
# FULL STACK JAVA
# Clase

CSS (2)

BA

# Temas

Selectores
Modelo de caja: Padding, Border y Margin
Box Sizing
Position: Absolute, Relative, Fixed, Index-z, Float

# Selectores

# Selectores

Podemos pasar propiedades a diferentes etiquetas HTML, también sabemos que los elementos hijos de un padre van a heredar cualidades de él.

Ahora veremos que podemos pasar atributos a determinadas etiquetas sin necesidad de construir infinitas clases, y aprovechar las jerarquías y tags para lograr esta personalización.

# Selectores descendientes

Podemos asociar una clase (o tag) según su jerarquía, por ejemplo:

Aquí podemos dar propiedades al div que tiene la clase ‘contenedor3’, y decirle a la página que estas propiedades sólo lo afecten a él, que si existiera otro div fuera de ‘contenedor1’ que tenga la clase ‘contenedor3’ no tome las propiedades de este contenedor.

```
<div class="contenedor1">
    Lorem ipsum dolor sit amet consectetur
    corporis
    neque, vel sed eveniet, possimus quasi
    eos.
    <div class="contenedor2">
        Lorem ipsum dolor sit amet consecte
        sunt
        fuga aspernatur modi consectetur fa
        doloribus a
        nostrum iure?
        <div class="contenedor3">
            Lorem ipsum dolor sit amet cons
            asperiores
            sunt
            fuga aspernatur modi consectetu
            consequatur,
            doloribus a
            nostrum iure?
        </div>
    </div>
</div>
```

# Selectores descendientes

Esto se puede hacer con selectores descendientes, éstos restringirán los atributos que le demos a una clase o etiqueta, y hará respetar una jerarquía..

**Espacio**: En el primer caso vemos que .contenedor1 tiene a su lado a .contenedor3 separado por un espacio, esto le indica a la página que toda etiqueta que esté dentro de .contenedor1 y que tenga como atributo de clase a .contenedor3 tendrá color 'white' y como background color a dicho color hexadecimal. No importa que sea exactamente un hijo, sólo debe estar dentro de .contenedor1.

# Selectores descendientes: de hijos

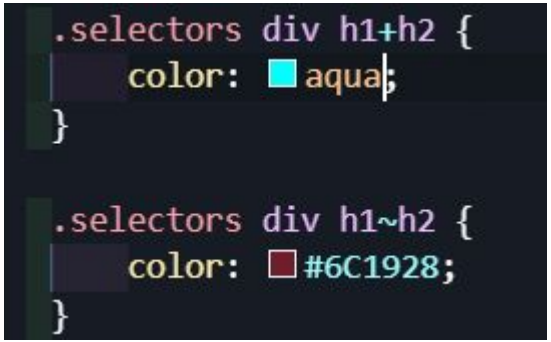
**Señalador**: Este selector hace la misma relación que el selector anterior, pero su función es más estricta: Aquí el tag con la clase .contenedor3  sí debe ser hijo directo de .contenedor1, en nuestro ejemplo el atributo 'bold' nunca se verá en acción, ya que .contenedor3 es hijo de .contenedor2..

# Selectores hermanos

Así como vimos los selectores descendientes también podemos hacer relación entre los propios hijos:

**Signo positivo**: De carácter estricto exige que la segunda clase (o tag) sea consecutiva del primero, en nuestro ejemplo h2 debe ser el primer hermano continuo de h1. En nuestro ejemplo el color aqua no se verá presente, porque h1 tiene como hermano al tag u.

**Virgulilla**: Es un selector hermano de carácter no estricto, por lo cual el color hexadecimal se hará presente en nuestra página ya que h2 es hermano de h1, no es el más próximo, pero es hermano al fin.



# Selector de atributo

**Corchetes**: Podemos hacer uso de este selector para condicionar a la clase, entonces sólo actuará si el atributo dentro de los corchetes existe y coincide en el tag.

Podemos utilizar sólo el atributo (sin su valor), entonces actuará en aquellos tags que contengan ese atributo.

También podemos usar virgulilla antes del =, éste indicará que la clase actuará sobre el tag que contenga (como valor del atributo) 'contenedor3', recordemos que una etiqueta puede tener más de una clase declarada (ej: class="contenedor1 contenedor2 contenedor3").

```
a[href='https://aulasvirtuales.bue.edu.ar'] {
    background-color: black;
    color: white;
    padding: 2.5px 10px;
    font-weight: 500;
    text-decoration: none;
    border-radius: 5px;
}
```

```
a[href] {
    text-decoration: none;
}

div[class~="contenedor3"] {
    text-decoration: overline;
}
```

# Modelos de Caja

## Padding, Border, Margin

# Modelos de caja

Como aprendimos en clases anteriores, los elementos que tenemos en nuestra página pueden tener características de línea (inline) o de bloque (block), unos ocupando una fracción del 'renglón' en el que están parados (como el *span*), y otros ocupan la totalidad del mismo, abarcando el ancho total de su contenedor (como un *div*).

# Modelos de caja

En CSS nos referimos a los modelos de caja con el atributo ‘display’.

Cabe destacar que los modelos de cajas tienen dos tipos de visualizaciones:

Visualizaciones externas: Cómo se manifiestan las cajas con respecto a su alrededor.

Visualizaciones internas: Cómo se manifiesta el interior de la caja.

En el caso de inline y block ya vimos cómo afectan a su entorno (de manera externa).

En el caso de flex veremos que su interior se verá afectado como si cada elemento (dentro del mismo) fuese inline, mientras que con su exterior actuaría como si tuviese la propiedad block.

Con grid podremos armar grillas, una cuadrícula de elementos que por defecto nos dará su contenido como si tuviesen el atributo block, pero podemos dividir su contenido en partes, para poder construir una disposición personalizada de los elementos en pantalla.

```
.contenedor-inline {
    display: inline;
}

.contenedor-block {
    display: block;
}

.contenedor-flex {
    display: flex;
}

.contenedor-grid {
    display: grid;
}
```

# Modelos de caja: Flexbox y Grid

Ambas propiedades dan flexibilidad a la hora de organizar nuestro contenido.

Mientras el atributo flex genera una caja donde podemos manipular su contenido (alineación vertical y horizontal, espaciado, tamaño, orden) en grid podemos gestionar un orden completo de una cuadrícula, gestionando también espacios verticales dentro de nuestro trabajo.

Ninguna de estas dos herramientas es mejor que la otra, la que elijamos será parte de nuestro estilo, podemos llegar a los mismos fines con ambas.

# Padding, Border, Margin

Sus atributos modificarán los espacios, bordes y márgenes de su contenido. Sus valores pueden combinarse en uno solo, o separarse por valor individual.

El padding indicará la distancia que tendrá su contenido con su relleno. el margin la distancia que tendrá su contenido y relleno respecto a su contenedor, y el border hace referencia al borde de su contenido (líneas, sin bordes, grosor, redondeado).

# Box Sizing

# Box Sizing

Cuando nosotros agregamos propiedades como padding y border nuestra caja sufrirá cambios visuales, quizás no respete los tamaños que pensamos que tendría, eso se debe a que el valor por defecto de este atributo (**box-sizing**) es content-box.

# Box Sizing

**Content-box**: Lo que hace este valor es decirle al navegador que *el contenido de la caja* donde nos encontramos parados debe mantener el tamaño que nosotros le indicamos.. Si agregamos valores de padding y border éstos no deben alterar el tamaño del contenido, más no así de la caja.

**Border-box**: Con este valor lo que hacemos es bloquear el tamaño de la caja y no del contenido, por lo que los valores padding y border no modifican el tamaño de la misma, pero pueden afectar su contenido.

# Position

## Absolute, Relative, Fixed, Index-z, Float

# Position

Lo que hará el atributo Position es indicarle al navegador cómo debe tratar al bloque en el que estamos (en cuanto a su posición y disposición en pantalla):

**Absolute**: Tomará como valor el marco completo del navegador (excepto si su padre tiene declarada su position), lo tomará como referencia para ubicarlo en pantalla. Puede contar con atributos top, left, right y bottom para asignarle una distancia respecto a ese margen.

# Position

**Relative**: Tratará al bloque como lo venimos conociendo, su ubicación será relativa según el entorno en el que se encuentre (no tiene atributos top, left, bottom, right).

**Fixed**: Su ubicación se inicia como un relative (a menos que modifiquemos los atributos top, left, bottom, right), luego quedará fijo en pantalla, aunque nos movamos.

**Sticky:** Actuará como un relative, a menos que usemos los atributos top, left, bottom, right; entonces anclará en el atributo que hayamos mencionado.

# Index-z

Con estas modificaciones podemos sufrir determinados problemas, como la superposición de los elementos.

El atributo index-z nos soluciona ese problema, como su nombre lo menciona: el atributo gestiona el eje z de los elementos (profundidad), por lo que su valor será un número que puede ser positivo, neutral (cero) o negativo. Mientras mayor sea su valor su superposición estará por sobre los que tengan un valor inferior (por defecto los elementos tienen un z-index de 0).

# Float

Alineará su contenido a la izquierda o derecha del padre principalmente.

# Juegos CSS

- https://flukeout.github.io
- http://cssgridgarden.com
- http://www.flexboxdefense.com
- https://flexboxfroggy.com
- https://mastery.games/flexboxzombies
- https://cssbattle.dev

# Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

**Todo en el Aula Virtual.**