CSS

Cascade Style Sheet

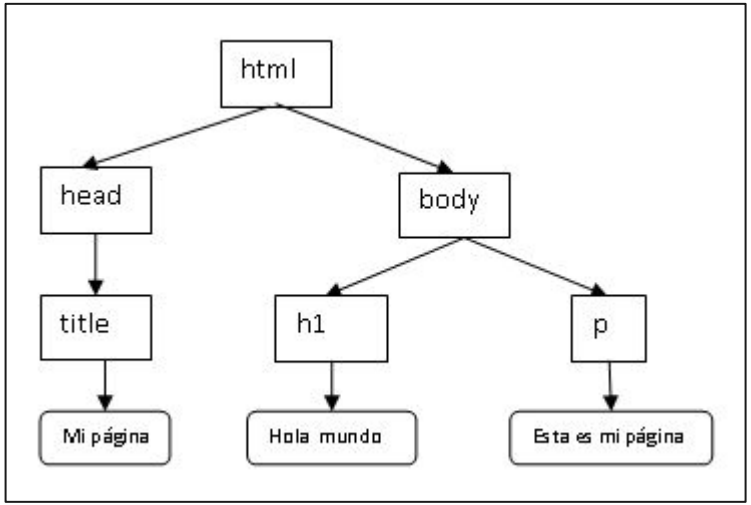
¿Qué es CSS?

CSS, en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado (HTML).

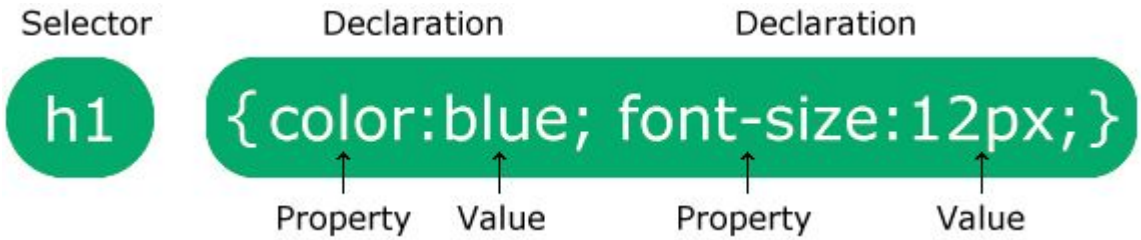
Se usa para separar el contenido de la presentación

Antes de empezar, el DOM

```
<html>
<head>
    <title>mi página</title>
</head>
<body>
     <h1>Hola mundo</h1>
          <p>Esta es mi página.</p>
</body>
</html>
```



Sintaxis


Selector
Declaration
Declaration
h1
{color:blue; font-size:12px;}
Property
Value
Property
Value

Insertar CSS

1. Inline

```
<p style="color:red;">This is a paragraph.</p>
```

2. Interno

```
<style>
body {
  background-color: linen;
}
</style>
```

3. Externo

```
<link rel="stylesheet" href="mystyle.css">
```

Comentarios

```
/* This is
a multi-line
comment */

p {
  color: red;
}
```

```
p {
  color: red; /* Set text color to red */
}
```

Visibilidad de un elemento

- `display: block | inline | inline-block | none ;`
- `visibility: visible | hidden ;`

Prueba esto:

1. Crea 3 divs que sean iguales.
2. Configura ‘display’ para que aparezcan en línea.
3. Encuentra la diferencia entre display y visibility

Selectores

- **Simple**
- Combinadores
- Pseudoclases
- Pseudoelementos
- Atributos

```
p {
  text-align: center;
  color: red;
}
```

```
.center {
  text-align: center;
  color: red;
}
```

```
#para1 {
  text-align: center;
  color: red;
}
```

Selectores

- Simple
- **Combinadores**
- Pseudoclases
- Pseudoelementos
- Atributos

se usan en conjunto con selectores simples (clase, id, elemento) para hacer que una regla sea más específica.

- Selector Descendente (**espacio**)
- Selector de hijo directo (**>**)
- Selector de elemento adyacente (**+**)
- Selector basado en elemento precedente con el mismo padre(**~**)

Selector Descendente (espacio)

Este selector afecta a todos los elementos que sean descendientes de otro elemento especificado, sin importar el nivel de profundidad al que se encuentren.

```
1
2 <div class="contenido">
3     <p>Párrafo 1</p>
4     <div class="noticias">
5         <p>Párrafo 2</p>
6         <p>Párrafo 3</p>
7     </div>
8 </div>
9
```

```
1
2 div.contenido p{
3     color: red;
4 }
5
```

Selector de hijo directo (>)

actúa sobre todos aquellos elementos que sean hijos de otro elemento especificado, pero que se encuentren en el primer nivel, es decir, si están dentro de otro elemento hijo de ese mismo padre, no serán tomados en cuenta.

```
1
2 <div class="contenido">
3     <p>Párrafo 1</p>
4     <div class="noticias">
5         <p>Párrafo 2</p>
6         <p>Párrafo 3</p>
7     </div>
8 </div>
9
```

```
1
2 div.contenido > p{
3     color: red;
4 }
5
```

Selector de elemento adyacente (+)

afecta a los elementos que, teniendo el mismo elemento como padre, estén inmediatamente seguidos uno de otro, esta relación se representa con el símbolo + entre los selectores.

```
1
2 <div class="contenido">
3     <p>Párrafo 1</p>
4     <div class="noticias">
5         <p>Párrafo 2</p>
6         <p>Párrafo 3</p>
7     </div>
8 </div>
9
```

```
1
2 div.noticias p + p {
3     color: red;
4 }
5
```

Selector basado en elemento precedente con el mismo padre(~)

actúa sobre aquellos elementos que se encuentren precedidos por un elemento específico y que tengan como padre al mismo elemento

```
<div class="contenido">
    <p>Párrafo 1</p>
    <div class="anuncios">
     <a href="publicidad.html" >Link a publicidad</a>
    </div>
    <div class="noticias">
        <p>Párrafo 2</p>
        <p>Párrafo 3</p>
    </div>
</div>
```

```
p ~ div {
  margin-bottom: 20px;
  border: 1px solid blue;
}
```

Selectores

- Simple
- Combinadores
- **Pseudoclases**
- Pseudoelementos
- Atributos

se utilizan para dar estilos a elementos respecto al comportamiento que experimentan en determinado momento

:hover Aplica estilos cuando pasamos el ratón sobre un elemento.

```
a:hover {
 background-color: cyan;
 padding: 2px
}

div:hover a {
 background-color: steelblue;
 color: white;
}
```

Selectores

- Simple
- Combinadores
- **Pseudoclases**
- Pseudoelementos
- Atributos

se utilizan para dar estilos a elementos respecto al comportamiento que experimentan en determinado momento

:hover Aplica estilos cuando pasamos el ratón sobre un elemento.

:focus cuando el elemento tiene el foco del navegador

:active cuando pulsa con el ratón sobre un elemento

Selectores

- Simple
- Combinadores
- **Pseudoclases**
- Pseudoelementos
- Atributos

se utilizan para dar estilos a elementos respecto al comportamiento que experimentan en determinado momento

:first-child selecciona la primera aparición de elemento
:last-child

```
p em:first-child {
  color: red;
}
```

```
p:first-child {
   color:red;
}
```

```
p:first-child em {
  color: red;
}
```

Selectores

se utilizan para dar estilos a elementos respecto al comportamiento que experimentan en determinado momento

- Simple
- Combinadores
- **Pseudoclases**
- Pseudoelementos
- Atributos

:nth-child(n):
:nth-last-child(n) selecciona a partir del último elemento.

```
:nth-child(7) Representa el séptimo elemento.
:nth-child(5n) Representa los elementos 5, 10, 15, etc.

:nth-child(odd) Representa los elementos impares
:nth-child(even) Representa los elementos pares
```

Selectores

- Simple
- Combinadores
- Pseudoclases
- **Pseudoelementos**
- Atributos

Los pseudo-elementos en CSS son un mecanismo para acceder a partes del HTML que no tienen asociado un nodo en el **DOM**

```
selector::pseudo-element {
  property: value;
}
```

Selectores

- Simple
- Combinadores
- Pseudoclases
- **Pseudoelementos**
- Atributos

Los pseudo-elementos en CSS son un mecanismo para acceder a partes del HTML que no tienen asociado un nodo en el **DOM**

```
p::first-line {
  color: blue;
  text-transform: uppercase;
}
```

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes#Index_of_standard_pseudo-classes ← Aquí están todos los pseudo-elementos

Selectores

- Simple
- Combinadores
- Pseudoclases
- **Pseudoelementos**
- Atributos

`::before` sirve para añadir contenido *antes* de un elemento

`::after` sirve para añadir contenido *después* de un elemento

`::first-letter` selecciona la *primera letra* de un elemento de tipo «bloque» (es decir, los que tienen como `display` valores tipo `block`, `inline-block`, `table-cell`, etc)

`::first-line` selecciona la *primera línea* de un elemento que, como en el caso anterior, sea de tipo «bloque»

`::selection` hace referencia a las partes del documento que el visitante haya seleccionado

Selectores

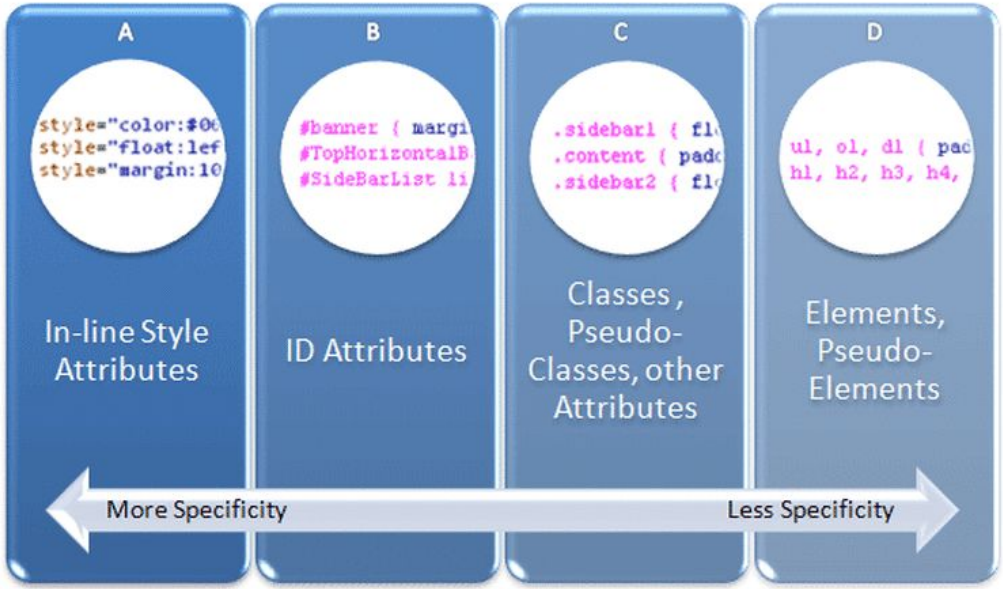
- Simple
- Combinadores
- Pseudoclases
- Pseudoelementos
- **Atributos**

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore ****magna aliqua.

```
a[title$="sample"]{
    color:#ea1d1d;
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing 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.

¿Quién tiene preferencia?



¿Quién tiene preferencia?

```
p {
  background-color: red !important;
}
```

Image

```
<img src="img_girl.jpg"
      alt="Girl in a jacket"
      width="500"   height="600">
```

Image

```
.img{
vertical-align: bottom | middle | top
float: left | right
}
```

+ info en Tryit Editor v3.7 (w3schools.com)

background

```
body {
    background-color: red;
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
    background-attachment: scroll;
}
```

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

table

```
<table class="t-products"  >
        <caption>HTML5 - Table</caption>
        <thead>
        ...
        </thead>
        <tfoot>
        ...
        </tfoot>

        <tbody>
                <tr>
                        <td>Line</td>.
                </tr>
                <tr>
                        ...
                </tr>
        </tbody>

</table>
```

Unidades de medida absolutas

- `in`, del inglés "inches", pulgadas (1 pulgada son 2.54 centímetros)
- `cm`, centímetros
- `mm`, milímetros
- `pt`, puntos (1 punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros)
- `pc`, picas (1 pica equivale a 12 puntos, es decir, unos 4.23 milímetros)

Unidades de medida relativas

- `em`, relativa respecto del tamaño de letra empleado.
- `ex`, relativa respecto de la altura de la letra `x` *"equis minúscula"*) del tipo de letra que se esté utilizando
- `px`, (píxel) relativa respecto de la pantalla del usuario

Unidades de medida relativas

- `em`, para textos
- `ex`, relativa respecto de la altura de la letra `x` *"equis minúscula"*) del tipo de letra que se esté utilizando
- `px`, para divs y otros elementos

¿rem vs em? +info Como utilizar em y rem en CSS – CybMeta

border

```
p {
  border: 5px solid red;
}
```

```
p.one {

    border-style: solid;
    border-width: 5px;
}
```

border

```
p {
  border: 5px solid red;
}
```

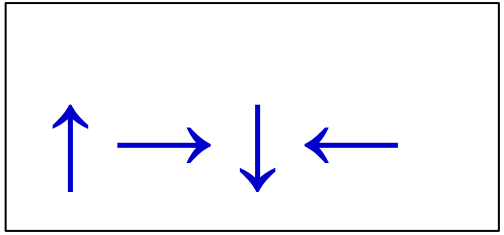
```
p.one {
    border-style: solid;
    border-width: 5px;
}
```

border-width: 5px ↑→↓←

border-width: 5px **10px** ↑→↓←

border-width: 5px **10px** **15px** ↑→↓←

border-width: 5px 10px 15px 20px ↑→↓←



border

```
p {
  border: 5px solid red;
}
```

```
border-width: 5px ↑→↓←

border-width: 5px 10px ↑→↓←

border-width: 5px 10px 15px ↑→↓←

border-width: 5px 10px 15px 20px ↑→↓←
```

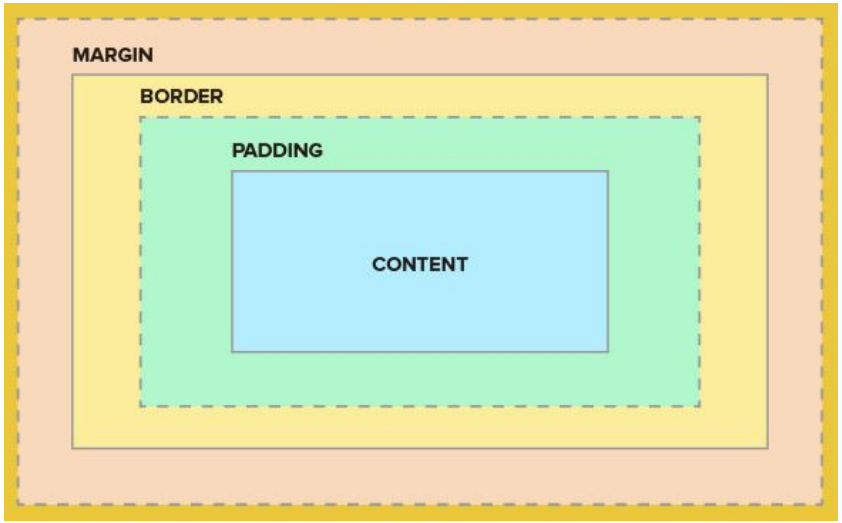
```
p.one {

    border-style: solid;
    border-width: 5px;
}
```

margin
padding

border

+ info en https://developer.mozilla.org/en-US/docs/Web/CSS/border

Box model



La “caja” es la suma del contenido, padding, border y margin, **pero se puede cambiar**

Box sizing

La propiedad box-sizing de CSS puede tener los valores

- **content-box:** por defecto.
- **border-box:** la propiedad border-box recalcula el ancho de una caja para que el width sea la estrictamente la suma de border + padding + ancho o altura que sea necesario para que se cumpla.

Crea una estructura de web con 3 columnas iguales que ocupen 33.33% cada una

https://codepen.io/carlosgv/pen/gOxLwxZ

Overflow

La propiedad overflow permite regular la visibilidad de los contenidos que sobresalen de una caja html.

Permite regular si los contenidos que sobresalen se seguirán viendo, si se ocultarán o si aparecerá una barra de scroll en el documento.

White-space

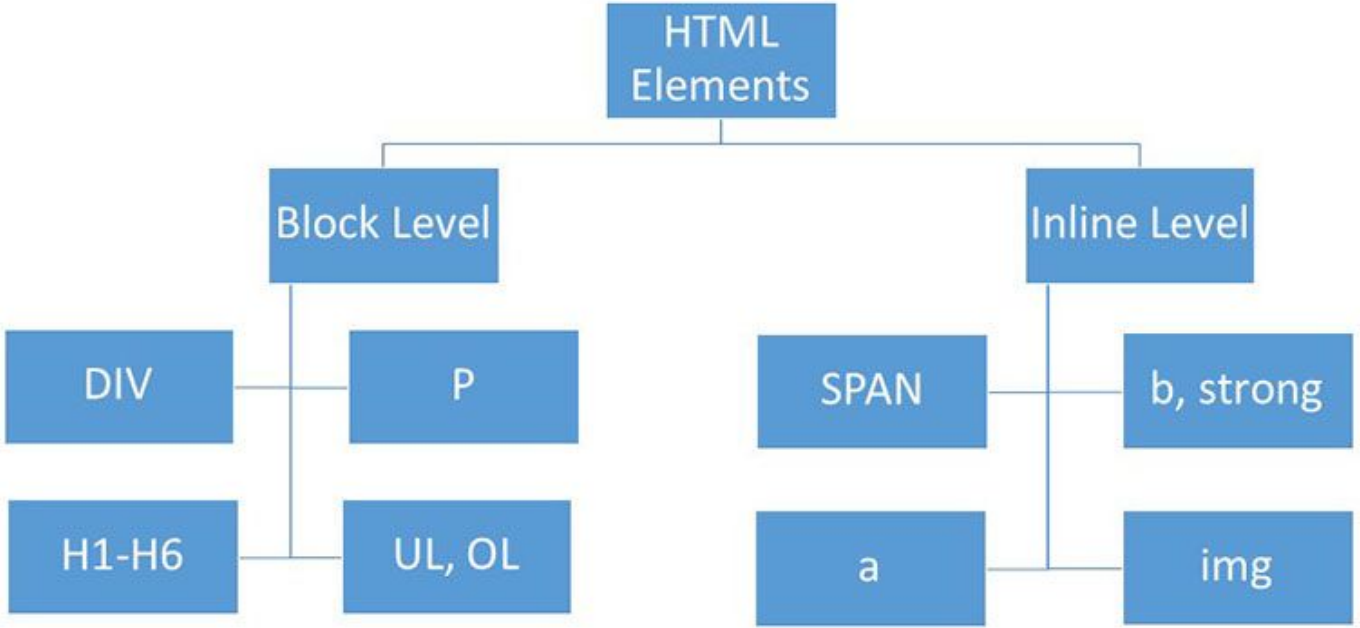
un "espacio en blanco" puede ser un salto de línea, un tabulador y un espacio en blanco normal. Los navegadores eliminan de forma automática todos los espacios en blanco sobrantes salvo el espacio en blanco que separa las palabras del texto.

Valor	Respeta espacios en blanco	Respeta saltos de línea	Ajusta las líneas
`normal`	no	no	si
`pre`	si	si	no
`nowrap`	no	no	no
`pre-wrap`	si	si	si
`pre-line`	no	si	si

Display

La propiedad display es tan compleja que casi ningún navegador es capaz de mostrar correctamente todos sus valores.

Definición	Establece el tipo de caja generada por un elemento
Valores permitidos	Uno y sólo uno de los siguientes valores: ▪ inline ▪ block ▪ list-item ▪ run-in ▪ inline-block ▪ table ▪ inline-table ▪ table-row-group ▪ table-header-group ▪ table-footer-group ▪ table-row ▪ table-column-group ▪ table-column ▪ table-cell ▪ table-caption ▪ none ▪ inherit


HTML Elements
Block Level
Inline Level
DIV
P
H1-H6
UL, OL
SPAN
b, strong
a
img

Display: none

El valor más sencillo de display es none que hace que el elemento no genere ninguna caja.

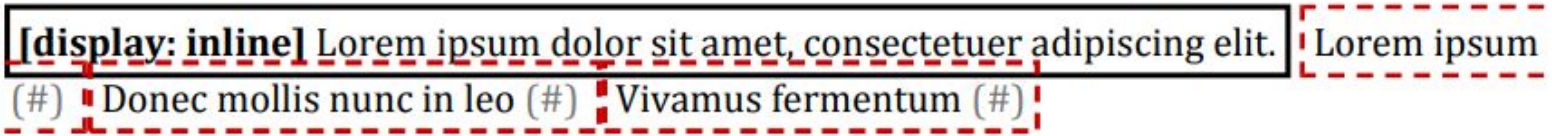
Si se utiliza la propiedad **display: none** sobre un elemento, todos sus descendientes también desaparecen por completo de la página.

Si se quiere hacer un elemento invisible, es decir, que no se vea pero que siga ocupando el mismo sitio, se debe utilizar la propiedad **visibility:hidden**.

Display:block | inline

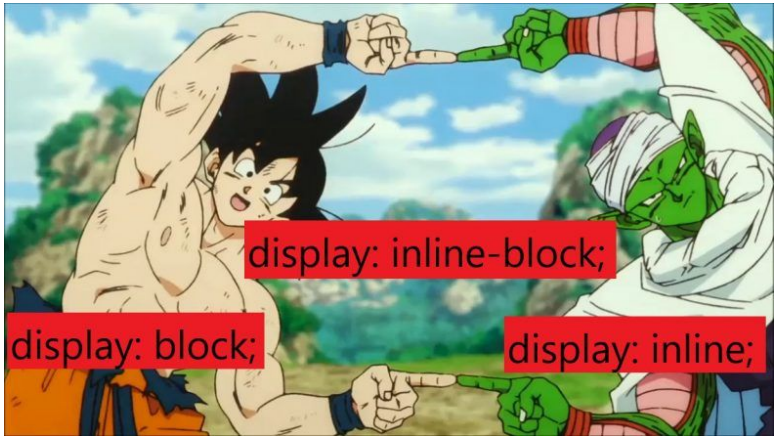
Hacen que la caja de un elemento sea de bloque o en línea respectivamente.

Si se aplica la propiedad display: inline a un párrafo, su caja se convierte en un elemento en línea y por tanto sólo ocupa el espacio necesario para mostrar sus contenidos.



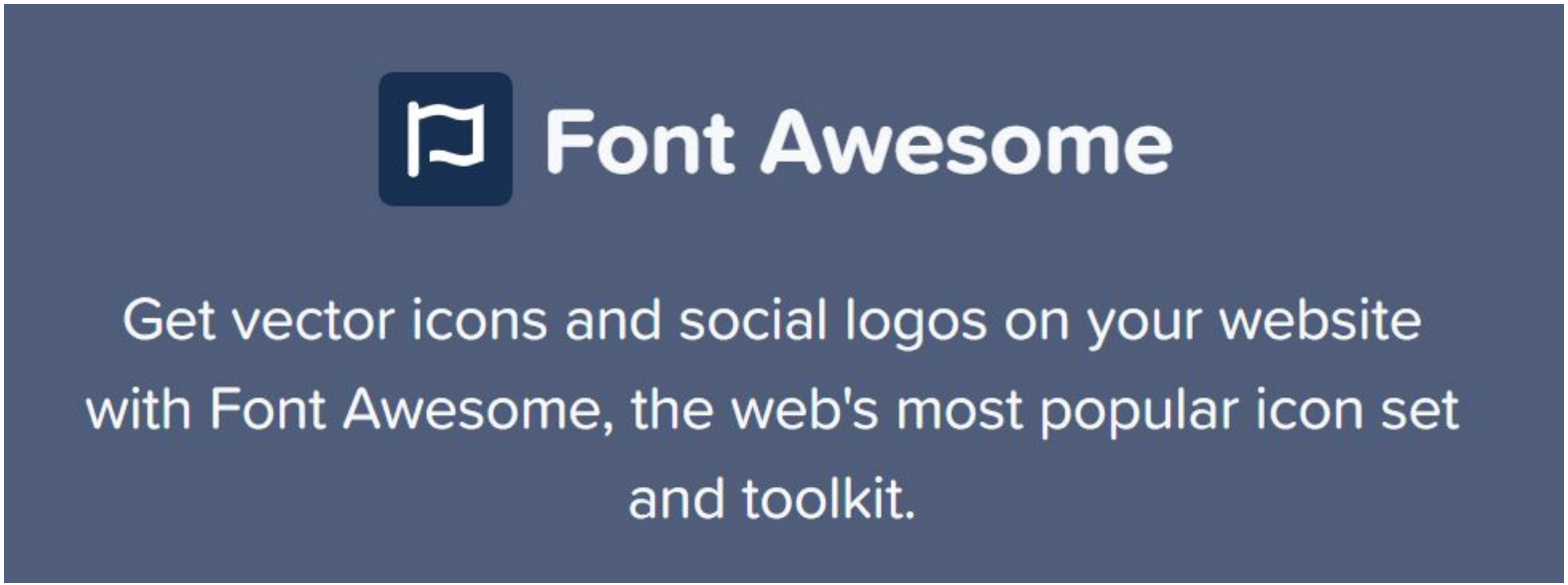
Display: inline-block

Una caja de tipo inline-block se comporta como si fuera de bloque, pero respecto a los elementos que la rodean es una caja en línea.



	Padding				Margin			
	Left	Right	Top	Bottom	Left	Right	Top	Bottom
Block	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Inline	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Inline-Block	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Font-awesome

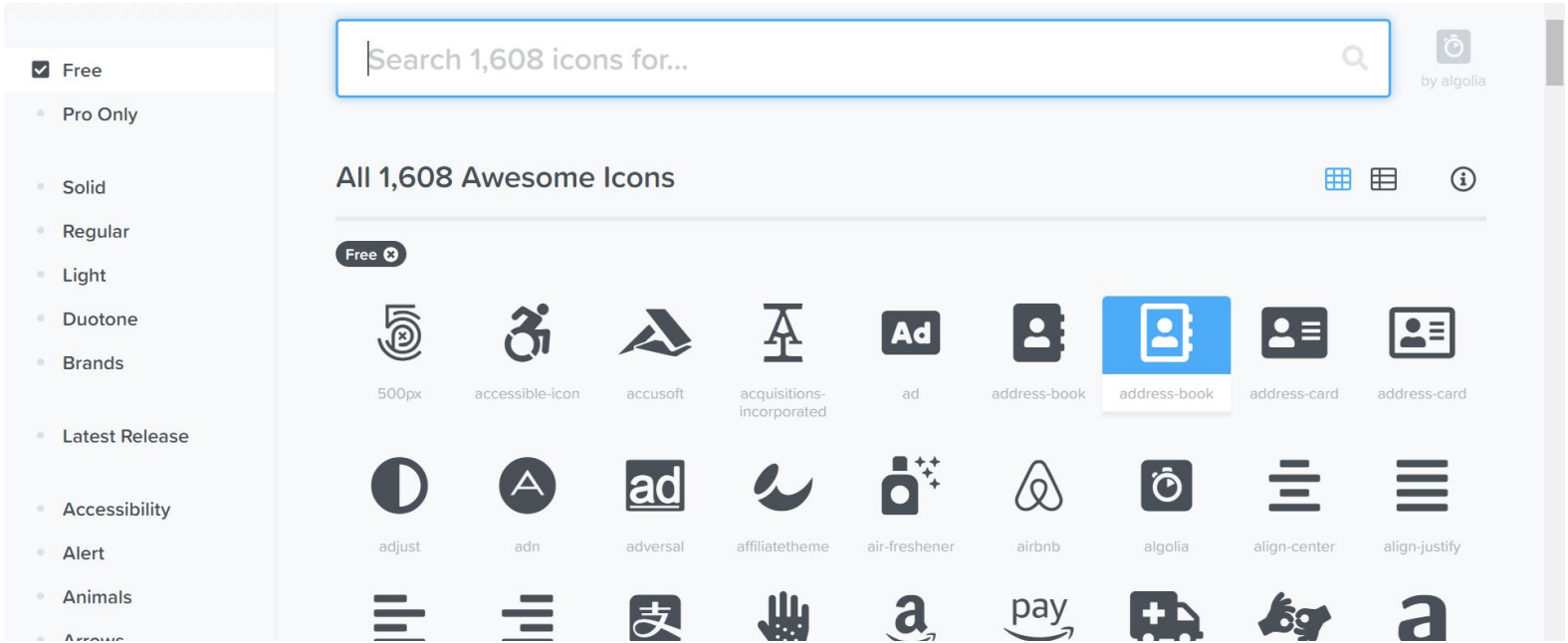


Font-awesome: Cómo se utiliza

1. Añade la librería a tu web HTML
   <link rel="stylesheet"
   href="**https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css**" />

2. Insertar los iconos con la etiqueta

Font-awesome: Galería completa v6



Font-awesome

```
<i class="far fa-address-book"></i>
```

FAR: Iconos de tipo regular
FAB: Iconos de marcas comerciales
FAS: Iconos solidos.


fas: Solid


fab: Brands


far: Regular

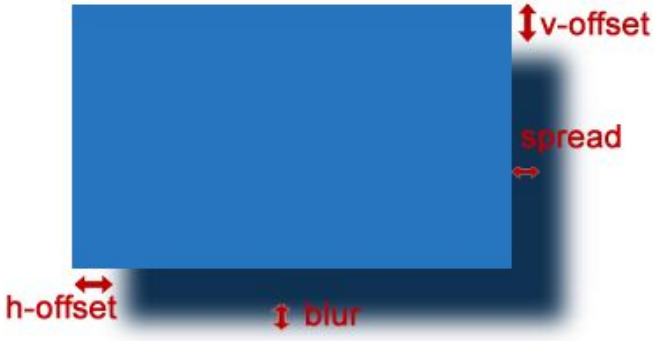
All Awesome Icons

Welcome Elements Console Sources Network Performance Memory

```
<html>
▼ <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="fontawesome-free-5.15.4-web/css/all.min.css">
    <style> i{font-size:40px;} .fas{color:red;} .fab{color:blue;} span{margin-
    left:10px;} </style>
  </head>
▼ <body>
  ▼ <p>
    ▶ <i class="fas fa-ambulance">…</i>
••• ▶ <i class="fas fa-balance-scale">…</i> == $0
      <span> fas: Solid </span>
    </p>
  ▶ <p>…</p>
  ▶ <p>…</p>
    <a href="https://fontawesome.com/v5.15/icons?d=gallery&p=2&m=free" target="_blank">
    All Awesome Icons</a>
  </body>
</html>
```

box-shadow

```
/* offset-x | offset-y | color */
box-shadow: 60px -16px teal;

/* offset-x | offset-y | blur-radius | color */
box-shadow: 10px 5px 5px black;

/* offset-x | offset-y | blur-radius | spread-radius | color */
box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);

/* inset | offset-x | offset-y | color */
box-shadow: inset 5em 1em gold;

/* Any number of shadows, separated by commas */
   box-shadow: 3px 3px red, -1em 0 0.4em olive;
```

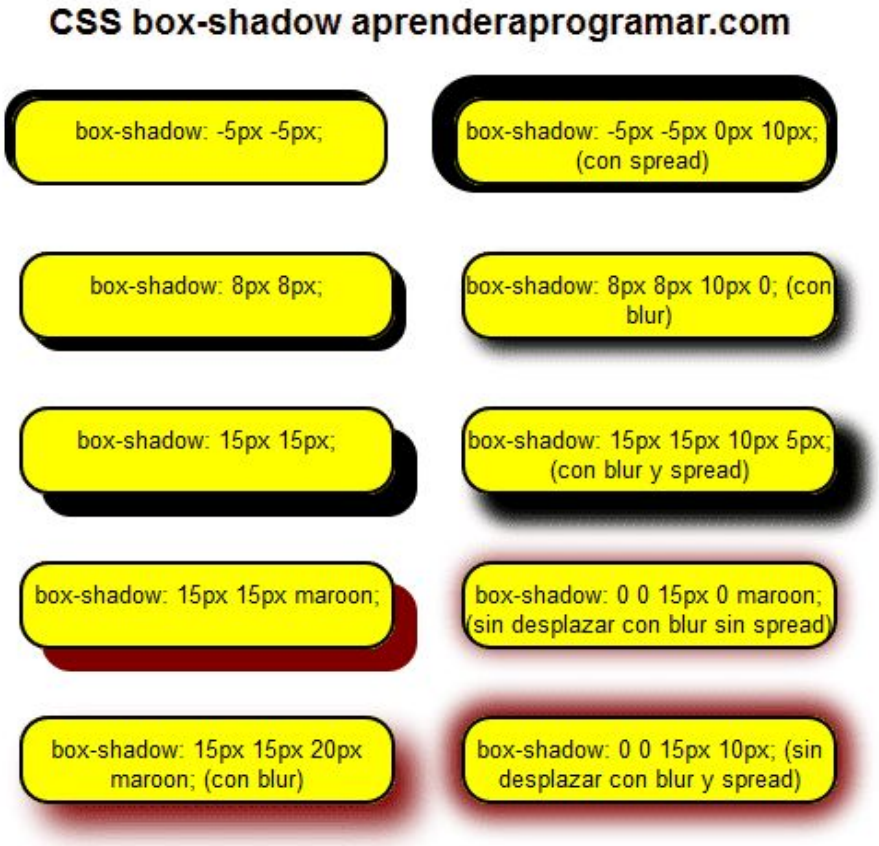


+info aquí

box-shadow

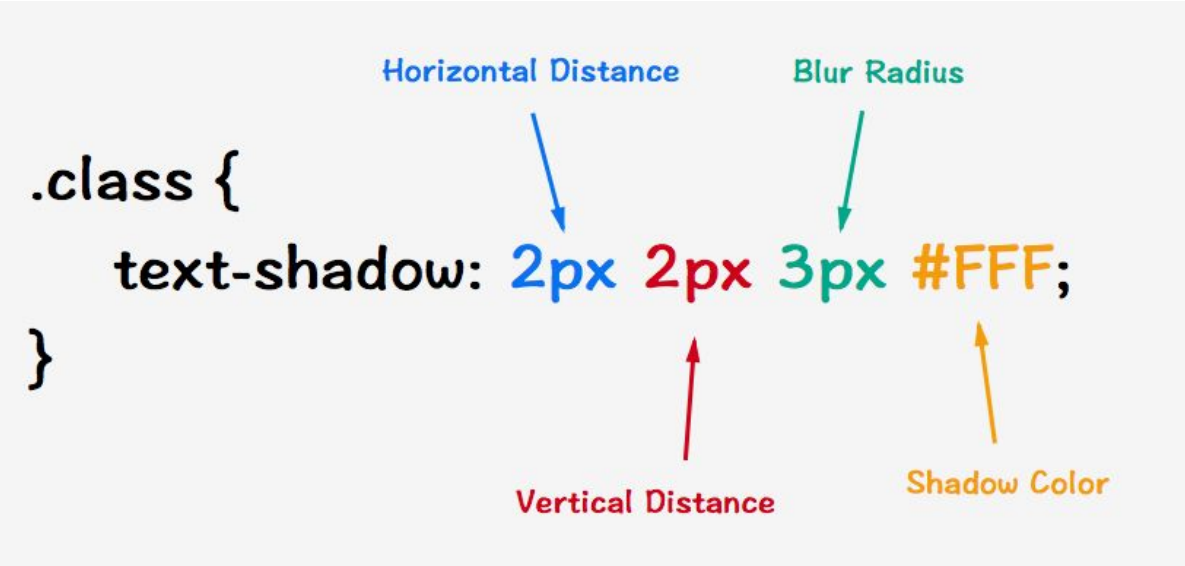
También podemos usar generadores de sombras:

https://html-css-js.com/css/generator/box-shadow/



text-shadow

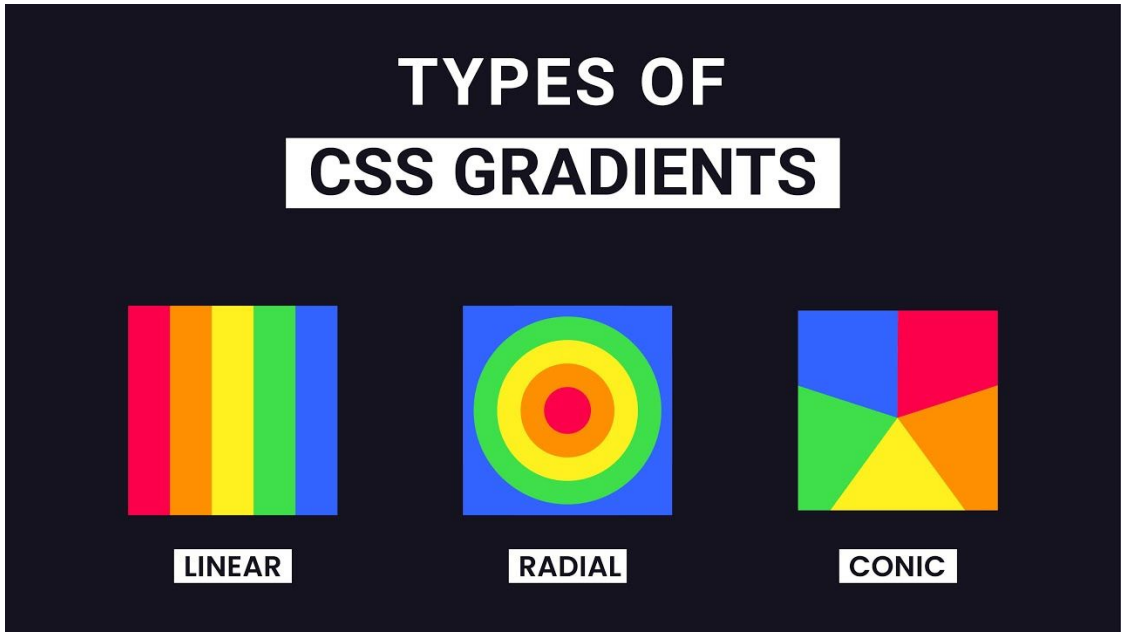
CSS3 Text Shadow Effect



También podemos usar generadores de sombras:

https://html-css-js.com/css/generator/box-shadow/

Gradient



Gradient

/* A gradient tilted 45 degrees, starting blue and finishing red */
linear-gradient(45deg, blue, red);

/* A gradient going from the bottom right to the top left corner, starting blue and finishing red */
linear-gradient(to left top, blue, red);

/* Color stop: A gradient going from the bottom to top, starting blue, turning green at 40% of its length, and finishing red */
linear-gradient(0deg, blue, green 40%, red);

/* Color hint: A gradient going from the left to right, starting red, getting to the midpoint color 10% of the way across the length of the gradient, taking the rest of the 90% of the length to change to blue */
linear-gradient(.25turn, red, 10%, blue);

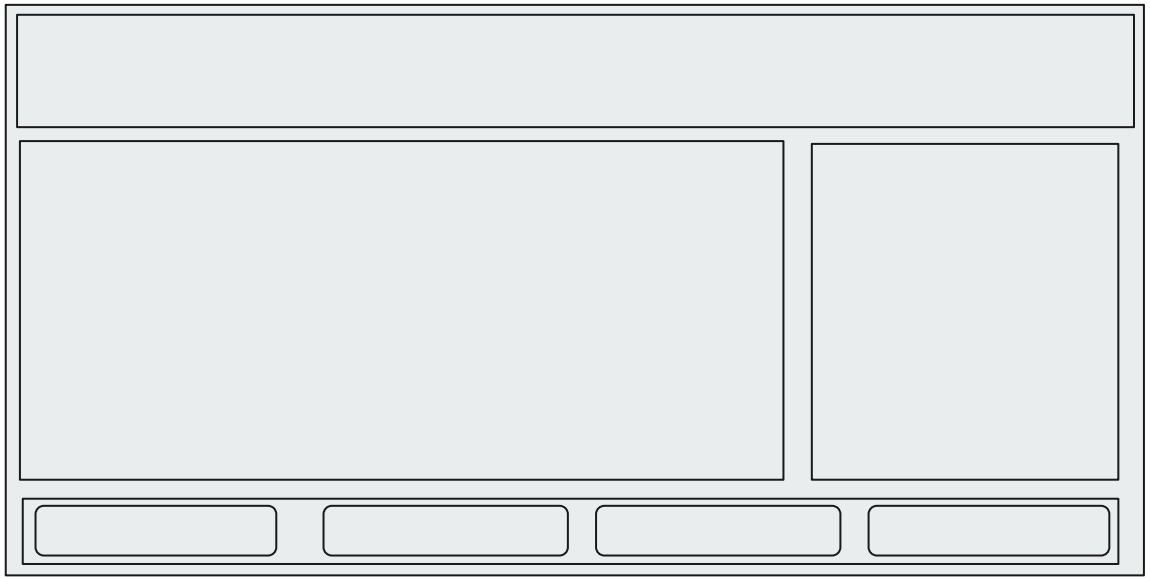
Gradient

Todos los tipos de degradados en
<gradient> - CSS: Cascading Style Sheets | MDN (mozilla.org)

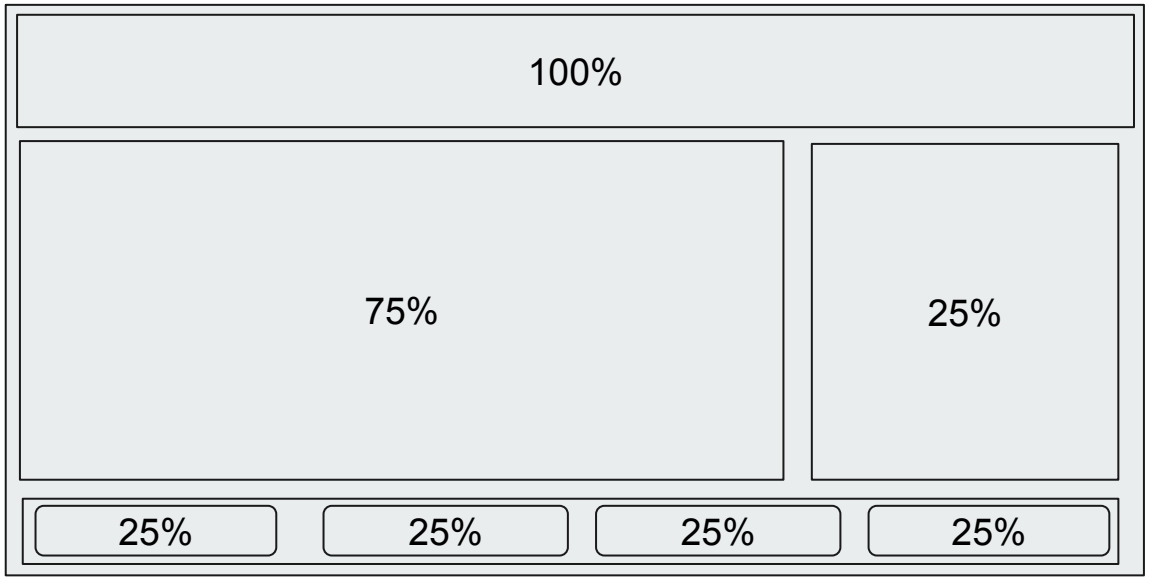
También podemos usar generadores de degradados:

https://html-css-js.com/css/generator/gradient/

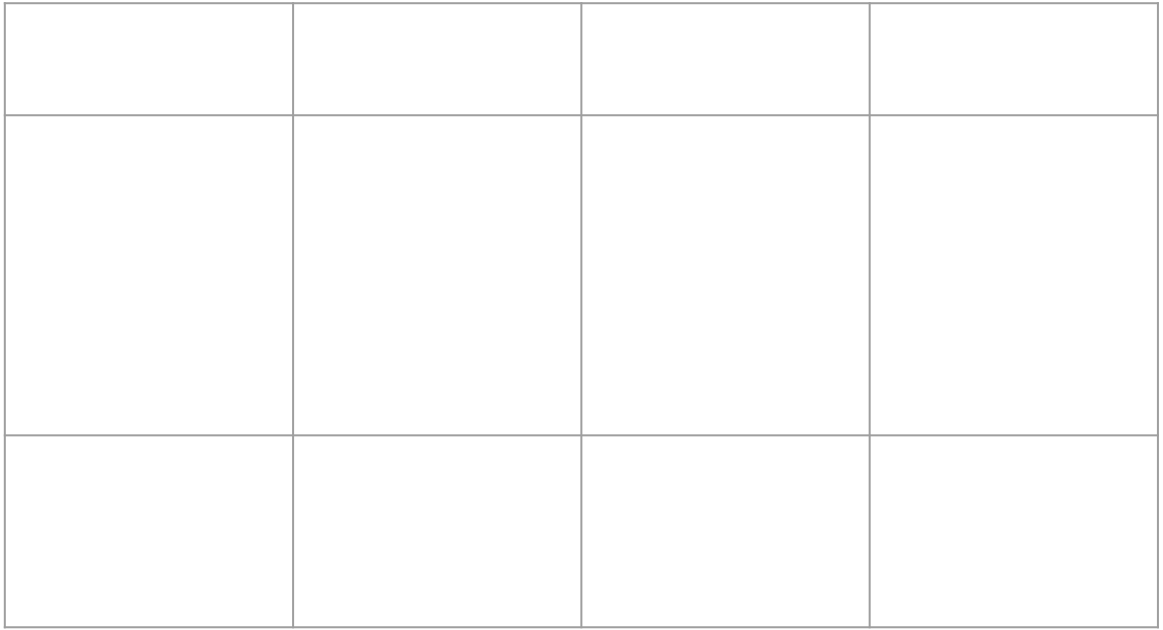
Grid System



Grid System

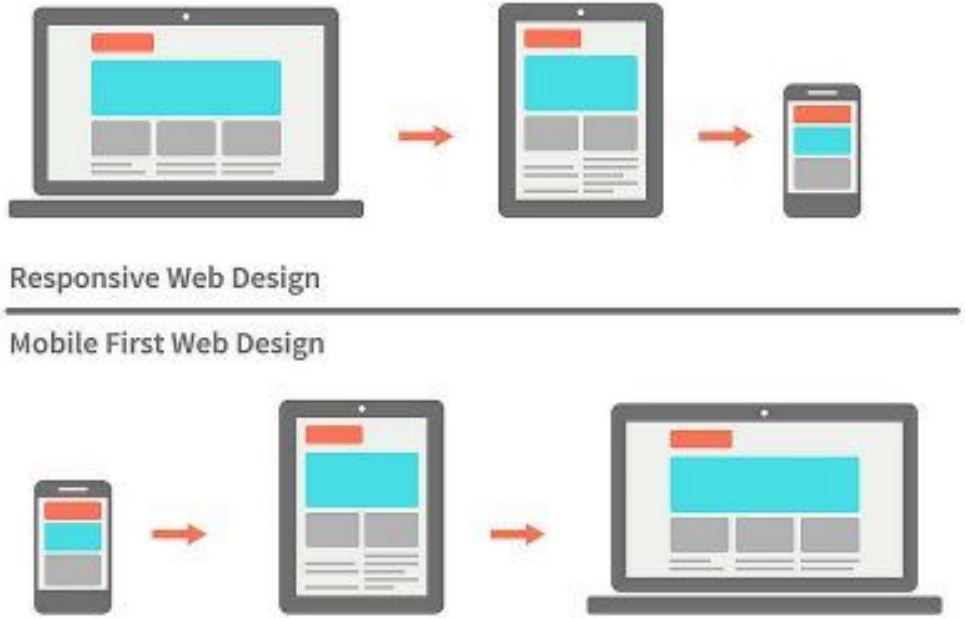


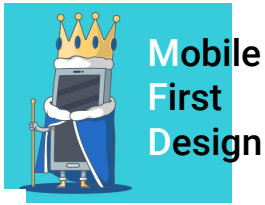
Grid System



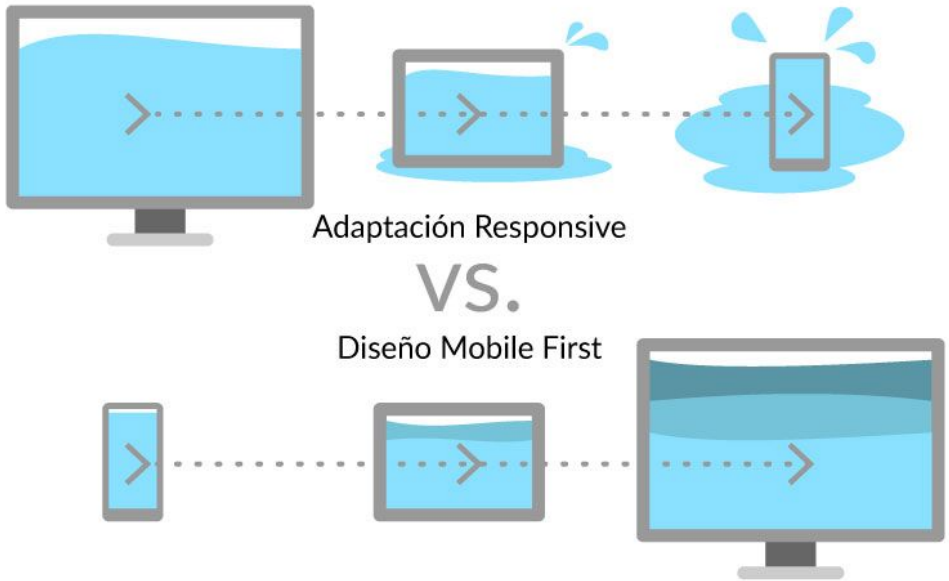
Diseño Responsive

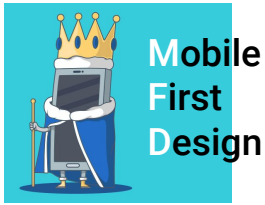
Diseño responsive





Diseño responsive

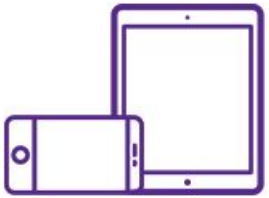




Device screen sizes for media queries

0-480

Smaller smartphones



481-768

Tablets & larger smartphones

769-1279

Laptops, larger tablets in landscape, and small desktops

1280+

Larger desktops and monitors

Media Query

Conjunto de reglas que se introducen en una hoja de estilo CSS con el objetivo de definir propiedades específicas para distintos tipos de medios

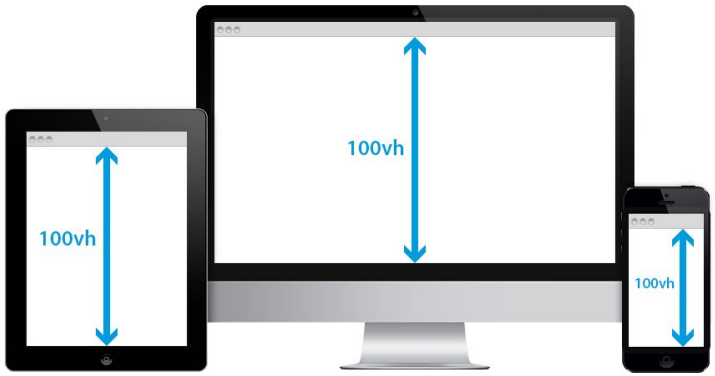
```
@media only screen and (max-width: 800px) {
  body {
    background-color: red;
  }
}
```

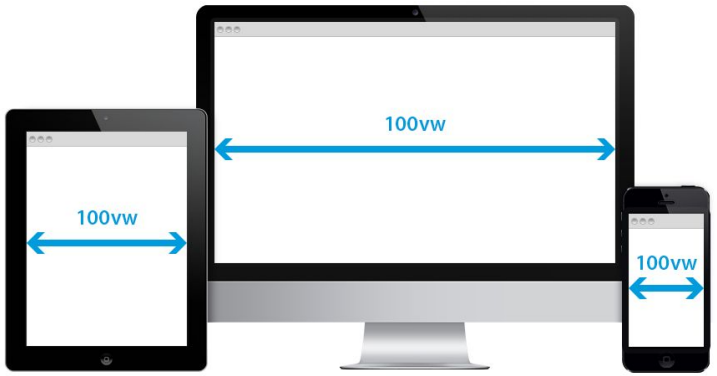
Device screen sizes for media queries

```
/* Extra small (XS) devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (S) (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (MD) (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (LG) (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (XL) (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

viewport


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





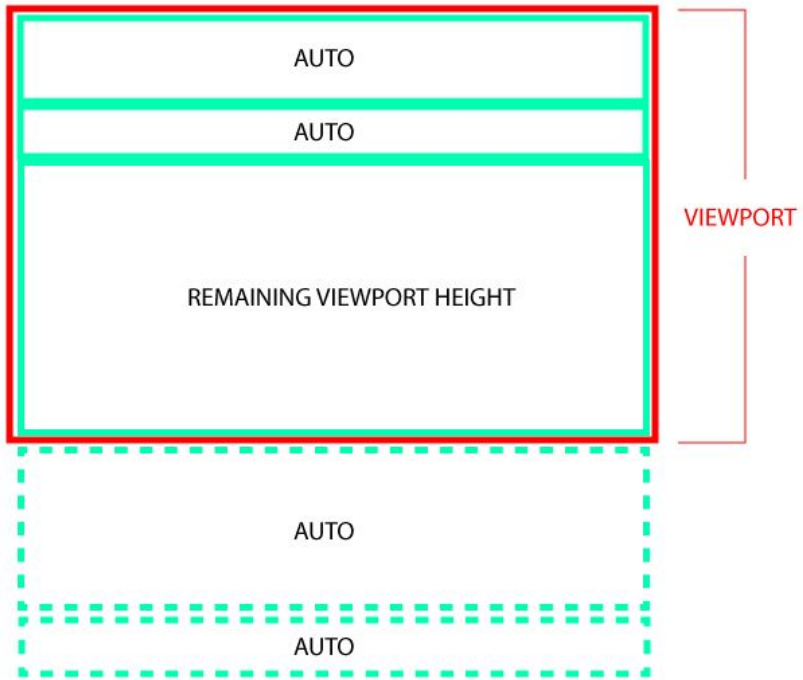
viewport

Without the viewport meta tag

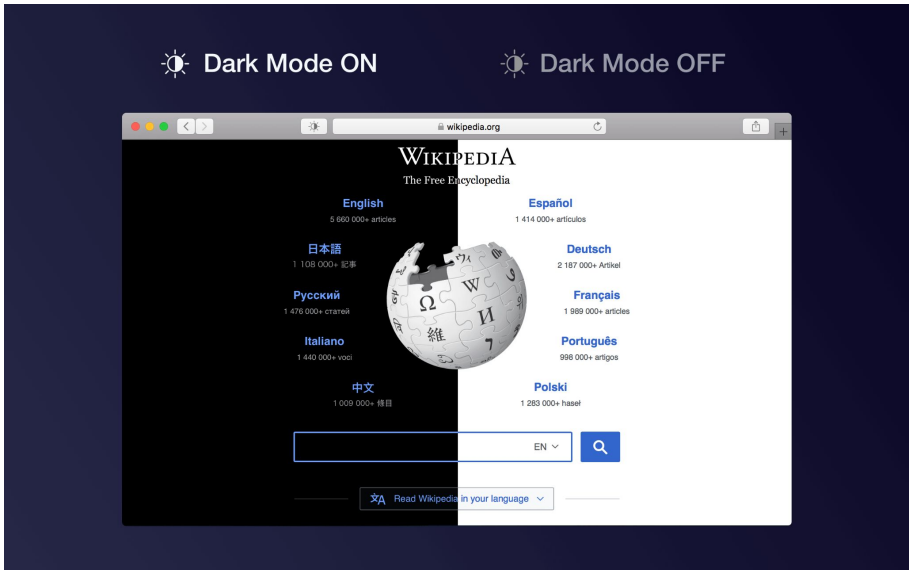
With the viewport meta tag

medidas vh y vw

- **1vh** = 1% de la altura del viewport
- **100vh** = altura del viewport
- **1vw** = 1% del ancho del viewport
- **100vw** = ancho del viewport



dark mode



"reduce el uso de energía y aumenta la accesibilidad para usuarios con baja visión o sensibles a la luz."

dark mode

```
@media (prefers-color-scheme: dark | light) {...}
```

`light`
Indicates that user has notified that they prefer an interface that has a light theme, or has not expressed an active preference.

`dark`
Indicates that user has notified that they prefer an interface that has a dark theme.

dark mode

`@media (prefers-color-scheme: dark | light) {...}`

```
body {
  background-color: white;
  color: black;
}
@media screen and (prefers-color-scheme: dark) {
  body {
    background-color: black;
    color: white;
  }
}
```

CSS custom properties

permite dar un valor personalizado a las propiedades.

```
.element {
    property: var(--variableName);
}
```

CSS custom properties

permite dar un valor personalizado a las propiedades.

```
.element {
    property: var(--variableName);
}
```

```
:root {
  --background-color: black;
 }
```

CSS custom properties

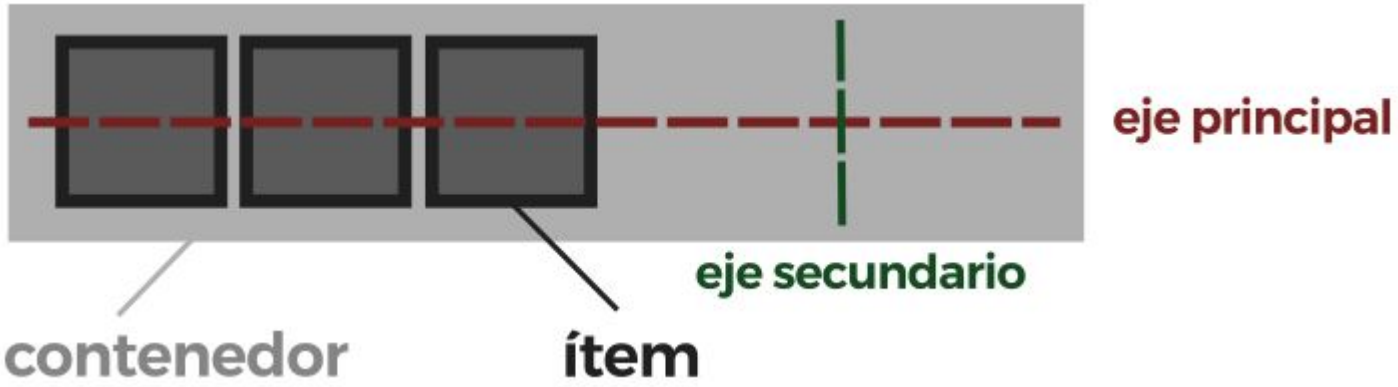
permite dar un valor personalizado a las propiedades.

```
:root {
  --background-color: black;
 }
```

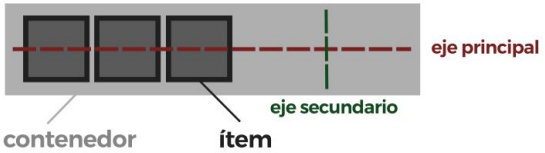
```
.element {
   background: var(--background-color, blue);
}
```

+info en [Variables CSS - CSS en español (lenguajecss.com)]

CSS Flex



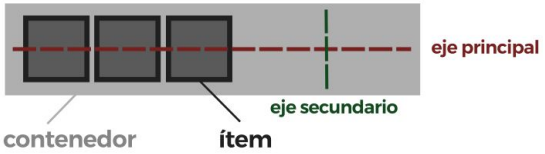
CSS Flex



Contenedor:

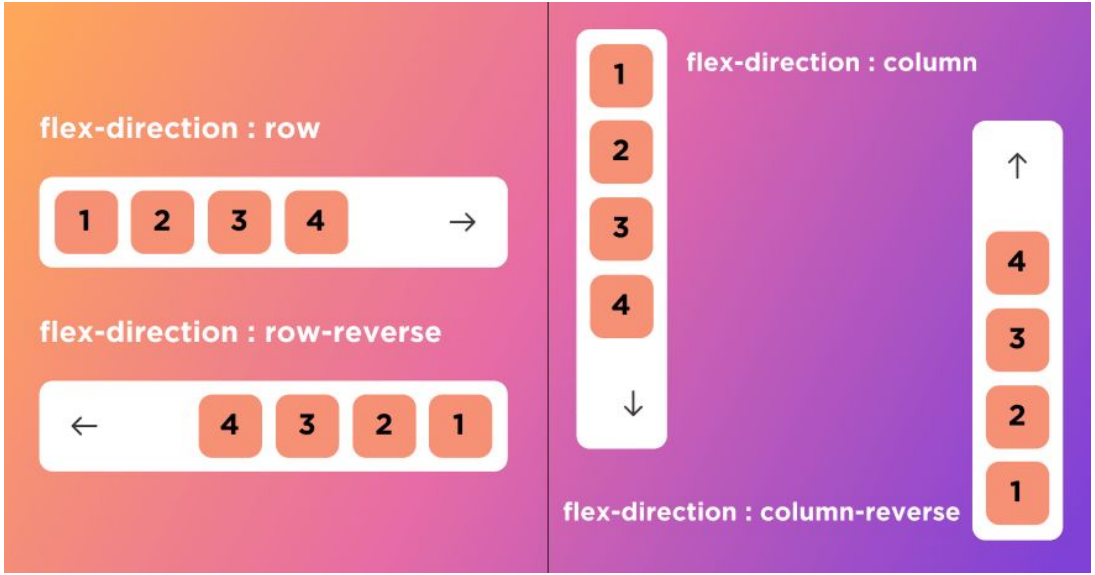
- **inline-flex** Establece un contenedor en línea, similar a inline-block (ocupa solo el contenido).
- **flex** Establece un contenedor en bloque, similar a block (ocupa todo el ancho del padre).

CSS Flex



Direction:

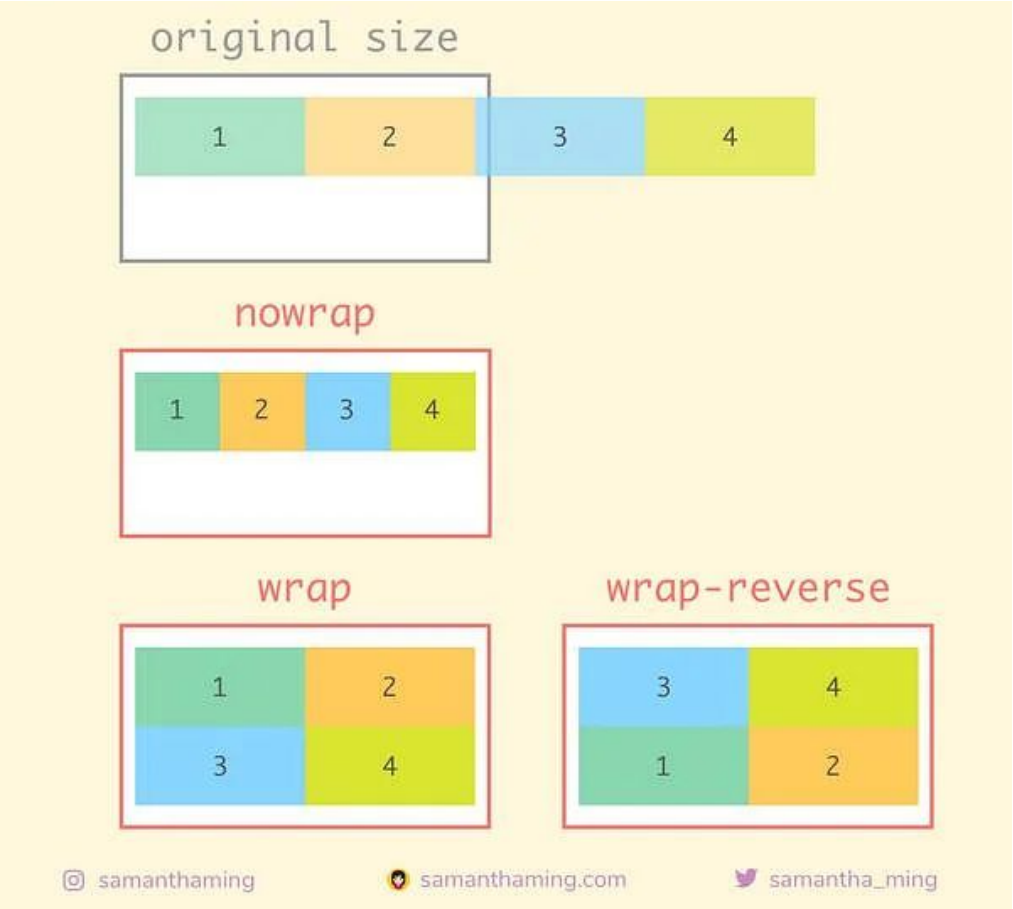
- **flex-direction:**
  - row
  - column
  - row-reverse
  - column-reverse



CSS Flex

Desbordamiento:

- **flex-wrap:**
  - **nowrap:** *default*
  - **wrap**
  - **wrap-reverse**



CSS Flex

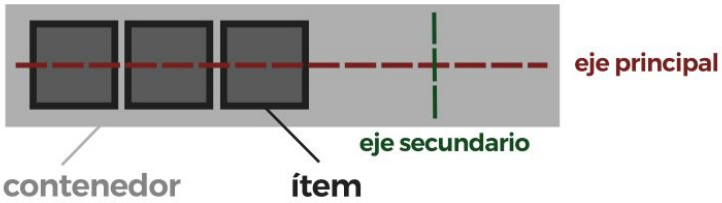
Atajo: flex-direction y flex-wrap se convierten en flex-flow

```
.container {
  /* flex-flow: <flex-direction> <flex-wrap>; */
  flex-flow: row wrap;
 }
```

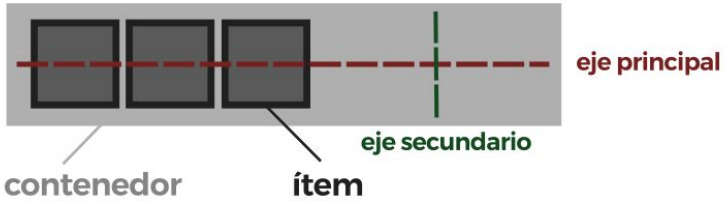
CSS Flex: justify-content

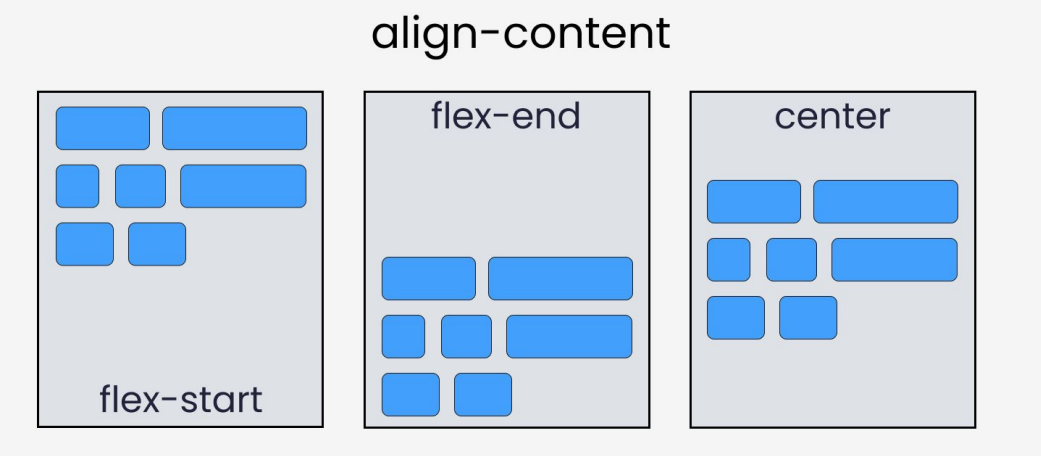
Alinear contenido:

- **justify-content: actúa en el eje principal**



CSS Flex: align-content

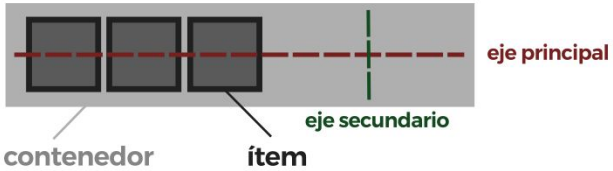




Alinear contenido:

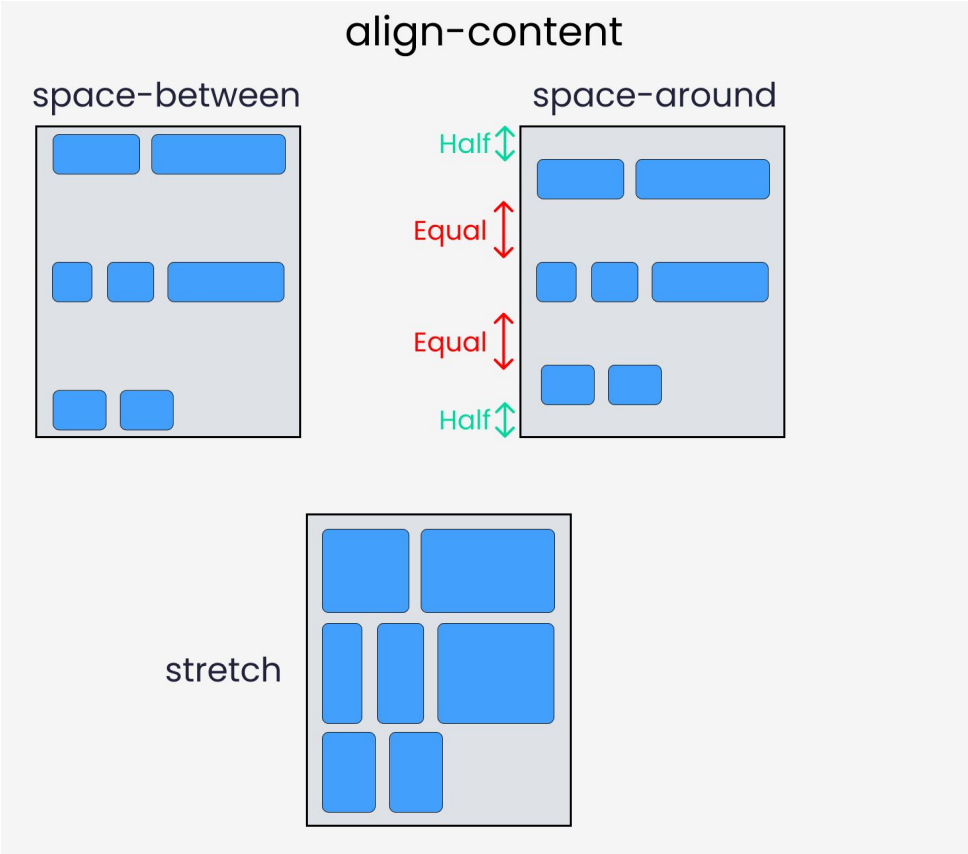
- **align-content:** actúa en el **eje secundario**

CSS Flex: align-content

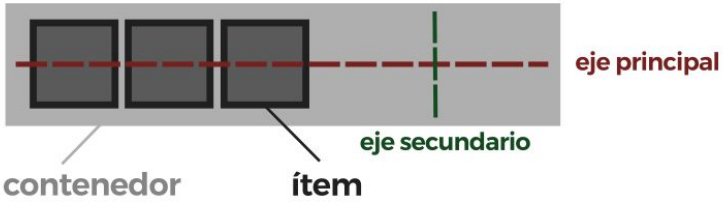


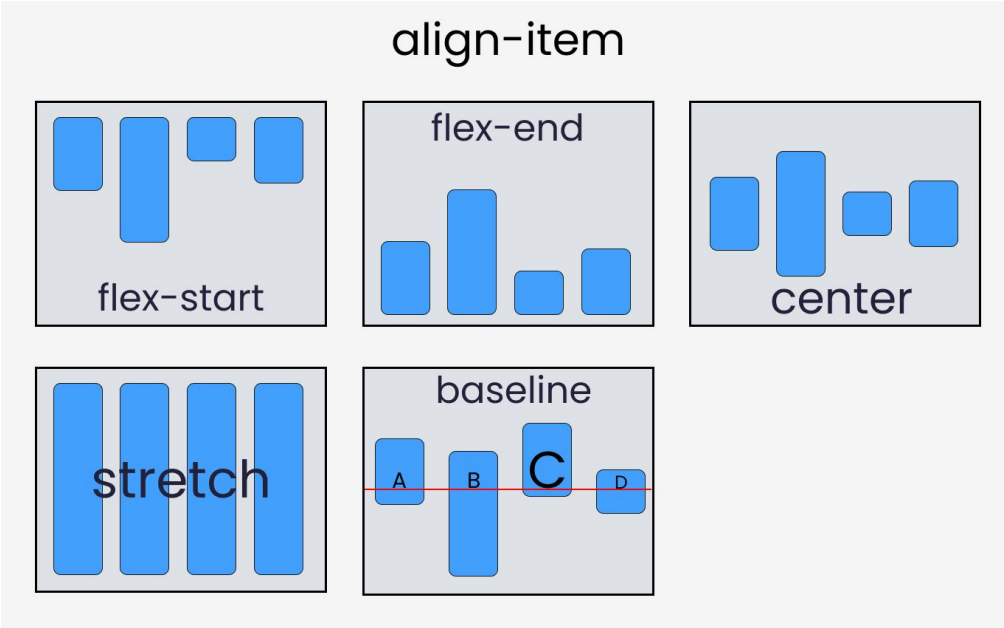
Alinear contenido:

- **align-content:** actúa en el **eje secundario**



CSS Flex: align-item

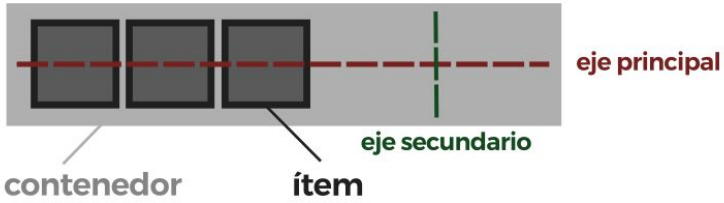




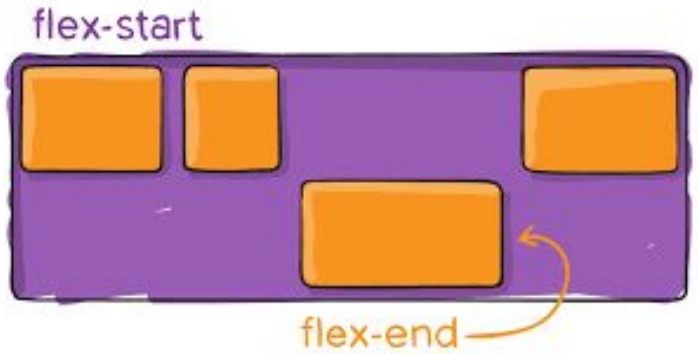
Alinear contenido:

- **align-item:** actúa en el **eje secundario**

CSS Flex: align-self



Se utiliza sobre un ítem hijo específico y no sobre el elemento contenedor. Salvo por este detalle, funciona exactamente igual que align-items.

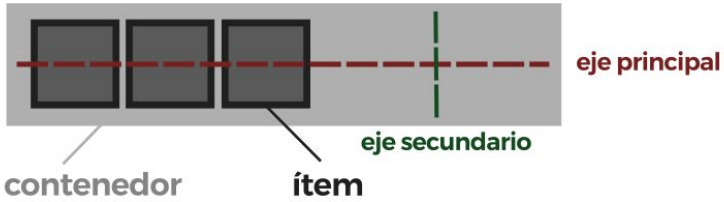


CSS Flex

Atajo: se pueden establecer los valores de **align-content** y de **justify-content** de una sola vez, denominada **place-content**

```
.container {
  display: flex;
  place-content: flex-start flex-end;
  /* Equivalente a... */
  align-content: flex-start;
  justify-content: flex-end;
}
```

CSS Flex:



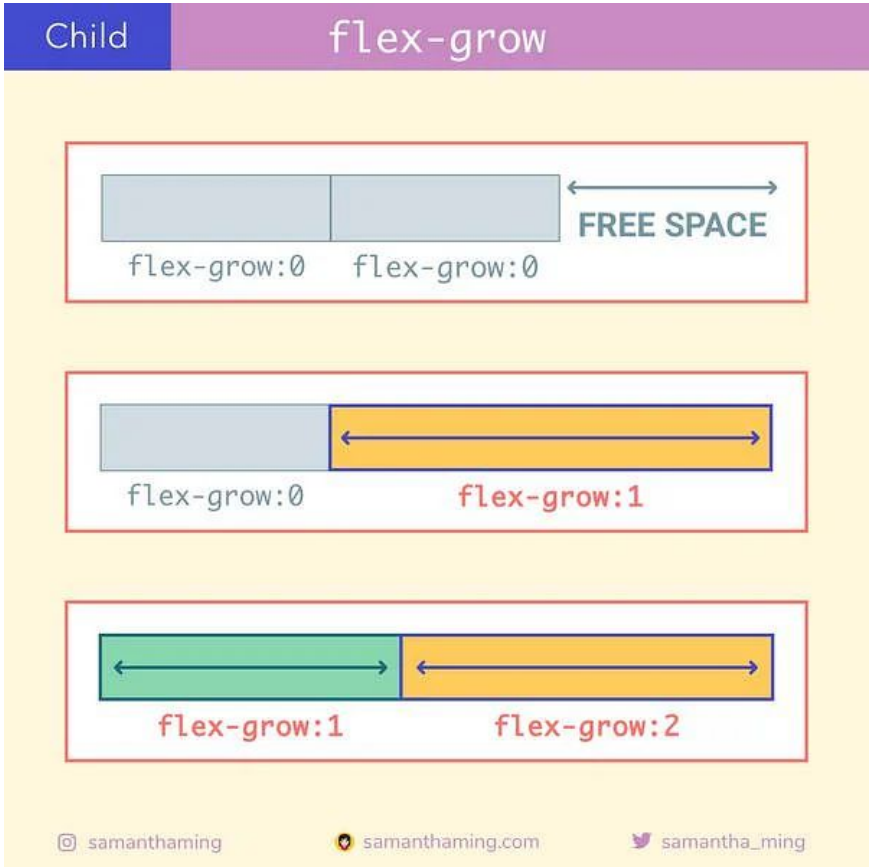
Propiedades de los hijos:

- **flex-grow:** factor de crecimiento del ítem respecto al resto.
- **flex-shrink:** factor de decrecimiento del ítem respecto al resto.
- **flex-basis:** Tamaño base de los ítems antes de aplicar variación.
- **order:** orden de aparición de los ítems.

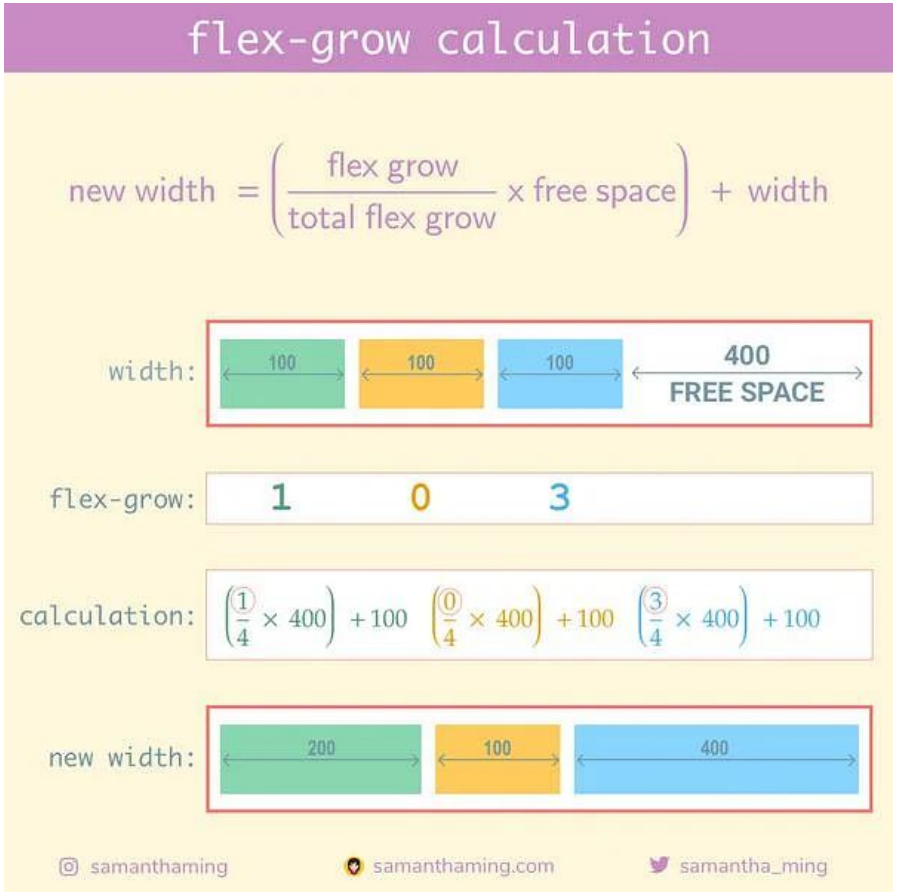
CSS Flex:

Propiedades de los hijos:

- **flex-grow:** factor de crecimiento del ítem respecto al resto.



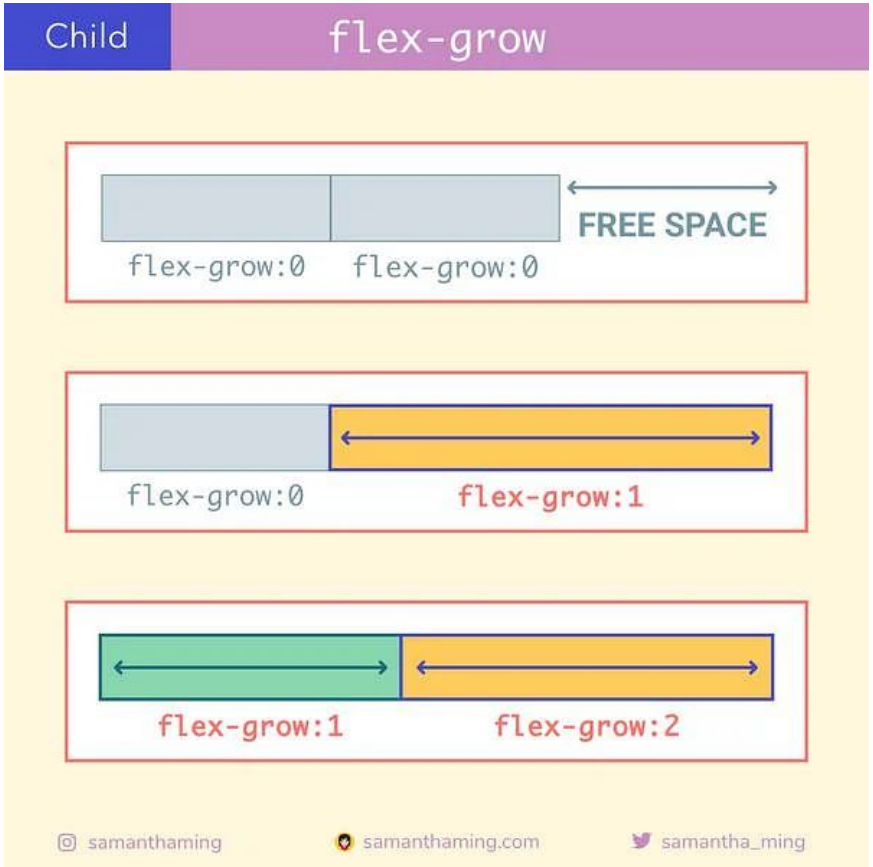
CSS Flex: flex-grow



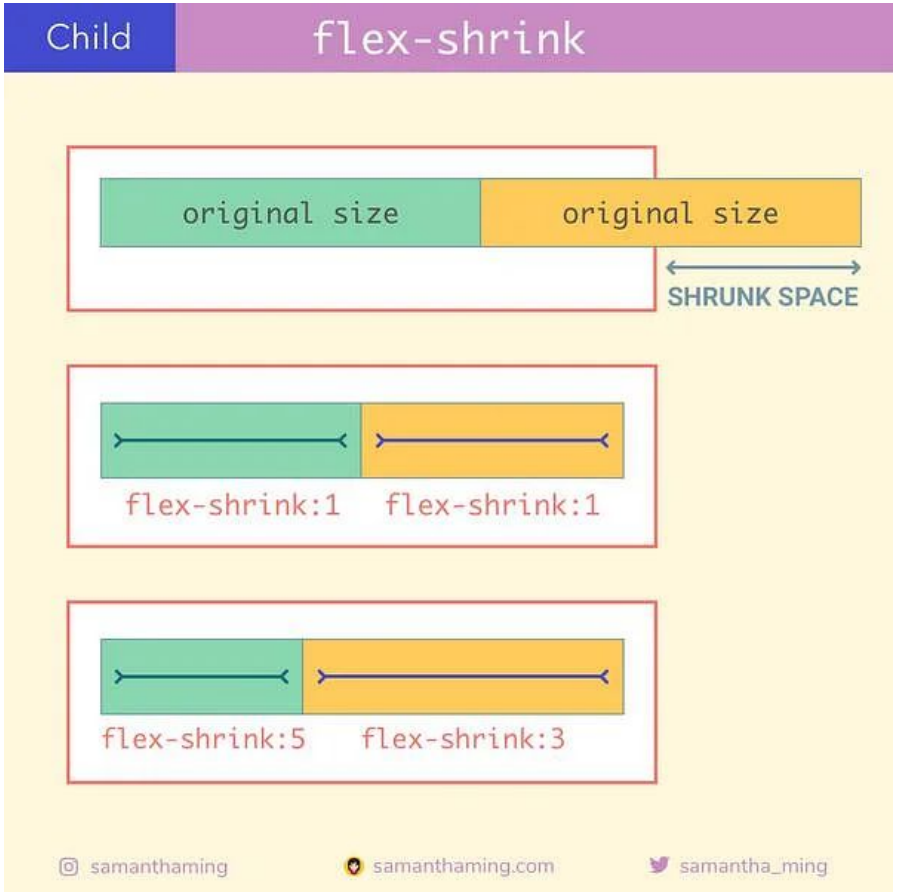
CSS Flex:

Propiedades de los hijos:

- **flex-shrink:** factor de decrecimiento del ítem respecto al resto.

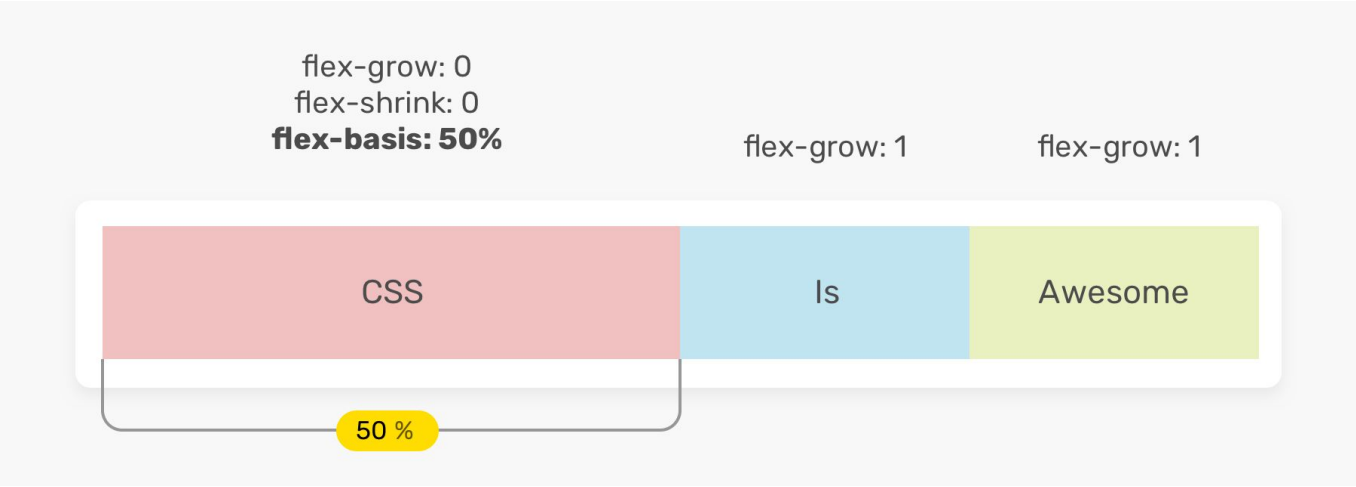


CSS Flex: flex-shrink



CSS Flex:

- **flex-basis:** tamaño por defecto (de base) que tendrán los ítems antes de aplicarle la distribución de espacio.



CSS Flex

Atajo: se pueden establecer los valores de **flex-grow, flex-shrink** y **flex-basis** de una sola vez, con **flex**

```
.item {
  /* flex: <flex-grow> <flex-shrink> <flex-basis> */
  flex: 1 3 35%;
}
```

CSS Flex

Order: un valor más bajo indica preferencia sobre otro elemento con valor más alto. Por defecto es 0.

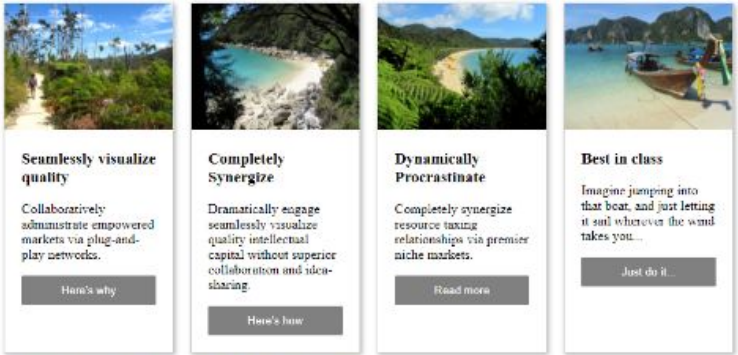

```
.box {
  display: flex;
  flex-direction: row;
}
.box :nth-child(1) { order: 2; }
.box :nth-child(2) { order: 3; }
.box :nth-child(3) { order: 1; }
.box :nth-child(4) { order: 3; }
.box :nth-child(5) { order: 1; }
```

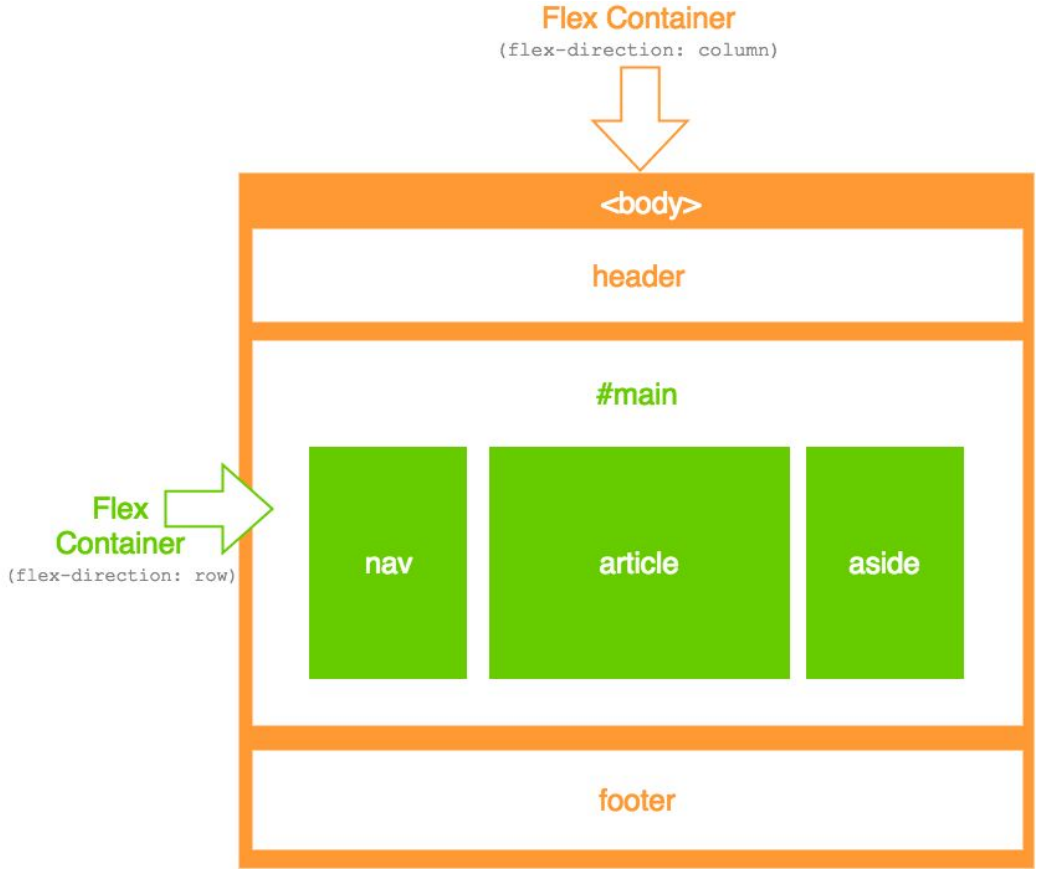
CSS Flex: flex in flex


```
<div class="container">
        <div class="red">1
                <div class="green">1a</div>
                <div class="blue">1b</div>
        </div>
        <div class="green">2</div>
        <div class="blue">3</div>
</div>
```

```
.container {
  display: flex;
  }
.red {
  background: red;
  display: flex;
  flex-direction: column;
}
```

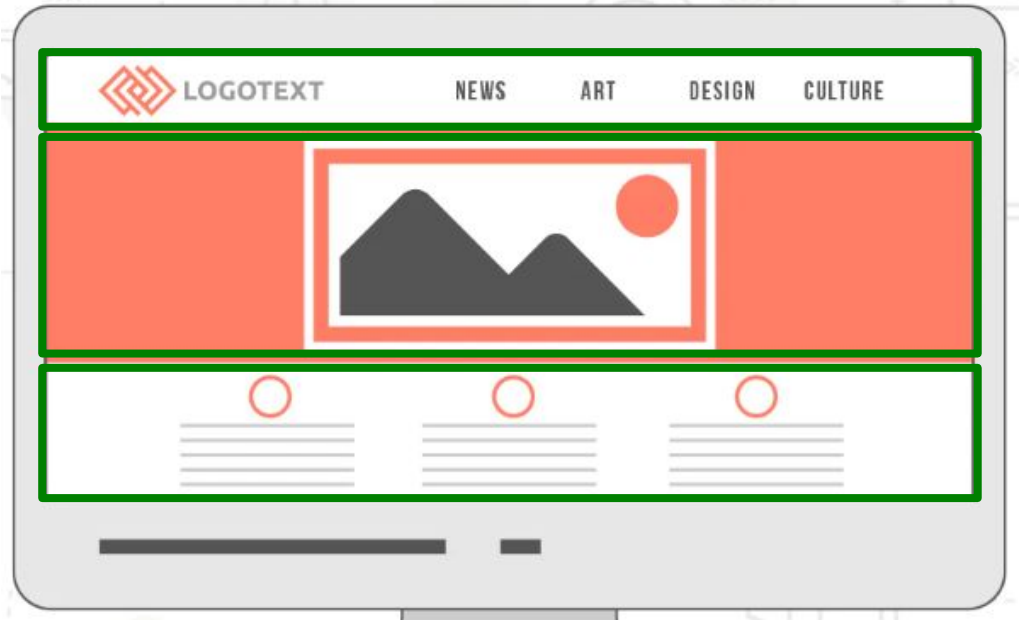
CSS Flex: flex in flex



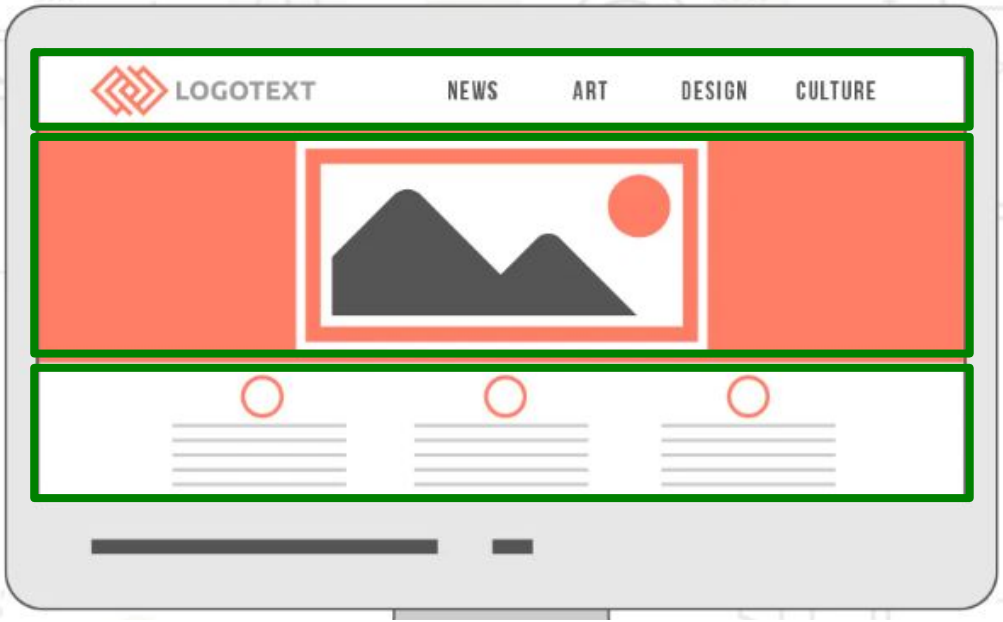


CSS Flex Layout

CSS Flex Layout



CSS Flex Layout




```
<body>
1   <div></>
2   <div></>
3   <div></>
</body>
```

CSS Flex Layout


```
<body>
1   <header></>
2   <carousel></>
3   <content></>
</body>
```

CSS Flex Layout


```
<body>
1    <header>


 </>
</body>
```

CSS Flex Layout


```
<body>
  1  <header></>😁
  2  <carousel></>
  3  <content></>
</body>
```

CSS Flex Layout


```
<body>
1   <header></>😁
2   <carousel></>😁
3   <content></>
</body>
```

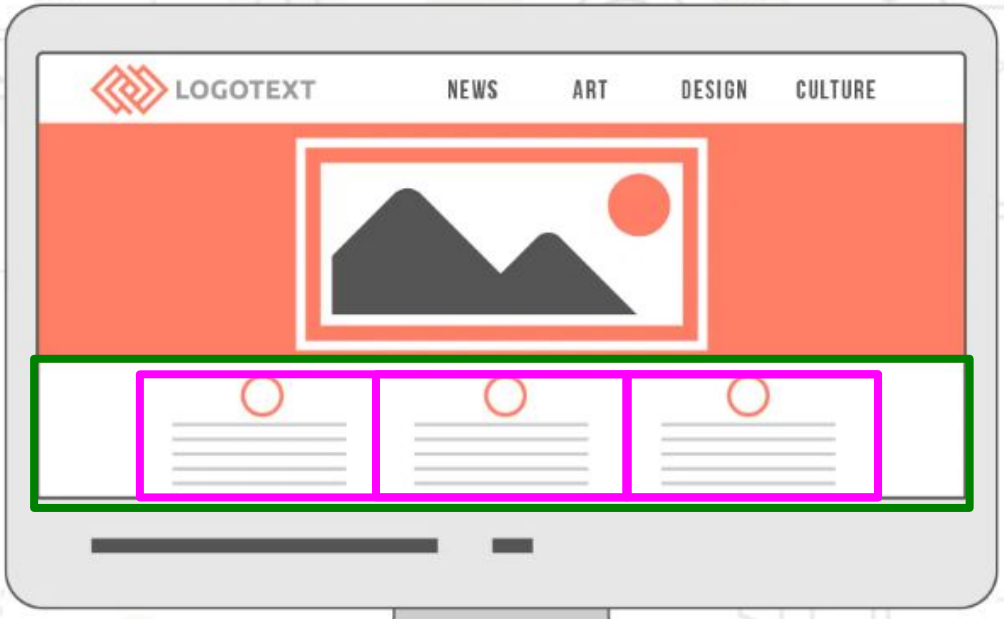
CSS Flex Layout


```
<body>
1   <header></>😁
2   <carousel></>😁
3   <content></>🙄
</body>
```

CSS Flex Layout

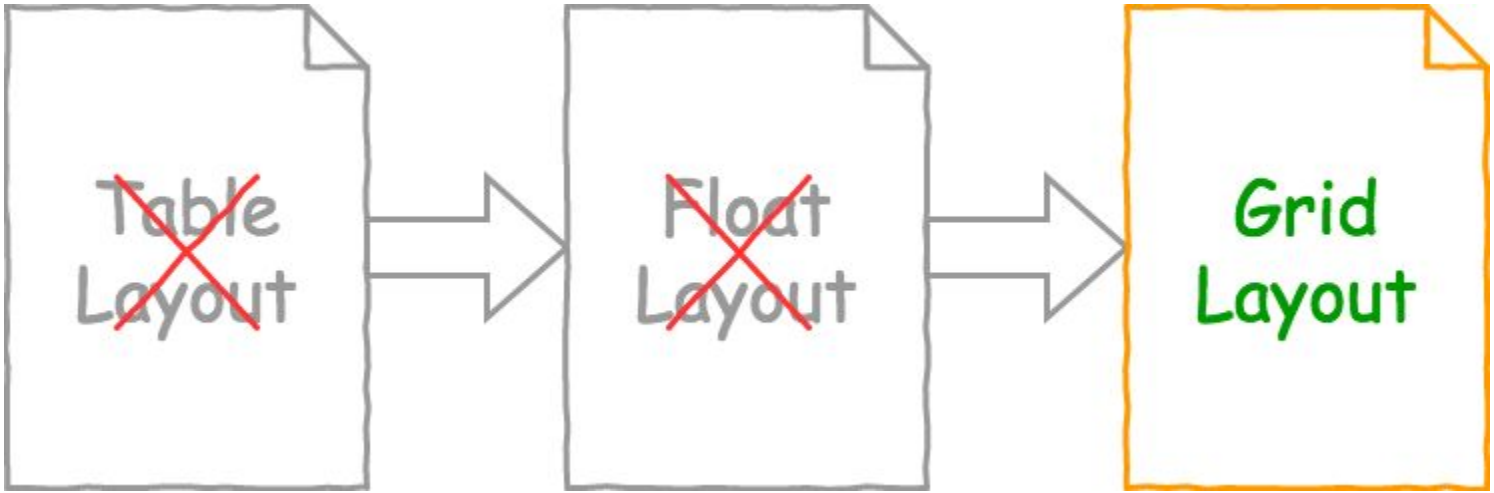

```
<body>
  <content>

  </content>
</body>
```

CSS Flex Layout

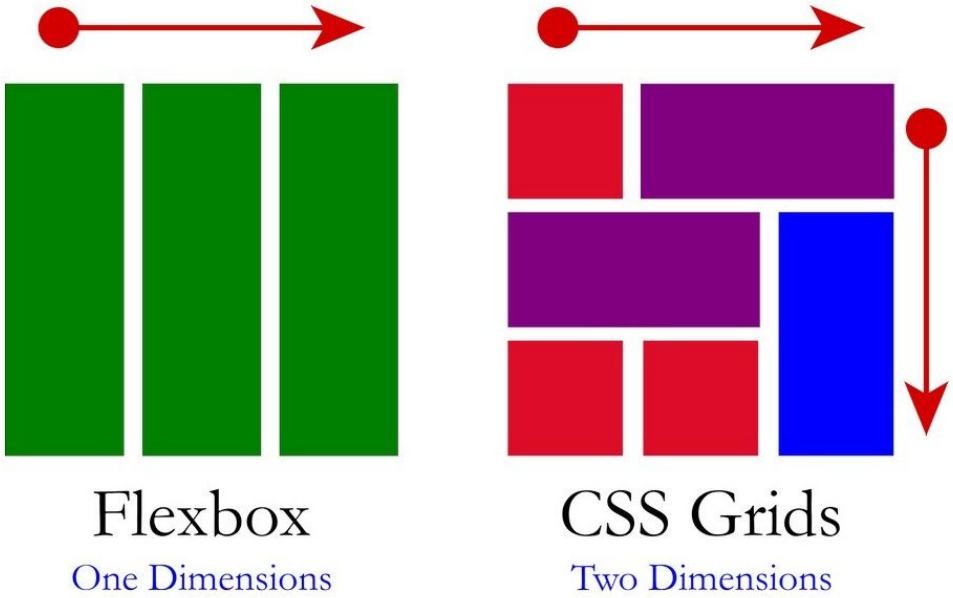



```
<body>
  <content>
    <div></>
    <div></>
    <div></>

  </content>
</body>
```

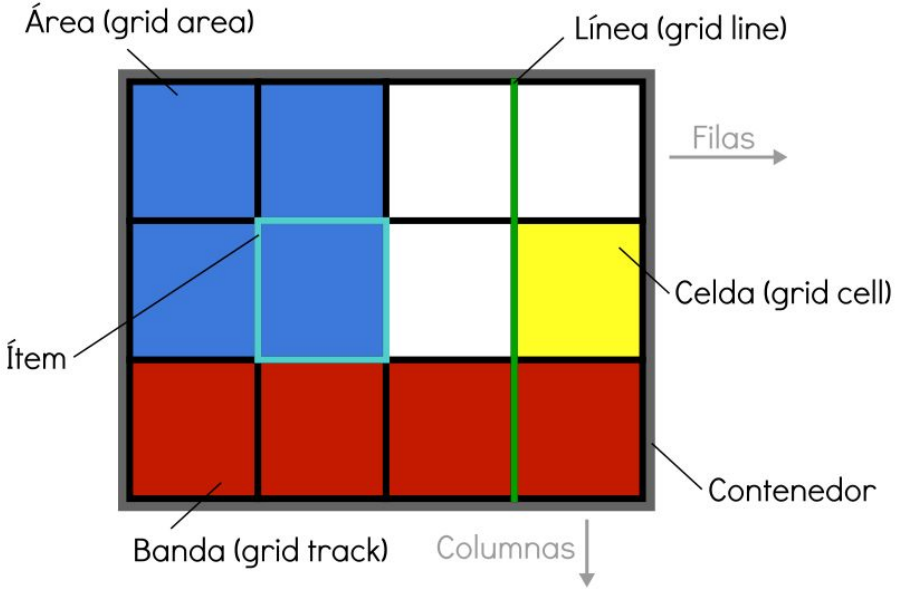
CSS Grid



CSS Grid



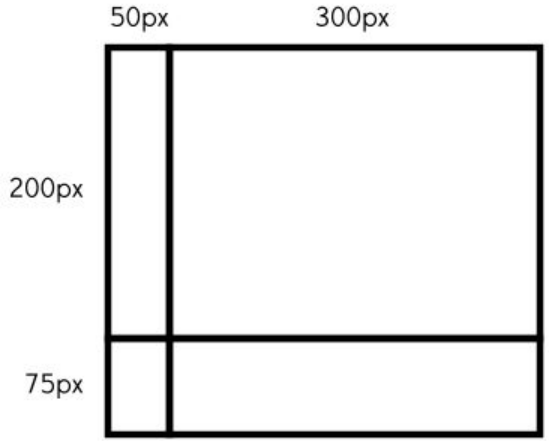
CSS Grid


Área (grid area)
Línea (grid line)
Filas
Celda (grid cell)
Ítem
Contenedor
Banda (grid track)
Columnas

CSS Grid

```
<div class="grid"> <!-- contenedor -->
  <div class="a">Item 1</div>
  <div class="b">Item 2</div>
  <div class="c">Item 3</div>
  <div class="d">Item 4</div>
 </div>
```

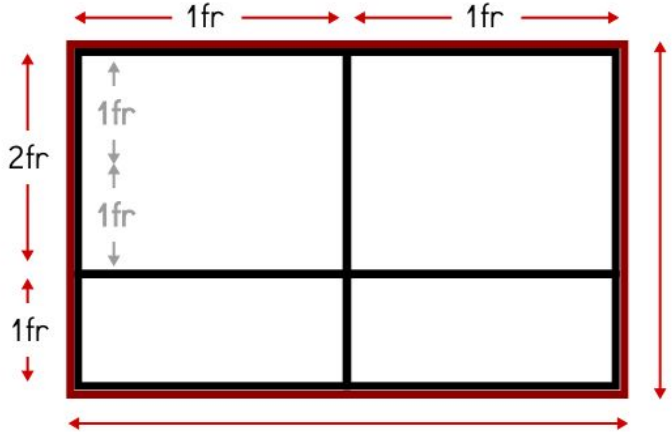
CSS Grid




```
.grid {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
}
```

CSS Grid: fr

fr = fracción de espacio restante en el grid




```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```

CSS Grid

```
.grid {
  display: grid;
  grid-template-columns: 100px 50px 50px 200px;
  grid-template-rows: 50px 100px 50px 100px;
}
```

CSS Grid: repeat()

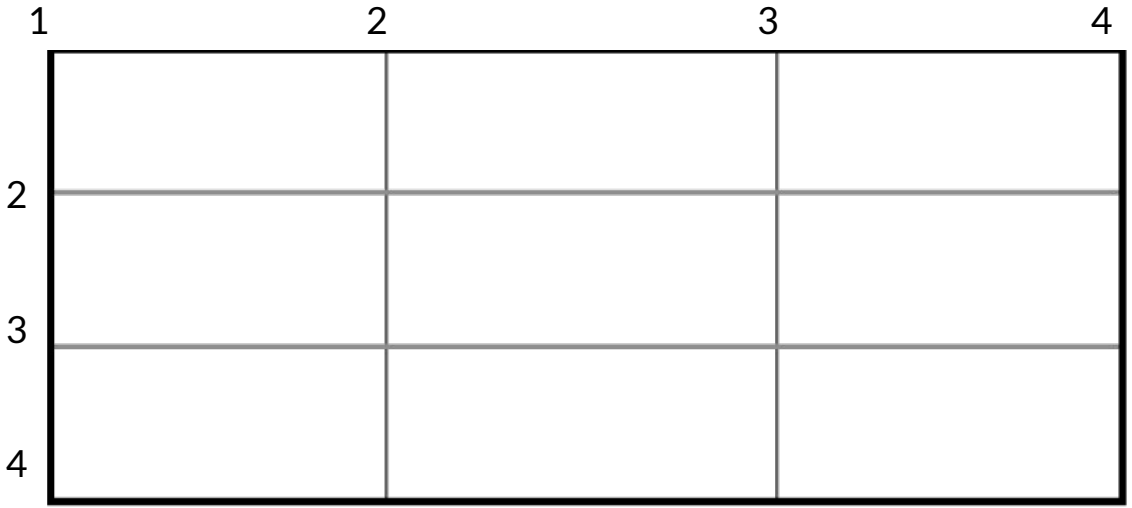
repeat(núm_de_veces, valor_o_valores **)**

```
.grid {
  display: grid;
  grid-template-columns: 100px repeat(2, 50px) 200px;
  grid-template-rows: repeat(2, 50px 100px);
}
.grid {
  display: grid;
  grid-template-columns: 100px 50px 50px 200px;
  grid-template-rows: 50px 100px 50px 100px;
}
```

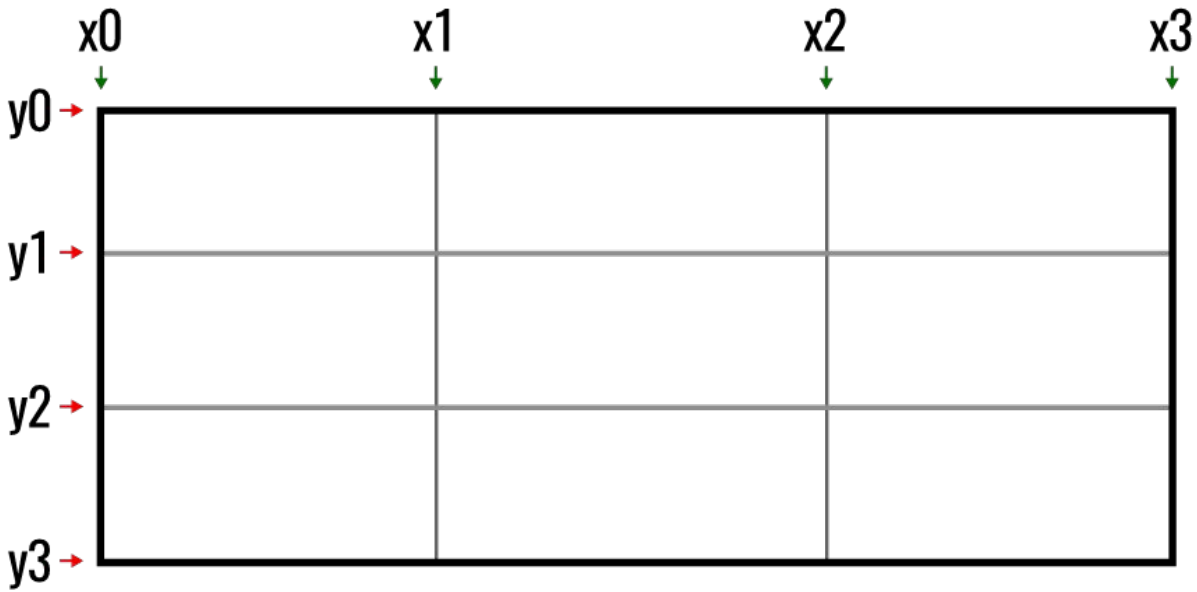
CSS Grid: linenames

```
<div class="grid">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="content">Content</div>
  <div class="footer">Footer</div>
</div>
```

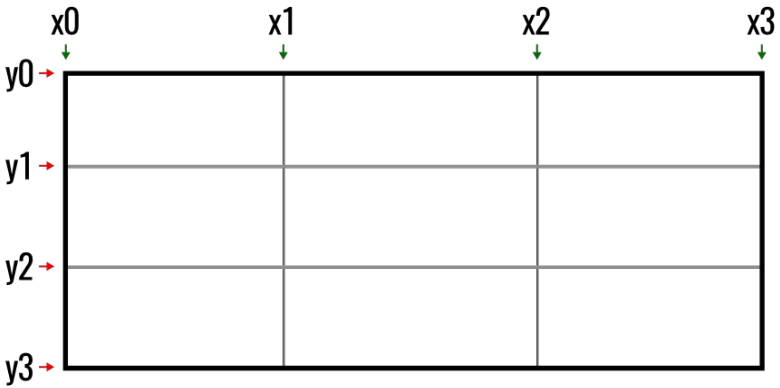
CSS Grid: line numbers by default



CSS Grid: linenames

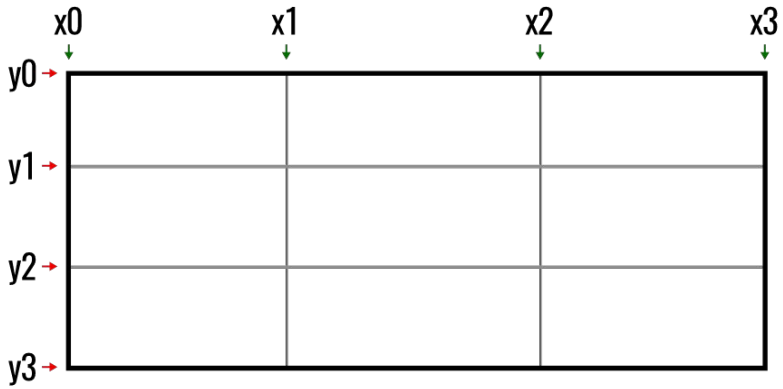


CSS Grid: linenames




```
.grid {
  display: grid;
  grid-template-columns: [x0] 1fr [x1] 1fr [x2] 1fr [x3];
  grid-template-rows: [y0] 1fr [y1] 1fr [y2] 1fr [y3];
 }
```

CSS Grid: linenames



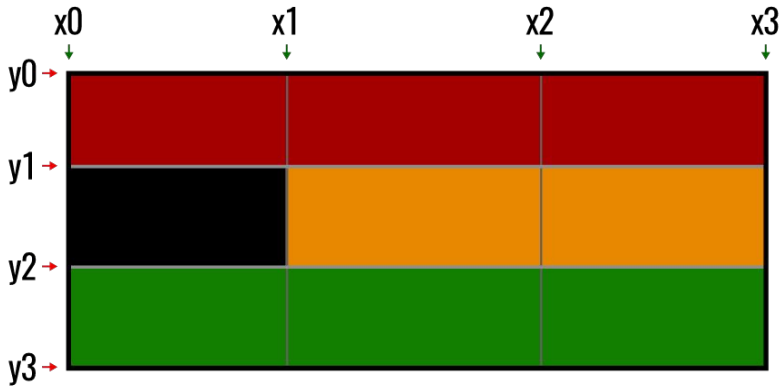

```
.header {
  grid-column-start: x0;
  grid-column-end: x3;
 }/*grid-column: x0 / x3;*/
```

```
.content {
  grid-column: x1 / x3;
  grid-row: y1 / y3;
 }
```

```
.footer {
  grid-column: x0 / x3;
  grid-row: y2;
 }
```

```
.sidebar {
  grid-row: y1 / y2;
}
```

CSS Grid: linenames




```
.header {
  grid-column-start: x0;
  grid-column-end: x3;
 }/*grid-column: x0 / x3;*/
```

```
.content {
  grid-column: x1 / x3;
  grid-row: y1 / y3;
 }
```

```
.footer {
  grid-column: x0 / x3;
  grid-row: y2;
 }
```

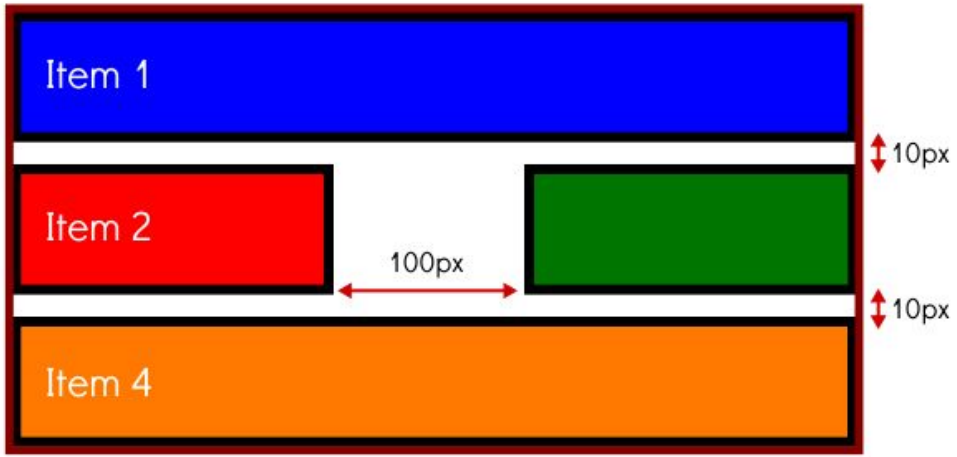
```
.sidebar {
  grid-row: y1 / y2;
}
```

CSS Grid: grid-area

```
.grid {
display: grid;
grid-template-areas:
   "head head"
   "menu main"
   "foot foot";
}
```

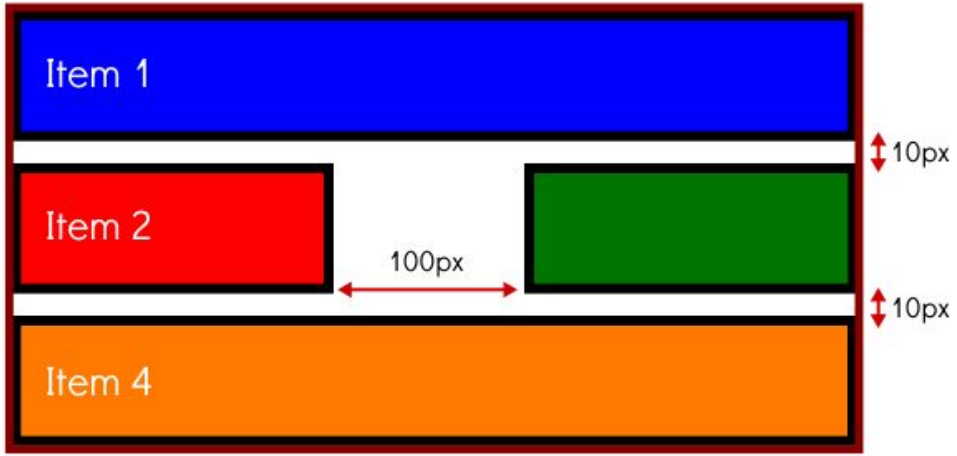
CSS Grid: gap

```
.grid {
   column-gap: 100px;
   row-gap: 10px;
}
```



CSS Grid: gap

```
.grid {
   column-gap: 100px;
   row-gap: 10px;
}
```



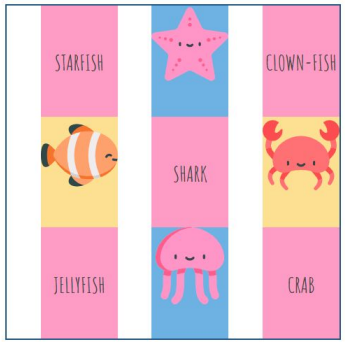

```
.grid {
/* gap: <row-gap> <column-gap> */
   gap: 10px 100px;
}
```

CSS Grid: justify items

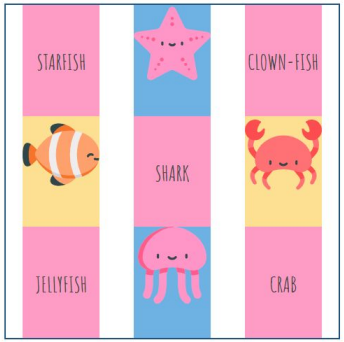
```
.container {

  justify-items:

        start | end | center | stretch;

}
```

START

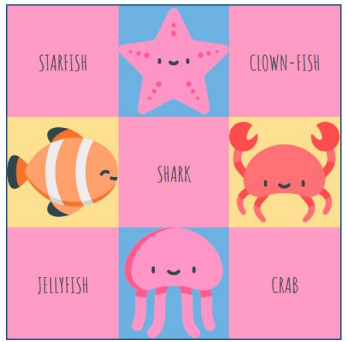
END



CENTER



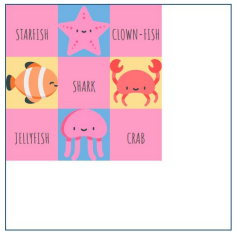
STRETCH



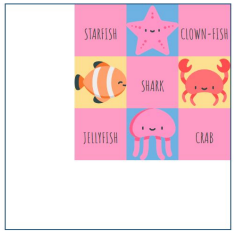
CSS Grid: justify contents

```
.container {
  justify-content:
          start | end | center |
          stretch | space-around |
          space-between | space-evenly;
 }
```

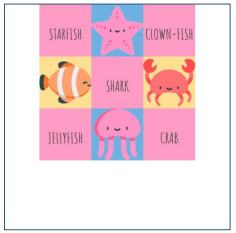
START



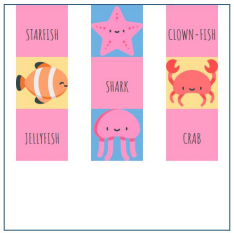
END



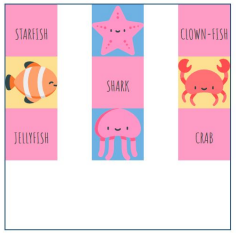
CENTER



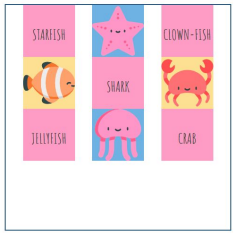
SPACE-AROUND



SPACE-BETWEEN

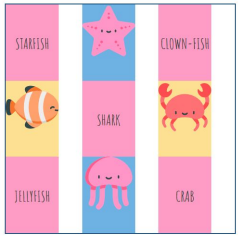


SPACE-EVENLY

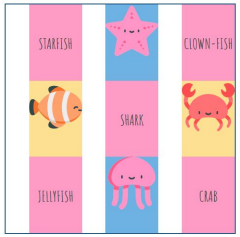


CSS Grid: items vs contents

START

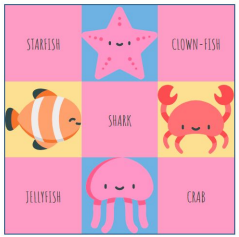


END

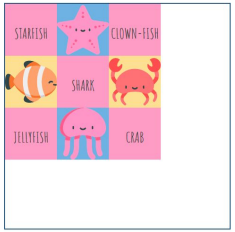


CENTER

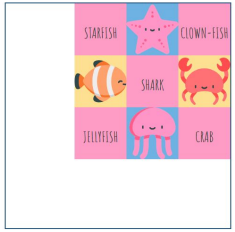
STRETCH



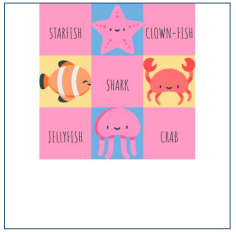
START



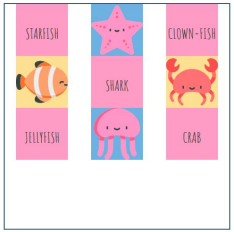
END



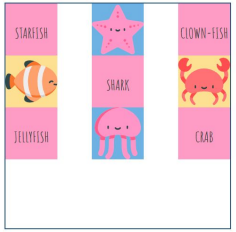
CENTER



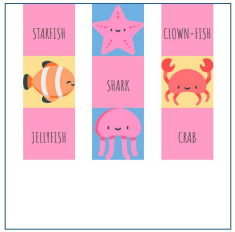
SPACE-AROUND



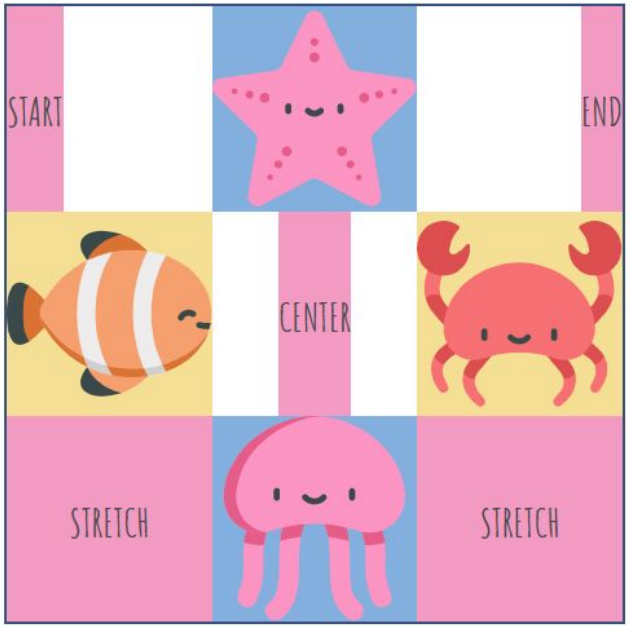
SPACE-BETWEEN



SPACE-EVENLY



CSS Grid: justify-self

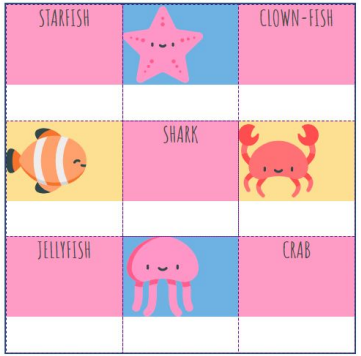



```
.item {
  justify-self:
       start | end |
       center | stretch;
 }
```

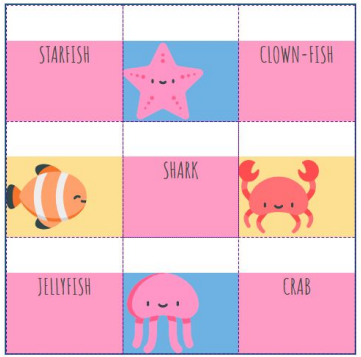
CSS Grid: align-items

```
.container {

  align-items:

        start | end | center | stretch;

}
```

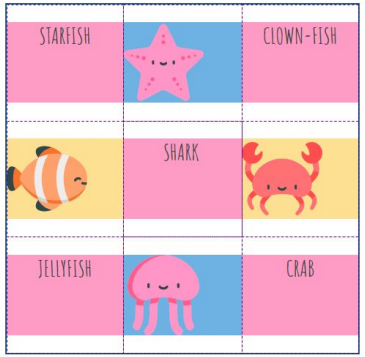
START



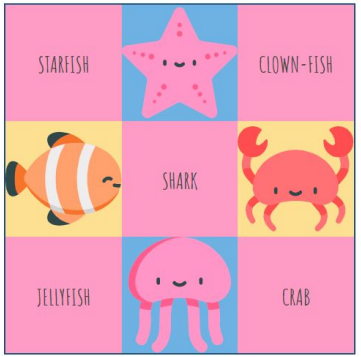
END



CENTER



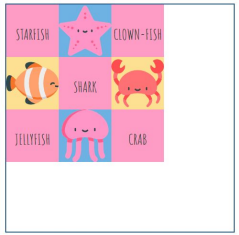
STRETCH



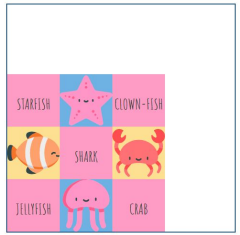
CSS Grid: align-content

```
.container {

  align-items:

        start | end | center | stretch;

}
```

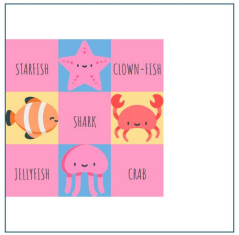
START



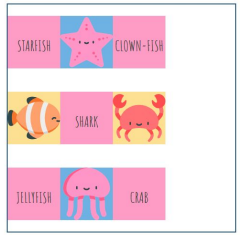
END



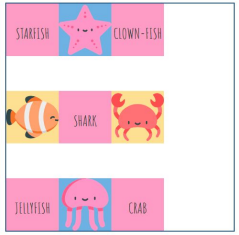
CENTER



SPACE-AROUND

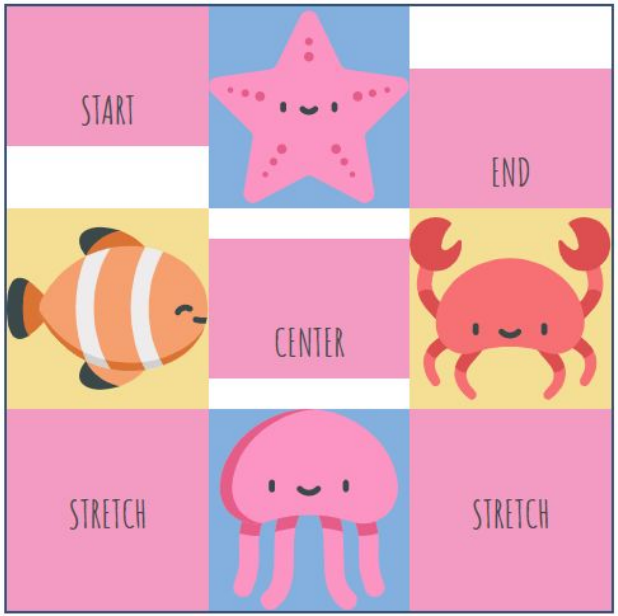


SPACE-BETWEEN



SPACE-EVENLY

CSS Grid: align-self




```
.item {
  align-self:
       start | end |
       center | stretch;
 }
```

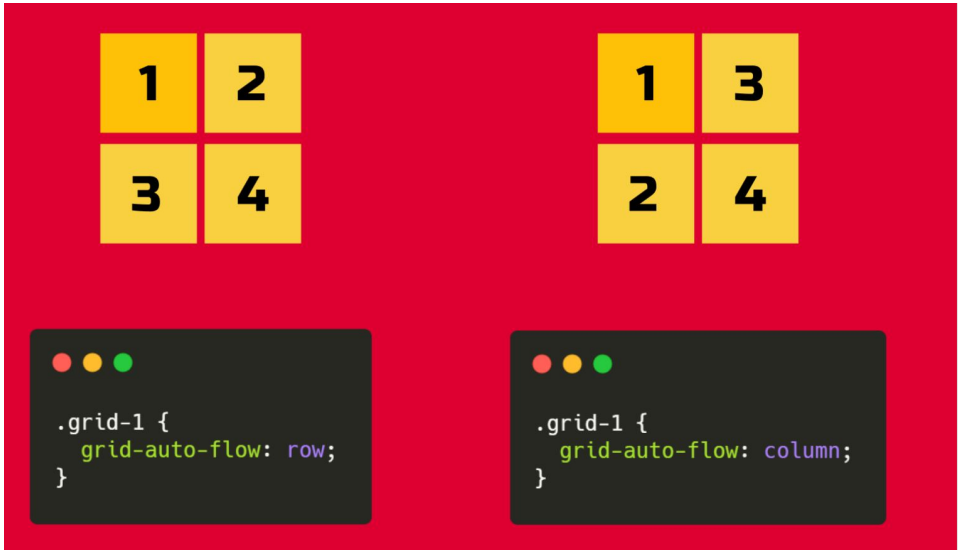
CSS Grid: place-content & place-items

```
place-content: <'align-content'> <'justify-content'>
```

```
place-items: <'align-items'> <'justify-items'>
```

place-items - CSS: Cascading Style Sheets | MDN (mozilla.org)

CSS Grid: grid-auto-flow



grid-auto-flow - CSS: Cascading Style Sheets | MDN (mozilla.org)

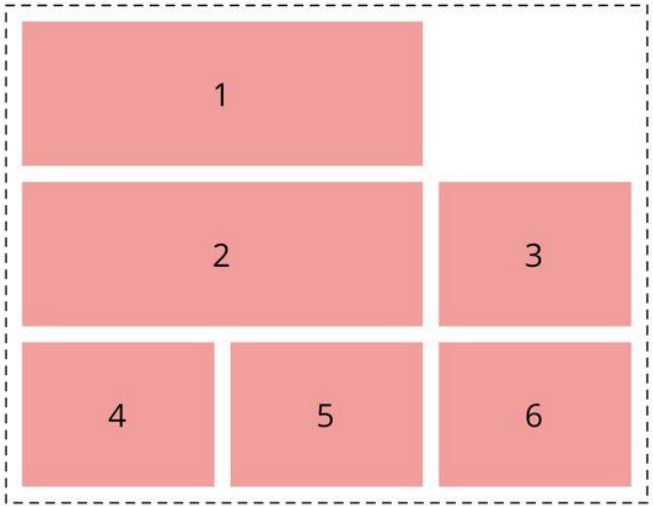
CSS Grid: grid-auto-flow

```
/* Keyword values */
grid-auto-flow: row;
grid-auto-flow: column;
grid-auto-flow: dense;
grid-auto-flow: row dense;
grid-auto-flow: column dense;
```

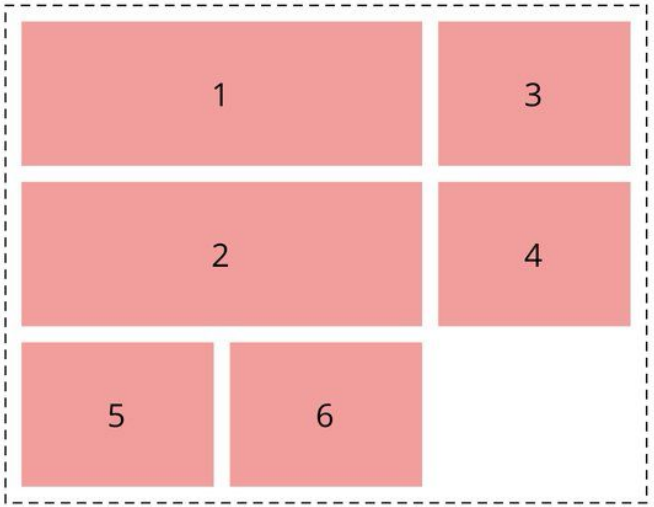
grid-auto-flow - CSS: Cascading Style Sheets | MDN (mozilla.org)

CSS Grid: grid-auto-flow

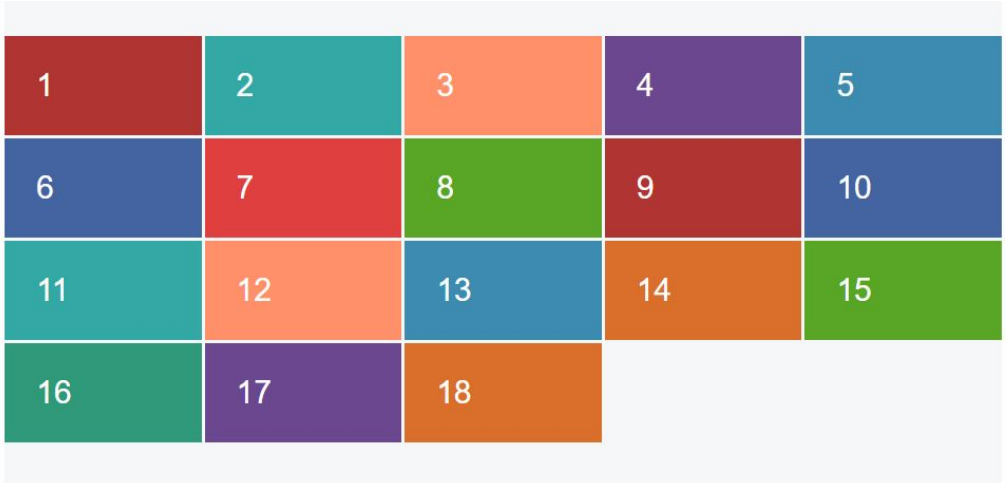
`grid-auto-flow: row`



`grid-auto-flow: row dense`



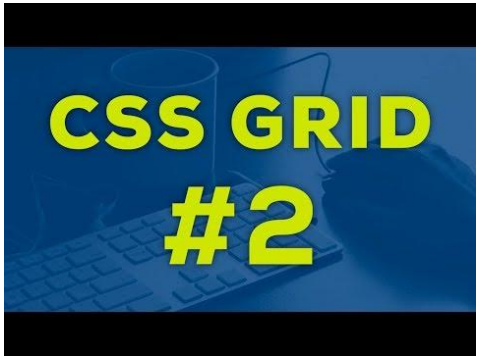
CSS Grid: colocación automática en Grid



Comprendiendo el "Algoritmo de Colocación Automática" de CSS Grid (tutsplus.com)

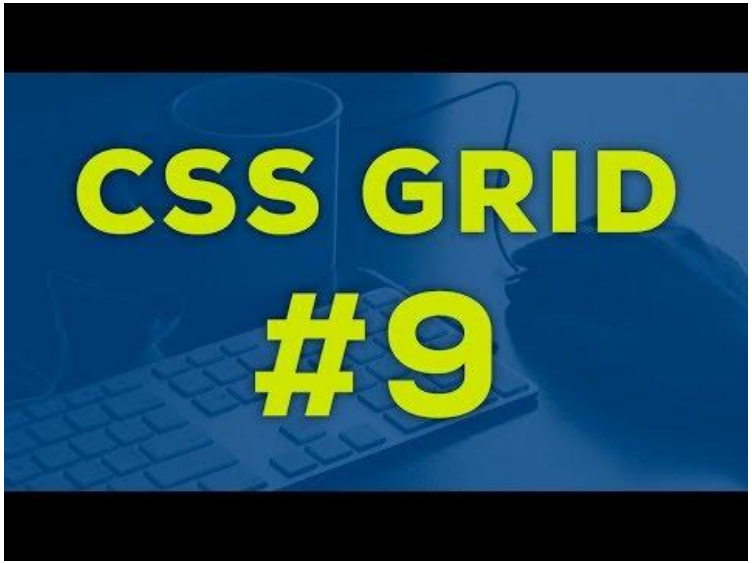
CSS Grid: minmax()

Se utiliza dentro de **grid-template** , permite definir un rango dinámico de valores.



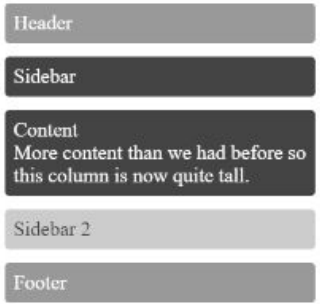
Curso de CSS GRID: 2.- Medidas Mínimas y Máximas - YouTube

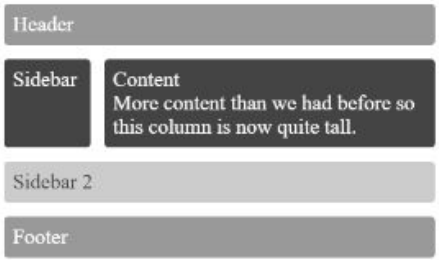
CSS Grid: autofill & autofit

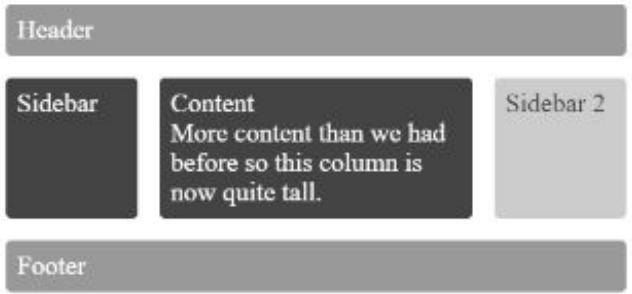


Curso de CSS GRID: 9.- Auto Fill y Auto Fit - YouTube

CSS Grid & Media Queries







[Grid by Example 13: Redefining grid areas with media queries (codepen.io)]

[Using media queries and grid CSS to reflow columns (w3.org)]