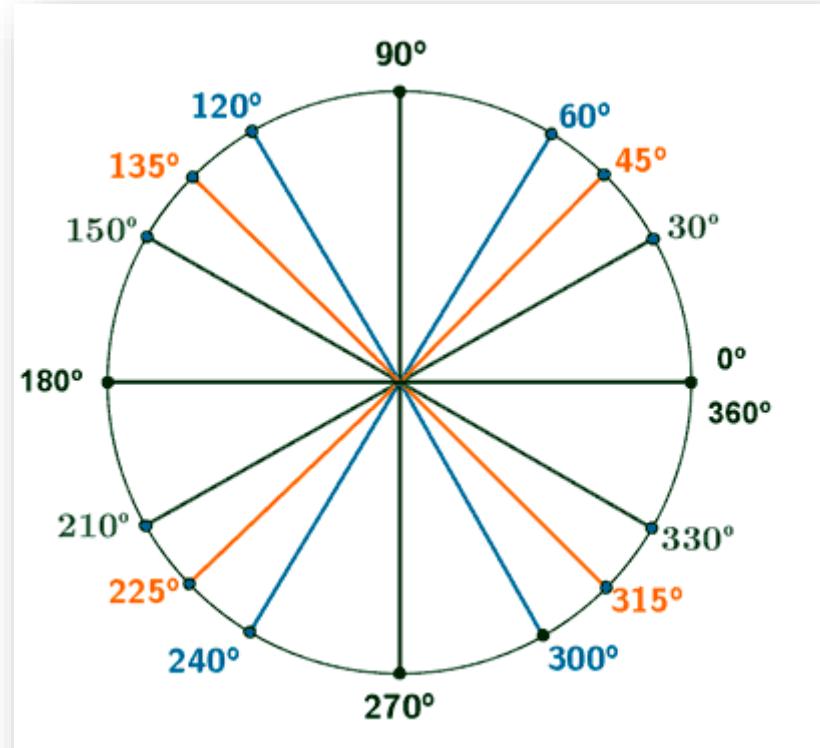
La fuerza del efecto viene determinada por el valor. Cuanto menor sea el valor, más cerca estará del plano Z y más impresionante será el efecto visual. Cuanto mayor sea el valor, más sutil será el efecto.

```
transform: perspective(100px);
```

IMPORTANTE, en CSS también **existe la propiedad perspective**, que se aplica al contenedor padre y su funcionamiento es muy diferente al de la función perspective()

GRADOS SEXAGESIMALES

El modo DEG se refiere a los grados sexagesimales, es decir, la porción resultante de dividir una circunferencia en 360 partes iguales. Éstos son los que tenemos que utilizar cuando vamos a trabajar con grados en CSS. El nombre DEG proviene del inglés «Degree» que significa grado



LA PROPIEDAD TRANSFORM-ORIGIN O PUNTO DE ORIGEN

La propiedad transform-origin nos permite cambiar el punto de origen de una transformación. Esta recibe por parámetro la posición de origen de cada eje (X e Y) y que por defecto, está establecida a 50% 50%:

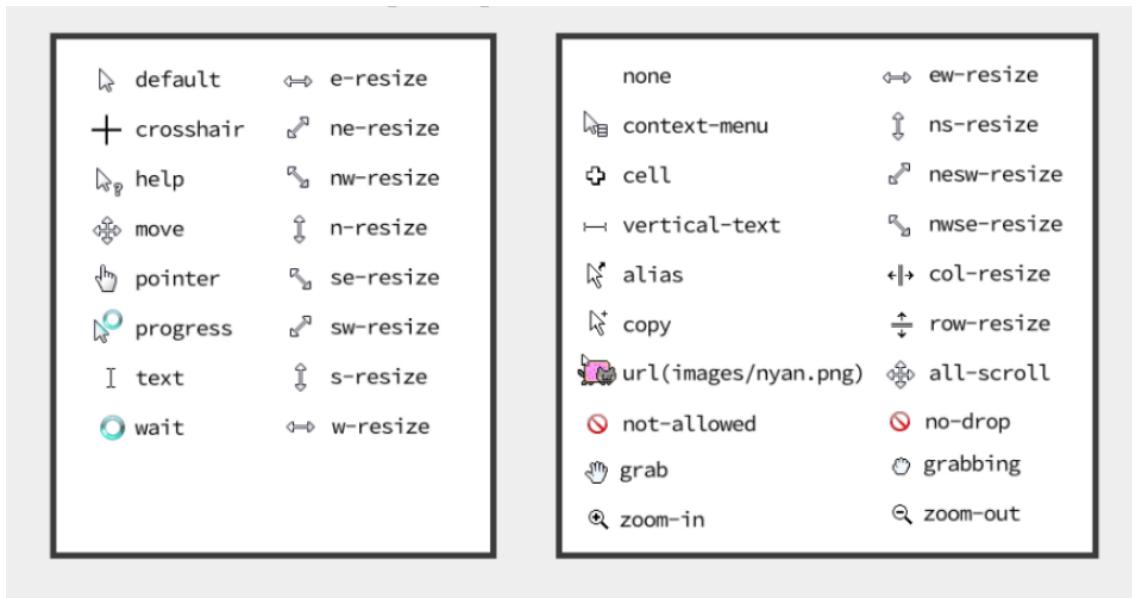
Propiedades	Formato	Significado
transform-origin	posX, posY	Cambia el punto de origen del elemento en una transformación.

El valor de la propiedad	Descripción
eje x	Define dónde se coloca la vista en el eje x. <ul style="list-style-type: none">• izquierda (left)• centro (center)• derecha (right)• longitud en px• %
eje y	Define dónde se coloca la vista en el eje y. <ul style="list-style-type: none">• parte superior (top)• centro (center)• abajo (bottom)• longitud en px• %

PROPIEDAD CURSOR

Para modificar el cursor del ratón solo tenemos que especificar la propiedad cursor dentro del elemento que busquemos, junto al valor del cursor deseado:

Propiedad	Valor	Significado
cursor	<u>palabra clave de cursor</u>	Muestra un cursor de ratón específico.



url() : Especifica un archivo donde se encuentra la imagen que deseamos usar como cursor. El archivo de imagen especificado en la URL debe tener el formato cursor (.cur) o cursor animado (.ani). Algunos navegadores aceptan formatos de imágenes (.png .jpg .gif). Utilizad imágenes de aproximadamente de 50 px y con fondo transparente.

```
div {  
    cursor: url(imagen.ext), text;  
    /* Formato básico */  
    cursor: url(imagen.ext) pos-x pos-y, pointer;  
    /* Con posición */  
  
    cursor: url(imagen1.ext), url(imagen2.ext2), wait;  
    /* Con dos extensiones para distintos navegadores*/  
}
```

Soporte:

- Internet Explorer sólo soporta cursores en formato . CUR (propietario de Microsoft).
- Chrome, Firefox y Safari requieren una imagen (recomendado: PNG de 24-bits).
- Firefox requiere especificar un segundo cursor para el caso en el que no pueda cargar la imagen.
- Chrome, Firefox y Safari aceptan el uso de pos-x y pos-y para especificar el píxel donde empezar a recortar la imagen. Por defecto, utiliza 0 0.
- Los cursores estáticos de tipo .cur los podemos utilizar en todos los navegadores, por lo que son una opción muy indicada si no queremos tener problemas.
- Los cursores de tipo .ani (cursores animados) solamente funcionan en Internet Explorer, por lo que si deseamos utilizarlos, debemos especificar una alternativa para los otros navegadores.
- Los cursores de tipo imagen (GIF o PNG) no funcionan en Internet Explorer, por lo que deberíamos asignar una alternativa para este navegador. Cursores tipo JPG no me funcionan
- El tipo especial de cursor GIF Animado en Firefox y Chrome solo muestra el primer fotograma del GIF, por lo que no veremos animación.

Valor	Significado
default	Muestra el cursor del ratón por defecto del sistema. Usualmente, una flecha/cursor.
crosshair	Muestra una cruceta. Útil para tareas en las que requieres precisión.
help	Muestra un cursor de ayuda. Generalmente, una interrogación o un puntero con interrogación.
move	Muestra un cursor para mover elementos. Se suele representar con flechas hacia todos lados.
pointer	Muestra un cursor para hacer click. Usualmente, una mano o algún tipo de apuntador.
progress	Muestra un cursor que indica que se está trabajando en segundo plano.
text	Muestra un cursor que permite seleccionar texto de una forma más cómoda.
wait	Muestra un cursor que indica que se está trabajando en primer plano y deberías esperar.
none	No muestra ningún cursor, pero el cursor del ratón sigue funcionando. Útil para bromas.

Valor	Significado
<code>context-menu</code>	Muestra un cursor de texto junto a un menú contextual (al hacer clic con botón derecho).
<code>cell</code>	Muestra un cursor de cruceta, utilizado en hojas de cálculo para ajustar celdas.
<code>vertical-text</code>	Idem al cursor <code>text</code> , pero con orientación para texto vertical.
<code>alias</code>	Muestra un cursor que representa un alias o «acceso directo» a algo.
<code>copy</code>	Muestra un cursor que representa una copia. Usualmente un cursor con un símbolo + .
<code>no-drop</code>	Muestra un cursor que indica que no se puede arrastrar y soltar en ese lugar.
<code>not-allowed</code>	Muestra un cursor que indica que no se puede realizar una acción (prohibido).
<code>grab</code>	Muestra un cursor que indica que algo se puede agarrar y arrastrar (drag & drop).
<code>grabbing</code>	Muestra un cursor que indica que algo se está arrastrando (drag & drop).
<code>zoom-in</code>	Muestra un cursor que indica que se puede acercar la imagen (lupa con signo +).
<code>zoom-out</code>	Muestra un cursor que indica que se puede alejar la imagen (lupa con signo -).

En CSS3 se añaden otros cursor de orientación a puntos cardinales como:

`ew-resize` , `ns-resize` , `nesw-resize` , `nwse-resize` ,

Otros para redimensionar filas o columnas de una tabla

`row-resize` y `col-resize` o `all-scroll` que indica que se puede hacer scroll en cualquier dirección.

http://webdesigntutsplus.s3.amazonaws.com/tuts/405_cursors/table/table.html

Página para descargar imágenes transparentes gratis

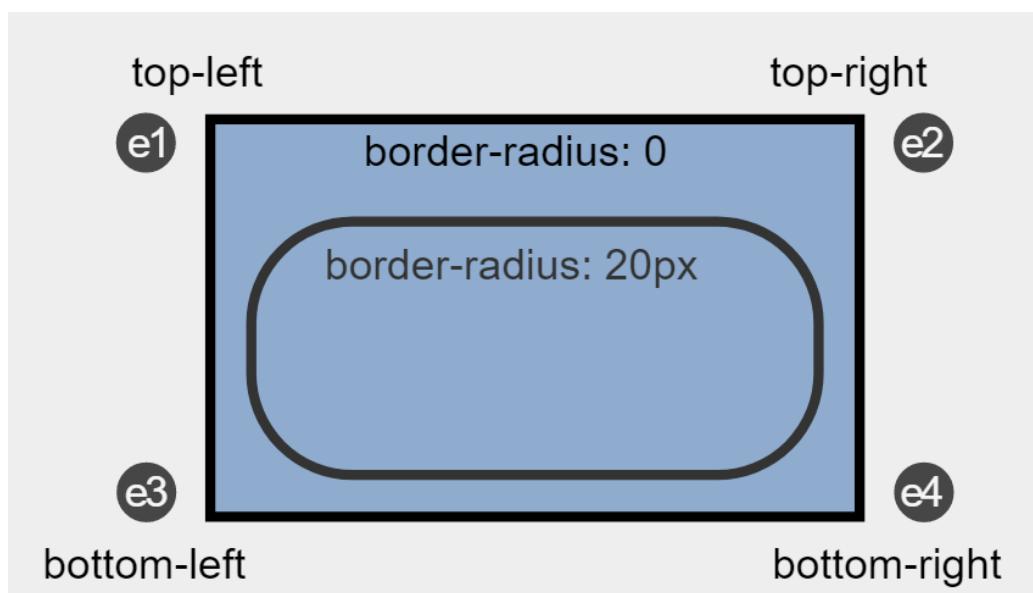
<https://pngimg.es/>

<http://www.rw-designer.com/cursor-library>

PROPIEDAD BORDER-RADIUS

CSS3 añade interesantes características en materia de bordes, como la posibilidad de crear bordes con esquinas redondeadas.

La idea es especificar un radio para el borde de las esquinas de la caja. Por defecto, este borde es de tamaño 0, por lo que no hay borde redondeado. A medida que se aumenta este valor, el borde se redondea más. Una vez llegado a su máximo, no se apreciará ningún cambio.



Propiedad	Valor	Significado
<code>border-radius</code>	<code>size</code>	1 parámetro. Aplica el radio a todas y cada una de las esquinas.
	<code>size size</code>	2 parámetros: top-left + bottom-right y a top-right + bottom-left .
	<code>size size size</code>	3 parámetros: top-left , a top-right y bottom-left y a bottom-right .
	<code>size size size size</code>	4 parámetros. Orden de las agujas del reloj, empezando por top-left .

- **El primer formato**, un único parámetro, aplica ese tamaño a todas las esquinas del borde.
- **El segundo formato**, con dos parámetros, aplica el primer valor, **e1**, a las esquinas superior-izquierda e inferior-derecha, y el segundo valor, **e2**, a las esquinas superior-derecha e inferior-izquierda.
- **En el tercer formato**, se aplica el parámetro **e1** a la esquina superior-izquierda, el parámetro **e2** a las esquinas superior-derecha e inferior-izquierda y el parámetro **e3** a la esquina inferior-derecha.
- **En el cuarto formato**, se aplica el tamaño de cada valor a cada esquina por separado, en el sentido de las agujas del reloj. O lo que es lo mismo, e1 a la esquina superior-izquierda, e2 a la esquina superior-derecha, e3 a la esquina inferior-derecha y e4 a la esquina inferior-izquierda.

```
div {
    border-radius: 25px;                  /* 1 parámetro */
    border-radius: 25% 50%;              /* 2 parámetros */
    border-radius: 50px 25px 10px;      /* 3 parámetros */
    border-radius: 25px 0 15px 50px;    /* 4 parámetros */
}
```

Esquinas irregulares

Es posible **diferenciar el radio horizontal del radio vertical** de una esquina determinada, creando una esquina redondeada irregular. Para conseguirlo, no hay más que añadir una barra (/) y repetir nuevamente el número de parámetros escogido. De esta forma, los parámetros a la izquierda de la barra representan el radio horizontal, mientras que los que están a la derecha, representan el radio vertical.



```

div {
    border-radius: 25% / 50%;
    /* 1 parámetro (x / y) */
    border-radius: 5px 50px / 50px 15px;
    /* 2 parámetros (x / y) */
    border-radius: 50px 25px 10px / 50px 75px 10px;
    /* 3 parámetros (x / y) */
    border-radius: 5px 50px 5px 50px / 50px 15px 50px 15px;
    /* 4 parámetros (x / y) */
}

```

Es posible especificar los valores de cada esquina mediante propiedades por separado:

Propiedad	Valor	Significado
border-top-left-radius	size	Indica un radio para redondear la esquina top-left .
border-top-right-radius	size	Indica un radio para redondear la esquina top-right .
border-bottom-left-radius	size	Indica un radio para redondear la esquina bottom-left .
border-bottom-right-radius	size	Indica un radio para redondear la esquina bottom-right .

`border-radius: 5px;`

`border-top-left-radius: 40%;`

`border-radius: 10px;`

`border-radius: 80px 20px;`

`border-radius: 20px;`

`border-radius: 60px 20px 5px;`

`border-radius: 40px;`

`border-radius: 60px 20px 5px
90px;`

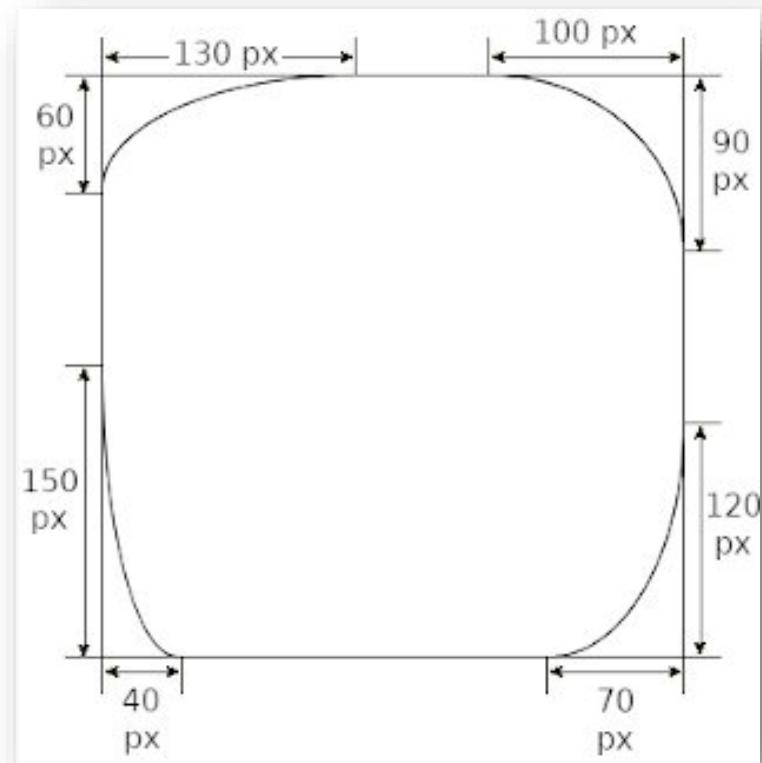
`border-radius: 60px;`

`border-radius: 60px/20px;`

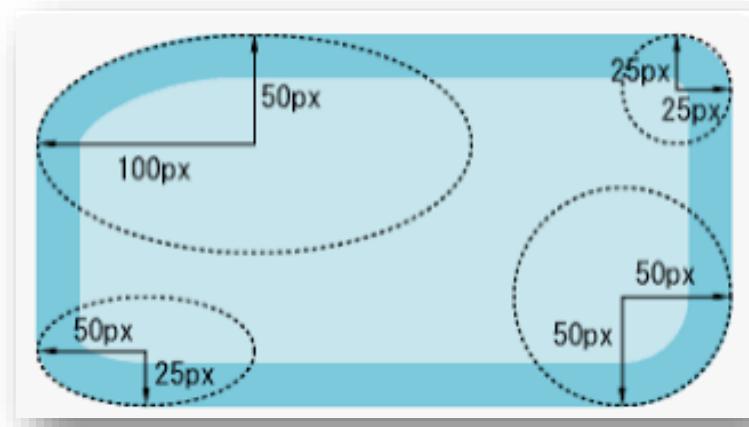
`border-radius: 13em/3em;`

`border-radius: 50%;`

`border-radius: 130px 100px 70px 40px / 60px 90px 120px 150px;`



`border-radius: 100px 25px 50px 50px / 50px 25px 50px 25px;`



PROPIEDAD BOX-SHADOW

Sombras en elementos, las sombras box-shadow están desactivadas sobre cualquier elemento.

Propiedad	Valor	Significado
box-shadow	none	Elimina (o simplemente no establece) sombra sobre un elemento.
box-shadow	posX poY	Crea una sombra color negro desplazándola ligeramente en horizontal y/o vertical.
box-shadow	posX poY size	Desenfocando o difuminando la sombra.
box-shadow	posX poY size size	Aplica un factor de crecimiento a la sombra.
box-shadow	posX poY size color	Cambia el color de la sombra.
box-shadow	posX poY size color inset	Estableciendo una sombra interna en lugar de externa.

Los dos primeros parámetros, **posX** y **poY**, son los parámetros obligatorios mínimos para hacer funcionar la propiedad box-shadow, donde indicamos el desplazamiento que tendrá la sombra en el eje x (horizontal) y en el eje y (vertical). Los valores negativos invierten la dirección de la sombra. Si 5px 5px mueve la sombra 5 píxeles a la derecha y hacia abajo, -5px -5px movería la sombra 5 píxeles a la izquierda, y 5 píxeles hacia arriba.

```
.element {  
    box-shadow: 5px 5px;  
}
```

El tercer parámetro de la propiedad box-shadow indica **la cantidad de desenfoque o difuminado que queremos utilizar en nuestra sombra**. Por defecto, tiene un valor de 0, o lo que es lo mismo, la sombra será igual a la caja original, por lo que será completamente lisa, sin difuminar. Este valor puede irse ampliando y de esta forma conseguiremos una sombra más desenfocada:

```
.element {  
    box-shadow: 5px 5px 0;          /* Sombra sin desenfoque */  
    box-shadow: 5px 5px 2px;        /* Sombra con ligero desenfoque */  
    box-shadow: 5px 5px 10px;       /* Sombra desenfocada */  
    box-shadow: 5px 5px 40px;       /* Sombra un desenfoque casi disipado */  
}
```

Un cuarto parámetro opcional permite indicar un factor de crecimiento para la sombra. El parámetro de factor de crecimiento es un parámetro donde **podemos indicar una unidad que hará crecer la sombra en todos sus lados** el tamaño indicado, de forma que crezca un poco más de lo que ocupa. Algo que puede ser realmente útil, por ejemplo, en el caso de que el desplazamiento sea 0 (la sombra está justo detrás del elemento) y queremos que se muestre ligeramente alrededor del elementos al que le aplicamos box-shadow.

```
.element {  
    box-shadow: 0 0 10px 5px;  
}
```

Esto hará que la sombra esté posicionada justo detrás del elemento, tenga un nivel de desenfoque de 10px y, además, la hagamos crecer unos 5px por cada lado.

Las sombras creadas con box-shadow permiten indicar la palabra clave **inset**, lo que hará que la sombra en lugar de colocarse por fuera de nuestro elemento y ser una sombra exterior (por defecto), pasará a ser una sombra interior y colocarse por dentro del elemento. Ten en cuenta que en este caso, los desplazamientos indicados se invierten, de modo que si teníamos una sombra por la zona inferior-derecha, tendríamos que invertir los valores para que la sombra interior también esté en la zona inferior-derecha:

```
.element {  
    /* Sombra exterior que se desplaza hacia la zona inferior-derecha */  
    box-shadow: 10px 10px 5px black;  
  
    /* Sombra interior que se desplaza hacia la zona inferior-derecha */  
    box-shadow: -10px -10px 5px black inset;  
}
```

La propiedad box-shadow permite indicar valores múltiples separados por coma, permitiendo en este caso, crear sombras múltiples e independientes y de diferentes colores.

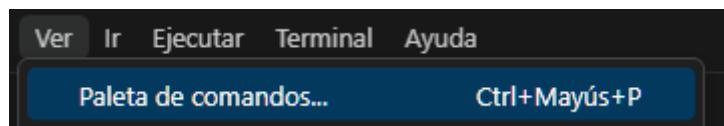
```
.element {  
  box-shadow:  
    5px 5px 10px black,  
    10px 10px 10px red,  
    20px 20px 10px blue,  
    10px 10px 10px rgba(12, 173, 7, 0.5) inset;  
}
```

ELEMENTOS DE MAQUETACIÓN: FLEX

Instalar en Visual Studio Code:

- CSS Flexbox Cheatsheet

Ejecutar en Visual Studio Code:



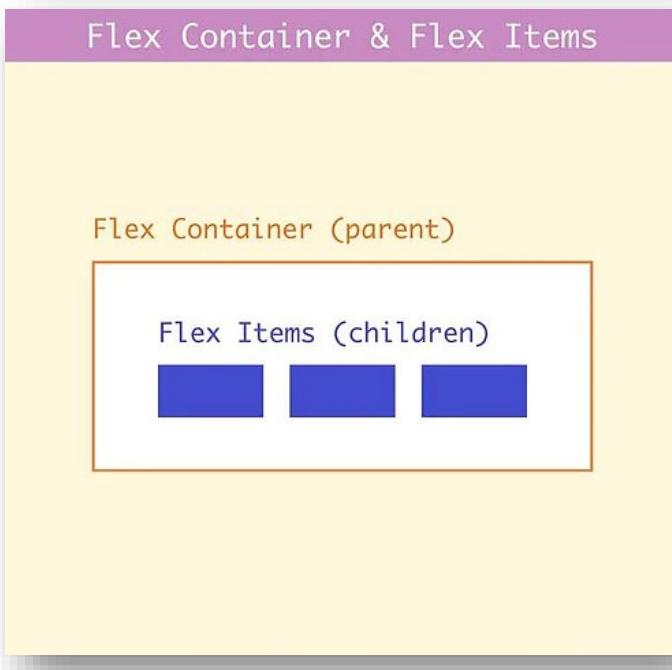
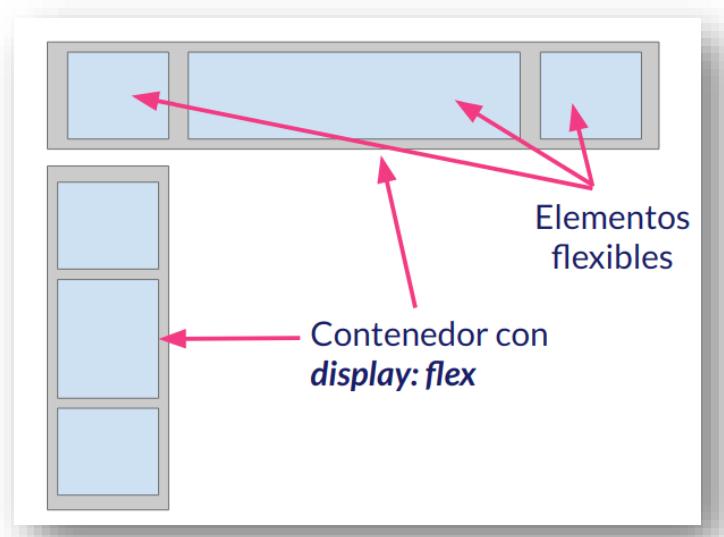
Open CSS Flexbox Cheatsheet

Elementos de maquetación flex

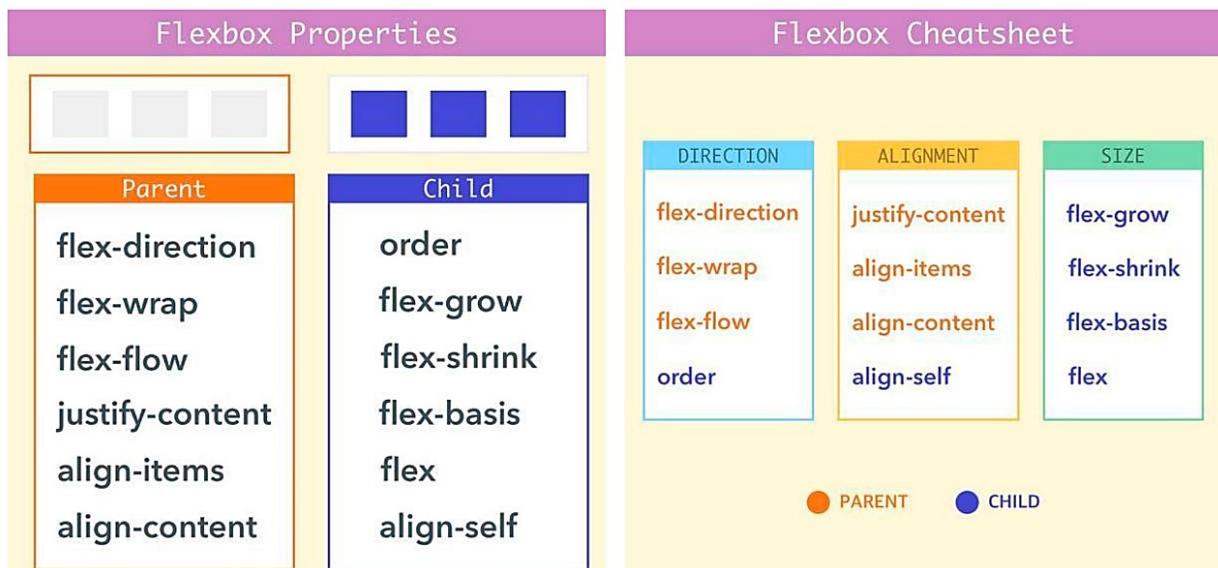
La principal idea que hay detrás de la maquetación FLEX es que vamos a tener un elemento, es decir, una etiqueta que va a poder controlar las propiedades de los elementos que contiene.

Por lo tanto, en esta situación, vamos a poder distinguir dos tipos de elementos:

- El **contenedor flex** que tendrá asignada la propiedad CSS ***display:flex*** y que va a controlar ciertas propiedades de los elementos que contiene.
- Los **elementos flexibles** que son los elementos que están dentro del contenedor y cuyas propiedades modificaremos.



Hay propiedades que afectan a los contenedores (padre) y otras a los elementos flexibles (hijos):



PROPIEDADES PARA LOS CONTENEDORES (PADRE)

El primer paso para maquetar usando elementos FLEX es añadir la propiedad FLEX al contenedor. En cuanto hagamos esto vamos a poder comprobar que empiezan a pasar cosas:

Si tengo este HTML:

```
<div class="container">
  <header>
    <div>
      <p>Primera frase</p>
    </div>
    <div>
      <p>Segunda frase</p>
    </div>
    <div>
      <p>Tercera frase</p>
    </div>
  </header>
</div>
```

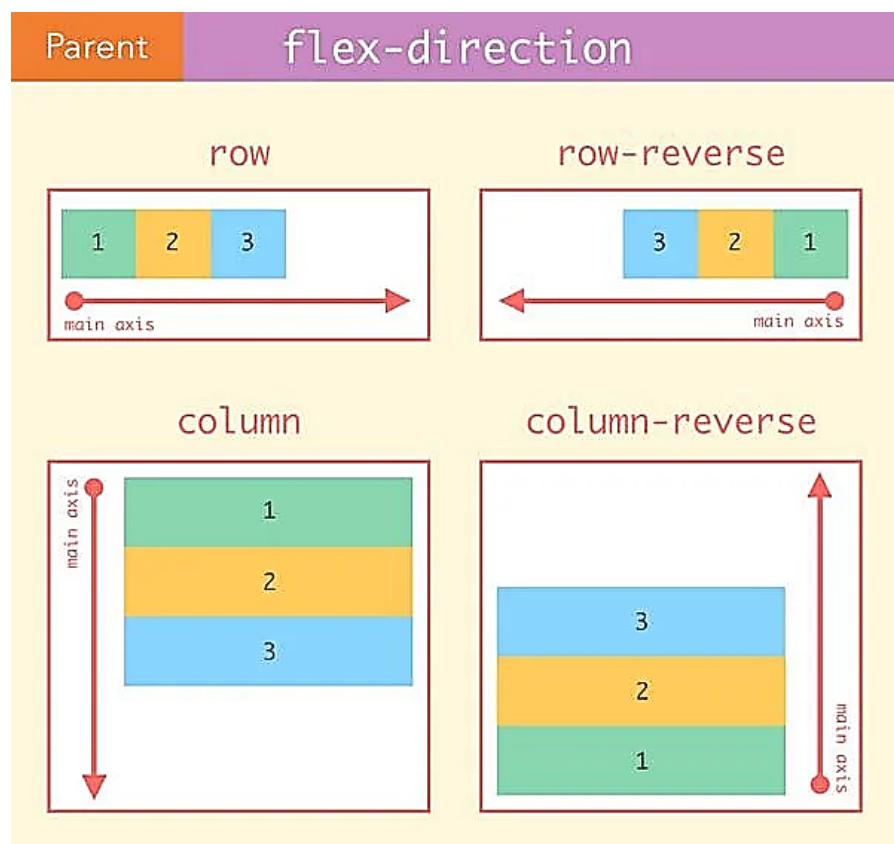
Simplemente añadiendo la propiedad **`display:flex`** podremos ver que, de manera automática, los elementos ajustan su anchura a su contenido y flotan a la izquierda.

FLEX-DIRECTION

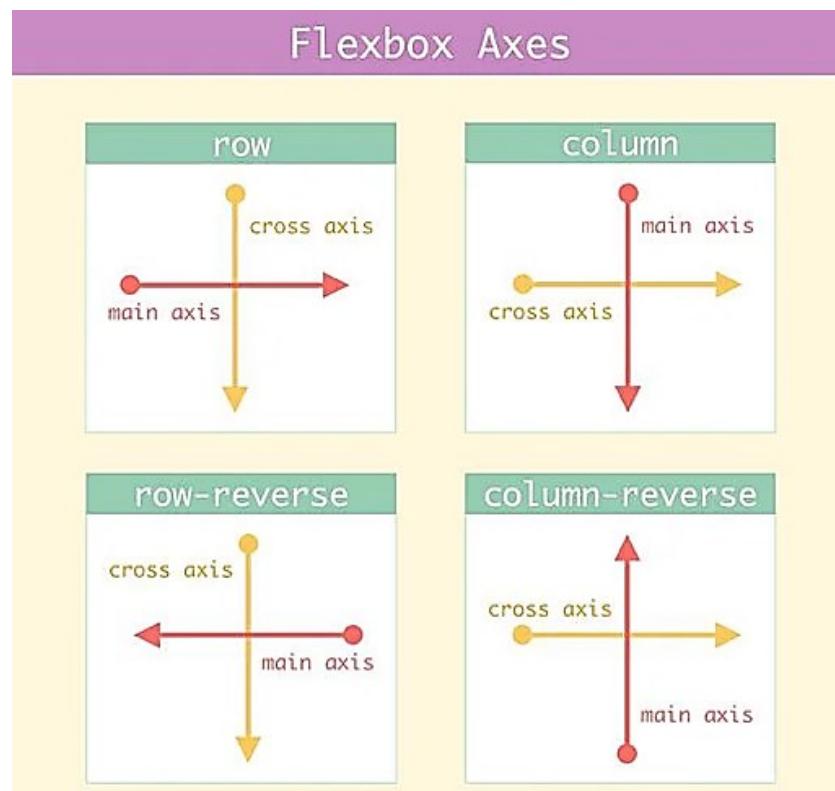
La propiedad **flex-direction** define cómo flotan los hijos:

```
.parent {  
    flex-direction: row /* default */  
    or row-reverse  
    or column  
    or column-reverse  
}
```

- **row**: es la opción por defecto y ajustará los elementos flexibles de izquierda a derecha. Esta opción sería análoga a flotación izquierda.
- **row-reverse**: igual que la anterior, pero de derecha a izquierda. Esta opción sería análoga a flotación derecha.
- **column**: ajustará los elementos flexibles en columna, de arriba a abajo.
- **column-reverse**: igual que la de arriba, pero de abajo a arriba.



En función de la dirección elegida, tendremos un eje principal, que es el que va en consonancia con la dirección y uno secundario:

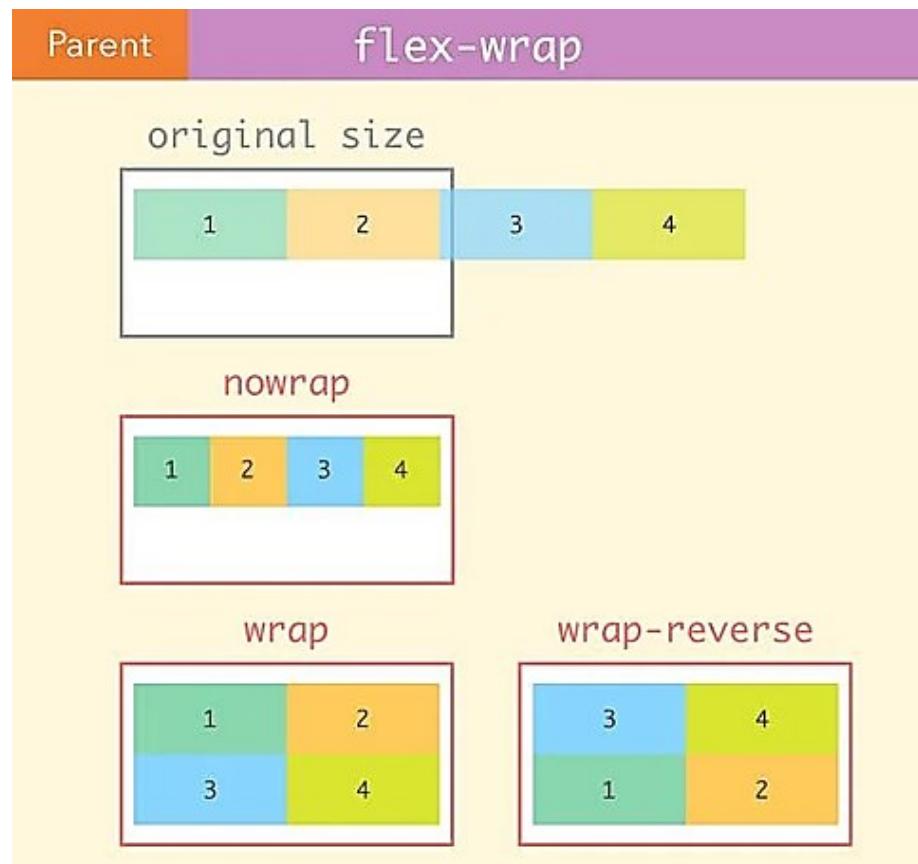
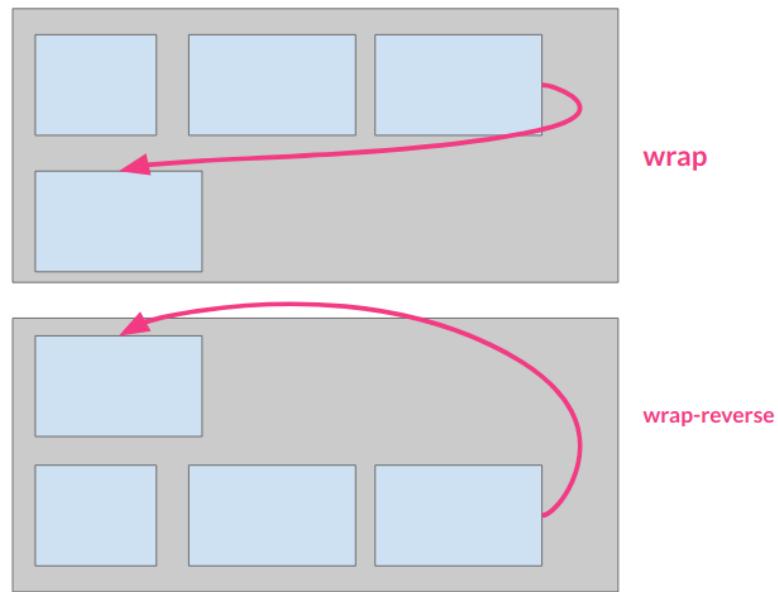


FLEX-WRAP

La propiedad **flex-wrap** indica si al llenarse una fila o columna, los elementos saltan a la siguiente:

```
.parent {  
  flex-wrap: nowrap /* default */;  
  or wrap  
  or wrap-reverse  
}
```

- **no-wrap:** es el valor por defecto y fuerza para que siempre los elementos estén en la misma línea, aunque esto suponga que se salgan del contenedor (les haya dado o no les haya dado anchura).
- **wrap:** provoca un salto de línea si la anchura de los elementos (fijada por nosotros o por el contenedor) es superior a la del contenedor.
- **wrap-reverse:** igual que la anterior, pero de abajo a arriba.



FLEX-FLOW

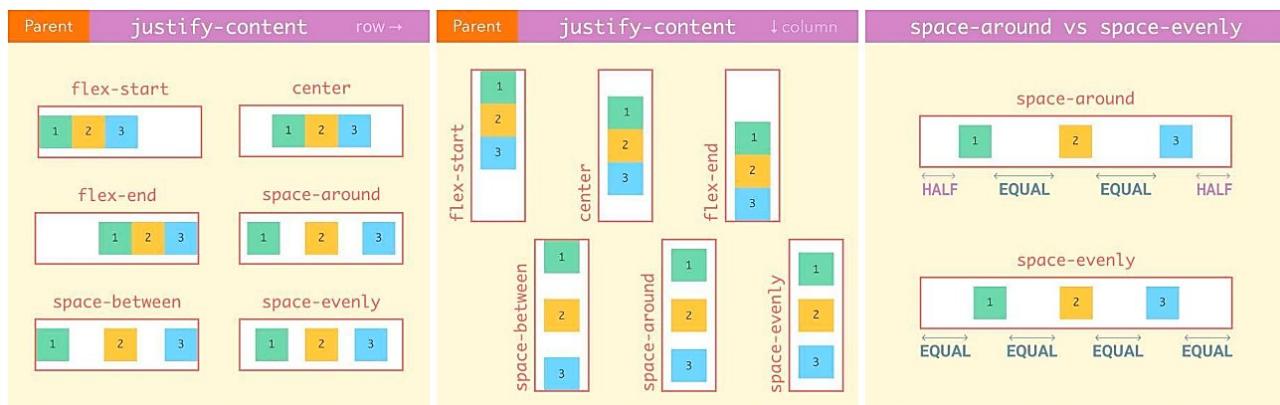
Las dos propiedades, **flex-direction** y **flex-wrap** podemos juntarlas en la propiedad **flex-flow**:

```
flex-flow: column wrap;  
/* Equivalente a:  
flex-direction: column;  
flex-wrap: wrap; */
```

JUSTIFY-CONTENT

La propiedad **justify-content** nos permite distribuir los elementos a lo largo del eje principal, en horizontal si hemos definido **flex-direction:row**, o en vertical, si tenemos **flex-direction:column**.

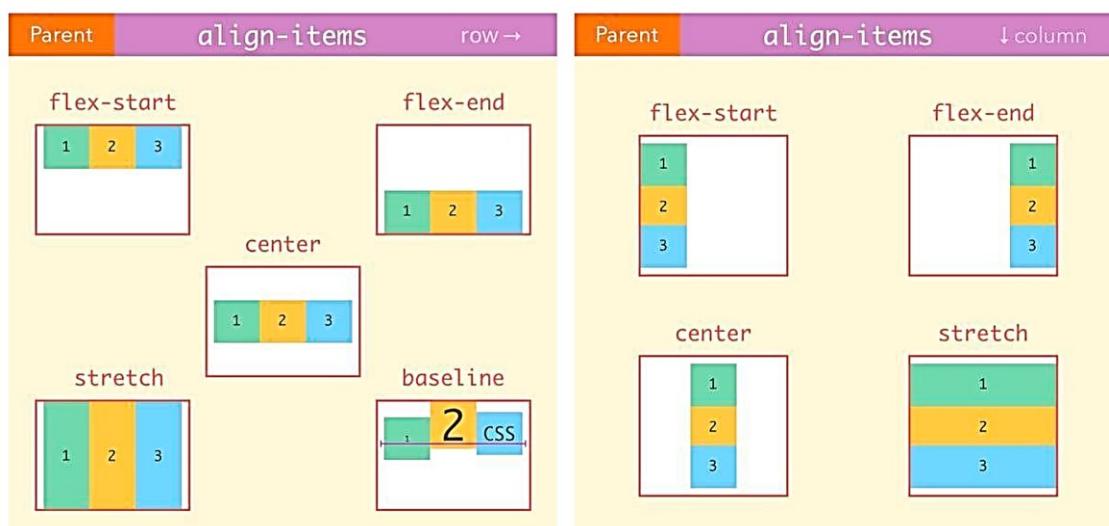
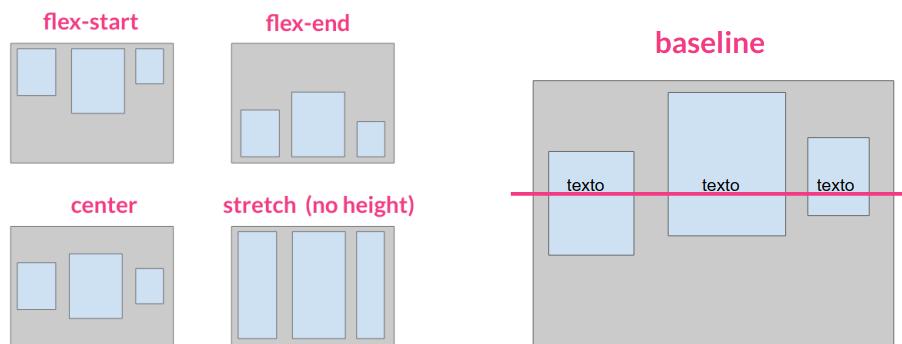
- **flex-start**: los elementos flexibles se sitúan al principio.
- **flex-end**: los elementos flexibles se sitúan al final.
- **center**: los elementos se centran en el contenedor.
- **space-between**: se coloca el primer elemento al principio del contenedor y el último al final, y reparte los demás de forma homogénea.
- **space-around**: similar al anterior, pero deja distancia con los border del contenedor (la distancia a cada borde es la mitad de un espacio entre elementos).
- **space-evenly**: similar al anterior, pero en este caso la distancia al borde cuenta como un hueco más.



ALIGN-ITEMS

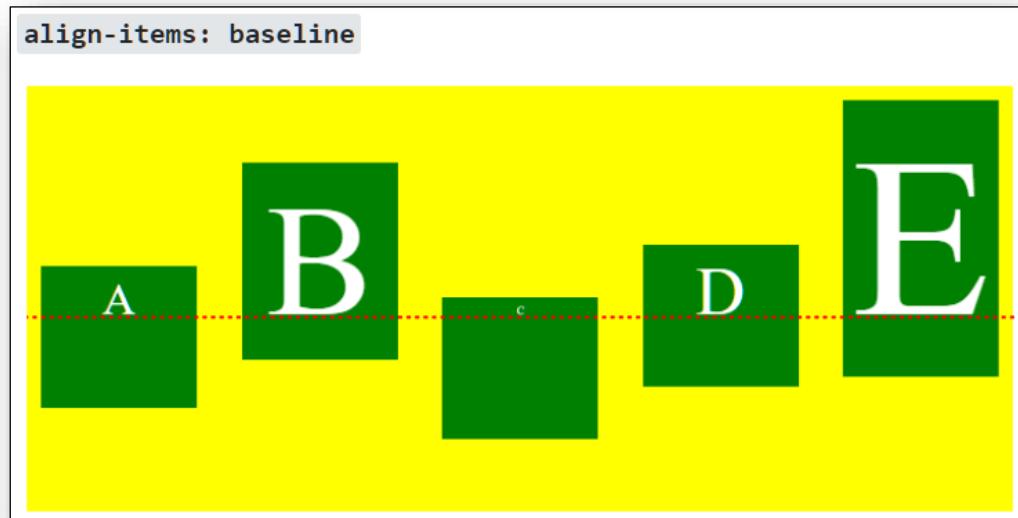
La propiedad **align-items** controla el comportamiento del eje secundario:

- **flex-start**: alinea los elementos al inicio del eje secundario.
- **flex-end**: alinea los elementos al final del eje secundario.
- **center**: alinea los elementos en el medio del eje secundario.
- **stretch**: estira los elementos para que se ajusten al eje secundario. No deben tener altura fija establecida.
- **baseline**: centra los elementos en función de su tamaño individual. Esta opción hace que los elementos se compriman al tamaño de texto y se dispongan teniendo como base la línea inferior del texto.



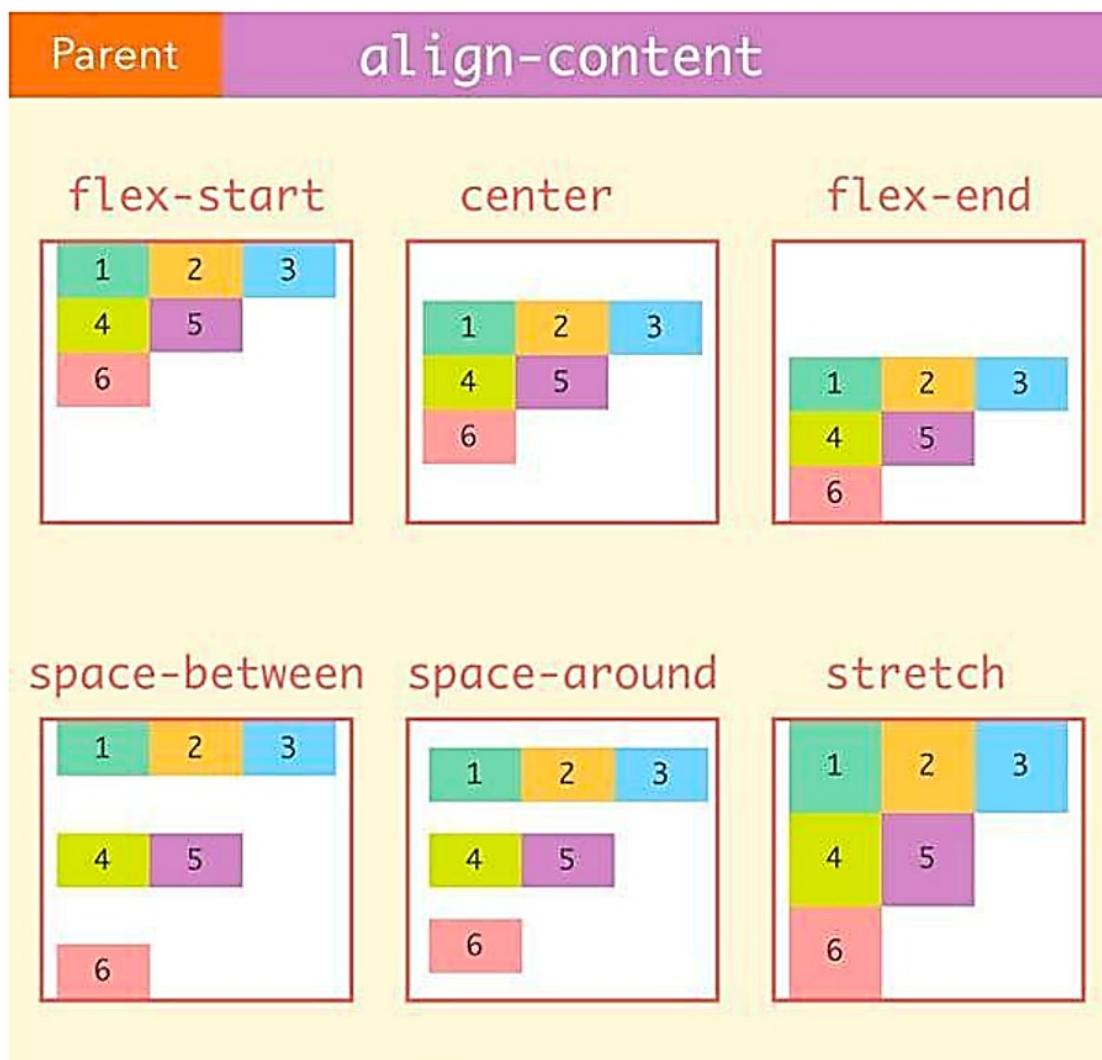
Baseline

Todos los elementos flexibles de la línea se colocan de forma que sus líneas de base se alinean con el elemento **con el mayor tamaño de texto (tamaño de fuente)**, no al elemento que es el más grande.



ALIGN-CONTENT

La propiedad **align-content** solo tiene efecto cuando se usa wrap y hay más de una línea



DIFERENCIAS ENTRE ALIGN-ITEMS Y ALIGN-CONTENT

Podrían explicarme la diferencia entre align-items y align-content

2

css css3 flexbox

Compartir Mejora esta pregunta

Seguir

Añade un comentario

editada el 21 ene. 19 a las 17:35



Shaz

28.6k • 18 • 37 • 61

formulada el 21 ene. 19 a las 17:10



Antonio U.

181 • 1 • 10

3 respuestas

Activo

Más antiguo

Puntuación

@Antonio, `align-items` alinea los elementos hijos de una misma fila entre sí mientras que `align-content` lo hace respecto del padre.

2

Fíjate que digo que alinea los elementos hijos, es decir que lo aplicas al parente.

✓ Si usamos `align-content` el alineado sólo se producirá si los elementos ocupan más de una fila. Es decir que p.e. `align-content: center` no tendrá efecto a menos que el contenido tenga mínimo 2 filas.

Si sólo usamos `align-items` y tenemos más de una fila el conjunto de filas no se centrará respecto del contenido sino que se repartirá entre el espacio disponible. Pero sí se centrarán los elementos de distinto tamaño.

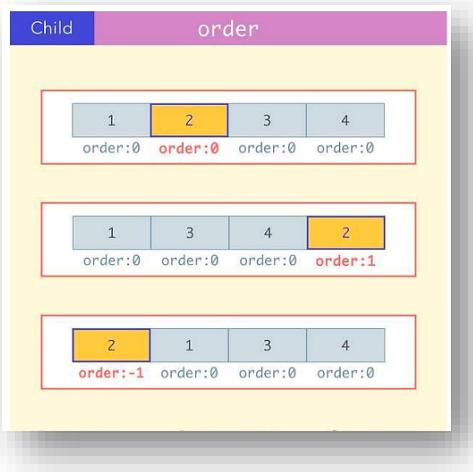
Por esa razón es recomendable usar ambos, de forma que si sólo ocupa una línea, sea `align-items` el que se encargue y si hay múltiples, se encargue `align-content`.

Propiedades para los elementos hijos

ORDER

Permite definir el orden de aparición de los hijos. Por defecto este valor es 0 y se mostrarán primeros aquellos elementos que tenga un mayor orden. En caso de empate se muestra antes el que primero estuviera en el código.

Si queremos modificar esto debemos añadir la propiedad CSS **order** a los elementos cuyo orden queremos modificar.



FLEX-GROW: CRECIMIENTO EN ESPACIO SOBRANTE DEL CONTENEDOR

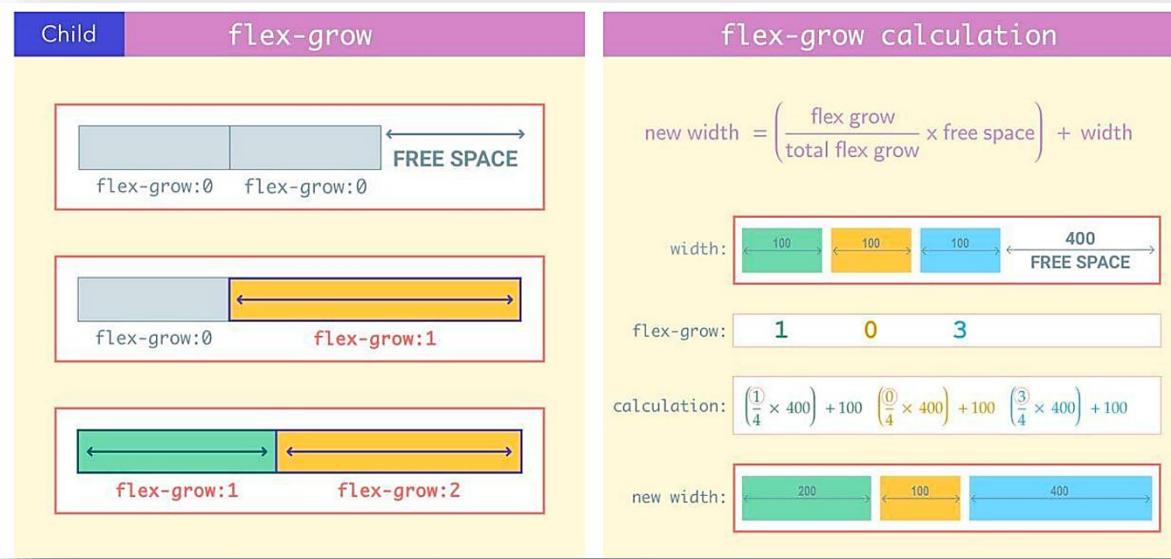
La propiedad **flex-grow** en CSS especifica cuánto debe crecer un elemento flexible en relación con el resto de los elementos flexibles en el contenedor **cuando hay espacio extra disponible**. Es decir, en caso de que la suma del tamaño de los hijos sea menor al tamaño del padre.

El valor por defecto de **flex-grow** es **0**, lo que significa que el elemento **no crecerá** más allá de su tamaño original para llenar el espacio extra en el contenedor.

Si todos los elementos en un contenedor tienen el mismo valor de **flex-grow**, todos crecerán a la misma tasa para llenar el espacio extra. Por ejemplo, si tienes tres elementos con **flex-grow: 1**, cada uno crecerá para ocupar un tercio del espacio extra.

Si un elemento tiene un valor de **flex-grow** mayor que los demás, tomará una mayor proporción del espacio extra. Por ejemplo, si tienes dos elementos y uno tiene **flex-grow: 1** y el otro tiene **flex-grow: 2**, el segundo elemento tomará dos tercios del espacio extra, mientras que el primero tomará un tercio.

Es importante tener en cuenta que flex-grow solo tiene **efecto si hay espacio extra en el contenedor**. Si no hay espacio extra, los elementos no crecerán más allá de su tamaño original, independientemente del valor de flex-grow.



FLEX-SHRINK: REDUCCIÓN DE UN ELEMENTO FLEXIBLE PARA CABER EN EL CONTENEDOR

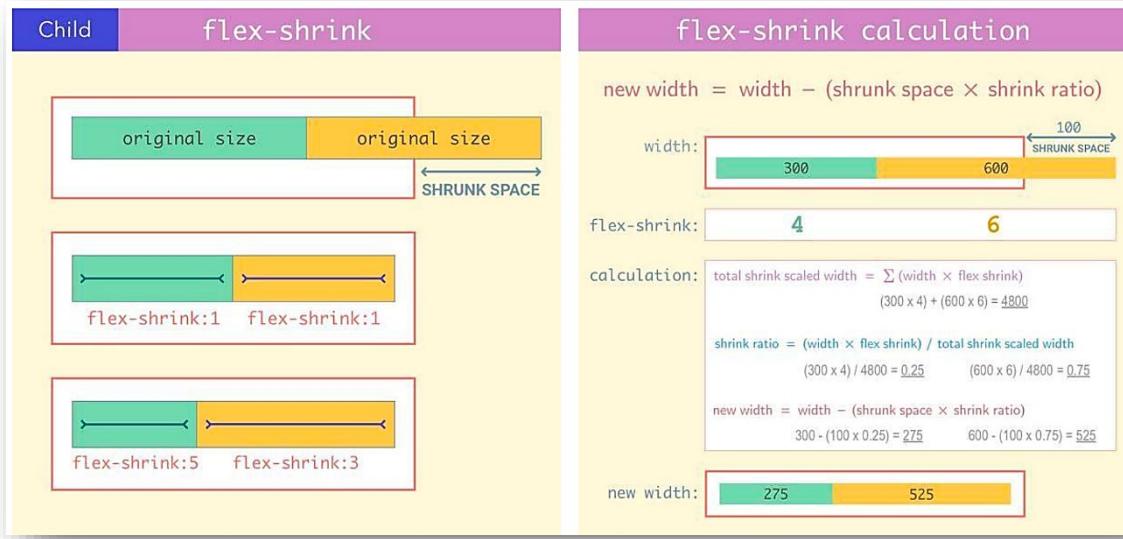
La propiedad flex-shrink especifica cómo debe reducirse un elemento flexible en relación con el resto de los elementos flexibles en el contenedor cuando no hay suficiente espacio disponible. En caso de que la suma del tamaño de los hijos sea mayor al tamaño del padre, los hijos deberán reducir su tamaño.

El valor por defecto de **flex-shrink** es 1, lo que significa que el elemento se reducirá a un tamaño que permita a todos los elementos del contenedor caber en la línea. Si todos los elementos en un contenedor tienen el mismo valor de **flex-shrink**, todos se reducirán a la misma tasa para caber en el contenedor.

Si un elemento tiene un valor de **flex-shrink** mayor que los demás, se reducirá a un tamaño más pequeño que los demás. Por ejemplo, si tienes dos elementos y uno tiene flex-shrink: 1 y el otro tiene flex-shrink: 2, el segundo elemento se reducirá el doble que el primero.

Es importante tener en cuenta que flex-shrink **solo tiene efecto si los elementos no caben en el contenedor**. Si hay suficiente espacio, los elementos no se reducirán, independientemente del valor de flex-shrink.

Por defecto vale 1, todos los elementos se reducen por igual.



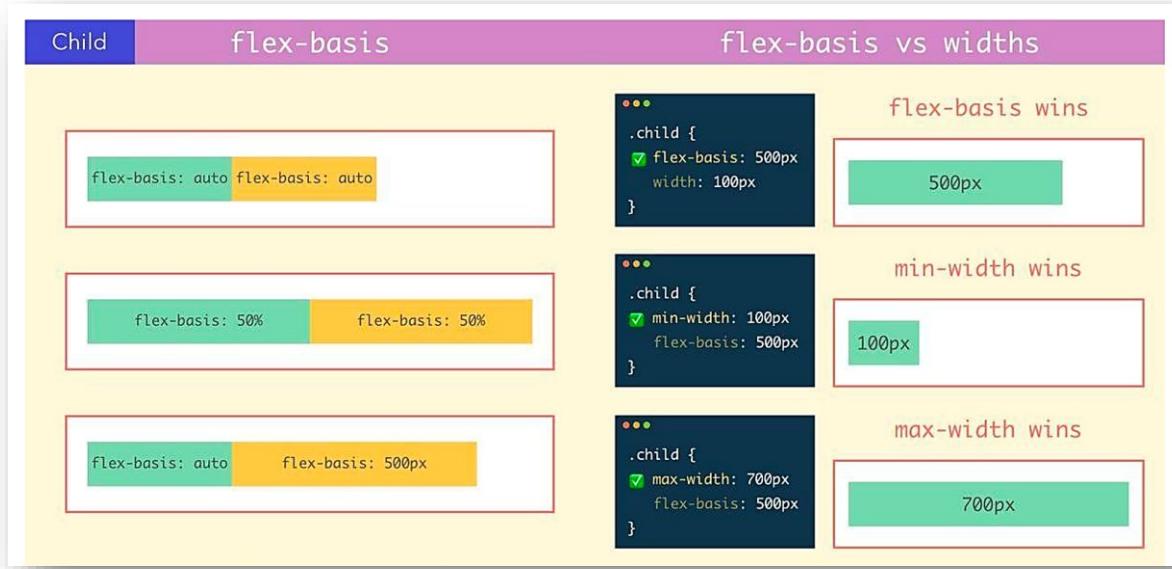
FLEX-BASIS

La propiedad **flex-basis** es el tamaño inicial de un elemento flexible **antes** de que se distribuya el espacio restante según las propiedades **flex-grow** y **flex-shrink**.

El valor por defecto **de flex-basis es auto**, lo que significa que el navegador buscará otras propiedades de tamaño (como width, height, max-width, max-height) para determinar el tamaño inicial. Si ninguna de estas propiedades está establecida, entonces el tamaño inicial será el tamaño del contenido del elemento.

Puedes establecer flex-basis a cualquier valor de longitud válido, como 'px', 'em', '%', etc. Por ejemplo, si estableces flex-basis: 200px, el elemento tendrá inicialmente 200 píxeles de ancho antes de que se apliquen flex-grow y flex-shrink.

Es importante tener en cuenta que 'flex-basis' es solo el punto de partida para el tamaño del elemento. 'flex-grow' y 'flex-shrink' luego ajustan este tamaño en función del espacio disponible en el contenedor.



Ojo, **min-width** y **max-width** tienen preferencia sobre **flex-basis**.

En un contexto de diseño flexible, `flex-basis` tiene prioridad sobre `width` o `height`, pero `min-width`, `min-height`, `max-width` y `max-height` pueden anular `flex-basis`.

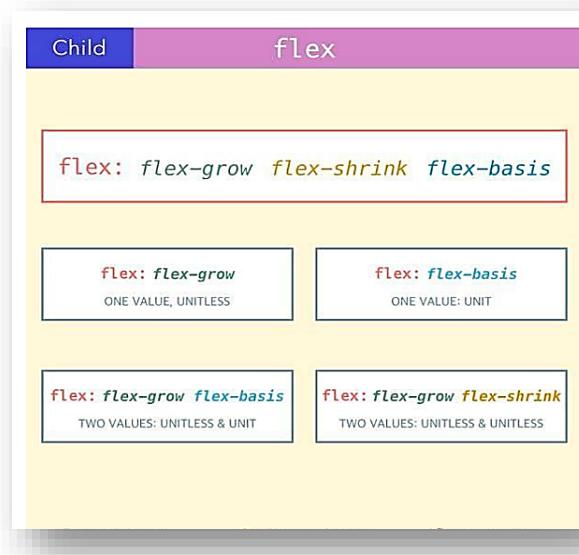
Orden de prioridad:

- min-width** y **min-height**: Estos tienen la prioridad más alta. Independientemente del valor de `flex-basis`, un elemento no se reducirá por debajo de su `in-width` o `min-height`.
- max-width** y **max-height**: Estos anulan `flex-basis` si el valor de `flex-basis` es mayor que el valor de `max-width` o `max-height`. En otras palabras, un elemento no crecerá más allá de su `max-width` o `max-height`, incluso si `flex-basis` es mayor.
- flex-basis**: Este tiene prioridad sobre `width` y `height` en un contexto de diseño flexible. Define el tamaño inicial del elemento antes de que se apliquen `flex-grow` y `flex-shrink`.

Por lo tanto, si estás trabajando con un diseño flexible y quieres asegurarte de que un elemento tenga un tamaño específico independientemente de las reglas de flexibilidad, debes usar `min-width`, `min-height`, `max-width` y/o `max-height`.

FLEX

Es una propiedad para agrupar las 3 vistas anteriormente, en ese orden:



Por defecto los hijos tienen

```
.child {  
    flex-grow: 0;          /* No crece  
    flex-shrink: 1;        /* Si es necesario se encoje como todos  
    flex-basis: auto;     /* parte de sus propiedades width height  
}
```

Notación completa

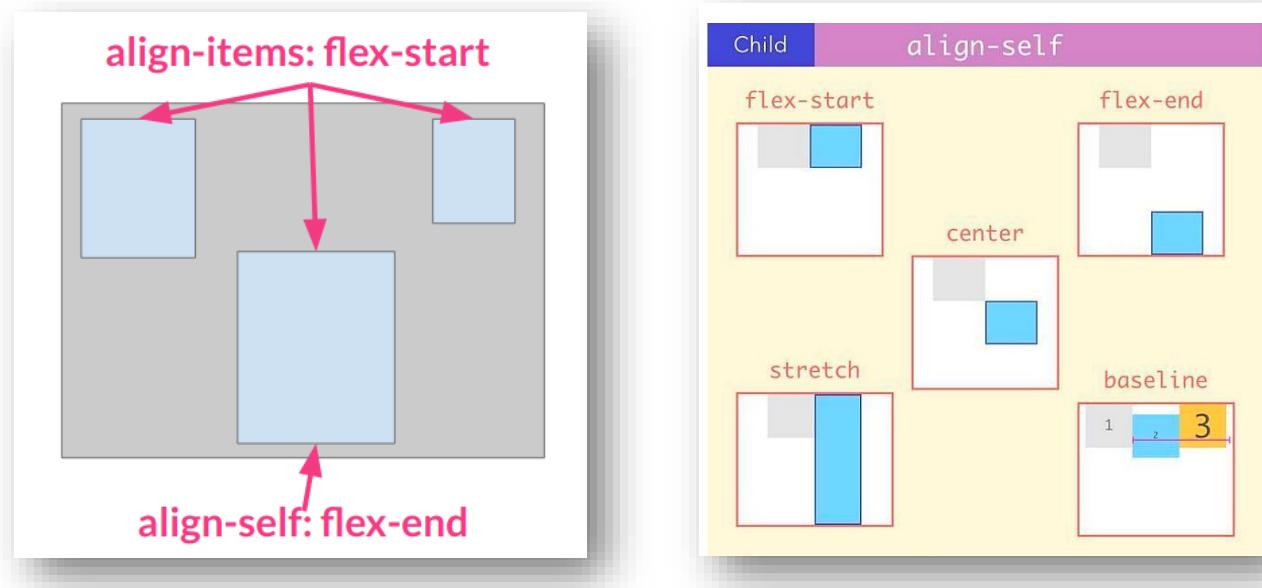
```
.container {  
    display: flex;  
}  
  
.item1 {  
    flex-grow: 2;  
    flex-shrink: 1;  
    flex-basis: 200px;  
}  
  
.item2 {  
    flex-grow: 1;  
    flex-shrink: 1;  
    flex-basis: 100px;  
}
```

Notación abreviada

```
.container {  
    display: flex;  
}  
  
.item1 {  
    flex: 2 1 200px;  
}  
  
.item2 {  
    flex: 1 1 100px;  
}
```

ALIGN-SELF

En ocasiones puedo necesitar que un elemento flexible tenga una alineación diferente al resto. En este caso, en el elemento para el que quiero una alineación diferente, debo añadir la propiedad **align-self**.



PARA PRACTICAR

Para poder practicar tenemos disponible la siguiente web:

<https://codepen.io/Marisol-FP/pen/NWvjoKE>

CUSTOM PROPERTIES O PROPIEDADES PERSONALIZADAS

Las CSS Custom Properties (muchas veces conocidas por variables CSS) son un mecanismo de CSS que permite dar un valor personalizado a las propiedades. El objetivo principal suele ser evitar escribir múltiples veces ese valor, y en su lugar, ponerle un nombre más lógico y fácil de recordar, que hará referencia al valor real.

DEFINIR UNA CUSTOM PROPERTY O VARIABLE CSS

Para definir una custom property haremos uso de los dos guiones -- previos al nombre que queramos utilizar. Además, debemos fijarnos en el elemento que definimos la variable, en este ejemplo la pseudoclase :root

```
:root {  
    --fondoColor: black ;  
}
```

En primer lugar, la **pseudoclase :root** hace referencia al elemento raíz del documento, o lo que es lo mismo, al elemento <html>. La diferencia de utilizar html o :root como selector es que este último tiene algo **más de especificidad CSS**. Mientras que html tiene 001, :root tendría 010.

Al colocarla en :root estamos definiendo que la custom property estará definida para el ámbito de esa etiqueta <html> (o cualquier elemento hijo), es decir, a todo el documento.

UTILIZAR UNA CUSTOM PROPERTY (VARIABLE CSS)

A la hora de utilizar una custom property, hay que utilizarla dentro de la expresión **var()**:

```
.element {  
    background-color: var(--fondoColor, blue);  
}
```

En este caso estamos aplicando a la propiedad background el valor que contiene --fondoColor para el elemento .element.

Esto último es muy importante entenderlo, ya que una custom property puede tener diferentes valores dependiendo del ámbito en el que se encuentra.

Además, es muy recomendable que la **expresión var() tenga dos parámetros**:

- El primero de ellos, la custom property en cuestión,
- El segundo de ellos, el valor por defecto en el caso de que esa propiedad no esté definida en el ámbito actual.

En nuestro caso, el elemento con clase element tendrá siempre el color de fondo negro, pero podría adoptar el color azul si la custom property no se hubiera declarado.

ÁMBITO DE LAS CUSTOM PROPERTIES

```
<div class="parent">
    <div class="first child">First child</div>
    <div class="second child">Second child</div>
</div>
<div class="third child">Third child</div>
<style>
    .parent {
        --fondoColor: black;
        color: white;
    }
    .first {
        --fondoColor: purple;
    }
    .child {
        background: var(--fondoColor, blue);
    }
</style>
```

Estamos definiendo la variable --fondoColor en diferentes ámbitos:

- Los dos primeros elementos .child tomarán color negro, ya que se le aplica a .parent (e hijos).
- El primer elemento .child se sobrescribe con color púrpura, ya que se le aplica a .first
- El tercer elemento no tendrá ninguna variable definida, por lo que tomará color azul.

Observa el marcado HTML del siguiente ejemplo, donde tenemos tres elementos con clase child, los dos primeros dentro de parent y el tercero fuera:



First child
Second child
Third child

PROPIEDAD Z-INDEX

El navegador dispone los elementos en una página de acuerdo con un flujo natural. Cuando los elementos se superponen en la página, el navegador tiene que decidir qué elemento colocar encima de los demás (cuál está más cerca del espectador), es decir, qué elemento recibe un orden **z mayor**.

z-index es la propiedad que nos permite colocar los elementos unos encima de otros cuando hay superposiciones o solapamientos.

SOLAPAMIENTO DE CAJAS EN CSS

Pero recuerda que por el modelo de cajas de CSS, por definición, los elementos del HTML ocupan su lugar y fuerzan al resto a que lo respeten. Así que para haya solapamientos hay que modificar dicho comportamiento desplazando alguno de ellos.

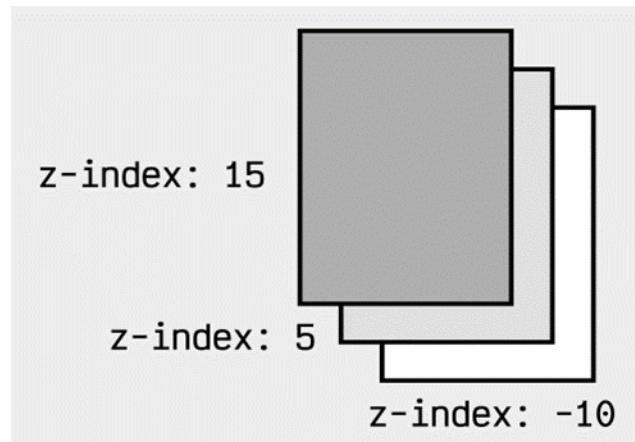
Cuando se producen solapamientos:

- **Al posicionar los elementos** (`position`) con las propiedades position puede suceder que nos encontremos que haya elementos que lleguen a solaparse por tener que ocupar la misma área.
- **Los elementos flotantes** son aquellos que se le asigna la propiedad `float`. Esta propiedad “rompe” el flujo en bloque y los elementos flotan en línea. Las cajas flotantes **no** se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Sin embargo, con elementos flotantes se puede producir superposición o solapamiento con otros elementos que NO estén flotados, ya que una caja flotante NO RESPETA a un elemento que no esté flotado.

Con z-index podemos establecer capas decidiendo el orden de solapamiento de estos elementos.

Lo que dicen las especificaciones es muy sencillo, **z-index** se aplica a cualquier elemento posicionado `position≠static`, los valores posibles admitidos son **auto** |**número entero**. La propiedad **z-index no se hereda**.

Se mostrará arriba del todo aquel elemento de los que se solapan con el mayor valor de **z-index dentro del mismo contexto de apilamiento**.



CONTEXTO DE APIALAMIENTO

Ciertas propiedades especiales que causan que formen un contexto de apilamiento en un elemento contenedor CSS:

1. Un el **elemento HTML posicionado con valor z-index crea una "caja" o contexto de apilamiento** para él y todo su contenido. Así que sus hijos, a efectos del apilamiento, estarán en el mismo nivel de z-index. Por muy elevado que sea el valor en negativo que le demos a los hijos, no salen de la caja de apilamiento de su padre. Si para el padre z-index vale 10, aunque le asignes un valor de -50 a sus hijos, estos hijos **siempre estarán encima de otros elementos hermanos de su padre** que tengan un valor menor de 10.
2. Si queremos que los hijos se coloquen bajo su padre ¿qué debemos hacer? Sencillo, **no declarar z-index en el padre**. Y no declarar es no declarar. Si damos un valor cualquiera, incluido el 0, ya crea la caja de apilamiento y coloca a sus hijos dentro.
3. Y si algún ancestro ya tiene z-index, por ejemplo, el caso que el abuelo tenga declarado un z-index y queramos que los nietos de éste estén debajo de su padre, **basta que al padre le declares position: static y los hijos con z-index negativo se colocarán debajo del padre.**

4. También crean cajas de apilamiento las siguientes características:
 - a. **Elementos que tiene posición fixed o sticky.** Los elementos posicionados como fixed o sticky se apilan sobre los elementos normales en la página, incluso si tienen un valor z-index menor.
 - b. **Elementos html con un valor opacity menor de 1** (con opacidad parcial) o **con un valor transform distinto de "none"**. Los hijos de estos elementos, así como los propios elementos, se apilarán de manera diferente en relación con otros elementos en la página.
 - c. Elementos html que sean hijos de un elemento Flexbox o Grid que tenga un z-index aplicado (diferente a auto).
5. Contextos de apilamiento pueden ser contenidos en otros contextos de apilamiento, y juntos crean una jerarquía de contextos de apilamiento
6. Cada contexto de apilamiento es completamente independiente de sus hermanos.

RESUMEN

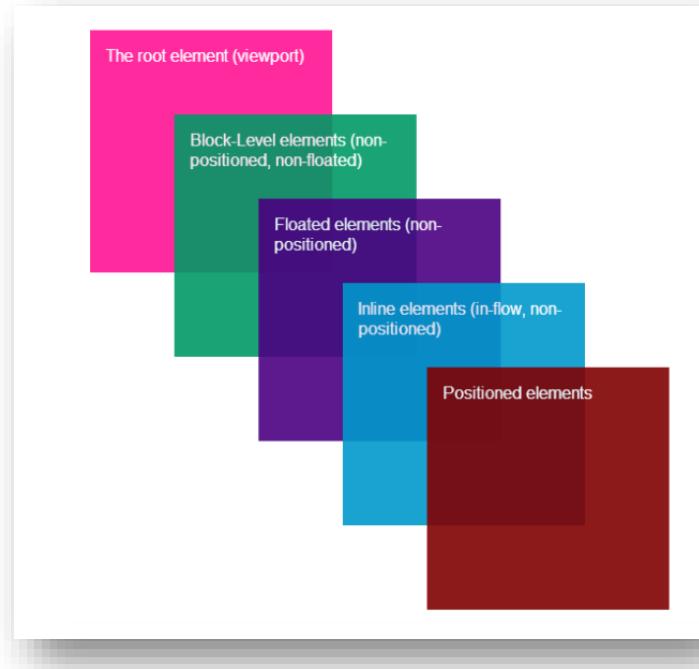
- **Posicionar y asignar un valor z-index a un elemento HTML** crea un contexto de apilamiento.
- Un contexto de apilamiento también se crea al asignar **una opacidad menor que 1** o al establecer la propiedad **transform distinta de none**.
- Cuando se forma un nuevo contexto de apilamiento lo que ocurre es que todos los elementos hijos del elemento que lo ha formado pasan a gestionarse con él, en grupo, y **su propiedad z-index ya no tiene efecto sobre los demás elementos fuera de este contexto**.
- La posición en z-index de sus hijos sólo tendrá influencia sobre otros elementos del mismo contexto de apilamiento.
- **z-index no afecta a elementos flotados.** Si necesitas controlar el orden de apilamiento de elementos flotados, deberás utilizar técnicas alternativas para la maquetación.

INTERFERENCIA DE LOS CONTEXTOS DE APIALAMIENTO

Si hay más de un elemento que forma su propio contexto de apilamiento en una página, **en caso de que estos elementos interfieran entre sí**, el orden de visualización es el siguiente:

1. Abajo de todo de la pila, en el nivel 0, **se visualiza el elemento raíz**, que forma su propio contexto de apilamiento, el global. Lo que se ven son **los bordes y el fondo del elemento raíz de la página (el elemento <html>)**. Es importante señalar que, aunque le pongas a algún elemento hijo del raíz un z-index negativo, no podrá estar nunca por debajo del elemento raíz.
2. Justo encima se visualizan los **elementos posicionados con un z-index negativo**. Estos no pueden estar por debajo del raíz, pero sí por debajo de otros elementos.
3. A continuación **se colocan los elementos no posicionados**, es decir, los que tienen **position: static**, que es el valor por defecto, en el orden de aparición en el código fuente (los que están definidos más tarde, se ven por encima en caso de interferencia).
4. Los **bloques flotantes** son colocados entre bloques no posicionados y bloques posicionados.
5. Luego se colocan los **elementos posicionados de diversas formas (absoluta, relativa...)** que **no tienen un z-index especificado** (lo tienen en auto).
6. Finalmente se colocan los **elementos posicionados que además tienen un z-index positivo**.

Este es el orden de apilamiento por defecto que aplica el navegador cuando renderiza los elementos de la página.



Cuando la página contiene elementos flotantes, elementos absolutamente posicionados, elementos fijos o elementos relativamente posicionados (elementos desplazados de su posición normal en una cierta cantidad), junto con elementos en línea, el navegador los dispone de una manera diferente:



ANEXO: SELECTORES CSS

Selector	Ejemplo	Descripción del ejemplo
.class	.intro	Selecciona todos los elementos de con class="intro"
.class1.class2	.name1.name2	Selecciona todos los elementos con el atributo clase <i>name1</i> y <i>name2</i>
.class1 .class2	.name1 .name2	Selecciona todos los elementos de la clase <i>name2</i> que son hijos o están dentro de un elemento de la clase <i>name1</i>
#id	#firstname	Selecciona el elemento con id="firstname"
*	*	Selecciona todos los elementos Selector universal
elemento	p	Selecciona todos los elementos <p>
elemento.class	p.intro	Selecciona todos los elementos <p> de la clase class="intro"
elemento,elemento	div, p	Selecciona todos los elementos<div> y todos los <p>
elemento elemento	div p	Selecciona todos los elementos <p> dentro de un elemento <div>
elemento>elemento	div > p	Selecciona todos los elementos <p> cuyo padre es un elemento <div>
elemento+elemento	div + p	Selecciona el primer elemento <p> que está justo después de un elemento <div> (p que es el hermano siguiente a un div)
elemento1~elemento2	p ~ ul	Selecciona cada elemento que es precedido por un elemento <p>
[attribute]	[target]	Selecciona todos los elementos con el atributo target
[attribute=value]	[target=_blank]	Selecciona todos los elementos con el atributo target="_blank"
[attribute~=value]	[title~=flower]	Selecciona todos los elementos con un atributo a title que contienen la palabra "flower"
[attribute =value]	[lang =en]	Selecciona todos los elementos con un atributo lang con un valor igual a "en" o que empiece por "en-"
[attribute^=value]	a[href^="https"]	Selecciona cada elemento <a> cuyo atributo href empieza por "https"
[attribute\$=value]	a[href\$=".pdf"]	Selecciona cada elemento <a> cuyo atributo href termine por ".pdf"
[attribute*=value]	a[href*="W3C"]	Selecciona cada elemento <a> cuyo atributo href contenga la subcadena "W3C"
:active	a:active	Selecciona los enlaces activos
::after	p::after	Inserta algo después del contenido de cada elemento <p>
::before	p::before	Inserta algo antes del contenido de cada elemento <p>
:checked	input:checked	Selecciona cada elemento <input> marcado

:default	input:default	Selecciona el elemento <input> por defecto
:disabled	input:disabled	Selecciona cada elemento <input> desactivado
:empty	p:empty	Selecciona cada elemento <p> que no tiene hijos (incluyendo nodos texto)
:enabled	input:enabled	Selecciona cada elemento <input> enabled (activado)
:first-child	p:first-child	Selecciona cada elemento <p> que es el primer hijo de su padre
::first-letter	p::first-letter	Selecciona la primera letra de cada elemento <p>
::first-line	p::first-line	Selecciona la primera línea de cada elemento <p> e
:first-of-type	p:first-of-type	Selecciona cada elemento <p> que es el primer <p> de su padre
:focus	input:focus	Selecciona cada elemento input el cual tiene el foco
:fullscreen	:fullscreen	Selecciona el elemento que está en modo full-screen
:hover	a:hover	Selecciona los enlaces que tienen el ratón encima
:in-range	input:in-range	Selecciona el elemento input con un valor dentro de un rango específico
:indeterminate	input:indeterminate	Selecciona el elemento input que está en un estado indeterminado
:invalid	input:invalid	Selecciona todos los elementos input con un valor invalido
:lang(<i>language</i>)	p:lang(it)	Seleccionada cada elemento <p> con un valor del atributo lang igual a "it" (Italiano)
:last-child	p:last-child	Selecciona cada elemento <p> que es el último hijo de su padre
:last-of-type	p:last-of-type	Selecciona cada elemento <p> que es el último elemento <p> de su padre
:link	a:link	Selecciona todos los enlaces no visitados
::marker	::marker	Selecciona todos los list ítems que están marcados
:not(<i>selector</i>)	:not(p)	Selecciona cada elemento que no es un elemento <p>
:nth-child(n)	p:nth-child(2)	Selecciona cada elemento <p> que es el segundo hijo de su padre
:nth-last-child(n)	p:nth-last-child(2)	Selecciona todos los elementos <p> que son el segundo hijo de su padre, contando desde el último hijo.
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selecciona cada elemento <p> que es el segundo elemento <p> de su padre, contando desde el último hijo
:nth-of-type(n)	p:nth-of-type(2)	Selecciona cada elemento <p> que es el segundo elemento <p> de su padre

:only-of-type	p:only-of-type	Selecciona cada elemento <p> que es el único elemento <p> de su padre
:only-child	p:only-child	Selecciona cada elemento <p> que es el único hijo de su padre
:optional	input:optional	Selecciona cada elemento input con el atributo no "required"
:out-of-range	input:out-of-range	Selecciona cada elemento input con un valor fuera de un rango especificado
::placeholder	input::placeholder	Selecciona cada elemento input con un atributo "placeholder"
:read-only	input:read-only	Selecciona los elementos input con el atributo "readonly"
:read-write	input:read-write	Selecciona los elementos input con el atributo "readonly" NO especificado
:required	input:required	Selecciona los elementos input con el atributo "required"
:root	:root	Selecciona el elemento root del documento, es idéntico al selector <html> excepto porque tiene mayor especificidad.
::selection	::selection	Selecciona la porción de un elemento que está seleccionada por el usuario
:target	#news:target	Selecciona el actual elemento #news activo (pinchado en una URL)
:valid	input:valid	Selecciona todos los elementos input con un valor válido
:visited	a:visited	Selecciona todos los enlaces visitados