Introducción a CSS3

Introducción a las hojas de estilo con CSS3

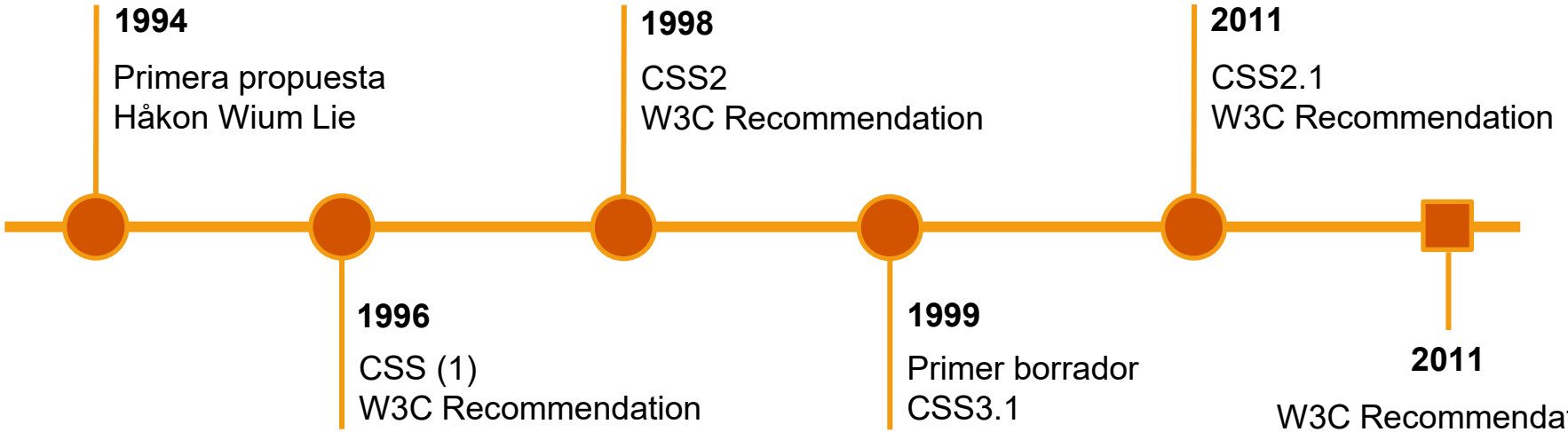
Tony G. Bolaño
@tonybolanyo – tony@tonygb.com
KeepCoding HTML5 / CSS3
Junio 2021

Introducción a CSS3

¿Qué es CSS?

- CSS: Cascading Style Sheets (hojas de estilo en cascada).
- Lenguaje para definir la presentación de un documento estructurado escrito en un lenguaje de marcado.
- Diseñado para permitir la separación entre el contenido del documento y la forma de presentarlo.

Un poco de historia



Fuente: https://en.wikipedia.org/wiki/HTML

¿Cómo usar CSS?

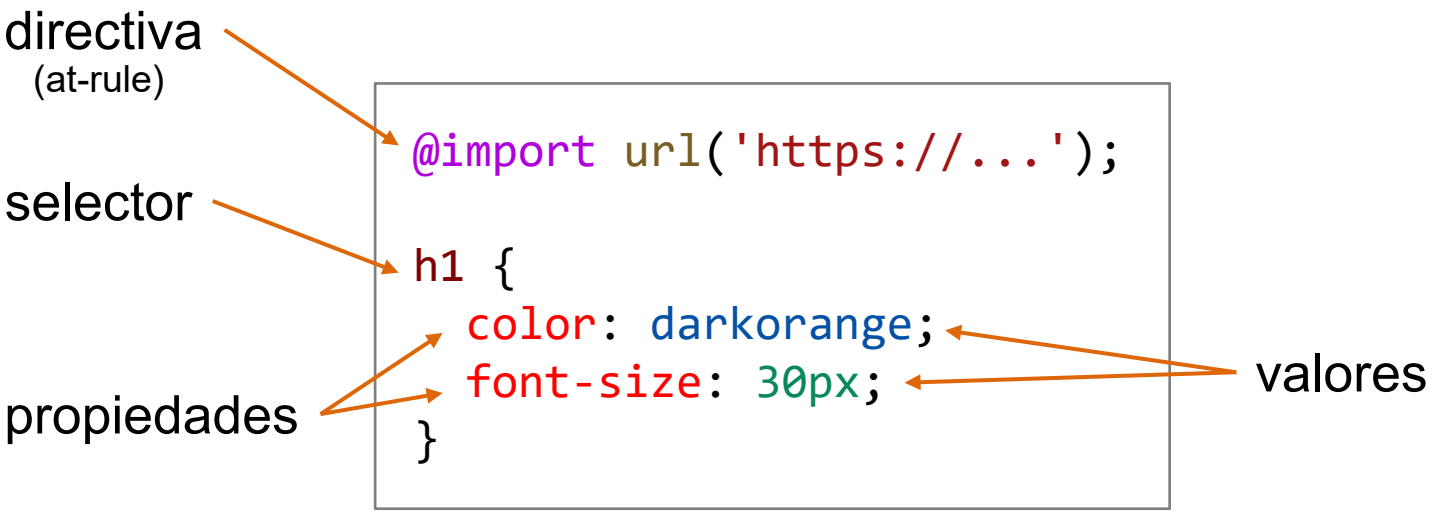
- Estilos en línea, mediante el atributo `style`:

```
<p style="propiedad: valor">
```

- Estilos independientes. Uso de selectores:
  - Embebidos con la etiqueta `<style>`
  - En ficheros CSS independientes

```
<link rel="stylesheet" href="./styles.css">
```

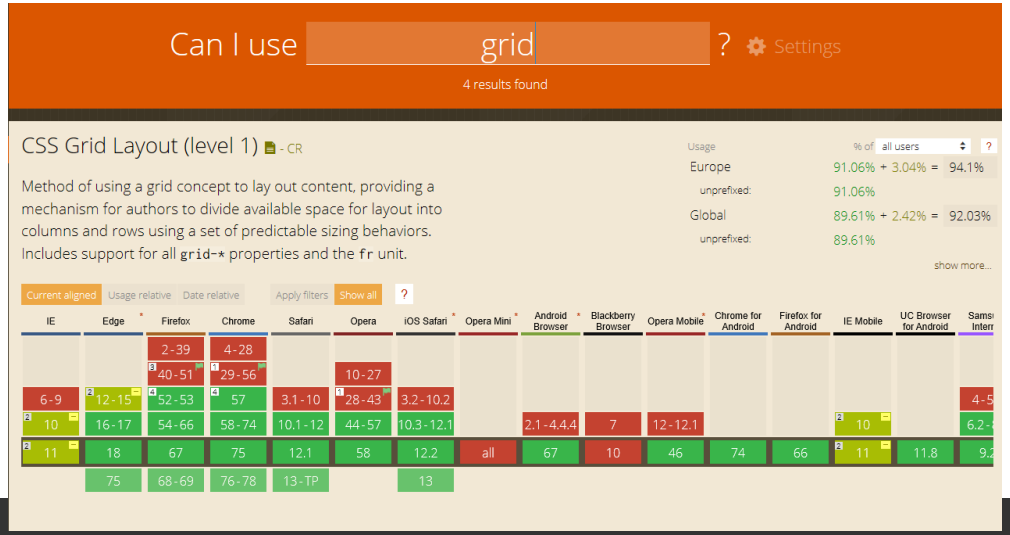
Sintaxis: selectores, propiedades y directivas



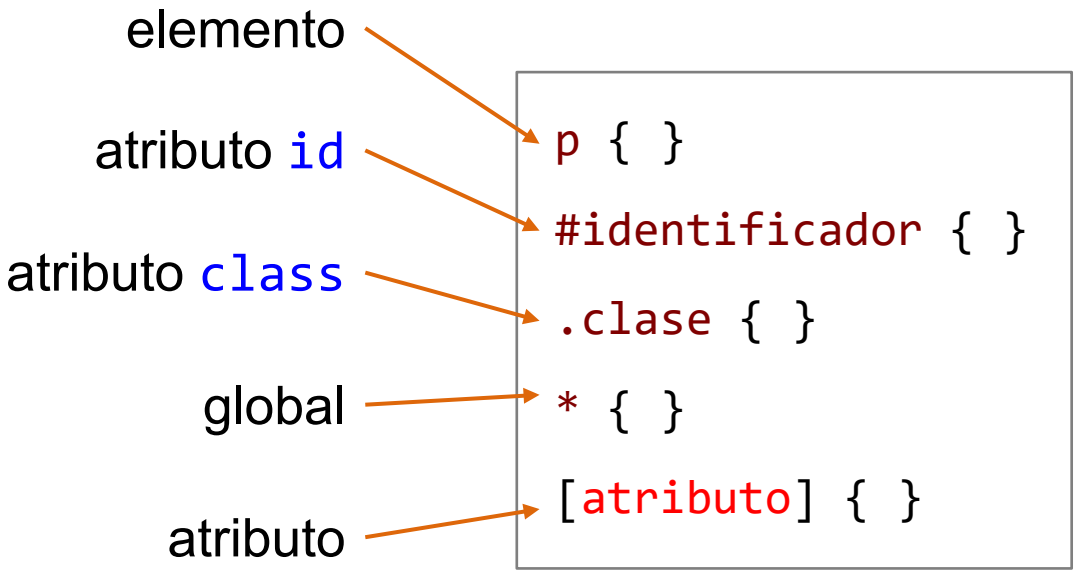
Aplicando CSS3 básico

Herramientas y referencias

- MDN: https://developer.mozilla.org/en -US/docs/Web/CSS/Reference
- Can I Use: https://caniuse.com
- Navegador: inspector



Selectores básicos




```
a[href$=".org"] {
font-style: italic;
}
```

`[atributo]`
elemento que tienen el atributo

`[atributo="valor"]`
elementos con ese valor

`[atributo^="valor"]`
elementos que comienzan con el valor

`[atributo$="valor"]`
elementos que terminan con ese valor

`[atributo*="valor"]`
elementos que contienen el valor

`[atributo|="valor"]`
valor exacto o principio seguido de guión

`[atributo~="valor"]`
lista de palabras y una es valor

Colores

- Sistemas de colores
  - Nombres
  - rgb() - red, green, blue
  - hexadecimal (#)
  - hsl()
  - hueº (matiz), saturation%, luminosity%
- Transparencia canal alfa entre 0 (transparente) y 1 (opaco)
  - rgba()
  - hsla()
  - Opacity / se hereda

Tipografía

- Podemos incluir nuevas fuente de varias formas:
  - Mediante la propiedad `@font-face`

```
@font-face {
  font-family: miFuente;
  src: url();
}
```

  - Importándola de webs como Google Fonts

```
@import url();
```

```
<link href='' type='text/css'>
```

Propiedades para fuentes

Propiedad	Significado
`font-style`	Define el estilo de un texto. Normal, italic,…
`font-variant`	Variante del tipo de letra. Normal, small-caps
`font-weight`	Peso de la letra. Normal, bold, 100, 900,…
`font-size`	Tamaño de la fuente.
`font-family`	Familia de la fuente
`font`	Abreviatura de todas las anteriores en el mismo orden de aparición

Otras propiedades:

- `text-decoration, text-transform`
- `line-height, letter-spacing, word-spacing`
- `text-align, vertical-align, text-indent`

Unidades de medida

UNIDADES ABSOLUTAS	
Unidad	**Medida**
`in`	Pulgada (2.54cm)
`cm`	Centímetro
`mm`	Milímetro
`pt`	Punto. 1pt = 1/72in
`pc`	Pica. 1pc=12pt
`px`	Relativos a la pantalla

UNIDADES RELATIVAS	
Unidad	**Relativo a**
`em`	Tamaño de letra del elemento
`ex`	Altura de la “x” del elemento
`vw`	Anchura del viewport
`vh`	Altura del viewport
`%`	% del tamaño del elemento
`rem`	Tamaño de letra en el root `<html>`

```
.container {
  font-size: 2rem;
  width: 800px;
  height: 100vh;
}
```

Selectores múltiples y especificidad

Selectores múltiples y anidados

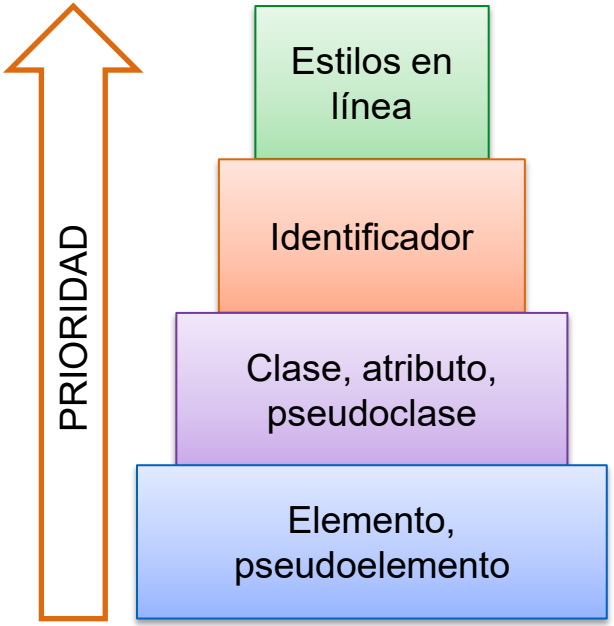
`h1, h2, h3 { }`	• Aplica estilos a múltiples selectores
`div p { }`	• Descendientes de cualquier nivel
`div > p { }`	• “Hijos” directos
`div + p { }`	• “Hermanos” adyacentes (inmediatos)
`div ~ p { }`	• “Hermanos” en general

Aplicación de múltiples estilos

- **Herencia**
  - Propiedades que se heredan en los elementos hijos
    `font-family`, `color`… => afectan al contenido
  - Propiedades que no se heredan
    `margin`, `padding`, `border`… => afectan al contenedor
- **Cascada**
  - agente de usuario: por defecto / definidos por el usuario
  - bloques de CSS: externos / embebidos
  - estilos en línea
- **Especificidad**

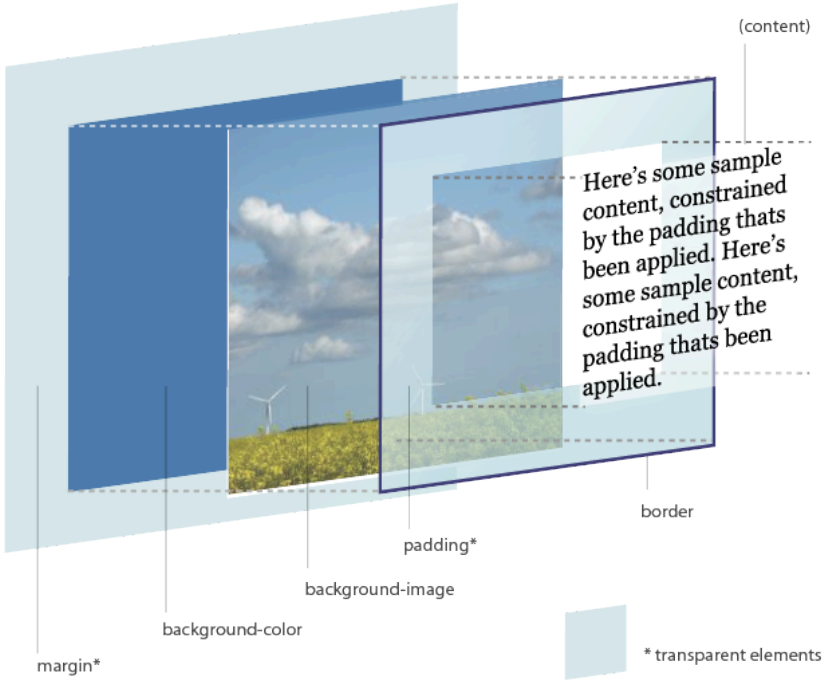
Especificidad

- Orden de prioridad (menor a mayor especificidad):
  - Selectores de tipo (`p`, `h1`) y pseudo-elementos (`::before`, `::after`)
  - Selectores de clase (.ejemplo), selectores de atributos (`[type="text"]` y pseudo-clases (`:hover`)
  - Selectores ID (`#ejemplo`)
- Los estilos inline (`style="color: darkorange"`) tienen prioridad sobre los externos



Modelo de caja

Modelo de caja en CSS



- Contenido
- Espaciado interno (padding)
- Borde
- Imagen de fondo
- Color de fondo
- Margen

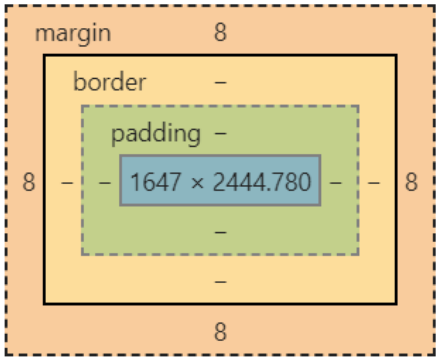
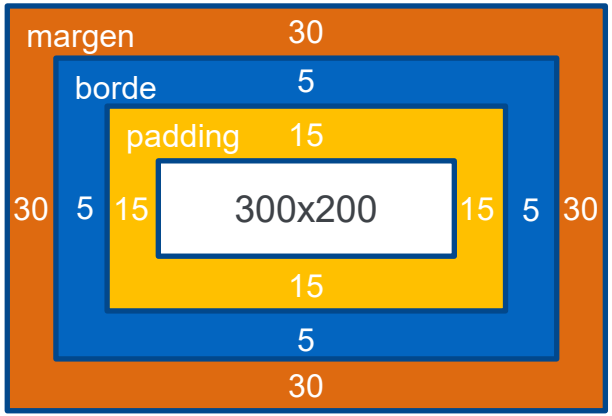


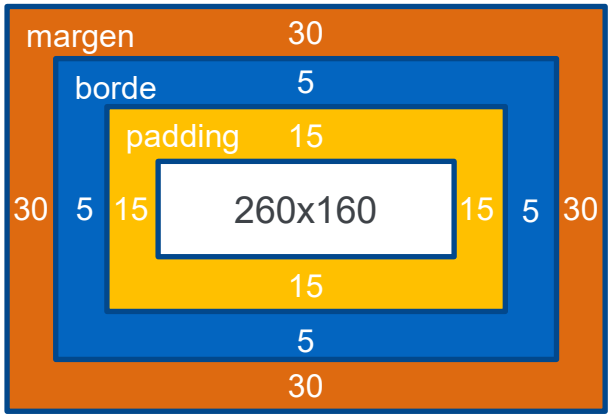
Imagen: https://hicksdesign.co.uk/boxmodel/

Modelo de caja. Sistemas de medida

box-sizing: content-box;



box-sizing: border-box;




```
.container {
  width: 300px;
  height: 200px;
  padding: 15px;
  border: 5px solid gray;
  margin: 30;
}
```

Modelo de caja. Propiedades y usos.

- Propiedades
  - margin
  - padding
  - border
  - border-radius
  - border-image
- Usos
  - wrappers
  - centrado de bloques

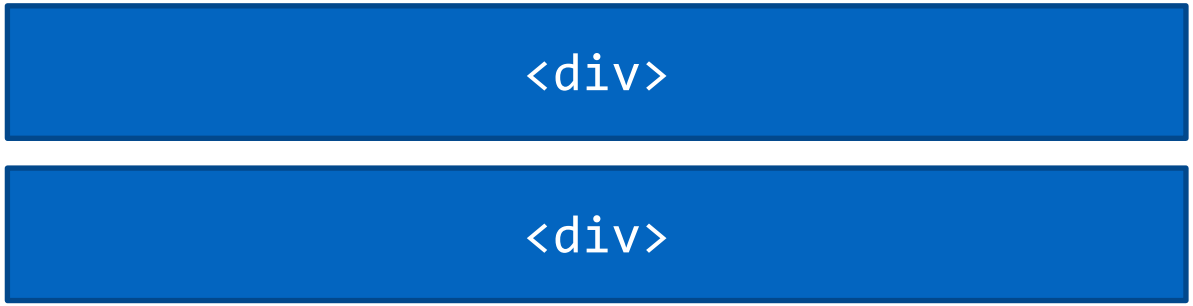
Display

Display. Opciones básicas

- La propiedad `display` en la más importante para controlar la estructura de la web.
- Cada elemento HTML tiene un valor por defecto.
- Algunos elementos de “nivel bloque” son: `<div>`, `<h1>…<h6>`, `<p>`, `<form>`, `<section>`
- Algunos elementos de “nivel inline” son: `<span>`, `<a>`, `<img>`

display: block

- Un elemento block comienza en una nueva linea y se expande en el eje horizontal tanto como pueda.



display: inline

- Un elemento *inline* continua en la misma línea que el resto del contenido.
- No respeta ni márgenes, ni los paddings **top** ni **bottom** .

Esto es `<span>un elemento</span>` inline.

display: inline-block

- Se comporta igual que un elemento *inline*, pero a diferencia de él, puede tener tanto un **width** como un **height** .
- También respeta los márgenes y los paddings.

Esto es `<span>un elemento</span>` inline.

display: none

- Oculta el elemento sin dejar espacio en el sitio que debería ocupar.
- Usado por algunos elementos como la etiqueta <script>.

Posicionamiento clásico

Posicionamiento

- La propiedad `position` nos permite crear estructuras más complejas en nuestros diseños.
- Por defecto, los navegadores posicionan todos los elementos de forma automática.
- Puede tomar lo valores:
  - `static` (por defecto): sin posicionamiento específico.
  - `relative`: se posiciona como static y luego se desplaza
  - `absolute`: referencia al ancestro más cercano (no static).
  - `fixed`: variante del absoluto. Su posición es inamovible.

Posicionamiento

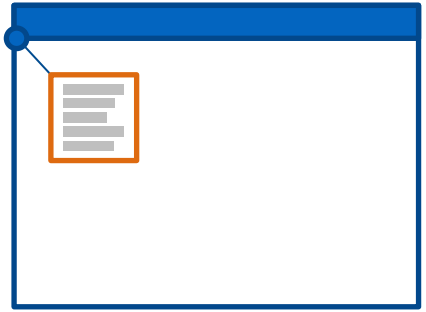
position: static;

ESTÁTICO O
NO POSICIONADO

position: relative;

RELATIVO

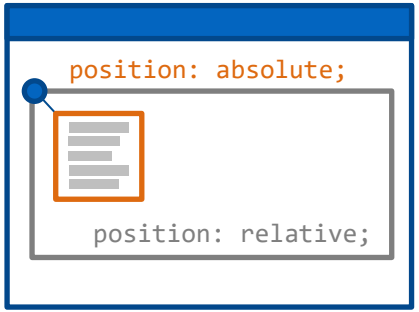
position: absolute;



ABSOLUTO

position: fixed;

FIJO



(RELATIVAMENTE) ABSOLUTO

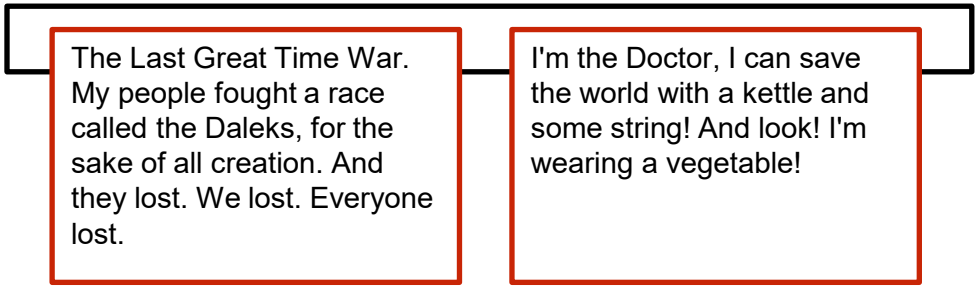
Cambios del flujo estático

- Float: left / right
- Clear: left / right / both

- Overflow: hidden / auto

Técnica de clear-fix con ::after

```
::after {
  content : "";
  display: block;
  clear: both;
}
```



"Cuando un elemento tiene a todos sus hijos flotando, pierde su altura"

Más propiedades

Backgrounds: imágenes y patrones para el fondo

```
background-image: url(./assets/about-bg.jpg);
background-repeat: no-repeat;
background-size: cover;
background-position: center center;
background-attachment: fixed;
background-clip: border-box;
background-blend-mode: normal;
```

Controlando el desplazamiento

```
scroll-behavior: auto | smooth;
```

- Solamente se aplica a los desplazamientos provocados por la navegación o las APIs del DOM.
- Cuando esta propiedad está especificada en el elemento raíz, se aplica al viewport.

Sombras

- Sombra de caja:

```
.container {
  box-shadow: offset-x offset-y blur-radius spread-radius color
}
```

- Sombra de texto:

```
.foo {
  text-shadow: offset-x offset-y blur-radius color
}
```

Variables, cálculos y selectores avanzados

Variables. Uso de `calc()`

- Variables definidas globalmente:

```
:root {
  --color-principal: #06c;
}
```

- Se utilizan mediante la función `var()`

```
#foo h1 {
  color: var(--color-principal);
}
```

- La función `calc()` opera con variables y números

```
.container {
  --separacion: 20;
  margin-top: calc(var(--separacion) * 2px);
}
```

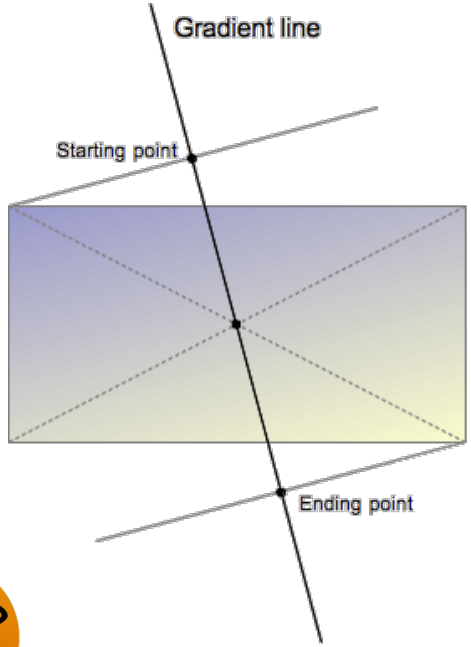
Pseudo-elementos y Pseudo-clases

- **Pseudoelementos** : elementos definidos desde CSS.
  - `::first-letter`
  - `::first-line`
  - `::before {content=""}`
  - `::after {content=""}`

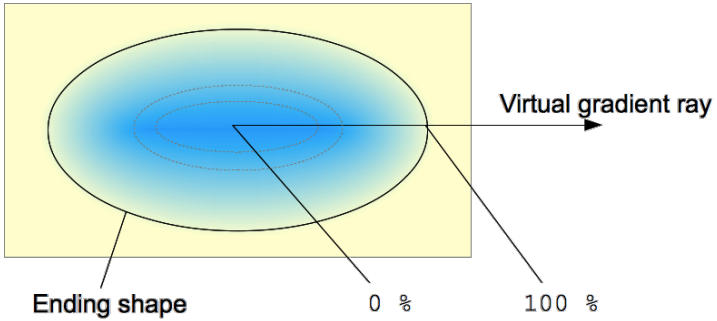
- **Pseudoclases** : clases aplicadas dinámicamente a elementos reales de HTML en función de su estado
  - `:link, :visited, :active, :focus :hover, :target, :lang()`
  - `:nth-child(n), :first-child, :last-child, only-child`
  - `:nth-of-type(n), :first-of-type, :last-of-type,`
  - `:only-of-type, :not()`

Degradados

```
background: linear-gradient(
  dirección, color-stop1,
  color-stop2, color-stop-n);
```



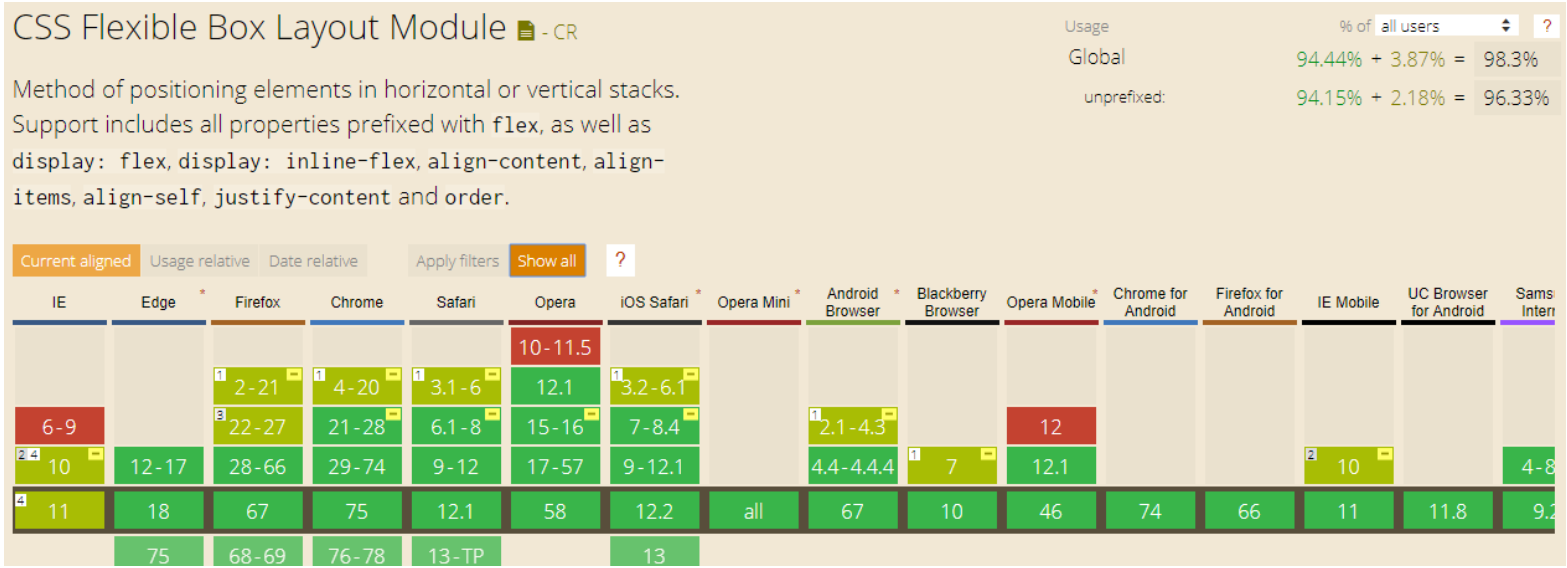

```
background: radial-gradient(
  figura at posición,
  color-stop1, color-stop2,
  color-stop-n);
```

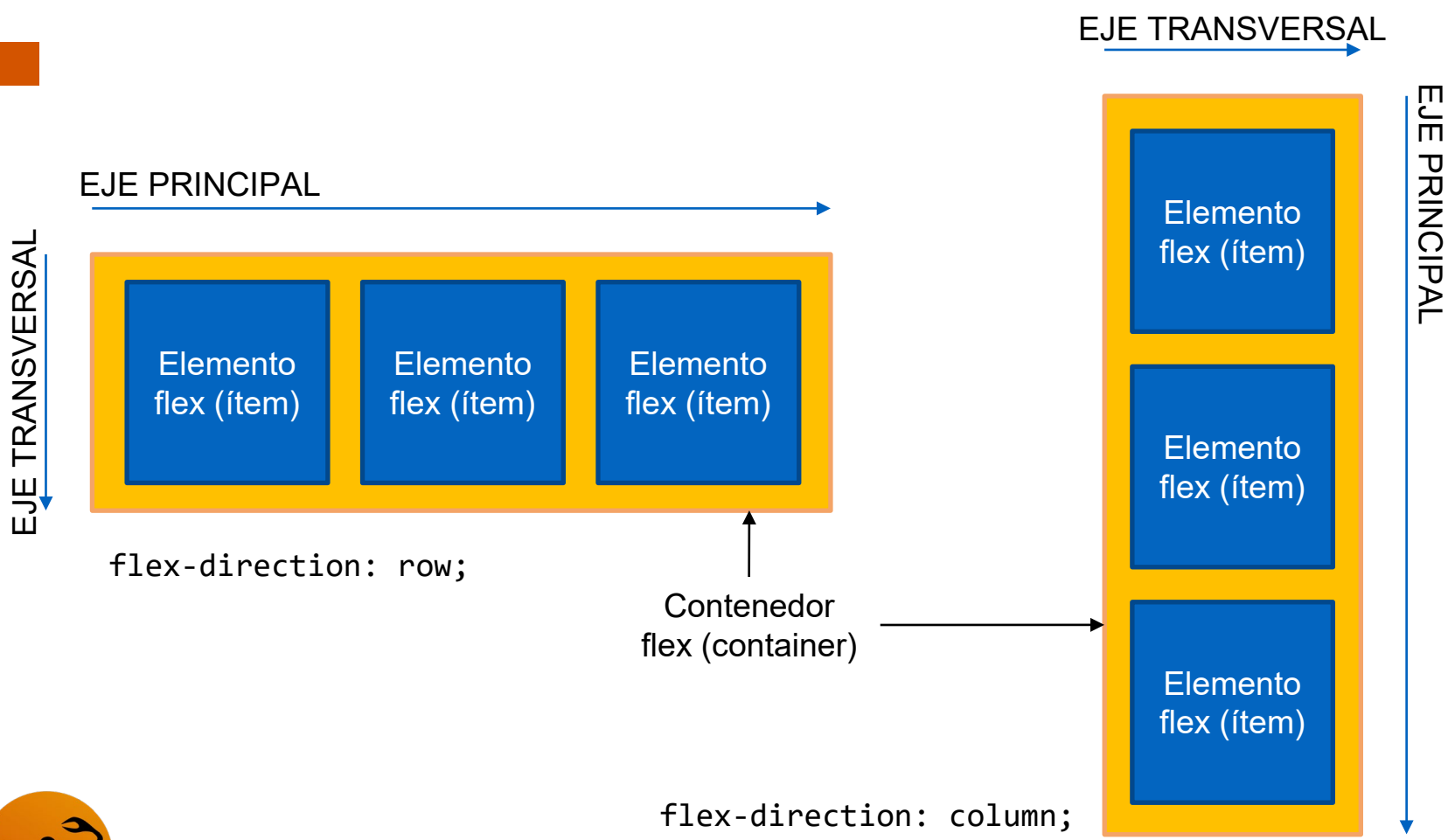


Posicionamiento avanzado con Flexbox

Flexbox

- Método para posicionar elementos en una sola dimensión





Flex Container

```
display: flex;
display: inline-flex;

flex-flow: row nowrap; (flex-direction flex-wrap)

flex-direction: row;
flex-direction: row-reverse;
flex-direction: column;
flex-direction: column-reverse;

flex-wrap: nowrap;
flex-wrap: wrap;
flex-wrap: wrap-reverse;

justify-content: flex-start;
justify-content: center;
justify-content: flex-end;
justify-content: space-between;
justify-content: space-around;
```

```
align-items: stretch;
align-items: flex-start;
align-items: center;
align-items: flex-end;
align-items: baseline;

align-content: stretch;
align-content: flex-start;
align-content: center;
align-content: flex-end;
align-content: space-between;
align-content: space-around;
```

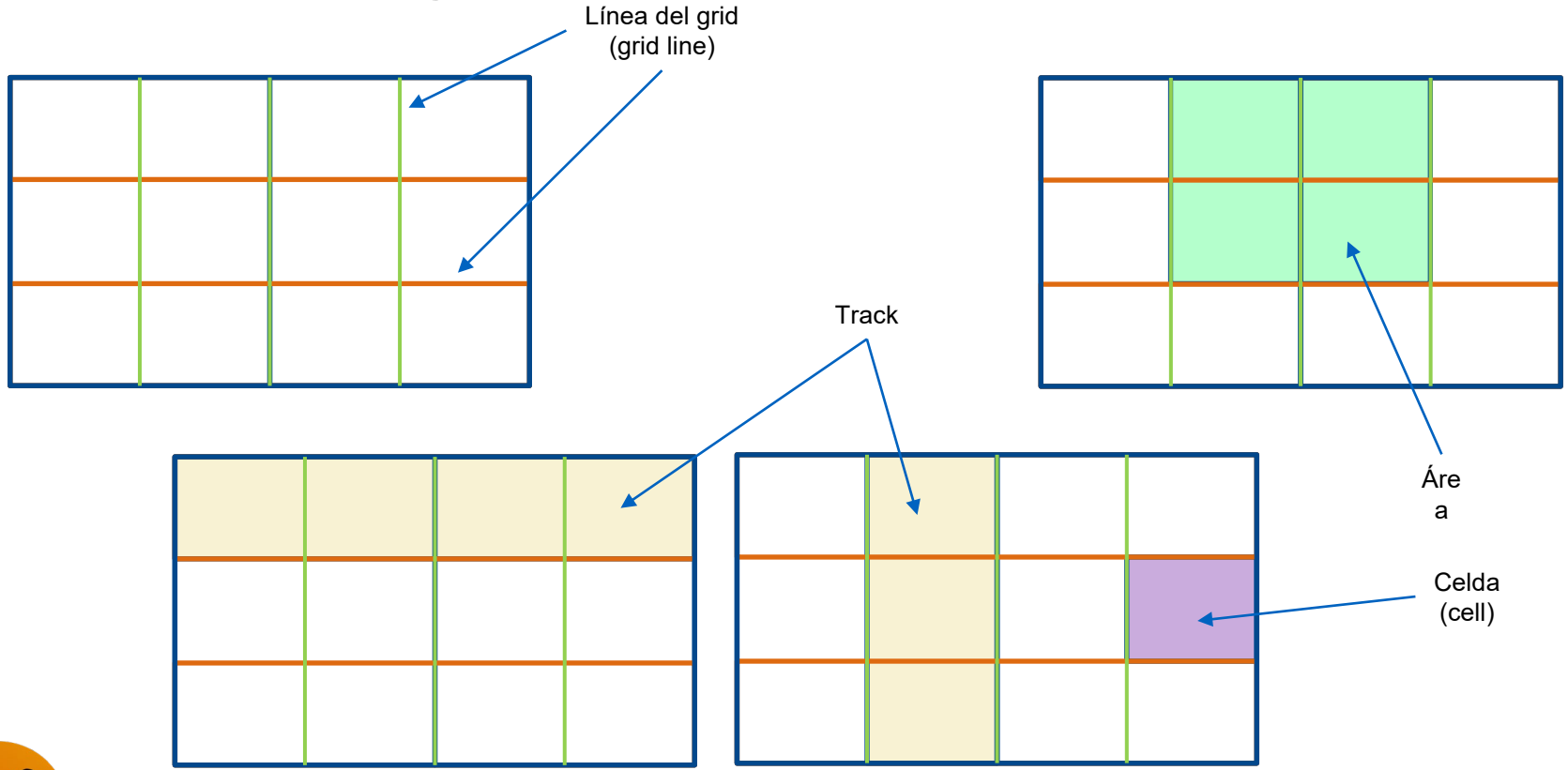
Flex Item

```
order: 0;

flex: 0 1 auto; (flex-grow flex-shrink flex-basis)

flex-grow: 0;
flex-grow: 1;

flex-shrink: 0;
flex-shrink: 1;

flex-basis: auto;

align-self: auto;
align-self: flex-start;
align-self: center;
align-self: flex-end;
align-self: baseline;
align-self: stretch;
```

Fuente: https://darekkay.com/dev/flexbox-cheatsheet.html

Posicionamiento avanzado con CSS Grid

El modelo de grid



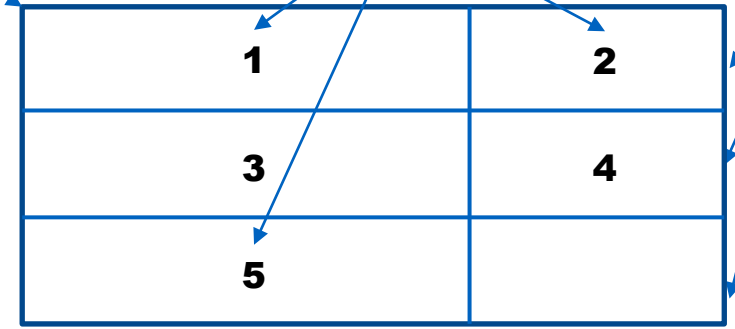
HTML Markup y CSS básico

Contenedor como grid

Elementos del grid

Las filas se crean según sea necesario (implícitas).

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```



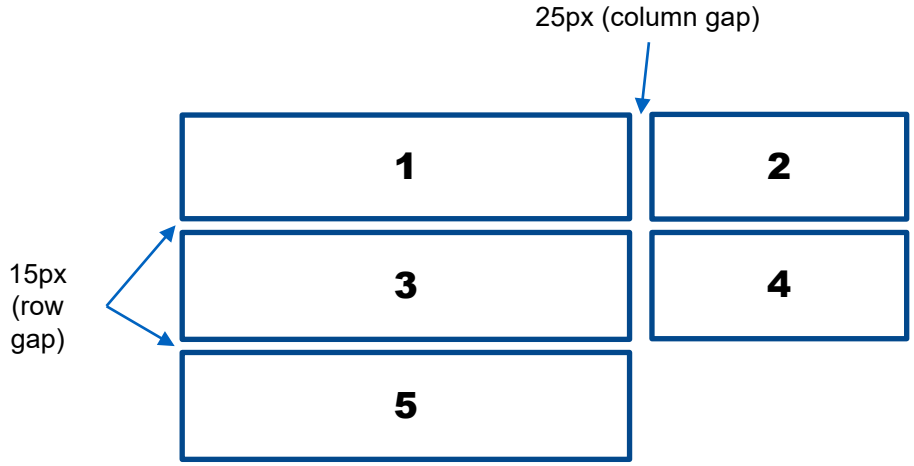

```
.container {
  display: grid;
  grid-template-columns: 400px 200px;
}
```

Dos columnas con ancho específico.
Las columnas están definidas explícitamente

Espaciado entre celdas

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

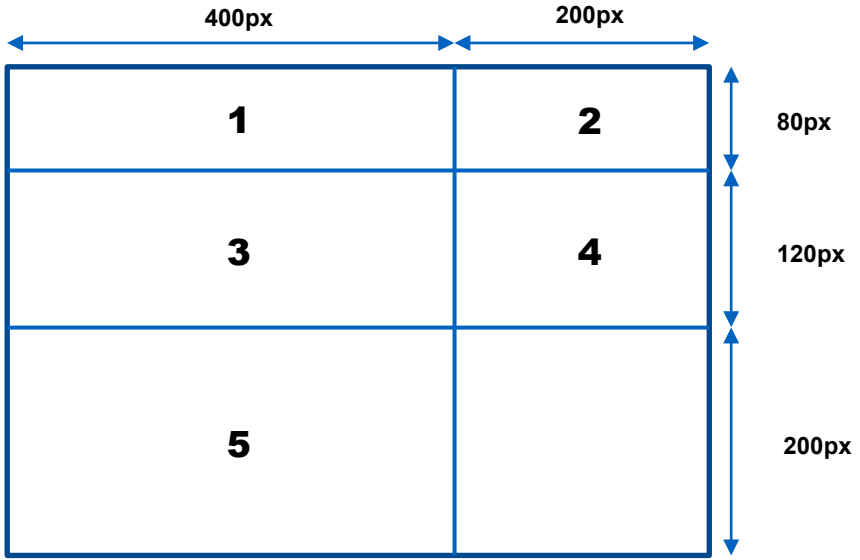
```
.container {
  display: grid;
  grid-template-columns: 400px 200px;
  grid-gap: 15px 25px;
}
```



Definir filas y columnas explícitamente

```html
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

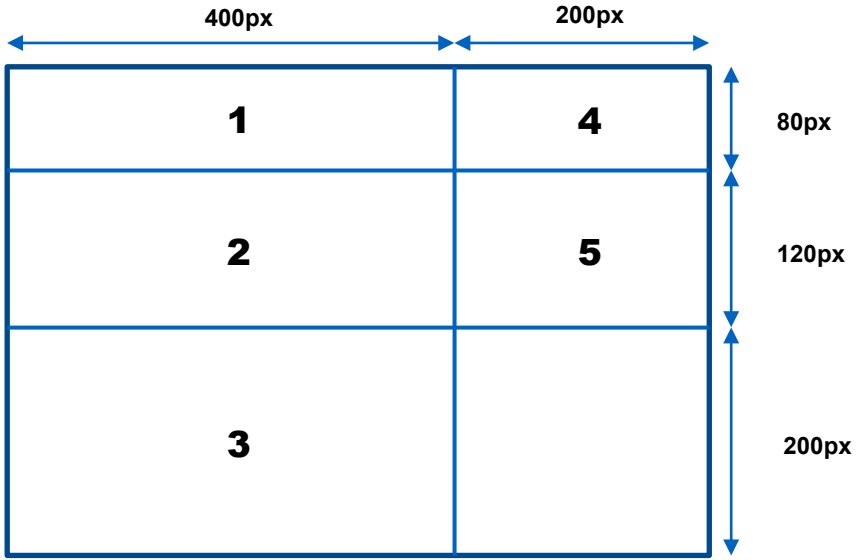
```css
.container {
  display: grid;
  grid-template-columns: 400px 200px;
  grid-template-rows: 80px 120px 200px;
}
```



Elementos en columnas

```html
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

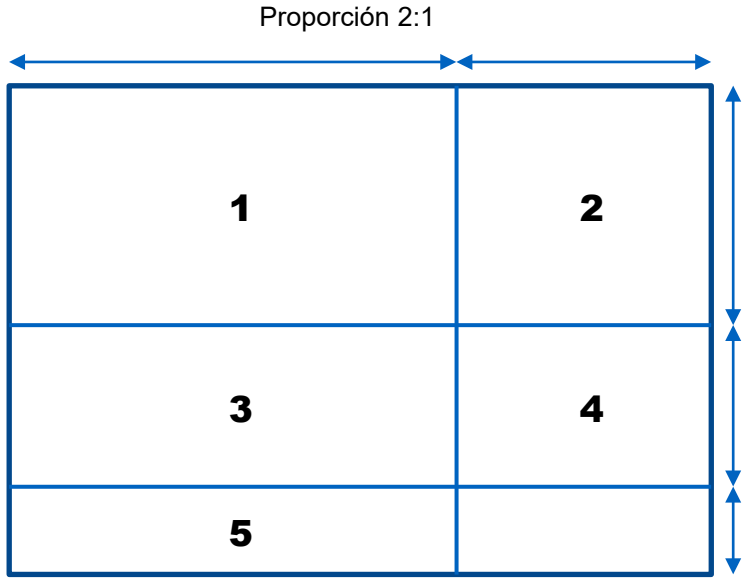
```css
.container {
  display: grid;
  grid-template-columns: 400px 200px;
  grid-template-rows: 80px 120px 200px;
  grid-auto-flow: column;
}
```



Unidades relativas: fr

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr;
  grid-template-rows: 3fr 2fr 1fr;
}
```

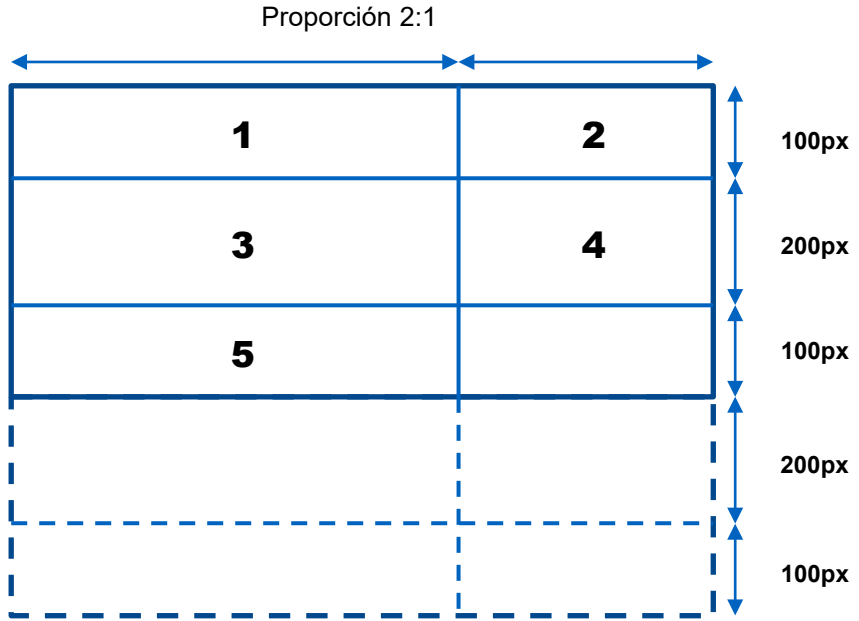


grid-auto-rows

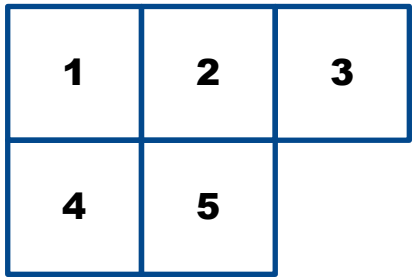
```html
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

```css
.container {
  display: grid;
  grid-template-columns: 2fr 1fr;
  grid-auto-rows: 100px 200px;
}
```

Según se necesite, se agregarán filas siguiendo el mismo patrón



Funciones repeat() y minmax()




```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
</div>
```

Tres columnas de 200px

```
.container {
  display: grid;
  grid-template-columns: repeat(3,
200px);
  grid-auto-rows: 100px;
}
```

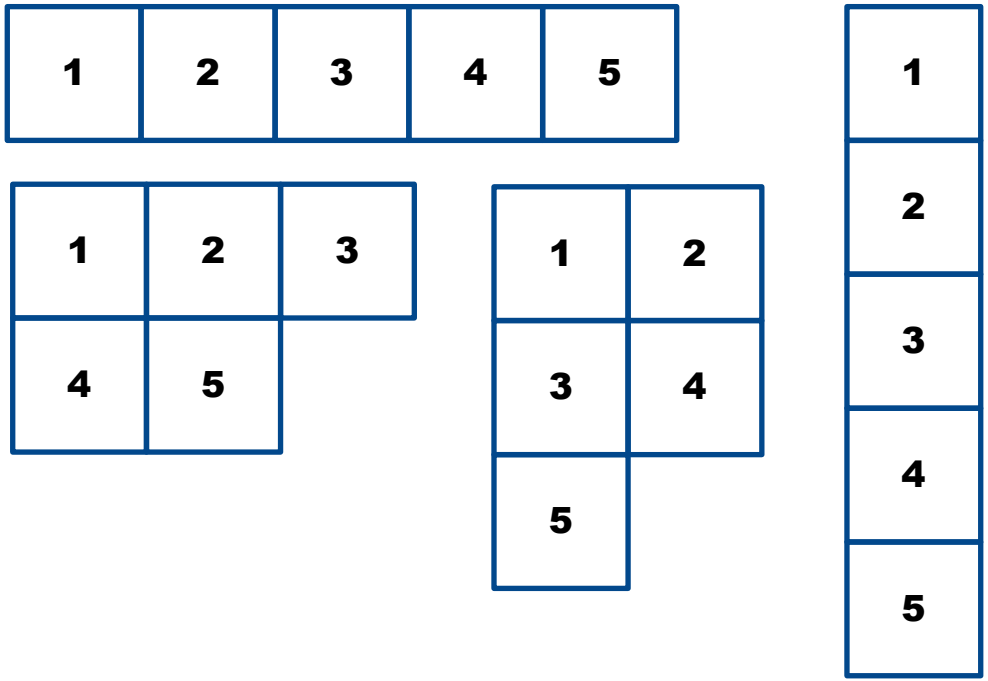
Tres columnas ajustadas al espacio disponible

```
.container {
  display: grid;
  grid-template-columns: repeat(3,
1fr);
  grid-auto-rows: 100px;
}
```

Tres columnas ajustadas al espacio disponible con un mínimo de 200px

```
.container {
  display: grid;
  grid-template-columns: repeat(3, minmax(200px,
1fr));
  grid-auto-rows: 100px;
}
```

auto-fill y auto-fit



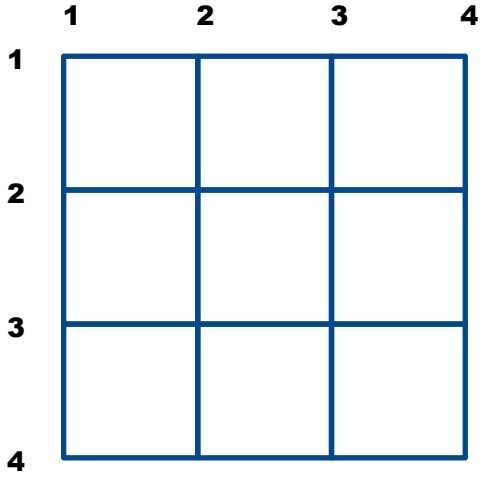
Con `auto-fill` se ajusta cuando no hay espacio suficiente

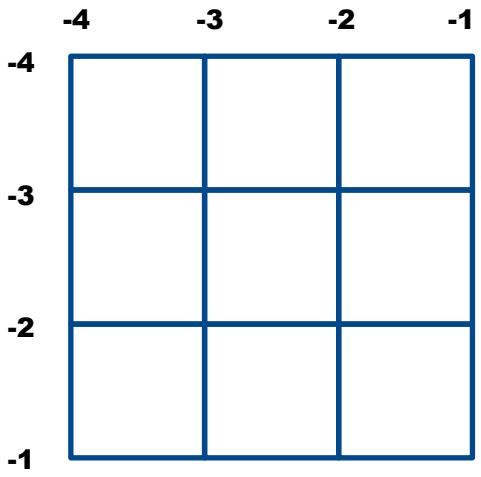
```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(150px,
1fr));
  grid-auto-rows: 150px;
}
```

Con `auto-fit`, además, se ajusta a todo el espacio, incluso cuando es mayor

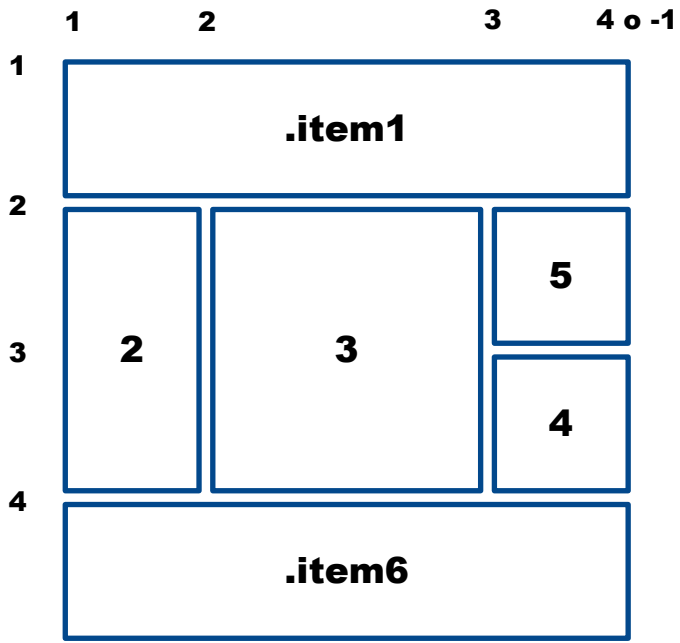
```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(150px,
1fr));
  grid-auto-rows: 150px;
}
```

Cómo se numeran las líneas del grid





Colocar objetos en el grid



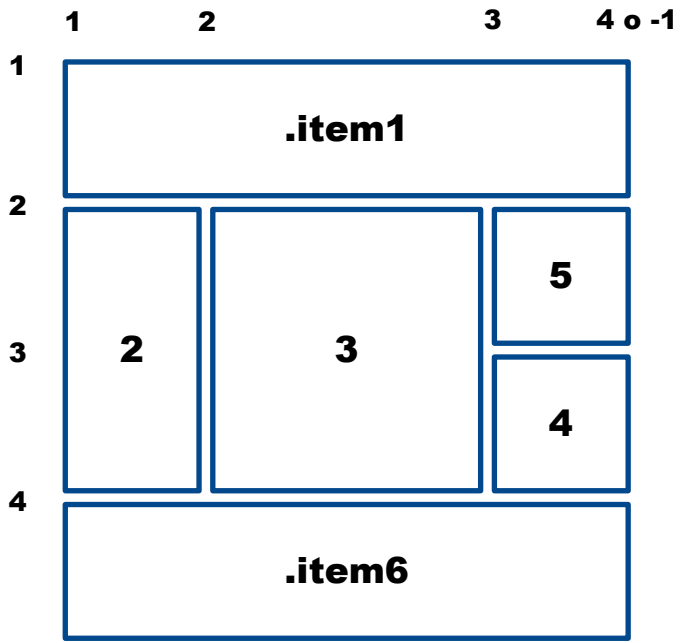

```css
.container {
  display: grid;
  grid-template-columns: 100px minmax(200px, auto) 100px;
  grid-auto-rows: 100px;
  grid-gap: 10px;
}

.item1 {
  grid-column-start: 1;
  grid-column-end: -1;
}

.item6 {
  grid-column: 1 / -1;
}

.item2, .item3 {
  grid-row: span 2;
}

.item5 {
  grid-column: 3;
  grid-row: 2;
}
```

De la primera a la última línea

Ocupa dos filas

Se mueva a la columna 3, fila 2

Nombrar las líneas del grid

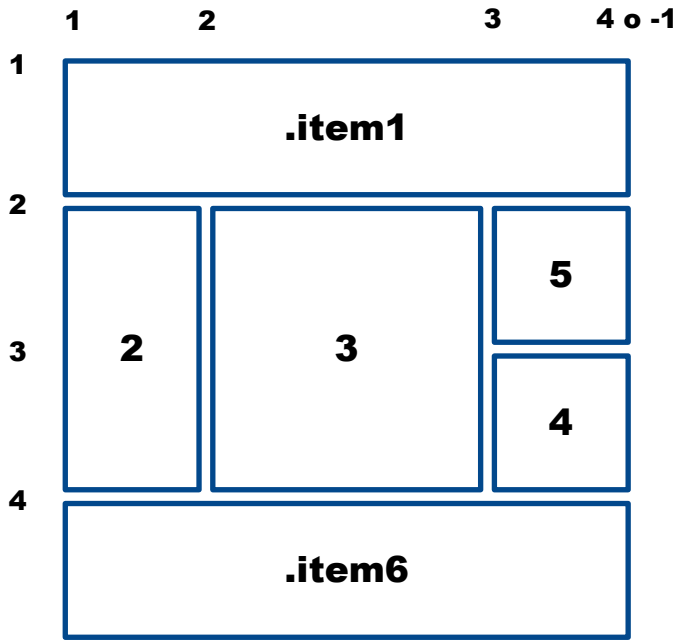



```
.container {
  display: grid;
  grid-template-columns: [grid-start nav-start] 100px
                         [nav-end main-start] minmax(200px, auto)
                         [main-end] 100px
                         [grid-end];
  grid-auto-rows: 100px;
  grid-gap: 10px;
}

.item1 {
  grid-column: grid-start / grid-end;
}

.item6 {
  grid-column: 1 / -1;
}

.item2 {
  grid-column: nav-start / nav-end;
  grid-row: span 2;
}

.item3 {
  grid-column: main-start / main-end;
  grid-row: span 2;
}

.item5 {
  grid-column: 3;
  grid-row: 2;
}
```

Definimos los nombres al definir la plantilla

Los usamos para establecer la posición inicial y final

Definir áreas del grid




```
.container {
  display: grid;
  grid-template-columns: 100px minmax(200px, auto) 100px;
  grid-template-rows: repeat(4, 100px);
  grid-gap: 10px;
  grid-template-areas: "header header header"
                       "nav main side1"
                       "nav main side2"
                       "footer footer footer";
}

.item1 {
  grid-area: header;
}

.item2 {
  grid-area: nav;
}

.item3 {
  grid-area: main;
}

.item4 {
  grid-area: side2;
}

.item5 {
  grid-area: side1;
}

.item6 {
  grid-area: footer;
}
```

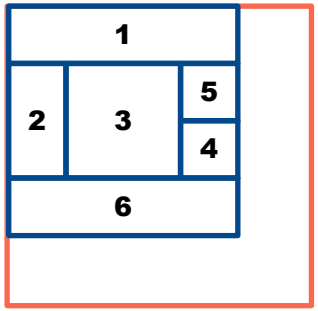
Definimos los nombres de las áreas del grid

Asociamos los elementos a su área usando el nombre

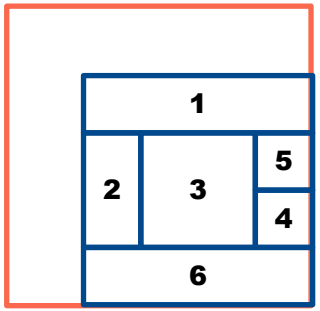
Cómo se calcula la posición de los elementos

1. Se generan los elementos del grid como anónimos
2. Se colocan los elementos con posición explícita
3. Se colocan los elementos con fila explícita, pero sin columna
4. Se calcula el número de columnas implícitas del grid
5. Se coloca el resto de elementos

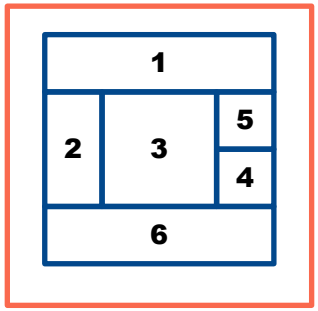
Alineación del grid



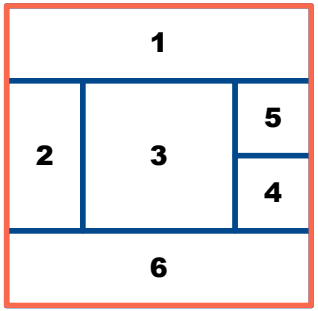

```
justify-content:
start;
align-content: start;
```




```
justify-content: end;
align-content: end;
```

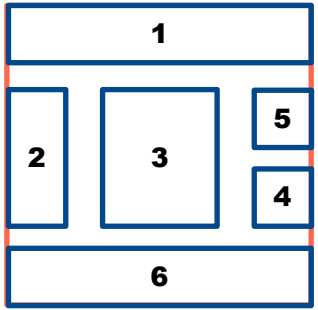



```
justify-content:
center;
align-content: center;
```

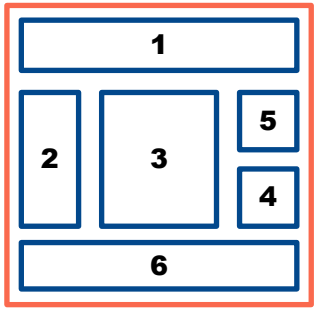



```
justify-content:
stretch;
align-content: stretch;
```

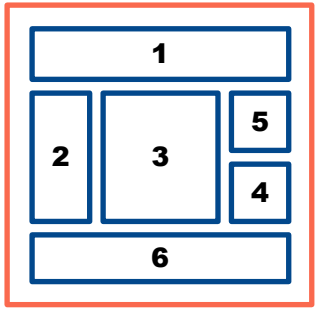
```
justify-content:
            (eje de
columnas);

align-content:
            (eje de
filas);
```




```
justify-content: space-
between;
align-content: space-between;
```




```
justify-content: space-around;
align-content: space-around;
```




```
justify-content: space-
evenly;
align-content: space-evenly;
```

Alineación de elementos

- En el contenedor, se aplica para todos los elementos:
  - `justify-items`: eje de columna ↔
  - `align-items`: eje de fila ↕
  - Valores: `start` | `end` | `center` | `stretch`

- Para un solo elemento:
  - `justify-self`: eje de columna ↔
  - `align-self`: eje de fila ↕
  - Valores: `start` | `end` | `center` | `stretch`

Introducción al diseño responsive

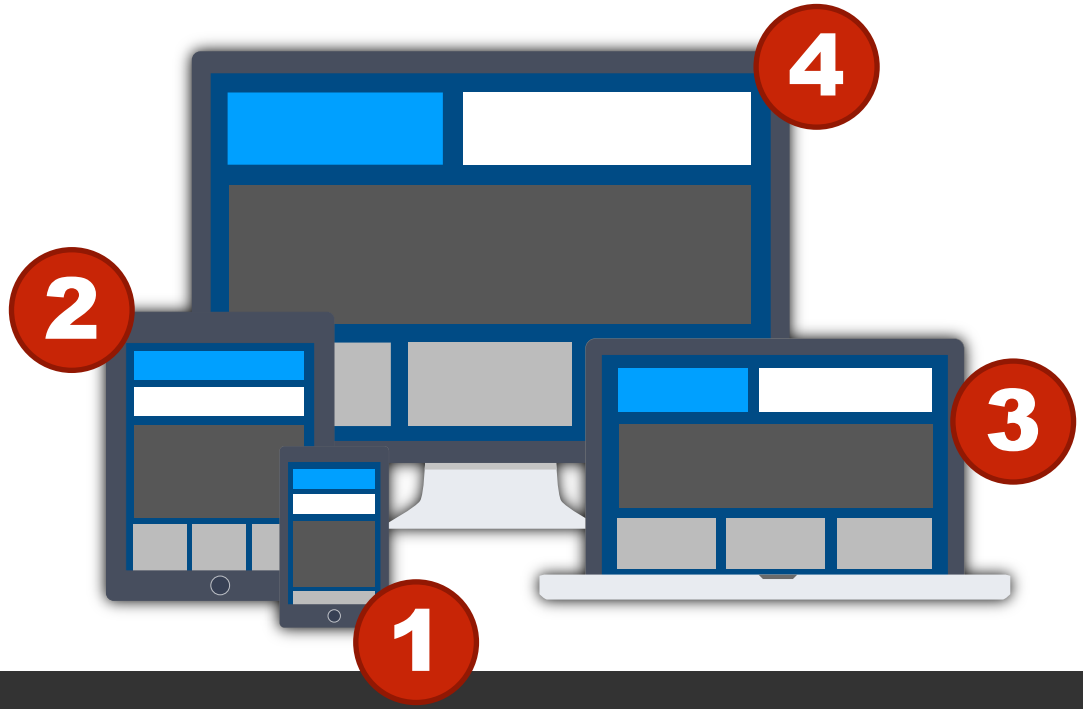
Objetivo

El diseño web responsive pretende:

- Adaptar la apariencia de una página o aplicación web al dispositivo donde se muestra
- Es decir, tener un solo diseño y que éste pueda adaptarse
- Evitar desarrollos ad-hoc para cada dispositivo
- Evitar la complejidad de tener varias versiones de una misma página o aplicación

Mobile First

Consiste en desarrollar primero los componentes para el dispositivo que más restricciones de tamaño tiene.



Especificación del viewport

Mobile First

```
<meta name="viewport“ content="width=device-width,
              initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
```

Define el comportamiento del área visible de una página web (viewport) en un dispositivo.

- `width=device-width`: el ancho de la web es el ancho del dispositivo
- `initial-scale=1.0`: el zoom inicial es 1
- `maximum-scale=1.0`: el máximo zoom es 1 (no hay más zoom)
- `user-scalable=no`: el usuario no puede hacer zoom

Media queries

Las media queries nos permiten establecer puntos de ruptura donde se modifican propiedades de estilos CSS en función de condiciones del dispositivo (como el ancho de la pantalla).

Sintaxis de la media queries

```
@media tipo_de_dispositivo and|not|only (atributo) {
  /* Estilos */
}
```

Tipos de dispositivo:

- `all`: para cualquier tipo de dispositivo.
- `print`: para impresoras.
- `screen`: para pantallas de ordenador, tablet o smartphone.
- `speech`: para dispositivos que leen la pantalla.

Sintaxis de la media queries

```
@media tipo_de_dispositivo and|not|only (atributo) {
  /* Estilos */
}
```

Atributos:

- `max-width`: indica el ancho máximo de área disponible
- `max-device-width`: indica el ancho máximo del dispositivo
- `orientation`: indica la disposición del dispositivo
- Existen muchas más: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Ejemplos

```
/* Pantalla de ordenador, tablet o smartphone dispuesta horizontalmente */
@media screen and (orientation: landscape) { ... }

/* Impresión dispuesta verticalmente, o bien cualquier dispositivo cuyo
   ancho máximo sea 800px */
@media print and (orientation: portrait),  (max-width: 800px) { ... }

/* Solo pantallas de ordenador, tablet o smartphone en color */
@media only screen and (color) { ... }
```

GRACIAS

www.keepcoding.io