

LIND-2C Programación Web

CSS

Mg. Pablo Ezequiel Inchausti
Ing. David Túa

UNIVERSIDAD DEL CEMA
UCEMA

Jueves 17 de agosto, 09:00 hs

Segundo paso... ¡el estilo!

Nuestro frontend empieza a tomar forma, pero aún es un poco triste. No tiene colores, los elementos no están ubicados donde nos gustaría, los formatos de los botones, las líneas, los párrafos, etc no son los que queremos. En esta sección aprenderemos CSS, que nos permitirá dar estilo a las páginas web acercándonos de a poco a las que ya conocemos.

CSS



¿Qué es CSS? Conceptos generales



**AL IGUAL QUE HTML, TAMPOCO ES
UN LENGUAJE DE PROGRAMACION**

- ☐ Son las siglas de ***Cascading Style Sheets*** (hojas de estilo en cascada)
- ☐ Su versión más reciente, y usada en la actualidad, es la 3
- ☐ A diferencia de HTML, no es un lenguaje de marcado sino un lenguaje de estilo
- ☐ Es un estándar que permite aplicar estilos de manera selectiva a elementos en documentos HTML. La evolución de dicho estándar está a cargo del consorcio W3C
- ☐ Mediante CSS podremos aplicar estilos a una página, por ejemplo colores, formatos, ubicación de los componentes, etc.
- ☐ Todos los navegadores saben interpretar CSS y aplicar el estilo allí descripto a páginas HTML
- ☐ Los archivos tienen usualmente extensión css
- ☐ En el archivo html, se especificará la ubicación de la hoja de estilos que aplica a la página
- ☐ Se pueden escribir con cualquier programa editor de textos. Nosotros utilizaremos un IDE

Asignación de hoja de estilos

La forma más usual para especificar la ubicación de la hoja de estilos de la página es mediante la siguiente etiqueta, que debe ser incorporada dentro del <head> de la página.

```
<link href="estilos/estilos.css" rel="stylesheet" type="text/css">
```

Analicemos su contenido:

- La etiqueta <link> nos permite vincular la hoja de estilos a la pagina web. La misma no necesita etiqueta de cierre.
- Mediante href="estilos/estilos.css" estamos definiendo la ubicación del archivo de estilos. Si bien la carpeta "estilos" y el archivo "estilos.css" son dos estándar, la hoja de estilos puede tener cualquier nombre y estar ubicada en cualquier carpeta accesible.
- El atributo rel="stylesheet" le indica al navegador que se está vinculando una hoja de estilos.
- Finalmente, "type="text/css" especifica que el archivo tiene formato texto y especificación css

Anatomía de una regla CSS

Para comprender el formato en que debemos asignar los estilos, veamos el siguiente ejemplo. La estructura que se visualiza en el ejemplo representa una regla css.



Analicemos en detalle cada una de las partes de esta regla:

- **Selector:** indica el elemento HTML al que deberá aplicar el estilo. En este caso, el estilo se aplica a todos los elementos `<p>`.
- **Declaración:** determina el estilo a aplicar
- **Propiedad:** qué atributo de estilo queremos modificar
- **Valor de propiedad:** el valor que tomará el atributo de estilo

Anatomía de una regla CSS

Veamos ahora esta regla en acción. Armaremos una página html muy sencilla y le vincularemos una hoja de estilos con la regla recién ejemplificada.

```
index.html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width,initial-scale=1.0">
6   <title>Aplicando estilos a la página</title>
7   <link href="estilos/estilos.css" rel="stylesheet" type="text/css">
8 </head>
9
10 <body>
11   <h1> Bienvenidos al primer ejemplo </h1>
12
13   <p>
14     Lorem ipsum dolor sit amet, consectetur adipiscing elit.
15     Phasellus volutpat pretium mauris. Nullam vitae egestas tortor,
16     fringilla sodales eros. Phasellus convallis tristique leo, in facilisis
17     urna eleifend quis. Donec magna quam, malesuada lobortis porttitor ac,
18     eleifend eu lectus. In laoreet sed risus at vehicula. Maecenas in sodales est.
19     Vestibulum condimentum sem nisi, in laoreet diam euismod tincidunt.
20   </p>
21
22   <p>
23     In orci dui, efficitur feugiat gravida et, congue rhoncus lectus.
24     Maecenas porta, nulla et cursus ornare, nibh quam maximus nisi,
25     at facilisis nunc purus vitae justo. Suspendisse tempor tortor sodales
26     nisl varius placerat. Vivamus rutrum vestibulum nibh, vel sagittis
27     justo auctor at. In eu rutrum mi. Sed at magna facilisis, fermentum
28   </p>
29 </body>
30 </html>
```

```
estilos.css
1 p {
2   color: red;
3 }
```



En la página se visualiza que el estilo sólo vio afectado al contenido de la etiqueta `<p>` y no tuvo ningún efecto sobre la etiqueta `<h1>` que permanece en negro.

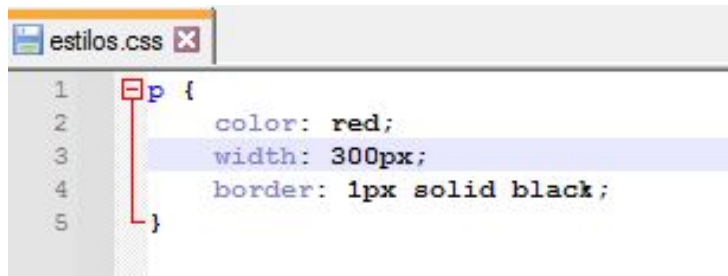
Formas de asignar estilos

Hemos visto la forma más común de asignar estilos (y la que usaremos de aquí en más), que es la que se realiza mediante el uso de un archivo externo. Pero es importante conocer que existen dos formas más de realizarlo.

Tipo	Forma	Ejemplo	Ventajas	Desventajas
CSS externo	En un archivo independiente al html	<pre><link rel="stylesheet" type="text/css" href="style.css"></pre>	Modularización: ambos archivos (html y css) tendrán una estructura más limpia. Reutilización: podremos utilizar una misma hoja de estilos para varias páginas.	Si demora la carga de la hoja de estilos, es probable que se demore la aplicación de mismos. Si hay que cargar varias hojas de estilos, es probable que exista un delay mayor.
CSS interno	Dentro del archivo html, en la sección <head>	<pre><head> <style> p { color: red; } </style> </head></pre>	Se carga en el mismo momento en que se levanta el html, por lo que no hay delay entre uno y otro.	Aumenta el tamaño de la página por lo que demora su carga. Además, cualquier modificación a los estilos implicará modificar la página. No podremos reutilizar estilos.
CSS inline	En una elemento específico del <body> del html	<pre><p style="color:red;"> Esto es un párrafo </p></pre>	Podremos insertar fácil y rápidamente reglas CSS en una página, por lo cual este método es útil para probar o previsualizar cambios y realizar correcciones rápidas	Añadir reglas CSS a cada elemento HTML lleva mucho tiempo y desordena la estructura HTML.

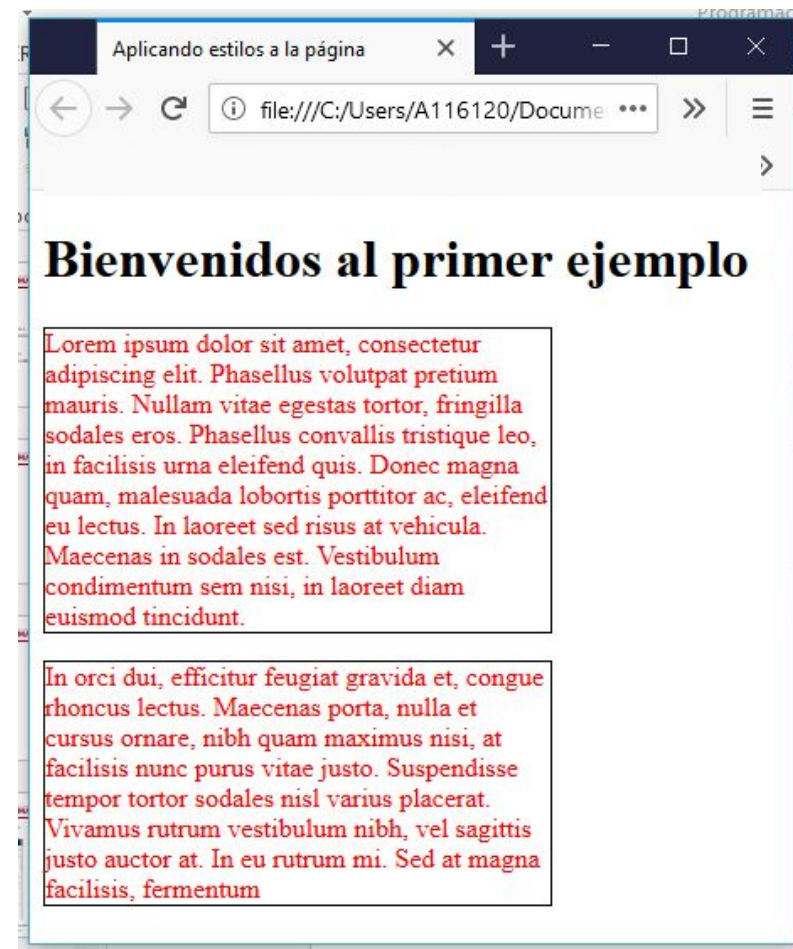
Aplicando más reglas a un mismo selector

Si quisiéramos aplicar más de una regla al selector, debiéramos separarlas mediante un punto y coma de acuerdo a la siguiente sintaxis:



```
1 p {  
2   color: red;  
3   width: 300px;  
4   border: 1px solid black;  
5 }
```

En este ejemplo, estamos modificando 3 atributos: el color, el ancho del párrafo y el borde alrededor del mismo.



Aplicando una regla a más de un selector

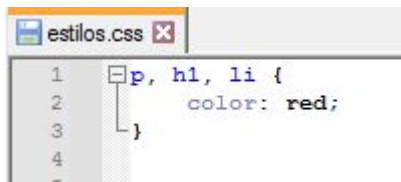
Si quisiéramos en cambio aplicar una regla a más de un selector deberemos separarlos mediante una coma, como se ve en el siguiente ejemplo:

```
<body>
  <h1> Bienvenidos al primer ejemplo </h1>
  <h2> Este texto quedará en negro </h2>

  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    Phasellus volutpat pretium mauris. Nullam vitae egestas tortor,
    fringilla sodales eros. Phasellus convallis tristique leo, in facilisis
  </p>

  <ul>
    <li> Primer elemento </li>
    <li> Segundo elemento </li>
    <li> Tercer elemento </li>
  </ul>

  <p>
    In orci dui, efficitur feugiat gravida et, congue rhoncus lectus.
    Maecenas porta, nulla et cursus ornare, nibh quam maximus nisi,
  </p>
</body>
```



En este ejemplo, estamos modificando el atributo color a los párrafos, los títulos h1 y los elementos de lista.



Tipos de selectores

En los ejemplos hemos visto que podemos aplicar estilos a una etiqueta html. Para ejemplificar utilizamos las etiquetas <p>, <h1> y pero podríamos haber utilizado cualquier otra. Sin embargo, podremos aplicar estilos a otros tipos de selectores de acuerdo a la tabla que se muestra a continuación.

Nombre del selector	Qué selecciona	Ejemplo
Selector de elemento o etiqueta	Todos los elementos HTML de la etiqueta especificada	p Selecciona <p>
Selector de identificación	El elemento en la página con el id especificado (recordemos que sólo debiera existir uno por página).	#id-cliente Selecciona la etiqueta cuyo atributo id sea igual a id-cliente. Ej:
Selector de clase	Los elementos en la página con la clase especificada (recordemos que una clase puede aparecer varias veces en una página).	.boton Selecciona todas las etiquetas cuyo atributo class sea igual a boton Ej: y <li class="boton">
Selector de atributo	Los elementos en una página que incluyan el atributo especificado.	img[src] selecciona pero no
Selector de pseudoclase	Los elementos especificados, pero solo cuando estén en el estado especificado, por ejemplo cuando el puntero esté sobre él.	a:hover Selecciona <a>, pero solo cuando el puntero esté sobre el enlace.

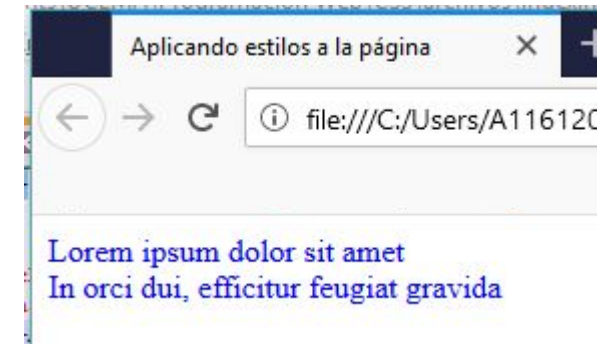
¿Por qué cascada?

Al comienzo de la sección aprendimos que CSS es un acrónimo de **Cascading Style Sheets**, que quiere decir “hojas de estilo en cascada”. Hasta aquí hemos dado estilo a las páginas html mediante las “hojas de estilo”, pero, ¿qué significa que sea “en cascada”? Entender este concepto nos permitirá aprender qué estilo prevalece cuándo varias reglas aplican a un determinado selector.

Comencemos el estudio con el siguiente ejemplo:

```
10 <body>
11   <div>
12     Lorem ipsum dolor sit amet
13   </div>
14
15   <div>
16     In orci dui, efficitur feugiat gravida
17   </div>
18 </body>
```

```
estilos.css  index.html
1  div {
2    color: red;
3  }
4
5  div {
6    color: blue;
7  }
8
```



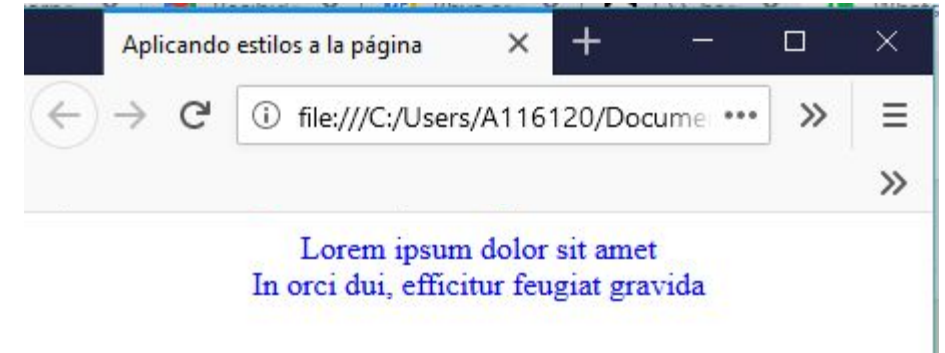
Como se puede visualizar en la salida, el estilo que se aplicó fue el último que aplicó a la etiqueta <div>. Esto nos enseña que, si hay varias reglas que aplican al mismo elemento, prevalece siempre la última regla definida, la cuál **mezcla** y **sobreescribe** las propiedades anteriores.

Mezcla y sobrescribe

Revisemos el concepto de **mezcla** y **sobrescribe**:

```
9
10
11
12
13
14
15
16
17
18
<body>
  <div>
    Lorem ipsum dolor sit amet
  </div>
  <div>
    In orci dui, efficitur feugiat gravida
  </div>
</body>
```

```
estilos.css x index.html x
1  div {
2    color: red;
3    text-align: center;
4  }
5
6  div {
7    color: blue;
8  }
```



Vemos que el texto sigue siendo azul pero ahora quedó centrado. Esto es porque el segundo estilo que aplica a <div> mezcla y sobrescribe, es decir, pisa el valor del atributo “color” pero el resto sigue siendo válido ya que no se sobrescribió (en este caso, el atributo “text-align”).

Veámoslo con un ejemplo un poco más detallado.

```
estilos.css x index.html x
1  div {
2    color: red;
3    text-align: center;
4  }
5
6  div {
7    color: blue;
8    background-color: yellow;
9  }
```

Hubiese sido lo mismo que...

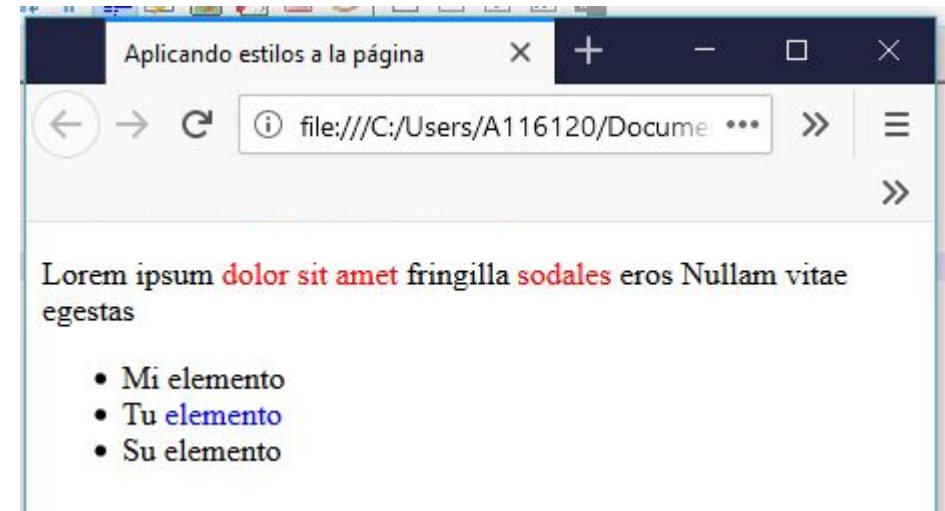
```
5
6  div {
7    color: blue;
8    background-color: yellow;
9    text-align: center;
10 }
11
```

Especificidad

A veces queremos que un estilo aplica específicamente a una etiqueta que se encuentre en determinado lugar del html.

```
<p>
  Lorem ipsum <span> dolor sit amet </span>
  fringilla <span> sodales </span>eros
  Nullam vitae egestas
</p>

<ul>
  <li> Mi elemento </li>
  <li> Tu <span> elemento </span> </li>
  <li> Su elemento </li>
</ul>
```



Este ejemplo nos permite entender el concepto de especificidad. Vemos que los `` de arriba fueron aplicados en rojo ya que se encuentran dentro de una etiqueta `<p>`. Sin embargo, los `` de abajo fueron aplicados en azul porque no están dentro de una etiqueta `<p>` y por lo tanto, tomó el estilo general de los ``.

Si no hubiese estado el segundo estilo para ``, se hubiese mostrado sin estilo.

Especificidad

La especificidad también es importante cuando hay varias reglas que aplican a una misma parte del html.

```
<p id="parrafo" class="parrafo">
  Lorem ipsum dolor sit amet
  fringilla sodales eros
  Nullam vitae egestas
</p>
```

```
index.html | estilos.css
1  p {
2      color: red
3  }
4
5  #parrafo {
6      color: blue
7  }
8
9  .parrafo {
10     color: yellow
11 }
```



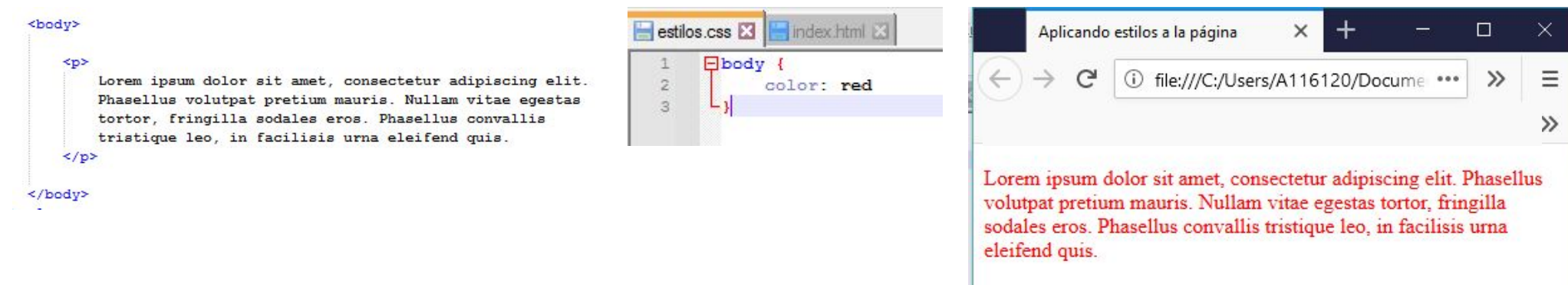
En este caso, ¿por qué prevaleció la regla del “id” del objeto por sobre la clase y la etiqueta? Esto es porque:

- Un selector de elemento es el menos específico
- Un selector de clase es más específico
- Un selector de identificación es aún más específico

Disclaimer: en realidad existe un puntaje que determina la especificidad, en el cual participan varios factores, no sólo los descritos. Sin embargo, esto queda fuera del alcance de la materia.

Herencia

El último concepto que nos ayudará a entender qué regla prevalece es el de herencia dentro de css.



¿Por qué se aplicó color rojo al selector `<p>` si sólo fue aplicado a `<body>`? Esto es por el concepto de herencia: algunas propiedades CSS se heredan desde los elementos padres a los elementos hijos, modificando el valor que tienen por defecto.

Textos y Fuentes

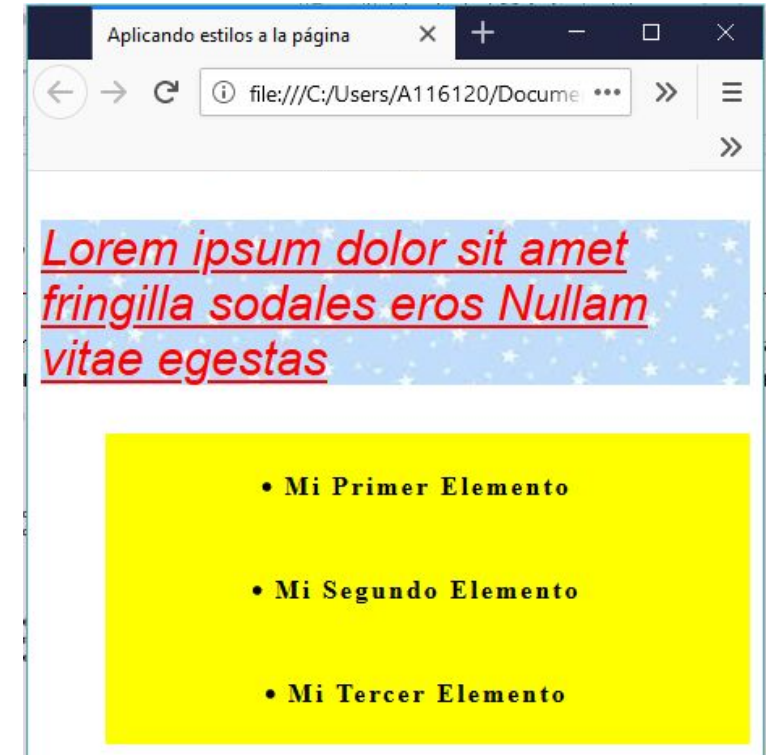
Hasta aquí hemos aprendido el funcionamiento conceptual de CSS. Vamos a trabajar ahora sobre algunos de sus atributos. Decimos algunos porque, dada la gran variabilidad, sería imposible abordar todos. Comenzaremos con los que aplican a los textos.

```
<p>
  Lorem ipsum dolor sit amet
  fringilla sodales eros
  Nullam vitae egestas
</p>

<ul>
  <li> mi primer elemento </li>
  <li> mi segundo elemento </li>
  <li> mi tercer elemento </li>
</ul>
```

```
estilos.css
p {
  color: red;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 30px;
  text-decoration: underline;
  font-style: italic;
  background-image: url("data:image/jpeg;base64");
}

li {
  text-align: center;
  line-height: 4;
  letter-spacing: 2px;
  background-color: yellow;
  text-transform: capitalize;
  font-weight: bold;
}
```



Textos y Fuentes

Atributo	Uso	Valores posibles
color	Color	Puede escribirse con su nombre (red, blue, yellow, green, etc), su código hexadecimal o su combinación RGB. Como ayuda, podemos utilizar https://htmlcolorcodes.com/es/ . Algunos ejemplos hexadecimal: #000000 (ausencia de color: negro), #FFFFFF (blanco)
font-size	Tamaño	Puede expresarse en pixels (px)
font-family	Tipo de letra	Tipos de letras a utilizar. En caso de no encontrar una, utiliza la siguiente. Ej: Courier, "Lucida Console", monospace, "Gill Sans Extrabold", Helvetica, sans-serif
text-decoration	Decoración	none, underline, overline, line-through, solid, double, dotted, dashed, wavy y cualquiera de sus combinaciones
font-style	Estilo	normal, italic, oblique
text-align	Alineación	start, end, left, right, center, justify
line-height	Altura de la línea	Puede expresarse en números (longitud) o en porcentaje (ej: 34%)
letter-spacing	Espaciado entre letras	Puede expresarse en pixels (px)
background-image	Imagen de fondo	url donde se encuentra la imagen. Puede ser local (pc) o remoto (internet)
background-color	Color de fondo	Mismas especificaciones que para el color de la fuente
text-transform	Tipo de capitalización de la letra	Capitalize, uppercase, lowercase, none
font-weight	Ancho de la letra	normal, bold, lighter, bolder, 100, 200, 300, 400, 500, 600, 700, 800, 900

Si bien ya hemos visto la aplicación de estilos a fondos de texto, veremos aquí algunos atributos con mayor profundidad. El atributo background, utilizado para dar estilo a los fondos, es un atributo que a su vez se desagrega en los siguientes: background-attachment, color, image, position, repeat.

Veamos el funcionamiento para los nuevos atributos (color e image ya los hemos visto anteriormente).

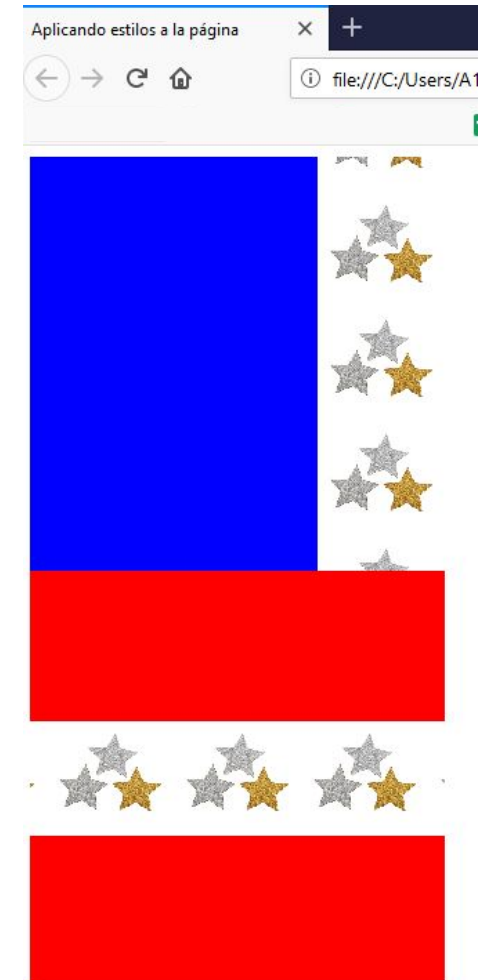
Atributo	Uso	Valores posibles
background-attachment	Determina si la posición de la imagen de fondo será fija dentro de la pantalla o se desplazará con su bloque contenedor	scroll o fixed
position	Define la posición inicial de la imagen de fondo	Top, center, bottom
repeat	Define cómo se repiten los fondos del documento. Un fondo de imagen puede ser repetido sobre el eje horizontal, el eje vertical, ambos ejes, o no estar repetido	repeat-x, repeat-y, repeat, space, round, no-repeat

Fondos

A continuación algunos fondos de ejemplo.

```
<div class="tipo1">  
</div>  
  
<div class="tipo2">  
</div>
```

```
ex.html x estilos.css x  
.tipo1 {  
  height:300px;  
  width: 300px;  
  background-image: url("../img/estrellas.png");  
  background-repeat: repeat-y;  
  background-position: right;  
  background-color: blue;  
}  
  
.tipo2 {  
  height:300px;  
  width: 300px;  
  background-image: url("../img/estrellas.png");  
  background-repeat: repeat-x;  
  background-position: center;  
  background-color: red;  
}
```



Hemos visto que en las listas existe un estilo por defecto, que es un formato de viñeta específico:

- Ejemplo de viñeta por defecto en listas

Sin embargo, podremos modificar este estilo mediante los siguientes atributos:

Atributo	Uso	Valores posibles
list-style-type	Tipo de viñeta	disc, circle, square, decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin, armenian, georgian, lower-alpha, upper-alpha, none
list-style-position	Posición de viñeta	inside, outside
list-style-image	Imagen de viñeta	<url>, none

Listas

A continuación algunos ejemplos.

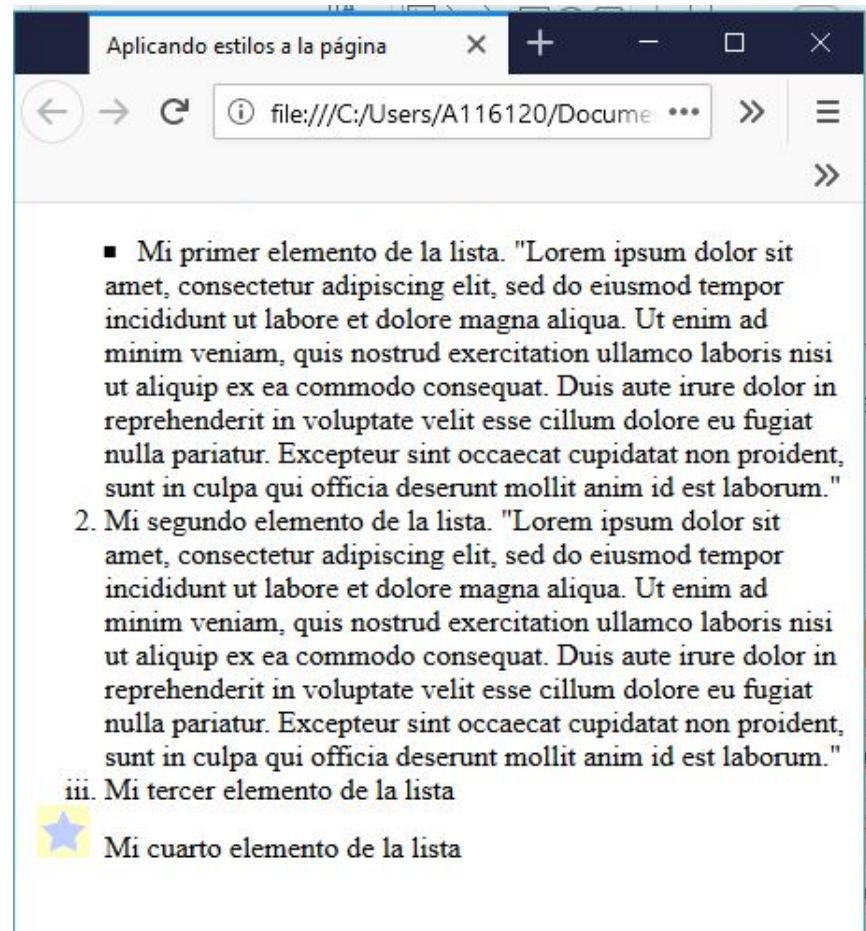
```
<ul>
  <li class="estilo1"> Mi primer elemento de la lista. "Lorem ipsum dolor sit amet, consec
  <li class="estilo2"> Mi segundo elemento de la lista. "Lorem ipsum dolor sit amet, conse
  <li class="estilo3"> Mi tercer elemento de la lista </li>
  <li class="estilo4"> Mi cuarto elemento de la lista </li>
</ul>
```

```
ul .estilo1 {
  list-style-type: square;
  list-style-position: inside;
}

ul .estilo2 {
  list-style-type: decimal;
  list-style-position: outside;
}

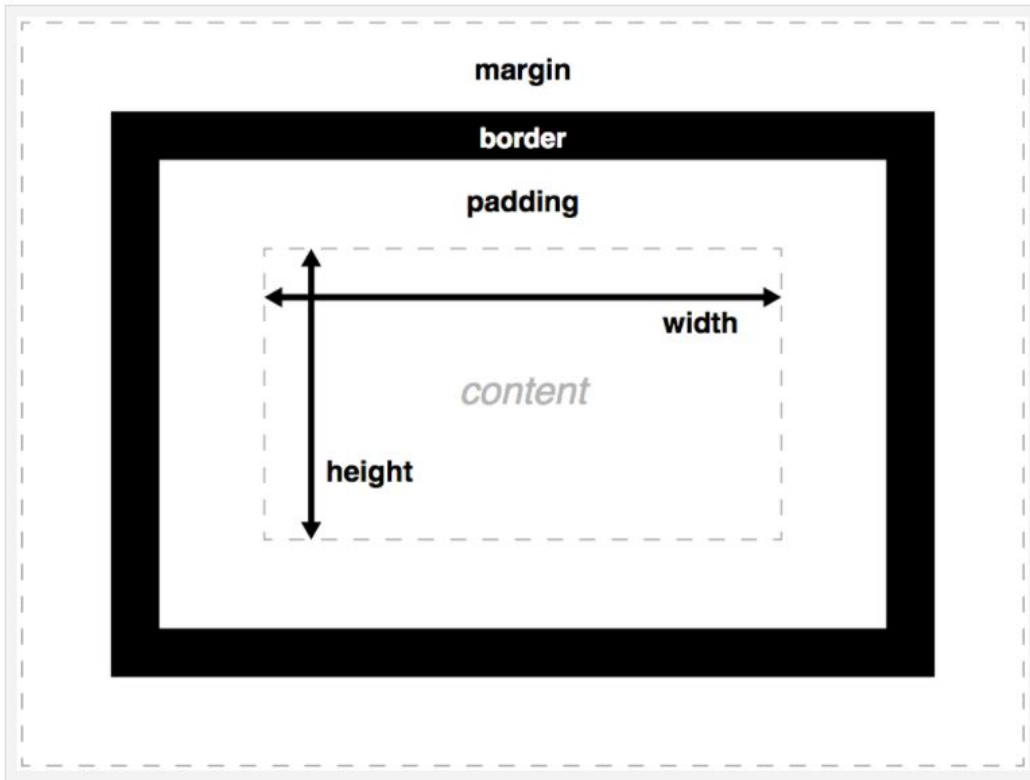
ul .estilo3 {
  list-style-type: lower-roman;
}

ul .estilo4 {
  list-style-image: url("https://mdn.mozillademos.org/files/11981/starsolid.gif");
}
```



Contenedores

Los elementos de un documento HTML son considerados como una caja rectangular invisible al usuario y en la que se pueden aplicar ciertas propiedades para ubicar los elementos respecto a otros.



margin: margen exterior, que separa la caja del elemento de las demás. Si escribimos `margin: 5px;` establecemos un margen de 5 píxeles en los 4 lados de la caja. Podemos especificarlos de forma independiente mediante *margin-top*, *margin-right*, *margin-bottom* y *margin-left*.

padding: margen interno o relleno, que se aplica del borde hacia adentro. Se puede usar de forma independiente también con *padding-top*, *padding-right*, *padding-bottom* y *padding-left*.

Cualquiera de las propiedades anteriores las podemos especificar también en una sola indicando los 4 valores uno detrás de otro, comenzando desde arriba y en sentido horario:

```
padding: 10px 5px 10px 6px;
```

Por tanto, la propiedad anterior especifica un padding de 10px arriba, 5 a la derecha, 10 abajo y 6 a la izquierda.

Un truco para centrar cajas dentro de su contenedor es asignar el valor auto a la derecha y a la izquierda:

```
padding: 10px auto 10px auto;
```

Adicionalmente, se pueden expresar los valores dos a dos si se repiten el de arriba y el de abajo o el de izquierda y derecha:

```
padding: 10px auto;
```

Esta propiedad provocaría un efecto igual a la anterior.

Contenedores. Bordes y Overflow

En cuanto a los bordes de los contenedores, podremos utilizar los siguientes atributos:

- `border-width`: para definir el grosor del borde, por ejemplo, 2px.
- `border-style`: para especificar el tipo de línea del borde. Hay varios tipos, aunque el común es solid: dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden
- `border-color`: para asignar el color del borde.
- `border-radius`: para redondear las esquinas de un element, por ejemplo, 10px.
- `border`: con esta propiedad podemos especificar las tres anteriores en una en el mismo orden que las hemos indicado. El único valor que no podemos omitir sería el segundo, indicando el estilo de línea. Por ejemplo:
 - `border: 4px dotted blue;`
- **overflow**: Con esta propiedad especificamos lo que ocurre cuando el contenido no cabe en la caja. Los posibles valores son:
 - visible. Valor predefinido. Aunque el contenido supere el tamaño de la caja, se mostrará como tal.
 - hidden. Sólo aparece el contenido que no supere el límite de la caja.
 - scroll. Aparece una barra de desplazamiento.
 - auto. Dejamos a criterio del navegador el comportamiento.

Contenedores. Bordes y Overflow

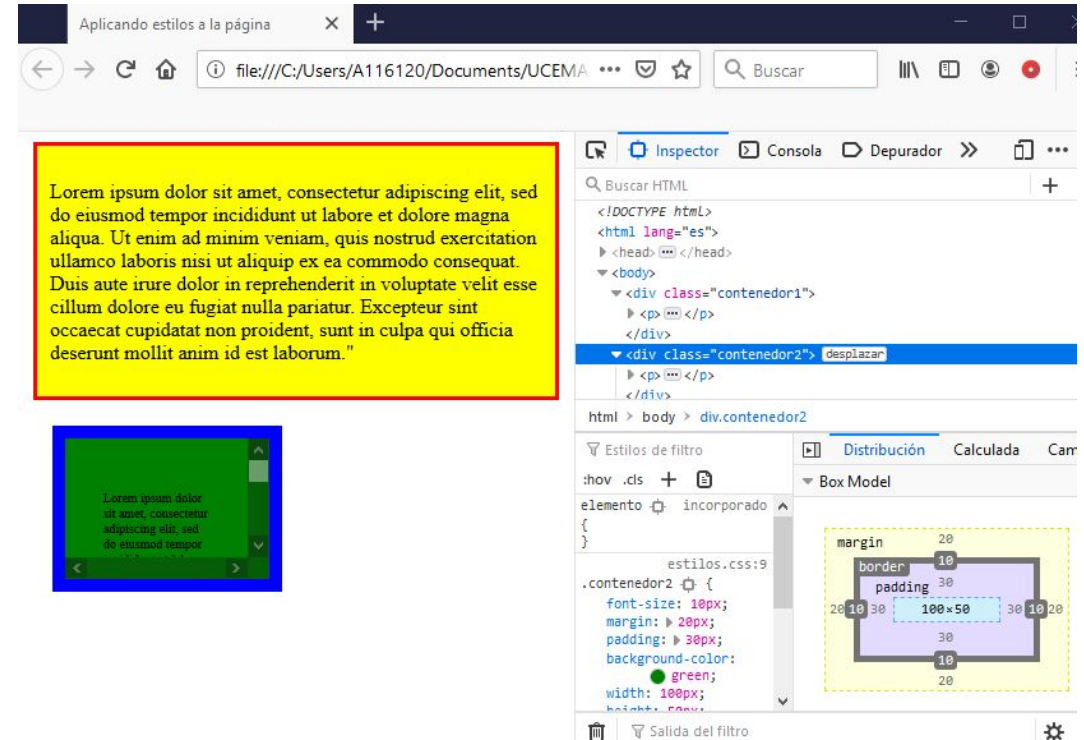
Veamos un ejemplo completo de este tipo de contenedores.

```
<div class="contenedor1">
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing
    elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Ut enim ad minim veniam, quis
    nostrud exercitation ullamco laboris nisi ut aliquip
    ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur. Excepteur sint occaecat cupidatat
    non proident, sunt in culpa qui officia deserunt mollit
    anim id est laborum."
  </p>
</div>

<div class="contenedor2">
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing
    elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Ut enim ad minim veniam, quis
    nostrud exercitation ullamco laboris nisi ut aliquip
    ex ea commodo consequat. Duis aute irure dolor in
    reprehenderit in voluptate velit esse cillum dolore eu
    fugiat nulla pariatur. Excepteur sint occaecat cupidatat
    non proident, sunt in culpa qui officia deserunt mollit
    anim id est laborum."
  </p>
</div>
```

```
.contenedor1 {
  margin: 5px;
  padding: 10px;
  background-color: yellow;
  border-color: red;
  border-style: solid;
}

.contenedor2 {
  font-size: 10px;
  margin: 20px;
  padding: 30px;
  background-color: green;
  width: 100px;
  height: 50px;
  border-width: 10px;
  border-color: blue;
  border-style: solid;
  overflow: scroll;
}
```



Selectores de pseudoclase

En el caso de un selector de pseudoclase, el mismo hará referencia a los elementos especificados sólo cuando estén en un determinado estado. Un ejemplo es el estado “hover”, que se presenta cuando el puntero está sobre él. Veamos un menú de navegación como ejemplo de aplicación.

```
<nav>
  <a href="#"> Clientes </a>
  <a href="#"> Proveedores </a>
  <a href="#"> Opciones </a>
  <a href="#"> Mi Perfil </a>
</nav>
```

```
1  nav {
2    height: auto;
3    width: 400px;
4    padding: 20px;
5    background-color: pink;
6  }
7
8  nav a {
9    padding: 15px;
10   text-decoration: none;
11 }
12
13 nav a:hover {
14   background-color: #B2E4C3;
15 }
```

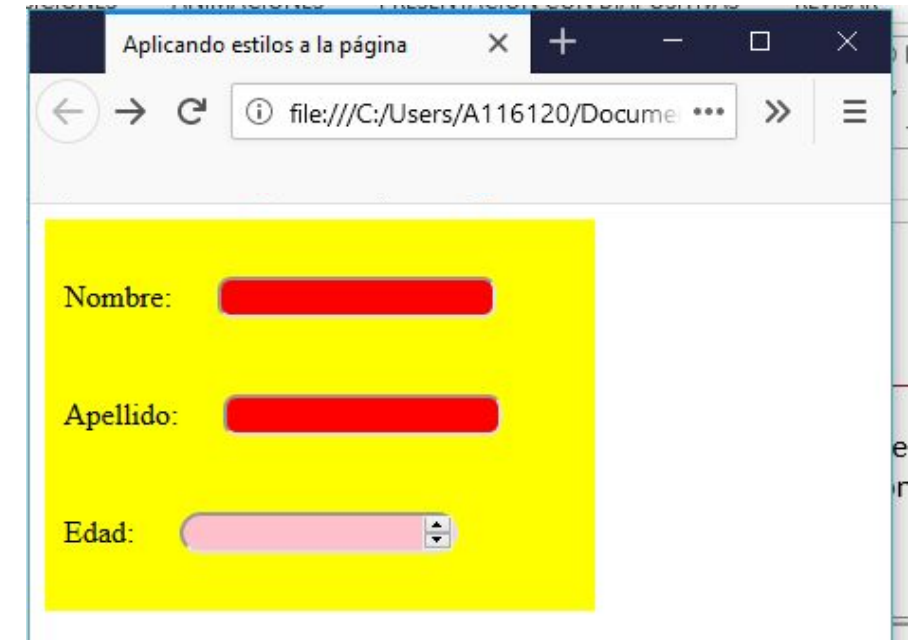


Selectores de atributo

En el caso de un selector de atributo, el mismo hará referencia a los elementos especificados sólo cuando esté presente determinado atributo. Veamos un formulario como ejemplo de aplicación, donde en función al tipo de dato se modifica el color de fondo.

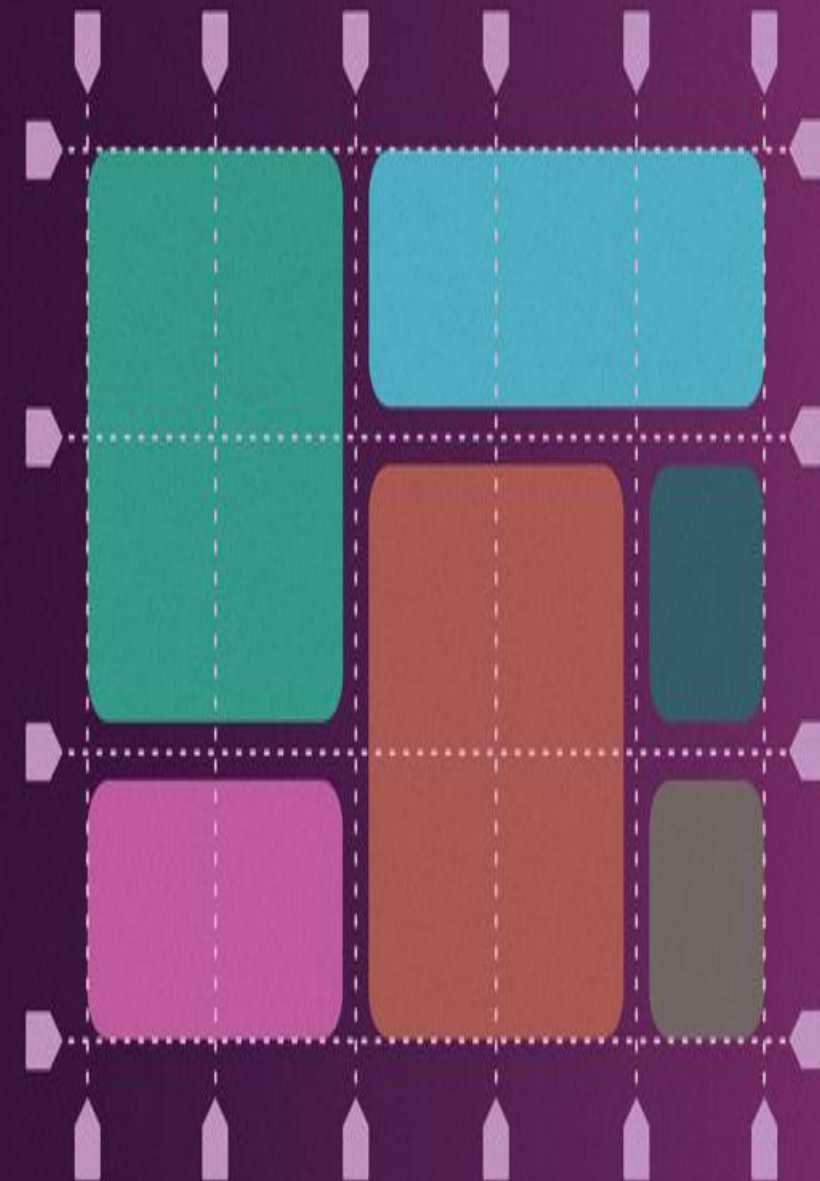
```
<form>  
  Nombre: <input type="text">  
  Apellido: <input type="text">  
  Edad: <input type="number">  
</form>
```

```
1  form {  
2    width: 270px;  
3    background-color: yellow;  
4    padding: 10px;  
5  }  
6  
7  input {  
8    margin: 20px;  
9  }  
10  
11 input[type="text"] {  
12   background-color: red;  
13   border-radius: 7px;  
14 }  
15  
16 input[type="number"] {  
17   background-color: pink;  
18   border-radius: 20px;  
19 }
```



Y cómo lo acomodamos en la pantalla?

El layout de los distintos elementos de una pantalla web no es trivial. Si los ubicamos de acuerdo al funcionamiento por defecto, los veremos de una forma muy distinta a la deseada. Para ello, CSS nos provee de distintas herramientas, una de las cuales es Flexbox. En esta sección aprenderemos su funcionamiento en profundidad.

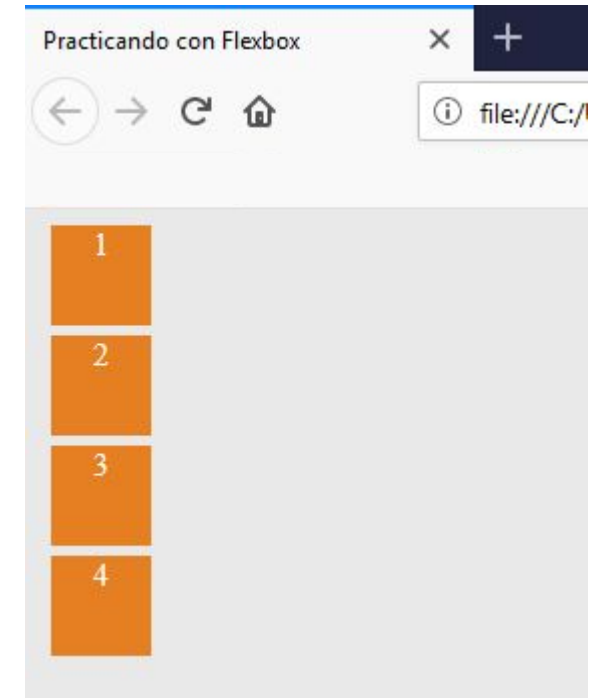


Layout por default

Comencemos con el siguiente ejemplo base:

```
flexbox0.html flexbox0.css
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1.0">
6     <title>Practicando con Flexbox</title>
7     <link href="flexbox0.css" rel="stylesheet" type="text/css">
8   </head>
9
10  <body>
11    <div class="elemento"> 1 </div>
12    <div class="elemento"> 2 </div>
13    <div class="elemento"> 3 </div>
14    <div class="elemento"> 4 </div>
15  </body>
16 </html>
17
```

```
flexbox0.html flexbox0.css
1 body {
2   background: #E9E9E9
3 }
4
5 .elemento {
6   color: #fff;
7   margin: 5px;
8   width: 50px;
9   height: 50px;
10  background: #E67E22;
11  text-align: center;
12 }
```



Como vemos, el funcionamiento por defecto es ubicar los cuadros uno a continuación del otro en distintas líneas. Por otro lado, si modificamos el tamaño de la pantalla (por ejemplo para visualización en tablets o celulares), es probable que alguno de los elementos deje de visualizarse. Entonces, ¿cómo podemos hacer para modificar la distribución de los componentes si quisiéramos, por ejemplo, ubicarlos en la misma línea? Aquí es donde entra en juego flexbox.

Flexbox

Lo primero que debemos saber es que, para usar flexbox, debe existir un **contenedor**. Y dentro de dicho contenedor, existirán **elementos “hijos”**. Incorporemos entonces un contenedor en el código anterior.

```
<body>

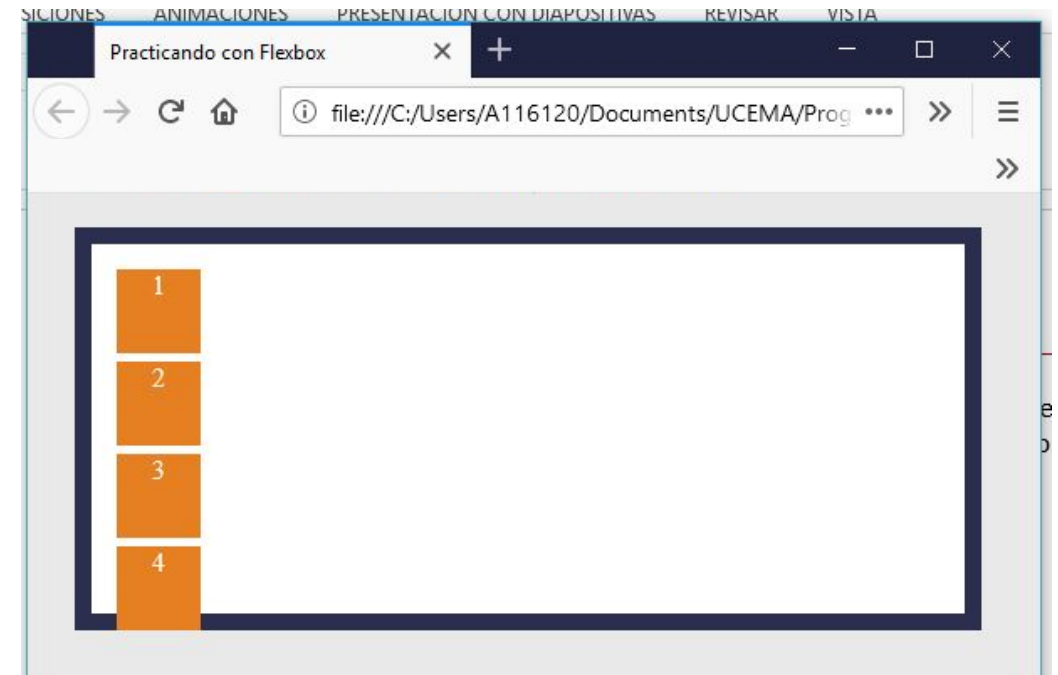
  <div class="contenedor">
    <div class="elemento"> 1 </div>
    <div class="elemento"> 2 </div>
    <div class="elemento"> 3 </div>
    <div class="elemento"> 4 </div>
  </div>

</body>
```

```
4
5  .contenedor {
6    width: 500px;
7    height: 200px;
8    background: #fff;
9    padding: 10px;
10   border: 10px solid #2C2E50;
11   margin: 20px;
12 }
```

El cuadro del contenedor se incorporó sólo a modo ilustrativo para que se visualice la posición relativa de cada uno de los elementos “hijo”. Tengamos en cuenta que cada elemento “hijo” podrá a su vez ser un contenedor de varios elementos.

Como vemos, salvo por el cuadro del contenedor, no se vio modificada la posición de ningún elemento.



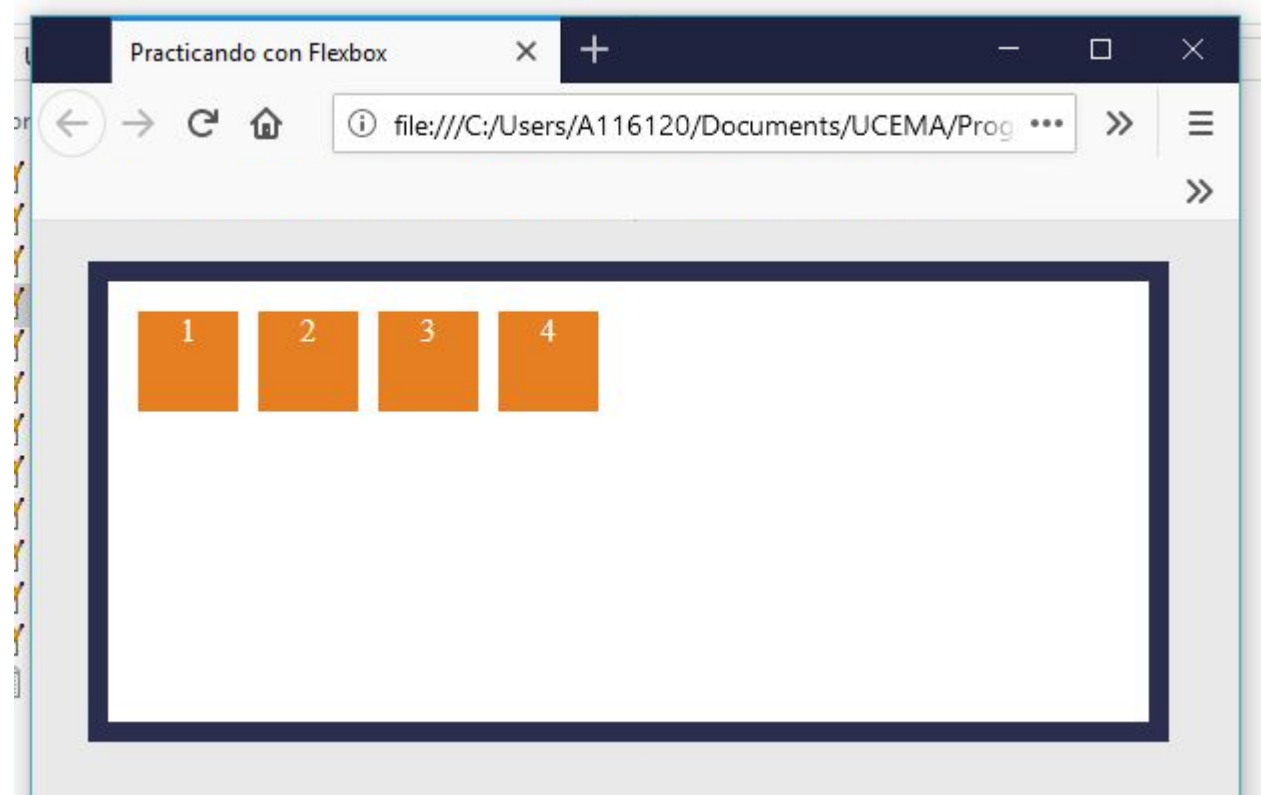
Display

Veamos ahora que sucede si incorporamos un **"display: flex"** al contenedor:

```
<body>
  <div class="contenedor">
    <div class="elemento"> 1 </div>
    <div class="elemento"> 2 </div>
    <div class="elemento"> 3 </div>
    <div class="elemento"> 4 </div>
  </div>
</body>
```

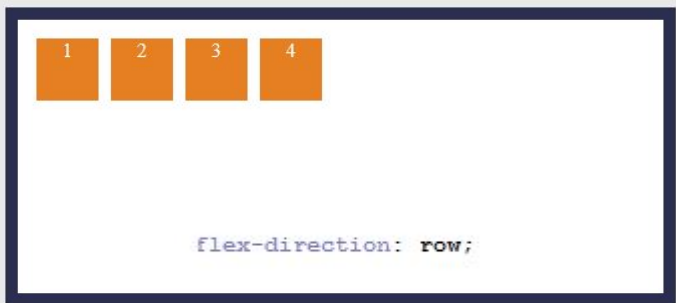
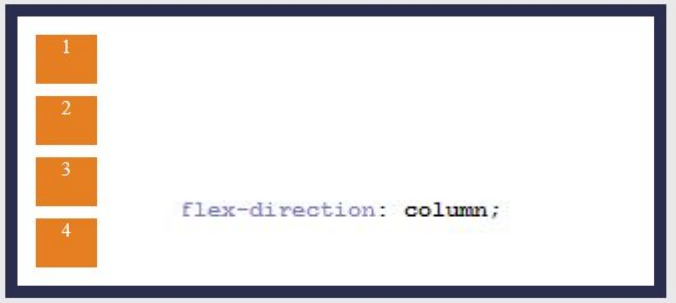
```
.contenedor {
  width: 500px;
  height: 200px;
  background: #fff;
  padding: 10px;
  border: 10px solid #2C2E50;
  margin: 20px;
  display: flex;
}
```

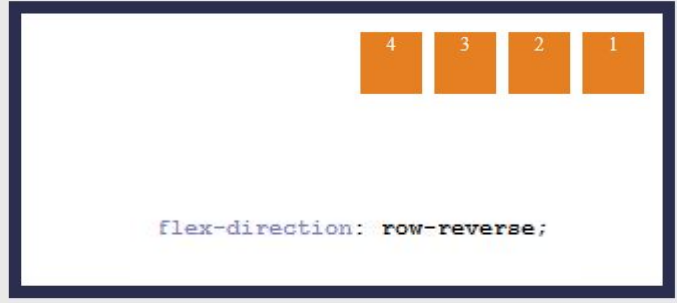
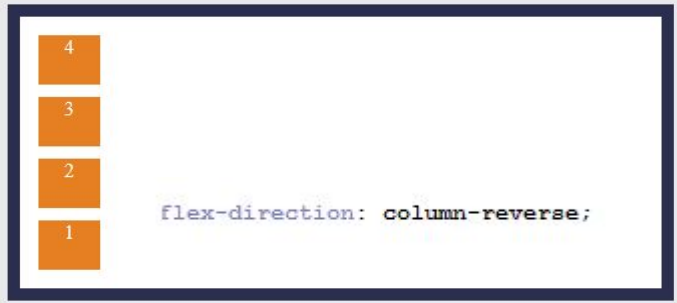
Los elementos "hijos" comienzan a tomar forma dentro del contenedor, pero de acuerdo a su comportamiento por default. Comenzaremos primero conociendo cada uno de los atributos del contenedor, para luego pasar a los atributos de los hijos.



Flex-direction

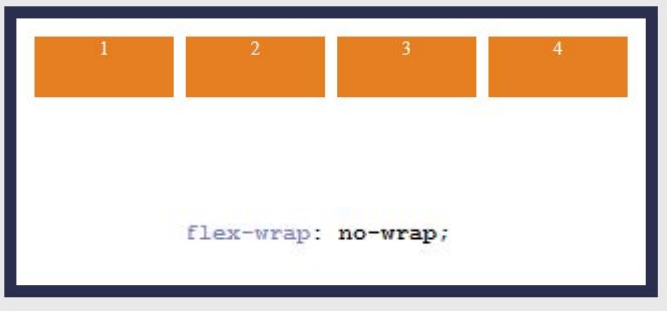
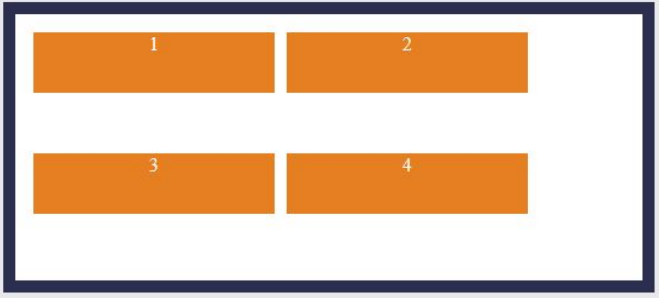
Como recién se dijo, **comenzaremos a ver cada uno de los atributos que aplican al contenedor** y que tienen efecto sobre los elementos allí contenidos. El primer atributo se denomina **flex-direction** y permite designar la dirección en que se mostrarán los hijos dentro del contenedor. La siguiente tabla define su funcionamiento:

Valor	Uso	Layout
row	Incorpora los elementos en fila	 <p>Diagrama que muestra cuatro elementos numerados 1, 2, 3 y 4 dispuestos horizontalmente en fila dentro de un contenedor. El código <code>flex-direction: row;</code> está visible en la parte inferior del contenedor.</p>
column	Incorpora los elementos en columna	 <p>Diagrama que muestra cuatro elementos numerados 1, 2, 3 y 4 dispuestos verticalmente en columna dentro de un contenedor. El código <code>flex-direction: column;</code> está visible en la parte inferior del contenedor.</p>

Valor	Uso	Layout
row-reverse	Incorpora los elementos en fila de derecha a izquierda	 <p>Diagrama que muestra cuatro elementos numerados 4, 3, 2 y 1 dispuestos horizontalmente en fila, pero en orden inverso (de derecha a izquierda) dentro de un contenedor. El código <code>flex-direction: row-reverse;</code> está visible en la parte inferior del contenedor.</p>
column-reverse	Incorpora los elementos en columna de abajo hacia arriba	 <p>Diagrama que muestra cuatro elementos numerados 4, 3, 2 y 1 dispuestos verticalmente en columna, pero en orden inverso (de abajo hacia arriba) dentro de un contenedor. El código <code>flex-direction: column-reverse;</code> está visible en la parte inferior del contenedor.</p>

Flex-wrap

Este atributo permite definir qué sucede con los elementos hijos si no entran en el contenedor. Es sumamente útil en las aplicaciones responsive (aquellas que pueden visualizarse en dispositivos de distinto tamaño). La siguiente tabla define su funcionamiento:

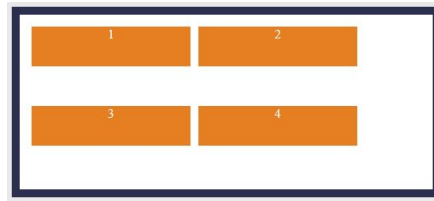
Valor	Uso	Layout
no-wrap	Deja los elementos en una misma fila/columna (no respeta el ancho de los hijos si el mismo supera al contenedor)	 <pre>flex-wrap: no-wrap;</pre>
wrap	Si los elementos no entran en la fila/columna, los pasa a la siguiente	 <pre>flex-wrap: wrap;</pre>

Valor	Uso	Layout
wrap-reverse	Mismo comportamiento que wrap pero empieza desde la última fila/columna	 <pre>flex-wrap: wrap-reverse;</pre>

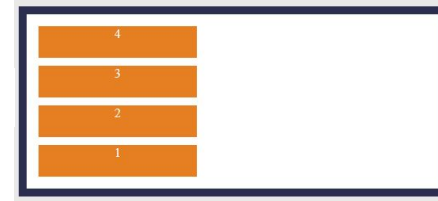
Flex-flow

Este atributo es un atajo de los dos vistos anteriormente: flex-direction y flex-wrap, pudiendo utilizarse ambos o uno solo de ellos. A continuación algunos ejemplos:

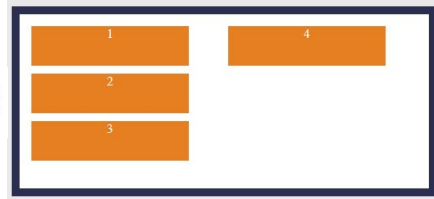
`flex-flow: row wrap;`



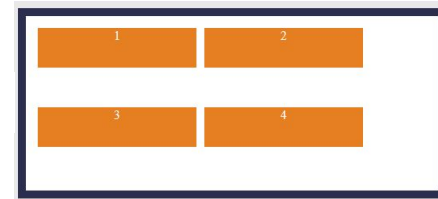
`flex-flow: column-reverse;`



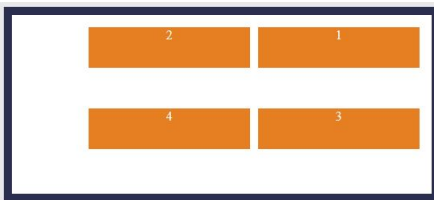
`flex-flow: column wrap;`



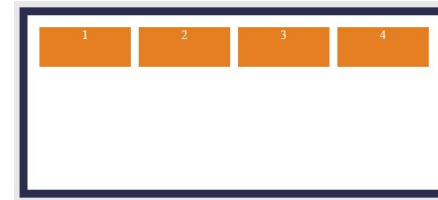
`flex-flow: wrap;`



`flex-flow: row-reverse wrap;`

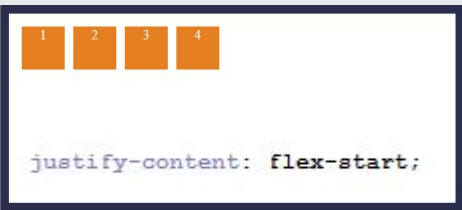




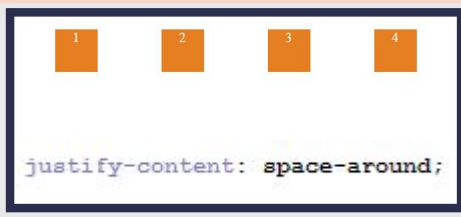
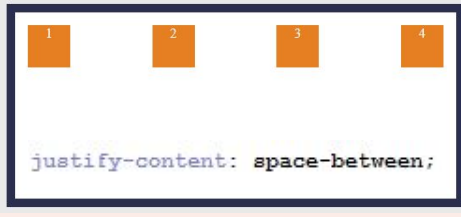
`flex-flow: no-wrap;`



Justify-content

Este atributo permite definir el justificado de los elementos dentro de la fila o columna. La siguiente tabla define su funcionamiento:

Valor	Uso	Layout
flex-start	Justifica hacia el inicio	 <pre>justify-content: flex-start;</pre>
flex-end	Justifica hacia el final	 <pre>justify-content: flex-end;</pre>
center	Centra los elementos en sin dejar espacios intermedios	 <pre>justify-content: center;</pre>

Valor	Uso	Layout
space-around	Centra los elementos dejando espacios intermedios y hacia afuera	 <pre>justify-content: space-around;</pre>
space-between	Centra los elementos dejando sólo espacios intermedios	 <pre>justify-content: space-between;</pre>

Align-items

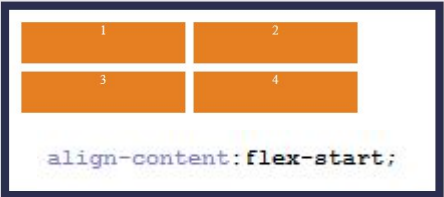
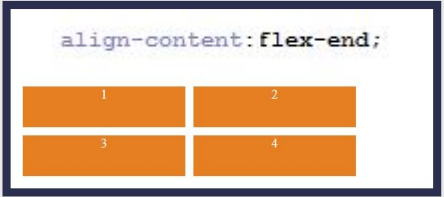
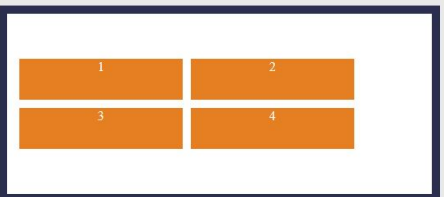
Este atributo permite alinear los ítems verticalmente en el caso de filas y horizontalmente en el caso de columnas. La siguiente tabla define su funcionamiento:

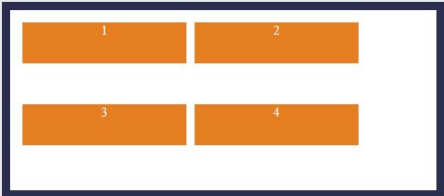

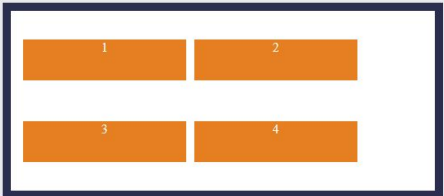
Valor	Uso	Layout
flex-start	Alinea los elementos al inicio	 <pre>align-items: flex-start;</pre>
flex-end	Alinea los elementos al final	 <pre>align-items: flex-end;</pre>
center	Alinea los elementos en el centro	 <pre>align-items: center;</pre>

Valor	Uso	Layout
stretch	Estira los elementos para que ocupen todo el alto o ancho (el height o width, respectivamente, no debe estar fijo)	 <pre>align-items: stretch;</pre>
baseline	Alinea los elementos en la base de la fuente (tiene utilidad cuando las fuentes no son idénticas o bien están posicionadas distinto)	 <pre>align-items: baseline;</pre> <div><pre><div class="elemento elemento2"> 2 </div></pre><pre>.elemento2 { font-size: 30px; }</pre></div>

Align-content

Este atributo permite alinear el contenido en bloque, **cuando ocupan más de una fila o columna**. La siguiente tabla define su funcionamiento:

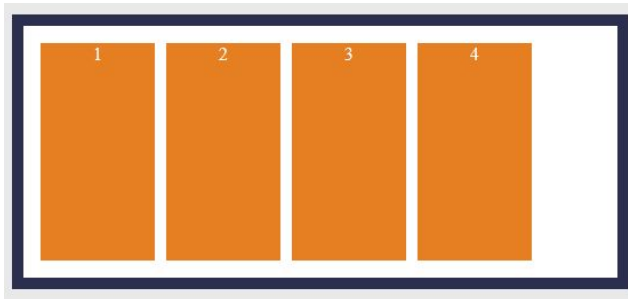
Valor	Uso	Layout
flex-start	Alinea los elementos al inicio	 <pre>align-content: flex-start;</pre>
flex-end	Alinea los elementos al final	 <pre>align-content: flex-end;</pre>
center	Alinea los elementos en el centro	 <pre>align-content: center;</pre>

Valor	Uso	Layout
stretch	Alinea los elementos estirándolos proporcional-mente para que ocupen toda el espacio	 <pre>align-content: stretch;</pre>
space-between	Alinea los elementos dejando un espacio en el medio de ellos	 <pre>align-content: space-between;</pre>
space-around	Alinea los elementos dejando un espacio en el medio y hacia afuera	 <pre>align-content: space-around;</pre>

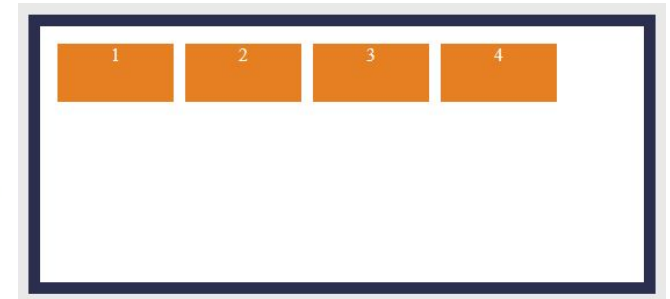
Flex-basis

A partir de aquí comenzaremos a trabajar sobre los atributos que aplican a los hijos, es decir, a los elementos dentro del contenedor. En este caso, el `flex-basis`, que funciona de la misma forma que asignar `width` o `height` pero en relación a la dirección. Entonces, para el caso de dirección “row”, estará asignando el ancho, y para dirección “column”, hará lo propio con la altura. A continuación algunos ejemplos:

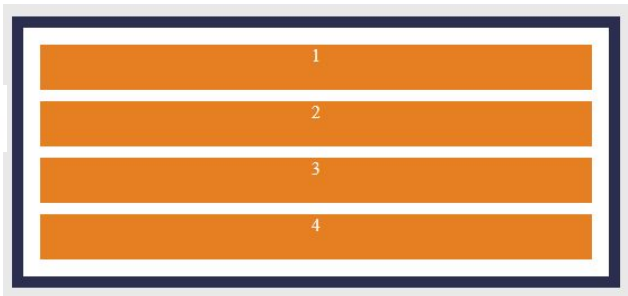
```
flex-direction: row;  
  
flex-basis: 100px;
```



```
flex-direction: row;  
  
flex-basis: 100px;  
height: 50px;
```



```
flex-direction: column;  
  
flex-basis: 100px;
```



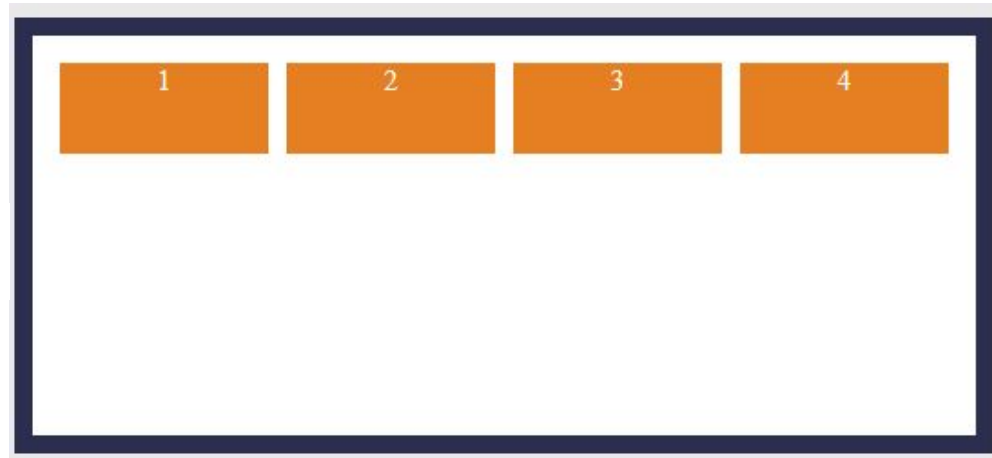
```
flex-direction: column;  
  
flex-basis: 30px;  
width: 20px;
```



Flex-grow

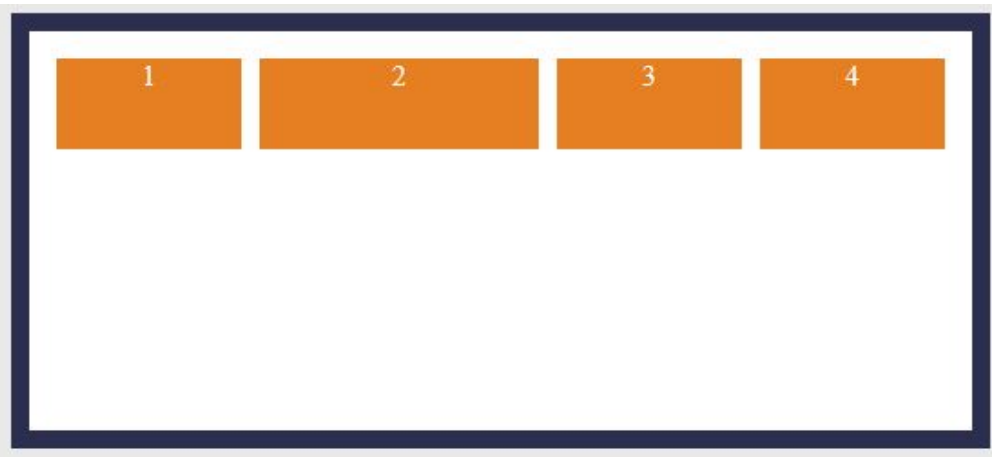
Este atributo permite definir la proporción de crecimiento de cada elemento, de forma tal que ocupen la fila o columna completa. Conozcamos su funcionamiento de acuerdo a los siguientes ejemplos:

```
flex-grow: 1;
```



```
flex-grow: 1;
```

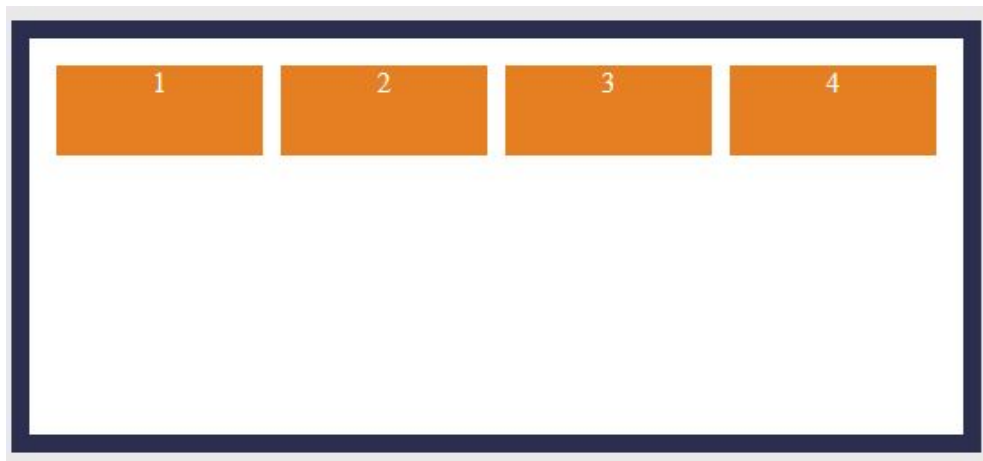
```
<div>  
  <div class="elemento2">  
    <div>  
      <div>  
        <div>  
          <div>  
            <div>  
              <div>  
                <div>  
                  <div>  
                    <div>  
                      <div>  
                        <div>  
                          <div>  
                        </div>  
                      </div>  
                    </div>  
                  </div>  
                </div>  
              </div>  
            </div>  
          </div>  
        </div>  
      </div>  
    </div>  
  </div>  
</div>
```




Flex-shrink

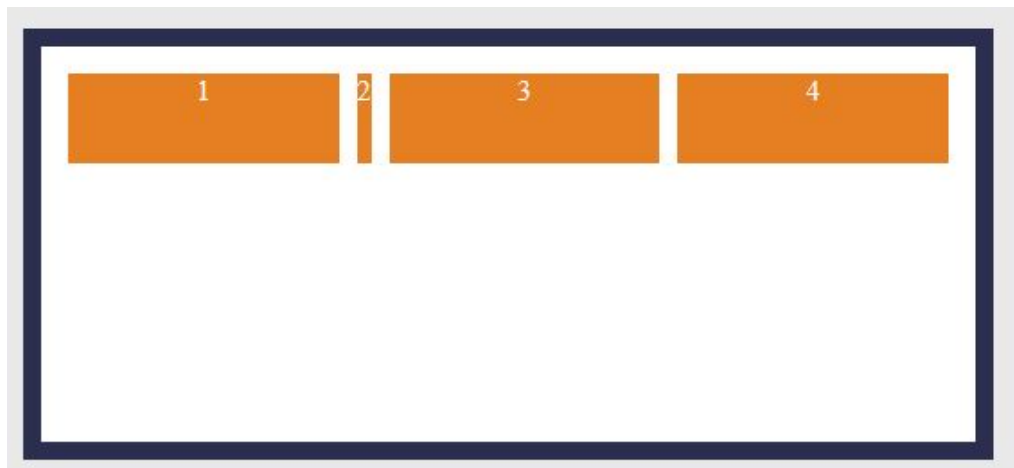
Funciona en forma análoga a flex-grow, pero en este caso es para definir la proporción de encogimiento de cada elemento, de forma tal que ocupen la fila o columna completa. Conozcamos su funcionamiento de acuerdo a los siguientes ejemplos:

```
width: 1000px;  
flex-shrink: 1;
```



```
width: 1000px;  
flex-shrink: 1;
```


```
 .elemento2 {  
  flex-shrink: 2;  
}
```

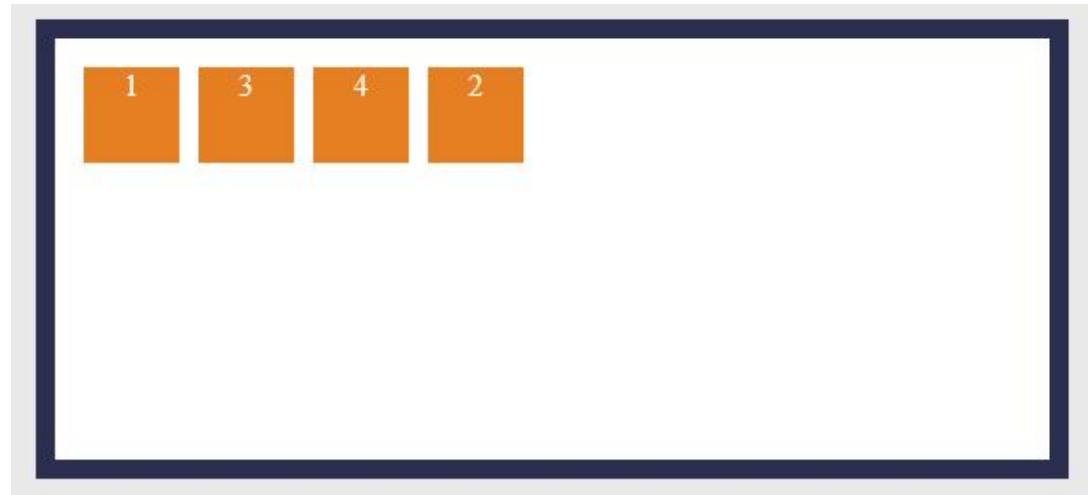


Order

Este atributo permite definir el orden de cada uno de los elementos que se encuentran dentro del contenedor. Conozcamos su funcionamiento de acuerdo a los siguientes ejemplos:


```
order: 1;
```


```
 .elemento2 {  
  order: 2;  
}
```

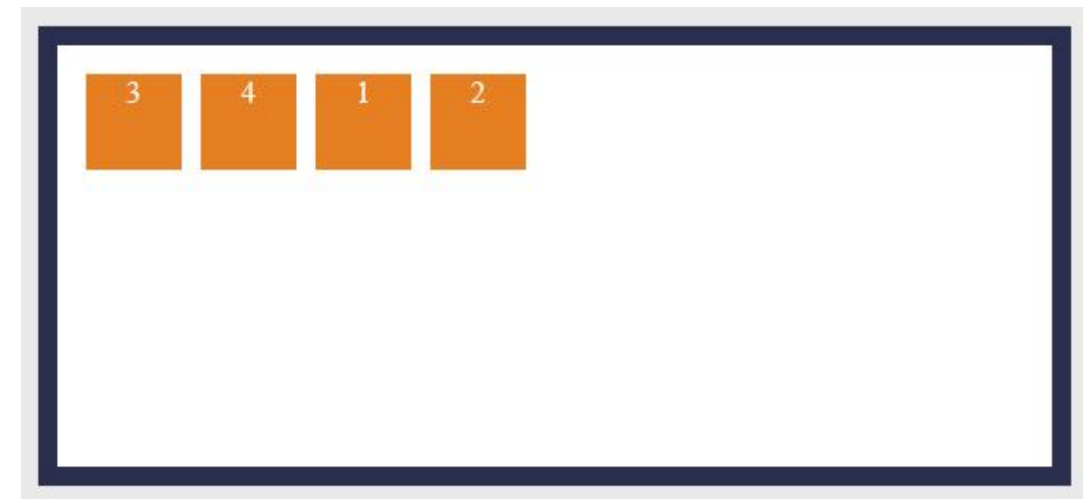


```
<div class="elemento elemento1"> 1 </div>
```

```
order: 1;
```

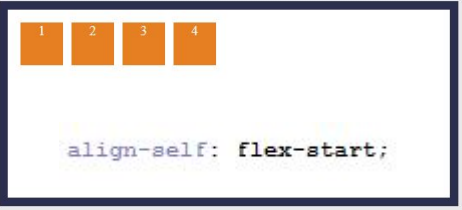
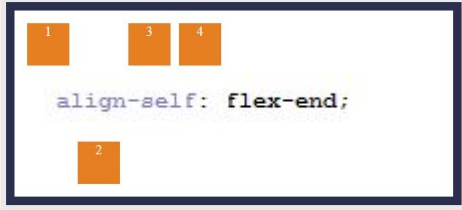
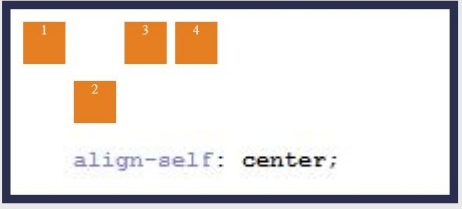
```
 .elemento1 {  
  order: 2;  
}
```

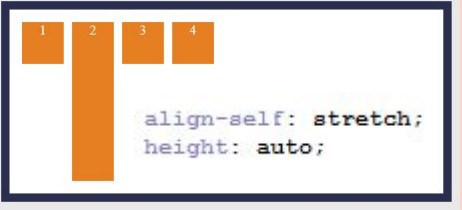

```
 .elemento2 {  
  order: 3;  
}
```



Align-self

Este atributo permite **alinear un elemento en particular** dentro de un contenedor. La siguiente tabla define su funcionamiento:

Valor	Uso	Layout
flex-start	Alinea el elemento al inicio	 <pre>align-self: flex-start;</pre>
flex-end	Alinea el elemento al final	 <pre>align-self: flex-end;</pre>
center	Alinea los elementos en el centro	 <pre>align-self: center;</pre>

Valor	Uso	Layout
stretch	Estira el elementos para que ocupen todo el alto o ancho	 <pre>align-self: stretch; height: auto;</pre>
baseline	Los elementos aparecen alineados en la base de la fuente	 <pre>.elemento1 { align-self: baseline; } .elemento2 { align-self: baseline; font-size: 40px; height: 100px; }</pre>

Resumen de Flexbox

Aplica a...	Atributo	Valores	Comportamiento
Contenedor	display	flex inline-flex	Define un contenedor como una caja flexible (flexbox)
Contenedor	flex-direction	row column row-reverse column-reverse	Designa la dirección en que se muestran los hijos dentro del contenedor
Contenedor	flex-wrap	no-wrap wrap wrap-reverse	Define el comportamiento de los hijos si no entran en el contenedor
Contenedor	flex-flow	flex-direction flex-wrap	Atajo de los atributos flex-direction y flex-wrap
Contenedor	justify-content	flex-start flex-end center space-around space-between	Define el justificado de los elementos dentro de la fila o columna
Contenedor	align-items	flex-start flex-end center stretch baseline	Alinea los ítems verticalmente en el caso de filas y horizontalmente en el caso de columnas
Contenedor	align-content	flex-start flex-end center stretch space-between space-around	Alinea el contenido de los hijos en bloque, cuando ocupan más de una fila o columna
Hijo	flex-basis	Valor	Funciona de la misma forma que asignar width o height pero en relación a la dirección
Hijo	flex-grow	Valor	Define la proporción de crecimiento de cada elemento para que ocupen la fila o columna completa
Hijo	flex-shrink	Valor	Define la proporción de encogimiento de cada elemento para que ocupen la fila o columna completa
Hijo	flex	flex-grow flex-shrink flex-basis	Atajo de los atributos flex-grow, flex-shrink y flex-basis
Hijo	order	Valor	Define el orden de cada elemento que se encuentra dentro del contenedor
Hijo	align-self	flex-start flex-end center stretch baseline	Alinea un elemento en particular dentro de un contenedor

Algunas referencias



HTML	https://www.w3schools.com/html/default.asp
CSS	https://www.w3schools.com/css/default.asp
Javascript	https://www.w3schools.com/js/default.asp



<https://www.cssportal.com>
<https://codepen.io/amazingcss>



Muchas Gracias!!

UNIVERSIDAD DEL CEMA
UCEMA

ucema.edu.ar