



# USAID

DEL PUEBLO DE LOS ESTADOS  
UNIDOS DE AMÉRICA



# Desarrollador web

USAID

# Módulo CSS

# Duración

- 4 módulos para un total de 200 horas de capacitación.
- 100 sesiones en modalidad virtual de 2 horas de duración cada una.

Módulo	Duración
Fundamentos de HTML	50 horas
Fundamentos de CSS y Flexbox	50 horas
Fundamentos de JavaScript y GIT	50 horas
Fundamentos de Backend	50 horas
Total	200 horas

# Metodología de evaluación

- Cada módulo se evaluará sobre 100 puntos, divididos de esta forma:

Actividad	Punteo
Tareas en línea	15 puntos
Laboratorios en línea	20 puntos
Ejercicios prácticos	10 puntos
Examen parcial	25 puntos
Zona	70 puntos
Examen final	30 puntos
Total por módulo	100 puntos

# Fundamentos de CSS

- Con HTML definimos la estructura de nuestras páginas, ahora con CSS haremos que esa estructura se vea bien con un poco de estilos.



# CSS Intro

- CSS -> Cascading Style Sheets (Hojas de estilo en cascada).
- Describe como los elementos HTML se desplegaran en pantalla.
- Nos permite ahorrar tiempo. Puede controlar la disposición de múltiples paginas al mismo tiempo.
- Hojas de estilo se almacenan en archivos CSS.



# ¿Por qué usar CSS?

- Usado para definir estilos de nuestras paginas, incluyendo el diseño, disposición y variaciones para diferentes dispositivos y tamaños de pantallas.

```
body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: white;  
    text-align: center;  
}  
  
p {  
    font-family: verdana;  
    font-size: 20px;  
}
```



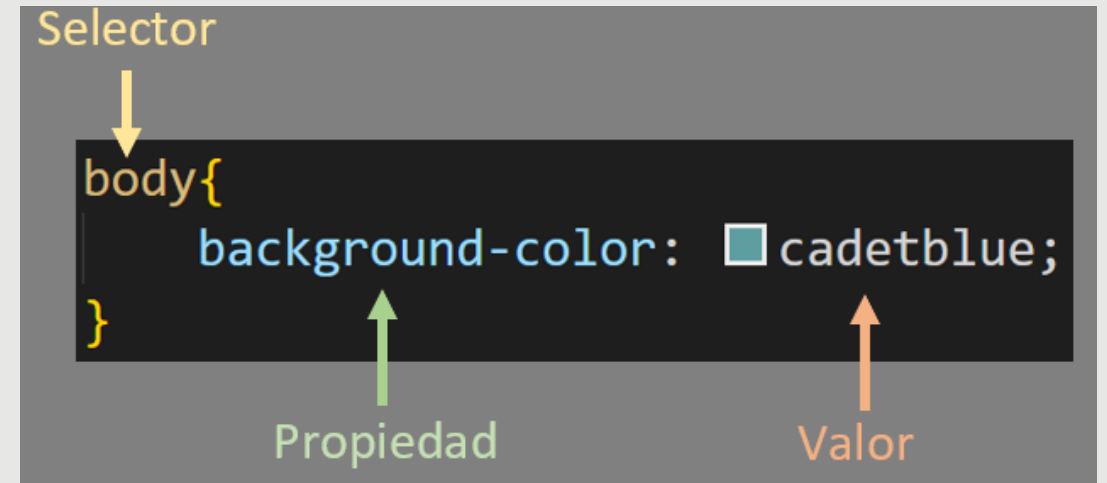
# CSS resuelve un problema

- El propósito de HTML nunca fue el dar estilo a las páginas web.
- Su objetivo es el describir el contenido.
- Cuando se agregaron etiquetas como `<font>` y `color` en la versión 3.2, convirtió el desarrollo web en un proceso largo y costoso.
- W3C creo CSS para resolver este problema.

```
<head>
  <title>HTML <font> tag</title>
</head>
<body>
<font size="6" color="red">This is a red color text with size 6!</font>
<font size="2" color="blue">This is a blue color text with size 2!</font>
<font face="verdana" color="green"></font>
</body>
```

# Sintaxis de CSS

- Los selectores hacen referencia a los elementos HTML a los cuales aplicaremos las reglas CSS.



# Selector universal

- Selecciona elementos de cualquier tipo, por eso es universal.

```
* {  
    color: purple;  
}
```

# Selector de tipo


- Conocido también como selector de elemento, seleccionara el elemento dado.

```
div{  
  color:  red;  
}
```

# Selector de clase

- Afectaran todos los elementos de la clase dada.
- La clase es un atributo asignado a un elemento HTML.
- Anteponemos un punto al nombre de la clase.
- Podemos aplicar una clase a varios elementos.
- Un elemento puede tener varias clases.


```
<div class="texto-alerta">  
    Por favor tome precauciones.  
</div>
```

```
.texto-alerta{  
    color:  red;  
}
```

# Selector de ID

- Similares a los selectores de clase, seleccionan a un elemento con el ID dado.
- Id es un atributo que podemos dar a un elemento HTML.
- Usamos # seguido del id.
- Un elemento puede tener solamente un Id.
- Este Id no debe repetirse ni contener espacios en blanco.

```
<div id="cancion">  
  <p>I've got feathers in my hair</p>  
</div>
```

```
#cancion{  
  background-color:  coral;  
}
```

# Agrupando selectores

- Si tenemos selectores que comparten algunas de sus declaraciones de estilo, los podemos agrupar.
- Se agrupan en un listado separado por coma.

```
.leido{
  color: ■white;
  background-color: □black;
  /* declaraciones unicas */
}

.sin-leer{
  color: ■white;
  background-color: □black;
  /* declaraciones unicas */
}
```

```
.leido,
.sin-leer{
  color: ■white;
  background-color: □black;
}


.leido{
  /* declaraciones unicas */
}

.sin-leer{
  /* declaraciones unicas */
}
```

# Encadenando selectores

- Podemos realizar esto al listarlos sin separación.

```
<div>
  <div class="subseccion encabezado">
    Ultimo post
  </div>
  <p class="subseccion previa">
    Acá aparecera el mensaje previo
  </p>
</div>
```


```
.subseccion.encabezado{
  color:  red;
}
```




# Encadenando selectores

- Podemos realizar esto al listarlos sin separación.
- Podemos encadenar cualquier combinación de selectores excepto para selectores de tipo.



```
<div>
  <div class="subseccion encabezado">
    Ultimo post
  </div>
  <p class="subseccion previa" id="texto">
    Acá aparecera el mensaje previo
  </p>
</div>
```

```
.subseccion.encabezado{
  color:  red;
}
```

```
.subseccion#texto{
  color:  blue;
}
```


# Propiedades CSS

- La propiedad **color** define el color de texto de un elemento.
- **background-color** define el color de fondo de un elemento.
- Aceptan valores HEX, RGB y HSL.

```
.ancestro .contenido{  
  background-color:  rgb(100,0,0);  
  color:  rgb(0,100,0);  
}
```

# Combinadores

- Estos nos permiten combinar múltiples selectores, de una forma diferente a agruparlos o encadenarlos, estos muestran una relación entre los selectores.
- Existen cuatro tipos de combinadores, de momento trabajaremos con el **combinador descendiente**.

```
.ancestro .contenido{  
    background-color:  blueviolet;  
}
```

```
<div class="ancestro">  
  <p>soy ancestro</p>  
  <div class="contenido">  
    <p>soy contenido</p>  
    <div class="contenido">  
      <p>soy contenido</p>  
    </div>  
  </div>  
</div>  
  
<div class="contenido">  
  <p>Yo también soy contenido!!!!</p>  
</div>
```

# Iniciando con tipografía

- **font-family** permite definir la fuente, recibe valores separados por coma.
- Podemos definir “nombre de fuente” como “Times New Roman” o nombres genéricos como sans-serif, colocamos “ ” por los espacios en blanco.
- **font-size** cambiara el tamaño de la fuente.
- **font-weight** afectara la negrita del texto.
- **text-align** alineara el texto horizontalmente dentro de un elemento.

```
div{  
    font-family: 'Courier New', Courier, monospace;  
    font-size: 50px;  
    font-weight: bold;  
    text-align: center;  
}
```

# Iniciando con tipografía

- **font-family** permite definir la fuente, recibe valores separados por coma.
- Podemos definir “nombre de fuente” como “Times New Roman” o nombres genéricos como sans-serif, colocamos “ ” por los espacios en blanco.
- **font-size** cambiara el tamaño de la fuente.
- **font-weight** afectara la negrita del texto.
- **text-align** alineara el texto horizontalmente dentro de un elemento.

```
div{  
    font-family: 'Courier New', Courier, monospace;  
    font-size: 50px;  
    font-weight: bold;  
    text-align: center;  
}
```

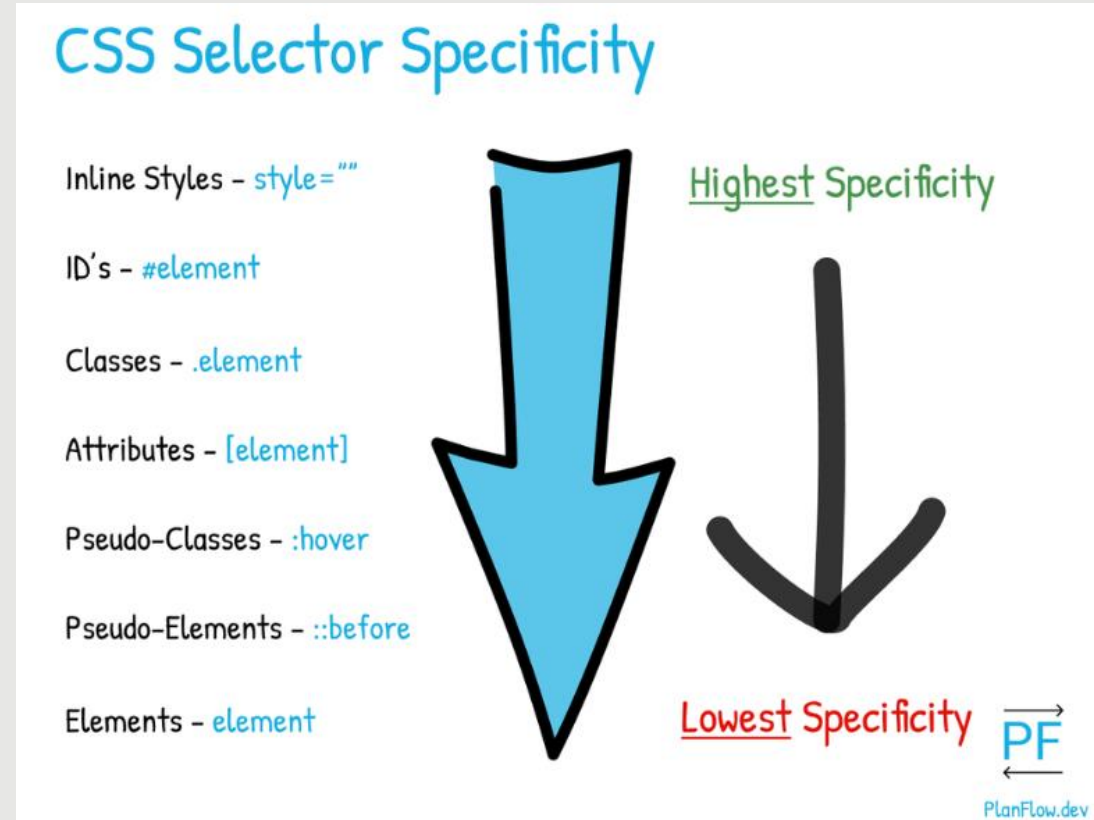
# Alto y ancho en imágenes

- Por defecto, un elemento `<img>` tendrá su `height` y `width` igual a la imagen original.
- Si quisiéramos ajustar el tamaño de la imagen sin perder la proporción, podemos aplicar el valor `auto`.
- Para evitar que al cargar la página, los elementos se redibujen, es recomendable definir estas propiedades, ya sea con CSS o bien con los atributos `height` y `width` de `<img>`.

```
.ancho{  
    width: 400px;  
    height: auto;  
}  
  
.alto{  
    width: auto;  
    height: 400px;  
}
```

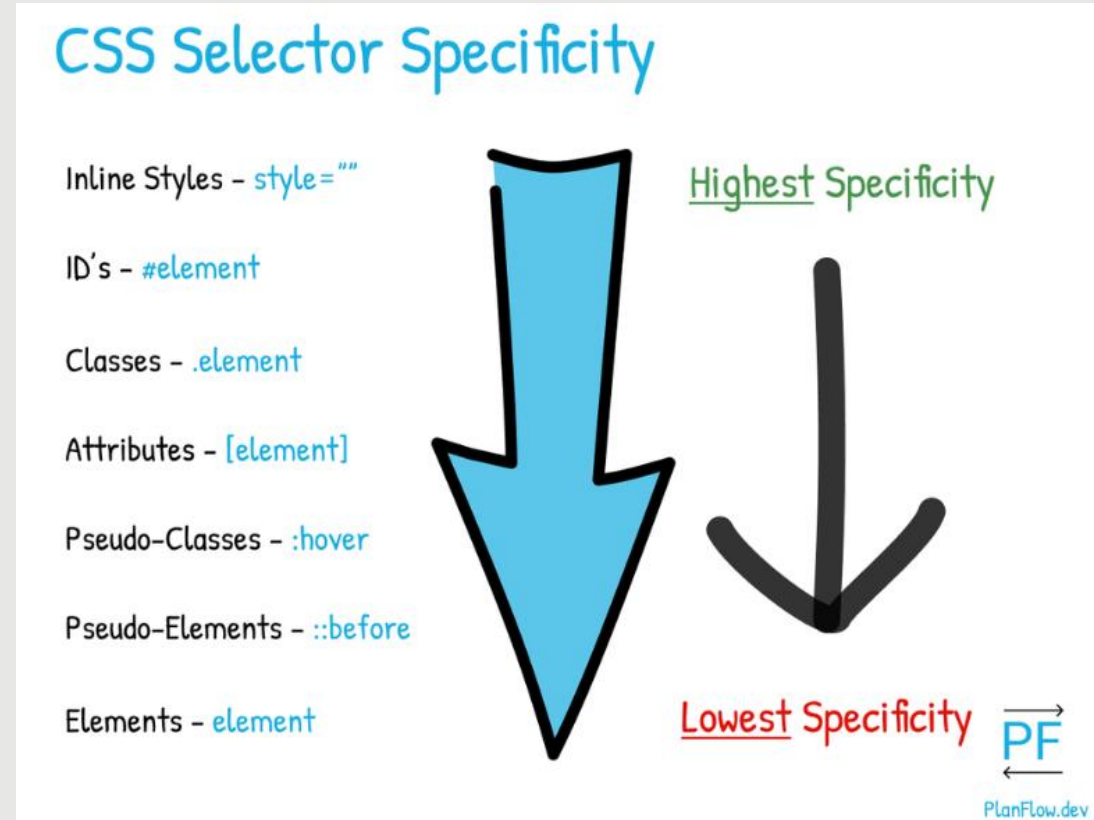
# La cascada de CSS

- Algunas veces tendremos reglas que entren en conflicto entre si, y terminamos con resultados inesperados.
- Debemos tomar en cuenta los estilos por defecto que define el navegador.
- La cascada es lo que determina que reglas se aplican a nuestro HTML.



# Especificidad

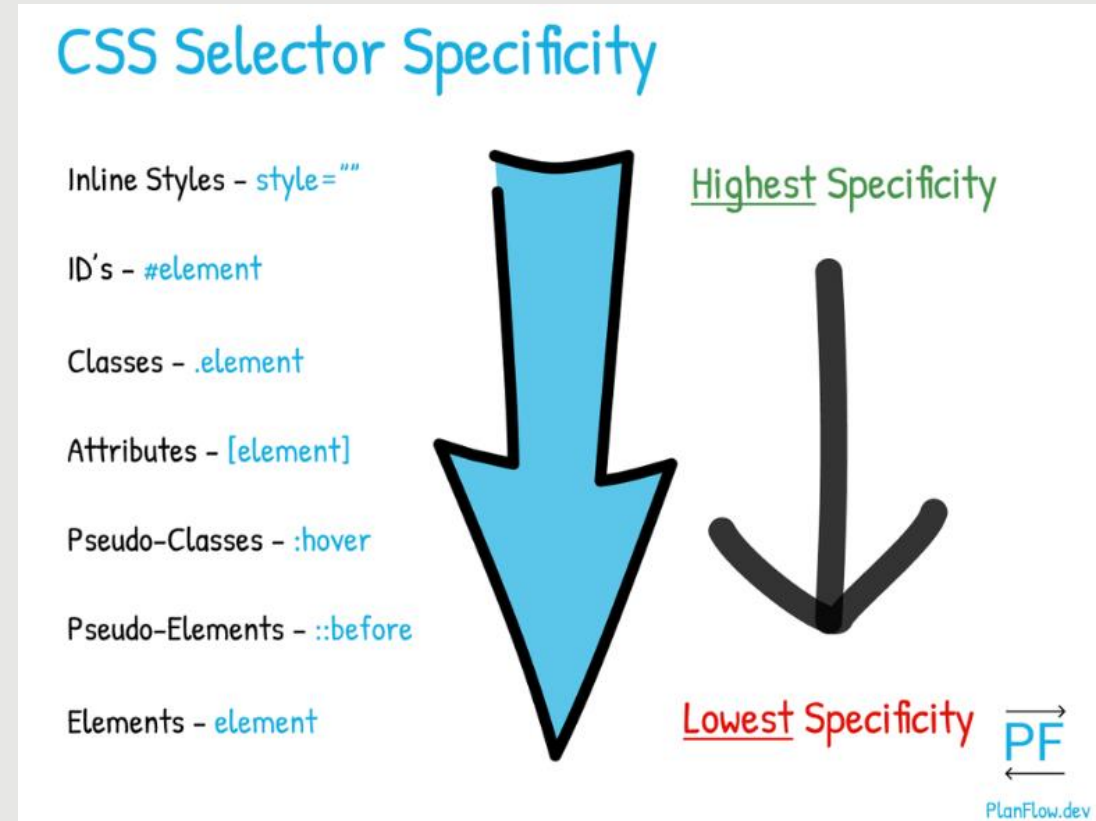
- Una declaración CSS que sea más específica tomara precedencia sobre otras que sean menos específicas.
- Los estilos Inline (atributo style) tienen más especificidad comparado con los selectores.





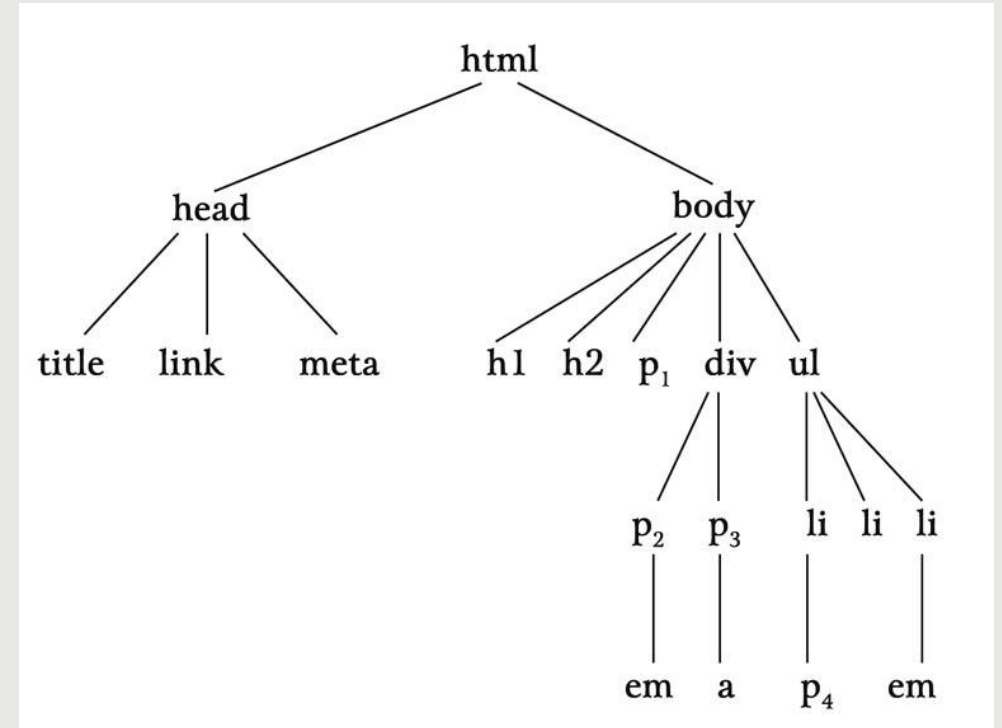
# Especificidad

- Contamos con varios selectores en CSS, nos enfocaremos en:
  1. Selectores de ID.
  2. Selectores de clase.
  3. Selectores de tipo.
- La especificidad solo será tomada en cuenta cuando el elemento tenga declaraciones en conflicto.



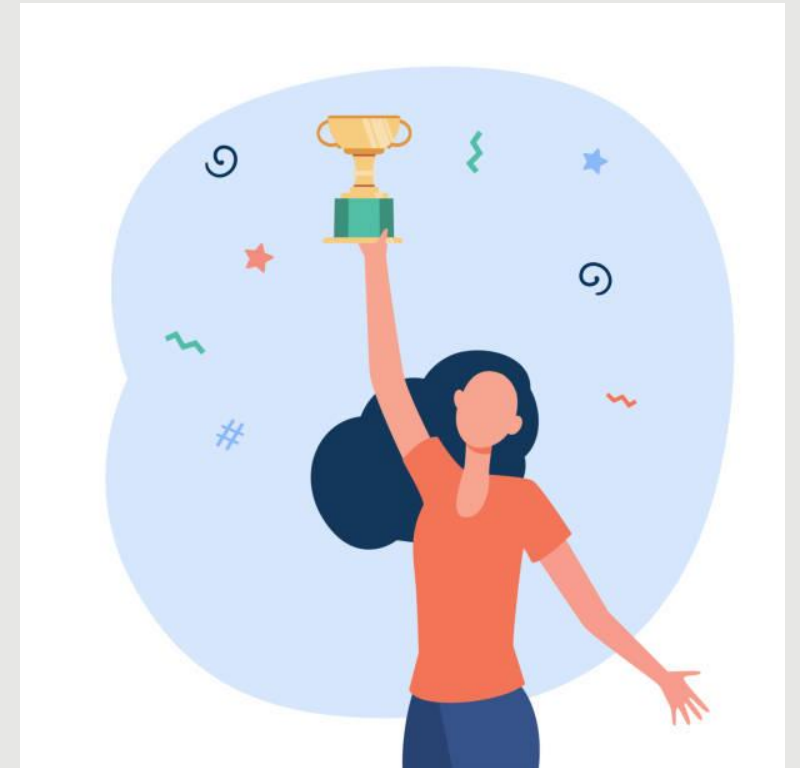
# Herencia

- Se refiere a ciertas propiedades CSS que al ser aplicadas a un elemento, se heredan a sus descendientes, incluso si no definimos una regla para ellos.
- La excepción a esto si directamente seleccionamos un elemento, esto siempre vence a la herencia.



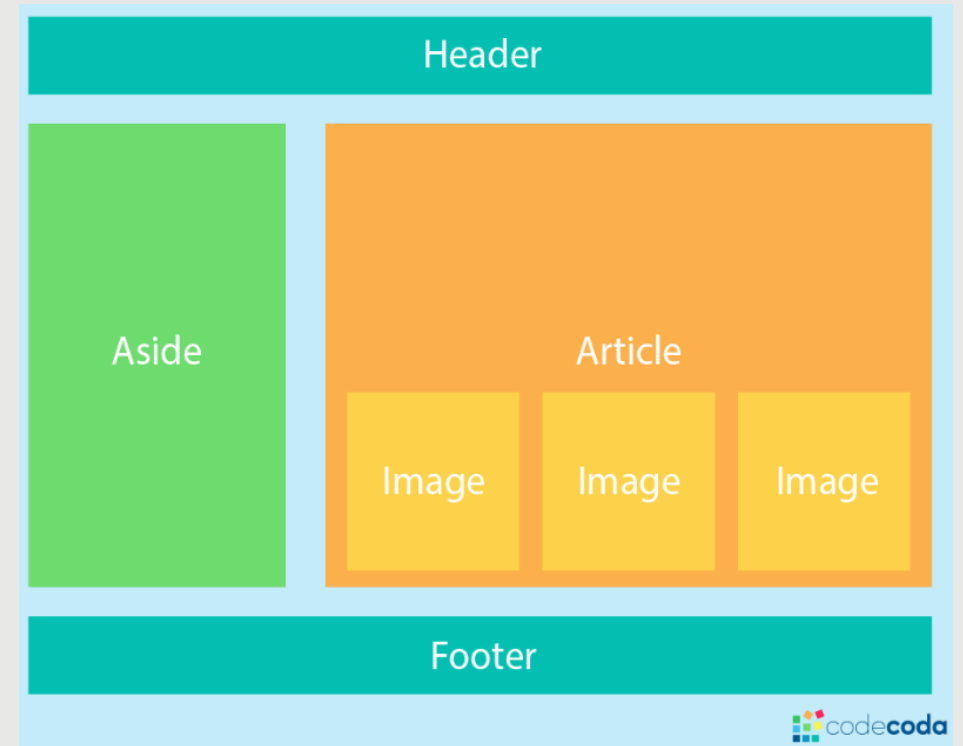
# Regla del orden

- El último factor a tomar en cuenta es el orden en que aparecen las reglas.
- Digamos que después de evaluar la especificidad, aun hay múltiples reglas en conflicto, el ganador será la última definida.



# Box model (modelo de caja)

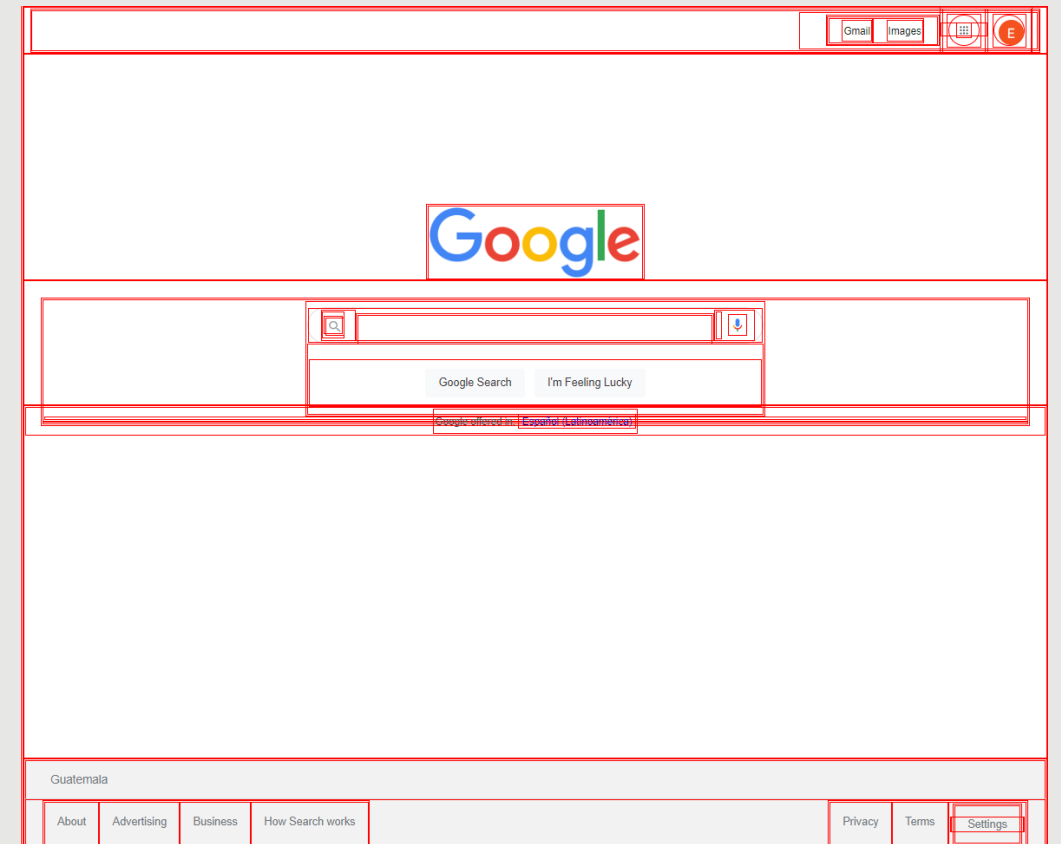
- Las habilidades más importantes en CSS son el posicionamiento y la distribución.
- Cambiar colores y fuentes es crucial, pero poder colocar las cosas exactamente dentro de la página web es más importante.



# Box model (modelo de caja)

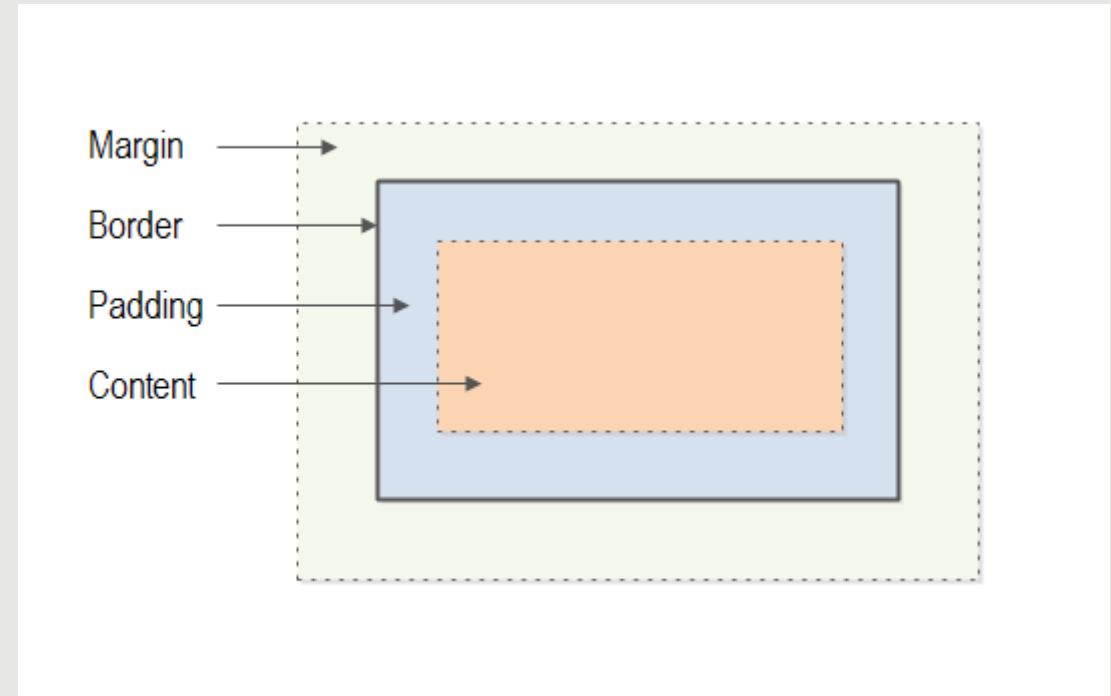
- El concepto importante a entender es que todo en una página web es una caja rectangular.
- Estas cajas pueden tener otras cajas en su interior o estar un al lado de otra.

```
* {  
  border: 2px solid red;  
}
```



# Box model (modelo de caja)

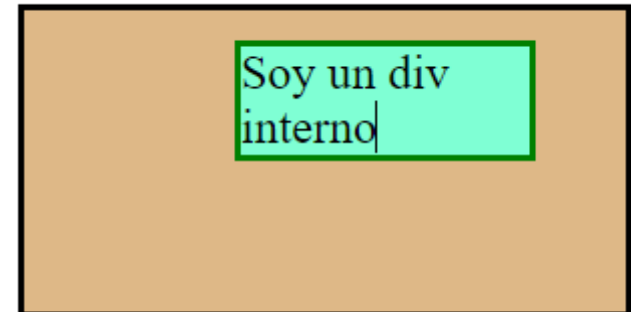
- Lo complicada es que hay muchas formas de manipular el tamaño de estas cajas y espaciarlas.
- Tenemos, padding, margin y border.
- Padding, controla el espacio entre el borde de una caja y su contenido.
- Margin, controla el espacio entre una caja y otras a su alrededor.
- Border, controla el espacio entre el margen y el padding.



# Margen – sintaxis abreviada

- Para escribir menos código es posible especificar todas las propiedades del margen en una sola propiedad.
- La propiedad margin es abreviado para las propiedades:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left

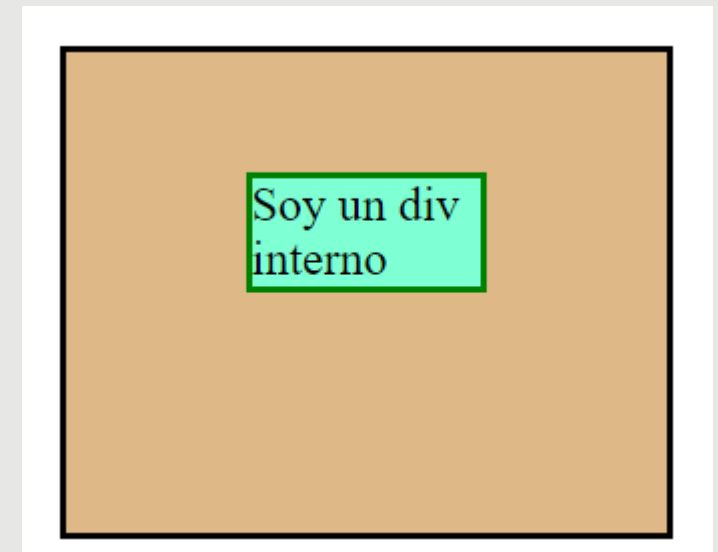
```
<style>
  #externo{
    border: 2px solid black;
    background-color: burlywood;
    width:200px;
    margin:auto;
  }
  #interno{
    border: 2px solid green;
    background-color: aquamarine;
    margin:10px 30px 50px 70px;
  }
</style>
```



# Margen – sintaxis abreviada

- Sí la propiedad margin tiene 3 valores:
- `margin: 40px 60px 80px`
  - El margen superior será de 40px
  - Los márgenes izquierdo y derecho serán 60px
  - El margen inferior será de 80px

```
#interno2{  
  border: 2px solid green;  
  background-color: aquamarine;  
  margin: 40px 60px 80px;  
}
```

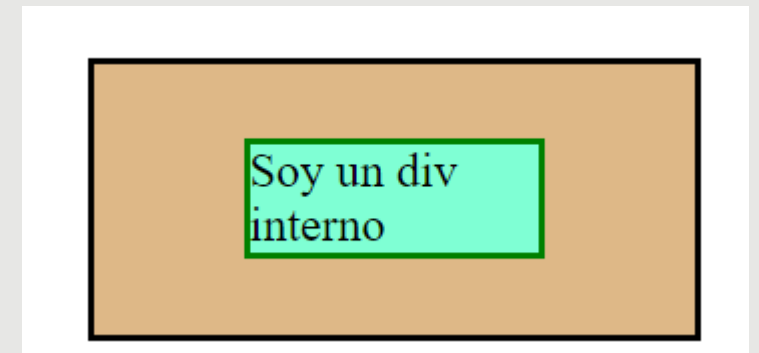




# Margen – sintaxis abreviada

- Sí la propiedad margin tiene 2 valores:
- margin: 25px 50px
  - Los márgenes superior e inferior serán de 25px
  - Los márgenes izquierdo y derecho serán 50px
- Con solamente un valor, aplicara ese valor a los 4 márgenes.

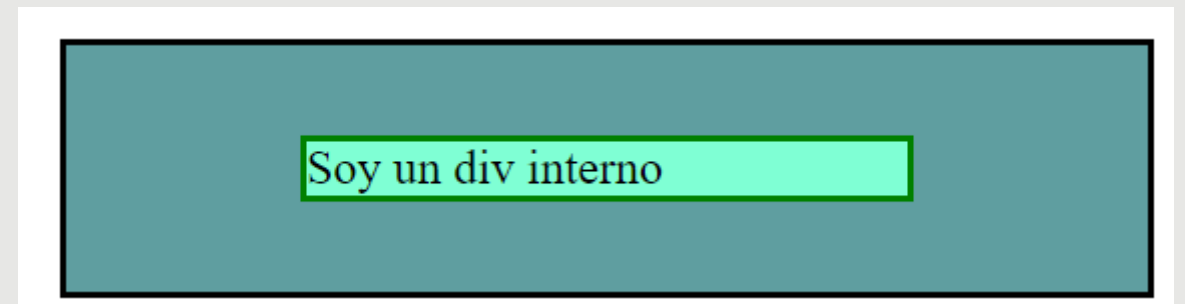
```
#interno3{  
  border: 2px solid green;  
  background-color: aquamarine;  
  margin: 25px 50px;  
}
```



# Margen – el valor auto

- Sí a la propiedad margin le colocamos el valor auto, el elemento se centrara horizontalmente dentro de su contenedor.

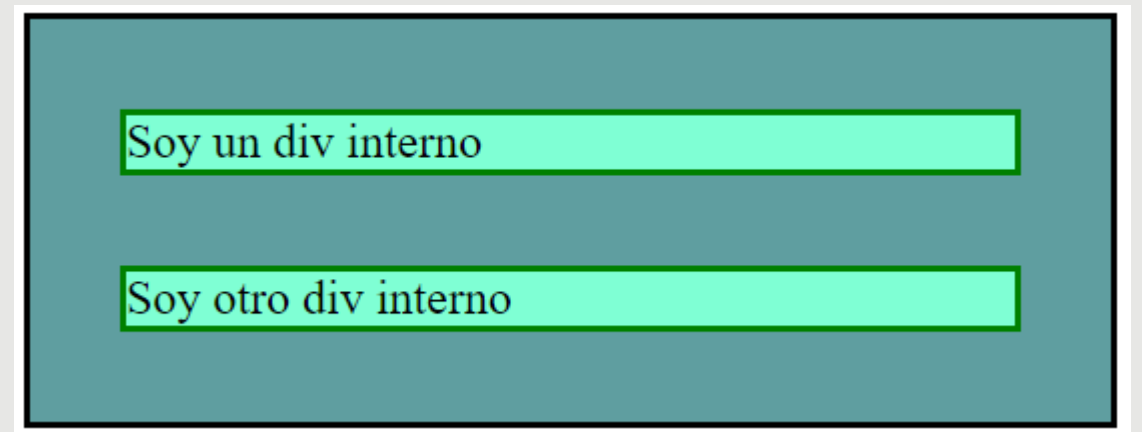
```
.externo2 {  
  border: 2px solid black;  
  background-color: cadetblue;  
  width: 300px;  
  margin: auto;  
  padding: 30px;  
}  
  
#interno4 {  
  border: 2px solid green;  
  background-color: aquamarine;  
  width: 200px;  
  margin: auto;  
}
```



# Margen – colapso de márgenes

- Los márgenes superior e inferior en ocasiones colapsan en un solo margen que es igual al mayor de estos.
- Esto no sucede con los márgenes de izquierda y derecha.

```
#interno6 {  
  border: 2px solid green;  
  background-color: aquamarine;  
  margin: 0 0 10px 0;  
}  
#interno7 {  
  border: 2px solid green;  
  background-color: aquamarine;  
  margin: 30px 0 0 0;  
}
```



# Padding

- El padding es usado para crear espacio alrededor del contenido del elemento, dentro de los bordes definidos.

```
#ejemplo1{  
  border: 1px solid green;  
  padding: 0px;  
}
```

El padding de este elemento es de 50px.

# Padding

- Podemos controlar el valor para el padding de cada uno de los lados.
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left
- No se permiten valores negativos.
- La notación corta es valida para padding.

```
#ejemplo1{  
  border: 1px solid green;  
  padding: 0px;  
}
```

El padding de este elemento es de 50px.

# Padding y width del elemento

- La propiedad width define el ancho del área de contenido de un elemento.
- El área de contenido es la porción dentro de padding, border y margin.
- Sí un elemento con un ancho específico, le agregamos padding, el ancho total del elemento aumentara.
- Por lo regular, este es un resultado no deseado.



```
.ejemplo1{  
  background-color: red;  
  width: 300px;  
}  
.ejemplo2{  
  background-color: chocolate;  
  width: 300px;  
  padding-right: 50px;  
}
```

Soy un div de 300px de ancho

Soy un div con width y padding

# Padding y width del elemento

- Para mantener el ancho del elemento sin importar la cantidad del padding, usamos la propiedad **box-sizing**.
- Esta propiedad permite que el elemento mantenga su ancho actual.
- Si incrementamos el padding, el espacio disponible para el contenido disminuirá.

```
.ejemplo3{  
  background-color:  red;  
  width: 300px;  
}  
.ejemplo4{  
  background-color:  chocolate;  
  width: 300px;  
  padding-right: 100px;  
  box-sizing: border-box;  
}
```

Soy un div de 300px de ancho

Soy un div con width, padding  
y box-sizing

# box-sizing

- Esta propiedad nos permite incluir el padding y el borde en el ancho y alto total del elemento.
- Por defecto, el ancho y alto de un elemento se calcula así:
- $\text{width} + \text{padding} + \text{border} = \text{ancho actual del elemento.}$
- $\text{height} + \text{padding} + \text{border} = \text{alto actual del elemento.}$

ancho: 200px, alto: 50px

ancho: 200px, alto: 50px,  
padding: 25px, border: 1px

```
.ejemplo1{  
  width: 200px;  
  height: 50px;  
  border: 1px solid blue;  
}  
  
.ejemplo2{  
  width: 200px;  
  height: 50px;  
  padding: 25px;  
  border: 1px solid red;  
}
```



# box-sizing

- Con box-sizing podemos incluir el padding y border en el ancho y alto totales del elemento.
- El valor para box-sizing debe ser border-box.

ancho: 200px, alto: 50px

ancho: 200px, alto: 50px, padding:  
25px, border: 1px

```
.ejemplo3{  
  width: 200px;  
  height: 50px;  
  border: 1px solid blue;  
}  
  
.ejemplo4{  
  width: 200px;  
  height: 50px;  
  padding: 25px;  
  border: 1px solid red;  
  font-size: x-small;  
  box-sizing: border-box;  
}
```

# box-sizing

- Para que los elementos mantengan las mismas proporciones, muchos desarrolladores aplican esta propiedad a todos los elementos.
- Muchos navegadores ya aplican box-sizing: border-box; para muchos elementos, pero no para todos.

```
input,  
textarea {  
  width: 100%;  
}  
  
form {  
  width: 200px;  
  background-color: chocolate;  
}
```



# box-sizing

- Para que los elementos mantengan las mismas proporciones, muchos desarrolladores aplican esta propiedad a todos los elementos.
- Muchos navegadores ya aplican box-sizing: border-box; para muchos elementos, pero no para todos.

```
/*Aplicando a todos los elementos*/
*{
  box-sizing: border-box;
}

input,
textarea {
  width: 100%;
  box-sizing: border-box;
}

form {
  width: 200px;
  background-color: chocolate;
}
```



The image shows a visual representation of a form with a chocolate brown background. It contains two input fields, both labeled 'Ingresa texto'. The top field is a single-line text input, and the bottom field is a multi-line text area. Both fields have a white background and a thin border. The labels 'Text field' and 'Text area' are positioned above their respective input fields.

# Flexbox

- El modelo de Flexbox nos da una forma eficiente de diseñar, alinear y distribuir el espacio entre los elementos dentro del documento.
- Esto es así aunque el viewport y el tamaño de nuestros elementos sea dinámico o desconocido.



# Usando flexbox

- Para empezar a usar flexbox, necesitamos definir un contenedor flex.
- Esto lo logramos con:
  - `display: flex`
  - `display: inline-flex`
- Con esto, se aplica flexbox sobre los elementos dentro del contenedor.

- Elemento 1
- Elemento 2
- Elemento 3|

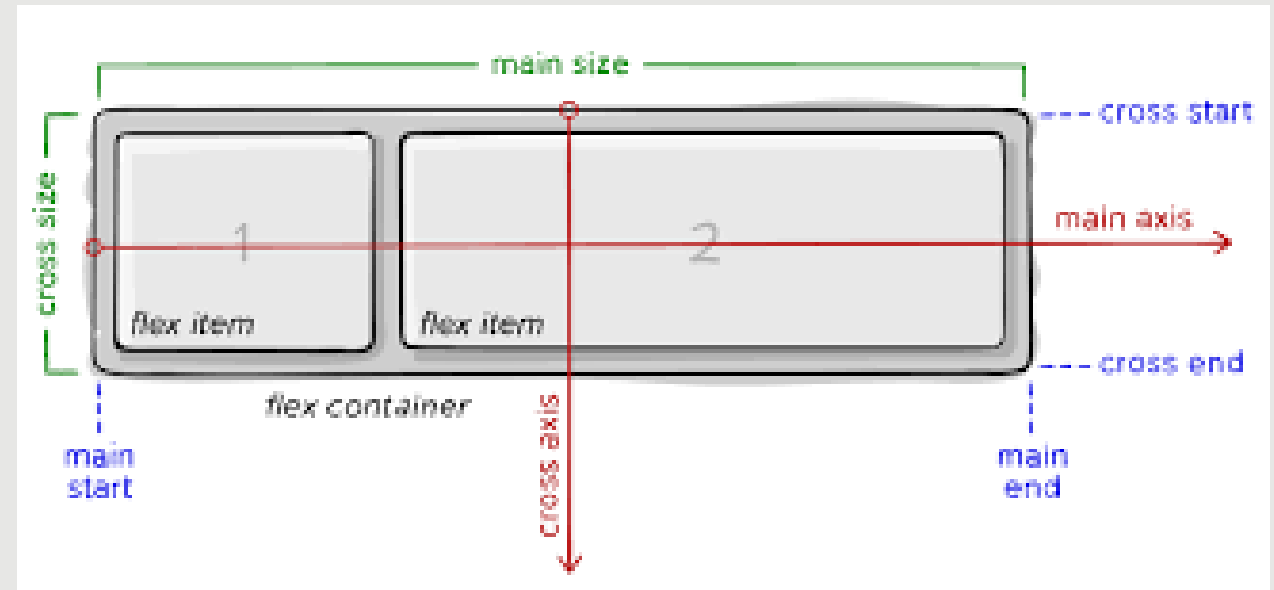


```
1 <ul style="display: flex;">
2   <li>Elemento 1</li>
3   <li>Elemento 2</li>
4   <li>Elemento 3</li>
5 </ul>
```

- Elemento 1Elemento 2Elemento 3

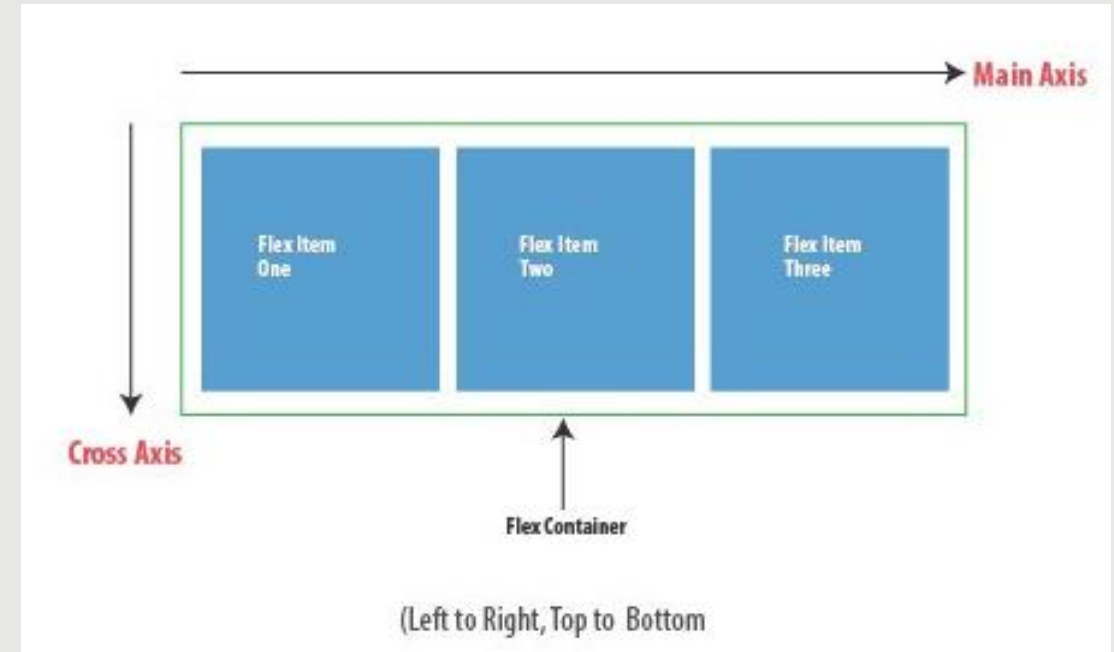
# Propiedades del contenedor flex

- Al tener definido un contenedor como flex, se habilitan algunas propiedades de alineación.
- Flex-direction.
- Flex-wrap.
- Flex-Flow.
- Justify-content.
- Align-items.
- Align-content



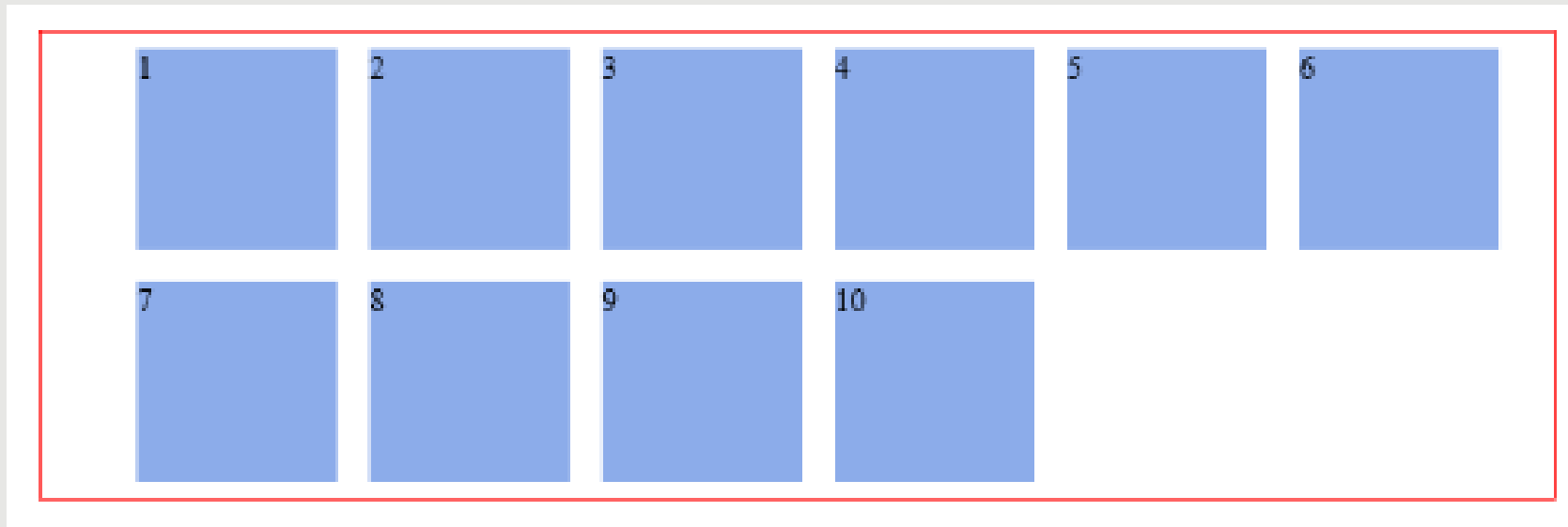
# Flex-direction

- Propiedad que controla en que dirección los elementos flex se colocan a lo largo del eje main.
- Nos permite decidir como se colocan los elementos, horizontalmente, verticalmente o en reversa en ambas direcciones.
- Para ser correctos, el eje main se compara con la horizontal y el eje cross con la vertical.
- Por defecto su valor esta en: row.
- Posibles valores: row, column, reverse-row, reverse-column.



# Flex-wrap

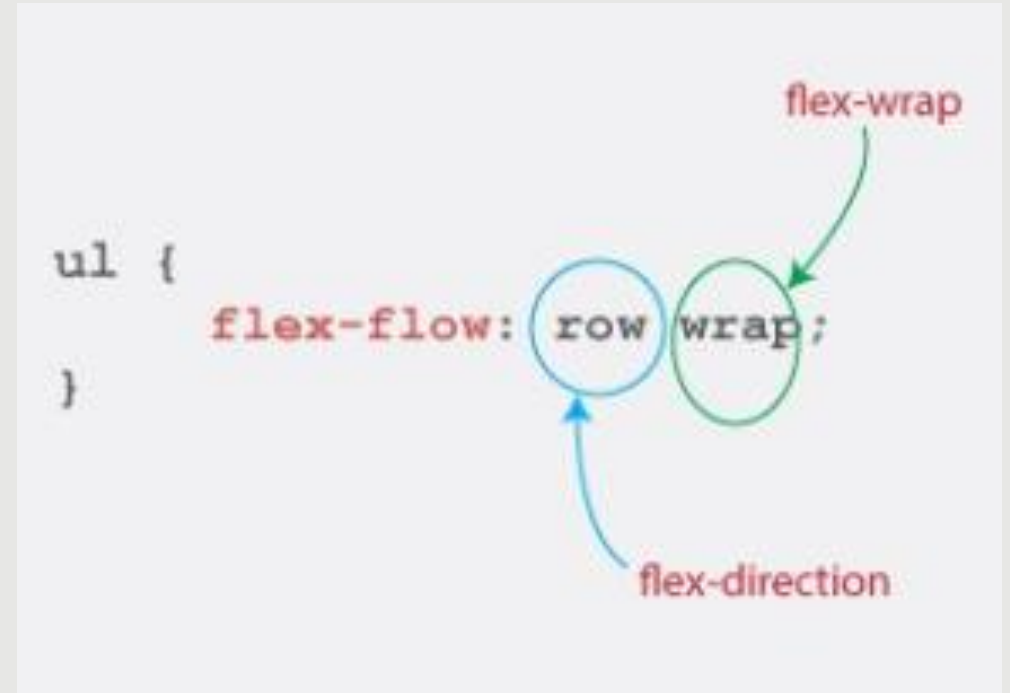
- Con esta propiedad podemos definir si los elementos dentro del contenedor permanecen en una sola línea o si fluirán en varias.
- El valor por defecto es: nowrap
- Posibles valores: nowrap, wrap, wrap-reverse





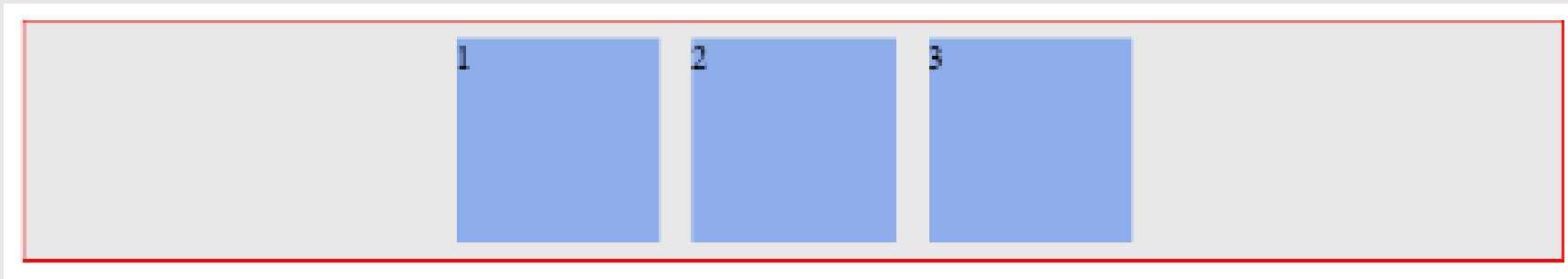
# Flex-flow

- Esta es la notación corta para flex-direction y flex-wrap.
- Permite enviar múltiples valores en una sola línea.
- flex-flow: row wrap;
- Combinaciones posibles: flex-flow: row nowrap, flex-flow: column wrap, flex-flow: column nowrap



# justify-content

- Define como los elementos flex se colocaran sobre el eje main.
- Valor por defecto: flex-start
- Posibles valores: flex-start, flex-end, center, space-between, space-around.



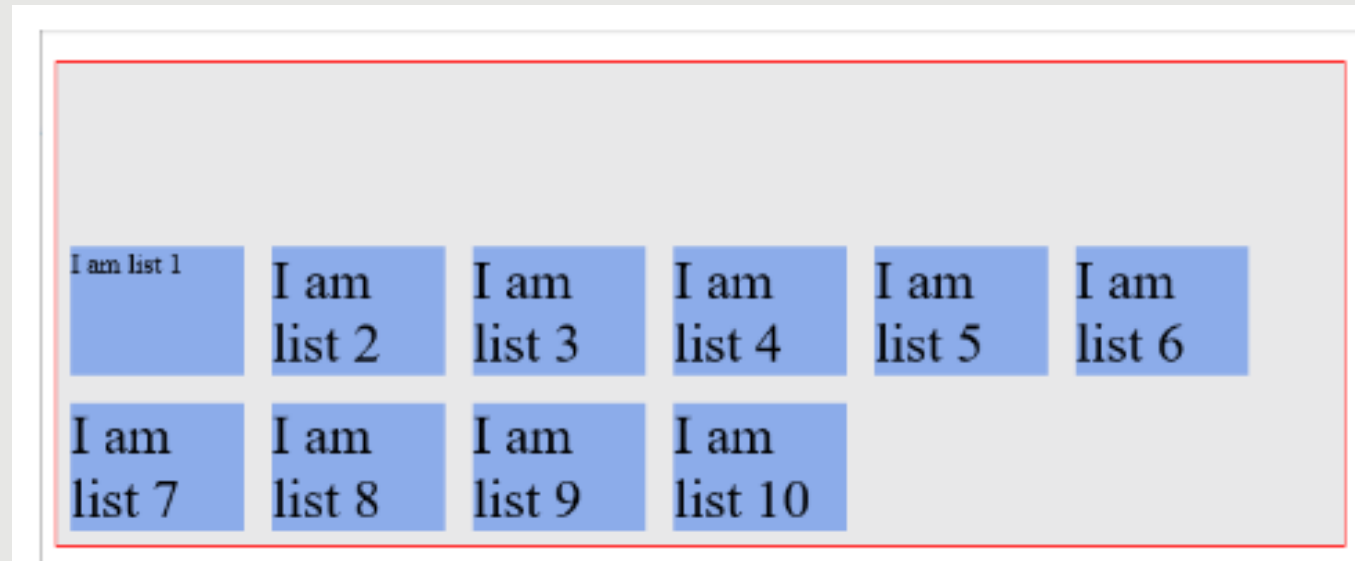
# align-items

- Similar a justify-content pero aplicada sobre el eje cross.
- Valor por defecto: stretch
- Posibles valores: flex-start, flex-end, center, stretch, baseline



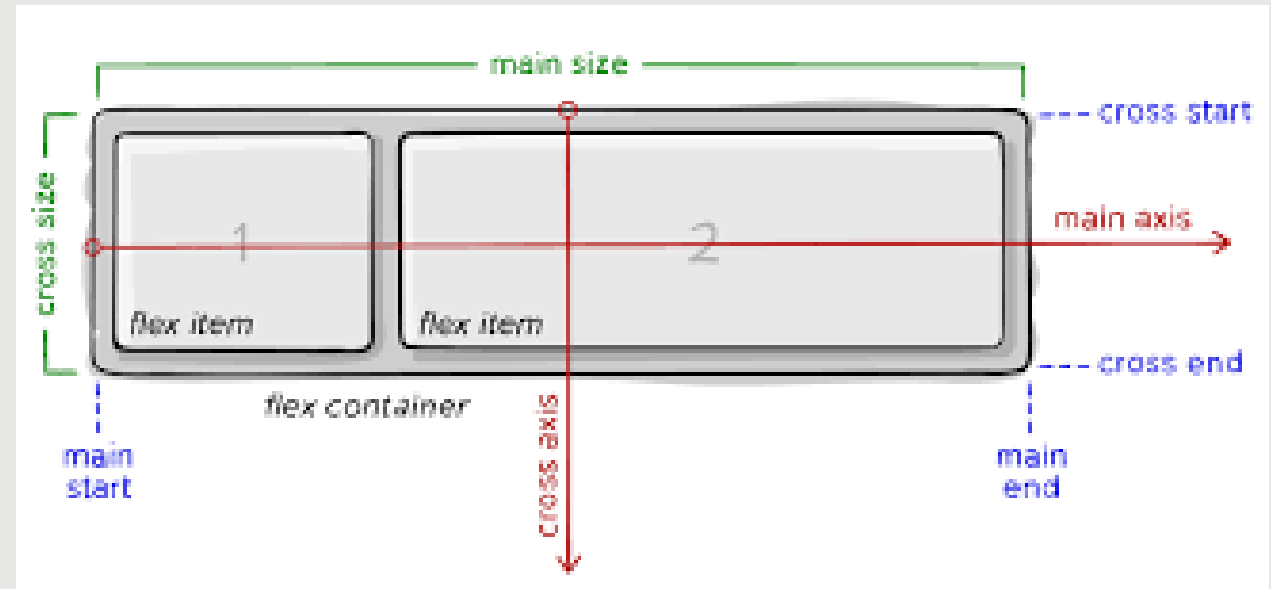
# align-content

- Usada en contenedores con varias líneas de elementos.
- Controla como los elementos se alinean en un contenedor multi línea.
- Posibles valores: stretch, flex-start, flex-end, center.



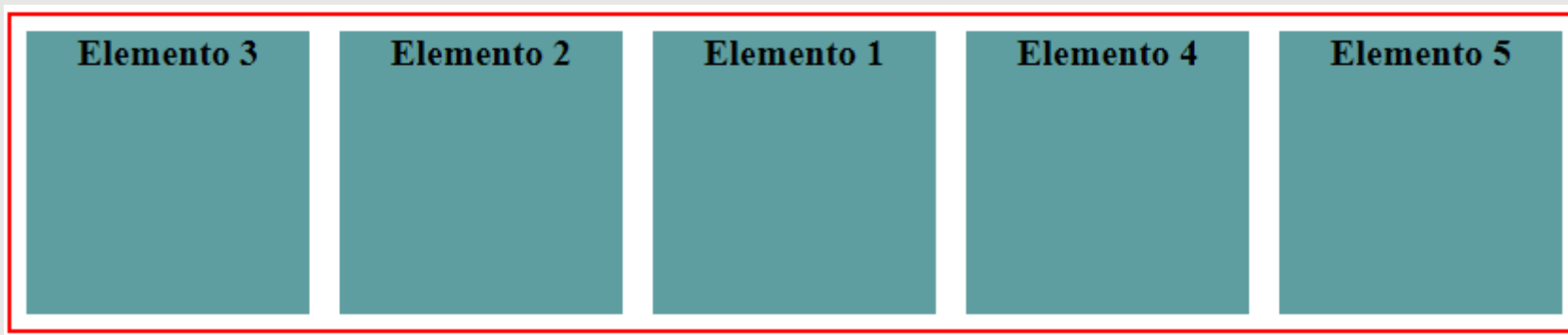
# Propiedades de los elementos flex

- Al igual que un contenedor, se nos habilitan unas cuantas propiedades para los elementos flex.
- Order
- Flex-basis
- Flex
- Align-self
- Flex-grow
- Flex-shrink



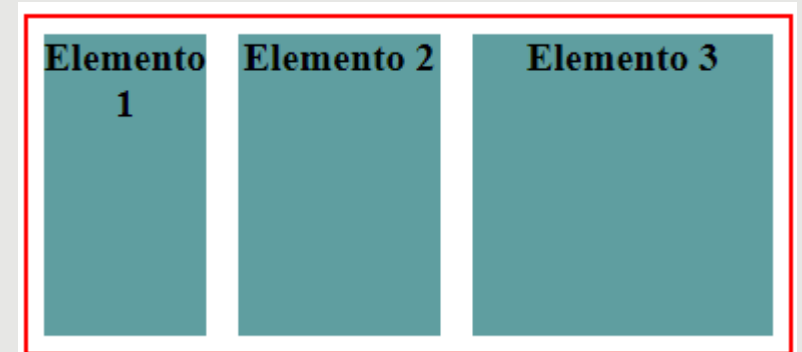
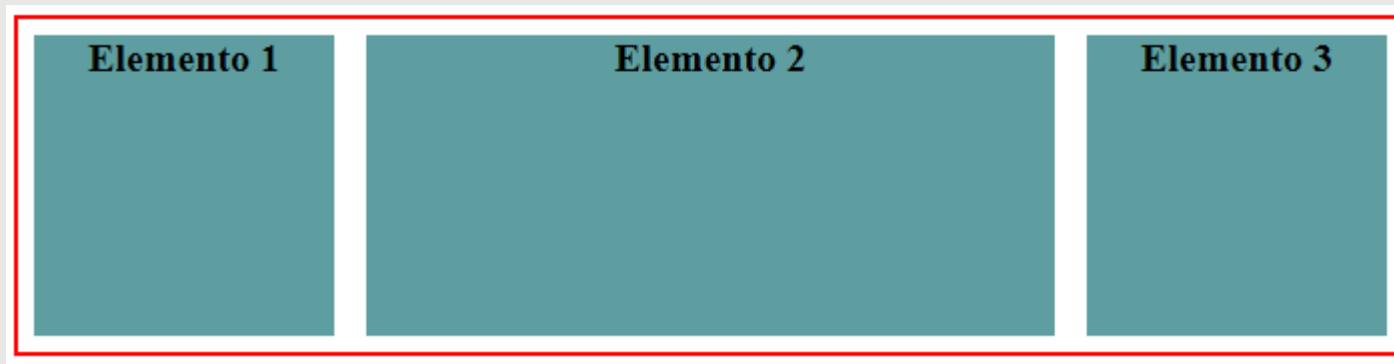
# Order

- Permite controlar el orden de los elementos flex dentro de un contenedor.
- Podemos mover un elemento flex de una posición a otra.
- Los elementos cambian sin afectar el código fuente.
- El valor por defecto es 0.
- Puede aceptar valores negativos o positivos.
- Los elementos se ordenan del menor al mayor.



# Flex-grow y flex-shrink

- Estas propiedades permiten controlar la cantidad de espacio que se extenderán o encogerán los elementos si hay espacio disponible.
- Aceptan valores iniciando en 0 hasta cualquier numero positivo.
- Por defecto el valor de flex-grow es 0 y de flex-shrink es 1



# Flex-basis

- Define el tamaño inicial de un elemento flex.
- Se aplica antes de que flex-grow o flex-shrink ajusten el tamaño.
- El valor por defecto es: auto
- Toma valores: en porcentajes, ems, rems, pixeles, etc

## Trabajando con flex-basis

**Elemento 1, Lorem  
ipsum dolor sit  
amet, consectetur  
adipiscing elit.  
Omnis, totam.**



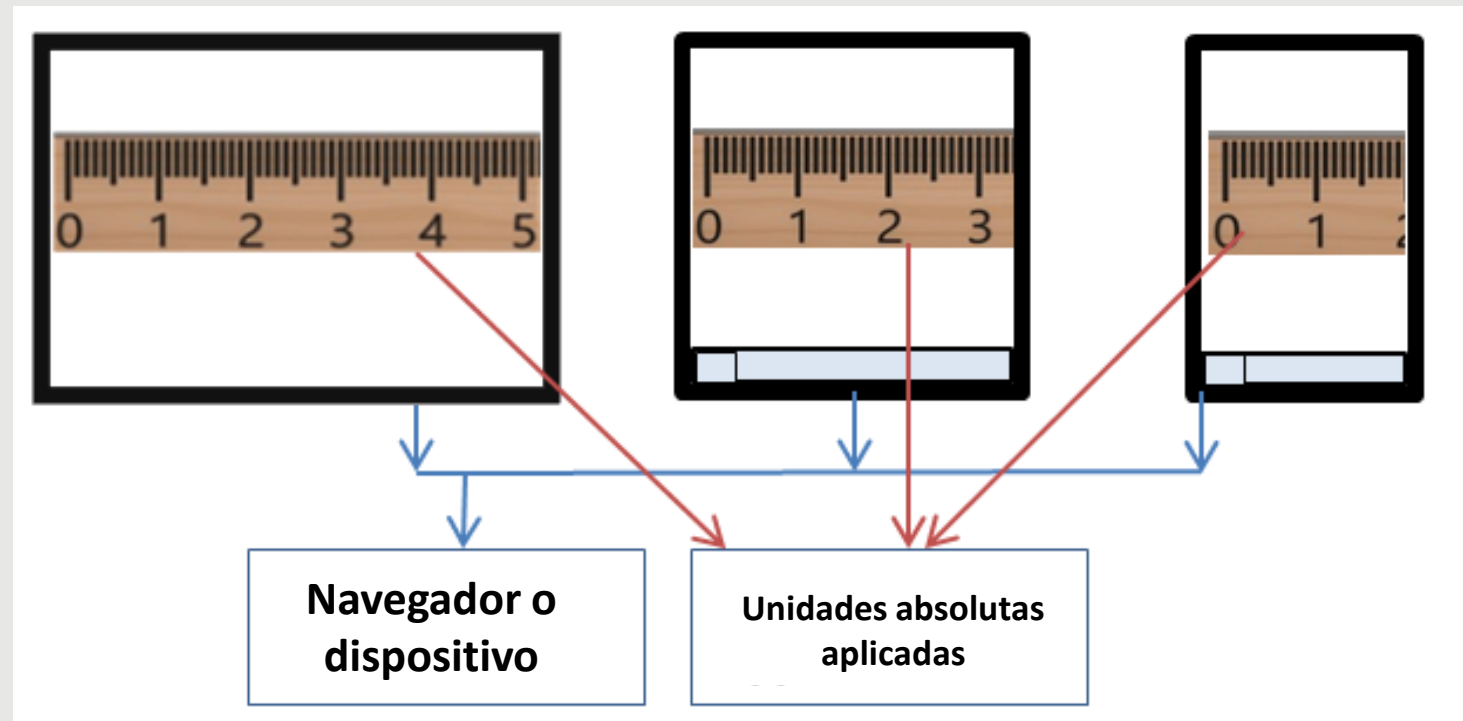
# Unidades CSS

- CSS cuenta con múltiples unidades para expresar longitudes.
- Propiedades como width, margin, padding, font-size, hacen uso de ellas.
- Las longitudes se expresan como un número seguido de una unidad (10px, 2em, etc).



# Unidades absolutas

- Estas unidades son fijas, no se ajustan en base al elemento padre.
- El elemento padre podría ser cualquier elemento o el navegador.

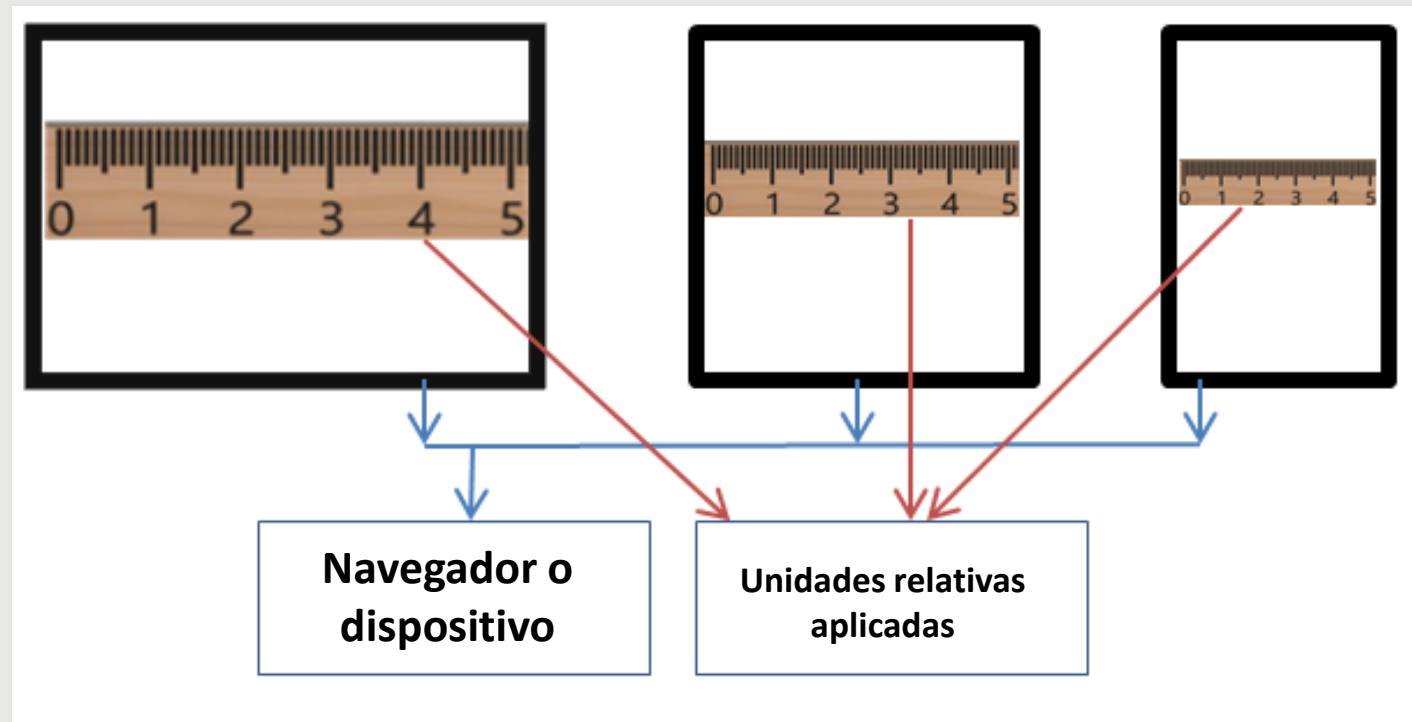


# Unidades absolutas

px	Pixeles. 1px es igual a 1/96 de una pulgada (in). Aunque sea una unidad absoluta, es relativa al dispositivo.
pt	Puntos (points) 1pt es igual a 1/72 in
cm	Centímetros
in	Pulgadas (inches) 1in es igual a 2.54cm
mm	Milímetros
pc	Pica. 1pc es igual a 12points (pt)

# Unidades relativas

- Trabajan en base a la longitud del elemento padre.
- Las unidades relativas aplicadas ajustan la longitud según elemento padre.

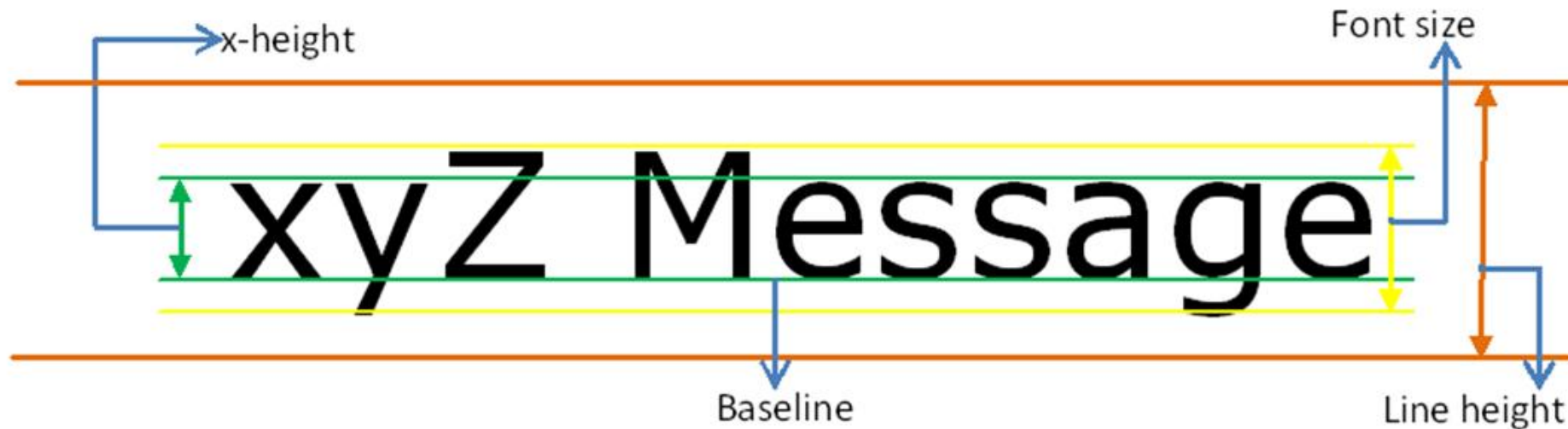


# Unidades relativas

em	Depende del tamaño de fuente
%	Se relaciona al tamaño de fuente base o del propio elemento
rem	Relativo al tamaño de fuente del elemento raíz (root) HTML
ex	El tamaño actual de "x" igual al tamaño de una "x" minúscula
vw	1vw (viewport width) es igual al 1% del ancho del viewport (ventana del navegador)
vh	1vh (viewport height) es igual al 1% de la altura del viewport
vmin	1vmin (viewport minimum) igual al 1% del menor valor entre vw o vh
vmax	1vmax (viewport maximum) igual al 1% del mayor valor entre vw o vh

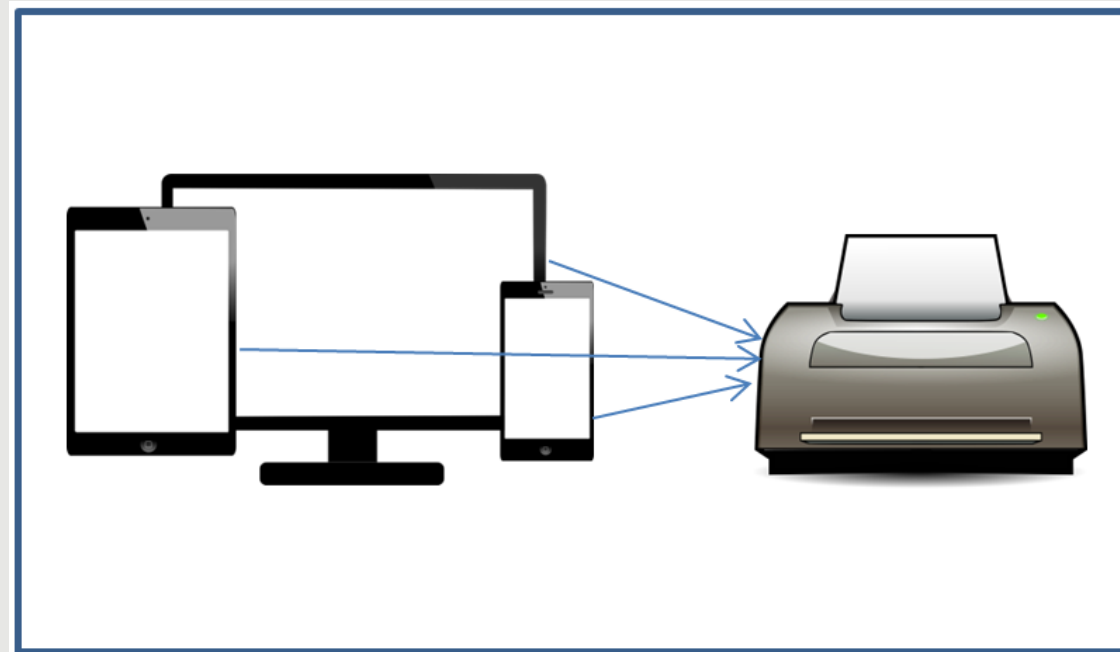
# Unidades CSS

- Las más usadas son em, px y %.
- Si no se especifica un tamaño de fuente, se colocara el valor por defecto, siendo este 15 o 16px.
- Si el tamaño de fuente es 16px, entonces 1em es igual a 16px cuando no cambia el tamaño de fuente del elemento padre.



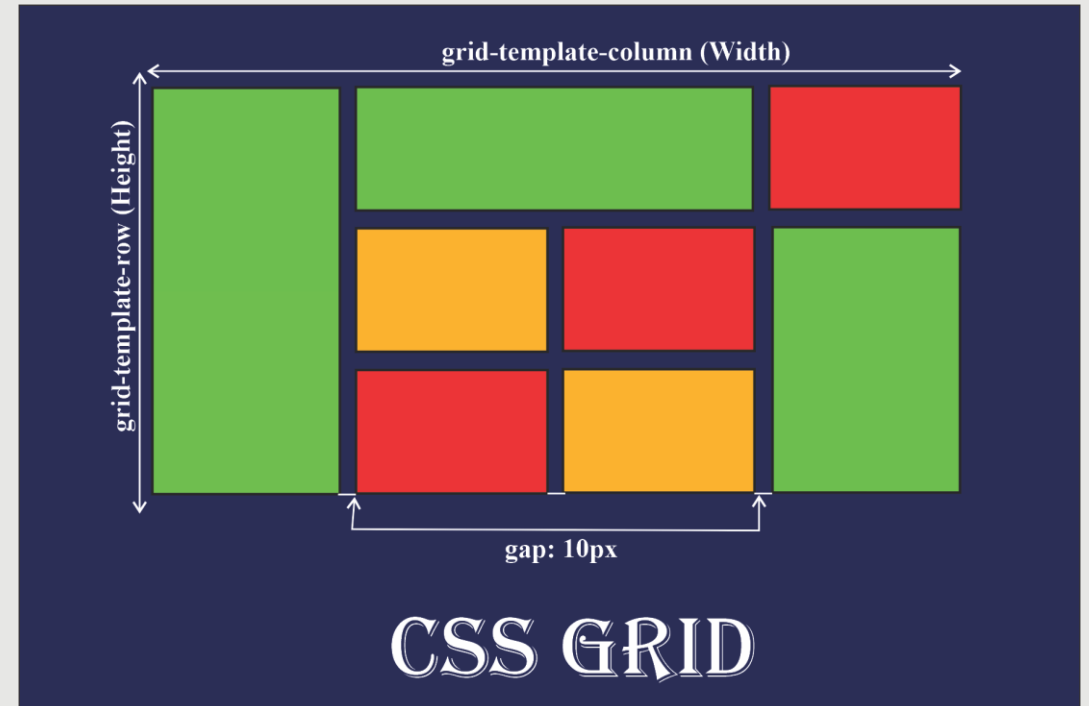
# Diferentes dispositivos

- Las páginas web pueden ser visualizadas usando diferentes dispositivos (teléfonos inteligentes, tabletas, laptops, PCs).
- Cada dispositivo contara con alto y ancho diferente.
- Lo mejor es usar unidades relativas para uso en pantalla y relativas para impresión.



# CSS Grid

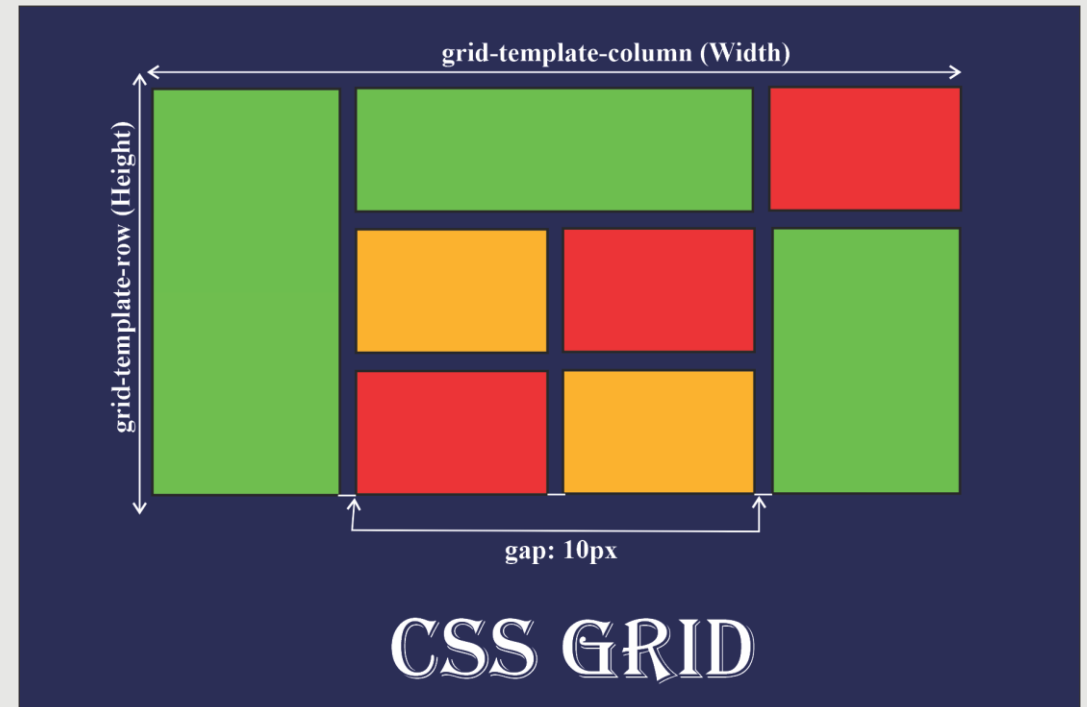
- CSS Grid es una forma de controlar la disposición de los elementos en nuestras páginas.
- La cuadrícula (Grid) esta ordenada en filas y columnas.
- Permite la creación de layouts bi-dimensionales.





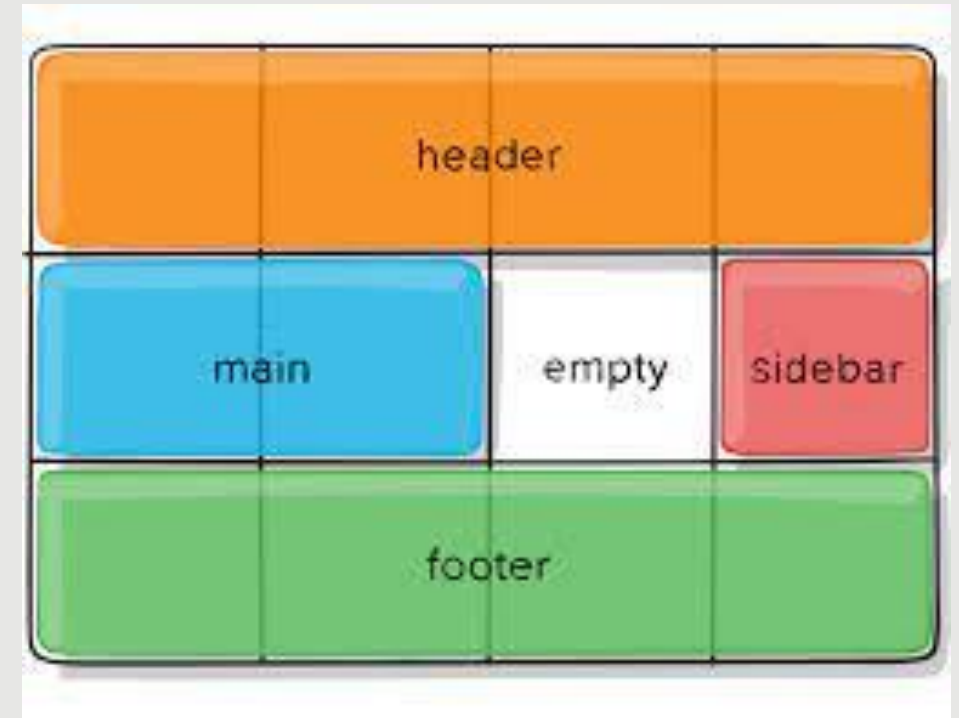
# Características de Grid

- **Tamaños flexibles:** Con el uso de la unidad fr (fracción), podemos crear rejillas organizadas y responsivas.
- **Colocación de elementos:** Es mucho más fácil organizarlos en el contenedor.
- **Control de la alineación:** Podremos organizar elementos vertical y horizontalmente.



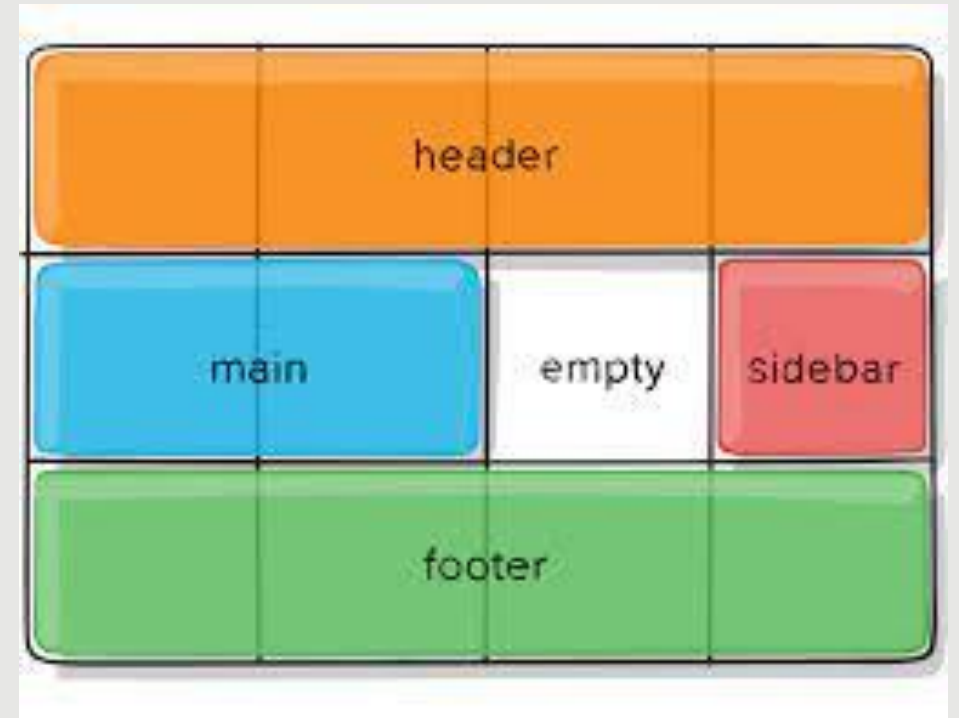
# Beneficios de Grid

- Es soportada por la mayoría de navegadores web.
- No es necesario cambiar nuestro HTML para cambiar la posición de estos.
- Ideal para diseños complejos debido a su disposición en dos dimensiones.
- Permite agregar espacio(gap) entre filas y columnas.



# Usando Grid

- Las siguientes propiedades se aplicaran al contenedor,
- Iniciamos con `display: grid`
- O bien con `display: in-line grid`;



# Columnas y filas

- La disposición de los elementos se hace en 2 dimensiones.

Column	Column	Column

Row			
Row			
Row			

# Espacios (gaps)

- Los espacios entre cada columna o fila son llamados gaps.

