

Guía CSS

Sitio: [Instituto Superior Politécnico Córdoba](#)
Curso: Programador Full Stack - TSDWAD - 2022
Libro: Guía CSS

Imprimido por: Ezequiel Maximiliano GIAMPAOLI
Día: jueves, 8 septiembre 2022, 4:14

Tabla de contenidos

1. Introducción

2. ¿Qué es CSS?

3. ¿Para qué utilizar CSS?

4. Reglas CSS

5. Selectores

5.1. Selectores básicos

5.2. Selector descendente

5.3. Selector de clase

5.4. Selector de ID

5.5. Selectores avanzados

6. Herencia

7. Agrupar reglas

8. Formas de insertar CSS

8.1. Hojas de estilo en línea

8.2. Hojas de estilo interna

8.3. Hojas de estilo externas

9. Modelo de Cajas

9.1. Dimensiones de las cajas

9.2. Margen

9.3. Relleno o padding

10. Web Responsive

10.1. Vista de cuadrícula

10.2. Media Query

10.3. Flexbox

10.4. Desafío Final

1. Introducción

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento exponencial del lenguaje HTML. Entre la fuerte competencia de navegadores web y la falta de un estándar para la definición de los estilos, se dificultaba la creación de documentos con la misma apariencia en diferentes navegadores.

A mediados de la década de 1990 el [W3C](#), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS, Cascading HTML Style Sheets y la SSP, Stream-based Style Sheet Proposal. La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS, Cascading Style Sheets.

En 1995, el W3C decidió gestionar el desarrollo y estandarización de CSS y añadió el proyecto a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1". (<https://www.ecured.cu/CSS>)

Objetivos

1. Crear páginas web básicas con estilos CSS.
2. Incorporar conocimientos básicos del lenguaje CSS para tener una visión amplia y poder complementarlo con otros lenguajes de front-end tales como HTML.
3. Incorporar los conceptos básicos sobre diseño web responsive en CSS.

2. ¿Qué es CSS?

Hojas de estilo en cascada o CSS por sus siglas en inglés, Cascading Style Sheets, es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos de un documento web.

Características:

- Ahorra trabajo. Se puede controlar el diseño de varias páginas HTML a la vez.
- Se pueden almacenar en archivos *.css

CSS3 es la última versión estándar.



Desafío: Investiga ¿Por qué las hojas de estilo reciben el nombre "en cascada"?

3. ¿Para qué utilizar CSS?

Para definir estilos en los documentos web, incluyendo el diseño, la disposición de los elementos y para responder a las variaciones en los tamaños de pantalla en cuanto a diferentes dispositivos.

Ventajas

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza considerablemente la actualización y mantenimiento.
- Los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea) (<https://www.ecured.cu/CSS>)



Desafío: De acuerdo a lo expresado arriba, CSS nos permite responder a las variaciones en los tamaños de pantalla ¿A qué se refiere? ¿Consideras este punto importante al momento de crear un sitio o aplicación web? Busca en la web un ejemplo de sitio o aplicación web que sea capaz de responder a las variaciones en los tamaños de pantalla.

4. Reglas CSS

CSS define conjunto de reglas que permiten describir cada una de las partes que componen los estilos CSS.

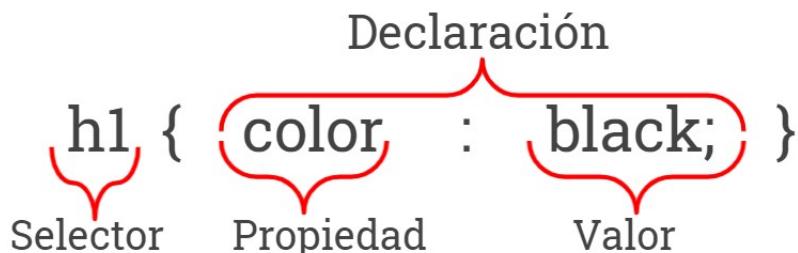


Figura: Sintaxis CSS

- **Selector:** indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración:** especifica los estilos que se aplican a los elementos.
- **Propiedad:** permite modificar el aspecto de una característica del elemento.
- **Valor:** indica el nuevo valor de la característica modificada en el elemento.

5. Selectores

Nos indican a qué elemento HTML hay que aplicar el estilo.

Una misma regla puede aplicarse a varios selectores y, a un mismo selector se le pueden aplicar varias reglas.
Para mayor información referirse a https://www.w3.org/wiki/CSS_Selectors_CSS#Lista_de_selectores_css;



A continuación, los invitamos a mirar un video explicativo:

CSS - Selectores



Visita en el siguiente enlace para acceder a la información sobre selectores que nos provee la W3C: https://www.w3.org/wiki/CSS_Selectors_CSS#Lista_de_selectores_css

5.1. Selectores básicos

Selector universal: Se utiliza para seleccionar todos los elementos de la página.

Selector de tipo o etiqueta: Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector.

5.2. Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos. Se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

La **sintaxis formal** del selector descendente se muestra a continuación:

elemento1 elemento2 elemento3 ... elementoN

Ejemplo:

Si el código HTML de la página es el siguiente:

El selector p span selecciona tanto texto1 como texto2. ¿Por qué sucede esto? Simplemente porque es un selector descendente. Es decir, el elemento no tiene que ser descendiente directo. La única condición es que un elemento debe estar dentro de otro elemento, sin importar el nivel de profundidad en el que se encuentre.

Los selectores descendentes nos permiten aumentar la precisión del selector de tipo o etiqueta. Así, utilizando el selector descendente podemos aplicar diferentes estilos a los elementos del mismo tipo (<https://uniwebsidad.com/libros/css/capitulo-2/selectores-basicos>)

5.3. Selector de clase

Selector de clase o selector Class

Si consideramos el siguiente código HTML de ejemplo:

¿Cómo podemos aplicar estilos CSS sólo al primer párrafo?

El selector universal (*) no se puede utilizar porque seleccionaría todos los elementos de la página. El selector de tipo o etiqueta (p) tampoco se puede utilizar porque seleccionaría todos los párrafos. Por último, el selector descendente (body p) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en utilizar el atributo class de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar:

A continuación, creamos en el archivo CSS una nueva regla llamada **destacado** con todos los estilos que vamos a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, pre fijamos el valor del atributo class con un punto (.) tal como vemos en la siguiente línea:

El selector **.destacado** se interpreta como "cualquier elemento de la página cuyo atributo class sea igual a **destacado**", por lo que solamente el primer párrafo del anterior ejemplo cumple esa condición.

Este tipo de selectores son los más utilizados junto con los selectores de ID que veremos a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden utilizar el mismo valor en el atributo class:

Los **selectores de clase** resultan imprescindibles para diseñar páginas web complejas, ya que nos permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores nos permiten reutilizar los mismos estilos para varios elementos diferentes.

El elemento tiene un atributo **class="error"**, por lo que se le aplican las reglas CSS indicadas por el selector **.error**. Por su parte, el elemento <div> tiene un atributo **class="aviso"**, por lo que su estilo es el que define las reglas CSS del selector **.aviso**.

En ciertos casos, es necesario restringir el alcance del selector de clase. Si utilizamos el ejemplo anterior:

¿Cómo podemos aplicar estilos solamente al párrafo cuyo atributo class sea igual a destacado?

Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

Interpretamos al selector **p.destacado** como "aquellos elementos de tipo <p> que dispongan de un atributo class con valor destacado". De la misma forma, el selector **a.destacado** solamente seleccionaría los enlaces cuyo atributo class sea igual a destacado.

De lo anterior deducimos que el atributo **.destacado** es equivalente a ***.destacado**, por lo que todos los diseñadores obvian el símbolo ***** al escribir un selector de clase normal.

No debemos confundir el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */  
p.aviso { ... }  
  
/* Todos los elementos con atributo class="aviso" que estén dentro  
de cualquier elemento de tipo "p" */  
p .aviso { ... } /*notar el espacio entre p y la clase*/  
  
/* Todos los elementos "p" de la página y todos los elementos con  
atributo class="aviso" de la página */  
  
p, .aviso { ... } /*notar la , (coma) entre p y la clase*/
```

Para completar, es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo class se separan con espacios en blanco.

Al párrafo anterior le aplicamos los estilos definidos en las reglas **.especial**, **.destacado** y **.error**, por lo que en el siguiente ejemplo, el texto del párrafo se vería de color rojo, en negrita y con un tamaño de letra de 15 píxel:

Si un elemento dispone de un atributo class con más de un valor, es posible utilizar un selector más avanzado:

En el ejemplo anterior, el color de la letra del texto es azul y no rojo. Esto se debe a que hemos utilizado un selector de clase múltiple **.error.destacado**, que se interpreta como "aquellos elementos de la página que dispongan de un atributo class con al menos los valores error y destacado" (<https://uniwebsidad.com/libros/css/capitulo-2/selectores-basicos>)

5.4. Selector de ID

Selectores de ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo id no se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS:

En el ejemplo anterior, el selector #destacado solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).

La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. En una misma página, el valor del atributo id debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de id. Sin embargo, el atributo class no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo class.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando queremos aplicar un estilo determinado a varios elementos diferentes de la página HTML.

Al igual que los selectores de clase, en este caso también podemos restringir el alcance del selector mediante la combinación con otros selectores. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo <p> que tenga un atributo id igual al indicado:

A primera vista, restringir el alcance de un selector de ID puede parecer absurdo. En realidad, un selector de tipo p#aviso sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas HTML diferentes.

En este caso, algunas páginas pueden disponer de elementos con un atributo id igual a aviso y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.

No debe confundirse el selector de ID con los selectores anteriores:

code2

5.5. Selectores avanzados

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

```
30 | p > span { color: blue; }
```

```
30 <p><span>Texto1</span></p>
31 <p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector `p > span` se interpreta como "cualquier elemento `` que sea hijo directo de un elemento `<p>`", por lo que el primer elemento `` cumple la condición del selector. Sin embargo, el segundo elemento `` no la cumple porque es descendiente pero no es hijo directo de un elemento `<p>`.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos:

```
32 | p a { color: red; }
33 | p > a { color: red; }
```

```
33 <p><a href="#">Enlace1</a></p>
34 <p><span><a href="#">Enlace2</a></span></p>
```

El primer selector es de tipo descendente y por tanto se aplica a todos los elementos `<a>` que se encuentran dentro de elementos `<p>`. En este caso, los estilos de este selector se aplican a los dos enlaces.

Por otra parte, el selector de hijos obliga a que el elemento `<a>` sea hijo directo de un elemento `<p>`. Por lo tanto, los estilos del selector `p > a` no se aplican al segundo enlace del ejemplo anterior. (<https://uniwebsidad.com/libros/css/capitulo-2/selectores-avanzados>)

task Desafío: A fin de ejercitarnos en el dominio de selectores, te invitamos a jugar [CSS Diner!](#)

Nota: Lee bien las instrucciones a la derecha de la pantalla. Revisa el código HTML ya que, utiliza etiquetas que son propias del juego.

6. Herencia

Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan inicialmente ese mismo valor.

```
1  <!doctype html>
2  <html>
3  <head>
4  <title>Ejemplo de herencia de estilos</title>
5  <style type="text/css">
6  body { font-family: Arial; color: black; }
7  h1 { font-family: Verdana; }
8  p { color: red; }
9  </style>
10 </head>
11 <body>
12 <h1>Titular de la página</h1>
13 <p>Párrafo de texto.</p>
14 </body>
15 </html>
```

7. Agrupar reglas

CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple.

En CSS es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas

```
12  h1, h2, h3 {  
13      color: #8A8E27;  
14      font-weight: normal;  
15      font-family: Arial, Helvetica, sans-serif;  
16  }  
17  h1 { font-size: 2em; }  
18  h2 { font-size: 1.5em; }  
19  h3 { font-size: 1.2em; }
```

8. Formas de insertar CSS

En CSS existen tres formas para insertar los estilos en el documento HTML:

- En línea.
- Interna.
- Externa.

8.1. Hojas de estilo en línea

Utilizamos hojas de estilo en línea cuando incluimos CSS en los mismos elementos HTML.

Esta forma para incluir estilos CSS en documentos HTML es el menos recomendado, ya que es una manera muy dura de establecer estilos. Esta forma es la más específica para introducir estilos y se debe hacer elemento por elemento.

Debido a que los estilos en línea son los estilos más específicos, pueden anular a otros estilos definidos. También niegan uno de los aspectos más importantes de CSS que es la capacidad de crear estilos para muchas páginas web desde un único archivo CSS central para de esta manera hacer que las futuras actualizaciones y cambios de estilo sean mucho más fáciles de gestionar.

Si sólo tuviéramos que utilizar estilos en línea, nuestros documentos se complejizarán rápidamente y se tornarán muy difíciles de mantener. Ya que cualquier cambio en la estética del sitio web significará cambiar los estilos en cada uno de los elementos HTML de cada página.

Esta forma de incluir CSS directamente en los elementos HTML no se debería utilizar, con la excepción de situaciones muy exclusivas.

8.2. Hojas de estilo interna

Utilizamos hojas de estilo interna cuando insertamos CSS en el mismo documento HTML.

Estos estilos se definen en la cabecera del documento HTML, exclusivamente dentro del elemento `<head>` y para definirlos empleamos el elemento `<style>`.

Fundamentalmente empleamos este método cuando definimos un número pequeño de estilos o cuando queremos incluir estilos específicos en una determinada página HTML para que completen los estilos que incluimos por defecto en todas las páginas del sitio web.

El principal inconveniente de usar estilos internos para todo el sitio web, es que si queremos hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que vayamos a modificar.

```
1  <!doctype html>
2  <html>
3  <head>
4  |  <title>Insertar CSS en el mismo documento HTML</title>
5  |  <style type="text/css">
6  |  |  p {
7  |  |  |  color: black;
8  |  |  |  font-family: Verdana;
9  |  |  }
10 |  </style>
11 |  </head>
12 |  <body>
13 |  |  <p>Un párrafo de texto.</p>
14 |  </body>
15 </html>
```

8.3. Hojas de estilo externas

Utilizamos hojas de estilo externas cuando definimos un archivo CSS externo al html.

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS donde las páginas HTML lo enlazan mediante la etiqueta <link>. Un archivo de tipo CSS no es más que un simple archivo de texto cuya extensión es .css.

Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como se necesite.

A continuación veremos el paso a paso para definir CSS externos.

Paso 1 - Creamos un archivo de texto y le añadimos solamente la siguiente línea.

```
10  p { color: black; font-family: Verdana; }
```

Paso 2 - Guardamos el archivo de texto con el nombre estilos.css, luego ponemos especial atención a que el archivo tenga extensión .css y no .txt.

Para esto podemos crear una carpeta que se llame CSS al mismo nivel que la carpeta HTML y guardar el archivo dentro de la carpeta CSS.

Paso 3 - En la página HTML enlazamos el archivo CSS externo mediante la etiqueta <link>

```
1  <!doctype html>
2  <html>
3  <head>
4  <title>Estilos CSS en un archivo externo</title>
5  <link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
6  </head>
7  <body>
8  <p>Un párrafo de texto.</p>
9  </body>
10 </html>
```

A la hora de ejecutar el sitio web en un navegador, cuando el navegador carga la página HTML del ejemplo anterior, antes de mostrar sus contenidos, también descarga los archivos CSS externos enlazados mediante la etiqueta <link> y aplica los estilos a los contenidos de la página.

Normalmente, la etiqueta <link> incluye cuatro atributos cuando enlaza un archivo CSS:

- **rel:** indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet
- **type:** indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es text/css
- **href:** indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- **media:** indica el medio en el que se van a aplicar los estilos del archivo CSS.

De todas las formas de incluir CSS en las páginas HTML, las hojas de estilo externas son las más utilizadas. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con esta forma, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.



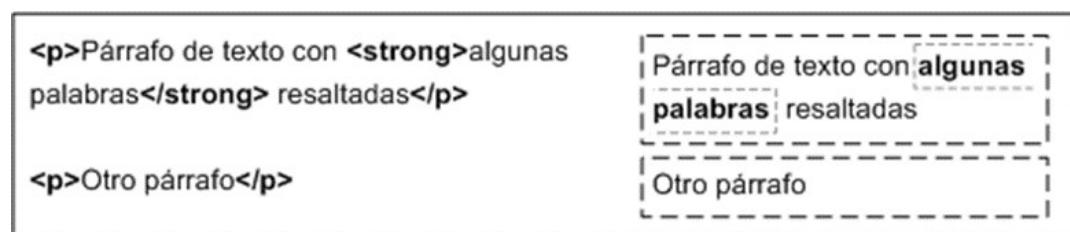
Desafío: ¿Cuál es la diferencia entre estas formas de insertar CSS? ¿Cuál conviene utilizar?

9. Modelo de Cajas

Tal como dijimos en la unidad de HTML, si hacemos una analogía con una estructura de cajas de cartón, podemos decir que hay ciertas cajas que van dentro de otras y ciertas cajas que van una al lado de otra.

El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas sean representados mediante cajas rectangulares.

Cada vez que se inserta un elemento HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La siguiente imagen muestra tres cajas rectangulares que crean los tres elementos HTML de una porción de página de ejemplo.



- **Content (contenido):** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Padding (relleno):** espacio libre opcional existente entre el contenido y el borde.
- **Border (borde):** línea que encierra completamente el contenido y su relleno.
- **Background image (Imagen de fondo):** imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color):** color que se muestra por detrás del contenido y el espacio de relleno.
- **Margin (margen):** separación opcional existente entre la caja y el resto de cajas adyacentes.

9.1. Dimensiones de las cajas

Anchura

Para los elementos de bloque y las imágenes, la propiedad width (anchura) permite establecer la anchura directamente mediante una medida.

Anchura = width

En CSS:

```
4 #lateral { width: 200px; }
```

En HTML:

```
11 | <div id="lateral">...</div>
```

Si se utilizan unidades de medida, los valores indicados no pueden ser negativos. Si en vez de una unidad de medida se indica un porcentaje, la referencia de ese valor es la anchura del elemento que lo contiene. El valor **inherit** indica que la anchura del elemento se hereda de su elemento padre.

Si se establece la anchura de un elemento con la unidad de medida **em**, el valor indicado toma como referencia el tamaño de letra del propio elemento.

El valor **auto** es el valor por defecto de la anchura de todos los elementos. En este caso, el navegador determina la anchura de cada elemento teniendo en cuenta, entre otros, el tipo de elemento que se trata (de bloque o en línea), el sitio disponible en la pantalla del navegador y los contenidos de los elementos.

Altura

Al igual que sucede con width, la propiedad height (altura) no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor **auto** a la altura.

El valor **inherit** indica que la altura del elemento se hereda de su elemento padre. El valor **auto**, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

Altura = height

En CSS:

```
4 #cabecera { height: 60px; }
```

En HTML:

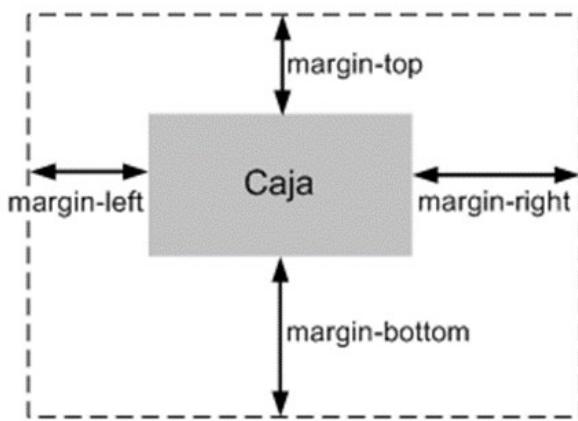
```
11 | <div id="cabecera">...</div>
```

9.2. Margen

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

- **margin-top:** Margen superior
- **margin-right:** Margen derecho
- **margin-bottom:** Margen inferior
- **margin-left:** Margen izquierdo

Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:



Los márgenes verticales (margin-top y margin-bottom) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (margin-left y margin-right) se pueden aplicar a cualquier elemento.

En CSS:

```
4  div img {  
5      margin-top: 0.5em;  
6      margin-bottom: 0.5em;  
7      margin-left: 1em;  
8      margin-right: 0.5em;  
9  }
```

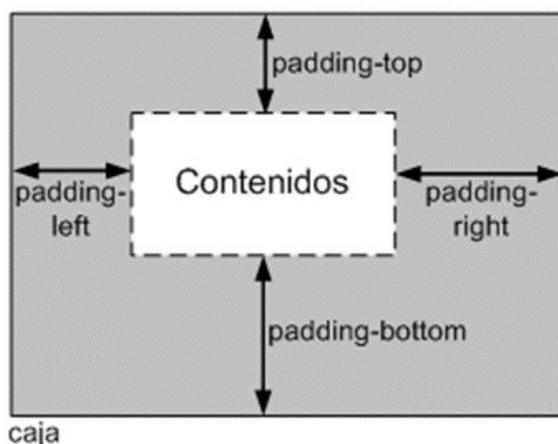
Alternativa:

```
4  div img {  
5      margin: 0.5em 0.5em 0.5em 1em;  
6  }
```

9.3. Relleno o padding

Relleno o padding. CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento.

- padding-top: Relleno superior
- padding-right: Relleno derecho
- padding-bottom: Relleno inferior
- padding-left: Relleno izquierdo



Ejemplo:

```
4  body {  
5    padding: 2em;  
6  } /* Todos los rellenos valen 2em */  
7  body {  
8    padding: 1em 2em;  
9  } /* Superior e inferior = 1em, Izquierdo y derecho = 2em */  
10 body {  
11    padding: 1em 2em 3em;  
12  } /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 2em */  
13  body {  
14    padding: 1em 2em 3em 4em;  
15  } /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
```

10. Web Responsive

El diseño web responsive hace que nuestra página web se vea bien en todos los dispositivos utilizando sólo los lenguajes HTML y CSS. El diseño web responsive no es un programa ni un JavaScript.

Con el diseño web responsive las páginas web se pueden ver utilizando muchos dispositivos diferentes: computadoras de escritorio, tabletas y teléfonos. La página web debe verse bien y ser fácil de usar, independientemente del dispositivo.

El concepto fundamental es que las páginas web no deben omitir información para adaptarse a dispositivos más pequeños, sino adaptar su contenido para así poder adaptarse a cualquier dispositivo. En la siguiente imagen observamos este concepto.



Fuente:

https://www.w3schools.com/css/css_rwd_intro.asp

¿Qué es el Viewport?

La ventana gráfica o viewport en inglés, es el área visible para el usuario de una página web y varía según el dispositivo. El viewport será más pequeño en un teléfono móvil que en una pantalla de computadora.

Antes de las tabletas y los teléfonos móviles, las páginas web se diseñaron sólo para pantallas de computadora, y era común que las páginas web tuvieran un diseño estático y un tamaño fijo. Luego, cuando comenzamos a navegar por Internet usando tabletas y teléfonos móviles, las páginas web de tamaño fijo eran demasiado grandes para caber en el viewport. Para solucionar esto, los navegadores, en esos dispositivos, redujeron la página web completa para adaptarse a la pantalla.

Configuración

HTML5 introdujo un método para permitir que los diseñadores web tomen el control del viewport, a través de la etiqueta <meta>. Para ello debemos incluir el siguiente elemento <meta> de ventana gráfica en todas nuestras páginas web:

```
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Esto le da al navegador instrucciones sobre cómo controlar las dimensiones y la escala de la página.

- La propiedad width=device-width establece el ancho de la página para seguir el ancho de la pantalla del dispositivo (que variará según el dispositivo).
- La propiedad initial-scale=1.0 establece el nivel de zoom inicial cuando el navegador carga la página por primera vez.

Tamaño

Debemos tener en cuenta que los usuarios están acostumbrados a desplazarse por los sitios web verticalmente en dispositivos móviles y de escritorio, **¡pero no están acostumbrados a desplazarse horizontalmente!**

Por lo tanto, si el usuario se ve obligado a desplazarse horizontalmente o alejarse, para ver toda la página web, la experiencia del usuario se torna deficiente.

Algunas reglas adicionales a seguir:

1. NO utilizar elementos grandes de ancho fijo: por ejemplo, si una imagen se muestra con un ancho más ancho que la

ventana, puede hacer que la ventana se desplace horizontalmente. Recuerde ajustar este contenido para que se ajuste al ancho del viewport.

2. NO permitir que el contenido dependa de un ancho de ventana en particular para renderizarse adecuadamente. Las dimensiones y el ancho de la pantalla en píxeles varían ampliamente entre dispositivos, el contenido no debe depender de un ancho de ventana en particular para renderizarse bien.
3. Usar media queries para aplicar diferentes estilos para pantallas pequeñas y grandes. Establecer anchos CSS absolutos grandes para elementos de página hará que el elemento sea demasiado ancho para el viewport en un dispositivo más pequeño. En su lugar, considerar usar valores de ancho relativo, como ancho: 100%. Además, tener cuidado con el uso de valores de posicionamiento absolutos grandes. Puede hacer que el elemento se salga del viewport en dispositivos pequeños.

10.1. Vista de cuadrícula

¿Qué es una vista de cuadrícula?

Muchas páginas web se basan en una vista de cuadrícula, lo que significa que la página está dividida en columnas. Usar una vista de cuadrícula es muy útil al diseñar páginas web. Facilita la colocación de elementos en la página.



Fuente: https://www.w3schools.com/css/css_rwd_grid.asp

Una vista de cuadrícula responsive tiene 12 columnas y tiene un ancho total del 100%, y se encogerá y expandirá a medida que cambie el tamaño de la ventana del navegador. Para mayor información acerca de la vista de cuadrículas, sugerimos ver el ejemplo de la W3Schools [Responsive Web Design Grid-View\[1\]](#)

10.2. Media Query

Media query es una técnica CSS introducida en CSS3. Utiliza la regla **@media** para incluir un bloque de propiedades CSS solo si una determinada condición es verdadera.

Una media query es la base principal del responsive web design. De forma sencilla podemos decir que los media queries son un módulo de CSS que nos permiten identificar el medio en el cual se está mostrando nuestro sitio y bajo qué condiciones aplicar estilos específicos.

Usando media queries, podremos saber, por ejemplo, cuándo el sitio se está presentando en una ventana que mide 600px de ancho o menos, o cuando se está enviando a imprimir. Por ejemplo, si la ventana del navegador tiene 600px o menos, el color de fondo será celeste.

```
4 4 < @media only screen and (max-width: 600px) {  
5 5   body {  
6 6     background-color: #lightblue;  
7 7   }  
8 8 }
```

Veamos otro ejemplo. Supongamos que tenemos un div con la clase «caja» y queremos que se muestre con un ancho de 500 píxeles en ventanas grandes, pero que se muestre con un ancho del 100% al ser visto en ventanas de 700 píxeles de ancho o menos. Para ello deberemos hacer lo siguiente.

```
4 .caja {  
5   width: 500px;  
6 }  
7 @media only screen and (max-width: 700px) {  
8   .caja {  
9     width: 100%;  
10 }  
11 }
```

En el siguiente ejemplo, notar que los estilos del div y el párrafo no están dentro de la media query. Eso es porque todo lo que no está dentro de los media queries se aplica de manera predeterminada. En este caso dejamos que el color del fondo del párrafo sea el mismo, independientemente del tamaño de la ventana, para ello dejamos fuera de la media query a esas propiedades.

En HTML:

```
11 <div class="caja">  
12   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
13   Nulla nulla massa, hendrerit non rutrum sed, blandit vestibulum velit.  
14   Cum sociis natoque penatibus et magnis dis parturient montes,  
15   nascetur ridiculus mus. Mauris ut volutpat mauris, et fringilla sem.</p>  
16 </div>
```

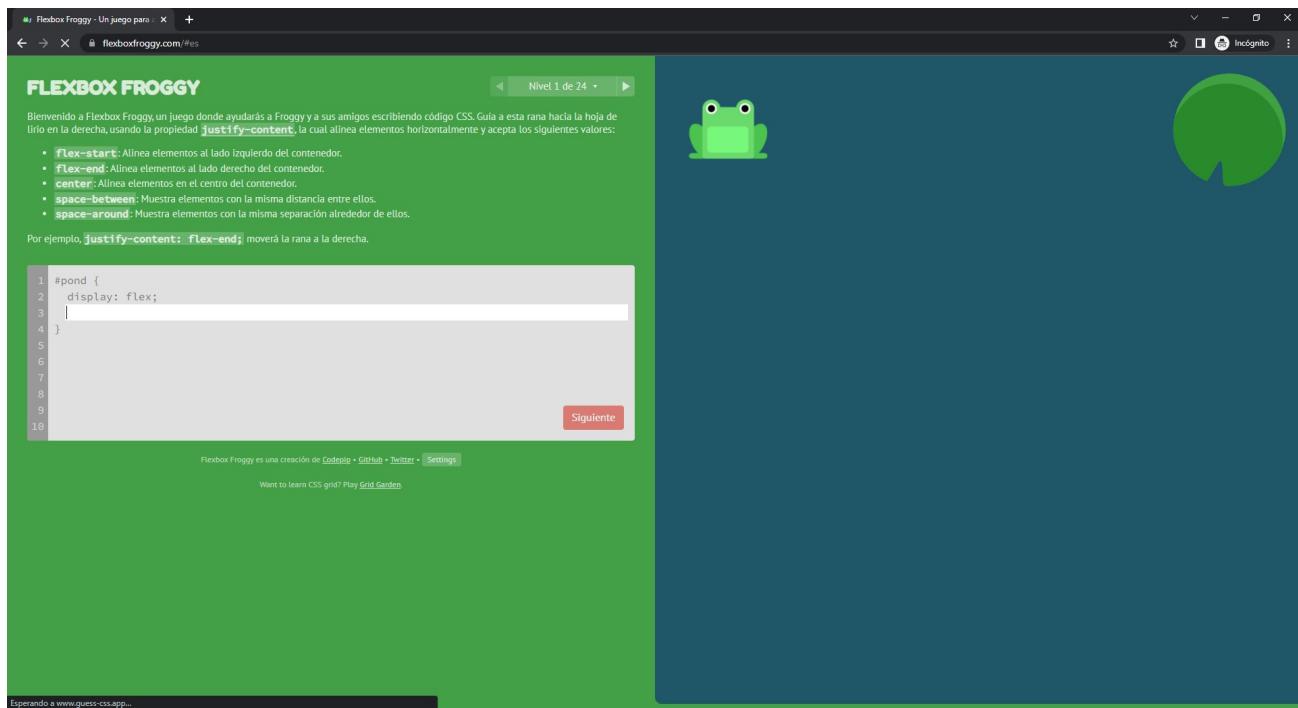
En CSS:

```
4  p {  
5      margin: 0;  
6      padding: 20px;  
7      background: #f1f1f1;  
8  }  
9  
10 .caja {  
11     margin: 0 auto;  
12     width: 500px;  
13  }  
14  
15 @media only screen and (max-width: 700px) {  
16     .caja {  
17         width: 100%;  
18     }  
19 }
```

10.3. Flexbox

El Módulo de Caja Flexible, comúnmente llamado flexbox, fue diseñado como un modelo unidimensional de layout, y como un método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.

task **Desafío:** A fin de aprender flexbox de manera dinámica y divertida, te invitamos a jugar [Flexbox Froggy!!](#)



Consulta la siguiente guía de Flexbos para más información: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

10.4. Desafío Final

Si llegaste hasta aquí! Te invitamos a desafiar tus conocimientos!!

Guess CSS: <https://www.guess-css.app/>

Juego que pone a prueba tus conocimientos en relación a grid layout, selectores y flexbox

The screenshot shows a browser window with the title 'Guess CSS!' at the top. A timer in the center says '✓ 0:0 X'. Below the title, there's a row of three boxes labeled 'c', 'b', and 'a' from left to right, with arrows for navigation between them. Below this row, there are three columns labeled 'CSS A', 'CSS B', and 'CSS C'. Each column contains a snippet of CSS code with some parts highlighted in green. Under each snippet is a button labeled 'THIS!'. At the bottom of the page, there's a code preview showing the HTML structure: <body> <div>a</div> <div>b</div> <div>c</div> </body>. Below the code, there are social media sharing icons for LinkedIn, Facebook, Twitter, GitHub, and DEV, followed by a 'Credits' link.

FLEXBOX ADVENTURE: <https://codingfantasy.com/games/flexboxadventure/play>

Juego para practicar las propiedades de Flexbox. Tiene 3 niveles de dificultades y cada uno tiene a sus vez 24 niveles.

← → ⌂ codingfantasy.com/games/flexboxadventure/play

G8

Games Log In Sign up

Level 1 ▶ Level 1 of 24 ▶

Heroes' health:

 / 
50/100 ❤

Hello, Hero! As you know, we must help Arthur defeat the three evil brothers and return stolen gold! Are you ready for the adventure?

Seems like Arthur lost some health after going through a shock with Bit Coins. Let's help him! On the right you can see Arthur. On the top you can see Arthur's health. And your goal is to restore this health. Sounds easy? Great, let's get started then!

Use the **justify-content** property and place Arthur on the apple.

Probably it'll help to restore some health.

justify-content defines how the browser distributes space between

```
1 #field {  
2   display: flex;  
3   type your answer here...  
4 }  
5  
6  
7  
8
```

Hide heroes Hide grid

Check Answer

