Contenidos a Trabajar

- Introducción a CSS.
- Formas de integrarlo en un documento HTML
- Selectores, propiedades y valores básicos del editor de código (Color - Tipografía)
- Agrupamientos (div y span) - Atributos Globales - Especificidad- Etiquetas semánticas
- Uso de Google Fonts y las distintas formas de colocar una tipografía
- Colores (Hexadecimal - RGB - RGBA) : herramientas para hacer mezcla de colores
- Iconos (fontawesome)

Atributos Globales HTML y CSS

Los atributos globales son atributos comunes a todos los elementos HTML y pueden usarse en todos los elementos, aunque pueden no tener efecto en algunos de ellos. Los atributos globales pueden especificarse en todos los elementos HTML, e incluso en aquellos no especificados en el estándar. Esto significa que cualquier elemento no estándar también debe permitir estos atributos, aún cuando el uso de tales elementos significa que el documento ya no cumple con HTML5.

 Por ejemplo:

> Los navegadores que cumplen HTML5 esconden contenido que se marque como <foo hidden>...</foo>, aún cuando <foo> no es un elemento HTML válido.

Descripción

A continuación, enumeramos atributos que se utilizan en HTML y que reciben o interactúan con CSS para cumplir con la norma de maquetado actual.

- **accesskey**

Proporciona y genera un acceso de teclado para el elemento actual. Este atributo consiste de una lista de caracteres, separadas por espacios. El navegador debe utilizar el primero que exista en la distribución del teclado del ordenador.

El valor del atributo debe constar de un solo carácter imprimible (que incluye caracteres acentuados y otros que pueden ser generados por el teclado).

HTML

```
<p>Para probar presioná una
<strong><u>X</u></strong> y estarás dando foco a este botón:</p>

<button accesskey="x">Botón de Ejemplo</button>
```

CSS

```
.output {
   font: 1rem 'Fira Sans', sans-serif;
   letter-spacing: 1px;
}
```

Funciona con Alt+Shift+key seleccionada en navegadores Chrome y Mozilla, puede modificarse en algún navegador su forma de llamada.

- **class**

Es una lista de clases del elemento, separadas por espacios. Las clases permiten a CSS y JavaScript seleccionar y acceder a elementos específicos a través de selectores de clase o funciones, como el método Document.getElementsByClassName(), que veremos en detalle y usaremos más adelante.

HTML

```
<p>Periodista: Gracias por la entrevista.</p>

<p class="nota editorial">Esta es una nota realizada en 2019.-</p>

<p>Periodista: Como fueron sus comienzos?.</p>

<p class="nota">[Podés continuar leyendo esta nota en la pag.20]</p>
```

CSS

```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}

.nota {
   font-style: italic;
   font-weight: bold;
}

.editorial {
   background: rgb(255, 0, 0, .25);
   padding: 10px;
}

.editorial:before {
   content: 'Editor: ';
}
```

- **contenteditable**

Es un atributo enumerado que indica si el elemento puede ser modificable por el usuario. Si es así, el navegador modifica el elemento para permitir la edición.

El atributo debe tener uno de los siguientes valores:

- true o un valor vacío, el cual indica que el elemento debe ser editable.
- false, el cual indica que el elemento no debe ser editable.

HTML

```
<blockquote contenteditable="true">
  <p>Edite el contenido de este texto</p>
</blockquote>

<cite contenteditable="true">-- Puede escribir algo aquí</cite>
```

CSS

```
.output {
  font: 1rem 'Fira Sans', sans-serif;
}

blockquote {
  background: #eee;
  border-radius: 5px;
  margin: 16px 0;
}

blockquote p {
  padding: 15px;
}

cite {
  margin: 16px 32px;
}

blockquote p::before {
  content: '\201C';
}

blockquote p::after {
  content: '\201D';
}

[contenteditable='true'] {
  caret-color: red;
}
```

Nota: content: '\201C'; y content: '\201D'; se refiere a las dobles comillas.

- **data-***

Son atributos globales que forman una clase denominados atributos de datos personalizados , y que se caracterizan por permitir intercambiar información entre el HTML en el momento de la carga del archivo HTML.

HTML

```
<h1>Socios del Club</h1>

<ul>
   <li data-socio="10784">Juan Carlos, Vitalicio.</li>
   <li data-socio="97865">Roberto Gomez, Juvenil.</li>
   <li data-socio="45732">Juana Martinez, Juvenil.</li>
</ul>
```

CSS

```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}

h1 {
   margin: 0;
}

ul {
   margin: 10px 0 0;
}

li {
   position: relative;
   width: 200px;
   padding-bottom: 10px;
}

li:after {
   content: 'Nº Socio: ' attr(data-socio);
   position: absolute;
   top: -22px;
   left: 10px;
   background: black;
```

```
  color: white;
  padding: 2px;
  border: 1px solid #eee;
  opacity: 0;
  transition: 0.5s opacity;
}

li:hover:after {
  opacity: 1;
}
```

- **dir**

Es un atributo enumerado que indica la direccionalidad del texto del elemento.

Puede tener los siguientes valores:

- **ltr**, significa left to right y se utiliza para idiomas que se escriben de izquierda a derecha (como el Español).
- **rtl**, significa right to left y se utiliza para idiomas que se escriben de derecha a izquierda (como el Árabe);
- **auto**, permite que el navegador decida utilizando un algoritmo básico, que analiza los caracteres dentro del elemento hasta que encuentra un caracter con una direccionalidad fuerte, a continuación, aplica la direccionalidad a todo el elemento.

HTML

```
<p dir="rtl">Esto está en español pero no está bien alineado.</p>
<p dir="ltr">Esto está en español y está bien alineado .</p>

<hr>

<p>هذه الفقرة باللغة العربية ولكن بشكل خاطئ من اليسار إلى اليمين.</p>
<p dir="auto">هذه الفقرة باللغة العربية ، لذا يجب الانتقال من اليمين إلى اليسار.</p>
```

CSS

```
.output {
  font: 1rem 'Fira Sans', sans-serif;
  letter-spacing: 1px;
}
```

- **hidden**

Es un atributo Booleano que indica si el elemento aún no es, o ya no es, relevante.

Por ejemplo, puede usarse para ocultar elementos de la página que no pueden usarse hasta que el proceso de ingreso se complete. El navegador no mostrará ni renderizará dichos elementos.

HTML
```
<p hidden>Este texto está oculto</p>

<p >Este texto se puede ver </p>
```

CSS
```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}

p {
   background: #ffe8d4;
   border: 1px solid #f69d3c;
   padding: 5px;
   border-radius: 5px;
}
```

- **id**

Define un identificador único (ID) el cual debe ser único en todo el documento. Su propósito es identificar el elemento para distinguirlo en programación de JavaScript o definir un estilo CSS.

HTML
```
<p>Este es un párrafo normal.</p>

<p id="resaltar">Este párrafo queremos que se note mucho!</p>
```

CSS
```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}

#resaltar {
```

```
  background: linear-gradient(to bottom, #ffe8d4, #f69d3c);
  border: 1px solid #696969;
  padding: 10px;
  border-radius: 10px;
  box-shadow: 2px 2px 1px black;
}

#resaltar:before {
  content: "i";
  margin-right: 5px;
}
```

- **lang**

Participa en la definición del lenguage del elemento, refiriéndonos exclusivamente al idioma en que son escritos estos elementos. El tag contiene un solo valor en el formato definido.

HTML

```
<p>Párrafos en diferentes idiomas.</p>

<p lang="en-GB">This paragraph is defined as British English.</p>

<p lang="fr">Ce paragraphe est défini en français.</p>
```

CSS

```
.output {
  font: 1rem 'Fira Sans', sans-serif;
}

p::before {
  padding-right: 5px;
}

[lang="en-GB"]::before {
  content: url('imagen de una bandera british-flag.png');
}

[lang="fr"]::before {
  content: url('imagen de una bandera french-flag.png');
```

}

ⓘ Puede consultarse la lista internacional de idiomas definidos por región en:

https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry

- **style**

Contiene declaraciones de estilo CSS para ser aplicadas al elemento.

Siempre es recomendado que los estilos sean definidos en archivo/s separados. Este atributo y el elemento <style> tienen el objetivo principal de permitir un estilizado rápido, por ejemplo con fines de testeo o pruebas.

HTML
```
<div style="background: #ffe7e8; border: 2px solid #e66465;">
   <p style="margin: 15px; line-height: 1.5; text-align: center;">
    Esta es la primer linea del párrafo y ahora un salto de línea<br>
      esta segunda línea sigue siendo aplicada por el style aunque <b>esto debe hacerse con CSS.</b></p>
</div>
```

CSS
```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}
```

- **tabindex**

Es un atributo que recibe un parámetro con un número entero y que indica si el elemento puede obtener el foco del cursor y si debe participar de la navegación secuencial con el teclado. Si se cumplen estas condiciones, también determina en qué posición estará ordenado según la tabulación.

Puede tomar diferentes valores:

- **un valor negativo** significa que el elemento debe ser focuseable, pero no debe ser alcanzado vía la navegación secuencial del teclado.
- **Un cero** significa que el elemento puede obtener el foco y alcanzable vía la navegación secuencial del teclado, pero el orden relativo es definido por la convención de la plataforma.

- **un valor positivo** que significa que puede obtener el foco vía la navegación secuencial del teclado, el orden relativo es definido por el valor del atributo y sigue la secuencia en orden ascendente según el valor de tabindex.

Si varios elementos comparten el mismo tabindex, su orden relativo sigue su posición en el documento.

HTML
```
<p>Hacé click en cualquiera de estos componentes.</p>

<label>Primer elemento:<input type="text"></label>

<div tabindex="0">Segundo elemento</div>

<div>Elemento no indexado</div>

<label>Tercer elemento<input type="text"></label>
```

CSS
```
.output {
  font: .9rem 'Fira Sans', sans-serif;
}

p {
  font-style: italic;
}

div,
label {
  display: block;
  letter-spacing: .5px;
  margin-bottom: 1rem;
}

div:focus {
  font-weight: bold;
}
```

- **title**

Contiene un texto que representa información de asesoramiento en relación al elemento al que pertenece. Dicha información puede típicamente, pero no necesariamente, presentarse al usuario como un tooltip.

HTML

```
<p>Estamos viendo las siguientes tecnologías:
<abbr title="Hypertext Markup Language">HTML</abbr>,
<abbr title="Cascading Stylesheets">CSS</abbr>, and
JavaScript.</p>
```

CSS

```
.output {
   font: 1rem 'Fira Sans', sans-serif;
}
```

- **translate**

Es un atributo enumerado que se utiliza para especificar si los valores de los atributos de un elemento son traducidos cuando su página es localizada, o si hay que dejarlos sin cambios.

Puede tener los siguientes valores:

- **cadena vacía y "yes"**, indica que el elemento será traducido.
- **"no"**, indica que el elemento no será traducido.

 Ejemplo:

En este ejemplo, el atributo se utiliza para pedir a las herramientas de traducción que no traduzcan la marca de la empresa en el pie de página.

```
<head>
        <meta http-equiv="Content-Language" content="es"/>

</head>
<html>
<body>
esta página está en idioma español
<footer>
  <small>© 2020 <code translate="no">Brand Name</span></code>
</footer>
</body>
```

Introducción a CSS

CSS es el segundo lenguaje más básico y esencial para crear páginas web.

1. El primero sería HTML, con el que se define el contenido de la página.
2. El segundo CSS, con el que se define la parte de la presentación, es decir, cómo deben mostrarse los elementos de la página, su posición, forma, espaciados, colores y en resumen, toda la parte estética.

CSS es un lenguaje que consiste en una serie de elementos mediante los cuales se declaran los estilos, básicamente éstos son los más importantes:

• Selectores, mediante los cuales podemos especificar qué elementos de la página nos estamos refiriendo
• Atributos de estilo para definir qué cosas queremos estilizar sobre los selectores indicados
• Una serie de valores, que indican qué estilo se debe aplicar a cada atributo sobre cada selector.

Los valores se expresan con unidades CSS, que sirven para cuantificar los valores (píxeles, puntos…).

Aprender CSS no es difícil, pero cuando se usa profesionalmente se deben tener en cuenta muchos detalles y buenas prácticas, como la organización del código, la reutilización, la optimización, etc.}

Separar el código CSS del código HTML
El enfoque de CSS es servir para definir la capa de presentación, es decir, la parte relacionada con el aspecto. Es algo que cualquier estudiante suele tener claro cuando está aprendiendo CSS, ya que al enseñar HTML probablemente se haya insistido, pero que siempre conviene reforzar.

En el código HTML colocamos el contenido, es decir, qué debe visualizarse, mientras que con CSS definimos la presentación, es decir, cómo debe visualizarse. Esto nos lleva a una serie de usos de CSS que debemos de respetar como buenas prácticas.

• Lo adecuado cuando trabajamos con CSS es escribir el código en ficheros independientes, que tendrán extensión .css.
• No conviene colocar código CSS por tanto dentro de archivos HTML. Debemos evitar colocar estilos en etiquetas <style> dentro del propio código HTML.
• Por supuesto, mucho menos aconsejable es colocar estilos en los atributos "style" de las etiquetas HTML.

La aparición de CSS 3 sólo se materializó en el año 2014 con el movimiento de HTML 5. Vendría a aportar soluciones a la mayoría de las necesidades de los diseñadores y a permitir finalmente cubrir el objetivo principal del lenguaje, la separación del contenido de la presentación. No obstante cabe decir que CSS 3 se presentó por medio de un nutrido grupo de especificaciones, que han ido mejorándose e incrementándose hasta la fecha, por lo que no es tanto un lanzamiento puntual, sino una continua mejora del estándar a diversos niveles.

Usos de las CSS

Hemos denominado a este apartado los diferentes usos de las CSS y relata justamente eso, los distintos niveles a los que podemos usar las Hojas de Estilo, que van desde definir los estilos de

manera específica, para un pequeño fragmento de una página, hasta los estilos para todo un sitio web completo.

Para definir estilos en secciones reducidas de una página se puede utilizar el atributo style en la etiqueta sobre la que queremos aplicar estilos.

Como valor de ese atributo indicamos en sintaxis CSS las características de estilos.

Lo vemos con un ejemplo, pondremos un párrafo en el que determinadas palabras las vamos a visualizar en color verde.

```
<p>
Esto es un párrafo en varias palabras <span style="color:green">en color verde</span>. resulta muy fácil.
</p>
```

Nota: La etiqueta del HTML quizás no sea tan conocida como otras. Es una etiqueta que, por si sola, no tiene ninguna representación en la página. Es muy habitual usarla justamente para lo que hemos hecho en el anterior ejemplo, separar partes del contenido de texto de una etiqueta, para aplicarle estilos determinados a esa parte de dentro de la etiqueta.

Estilo definido para una etiqueta
De este modo podemos hacer que toda una etiqueta muestre un estilo determinado. Por ejemplo, podemos definir un párrafo entero en color rojo y otro en color azul. Para ello utilizamos el atributo style, que es admitido por todas las etiquetas del HTML.

Nota: Este uso de las CSS podríamos decir que es en realidad el mismo que el anterior. Solo que la etiqueta es de bloque y no una etiqueta inline (en línea) como .

```
<p style="color:#990000">
Esto es un párrafo de color rojo.
</p>
<p style="color:#000099">
Esto es un párrafo de color azul.
</p>
```

Estilo definido en una parte de la página
Con la etiqueta podemos definir secciones de una página y aplicarle estilos con el atributo style, es decir, podemos definir estilos de una vez a todo un bloque de la página.

El uso de la etiqueta div es englobar partes de un documento HTML para que podamos aplicar estilos a todo el grupo de etiquetas, como el posicionamiento, color, borde, tamaño de letra…

```
<div style="color:#000099; font-weight:bold">
<h3>Estas etiquetas van en <i>azul y negrita</i></h3>
<p>
Seguimos dentro del DIV, luego permanecen los etilos
</p>
</div>
```

Hasta aquí hemos visto los usos de las CSS más específicos, que se consiguen usando el atributo style en la etiqueta. Realmente todos los usos anteriores eran el mismo, pero el enfoque era distinto

ya que las etiquetas del HTML que hemos usado tienen distintos alcances. Sin embargo, hay otras formas más avanzadas de usar las CSS, que deberías tener en cuenta porque son todavía más versátiles y recomendadas.

Estilo definido para toda una página
Podemos definir, en la cabecera del documento, estilos para que sean aplicados a toda la página. Es una manera muy cómoda de darle forma al documento y muy potente, ya que estos estilos serán seguidos en toda la página y nos ahorraremos así, generar mayor cantidad de etiquetas HTML colocando el atributo style.

Es común que los estilos declarados se quieran aplicar a distintas etiquetas dentro del mismo documento.

La aplicación de estilos para toda la página, se utiliza para escribir los estilos una vez y usarlos en un número indefinido de etiquetas. Por ejemplo podremos definir el estilo a todos los párrafos una vez y que se aplique igualmente, sea cual sea el número de párrafos del documento. Por último, también tendremos la ventaja que, si más adelante deseamos cambiar los estilos de todas las etiquetas, lo haremos de una sola vez, ya que el estilo fue definido una única vez de manera global.

Estilo definido para todo un sitio web
Una de las características más potentes del desarrollo con hojas de estilos es la posibilidad de definir los estilos de todo un sitio web en una única declaración.

Esto se consigue creando un archivo donde tan sólo colocamos las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas las páginas comparten una misma declaración de estilos, reutilizando el código CSS de una manera mucho más potente.

Este es el modelo más ventajoso de aplicar estilos al documento HTML y por lo tanto el más recomendable. De hecho, cualquier otro modo de definir estilos no es considerado una buena práctica y lo tenemos que evitar siempre que se pueda.

Las ventajas de este modelo de definición de estilos son las siguientes:

- Se ahorra en líneas de código HTML, ya que no tenemos que escribir el CSS en la propia página (lo que reduce el peso del documento y mejora la velocidad de descarga).
- Se mantiene separado correctamente lo que es el contenido (HTML) de la presentación (CSS), que es uno de los objetivos de las hojas de estilo.
- Se evita la molestia de definir una y otra vez los estilos con el HTML y lo que es más importante, si cambiamos la declaración de estilos, cambiarán automáticamente todas las páginas del sitio web.

Esto es una característica muy deseable, porque aumenta considerablemente la facilidad de mantenimiento del sitio web. Si en cualquier momento se desea cambiar el contenido, no tenemos que preocuparnos por los estilos y viceversa, si queremos cambiar el aspecto del sitio web, no necesitamos preocuparnos ni andar editando el contenido.

Incluir estilos con un fichero externo.

1- Creamos el fichero con la declaración de estilos
Es un fichero de texto normal, que puede tener cualquier extensión, aunque le podemos asignar la extensión .css para aclararnos qué tipo de archivo es. El texto que debemos incluir debe ser escrito exclusivamente en sintaxis CSS, es decir, sería erroneo incluir código HTML en el: etiquetas y demás. Podemos ver un ejemplo a continuación.

```
P  {
 font-size : 12pt;
 font-family : arial,helvetica;
 font-weight : normal;
}

H1  {
 font-size : 36pt;
 font-family : verdana,arial;
 text-decoration : underline;
 text-align : center;
 background-color : Teal;
}

TD  {
 font-size : 10pt;
 font-family : verdana,arial;
 text-align : center;
 background-color : 666666;
}

BODY  {
 background-color : #006600;
 font-family : arial;
 color : White;
}
```

2- Enlazamos la página web con la hoja de estilos

Para ello, vamos a colocar la etiqueta <link> con los atributos siguientes:

rel="STYLESHEET" indicando que el enlace es con una hoja de estilos
href="estilos.css" indica el nombre del fichero fuente de los estilos
Veamos una página web entera que enlaza con la declaración de estilos anterior:

```
<!DOCTYPE html>

<html>
<head>
 <link rel="STYLESHEET" href="estilos.css">
 <title>Página con estilos.css incorporado</title>
</head>

<body>
<h1>página con estilos.css</h1>
Esta página tiene un enlace a estilos.css
<br>
```


</body>
</html>

Reglas de importancia en los estilos
Los estilos se heredan de una etiqueta a otra, como se indicó anteriormente. Por ejemplo, si tenemos declarado en el <BODY> unos estilos, en muchos casos, estas declaraciones también afectarán a etiquetas que estén dentro de esta etiqueta.

La herencia de estilos desde padres a hijos no ocurre siempre. Depende del estilo utilizado.

Por ejemplo, el color del texto se hereda en todos los componentes, aunque no ocurre con el borde de un elemento.

Pero las declaraciones de estilos se pueden realizar de múltiples modos y con varias etiquetas, también entre estos modos hay una jerarquía de importancia para resolver conflictos entre varias declaraciones de estilos distintas para una misma porción de página. Se puede ver a continuación esta jerarquía, primero ponemos las formas de declaración más generales, y por tanto menos respetadas en caso de conflicto:

Declaración de estilos con fichero externo. (Para todo un sitio web)
Declaración de estilos para toda la página. (Con la etiqueta <STYLE> en la cabecera de la página)
Definidos en una etiqueta en concreto. (Utilizando el atributo style en la etiqueta en cuestión)

Para definir un estilo se utilizan atributos como font-size, text-decoration... seguidos de dos puntos y el valor que le deseemos asignar. Podemos definir un estilo a base de definir muchos atributos separados por punto y coma.

Ejemplo:
font-size: 10pt; text-decoration: underline; color: black;

(el último punto y coma de la lista de atributos es opcional)

Para definir el estilo de una etiqueta se escribe la etiqueta seguida de la lista de atributos encerrados entre llaves.

Ejemplo:
H1{text-align: center; color:black}

Los valores que se pueden asignar a los atributos de estilo se pueden ver en una tabla en el siguiente capítulo.

Medición y Unidades en CSS

Muchos estos valores son unidades de medida (Unidades CSS), por ejemplo, el valor del tamaño de un margen o el tamaño de la fuente. Las unidades de medida CSS se pueden clasificar en dos grupos, las relativas y las absolutas. Más la posibilidad de expresar valores en porcentaje.

Relativas: Se llaman así porque son unidades relativas al medio o soporte sobre el que se está viendo la página web, que dependiendo de cada usuario puede ser distinto, puesto que existen muchos dispositivos que pueden acceder a la web, como ordenadores o teléfonos móviles. En principio las unidades relativas son más aconsejables, porque se ajustarán mejor al medio con el que el usuario está accediendo a nuestra web. Son las siguientes:

- Fuente actual: em Es la fuente que se esta trabajando por defecto, si la fuente es de 10 puntos y se coloca 2em se estaría trabajando con una fuente de 20 puntos.
- Altura de la letra: ex 1ex sería igual a la altura de la letra de la fuente actual del usuario.
- Píxeles: px varían su tamaño real en base a la resolución de la pantalla del usuario.

Absolutas: Las unidades absolutas son medidas fijas, que deberían verse igual en todos los dispositivos. Como los centímetros, que son una convención de medida internacional. Pese a que en principio pueden parece más útiles, puesto que se verían en todos los sistemas igual, tienen el problema de adaptarse menos a las distintas particularidades de los dispositivos que pueden acceder a una web y restan accesibilidad a nuestro web. Puede que en tu ordenador 1 centímetro sea una medida razonable, pero en un móvil puede ser un espacio exageradamente grande, puesto que la pantalla es mucho menor. Se aconseja utilizar, por tanto, medidas relativas.

- Puntos pt Un punto equivale a 1/72 pulgadas
- Pulgadas in
- Centímetros cm
- Milímetros mm
- Picas pc (equivale a 12 puntos)

Porcentaje: el porcentaje se utiliza para definir una unidad en función de la que esté definida en un momento dado. Imaginemos que estamos trabajando en 12pt y definimos una unidad como 150%. Esto sería igual al 150% de los 12pt actuales, que equivale a 18pt.

Colores en CSS

Los colores se expresan con valores RGB, igual que los que conocemos para los colores HTML. Con la salvedad que un color se puede especificar también con tres números hexadecimales, en lugar de 6, como era obligatorio en HTML.

Por ejemplo #fff equivaldría a #ffffff, o #123 sería #112233.

Además, los colores se pueden especificar también en valores RGB decimales, con la notación rgb(r,g,b), siendo los valores de r, g, b números entre 0 y 255, por ejemplo rgb(100,0,255).

Otra notación posible es

rgb(r%,g%,b%)

, siendo cada uno de los r%,g%, b% un valor entre 0 y 100, por ejemplo rgb(100%,50%,0%), que sería todo de rojo, la mitad de verde y cero de azul.

Porque lo más habitual es que especifiquemos un color con su valor RGB, de una manera similar a como aprendimos a definir colores en HTML. Pero en CSS tenemos otras maneras de declarar colores que pueden interesarnos, como mínimo para poder entender el código CSS cuando lo veamos escrito.

Notación hexadecimal RGB
Esta notación es la que ya conocemos. Se especifican los tres valores de color (rojo, verde y azul) con valores en hexadecimal entre 00 y FF.

background-color: #ff8800;
Notación hexadecimal abreviada
Esta notación es muy parecida a la anterior, pero permite abreviar un poco la declaración del color, indicando sólo un número para cada valor rojo, verde y azul. Por ejemplo, para especificar el color de antes (#ff8800) podríamos haber escrito:

background-color: #f80;
Nombre del color
También podemos definir un color por su nombre. Los nombres de colores son en inglés, los mismos que sirven para especificar colores con HTML.

color: red;
border-color: Lime;
Notación de color con porcentajes de RGB
Se puede definir un color por los distintos porcentajes de valores RGB. Si todos los valores están al 100% el color es blanco. Si todos están al 0% obtendríamos el negro y con combinaciones de distintos porcentajes de RGB obtendríamos cualquier matiz de color.

color: rgb(33%, 0%, 0%);
Notación por valores decimales de RGB, de 0 a 255
De una manera similar a la notación por porcentajes de RGB se puede definir un color directamente con valores decimales en un rango desde 0 a 255.

color: rgb(200,255,0);
De entre todas estas notaciones podemos utilizar la que más nos interese o con la que nos sintamos más a gusto. Nosotros en nuestros ejemplos venimos utilizando la notación hexadecimal RGB por habernos acostumbrado a ella en HTML.

Color transparente
Para finalizar, podemos comentar que también existe el color transparente, que no es ningún color, sino que específica que el elemento debe tener el mismo color que el fondo donde está. Este valor, transparent, sustituye al color. Podemos indicarlo en principio sólo para fondos de elementos, es decir, para el atributo background-color.

background-color: transparent;

Márgenes de página

La propiedad CSS margin establece el margen para los cuatro lados. Es una abreviación para evitar tener que establecer cada lado por separado con las otras propiedades de margen: margin-top, margin-right, margin-bottom y margin-left.

También se permiten valores negativos.

```
/* Aplica a todos los cuatro lados */
margin: 1em;

/* Vertical | Horizontal */
margin: 5% auto;

/* Arriba | Horizontal | Abajo */
margin: 1em auto 2em;

/* Arriba | Derecha | Abajo | Izquierda */

margin: 2px 1em 0 auto;

/* Valores globales */
margin: inherit;
margin: initial;
margin: unset;
```

Valores
Acepta uno, dos , tres o cuatro valores de los siguientes:

<length>
Especifica un ancho fijo. Valores negativos son permitidos.

<percentage>
Un <percentage> relativo al ancho del bloque contenedor. Se permiten valores negativos.

auto
auto es reemplazado por algún valor apropiado. Por ejemplo, puede usarse para centrar horizontalmente un elemento bloque.

div { width:50%; margin:0 auto; } centrará el div horizontalmente.

- Un único valor aplicará para todos los cuatro lados.
- Dos valores aplicarán: El primer valor para arriba y abajo, el segundo valor para izquierda y derecha.
- Tres valores aplicarán: El primero para arriba, el segundo para izquierda y derecha, el tercero para abajo.
- Cuatro valores aplicarán en sentido de las manecillas del reloj empezando desde arriba. (Arriba, derecha, abajo, izquierda)

Sintaxis formal
[<length> | <percentage> | auto]{1,4}

HTML

```
<div class="ex1">
  margin:     auto;
  background: gold;
  width:      66%;
</div>
<div class="ex2">
  margin:     20px 0 0 -20px;
  background: gold;
  width:      66%;
</div>
```

CSS

```
.ex1 {
  margin: auto;
  background: gold;
  width: 66%;
}
.ex2 {
  margin: 20px 0px 0 -20px;
  background: gold;
  width: 66%;
}
```

Otro ejemplo

```
margin: 5%;               /* 5% para todos los lados */

margin: 10px;             /* 10px para todos los lados */

margin: 1.6em 20px;       /* 1.6em arriba y abajo, 20px izquierda y derecha */

margin: 10px 3% 1em;      /* 10px arriba, 3% izquierda y derecha, 1em abajo */

margin: 10px 3px 30px 5px; /* 10px arriba, 3px derecha, 30px abajo, 5px izquierda */

margin: 1em auto;         /* 1em arriba y abajo, centrado horizontalmente */

margin: auto;             /* 0px de margen vertical, centrado horizontalmente */
```

Centrado horizontal con margin: 0 auto;
Para centrar algo horizontalmente en navegadores modernos, se utiliza flex, una tecnología más moderna con la sentnecia: display: flex; justify-content: center; .

La propiedad position de CSS especifica cómo un elemento es posicionado en el documento. Las propiedades top, right, bottom, y left determinan la ubicación final de los elementos posicionados.

Tipos de posicionamiento
Un elemento posicionado es un elemento cuyo valor computado de position es relative, absolute, fixed, o sticky. (En otras palabras, cualquiera excepto static).

Un elemento posicionado relativamente es un elemento cuyo valor computado de position es relative.

Las propiedades top y bottom especifican el desplazamiento vertical desde su posición original; las propiedades left y right especifican su desplazamiento horizontal.

Un elemento posicionado absolutamente es un elemento cuyo valor computado de position es absolute o fixed. Las propiedades top, right, bottom, y left especifican el desplazamiento desde los bordes del bloque contenedor del elemento. (El bloque contenedor es el ancestro relativo al cual el elemento está posicionado).

Si el elemento tiene márgenes, se agregarán al desplazamiento. el elemento establece un nuevo contexto de formato de bloque para su contenido.

Un elemento posicionado fijamente es un elemento cuyo valor de position computado es sticky.

Es tratado como un elemento posicionado relativamente hasta que su bloque contenedor cruza un límite establecido (como por ejemplo dando a top cualquier valor distinto de auto), dentro de su flujo principal (o el contenedor dentro del cual se mueve), desde el cual es tratado como "fijo" hasta que alcance el borde opuesto de su bloque contenedor.

La mayoría de las veces, los elementos absolutamente posicionados que tienen su height y width establecidos en auto son ajustados hasta acomodarse a su contenido.

Sin embargo, elementos non-replaced y absolutamente posicionados se pueden crear para llenar el espacio vertical disponible, especificando tanto top como bottom, y dejando height sin especificar (es decir, auto). De igual manera se pueden utilizar para llenar el espacio horizontal disponible especificando tanto left como right, y dando a width el valor de auto.

A excepción del caso anteriormente descrito (de elementos posicionados absolutamente rellenando el espacio disponible):

Si ambos, top y bottom están especificados (técnicamente, no auto), top gana.
Si ambos, left y right, están especificados, left gana cuando es ltr (Inglés, japonés horizontal, etc.) y right gana cuando direction es rtl (Persa, árabe, hebreo, etc.).

Sintaxis
La propiedad position es especificada como una palabra única elegida de la siguiente lista de valores.

Valores

static
El elemento es posicionado de acuerdo al flujo normal del documento. Las propiedades top, right, bottom, left, and z-index no tienen efecto. Este es el valor por defecto.
relative
El elemento es posicionado de acuerdo al flujo normal del documento, y luego es desplazado con relación a sí mismo, con base en los valores de top, right, bottom, and left. El desplazamiento no afecta la posición de ningún otro elemento; por lo que, el espacio que se le da al elemento en el esquema de la página es el mismo como si la posición fuera static. Este valor crea un nuevo contexto de apilamiento, donde el valor de z-index no es auto. El efecto que tiene relative sobre los elementos table-*-group, table-row, table-column, table-cell, y table-caption no está definido.
absolute
El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado relativo a su ancestro posicionado más

cercano, si lo hay; de lo contrario, se ubica relativo al bloque contenedor inicial. Su posición final está determinada por los valores de top, right, bottom, y left.
Este valor crea un nuevo contexto de apilamiento cuando el valor de z-index no es auto. Elementos absolutamente posicionados pueden tener margen, y no colapsan con ningún otro margen.
fixed
El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado con relación al bloque contenedor inicial establecido por el viewport, excepto cuando uno de sus ancestros tiene una propiedad transform, perspective, o filter establecida en algo que no sea none (ver CSS Transforms Spec), en cuyo caso ese ancestro se comporta como el bloque contenedor. (Notar que hay inconsistencias del navegador con perspective y filter contribuyendo a la formación del bloque contenedor.) Su posición final es determinada por los valores de top, right, bottom, y left.
Estos valores siempre crean un nuevo contexto de apilamiento. En documentos impresos, el elemento se coloca en la misma posición en cada página.

sticky
El elemento es posicionado de acuerdo al flujo normal del documento, y luego es desplazado con relación a su ancestro que se desplace más cercano y su bloque contenedor (ancestro en nivel de bloque más cercano) incluyendo elementos relacionados a tablas, basados en los valores de top, right, bottom, y left. El desplazamiento no afecta la posición de ningún otro elmento.
Estos valores siempre crean un nuevo contexto de apilamiento. Nótese que un elemento sticky se "adhiere" a su ancestro más cercano que tiene un "mecanismo de desplazamiento" (creado cuando el overflow es hidden, scroll, auto, o bien overlay), aún si ese ancestro no es el ancestro con desplazamiento más cercano.

HTML
```
<div class="caja" id="uno">uno</div>
<div class="caja" id="dos">dos</div>
<div class="caja" id="tres">tres</div>
<div class="caja" id="cuatro">cuatro</div>
```
CSS
```
. caja {
  display: inline-block;
  width: 100px;
  height: 100px;
  background: red;
  color: green;
}

#dos {
  position: relative;
  top: 20px;
  left: 20px;
  background: black;
}
```

Posicionamiento absoluto
Los elementos posicionados relativamente se mantienen en el flujo normal del documento. Por el contrario, un elemento posicionado absolutamente es removido del flujoñ de esta manera, los demás elementos se posicionan como si el mismo no existiera. El elemento posicionado absolutamente se posiciona relativamente a suancestro posicionado más cercano (es decir, el ancestro más cercano

que no es static). Si no hay ningún ancestro posicionado se ubica relativo al bloque contenedor inicial. En el ejemplo siguiente, la caja "Two" no tiene un ancestro posicionado, por lo tanto se posiciona relativo al <body> del documento.

HTML

```
<div class="caja" id="uno">uno</div>
<div class="caja" id="dos">dos</div>
<div class="caja" id="tres">tres</div>
<div class="caja" id="cuatro">cuatro</div>
```

CSS

```
. caja {
  display: inline-block;
  width: 100px;
  height: 100px;
  background: black;
  color: orange;
}

#two {
  position: absolute;
  top: 20px;
  left: 20px;
  background: green;
}
```

Posicionamiento fijo

El posicionamiento fijo es similar al posicionamiento absoluto, con la excepción de que el bloque contenedor del elemento es el viewport. Esto puede usarse para crear un elemento flotante que se mantiene en la misma posición independientemente del desplazamiento. En el ejemplo siguiente, la caja "Uno" está fijada a 80 pixels del límite superior de la página y 10 pixels a la izquierda. Aún luego de desplazarse, se mantiene en el mismo lugar relativo al viewport.

HTML

```
<div class="vista">
 <p>
 Funcionamiento de un avión
</p>
<div class="caja" id="uno">Los aviones vuelan gracias a la actuación de una serie de fuerzas, tanto en el plano horizontal como en el plano vertical. Para que el aparato se eleve es imprescindible que la fuerza que se produce en el eje vertical (sustentación en lenguaje aeronáutico) supere al peso del avión. Por otra parte, en el eje horizontal y gracias a los motores que expulsan gases, tiene lugar el principio de acción-reacción que provoca una fuerza hacia adelante que vence la resistencia del aire. Cuando el avión asciende y llega a su altura de crucero y a una velocidad constante es porque se ha alcanzado el equilibrio de fuerzas tanto en el eje vertical, en el que la sustentación se iguala al peso, como en el eje horizontal, en el que el empuje del motor es igual a la resistencia que nos ofrece el aire.
```

La magia se produce al conseguir esa fuerza de sustentación. Ahí tenemos que acudir a una serie de principios que lo explican. Básicamente, la sustentación se consigue gracias a las alas del avión. Si

las cortáramos tendríamos lo que se llama el perfil del ala, la sección que tiene el ala por dentro. Esta sección tiene una forma muy eficiente desde el punto de vista aerodinámico. El borde por donde entra el aire según va volando el avión es redondeado y la parte de atrás del perfil es afilada y además está curvada por la parte de arriba (en lenguaje aeronáutico esta parte de arriba se llama extradós y la parte de abajo se llama intradós). Esa curvatura del perfil del ala hace que cuando la corriente de aire se encuentra con ella, se divida en dos caminos, una parte del flujo de aire se va por arriba del ala y otra parte, por abajo. Debido a la curvatura del ala, el camino que tiene que recorrer el flujo que va por arriba es más largo que el que va por debajo. Existe un teorema, el de Bernouilli, que es básicamente de conservación de la energía y que dice que para que esto ocurra, el flujo de aire que va por arriba tiene que ir a más velocidad. Eso implica que haya menos presión que en la parte de abajo donde va a menor velocidad y ejerce más presión. Esa diferencia de presión entre el flujo de aire por arriba y el de abajo genera una sustentación. Aunque esta sustentación debida al principio de Bernouilli no llega a explicar toda la que necesitamos para que el avión se eleve. Para explicar la elevación hay que recurrir a otra serie de principios físicos.

CSS

```
. caja {
  width: 100px;
  height: 100px;
  background: red;
  color: white;
}

#uno {
  position: fixed;
  top: 80px;
  left: 10px;
  background: blue;
    width: 500px;
}

.vista {
  width: 500px;
  height: 300px;

  padding-left: 150px;
    background: red;
}
```

Selectores CSS
Los selectores en CSS nos permiten acceder a cualquier elemento o grupo de elementos, para aplicar estilos en una única declaración y permiten seleccionar aquellos elementos sobre los que se van a aplicar las reglas de estilo.

Dentro del código CSS podemos usar selectores y aplicarles un conjunto de estilos determinado.

Para ello escribimos el selector y colocamos los atributos de estilos encerrados entre llaves:

```
selector {
  color: red;
  width: auto;
}
```

Existen selectores de lo más variados, que permiten ajustar de una manera muy precisa qué elementos se desea seleccionar.

Los más importantes son los siguientes:

• Etiqueta: sirven para seleccionar todos los elementos de una misma etiqueta o tag HTML.

```
h1 {
  font-size: 26px;
}
```

• Clase: selecciona todos los elementos de una clase determinada. (Class de CSS).

```
.destacado {
  font-weight: bold;
  color: orange;
}
```

• Identificador: permiten seleccionar etiquetas individuales por el atributo Id de la etiqueta.

```
#mifomulario {
 border: 1px solid #99c;
}
```

• Atributo: permiten seleccionar todas las etiquetas que tengan un atributo dado, o bien un atributo con un valor determinado.

```
[title] {
  text-decoration: none;
}

[align="center"] {
  border: 1px solid red;
}
```

Además, los selectores se pueden combinar entre sí para conseguir selectores mucho más precisos:

Estos selectores obtienen las imágenes que tengan el atributo alt y los párrafos que tengan la clase "desactivado".

```
img[alt] {
```

```
	border: none;
}

p.desactivado {
	color: #ddd;
}
```

También podemos relacionar los selectores con un espacio y entonces el significado cambia totalmente, ya que se estaría indicando que un elemento tiene que estar dentro de otro.

Este selector aplicaría estilos a todos los elementos <h2> que estén dentro de contenedores que tienen la clase "nota".

```
.nota h2 {
  font-weight: normal;
}
```

También podemos combinar los selectores de CSS usando una coma. Entonces estamos indicando que los atributos de estilo deben aplicarse a los dos selectores por separado.

Así estaríamos indicando que queremos aplicar estilos sobre todos los párrafos y todas las divisiones con la clase "bloque".

```
p, div.bloque {
	margin-bottom: 25px;
}
```

Práctica de Selectores

Selector de Tipo
Selecciona todos los elementos que coinciden con el nombre del elemento especificado.
Ejemplo: input se aplicará a cualquier elemento <input>.

El selector de tipo CSS hace coincidir elementos por nombre de nodo. En otras palabras, selecciona todos los elementos del tipo dado dentro de un documento.

```
/* Todos los <a> elements. */
a {
  color: red;
}
```

Ejercicio
CSS

```
span {
  background-color: skyblue;
}
```

HTML

```
<span>Acá hay un texto con color de fondo</span>
<p>Este es su segundo renglón</p>
<span>Acá la publicidad de Codo a Codo.</span>
```

Selector de clase
Selecciona todos los elementos que tienen el atributo de class especificado.
Sintaxis: .classname
Ejemplo: .index seleccionará cualquier elemento que tenga la clase "index".

El selector de clases de CSS hace coincidir los elementos en función del contenido de su atributo.class

```
/* Todos los elementos de class="espacios" */
. espacios {
  margin: 2em;
}

/* All <li> elements with class="espacios" */
li. espacios {
  margin: 2em;
}

/* Todos los elementos <li> con una clase que incluya ambos "espacios" y  "elegante" */
/* For example, class="elegant retro spacious" */
li.espacios.elegante {
  margin: 2em;
}
```

Ejercicio
CSS

```
.red {
  color: #f33;
}
```

```
.yellow-bg {
  background: #ffa;
}

.fancy {
  font-weight: bold;
  text-shadow: 4px 4px 3px #77f;
}
```
HTML
```
<p class="red">Este texto es rojo.</p>
<p class="red yellow-bg">Este texto es rojo con fondo amarillo.</p>
<p class="red fancy">Este renglón es rojo con sombra.</p>
<p>Este renglón no tiene ningún formato.</p>
```

Selector de ID

Selecciona un elemento basándose en el valor de su atributo id. Solo puede haber un elemento con un determinado ID dentro de un documento.

Sintaxis: #idname
Ejemplo: #toc se aplicará a cualquier elemento que tenga el ID "toc".

El selector de ID de CSS coincide con un elemento en función del valor del id atributo del elemento . Para que el elemento sea seleccionado, su idatributo debe coincidir exactamente con el valor dado en el selector.

```
/* El elemento con id="demo" */
#demo {
  border: red 2px solid;
}
```

Ejercicio
CSS
```
#confondo {
  background-color: skyblue;
}
```
HTML
```
<div id="confondo">Este renglón tiene color de fondo azul</div>
<div>Este renglón no tiene fondo.</div>
```

Selector universal

Selecciona todos los elementos. Opcionalmente, puede estar restringido a un espacio de nombre específico o a todos los espacios de nombres.

Sintaxis: * ns|* *|*
Ejemplo: * se aplicará a todos los elementos del documento.

El selector universal de CSS (*) coincide con elementos de cualquier tipo.
```
/* Seleccionar todos los elementos */
```

```
* {
  color: green;
}
```
A partir de CSS3, el asterisco se puede utilizar en combinación con namespaces:

- ns|*- coincide con todos los elementos en el espacio de nombres ns
- *|* - coincide con todos los elementos
- |* - coincide con todos los elementos sin ningún espacio de nombres declarado

Sintaxis

```
* { style properties }
```
El asterisco es opcional con selectores simples. Por ejemplo, *.warning y .warning son equivalentes.

Ejercicio

CSS
```
* [lang^=en] {
  color: green;
}

*.warning {
  color: red;
}

*#maincontent {
  border: 1px solid blue;
}
```

HTML
```
<p class="warning">
  <span lang="en-us">Esto es verde</span>en una linea roja.
</p>
<p id="maincontent" lang="en-gb">
  <span class="warning">Esto es rojo</span> en una linea verde.
</p>
```

Selector de atributo

Selecciona elementos basándose en el valor de un determinado atributo.

Sintaxis: [attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr$=value] [attr*=value]

Ejemplo: [autoplay] seleccionará todos los elementos que tengan el atributo "autoplay" establecido (a cualquier valor).

El selector de atributos CSS hace coincidir los elementos en función de la presencia o el valor de un atributo determinado.
```
/* <a> elementos que tengan una a  */
a[title] {
  color: purple;
}

/* <a> elementos que vayan a la url "https://ejemplo.com.ar" */
a[href="https://ejemplo.com.ar"] {
  color: green;
```

```
}

/* <a> ejemplos que apunten a una url que contenga "ejemplo" */
a[href*="ejemplo"] {
  font-size: 2em;
}

/* <a> elementos que apunten a una url que termine en ".org" */
a[href$=".org"] {
  font-style: italic;
}

/* <a> elementos que su atributo clase contenga la palabra "logo" */
a[class~="logo"] {
  padding: 2px;
}
```

Sintaxis

1. [attr] Representa elementos con un nombre de atributo de attr .
2. [attr=value] Representa elementos con un nombre de atributo de attr cuyo valor es exactamente value .
3. [attr~=value] Representa elementos con un nombre de atributo de attr cuyo valor es una lista de palabras separadas por espacios en blanco, una de las cuales es exactamente valor .
4. [attr|=value] Representa elementos con un nombre de atributo de attr cuyo valor puede ser exactamente value o puede comenzar con value seguido inmediatamente por un guión, -(U + 002D). A menudo se utiliza para coincidencias de subcódigo de idioma.
5. [attr^=value] Representa elementos con un nombre de atributo de attr cuyo valor está prefijado (precedido) por valor .
6. [attr$=value] Representa elementos con un nombre de atributo de attr cuyo valor tiene el sufijo (seguido) por valor .
7. [attr*=value] Representa elementos con un nombre de atributo de attr cuyo valor contiene al menos una aparición de valor dentro de la cadena.
8. [attr operator value i] Agregar un i(o I) antes del corchete de cierre hace que el valor se compare sin distinción entre mayúsculas y minúsculas (para caracteres dentro del rango ASCII).
9. [attr operator value s] Agregar un s(o S) antes del corchete de cierre hace que el valor se compare entre mayúsculas y minúsculas (para caracteres dentro del rango ASCII).

Ejercicio

Enlaces
CSS

```
a {
  color: black;
}

/* Links internos que comienzan con "#" */
a[href^="#"] {
  background-color: gold;
}

/* Links con "ejemplo" en la URL */
a[href*="ejemplo"] {
```

```
  background-color: silver;
}

/* link con la palabra "sinmayusculas"en la URL,
   sin chequear mayúsculas */
a[href*="insensitive" i] {
  color: blue;
}

/* link con la palabra "ConMAyusculas"en la URL,
   con chequeo de mayúsculas  */
a[href*="ConMAyusculas"] {
  color: orange;
}

/* Links que terminen en ".org" */
a[href$=".org"] {
  color: red;
}

/* Links que comienzan con "https" y terminan en ".org" */
a[href^="https"][href$=".org"] {
  color: green;
}
```

HTML

```
<ul>
  <li><a href="#interno">Link Interno</a></li>
  <li><a href="http://ejemplo.com">Link a ejemplo.com</a></li>
  <li><a href="#sinMayusculas">Sin chequear Mayusculas link</a></li>
  <li><a href="http://ejemplo.org">Ejemplo org link</a></li>
  <li><a href="http://ConMAyusculas.com">ConMAyusculas link</a></li>
  <li><a href="https://ejemplo.org">ejemplo https org link</a></li>
</ul>
```