**CSS**

**1.INTRODUCCIÓN AL CSS**

**1.1 ¿Qué es CSS?**

CSS es un lenguaje de hojas de estilo utilizado para describir las “reglas” que definirán el aspecto de su página web. Sin CSS, todas las páginas se verían como texto sin formato e imágenes.

HTML, CSS y JavaScript son los únicos idiomas que el navegador (cliente) puede entender, por lo que es realmente importante para entender las reglas básicas de estas hojas de estilo (archivos CSS) si queremos trabajar de manera eficiente con nuestras páginas web.

**1.2 ¿Cómo instalarlo en mi proyecto?**

El css no necesita ser instalado ni preparar ningún entorno de trabajo especial para usarlo.

Para usar html solo necesitamos crear un archivo con la terminación “.css”. Este archivo puede ser ejecutado por cualquier navegador. (tanto html, como css y javascript pueden ejecutarse y son entendidos perfectamente por los navegadores, también conocidos como clientes).

Para crear este archivo nos basta con cualquier editor de textos plano, como el blog de notas (sí, con algo tan simple se puede construir un css), aunque, por regla general utilizaremos editores de textos especializados que nos permiten escribir mucho más rápido y de forma profesional (por ejemplo, Atom).

**1.3 Estructura del CSS**

Un conjunto de reglas CSS está formado por dos elementos: un selector y un bloque de declaraciones.

|  |  |
| --- | --- |
| selectors {  property: value;  property: value; } | |

**Selector(s):** Señala el elemento(s) al que vamos a aplicar la regla de estilo a.

**Bloque de declaración:** Contiene una o más declaraciones separadas por punto y coma. Estas declaraciones especifican las propiedades del elemento y los valores que queremos establecer.

**2. Insertando estilos CSS en Html**

Existen varias maneras de insertar el código CSS en el archivo HTML. Podemos insertarlos en línea, dentro de una etiqueta (estilos inline), dentro de la etiqueta head o en un archivo separado que uniremos posteriormente.

**2.1 Estilos en línea**

Insertando el atributo style dentro de una etiqueta puedes personalizar un elemento con una enorme lista de propiedades si los separa por punto y coma. Pero esta configuración no se recomienda ya que es muy desorganizada y difícil de mantener.

**Ejemplo**

El código CSS se escribe como un atributo de la etiqueta HTML.

|  |  |
| --- | --- |
| <div style="background-color: #00D1AC; width: 200px; height: 100px"></div> | |

**2.2 Estilos en la etiqueta head**

Como no podemos dejar todos esos estilos CSS en nuestras etiquetas HTML, vamos a separar el código CSS de la estructura HTML. Esta separación de la estructura + contenido semántico con el estilo nos ayudará también.

**Ejemplo**

Vamos a crear una etiqueta style. Esta etiqueta permite embeber código CSS en el HTML:

|  |  |
| --- | --- |
| <!DOCTYPE html> <html lang="en">  <head>  <meta charset="UTF-8">  <title>Document</title>  <style type="text/css">  div {  background-color: #00D1AC;  width: 200px;  height: 100px;  border: 1px solid black;  }  </style>  </head>  <body>  <div></div>  <div></div>  </body>  </html> | |

**2.3. Estilos en un archivo separado**

En realidad, la forma correcta de aplicar estilos CSS en el código HTML es tener dos archivos separados, uno de los contenidos (index.html), y uno para estilos (style.css).

Pero, ¿por qué necesitamos crear otro documento para escribir un montón de reglas? La separación de responsabilidades nos ayuda a reutilizar los estilos en diferentes páginas, por lo que puede tener varias páginas que son muy similares, a pesar de tener diferente contenido.

**Ejemplo**

Vamos a crear un documento llamado style.css (puede seleccionar un nombre diferente, pero hay que mantener la extensión .css). En este archivo, vamos a escribir el código que teníamos en nuestra etiqueta de estilo:

|  |  |
| --- | --- |
| div{  background-color: #00D1AC;  width: 200px;  height: 100px;  border: 1px solid black;  } | |

Ahora que tenemos la hoja de estilo, vamos a vincularlo al archivo HTML. Vamos a añadir un enlace de la etiqueta dentro del <head>.:

|  |  |
| --- | --- |
| <link rel="stylesheet" href="style.css"> | |

El atributo <link> en nuestro HTML indica la posición relativa del archivo CSS que vamos a agregar.

Ahora que nuestro sistema de ficheros está organizado, el siguiente paso es comenzar a aplicar estilos y los cambios en nuestro html.

**3. Selectores básicos**

**3.1. Selectores de tipo**

Los selectores de tipo coinciden con los elementos por su nombre de nodo. El selector de elementos afectará a cada elemento de este tipo, no importa cuántos. En HTML, los elementos son las etiquetas a añadir, por ejemplo <p>, <header> o <div>.

**Ejemplo**

Esta pieza de código indica al documento HTML que convierta el color del texto dentro de todas las etiquetas <p> de color azul:

|  |  |
| --- | --- |
| p {  color: blue; } | |

**3.2. Selectores de clases**

En HTML y CSS, podemos identificar un grupo de elementos HTML con las mismas características o estilos utilizando un atributo de clase (class=”nombre”). La clase es un atributo del elemento que estamos tratando de modificar y requiere un nombre.

**Ejemplo**

|  |  |
| --- | --- |
| <div class="red-box">  ... // you can style this box with the .red-box selector </div>  <div class="blue-box">  ... // you can style this box with the .blue-box selector </div>  <div class="red-box">  ... // you can style this box with the .red-box selector </div> | |

Debemos nombrar clases utilizando letras minúsculas. Si quieres tener un nombre de clase con varias palabras, debes separarlos usando guiones -, en lugar de guiones bajos \_.

Las clases no hacen nada por sí mismas. En este código, no son más que referencias que agrupan los elementos en una categoría (red-box o blue-box).

**3.3. Selectores de id**

Un selector de ID identifica un elemento único, irrepetible en el código HTML. Los selectores de ID coinciden con un elemento basado en el contenido del atributo id de ese elemento, los cuales deben coincidir exactamente con el valor indicado en el selector.

Selectores de ID se identifican por el símbolo de hash (#). La convención de nomenclatura es la misma que class: si tenemos varias palabras, las concatenamos añadiendo guión - y no un guión bajo \_.

**Ejemplo**

Podemos dar un id a nuestra etiqueta div:

|  |  |
| --- | --- |
| <div id="cover-box">  ... </div> | |

Si queremos cambiar el fondo de ese elemento, seleccionaremos mediante la adición de una regla con el ID como selector (hashtag antes del nombre id).

|  |  |
| --- | --- |
| #cover-box {  background-color: black;  color: pink;  width: 100px;  height: 50px; } | |

**Class vs. ID's**

Añadir una clase o un id a un elemento no hace nada para ese elemento por defecto. No hay valores por defecto del navegador para cualquier ID o clase. Las clases no son únicas, los ID sí lo son.

**3.4 Selectores universales**

El selector universal coincide con un solo elemento de cualquier tipo. Se identifica por un asterisco ( \*).

Omitiendo el asterisco con selectores simples tiene el mismo efecto. Por ejemplo, \*.warning y .warning se consideran iguales, ya que ambos estarán dirigidas a todos los elementos que tienen la clase .warning, independientemente del tipo de elemento (div o el que sea)

Advertencia: Tenga mucho cuidado al usar el selector universal, ya que puede ralentizar considerablemente su página porque se aplica ese estilo a todos los elementos en su página

Este selector se aplica normalmente como el primero de todos en el documento CSS, por lo que se aplicará una regla más específica a un elemento o sobreescribirá una regla sencilla propuesta por el selector \*.

Un uso común para el selector universal es asignar una fuente en particular a través de todos los elementos de una página web.

**Ejemplo**

|  |  |
| --- | --- |
| \* {  font-family: monospace; } | |

**4. Propiedades CSS**

**4.1 Propiedades de color**

HTML tiene cuatro alternativas más conocidas para especificar un color elemento: Por nombre, Hexadecimal, RGB, RGBA

**4.1.1.Colores por nombre**

La primera y más fácil manera de representar un color está indicando su nombre.

**Ejemplo**

Por ejemplo, si queremos usar el color rojo, tendremos que escribir:

|  |  |
| --- | --- |
| p {  color: red; } | |

Esto hará el color de la fuente de la etiqueta <p> sea red.

**4.1.2. Colores en RGB**

Un valor RGB es una función rgb() que recibe tres parámetros: rojo [R], verde [G] y azul [B]. Ellos representan los valores de los canales de los colores.

En RGB cada componente está representado por un número decimal entre 0 y 255, donde 0 significa que no está presente y 255 significa que está presente al 100%.

La teoría del color nos dice que cualquier color puede ser representada como la combinación lineal de los otros tres colores. Esto significa que, básicamente, mediante la mezcla de diferentes cantidades de 3 colores, podemos especificar cualquier otro color.

**Ejemplo**

Vamos a utilizar los colores red y blue con la función rgb():

|  |  |
| --- | --- |
| .red {  color: rgb(255, 0, 0); }  .blue {  color: rgb(0, 0, 255); } | |

**4.1.3. Colores RGB en hexadecimal**

RGB también puede representarse de forma hexadecimal con números que corresponden a los valores de rojo, verde y azul.

Puede parecer un poco complicado al principio, pero en realidad es bastante sencillo. Cada númerodecimal se puede representar en número hexadecimal, la ventaja es que los números hexadecimales entre 0-255 serán siempre dos dígitos:

|  |  |
| --- | --- |
| **Decimal** | **Hexadecimal** |
| 0 | 00 |
| 1 | 01 |
| 2 | 02 |
| … | … |
| 9 | 09 |
| 10 | 0A |
| 11 | 0B |
| 15 | 0F |
| 16 | 10 |
| 17 | 11 |
| 255 | FF |

Cada valor hexadecimal consiste en una almohadilla (#) seguido por tres pares de hexanumbers entre 0-9 y A-F.

**Ejemplo**

En este caso, para indicar el color rojo, debemos hacer lo siguiente:

|  |  |
| --- | --- |
| .red {  color: #ff0000; } | |

Si cada par de valores tiene el mismo valor, como en nuestro ejemplo (ff-00-00), puede simplificar mediante el uso de un solo dígito (f-0-0).

**Ejemplo**

En nuestro ejemplo, podemos obtener el color rojo de la siguiente manera:

|  |  |
| --- | --- |
| .red { color: #f00;} | |

Si usamos otro color, por ejemplo #123456, no se puede reducir mediante el uso de esta técnica.

**4.1.4. Colores en RGBA**

El método RGBA() nos permite utilizar rgb() con otro valor como el cuarto parámetro: el canal alfa.

Esto nos permite ajustar la opacidad del color, con un número entre 0 y 1, donde 0 nos indica que el color será 100% transparente y 1 , 100% opaco.

**Ejemplo**

Vamos a agregar 50% de transparencia en nuestro color rojo en el párrafo:

|  |  |
| --- | --- |
| .half-transparent-green {  color: rgba(0, 255, 0, 0.5); } | |

**4.1.5. La propiedad color**

La propiedad color es la encargada de dar color a la fuentes.

**Ejemplo**

|  |  |
| --- | --- |
| h1 {  color: forestgreen; }  h2 {  color: #008080; }  p {  color: rgb(70, 130, 180); } | |

**4.1.6. La propiedad background-color**

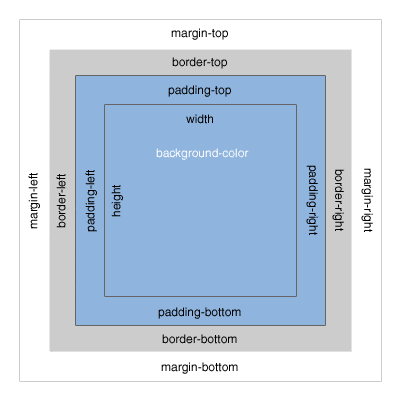
Podemos cambiar el color de fondo con la propiedad background-color

**Ejemplo**

|  |  |
| --- | --- |
| body { background-color: rgba(173, 216, 230, 0.4); } | |

**4.2 Model box**

El modelo de caja CSS es una serie de cuadrados concéntricos utilizados para modelar la forma en que los elementos ocupan espacio en una página web.



Con el fin de comprender plenamente el diseño web basado en CSS tiene que aprender algo realmente básico: cada elemento HTML es una caja rectangular.

**Ejemplo**

|  |  |
| --- | --- |
| <div>  Ejemplo de CSS Box Model  </div> | |
| div {  background-color: #00D1AE;  width: 170px;  } | |

En este código, el elemento <div> tiene un background-color verde. Esto nos permitirá ver su espaciado predeterminado. También hemos aplicado un width fijo para ayudarnos a tener en cuenta lo que sucede cuando se aplica cada propiedad al elemento <div>.

**4.2.1 La propiedad border**

La propiedad border permite configurar el borde de un elemento.

Puede especificar tres valores para esta propiedad: **border-width, border-style y border-color.**

Las propiedades border-width y border-color tienen valores por defecto, por lo que si no se especifica uno de ellos, tendrá un valor, que será cero o el valor por defecto del navegador.

**Ejemplo**

Vamos a añadir la propiedad border a nuestro <div>:

|  |  |
| --- | --- |
| border: 1px solid #17212e; | |

En el ejemplo anterior, el borde se define con 3 valores:

**width (1px)**: Podemos especificar cualquier tamaño de unidad para ajustar el ancho del borde. También podemos usar la propiedad **border-width** (para modificar únicamente el width del borde).

**style (solid)**: Podemos especificar cualquier estilo de borde, dependiendo del diseño que desea tener. También podemos usar la propiedad **border-style** para hacer eso. Esta propiedad no tiene un valor predeterminado. Si no lo especifica, no aparecerá su borde.

**color (#17212e)**: Finalmente, podemos especificar un color, en cualquiera de los formatos que ya hemos visto. También podemos utilizar la propiedad **border-color** para hacer eso.

La forma abreviada:

|  |  |
| --- | --- |
| border: 1px solid #17212e; | |

es equivalente a:

|  |  |
| --- | --- |
| border-color: #17212e; border-style: solid; border-width: 1px; | |

**El comportamiento del espaciado en el borde.** Si echamos un vistazo a nuestro CSS, el elemento <div> debe tener una anchura de 170px. Después de la aplicación del borde, el ancho no es 170px, si no 172px. ¿Por qué? Cuando aplica un borde a un elemento aumenta su anchura, incluso aunque se haya especificado un ancho fijo. El borde de nuestro elemento es 1px ancho, así que la anchura total del <div> será 172px (el <div> 170px anchura + 1px de border-left + 1px de border-right).

**Un solo borde.** Vamos a suponer que sólo queremos tener el borde inferior en su <div>. Hay dos maneras diferentes de hacerlo.

Número uno:

|  |  |
| --- | --- |
| border-bottom: 1px solid #17212e; | |

La otra forma es un poco más largo. Se trata de establecer el border con la propiedad border-style y la propiedad border-color y, a continuación, establecer el border-width siguiente:

|  |  |
| --- | --- |
| border: solid #17212e; border-width: 0 0 1px 0; | |

Como se puede ver, border-width recibe 4 valores diferentes. Esos valores corresponden a:

|  |  |
| --- | --- |
| border-width: [border-top] [border-right] [border-bottom] [border-left]; | |

El border-bottom tiene un valor de 1px, y es el único que se muestra. Puede utilizar esta técnica con todas las propiedades border, pero no es común:

|  |  |
| --- | --- |
| border-color: red blue green yellow; border-style: solid dashed dotted double; | |

En este punto, el CSS en codepen debe ser:

|  |  |
| --- | --- |
| div {  background-color: #00D1AE;  border: 1px solid #17212e;  width: 170px; } | |

**Border-radius.** La propiedad border-radius permite definir la forma de las esquinas redondeadas del border.

**Ejemplo**

Para establecer un borde redondeado en nuestro elemento, podemos hacer esto:

|  |  |
| --- | --- |
| border-radius: 12px; | |

Se va a crear un radio de 12 píxeles de borde. Como lo hicimos antes con la propiedad border-radius, puede establecer los valores de border-radius individualmente. La sintaxis es la siguiente:

|  |  |
| --- | --- |
| border-<top|bottom>-<left|right>-radius | |

Si usted quiere tener un radio de 12 píxeles de borde justo en el borde superior izquierdo, tiene que establecer la propiedad de la siguiente manera:

|  |  |
| --- | --- |
| border-top-left-radius: 12px; | |

**4.2.2. La propiedad padding**

La propiedad padding establece el espacio entre el contenido y el borde de un elemento.

**Ejemplo**

Añadir un poco de padding a su código:

|  |  |
| --- | --- |
| padding: 10px 10px 10px 10px; | |

**El comportamiento del espaciado en el padding.** Al igual que la propiedad border, la propiedad padding aumenta el tamaño del <div>. El ancho total no será el especificado en el CSS, será la suma de todos los modificadores.

**Ejemplo**

En este caso, será 192px = el <div> tiene 170px de width + 10px padding-left + 10px padding-right + 1px border-left + 1px border-right.

**Padding values.** Como hemos visto, podemos especificar valores diferentes para el relleno. El comportamiento es el mismo que vimos en la propiedad border-width.

|  |  |
| --- | --- |
| padding: [padding-top] [padding-right] [padding-bottom] [padding-left]; | |

Nosotros no tenemos que configurar los cuatro valores cada vez. De manera similar a la propiedad border, podemos indicar uno, dos, o tres valores. Si hacemos eso, cada valor afecta a un lado del elemento:

|  |  |
| --- | --- |
| padding: 10px 20px 15px;  /\* padding-top 10px \*/ /\* padding-right and padding-left 20px \*/ /\* padding-bottom 15px \*/ | |
| padding: 10px 20px;  /\* padding-top and padding-bottom 10px \*/ /\* padding-right and padding-left 20px \*/ | |
| padding: 10px;  /\* All the paddings are set to 10px \*/ | |

**Ejemplo**

En este punto, el CSS debe tener este aspecto:

|  |  |
| --- | --- |
| div {  background-color: #00D1AE;  border: 1px solid #17212e;  padding: 10px;  width: 170px; } | |

**4.2.3. Box-sizing Property**

Como hemos visto, cuando aplicamos un borde o un padding para un elemento de bloque, hay un cambio de anchura y altura. A veces, cuando tratamos de integrar los elementos de un diseño, no hay espacio para cambiar la anchura y la altura. Así que, ¿cómo podemos evitar este espacio extra en nuestros elementos? ¿Cómo podemos tener un <div> con 170px de ancho, un borde de 1px y un padding de 10px?

La propiedad **box-sizing** se utiliza para modificar el modelo de caja predeterminado que se utiliza para calcular la anchura y la altura de los elementos. Esta propiedad no es compatible, por defecto, por todos los navegadores. Aunque, es posible utilizarlo para simular el comportamiento de navegadores que no lo soportan correctamente.

**Ejemplo**

Volvamos a nuestro ejemplo en CodePen. Hemos calculado el espacio total utilizado por nuestro <div>. Añadiendo la propiedad de box-sizing podemos reducirlo a 170 píxeles de ancho:

|  |  |
| --- | --- |
| box-sizing: border-box; | |

La propiedad box-sizing puede tener dos valores diferentes:

**border-box:** Las propiedades de width y heigth se miden incluyendo sólo el contenido, pero no el padding, border o margin.

**fronter-box**: la propiedad width y height incluyen el contenido, el relleno y el borde, pero no el margen.

**Comportamiento de los espacios en box-sizing.** Con el ajuste de la propiedad box-sizing reducimos el espacio actual del elemento. No, es 170px:

el <div> tiene 170px de ancho

+ 10px padding-left (que aumenta 0px con box-sizing).

+ 10px padding-right (que aumenta 0px con box-sizing).

+ 1px border-left, (que aumenta 0px con box-sizing).

+ 1px border-right, (que aumenta 0px con box-sizing).

= 170px

**4.2.4. La propiedad margin**

La propiedad margin establece el espacio entre el borde de un elemento y los otros elementos a su alrededor. Si no hay elementos alrededor, establecerá el espacio entre el elemento y la etiqueta <body>.

**Ejemplo**

Añadir un poco de margen a su código y ver lo que sucede:

|  |  |
| --- | --- |
| margin: 10px 10px 10px 10px; | |

**Comportamiento de los espacios en margin.** Cuando se aplica el margen, el tamaño del <div> sigue siendo el mismo: 170px. Sin embargo, el espacio ocupado en el HTML es más grande. Es ahora es 190px:

el <div> tiene 170px anchura

+ 10px padding-left, (que aumenta 0px con box-sizing).

+ 10px padding-right, (que aumenta 0px con box-sizing).

+ 1px border-left, (que aumenta 0px con box-sizing).

+ 1px border-right, (que aumenta 0px con box-sizing).

+ 10px margin-right

+ 10px margin-left

= 190 pixels

**Valores de margin.** El establecimiento de margin también funciona como la propiedad padding.

Puede especificar los valores de los márgenes declarándolas de la siguiente manera:

|  |  |
| --- | --- |
| margin: [margin-top] [margin-right] [margin-bottom] [margin-left]; | |

**Ejemplo**

Una vez más, no tenemos para indicar todos los valores.

|  |  |
| --- | --- |
| margin: 10px 20px 15px;  /\* margin-top 10px \*/ /\* margin-right and margin-left 20px \*/ /\* margin-bottom 15px \*/ | |
| margin: 10px 20px;  /\* margin-top and margin-bottom 10px \*/ /\* margin-right and margin-left 20px \*/ | |
| margin: 10px;  /\* All the margins are set to 10px \*/ | |

**4.2.5. La propiedad position en CSS**

CSS nos da la capacidad de colocar y organizar los elementos y contenidos de múltiples maneras. La comprensión de cómo podemos colocar cajas en nuestro diseño es una clave para hacer el contenido más digerible.

La propiedad **position** CSS le permite organizar los elementos de la pantalla. La posición puede ser: static, relative, fixed, absolute.

Cuando se ajusta la position, podemos utilizar las siguientes propiedades: top, right, bottom, left.

**Position: static.** El valor inicial o por defecto de un elemento es static. Esto significa que el elemento en su posición no es de ninguna manera especial. Se muestra donde normalmente estaría en el documento.

Los elementos con position: static, se describen como no posicionados.

Los elementos con cualquier otra forma de la propiedad position se describen como posicionados.

**Position: relative.** Un elemento relative está posicionado con respecto a otro elemento.Un elemento posicionado como relative se comporta igual que static a excepción de algunas propiedades adicionales.

Puede ajustar la posición mediante el establecimiento de las propiedades top, left, right, y bottom. Se moverá el elemento de forma relativa al resto de elementos.

Para entender mejor la position: relative, echar un vistazo al siguiente ejemplo:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/JRoWwq> | |

**Position:fixed.** Un elemento fixed está posicionado con respecto a la pantalla de nuestro dispositivo (ordenador, móvil, etc.). El elemento estará siempre en el mismo lugar, incluso si se desplaza la página. Al igual que con relative la posición se controla con top, left, right y bottom.

Aquí se puede ver un ejemplo con los resultados de la position: fixed:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/qaEmWW> | |

**Position:absolute.** Este es el valor de la posición más difícil de entender. Absolute funciona como fixed pero en vez de situarse en relación con el área de visualización, está posicionada para el elemento posicionado más próximo en el DOM. Esto es, el elemento más cercano con una posición diferente que static encuentra en el CSS.

Si no hay ningún elemento antepasado posicionado en el documento, se utilizará el body del documento y que todavía se mueve junto con el desplazamiento de página.

Echa un vistazo al siguiente ejemplo para ver algo curioso de este posicionamiento:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/GjgvjK> | |

Como se puede ver, hay dos <div> uno sobre el otro. Si inspecciona el código de abajo se puede ver que los elementos de la página con posición absoluta se eliminan del flujo de la página. Esto significa que no afecten a la posición de otros elementos, y los otros elementos no los afectarán.

**4.2.6. La propiedad Float en CSS**

Normalmente, los elementos HTML siguen un flujo normal, la colocación de un elemento debajo del otro. La propiedad float especifica que un elemento debe seguir del flujo normal pero se coloca al lado de la izquierda o derecha de su contenedor (su padre).

Float puede tomar los siguientes valores:

**left.** Indica que el elemento debe flotar en el lado izquierdo de su bloque de contención.

**right.** Indica que el elemento debe flotar en el lado derecho de su bloque de contención.

**none.** Se indica que el elemento no debe flotar.

**inherit.** Se tomará el valor del float del contenedor principal del elemento.

Tenga en cuenta que los elementos que flotan siguen siendo una parte del flujo de la página web, sólo que su posición dentro del flujo se altera. Esto es claramente diferente de los elementos de la página que utilizan posicionamiento absoluto.

Puede utilizar float para varias cosas. Puede ajustar el texto alrededor de las imágenes o elementos HTML mediante su uso, como se puede ver aquí:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/LREzRV> | |

Pero esta no es la única aplicación de la propiedad float. Puede utilizar float para crear diseños web enteros. Por ejemplo, aquí se puede ver un pequeño ejemplo con un diseño web normal compuesta con elementos flotantes:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/vXEZmy> | |

**Clear de los Float.** Cuando trabajamos con float, tarde o temprano necesitaremos usar la propiedad clear. Un elemento con la propiedad clear no se moverá hacia arriba de forma adyacente al float como lo desea el float, sino que se colocara debajo del float. Será más fácil de entender esto con un ejemplo:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/Gjgryv> | |

La propiedad clear tiene cuatro valores válidos:

**left**. Despeja el float del lado izquierdo.

**right**. Despeja el float del lado derecho.

**both**. Despeja float procedentes de ambas direcciones.

**none**. Es el valor predeterminado.

**4.2.7. La propiedad Vertical Align**

La propiedad vertical-align CSS especifica la alineación vertical de una línea o una caja-celda de la tabla.

La propiedad vertical-align se puede utilizar en dos contextos: Para alinear verticalmente la caja de un elemento dentro de la caja que lo contiene o para alinear verticalmente el contenido de una caja de una tabla.

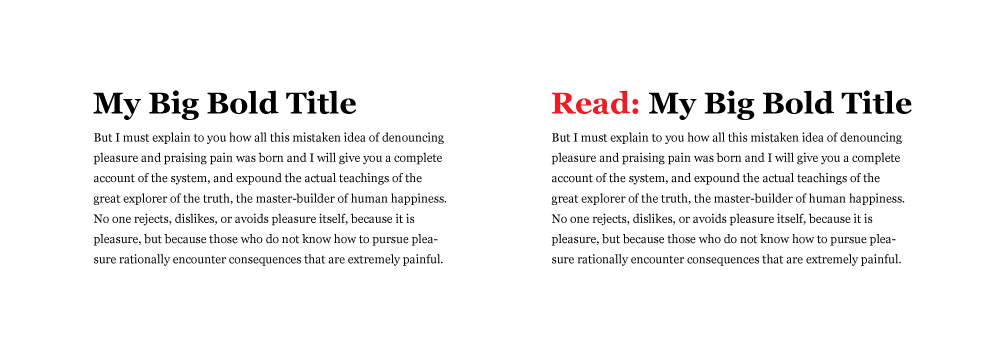
**4.3. Propiedades especiales**

**4.3.1 :before**

Este selector permite elegir un elemento CSS e insertar contenido antes de cada elemento con una clase específica aplicada a él. Digamos que tienes un sitio web donde quieres un texto específico antes de cada etiqueta H2. Utilizariamos esta configuración:

|  |  |
| --- | --- |
| h2:before {   content: "Read: ";  color: #F00;  } | |

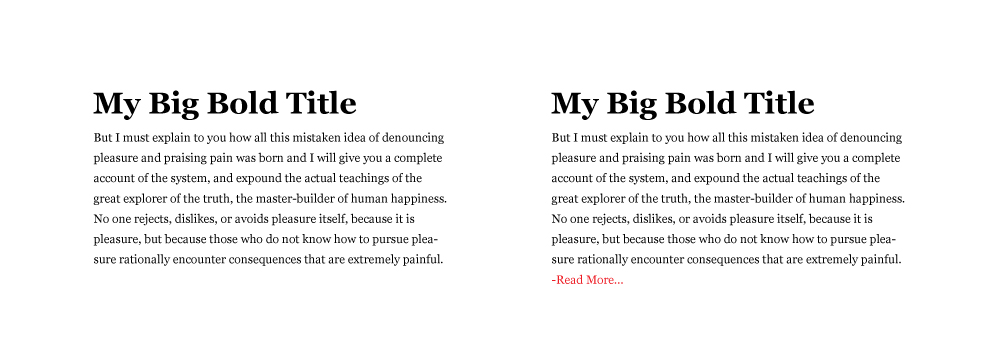
Esto es extremadamente útil, especialmente si está usando una fuente de icono. Puede colocar iconos antes de ciertos elementos y aplicarlos globalmente.



**4.3.2 :after**

Al igual que el selector: before, puede usar :after para insertar contenido globalmente en elementos específicos. Un uso práctico sería agregar "leer más" después de cada extracto en un blog. Así es como lo harías.

|  |  |
| --- | --- |
| p:after{   content: " -Read more… ";  color:#f00; } | |



**4.3.3 content**

content es una propiedad de CSS que es útil cuando necesita insertar un elemento que desea controlar. El uso más común que he visto para esto es insertar un icono de una fuente de icono en un lugar específico. En los ejemplos anteriores, puede ver que debe ajustar el texto que desea insertar entre comillas.

Hay que utilizar content para generar contenido dinámico en las propiedades :before y :after.

**5. CSS | Selectores avanzados**

Antes de aprender acerca de los selectores CSS combinados, es una buena idea comprender la estructura del documento en el Document Object Model (DOM).

Todos los documentos HTML pueden ser representados como árboles. Cada nivel del árbol se describe en la misma forma que un árbol de la familia humana, con ascendientes, descendientes, padres, hijos y hermanos. selectores CSS pueden describir los elementos de la estructura del documento.

Por ejemplo, en un documento HTML, el <body> es el padre de todos los demás elementos que aparecen en la página. Por el contrario, un hijo se refiere a cualquier elemento que está conectado, pero más abajo en la estructura del documento. No importa cuántos niveles más bajos.

Los padres y los hijos trabajan igual. Un padre es un elemento que está directamente encima y conectado a un elemento en la estructura del documento; un hijo está directamente debajo y conectado a un elemento de la estructura del documento.

|  |  |
| --- | --- |
| <ul>  <li class="menu-item"></li>  <li class="menu-item"></li> </ul> | |

En este ejemplo, el <ul> es un padre para todos <li>. Esos <li> son sus hijos.

**5.1. Selección de Descendientes**

**5.1.1. Direct Descendant (hijo)**

|  |  |
| --- | --- |
| padre > hijo {  property: value; } | |

El > separa dos selectores y coincide sólo con aquellos elementos que son hijos directos de elemento padre.

Imagine que tiene varias listas y desea el mismo estilo para todas ellas. Todos las etiquetas li dentro del menú ul etiqueta tendrán 50px 100px altura y anchura.

Una opción sería la de asignar una clase a cada etiqueta li en la lista del menú, a continuación, escribir las propiedades para ellos.

|  |  |
| --- | --- |
| <ul>  <li class="menu-item"></li>  <li class="menu-item"></li>  <li class="menu-item"></li>  <li class="menu-item"></li>  <li class="menu-item"></li> </ul> | |

|  |  |
| --- | --- |
| .menu-item {  width: 100px;  height: 50px; } | |

Pero ... tenemos una manera más fácil: El selector hijo >

El > seleccionará aquellos elementos que coinciden con el segundo selector que son hijos directos de elementos coincidentes por el primero.

Si añadimos un menu id para el ul, podemos simplemente seleccionar todos sus hijos li mediante el #menu > li selector:

|  |  |
| --- | --- |
| <ul id="menu">  <li>1</li>  <li>2</li>  <li>3</li>  <li>4</li>  <li>5</li> </ul> | |

|  |  |
| --- | --- |
| #menu > li {  width: 100px;  height: 50px; } | |

Ahora tenemos una regla que dice que cada li hijo del elemento del id=“menu” tendrá un height de 50px y un width de 100px.

**5.1.2. Descendant Selector**

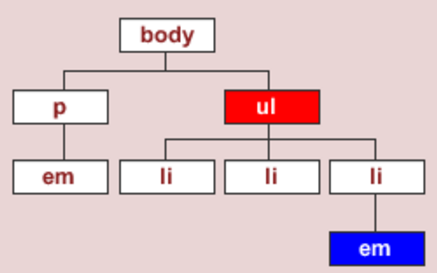
|  |  |
| --- | --- |
| selector1 selector2 {  property: value; } | |

El Selector Descendiente nos señala a aquellos elementos del selector 2, que son descendientes del selector1.

Vamos a tomar este fragmento de HTML un segundo y pensar en cómo nos gustaría apuntar al em interior de la li, sin apuntar al em interior de la p:

|  |  |
| --- | --- |
| <p>IronHack <em>Descendant</em> Selectors</p> <ul>  <li>item 1</li>  <li>item 2</li>  <li><em>item 3</em></li> </ul> | |

Bueno, podríamos añadir, por supuesto, una clase a la segunda em, o podríamos añadir una identificación también. Sin embargo, el uso de esta combinación de selectores ayuda a relacionar los elementos sin tener que perder el tiempo con el código HTML demasiado.



**5.2. Selección de Siblings (hermanos)**

**5.2.1. Adjacent Sibling**

|  |  |
| --- | --- |
| formerElement + targetElement {  property: value; } | |

Esto se conoce como un selector adyacente o selector de hermanos próximos. Se seleccionará únicamente el elemento especificado que sigue inmediatamente al primer elemento especificado.

Este selector le permite obtener los elementos que se encuentran uno junto al otro:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/amBQEG> | |

Si miramos cuidadosamente el código CSS:

|  |  |
| --- | --- |
| /\* All DIVs Green by default \*/ div { background-color: green;}  /\* ID=former... make it red\*/ #former { background-color: red; }  /\* First adjacent sibling of #former is YELLOW \*/ #former + div { background-color: yellow; } | |

**5.2.2. La selección múltiple**

|  |  |
| --- | --- |
| selector1, selector2 {  property: value; } | |

Imaginemos que queremos todos nuestros títulos de h1 a h6 sean de color azul. Podemos añadir una regla separada para h1, otra regla para h2, otra para h3... pero hay una manera mejor. La selección múltiple nos permite aplicar la misma lista de reglas a diferentes elementos:

|  |  |
| --- | --- |
| h1, h2 {  color: blue; } | |

Y este método también permite trabajar con selectores más complicados, aunque probablemente debería poner cada selector en una línea diferente para facilitar la lectura. Por ejemplo, es posible que queramos diferentes elementos tengan background-color amarillo:

|  |  |
| --- | --- |
| .todo-list > li, h2 + .red-box, h2 {  background-color: yellow; } | |

Usted puede mezclar selectores todo lo que necesita para crear páginas web impresionantes!

**5.3. Selector de atributo**

|  |  |
| --- | --- |
| element[attr-name="value"] {  property: value; } | |

Podemos seleccionar elementos por nombre de la etiqueta, por qué no seleccionar un elemento por sus atributos? Vamos a ver cómo funciona esto:

|  |  |
| --- | --- |
| <https://codepen.io/ironhack/pen/GjAWAo> | |

Sólo los vínculos con la palabra hack en su src atributo se ven afectados por la norma, y por lo tanto se muestran en verde.

|  |  |
| --- | --- |
| Otros ejemplos de este mismo selector: | |
| Selector | ¿Qué se elige? |
| [attribute] | Todos los elementos con el atributo especificado |
| [attribute=value] | Todos los elementos donde el atributo especificado es igual al 'valor' |
| [attribute~=value] | Todos los elementos con un atributo que contiene la palabra 'valor' |
| [attribute|=value] | Todos los elementos con una lista de atributos que comienzan con 'valor' |
| [attribute^=value] | Todos los elementos con un atributo que comienzan con 'valor' |
| [attribute$=value] | Todos los elementos con un atributo que termina con 'valor' |
| [attribute\*=value] | Todos los elementos con un atributo que contiene una subcadena con el 'valor' |
| Aquí tienes un ejemplo más para practicar: <https://codepen.io/ironhack/pen/RGGQWr> | |

**6. Recursos externos**

JUEGO PARA PRACTICAR CSS

|  |  |
| --- | --- |
| <http://flexboxfroggy.com/> | |

ANIMACIONES CSS

|  |  |
| --- | --- |
| <https://daneden.github.io/animate.css/> | |

TODAS LAS PROPIEDADES CSS EXPLICADAS

|  |  |
| --- | --- |
| <https://daneden.github.io/animate.css/> | |

|  |  |
| --- | --- |
| <https://cssguidelin.es/> | |