Caules son las etiquetas semántica?

* <header>: define el encabezado de pagina (no confundir con <HEAD>)
* <nav>: define una barra de navegación
* <section>: define una sección
* <footer>: define un pie de pagina
* <article>: define un articulo el cual puede tener su propio encabezado, navegación, sección, pie de pagina, etc.

Evitar el uso excesivo de <div>

Ventajas de utilizar HTML SEMANTICO:

* Ayuda a tu sitio a ser accesible
* Mejora tu posicionamiento (SEO)
* Código más claro, legible y mantenible
* Ayuda a buscadores (como Google) a encontrar tu página

/\*------------------------------------------------------------------------------------------------\*/

Selectores CSS:

/\* selector etiqta \*/

h3 {

    color: coral;

}

/\* selector de clase \*/

.titulo{

    color: red;

}

.div-bici{

    background-color: aqua;

}

/\* selector de atributo \*/

a[href='http://nosotros.com'] {

    /\* Todas las etiquetas <a> con una propiedad href con valor "https://platzi.com" \*/

}

/\* selector de id \*/

#comprar{

    color:blue

}

/\* selectores combinados \*/

/\* descendiente \*/

header nav {

    /\* Todas las etiquetas <nav> que estén dentro de una etiqueta <header> \*/

}

div p {

    /\* Todas las etiquetas <p> que estén dentro de una etiqueta <div> \*/

}

/\* hijo \*/

header > nav {

    /\* Todas las etiquetas <nav> que sean hijos directos de una etiqueta <header> \*/

}

div > p {

    /\* Todas las etiquetas <p> que sean hijos directos de una etiqueta <div> \*/

}

/\* elemento adyacente \*/

header + nav {

    /\* Todas las etiquetas <nav> que sean hermanos adyacentes de una etiqueta <header> \*/

}

div + p {

    /\* Todas las etiquetas <p> que sean hermanos adyacentes de una etiqueta <div> \*/

}

/\* hermano general \*/

header ~ nav {

    /\* Todas las etiquetas <nav> que sean hermanos generales de una etiqueta <header> \*/

}

div ~ p {

    /\* Todas las etiquetas <p> que sean hermanos generales de una etiqueta <div> \*/

}

PSEUDOCLASES:

:active

:focus

:hover

:nth-child

PSEUDOELELMENTOS:

::after

::before

/\*----------------------------------------------------------------------\*/

ATAJO agregar emoji: tecla Windows + . (punto)

/\*--------------------------------------------------------------------------------------------\*/

CSS:

DISPLAY:

Block: ocupa todo el renglón y el sgt elemento se sitará por debajo.

ES posible añadir medidas de anchura/altura.

Inline: establece q un elemnto ocupara el espacio del mismo y el sgt elemnto se situará a la derecha.

NO es posible añadir medidas de anchura/altura.

Inline-block: permite colocar medidas del modelo block, y colocar los elementos a la derecha COMO el inline.

El display flex y grid son formas de visualización de elementos recientes y cada uno tienen sus propias características para crear interfaces de manera efectiva, a partir de un contenedor padre que dotará a los elementos hijos de superpoderes del posicionamiento.

Ambas son herramientas muy útiles en el desarrollo, especialmente para la creación de interfaces amigables al usuario y aptas para cualquier dispositivo, que este último se lo conoce como [responsive design](https://platzi.com/clases/2467-frontend-developer/40845-responsive-design/).

Sin embargo, ambas herramientas tienen temas muy extensos de entender, y como mi intención no es estresarte con demasiada información, simplemente ten presente de manera general en qué consisten.

**Qué es flexbox**

*Flexbox* consiste en el **ordenamiento de elementos hijos en un solo eje**, por defecto horizontalmente. El elemento padre o contenedor deberá contener la propiedad display con el valor flex. A partir de aquí, ya puedes ordenar los hijos según sea necesario.

**Qué es grid**

*Grid* consiste en el **ordenamiento de elementos hijos en dos ejes**, como si fuera una cuadrícula o tabla. El elemento padre o contenedor deberá contener la propiedad display con el valor grid y debes definir las medidas de las columnas y de las filas. A partir de aquí, ya puedes ordenar los hijos según sea necesario.

Position:

## Posición estática

La posición static es el valor por defecto de todo elemento HTML, consiste en **respetar el flujo normal del documento donde las propiedades de posición no pueden ser establecidas**.

* [Ejemplo position static](https://bit.ly/3KjpQuN)

## Posición relative

La posición relative consiste en **respetar el flujo normal del documento donde las propiedades de posición sí pueden ser establecidas**.

* [Ejemplo position relative](https://bit.ly/3KB4Akp)

## Posición absoluta

La posición absolute consiste en **quitar al elemento del flujo normal del documento donde las propiedades de posición sí pueden ser establecidas**.

En el siguiente ejemplo, observa que pasa con el primer elemento con respecto a los demás.

* [Ejemplo position absolute](https://bit.ly/3Cy16gc)

Habrás notado que el elemento "“2"” desaparece, pero en realidad lo que sucede es que **sitúa por detrás del elemento con posición absoluta que salió del flujo normal del documento**. Este comportamiento se debe al eje Z de la pantalla y al [contexto de apilamiento](https://platzi.com/clases/2467-frontend-developer/40842-z-index-y-el-contexto-de-apilamiento/).

### Elemento padre más próximo con posición relativa

El elemento con posición absoluta se desplazará arriba, abajo, izquierda o derecha con respecto al **elemento padre más próximo con posición relativa.**

**Si no existe un padre con posición relativa de un elemento con posición absoluta, este se desplazará con respecto al elemento raíz del documento.**

En el siguiente ejemplo, te encontrarás varios contenedores padres, incluso las etiquetas <html> y <body>. Sigue los pasos y observa el comportamiento. Ignora los estilos iniciales, simplemente sirven para establecer la estructura del ejercicio.

* [Ejemplo de position en diferentes contenedores padre](https://bit.ly/3ThZRrx)

Como pudiste observar, **en el elemento con posición absoluta, su desplazamiento se basa con relación al elemento padre más próximo con posición relativa.**

## Posición fija

La posición fixed consiste en **quitar al elemento del flujo normal del documento y fijarlo en un lugar; donde las propiedades de posición sí pueden ser establecidas**.

En el siguiente ejemplo, desplázate por el documento, observa el comportamiento antes y después de colocar la posición fija.

* [Ejemplo position fixed](https://tinyurl.com/2p94fbc7)

## Posición variable fija

La posición sticky consiste en **quitar al elemento del flujo normal del documento y fijarlo en un lugar mientras su contenedor sea visible; donde las propiedades de posición sí pueden ser establecidas**.

En el siguiente ejemplo, desplázate por el documento, observa el comportamiento antes y después de colocar la posición variable fija.

* [Ejemplo de position sticky](https://tinyurl.com/4cnxu8rx)

/\*----------------------------------------------------------------------------------------------\*/

# Z-index y el contexto de apilamiento

El navegador está constituido de tres planos y ejes: **el ancho o X; el alto o Y; y el de profundidad o Z**.

## Qué es la propiedad z-index

**El contexto de apilamiento se configura con la propiedad z-index.**

Por defecto, todos los elementos tienen un valor auto, es decir, el orden está definido por la estructura del HTML. Los primeros elementos estarán detrás y los últimos estarán de frente.

Si se establece un valor positivo, este elemento se sitúa por delante de los demás. Si se establece un valor negativo, se sitúa por detrás.

Si un elemento tiene un z-index mayor a otro, estará por delante. Sin embargo, **si un elemento que tiene un z-index menor a otros, sus hijos nunca estarán por encima, aunque su z-index sea mayor**.