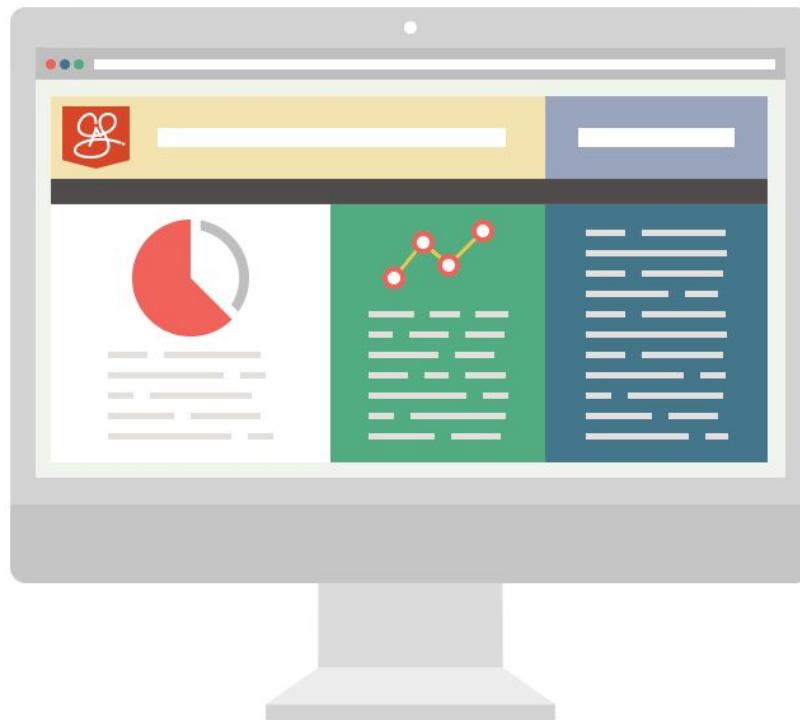


Diseño Responsive

¿Qué es el diseño responsive?

Es una técnica de diseño web que busca la correcta visualización de una **misma página** en distintos dispositivos.



Implica:

- cambio de dimensiones
- distribución dinámica de los elementos

Responsive vs Adaptive

Responsive (responsivo)

- El mismo sitio para todos los dispositivos
- + Mejora la experiencia del usuario
- + Menor costo de desarrollo
- + Menor costo de mantenimiento

Adaptive (adaptado)

- Diferentes layouts para diferentes dispositivos
- Por lo general cambian las URLs
- + Personaliza también el contenido
- + Permite minimizar datos

Las ventajas de uno son desventajas del otro

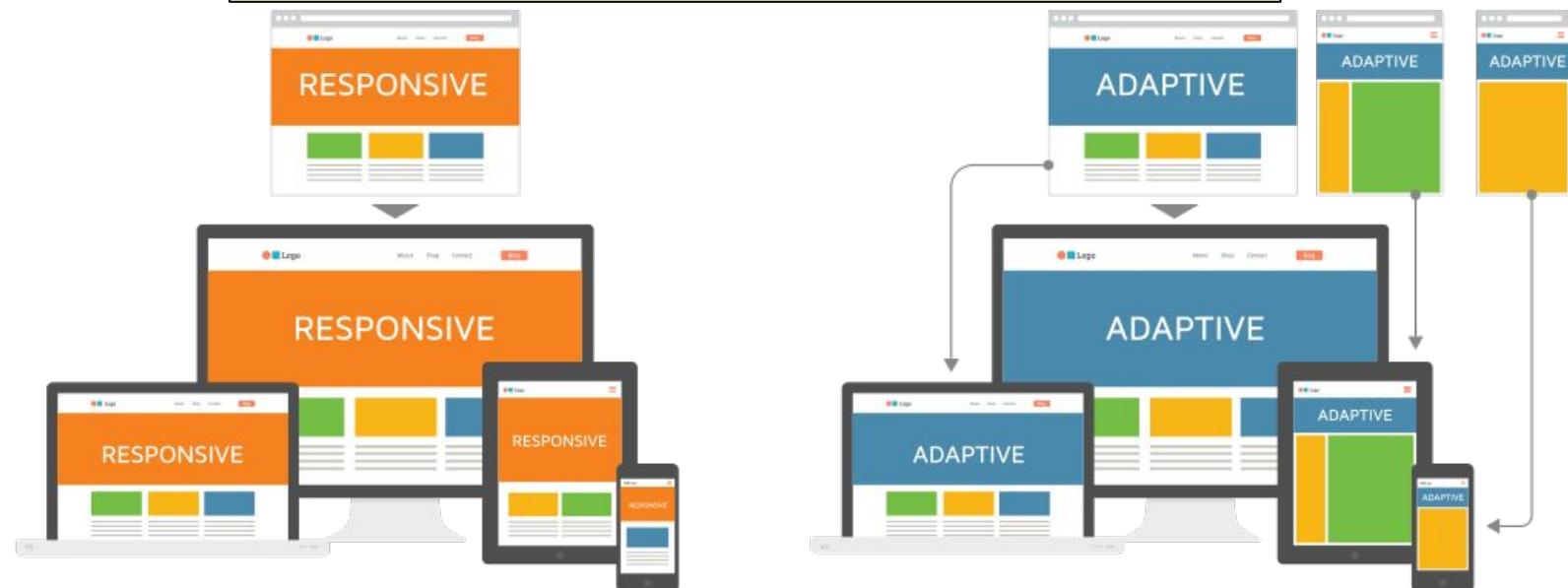


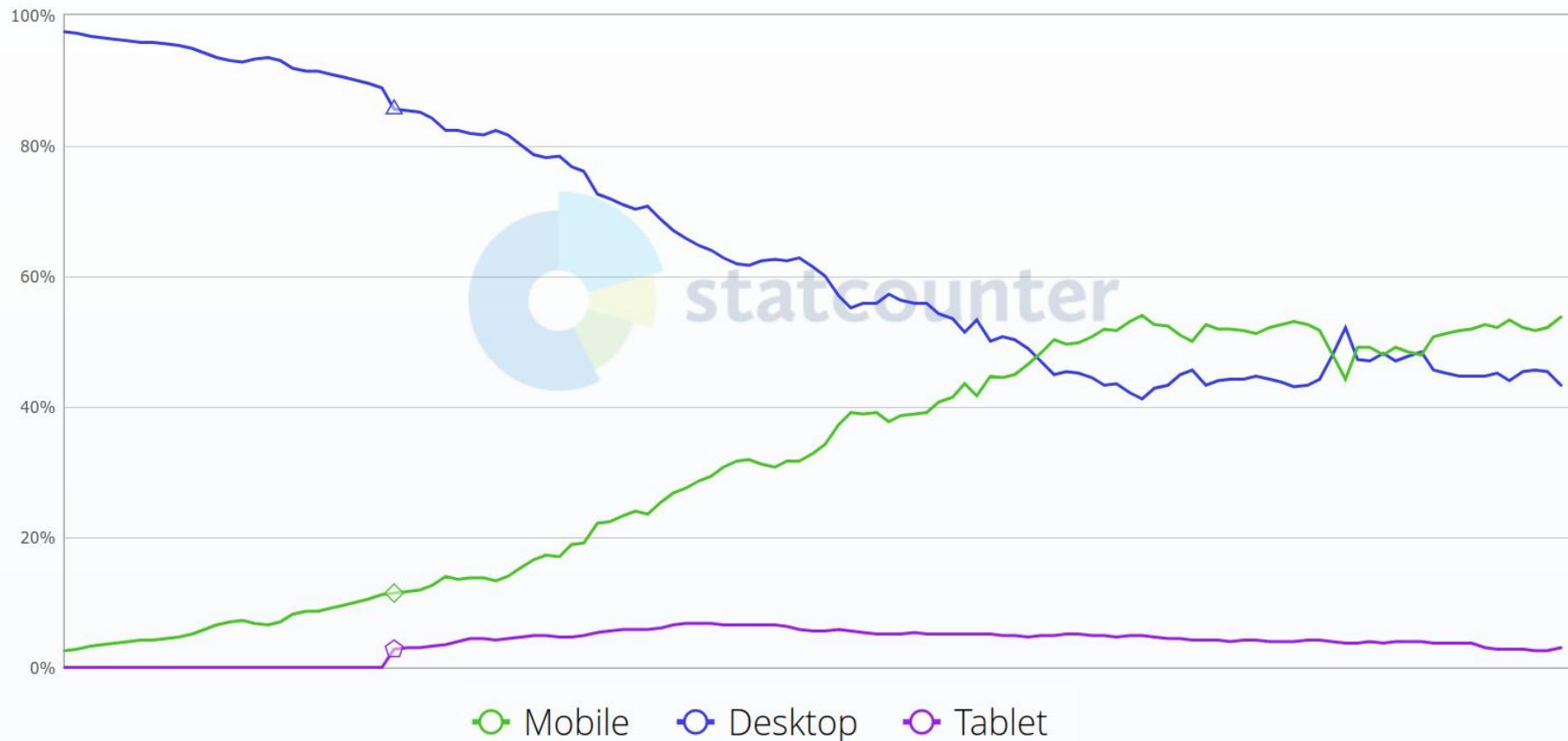
Imagen: <https://uxplanet.org/adaptive-vs-responsive-web-design-eead0c2c28a8>

Por qué?

Evolución de 10 años: Desktop vs Tablet Vs Mobile

Desktop vs Mobile vs Tablet Market Share Worldwide

June 2010 - May 2020



Media Queries

Media Queries ¿Qué son?

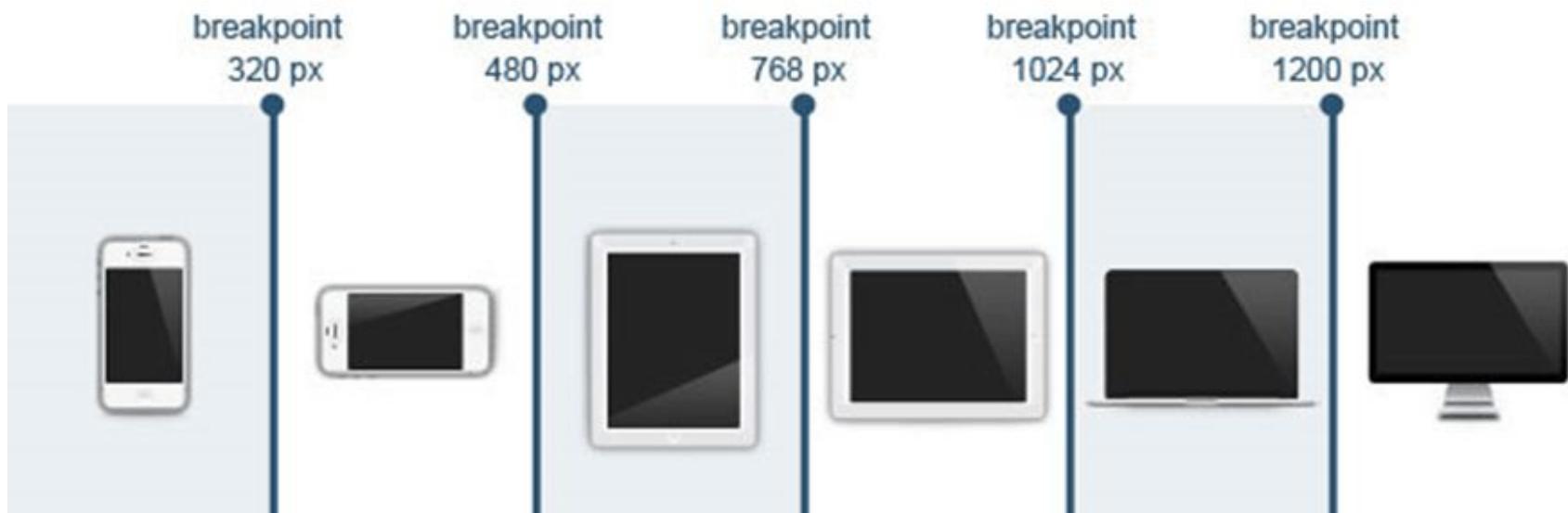
- Es un recurso de CSS que permite asignar diferentes estilos para **distintos tamaños y resoluciones de pantalla**
- Nos da la posibilidad de entregar distintas apariencias para diferentes dispositivos
- Es la **base del diseño responsive**

Actúan como un IF de CSS



Media Queries ¿Qué son?

Se utilizan **breakpoints** para indicar que ciertas partes del diseño se comportan diferentes bajo ciertas condiciones



Media Queries ¿Cómo se usan?

min-width

```
@media only screen and (min-width: 600px) { ... }
```

Si el **ancho del dispositivo** es **MAYOR o igual que 600px** entonces
{aplicá estas reglas}

```
@media only screen and (min-width: 600px) /* Umbral de cambio */  
body /* elemento de cambio */  
  
background-color: pink; /* propiedad que cambia */  
  
}  
}
```

DEMO

Live: <https://codepen.io/webUnicen/pen/ybpZBo>

MOBILE
FIRST

Media Queries ¿Cómo se usan?

max-width

```
@media only screen and (max-width: 600px) { ... }
```

DESKTOP
FIRST

Si el **ancho del dispositivo** es **MENOR o igual que 600px**
entonces {aplicá estas reglas}

DEMO

<https://codepen.io/webUnicen/pen/ybpZBo>

@media print

```
@media only print { ... }
```

PRINTER

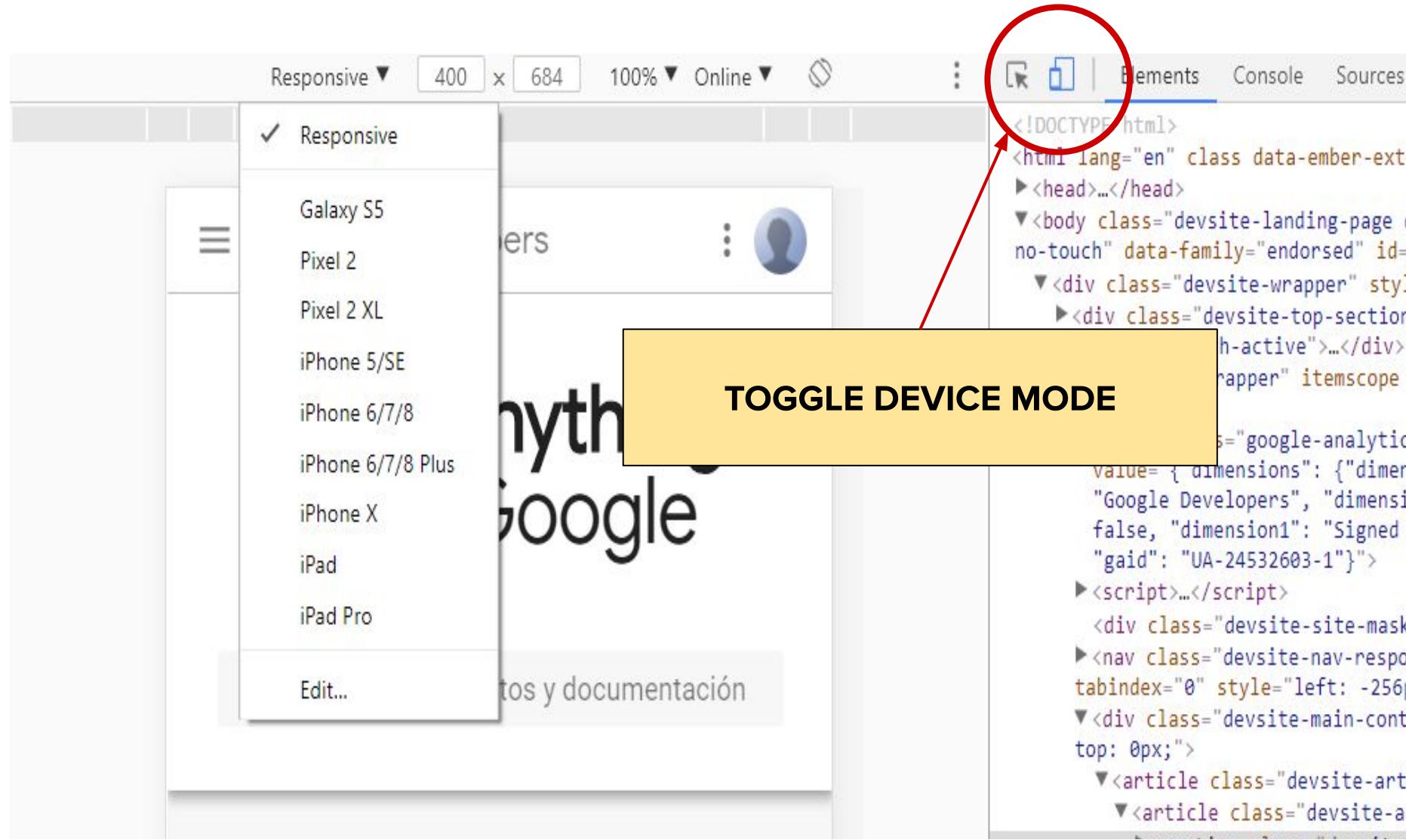
Si estás por **mandar a imprimir** {aplicá estas reglas}

ViewPort

Se necesita indicar al navegador que active el soporte de media queries:

```
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport"  
        content="width=device-width,  
        initial-scale=1">  
  ...  
</head>
```

Dev Tools



Mobile First

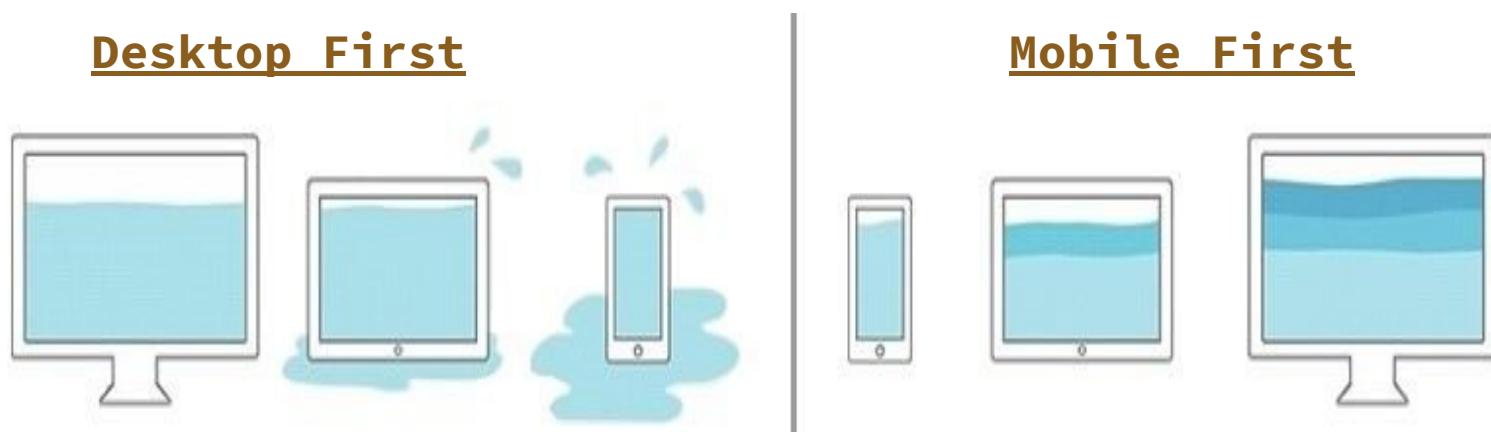
Mobile First vs Desktop First

“El contenido web es como el agua”

**Toma muchas formas y fluye dentro de
todos los diferentes contenedores**

Mobile First

1. Desarrollar el diseño móvil primero
 - Programas una pantalla pequeña con menos elementos
2. Luego ajustarlo a las grandes pantallas
 - Agregas otros elementos a medida que hay más resolución



Mobile First

Más sencillo de hacer, obliga a pensar
qué es lo “**más importante**”



En celular lo
importante va primero

Y en PC capaz va al centro
y es más grande
(el CSS debe reordenar los
elementos)

SEO + Mobile First

“Google has traditionally always crawled web pages as if it were viewing them from a desktop browser. But at some point in the near future, bots will *only* look at page content as if it were being accessed from a mobile phone or tablet.”

<https://www.freelanceseoesssex.co.uk/mobile-first-index-set-to-be-the-way-forward-for-google/>



Ejemplo

MOBILE



DESKTOP



Código base para el ejemplo HTML



```
<body>
  <header>
    <h1>LOGO</h1>
  </header>

  <section class="main">
    <h2>Main Content</h2>
  </section>

  <aside class="sidebar">
    <h3>Sidebar</h3>
  </aside>

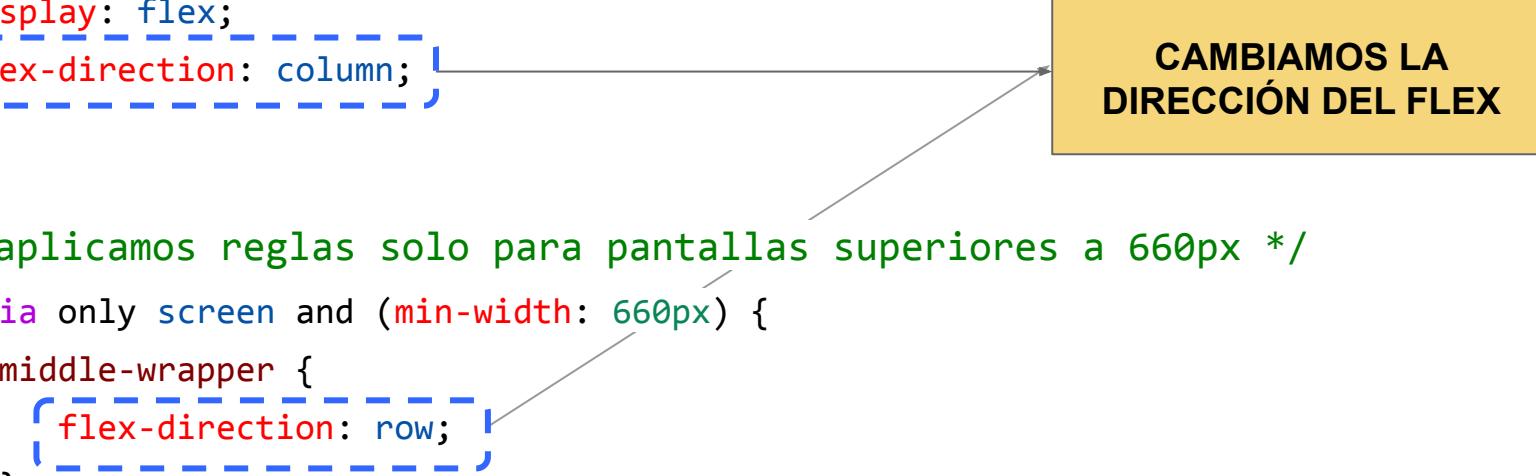
  <footer>
    <h3>Footer</h3>
    <p>Web Tudai</p>
  </footer>
</body>
```



Live: <https://codepen.io/webUnicen/pen/pV>

Posicionamiento de las secciones con Flex

```
/* agregamos un contenedor flex que envuelven las cajas centrales */  
.middle-wrapper {  
    display: flex;  
    flex-direction: column; }  
  
/* aplicamos reglas solo para pantallas superiores a 660px */  
@media only screen and (min-width: 660px) {  
    .middle-wrapper {  
        flex-direction: row; }  
  
    /* definimos anchos proporcionales a las columnas */  
.main {  
    flex-grow: 4; }  
.sidebar {  
    flex-grow: 1; }  
}  
}
```



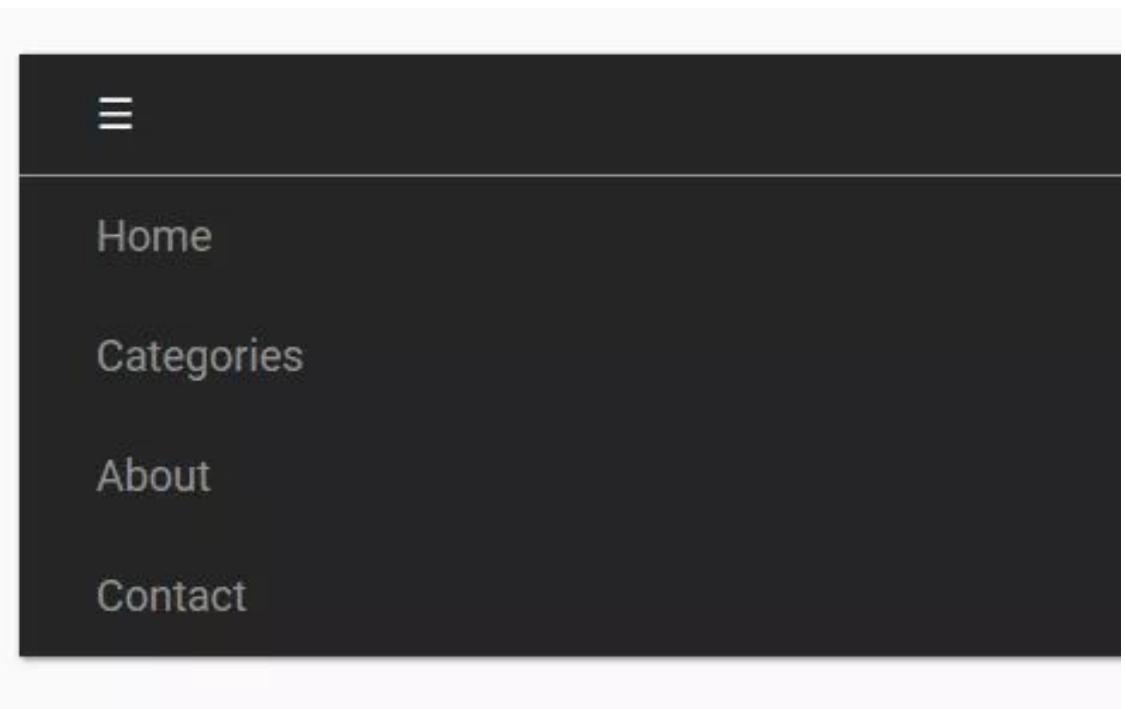
CAMBIAMOS LA
DIRECCIÓN DEL FLEX



Live: <https://codepen.io/webUnicen/pen/wvKPdNZ>

Navbar responsive

Podemos combinar las técnicas **responsive** con código JS para adaptar la barra de navegación.



<https://codepen.io/webUnicen/pen/JjYNXoZ>



Evolución de los Layouts

Flexbox y Grids

Tus nuevos mejores amigos para construir Layouts

Históricamente la **construcción de layouts complejos** utilizando CSS ha resultado **muy complicado** y ha generado enormes frustraciones para lograr un diseño **consistente y coherente** en todos los navegadores.

TABLAS

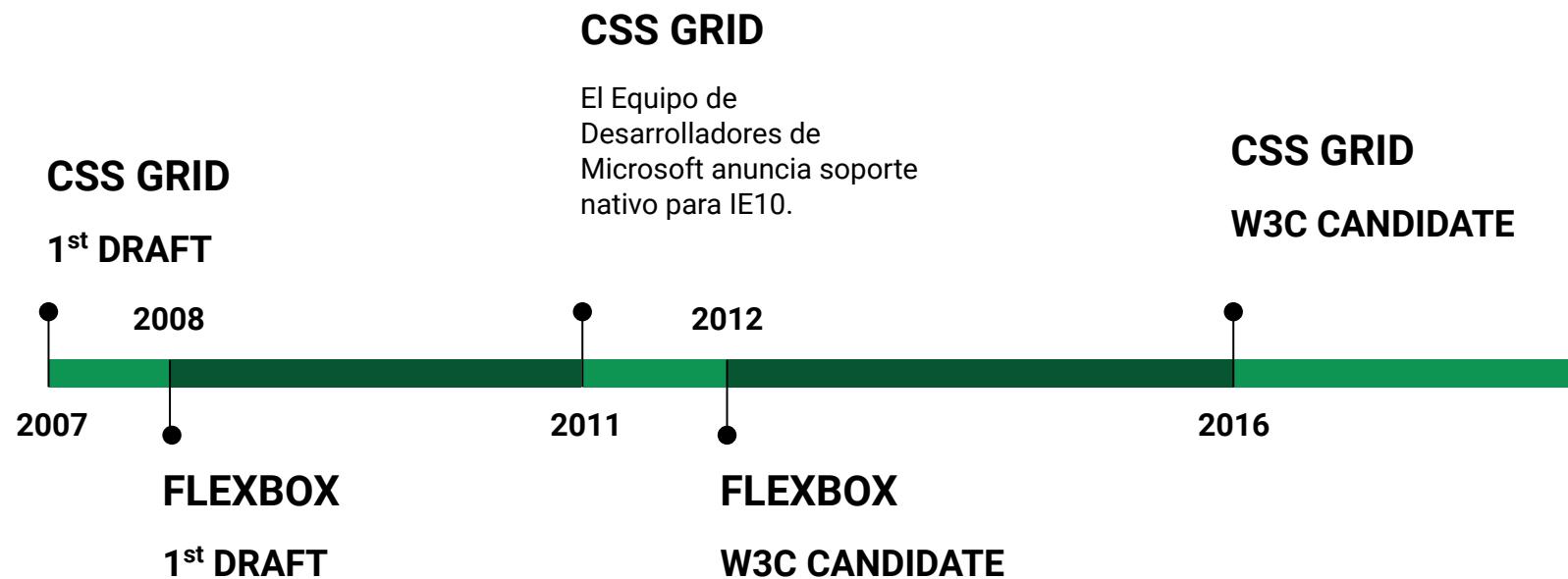
POSICIONAMIENTO

FLOTANTES

BLOQUES EN
LINEA

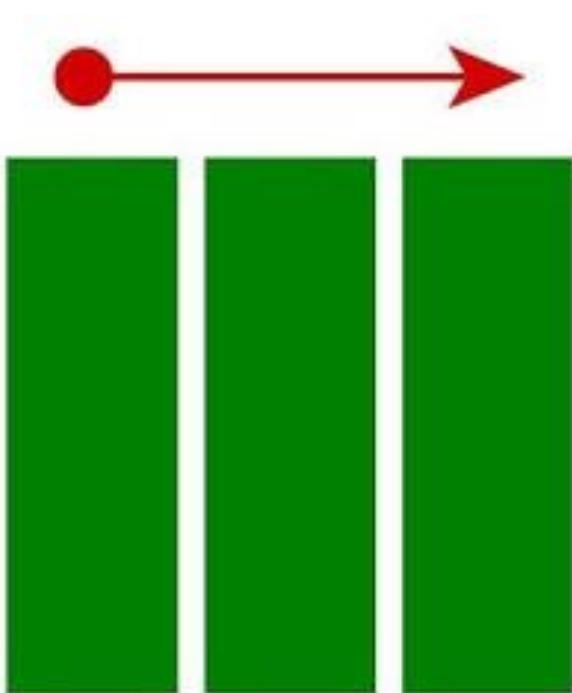
Responsive Layout

Para resolver estas complicaciones y generar diseños “responsive” de alta complejidad, fueron surgiendo técnicas avanzadas de diseño.



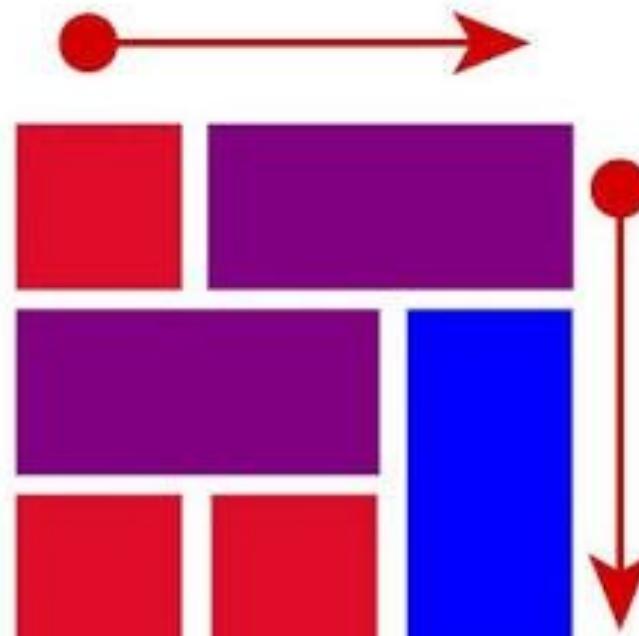
Responsive Layout

FUNDAMENTOS BÁSICOS



Flexbox

One Dimensions

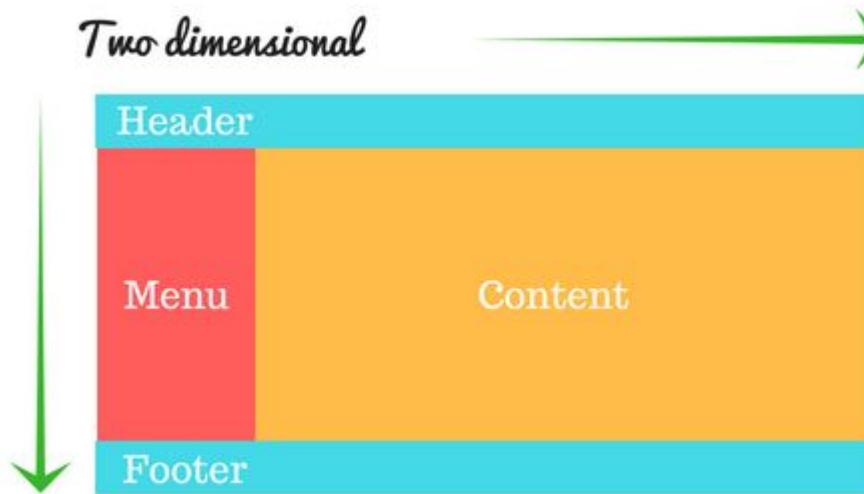


CSS Grids

Two Dimensions

CSS Grid

CSS Grid



MODELO BIIDIMENSIONAL DE LAYOUT

“CSS Grid permite dividir un **contenedor** en varias secciones, permitiendo posicionar y alinear **items** en columnas y filas.”

Puedes colocar los ítem donde quieras, en cualquiera de las celdas que lo forman. Existe un control detallado de la posición y dimensiones de cada elemento.

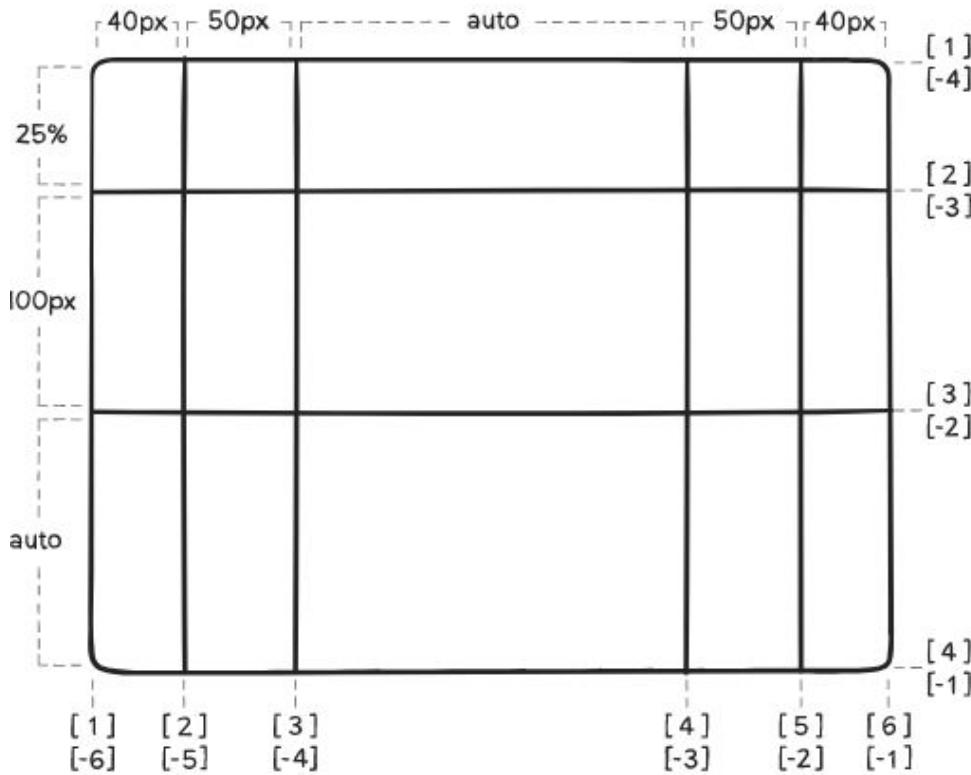
```
.container {  
  display: grid;  
}
```

Grid: filas y columnas

grid-template-columns

grid-template-rows

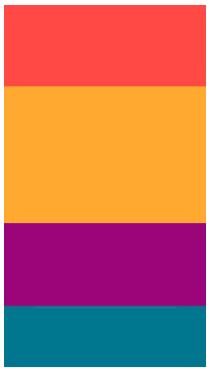
- La propiedad especifica el número y el ancho de las columnas y filas en un **diseño de cuadrícula**.
- Los valores son una lista separada por espacios, donde cada valor especifica el *tamaño de la columna respectiva*.



```
.container {  
    display: grid;  
    grid-template-columns: 40px 50px auto 50px 40px;  
    grid-template-rows: 25% 100px auto;  
}
```

Ejemplo responsive con GRID

MOBILE



DESKTOP



```
<header>
  <h1>LOGO</h1>
</header>
<section class="middle-wrapper">
  <section class="main">
    <h2>Main Content</h2>
  </section>
  <aside class="sidebar">
    <h3>Sidebar</h3>
  </aside>
</section>
<footer>
  <h3>Footer</h3>
  <p>Web Tudai 2020</p>
</footer>
```

```
/* agregamos un contenedor grid */
.middle-wrapper {
  display: grid;
  grid-template-columns: 1fr;
}

/* solo para pantallas superiores a 660px */
@media only screen and (min-width: 660px) {
  .middle-wrapper {
    grid-template-columns: 3fr 1fr;
  }
}
```

CAMBIAMOS LA
CANTIDAD DE
COLUMNAS



Live: <https://codepen.io/webUnicen/pen/xxwPLRe>

Grid Areas

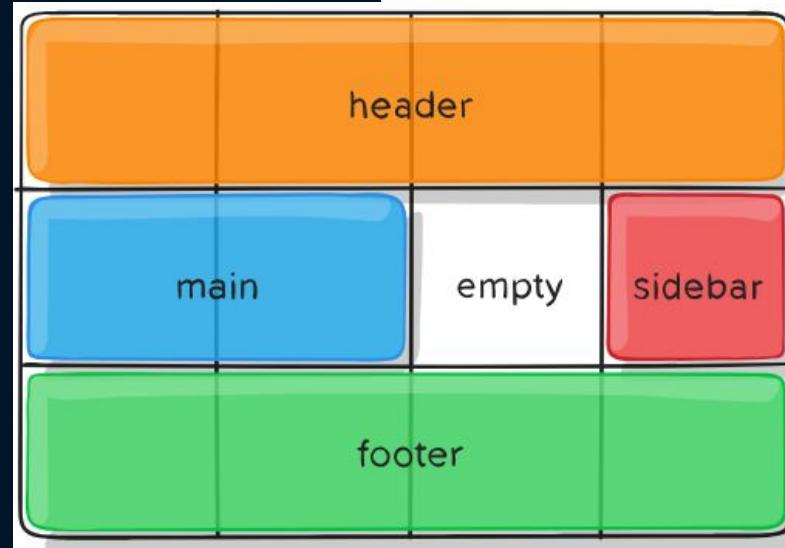
grid-template-areas

- Especifica **nombres** para cada una de las secciones del grid.
- Repetición del área permite que el contenido abarque múltiples celdas.
- La sintaxis visualiza la estructura de la cuadrícula.

```
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    ["header header header header"]  
    ["main main . sidebar"]  
    ["footer footer footer footer"];  
}  
.item-header { /* para cada item */  
  grid-area: header;  
}
```

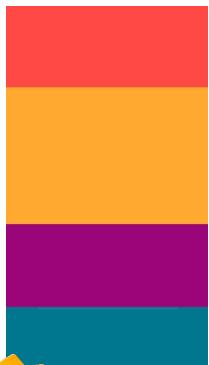
grid-area

- Asocia el nombre de área a un **item**.
- Especifica en qué **área** se posiciona el **item**.



Ejemplo responsive con GRID AREAS

MOBILE



DESKTOP



DEMO

Live: <https://codepen.io/webUnicen/pen/Jvyvmg>

```
<div class="container">
  <header>
    <h1>LOGO</h1>
  </header>
  <section class="main">
    <h2>Main Content</h2>
  </section>
  <aside class="sidebar">
    <h3>Sidebar</h3>
  </aside>
  <footer>
    <h3>Footer</h3>
    <p>Web Tudai 2020</p>
  </footer>
</div>
```

```
/* Page Layout */
.container {
  display: grid;
  grid-template-areas:
    "header"
    "main"
    "sidebar"
    "footer";
}

/* posicionamos los elementos donde queremos */
  "main sidebar"
  "footer footer";
}

header { grid-area: header; }
.main { grid-area: main; }
.sidebar { grid-area: sidebar; }
.footer { grid-area: footer; }

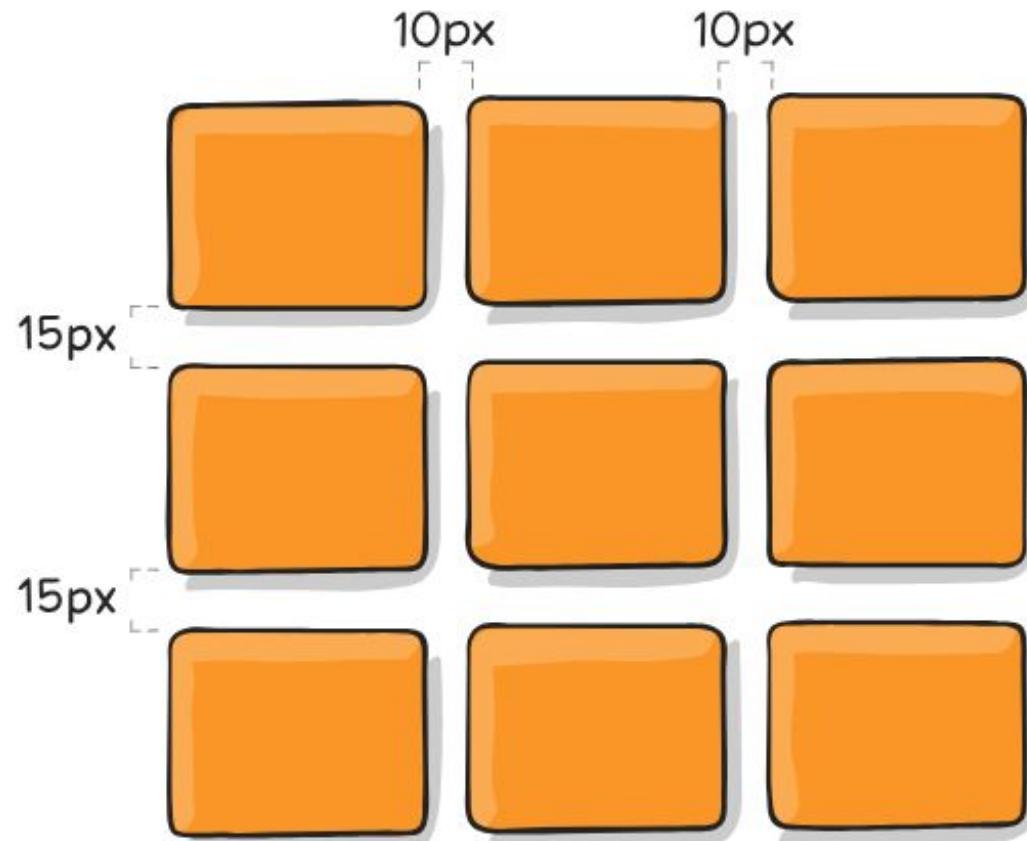
/* solo para pantallas superiores a 660px */
@media only screen and (min-width: 660px) {
  /* Page Layout */
  .container {
    display: grid;
    grid-template-columns: 4fr 1fr;
    grid-template-areas:
      "header header"
```

Grid: filas y columnas

grid-column-gap

grid-row-gap

Define el tamaño del espacio entre las filas y columnas en un diseño de cuadrícula.



Definiendo grid con distintas medidas

Fijas PX:

grid-template-columns: 320px 320px 320px;

Porcentaje %:

grid-template-columns: 33.33% 33.33% 33.33%;

Usando la nueva función repeat()

grid-template-columns: repeat(3, 33.33%);

La unidad fr: Fracción

Una unidad fr describe "una pieza de la muchas piezas en que estamos diviendo ésto". Por ejemplo, podríamos declarar nuestras columnas usando:

grid-template-columns: 1fr 1fr 1fr;

Mejor aun: repeat + fr

grid-template-columns: repeat(3, 1fr);

Otras propiedades para alinear los elementos



justify-items

Para todos los elementos del cuadro, dándoles a todos una forma predeterminada de justificar cada cuadro a lo largo del eje apropiado.

align-items

La propiedad align-items especifica la alineación predeterminada para los elementos dentro del contenedor flexible.

justify-content

La propiedad justify-content alinea los elementos del contenedor flexible cuando los artículos no usan todo el espacio disponible en el eje principal

align-content

Modifica el comportamiento de la propiedad flex-wrap. Es similar a alinear elementos, pero en lugar de alinear elementos flexibles, alinea las líneas flexibles.

Info de Grid recomendada

<https://css-tricks.com/snippets/css/complete-guide-grid/>

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout/Conceptos_B%C3%A1sicos_del_Posicionamiento_con_Rejillas

Curso: <https://cssgrid.io/>

GRID GARDEN

Nivel 1 de 28 ▶

¡Bienvenido a Grid Garden, donde escribirás tu código CSS para cultivar tu jardín de zanahorias! Riega solo las áreas que tienen zanahorias usando la propiedad **grid-column-start**.

Por ejemplo, **grid-column-start: 3;** regará el área comenzando por la tercera línea vertical, que es otra manera de decir el 3er borde vertical contando desde la izquierda de la cuadrícula.

```
1 #garden {  
2   display: grid;  
3   grid-template-columns: 20% 20% 20%;  
4   grid-template-rows: 20% 20%;  
5 }  
6  
7 #water {  
8   background-color: white;  
9 }  
10  
11  
12  
13  
14
```

¿Jugamos?

<https://cssgridgarden.com/#es>

Siguiente

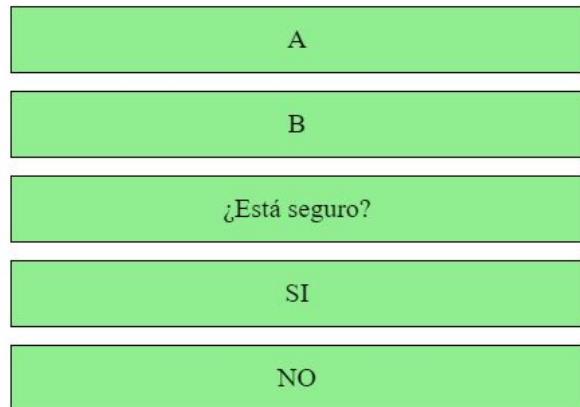
Grid Garden es una creación de [Codepip](#) • [GitHub](#) • [Twitter](#) • [Español](#)

Want to learn CSS flexbox? Play [Flexbox Froggy](#).

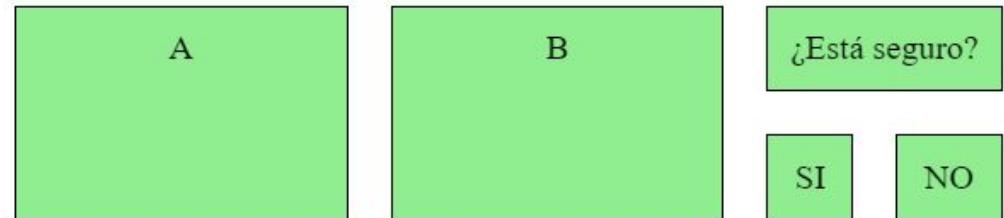
Comparativa Flex vs Grid

Armar este layout responsive con Flex y con Grid

MOBILE

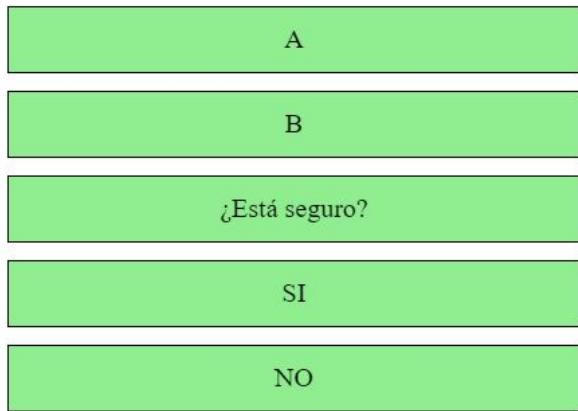


DESKTOP



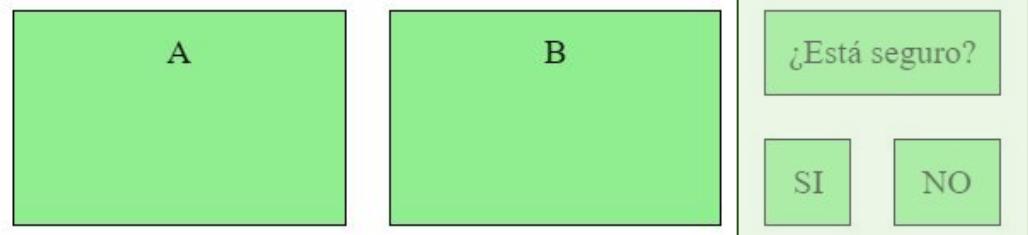
Necesidad de Flex anidados

MOBILE



```
<section class="container">
  <div>A</div>
  <div>B</div>
  <div>¿Está seguro?</div>
  <div>SI</div>
  <div>NO</div>
</section>
```

DESKTOP



```
<section class="container">
  <div>A</div>
  <div>B</div>
```

```
<section class="confirma">
  <div>¿Está seguro?</div>
  <div class="respuesta">
    <div>SI</div>
    <div>NO</div>
  </div>
</section>
</section>
```

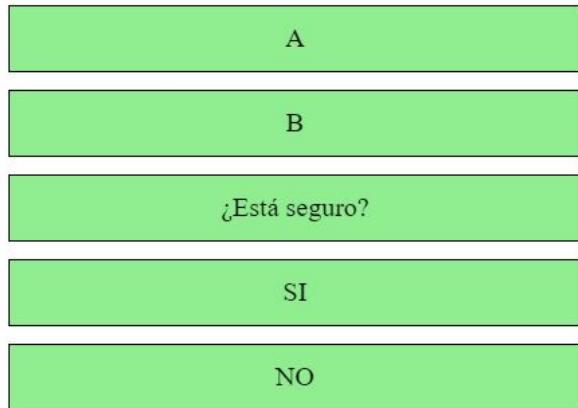


DEMO

<https://codepen.io/webUnicen/pen/XWmzgZV>

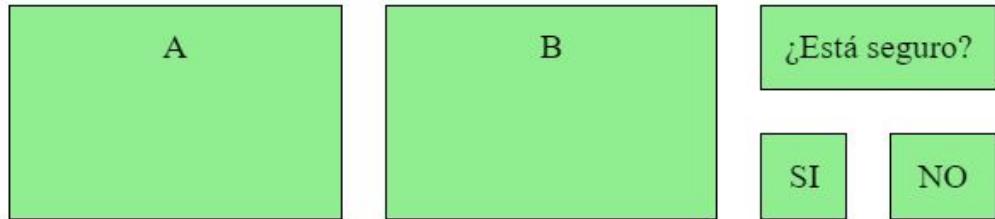
Con Grid Áreas podemos evitar anidamiento

MOBILE



```
<section class="container">
  <div class="a">A</div>
  <div class="b">B</div>
  <div class="pregunta">¿Está seguro?</div>
  <div class="res-ok">SI</div>
  <div class="res-no">NO</div>
</section>
```

DESKTOP



```
...
.a {grid-area: area-a}
.b {grid-area: area-b}
.pregunta {grid-area: area-preg}
.res-ok {grid-area: area-ok}
.res-no {grid-area: area-no}

/* solo para pantallas superiores a 660px */
@media only screen and (min-width: 660px) {
  /* layout */
  .container {
    display: grid;
    grid-template-areas:
      "area-a area-b area-preg area-preg"
      "area-a area-b area-ok   area-no"
  }
}
```

DEMO

<https://codepen.io/webUnicen/pen/xxwPWqY>

Debate

En este caso, el “section” que agregamos en el ejemplo de Flex tiene sentido semántico. Entonces, cual conviene?

Flex o Grid?

Flexbox vs Grid

CSS Grid es un *sistema bidimensional*, lo que significa que puede manejar columnas y filas, a diferencia de **Flexbox**, que es un *sistema unidimensional* que define un eje y el otro actúa en función a éste.

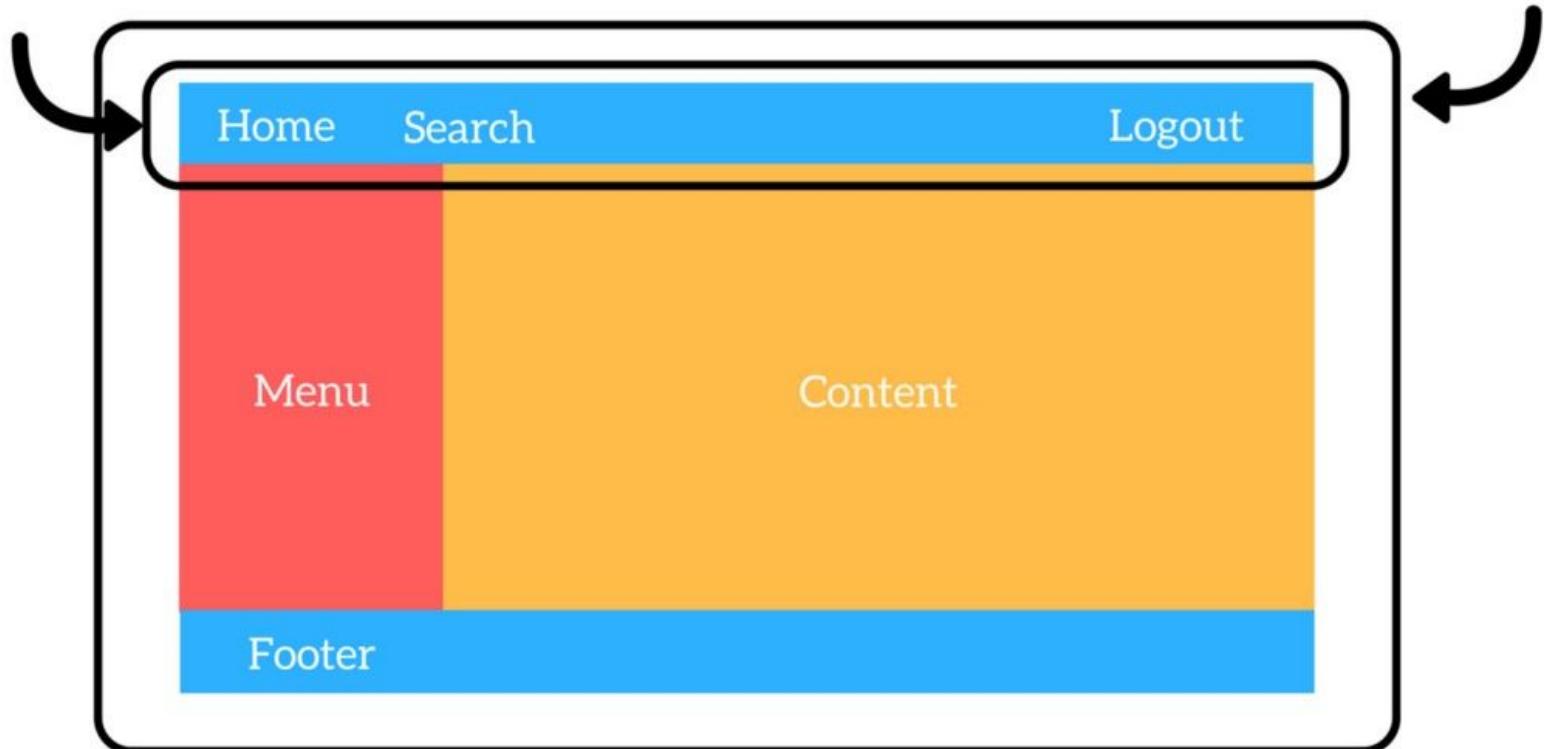
Layout-First vs Content-First



Flexbox vs Grid

Flexbox Container

Grid Container



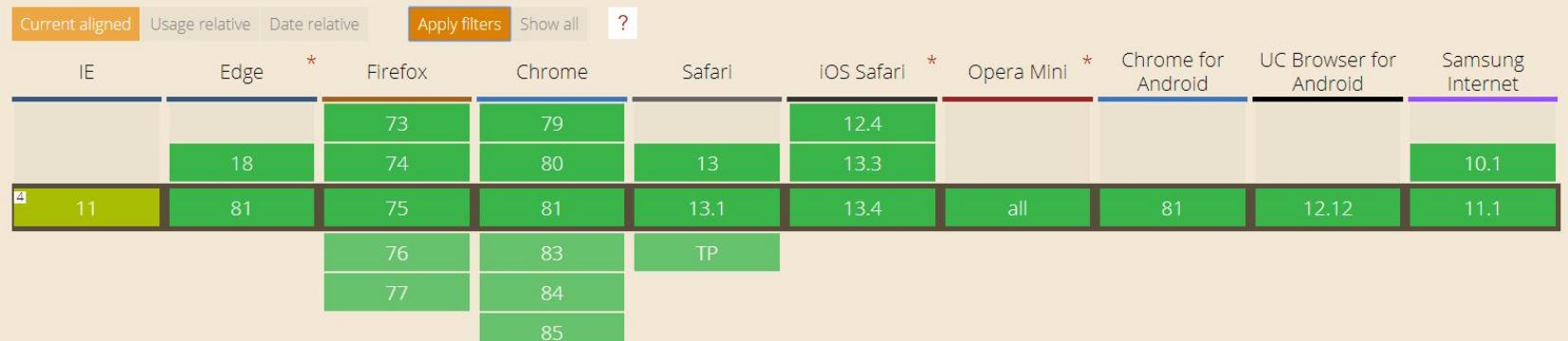
Soporte de Navegadores

CSS Flexible Box Layout Module - CR

Method of positioning elements in horizontal or vertical stacks.
Support includes all properties prefixed with `flex`, as well as
`display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Usage
Global
unprefixed:

% of all users
95.83% + 2% = 97.83%
95.69% + 1.49% = 97.18%

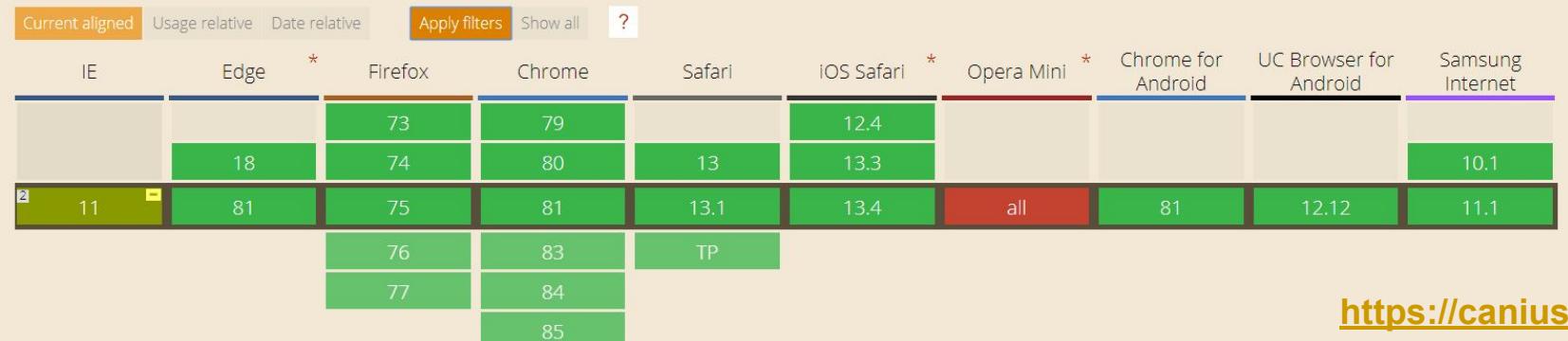


CSS Grid Layout (level 1) - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors.
Includes support for all `grid-*` properties and the `fr` unit.

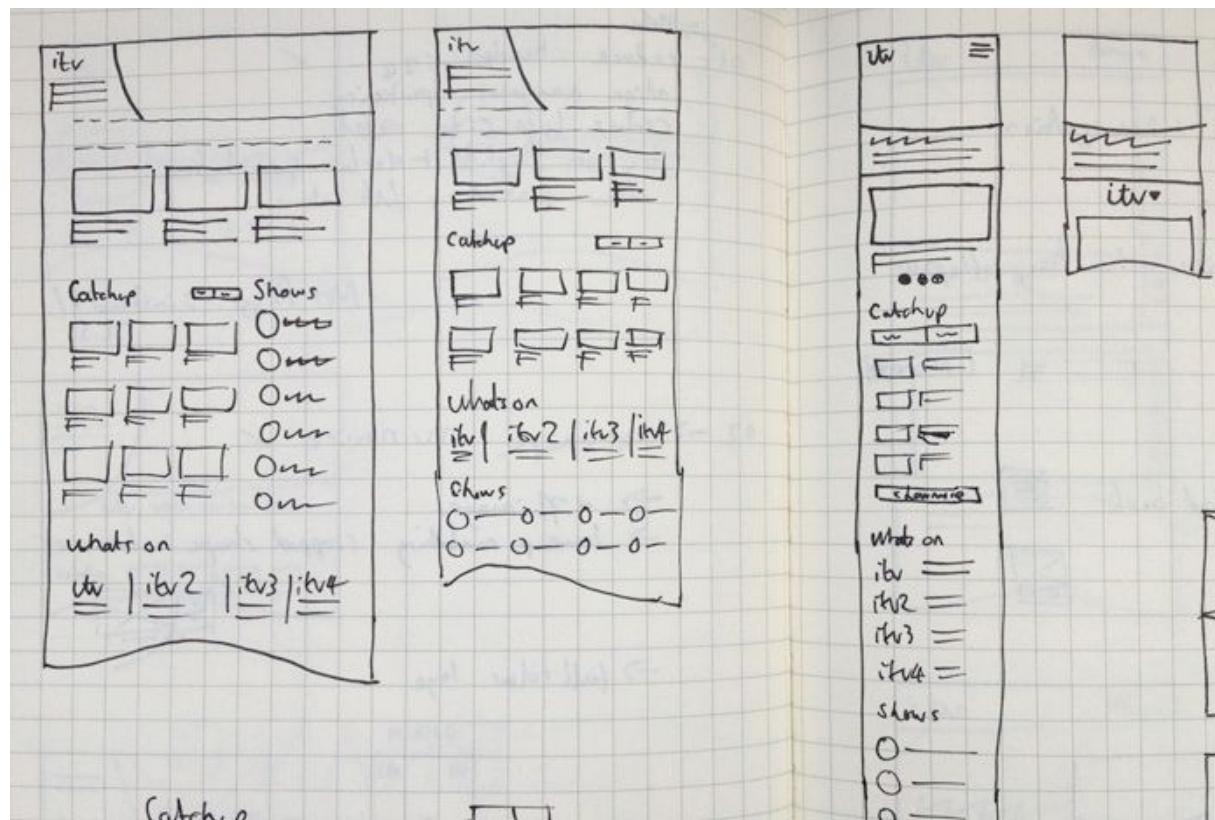
Usage
Global
unprefixed:

% of all users
92.94% + 1.55% = 94.49%
92.94%



Técnicas: Sketching

Dibujo rápido o bosquejo que no tiene muchos detalles y que reproduce un concepto o idea de una manera muy sencilla. Si no gusta o se valida con usuarios (Prototipo en baja) que no sirve, se tira sin mucho análisis y se crea uno nuevo.



**AHORA LES TOCA
PRACTICAR :D**



Bibliografía

- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- <https://css-tricks.com/snippets/css/complete-guide-grid>
- <https://developer.mozilla.org/es/docs/Web/CSS/flex>
- <https://developer.mozilla.org/es/docs/Web/CSS/grid>
- <https://caniuse.com/>
- https://en.wikipedia.org/wiki/Responsive_web_design
- <https://cssgrid.io/>