





CSS

 Created By	 Carlos Garcia
 Last Edited	@Jun 27, 2020 7:39 PM
 Tags	

Sección 4: Bases de CSS

Sintaxis y selectores

Sintaxis

AT-RULES

Tipos de selectores

Especificidad, herencia y cascada

Resetear estilos por defecto

BEM Organiza mejor tu CSS

Sección 5: Box Model

Introducción al Box Model

Modelo de caja (BOX MODEL)

Margin y Padding

Border

Outline

Sección 6: Position

Conceptos básicos

Position Relative

Position Absolute

Position Fixed

Position Sticky

z-index

Sección 7: Display, pseudoelementos y pseudoclases

Propiedad Display

Pseudoelementos

Pseudoclases I

Pseudoclases II y III

Sección 8: Variables y background

Variables CSS

Background

Sección 9: Textos y tipografías

Textos y tipografías I

Textos y tipografías II

Textos y tipografías III

Sección 10: Listas, tablas, imágenes y clip-path

Listas

Tablas

Imágenes

Object-fit, object-position y filter()

Clip-path

Sección 11: Colores, border-radius, sombras, degradados, overflow y float

Colores

Border-radius

Box-shadow y text-shadow

Degradados I y II

Overflow y float

Sección 12: Flexbox

Fundamentos de flexbox, Flex-direction y flex-wrap

Aliniamiento

flex-grow, flex-shrink, flex-basis, flex-flow y order

Práctica con flexbox

Sección 13: Grid

Fundamentos de grid

Terminología

Ventajas

Propiedades

[grid-column y grid-row](#)
[medidas y repeat\(\)](#)
[implicit grid y explicit grid](#)
[min-max\(\), auto-fill y auto-fit](#)
[Alineamiento y order](#)
[Grid template areas](#)
[Grid lines](#)
[Grid animados](#)
[Shorthands y grid track](#)
[Práctica con grid](#)
[Sección 14: Responsive](#)
[Responsive Web Design](#)
[Columnas Fluidas](#)
[Media queries](#)
SINTAXIS
[Metodologías](#)
[Responsive sin breakpoints ni media queries](#)
[Videos responsive](#)
[Breakpoints](#)

Sección 4: Bases de CSS

Sintaxis y selectores

Formas de incluir CSS al HTML

- **En la cabecera:** `<style> codigo </style>` Si se comparte estilos no es lo más recomendable
- **En línea dentro de etiqueta:** `<p style="codigo css">` normalmente se utiliza con JS cuando hay que cambiar estilos en tiempo real
- **Hoja externa:** `<link rel="stylesheet" href="styles.css">` es la forma más recomendada
- **@import dentro de etiquetas style:** NO SE UTILIZA. `<style> @import url("styles.css"); </style>` si se cae el import no carga la página

Sintaxis

```
selector{  
  propiedad: valor;  
}
```

El conjunto entero de **selector y sus propiedades** se le denomina **REGLA**, no existen **límite** de las propiedades que se pueden agregar.

AT-RULES

- `@media{}`
- `@font-face{}`
- `@keyframes{}`
- `@import ()`

Tipos de selectores

- **BÁSICOS**
 - **ETIQUETA:** `p, body, div`
 - **CLASS:** `'.{nombre de la clase} { propiedades; }`
 - **ID:** `#'` No se recomienda utilizar para dar estilos, se utiliza con JS para obtener elementos o con HTML para los saltos `#{id} { propiedades; }`
 - **GENERAL:** `'*'` selecciona todas las etiquetas

- **ATRIBUTO:** `{nombre atributo}{ propiedades; }` selecciona los elementos que tienen ese atributo.
 - **selección de elemento con un atributo al menos 1 vez:** `class ~="{nombre clase}" { propiedades; }`
 - **selección de elemento que tiene ese atributo y exactamente ese valor o empieza por el valor seguido de un guión:** `class |= "{nombre clase}" { propiedades; }`
 - **selección elemento que inicie por el atributo:** `class ^="{nombre clase}" { propiedades; }`
 - **selección elemento que termina por el atributo:** `class $="{nombre clase}" { propiedades; }`
 - **selección elemento que contenga ese valor:** `class *="{nombre clase}" { propiedades; }`
- **COMBINADORES**
 - **Hermano adyacente:** selecciona la etiqueta h2 hermana consecutiva del h1 → `{etiqueta1} + {etiqueta2} { propiedades; }` → `h1 + h2 { color: blue; }`
 - **Hermano general:** Se aplica a todos los hermanos → `{etiqueta1} ~ {etiqueta2} { propiedades; }` → `h1 ~ h2 { color: blue; }`
 - **Seleccionar descendentes:** aplica estilo a todos los hijos que están por debajo de etiqueta 1 `{etiqueta1} {etiqueta2} { propiedades; }` → `div p { color: blue; }`
 - **Hijo directo:** Dar estilo al hijo directo de etiqueta1 `{etiqueta1} > {etiqueta2} { propiedades; }` → `div > p { color: blue; }`

NOTA: En CSS no se puede subir de categoría, solo puede seleccionar por el mismo nivel o más bajo.

- PSEUDO-ELEMENTOS
- PSEUDO-CLASES

Especificidad, herencia y cascada

- **HERENCIA:** Se aplica con el valor **inherit**, obliga al elemento a heredar la propiedad de su elemento más cercano
 - `p { color: blue; } a { color: inherit }`
- **CASCADA:** Orden en el que se aplican los estilos al ir leyendo el CSS. Los estilos que **vienen después sobrescriben a los anteriores.**
- **ESPECIFICIDAD:** Peso que tiene un selector cuando hay conflicto de estilos.
 - Etiqueta → 0,0,0,1
 - Clases y pseudoclases → 0,0,1,0
 - ID → 0,1,0,0
 - Estilos en línea → 1,0,0,0
 - !important → GANA A TODOS

Resetear estilos por defecto

Los navegadores tienen sus plantillas para utilizar normalizar para resolver ese problema

Normalize.css

Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing.

🔗 <https://necolas.github.io/normalize.css/>

BEM Organiza mejor tu CSS

Consejos para estructurar nuestros clases de una forma sencilla, escalable y reutilizable

Block, Element, Modifier (BEM)

- **Block:** Cualquier elemento autónomo y aislado de nuestro documento.
 - Se denominan con 1 palabra o 2 separadas por guión **.menu / .main-menu**

- **Elements:** Cada uno de los elementos del bloque
 - Se nombran con el nombre del bloque al que pertenecen y su nombre con dos guiones bajos `.menu__item` `/.main-menu__link`
- **Modifier:** las propiedades, color de texto, tamaño de la fuente, etc
 - se nombran con el nombre bloque que pertenecen y su modificador con dos guiones por medio `.menu__item—active`

Sección 5: Box Model

Introducción al Box Model

En la web TODO son cajas.

LAYOUT es el conjunto de cajas y la forma en que el navegador las pinta

Propiedades principales: width y height

Elementos en HTML: inline y block estas propiedades se pueden modificar con el atributo display

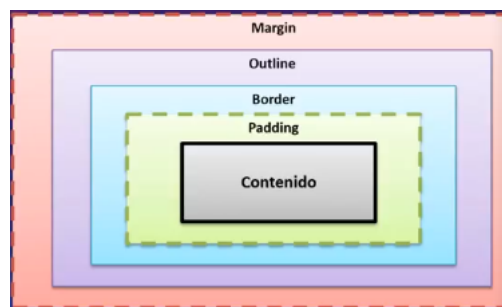
Elementos en línea (inline)

- solo ocupan su contenido
- no puede modificar ni su ancho ni su alto

Elementos en bloque (block)

- Ocupan todo el ancho disponible
- Se puede modificar ancho y alto

Modelo de caja (BOX MODEL)



Margin y Padding

- **Margin:** Separación entre 1 caja y las cajas adyacentes. Es un shorthand, es decir, tiene top, right, bottom y left incluidos
 - Margin-top
 - Margin-right
 - Margin-bottom
 - Margin-left
- **Padding:** Separación entre el CONTENIDO y su BORDE. Es un shorthand
 - Padding-top
 - Padding-right
 - Padding-bottom
 - Padding-left

Border

Es la línea que rodea la caja. Es un shorthand

ANCHO ESTILO COLOR: border: 5px solid red;

- **Propiedades**
 - width
 - style
 - color
- **Combinaciones:**
 - border-width: 2px; (**top right bottom left**)
 - border-style: solid dotted; (**X,Y**)
 - border-color: red blue green; (**top X bottom**)
 - border-width: 2px 3px 4px 5px (**top right bottom left**)

Outline

Es la línea que rodea a la caja entre el **borde** y el **margin**. Es un shorthand. Permite valores negativos.

- **Propiedades**
 - width
 - style
 - color
 - Offset

Mismas combinaciones que con border

Sección 6: Postion

Conceptos básicos

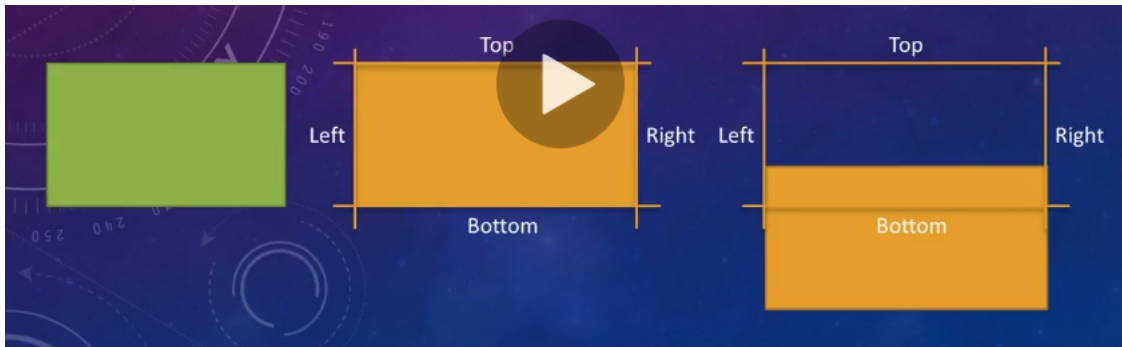
- **Flujo:** Es el orden de como se dibuja los elementos. Si se coloca un H1 y luego un P, primero se dibuja el H1 y luego el P
- **Valores de position:**
 - **Static:** Default. No está posicionado. Debe utilizar alguno de los otros para poder decir que el elemento está posicionado.
 - **Relative**
 - **Absolute**
 - **Fixed**
 - **Sticky**
- **Si el elemento ESTA POSICIONADO se puede mover en 3 ejes: Propiedades →**
 - **TOP:**
 - **RIGHT:**
 - **BOTTOM:**
 - **LEFT:**
 - **Z-INDEX:** mover elemento en z (si hay objetos solapados podemos indicar cual estará encima del otro)

NOTA: Hay **valores de position** que hace que el elemento ocupe el espacio o no lo ocupe. Es decir, hay unos que reservan el espacio aunque no se vea el elemento y otros que no lo reservan.

Position Relative

- Coloca el elemento **respecto a su posición en el flujo** (visualmente no pasa nada, por detrás si)

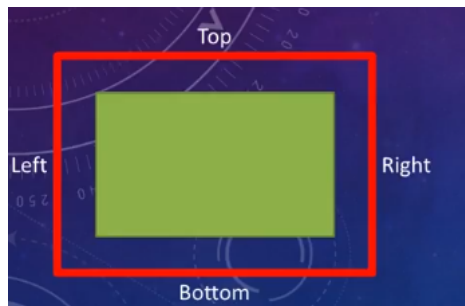
- Se podrá mover elemento se incorporan las propiedades **top, right bottom, left y z-index**
- El elemento **conserva su espacio reservado**
- Punto de referencia **no se modificará** aunque **se mueva**



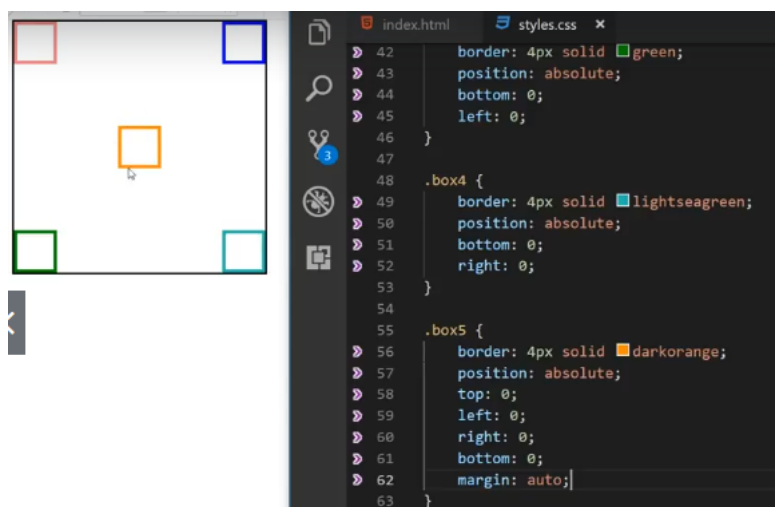
NOTA: Las propiedades aceptan **valores negativos**. **TOP y LEFT** vales más que **RIGHT y BOTTOM**.

Position Absolute

- Colocar el elemento con respecto a **su contenedor posicionado más cercano**, si no encuentra ninguno será el **viewport** (el body)
- No conserva el **espacio en el flujo (desaparecen del flujo HTML)**
- Si no tiene **dimensiones declaradas**, sus dimensiones se ajustan al contenido
- Punto de referencia no se modificará



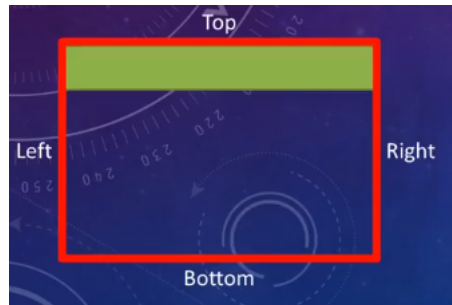
NOTA: como poner un elemento en el medio. Combinación de **position: relative** y **position: absolute**



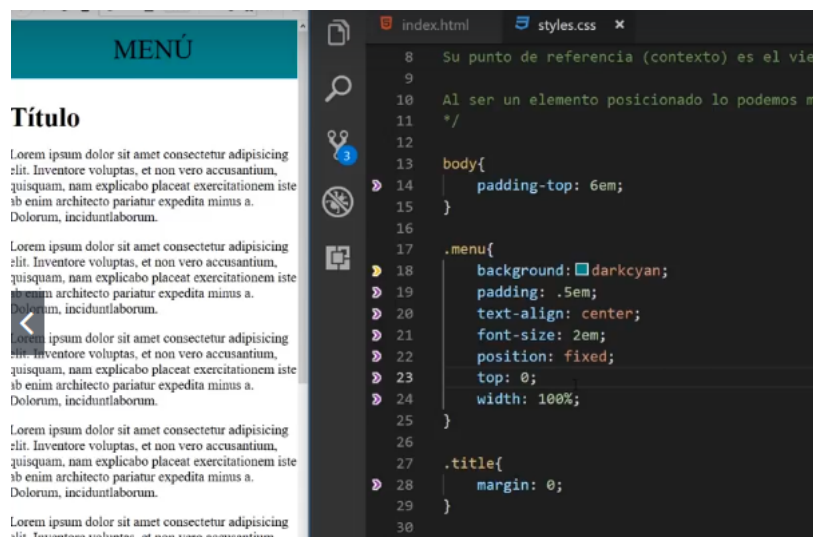
Position Fixed

- El elemento se coloca **respecto al viewport**
- No conserva el espacio en el flujo
- Si no tiene dimensiones entonces se ajustan al contenido
- No se modifica el punto de referencia
- Cuando **hacemos scroll** el elemento **no se moverá**. Estará fijo

Un menú de página



NOTA: Se utiliza position: fixed y top: 0px en conjunto de un padding en el body para cubrir el espacio y que no se sobre ponga a otros elementos



Position Sticky

- Mezcla entre **relative y fixed**
- Conserva el espacio en el flujo
- Si no tiene dimensiones declaradas se ajustara al contenido
- Su punto de referencia no se modificará al mover el elemento
- Cuando hacemos scroll el elemento se moverá hasta alcanzar el tope establecido
- Ej: position: sticky y con top:0 así indicamos que se comportará como fixed a partir de esa posición, si no, se comporta como relative

z-index

- Modifica el orden de las capas en el eje z (**profundidad**)
- Admite valores + y -
- RECOMENDACION: no utilizar valores consecutivos (100,200,..)

- SI EL PADRE TIENE Z-INDEX → EL HIJO NO PODRÁ ESTAR POR DEBAJO

NOTA: Colocar un hijo debajo el padre. El padre no debe tener z-index y el el hijo tenerlo con un valor negativo

Sección 7: Display, pseudoelementos y pseudoclases

Propiedad Display

- Nos permite definir cómo se comporta una caja respecto a las otras cajas adyacentes
- Elementos tipo **línea (ocupan lo que ocupa el contenido)** y **bloques (ocupan el 100%)**
- Valores
 - **inline**
 - **block**
 - **inline-block:** Comportamiento de tipo línea pero acepta **width y height**
 - **none:** Oculta el elemento pero se sigue renderizando. No tiene espacio en el flujo. (**recomienda utilizar dinamicamente con js**)
 - **table:** Imita comportamiento de tabl
 - **list-item:** imita una lista

NOTA: A los elementos tipo línea solo les afecta los **margin** y **padding horizontal**.

Los elementos inline no afecta el flujo en vertical **solo en horizontal**

Pseudoelementos

- Dar estilo a partes específicas de un elemento
- sintaxis: **{selector}::pseudo-elemento{ estilos }**
- Pseudoelementos:
 - **::first-line (elementos de bloque)**
 - **::first-letter (elementos de bloque)**
 - **::before**
 - **::after**
 - la propiedad **content** es obligatoria
 - son elementos de linea por defecto
 - son hijos del elemento al que pertenecen
 - **{selector}::before{ content: " ; }**
 - **::selection**

Pseudoclases I

- Son selectores que reaccinoan en tiempo real detectando la interacción del usuario con algunos elementos
- Funcionan con todos los selectores de css
- No están atados a ningún elemento
- Sintaxis:
 - **Selector:pseudoclase{ estilos }**
 - **:pseudoclase{ estilos }**
- Pseudoclases **Dinamic:**
 - **:link** → Links no visitados (**solo a links**)
 - **:visited** → Links visitados (**solo a links**)

- **:hover** → Poner mouse sobre elemento
- **:active** → (links) al darle click (**solo a links**)
- **:focus** → Cuando el elemento esta seleccionado
- **:target** → elemento que esta con target a través de un marcador
- **:lang({idioma})** → agregar estilos a elementos con atributo lang = "en"
- **UI States:**
 - **:enable** → todos los campos por default
 - **:disabled** →
 - **:checked** →
 - **:in-range** →
 - **:out-of-range** →
 - **:required** →
 - **:optional** → todos los campos por default
 - **:valid** →
 - **:invalid** →
 - **:read-only** →
 - **:read-write** →
 - **:fullscreen** → no pertenece a form

Pseudoclasas II y III

- **Estructural**
 - **:root** → representa la raiz del documento (html) usando una pseudoclase tenemos más especificidad que si usáramos html (etiqueta). **Se utiliza para declarar variables**
 - **:empty** → Se aplica para los elementos vacios, es decir, etiquetas que no tienen contenido
 - **Propiedad all** → **Reseta todas las propiedades menos direction y unicode-bidi**. Valores posibles (**inherit, initial, unset**)
 - **-child** - No importa el tipo de etiqueta
 - **:first-child** → Selecciona primer hijo.
 - **:last-child** → Selecciona ultimo hijo
 - **:nth-child(n)** → Selecciona al hijo n
 - **:nth-last-child(n)** → Selecciona al hijo n contando desde el final
 - **:only-child** → Selecciona al hijo único
 - **-type** - Si importa el tipo de etiqueta
 - **:first-of-type** →
 - **:last-of-type:nth-of-type(n)** →
 - **:nth-last-of-type(n)** →
 - **:only-of-type** →
- **VALORES**
 - **odd** → Impares
 - **even** → Pares
 - **numeros enteros** →
 - **ecuaciones** → $1n$, $3n+1$,...

- **Negacion**

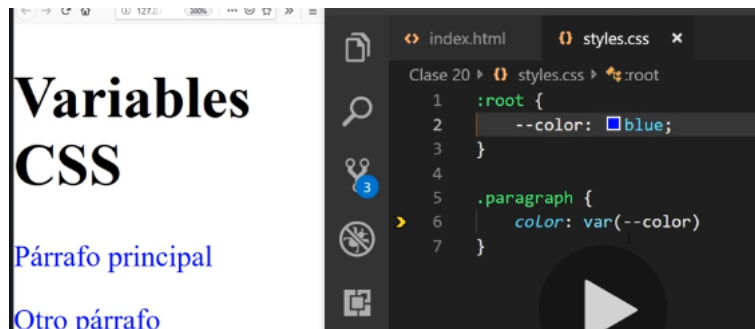
- **:not()** → Selecciona al elemento que no cumpla la condición , se puede colocar cualquier selector menos otro not

Sección 8: Variables y background

Variables CSS

Se puede escribir código en responsive web design

- **Variable:** ocupan memoria
- Deben estar dentro de un selector
- Tienen herencia y cascada
- Hay globales y locales, depende del selector donde lo declaremos
- las variables CSS **no son iguales a** variables SCSS
- sintaxis: **selector{ --{var}: valor }**
- utilizar: **propiedad: var(--{var})**



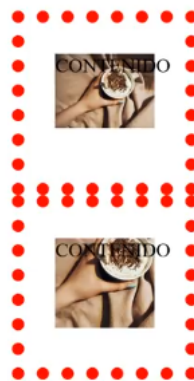
Background

- Dar fondos a las cajas
- es un **shorthand**
 - **background-color:** color de fondo
 - **background-image:**
 - **background-repeat:**
 - valores: no-repeat | repeat | repeat-x | repeat-y
 - **background-position:**
 - left top | left center | left bottom
 - right top | right center | right bottom
 - center top | center center | center bottom
 - x% y%
 - xpos ypos
 - initial
 - inherit (heredado)
 - **background-size:**

- dos valores: eje x y (opcional lo calcula automáticamente en dicho caso) | auto | cover (todo pero se escapa del box) | contain (ajusto al contenido)
- **background-origin:** donde comienza.. Redimensiona para que las cosas quepan
 - **border-box**
 - **padding-box**
 - **content-box**
- **background-clip:** en que parte del box se dibuja el fondo **NO REDIMENSIONA**
 - **border-box:** borde de caja
 - **padding-box:** contempla hasta el borde
 - **content-box:** contempla el contenido

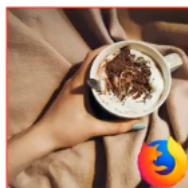
Diferencia entre origin **(1er box)** y clip **(2do box)**

Background



- **background-attachment:** establecer si el background es relativo a la caja o al viewport
 - scroll: relativo a la caja
 - fixed: relativo al viewport
- FONDOS MULTIPLES

Background



```

Clase 22 ▶ styles.css ▶ .bg-multiple
90 background-image: url(cafe.jpeg);
91 background-size: contain;
92 background-repeat: no-repeat;
93 padding: 25px;
94 }
95
96 .bg-attachment {
97 width: 100%;
98 height: 400px;
99 background-image: url(cafe.jpeg);
100 background-size: contain;
101 background-attachment: fixed;
102 }
103
104 .bg-multiple {
105 background-image:
106   url(Logo.png),
107   url(cafe.jpeg);
108 background-size: 30%, cover;
109 background-repeat: no-repeat;
110 background-position: bottom right, center;
111 }
  
```

Sección 9: Textos y tipografías

Textos y tipografías I

- fuentes o tipografía hacen referencia al tipo de letra
- grupos
 - **familias tipográficas:** Tienen nombre (Arial,etc)
 - **Familias genéricas:** Según características (**serif,sans-serif-cursive...**) **TODOS LOS NAVEGADORES LAS CONTIENEN**
- Al elegir una fuente también hay que seleccionar una de reserva en caso de no existir elija.
- Propiedades para cambiar fuentes
 - **font-family:**
 - **font-size:**
 - Relativas al documento
 - **px:** Medida absoluta
 - **em:** media relativa al contexto (al padre)
 - **rem:** media relativa al HTML
 - **%:** media relativa al tamaño actual
 - Relativas al tamaño de la ventana (**viewport**)
 - **vh:** media relativa al height del viewport
 - **vw:** media relativa al width del viewport
 - **vmin:** media relativa al valor minimo de viewpor
 - **vmax:** media relativa al valor maximo del viewport
 - **font-weight:**
 - **font-style:**

Textos y tipografías II

- Propiedades para cambiar texto
 - **text-transform:**
 - **uppercase**
 - **lowercase**
 - **capitalize**
 - **text-align:** FUNCIONA CON ELEMENTOS DE BLOQUES
 - **center**
 - **left**
 - **right**
 - **justify:** no se utiliza mucho en web
 - **text-decoration:**
 - **overline**
 - **underline**
 - **line-through**
 - **none**
 - **text-indent:** SE UTILIZA EN ELEMENTO DE TIPO BLOQUE
- Otras Propiedades
 - **line-height:**

- **letter-spacing:**

Textos y tipografías III

Google Fonts

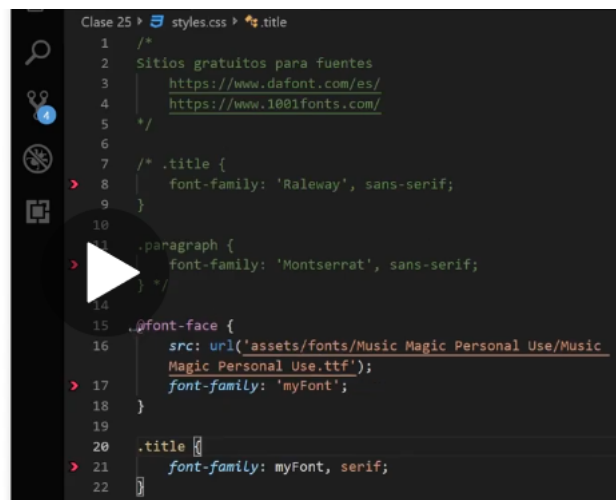
Making the web more beautiful, fast, and open through great typography

<https://fonts.google.com/>

Google Fonts

NOTA: Cuando se quiere guardar información, es decir, archivos externos como videos, imagenes, fuentes, etc, se **recomienda agregarlos en la carpeta ASSETS**

- @font-face: permite agregar una fuente que no esté en google fonts

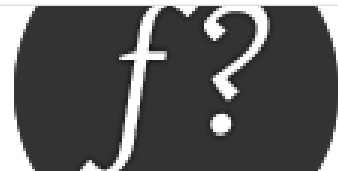


Extensión que permite saber que tipo de letra está utilizando una página web (**chrome y firefox**)

WhatFont

The easiest way to identify fonts on web pages.

https://chrome.google.com/webstore/detail/whatfont/jabopobgcpjmedljpbcaablpmlmfcogm?hl=es_419



Sección 10: Listas, tablas, imágenes y clip-path

Listas

Propiedades que hay que resetear que vienen por defecto

- margin-top: 0;
- margin-bottom: 0;
- padding-left: 0;
- list-style: none;

Hay 3 propiedades para las listas, se pueden aplicar al padre o hijos

- **list-style-type:** Establece el estilo de vineta
 - **disc:** circulo (default)
 - **circle:** circulo vacio
 - **square:**

- **decimal:** hace que se comporte como ol
- **decimal-leading-zero:** lo mismo anterior pero con 0 adelante
- **lower-roman:**
- **upper-roman:**
- **lower-greek:**
- **lower-latin:**
- **upper-latin:**
- **armenian:**
- **georgian:**
- **lower-alpha:**
- **upper-alpha:**
- **list-style-position:** Hace que las vinetas se coloquen por dentro o por fuera del elemento (**val: inside, outside**)
- **list-style-image:** Pone una imagen en lugar de la vineta, no se suele utilizar porque hay formas mucho mas eficientes de hacerlo **url(imagen)**
 - Mejor forma #1

✓ Item 1
✓ Item 2
✓ Item 3
✓ Item 4
✓ Item 5

```

Clase 26 > styles.css > .list_item::before
35 | list-style-type: lower-latin;
36 | */
37 |
38 | /* .list {
39 |   list-style-image: url(check.png)
40 | */
41 |
42 | .list{
43 |   list-style-type: none;
44 | }
45 |
46 | .list_item::before {
47 |   content: '';
48 |   display: inline-block;
49 |   width: 1.5em;
50 |   height: 1.5em;
51 |   background-image: url(check.png);
52 |   background-position: center;
53 |   background-size: contain;
54 |   background-repeat: no-repeat;
55 |   padding-right: 1em;
56 | }

```

- Mejor forma #2

✓ Item 1
✓ Item 2
✓ Item 3
✓ Item 4
✓ Item 5

```

Clase 26 > styles.css > .list_item
40 | */
41 |
42 | .list {
43 |   list-style-type: none;
44 | }
45 |
46 | /* .list_item::before {
47 |   content: '';
48 |   display: inline-block;
49 |   width: 1em;
50 |   height: 1em;
51 |   background-image: url(check.png);
52 |   background-position: center;
53 |   background-size: contain;
54 |   background-repeat: no-repeat;
55 |   padding-right: 1em;
56 | } */
57 |
58 | .list_item {
59 |   background-image: url(check.png);
60 |   background-position: 0 0;
61 |   background-size: contain;
62 |   background-repeat: no-repeat;
63 |   padding-left: 2em;
64 | }

```

Tablas

Las tablas tienen su propio layout o formato entonces no se puede utilizar como en CSS normal para darle estilo

Propiedades

- **table-layout:** Define como se comportan las dimensiones de una tabla y los anchos de las "columnas"
 - **automatic:** default
 - **fixed:** necesita un width declarado, si no se le da un ancho a cada columna se distribuyen equitativamente
- **caption-side:** Define donde se coloca el caption de una tabla
 - **top:** default
 - **bottom:** se coloca a pie de la tabla
- **border-collapse :** Controla si las celdas se mantienen juntas o separadas
 - **separate:** default
 - **collapse:**
- **border-spacing:** Controla el espacio entre las celdas recibe una medida en cualquier unidad
- **empty-cells:** Controla que hacer con las celdas vacías
 - **show:** default
 - **hide:** oculta las celdas

Imágenes

- Imágenes responsive: En el archivo base se recomienda poner esta regla
 - **img{ max-width: 100% }**
- Las imágenes por defecto son elementos inline, esto causa un **espacio por debajo** debido a su **line-height**, se puede solucionar de dos maneras
 - Dando un **line-height: 0** al contenedor o
 - **display:block** a la imagen
- Centrado horizontal:
 - imagen con **display:block** y márgenes laterales automáticos
 - **text-align: center**, si la imagen no es de bloque
- Centrado vertical
 - **Flexbox (mejor opción)**
 - **vertical-align: middle.** A la imagen y texto

Object-fit, object-position y filter()

- **object-fit:** se usa para especificar cómo se debe cambiar el tamaño de o <video> para que se ajuste a su contenedor
 - **fill:** valor por defecto
 - **contain:** El contenido se ajustará hasta rellenar de forma horizontal o vertical el contenedor sin deformarse
 - **cover:** El contenido se ajustará hasta rellenar de forma horizontal y vertical el contenedor sin deformarse
 - **none:** El contenido no se redimensiona y mantiene su tamaño original mostrando solo el trozo de las dimensiones
 - **scale-down:** Selecciona el menor de la comparación entre none y contain.
- **object-position:** Coloca la imagen cuando esta no se muestra completa en el contenedor
- **filter:**
 - **none**

- **blur(px)**: desenfoca la imagen
- **brightness(%)**: Ajusta el brillo de la imagen siendo 1 el original. De 0 a 1 dan oscuridad y de 1 a n dan sobreexposición
- **contrast(%)**: Ajusta el contraste de la imagen siendo 1 el original. De 0 es negro y de 1 a n dan más contraste
- **drop-shadow(h-shadow v-shadow blur (spread) color)**: Aplica una sombra paralela a la imagen
- **grayscale(%)**: Convierte la imagen a escala de grises, 0 es el original y 1 sería blanco y negro completamente
- **hue-rotate(deg)**: Añade matiz de color a la imagen. Se da un valor en grados según la rueda cromática. El valor máximo es 360
- **invert(%)**: Invierte el color de la imagen. Saca un negativo. 0 es el valor por defecto y 1 es totalmente invertido
- **opacity(%)**: Controla la opacidad de la imagen. 1 es el valor por defecto y 0 es transparente
- **saturate(%)**: Controla la saturación de color de la imagen. 1 es el valor por defecto, 0 es totalmente desaturado
- **sepia(%)**: Aplica un tono sepia a la imagen. 0 es el valor por defecto y 1 es totalmente sepia
- **url()** - Buscar información

Clip-path

- **clip-path**: Es una máscara que oculta partes de una caja
 - **circle()**: dibuja un círculo, puede tener una medida fija, circle(100px) o podemos especificar un centro con at, c
- **ellipse()**: dibuja una elipse, funciona igual que el círculo, pero en este caso tenemos que especificar 2 centros, el h
- **inset()**: dibuja un borde transparente por dentro de la caja


```
inset(all | Y X | top X bottom | top left bottom right)
```

existe la opción de redondear las esquinas, para ello después de los valores de inset pondremos round

```
inset(*** round all |
```

```
top-left & bottom-right top-right & bottom-left |
```

```
top-left top-right & bottom-left bottom-right
```

```
top-left top-right bottom-right bottom-left
```

```
)
```
- **polygon()**: Especifica una serie de puntos (mínimo 3) para definir la zona visible siendo el punto 0 0 la esquina superior izquierda

los puntos se especifican por parejas de ejeX y ejeY separados por comas

```
polygon(0 0, 100% 0, 0 100%)
```
- **Generador de clip-path**: <https://bennettfeely.com/clippy/>

Sección 11: Colores, border-radius, sombras, degradados, overflow y float

Colores

Hay varias formas de dar color a los elementos

- **Palabras clave** https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

- **RGB**: es una función de color que recibe 3 valores separados por comas, siendo 0 el mínimo y 255 el máximo
 rgb(0,0,0) sería un negro y rgb(255,255,255) sería blanco.

RGBA: Es una variación de RGB con un cuarto valor que sería el canal alpha, con este valor controlamos la opacidad. Ej: rgba(0,0,0,0.5)

- **Hexadecimal**: La notación hexadecimal tiene 16 valores, 0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F.

Se escriben con un # al inicio y se pueden usar 3 o 6 valores. Si se usan 6 valores deben ir en parejas #ffffff, pero también #fff.

El código hexadecimal se representa con los canales rgb de esta forma #rgb o #rrggbb. Utilizar el modo hexadecimal

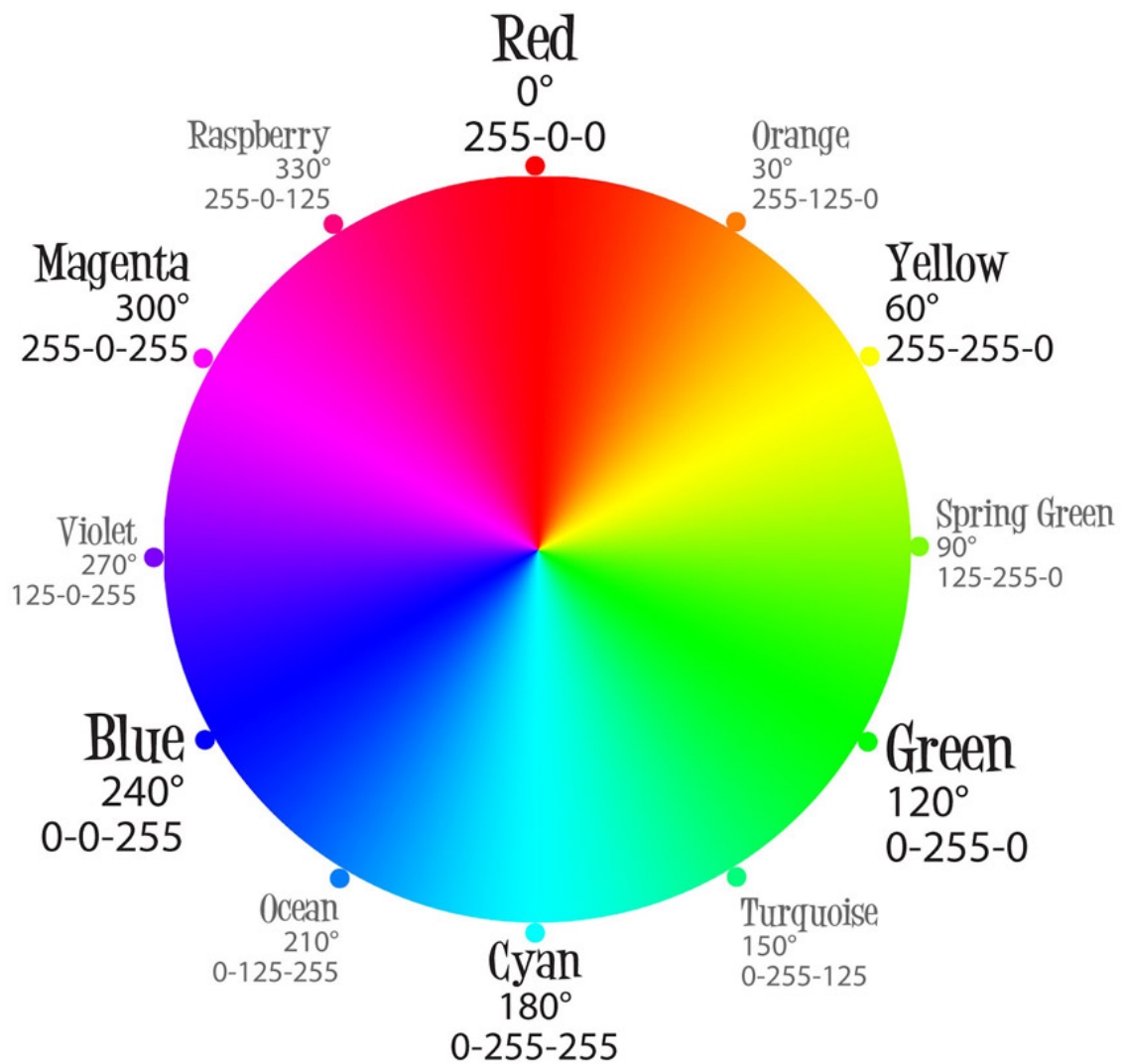
Cuando los 3/6 valores son iguales es un color neutro, #000 sería negro y #fff blanco, y todos los valores intermedios

HSL(hue, saturation, lightness): Es una función de color que nos permite controlar el tono, la saturación y la luminosidad

hue es el ángulo en la rueda cromática. Los valores van de 0 a 360 grados
saturation es la intensidad del color. Los valores van de 0 (gris) a 100% (color puro)
lightness es la intensidad de la luz. Los valores van de 0 (negro) a 100% (blanco)

HSLA: Exactamente igual que RGBA

Los grados representan los colores, se utiliza con HSL



Border-radius

Border radius

Es un shorthand que engloba:

- **border-top-left-radius**
- **border-top-right-radius**
- **border-bottom-right-radius**
- **border-bottom-left-radius**
- **border-radius: all;**
- **border-radius: top-left/bottom-right top-right/bottom-left;**
- **border-radius: top-left top-right/bottom-left bottom-right;**

- **border-radius: top-left top-right bottom-left bottom-right;**
- **Elipses**
 - border-radius: 10px / 50px;
border-top-left-radius: 10px 50px;
border-top-right-radius: 10px 50px;
border-bottom-right-radius: 10px 50px;
border-bottom-left-radius: 10px 50px;
 - border-radius: 10px 50px / 50px 100px;
border-top-left-radius: 10px 50px;
border-top-right-radius: 50px 100px;
border-bottom-right-radius: 10px 50px;
border-bottom-left-radius: 50px 100px;
 - border-radius: 10px 50px 60px / 50px 100px 200px;
border-top-left-radius: 10px 50px;
border-top-right-radius: 50px 100px;
border-bottom-right-radius: 60px 200px;
border-bottom-left-radius: 50px 100px;
 - border-radius: 10px 50px 60px 80px / 50px 100px 200px 300px;
border-top-left-radius: 10px 50px;
border-top-right-radius: 50px 100px;
border-bottom-right-radius: 60px 200px;
border-bottom-left-radius: 80px 300px;

Box-shadow y text-shadow

box-shadow

Es una propiedad que crea una una sombra del tamaño de la caja

box-shdow: h-offset v-offset blur spread color inset|outset

h-offset: Es el desplazamiento horizontal de la sombra

v-offset: Es el desplazamiento vertical de la sombra

blur: Optional Es la cantidad de desenfoque de la sombra

spread: Optional Es la extensión de la sombra

color: Optional color de la sombra, si no se especifica tomará el color del texto

inset|outset: Optional Determina si la sombra se dibuja por dentro o por fuera de la caja

Con valor outset(default) los valores positivos añaden a derecha y abajo y los valores negativos añaden a izquierda

Con valor inset se invierten los valores, positivo sería izquierda y arriba y negativo sería derecha y abajo

box-shadow: 10px 10px 2px 5px red;

Sombras múltiples.

Se añaden las sombras separadas por comas

box-shadow: 10px 10px 2px 5px red, -10px -10px 2px 5px blue;

text-shadow: h-offset v-offset blur color

Sombras múltiples.

Se añaden las sombras separadas por comas text-shadow: 10px 10px 2px red, -11px -11px 2px blue;

Degradados I y II

radial-gradient()

La función CSS radial-gradient() crea una imagen (image) que representa un gradiente (degradado) de colores, generando un radio desde un origen, el centro (center) del gradiente. El resultado de esta función es un objeto de tipo de dato CSS gradient.

<https://developer.mozilla.org/es/docs/Web/CSS/radial-gradient>



Un degradado es una transición entre un color y otro. El navegador calculará todos los pasos intermedios entre los colores del degradado

Es un valor de background-image

Existen dos tipos de degradados: lineales y radiales

- Degradados lineales

linear-gradient([direction], color-1, color-2....)

background-image: linear-gradient(red, blue);

La dirección es opcional, se puede establecer con un ángulo(20deg, 190deg...) o estableciendo la dirección del deg

background-image: linear-gradient(to right, red, blue);

Si no establecemos paradas de color, el navegador dividirá el espacio disponible entre los colores que tenga que p

2 colores 0% 100%

3 colores 0% 50% 100%

Para establecer las paradas se puede usar cualquier medida, px, em, %...

si las paradas/inicios empiezan en el mismo sitio se genera un corte sólido

background-image: linear-gradient(red 50%, blue 50%);

linear-gradient(direction, color-1 stop, color-2 start [stop]...)

background-image: linear-gradient(red 50%, blue 50% 70%, green 75%);

- Degradados radiales

Funcionan de una forma similar a los degradados lineales.

background-image: radial-gradient([shape], red, blue);

background-image: radial-gradient(red, blue);

Por defecto, shape se ajustará al tamaño de la caja, pero podemos establecer si queremos un círculo o una elipse .

background-image: radial-gradient(circle 100px, red, blue);

background-image: radial-gradient(ellipse 100px 50px, red, blue);

Si no establecemos un punto de origen, el punto 0 0 será el centro del elemento. Para establecer el punto de orige

Con la palabra "at" establecemos el punto de origen. Los valores que acepta son:

top | right | bottom | left | center | closest-side | closest-corner | farthest-side | farthest-corner

background-image: radial-gradient(circle 100px at top left, red 50%, blue 50%);

si establecemos un solo valor, el segundo por defecto será center

background-image: radial-gradient(circle 100px at top, red 50%, blue 50%);

background-image: repeating-radial-gradient(circle 20px,red 0,red 10px,blue 10px,blue 20px);

background-image: repeating-linear-gradient(30deg,red 0,red 10px,blue 10px,blue 20px);

Overflow y float

Overflow

Es una propiedad que controla como se va a comportar la caja con el contenido que se desborde de ella

Es un shorthand que engloba overflow-x y overflow-y

Tiene 3 valores posibles:

hidden: Oculta todo el contenido que se desborde

auto: Muestra la barra de scroll solo si hace falta

scroll: Muestra ambas barras de scroll independientemente de si se necesitan

Float

float

La propiedad float especifica si un elemento debe salir del flujo normal y aparecer a la izquierda o a la derecha de su contenedor, donde los elementos de texto y los en línea aparecerán a su alrededor. Un elemento flotante es un elemento en el que el valor calculado de float no es igual a none.

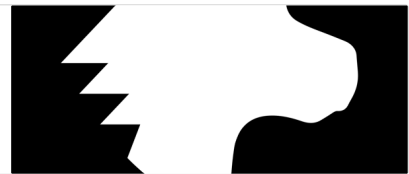
<https://developer.mozilla.org/es/docs/Web/CSS/float>



Clear

La propiedad CSS especifica si un elemento puede estar al lado de elementos flotantes que lo preceden o si debe ser movido (cleared) debajo de ellos. La propiedad clear aplica a ambos elementos flotantes y no flotantes.

<https://developer.mozilla.org/es/docs/Web/CSS/clear>



La propiedad float especifica si un elemento debe salir del flujo normal y aparecer a la izquierda o a la derecha de su c

Tiene 3 valores posibles:

left: Flota el elemento a la izquierda del contendor

right: Flota el elemento a la derecha del contendor

none: Elimina el float

NO EXISTE FLOAT: CENTER!!

Un elemento flotado hace que el padre deje de contenerlo, hay varias formas de solucionarlo, la más cómoda es overf

Sección 12: Flexbox

Fundamentos de flexbox, Flex-direction y flex-wrap

Flexbox

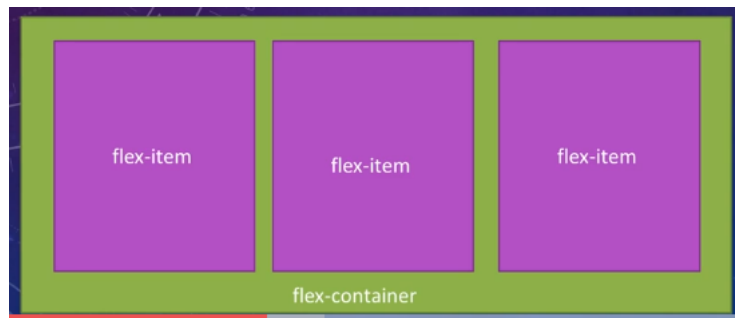
modelo de layout que permite que box sea flexible y es un valor de la propiedad display

Es necesario un contenedor (**flex-container**) y al menos un hijo (**flex-item**)

los elementos hijos de los flex-item **no son flex-item** hay que agregar la propiedad y se **convierte en flex-item y flex-container** al mismo tiempo

Los elementos que tengan display: flex y se utiliza **after y before** tambien se consideran flex

Los **textos** dentro de un **flex-item** también son flex

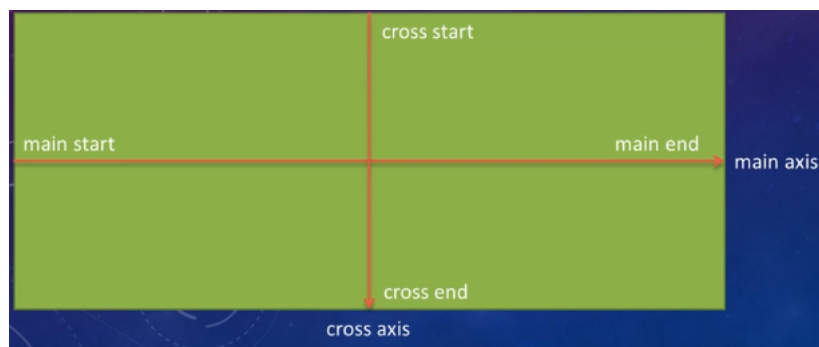


EJES

Hay 2 ejes para colocar y alinear los elementos flex

default: **EJE PRINCIPAL** es el horizontal y de **izquierda a derecha** y el **SECUNDARIO** es vertical de **arriba abajo**

Se puede intercambiar los ejes y la dirección de ambos



display:flex

Convierte el contenedor en un contexto para flexbox y hace que el contenedor sea un elemento de bloque para los

display:inline-flex

Convierte el contenedor en un contexto para flexbox y hace que el contenedor sea un elemento en línea para los el

flex-direction: row(default) | row-reverse | column | column-reverse

Modifica la dirección y cual es el eje principal.

Por defecto es row (horizontal de izquierda a derecha)

row-reverse (horizontal de derecha a izquierda)

column vertical de arriba abajo

column-reverse vertical de abajo arriba

flex-wrap: no-wrap(default) | wrap | wrap-reverse

Controla si los elementos saltarán de línea o no

nowrap es el valor por defecto, un contenedor flex va a hacer que todos los elementos se queden en una línea si no wrap hace que los elementos que no quepan en una línea (manteniendo sus dimensiones, si las tuvieran) salten a la

wrapreverse hace lo mismo que wrap pero en lugar de hacer que salten a la línea inferior, hace que salten a la línea

Aliniamiento

Estas propiedades para alinear se aplican SIEMPRE al flex-container

Existen propiedades para el main-axis y para el cross-axis

main-axis:

justify-content: flex-start(default) | center | space-between | space-around | space-evenly | flex-end

flex-start alinea los elementos al principio del main axis

center centra los elementos en la mitad del main axis

flex-end alinea los elementos al final del main axis

space-between distribuye los items a la misma distancia y no deja espacio exterior ni el primer ni el último flex-end.

space-around distribuye los items a la misma distancia y deja un espacio exterior en el primero y en el último flex-item

space-evenly distribuye los items y los espacios exteriores del primer y último flex-item dejando la misma distancia entre todos.

cross-axis

align-items: Una sola línea.

flex-start | center | flex-end | stretch(default) | baseline

align-content: varias líneas

flex-start | center | flex-end | stretch(default) | baseline

align-self: Se aplica al flex-item. Sirve para alinear un elemento en concreto en el cross-axis

TRUCO EXTRA: Al utilizar auto con la propiedad margin lo que sucede es que el elemento se va al lado contrario del r

flex-grow, flex-shrink, flex-basis, flex-flow y order

Flexbox

Propiedad flex: Es un shorthand que engloba las siguientes propiedades.

Todas las medidas se establecen en función del espacio disponible en el contenedor.

Todas las propiedades van en función del MAIN-AXIS.

flex-

grow: Establece qué hacer cuando hay espacio sobrante. Cuantas divisiones coge el elemento. El valor es un entero

flex-

shrink: Establece qué hacer cuando no hay espacio suficiente. Cuantas divisiones pierde el elemento. El valor es un

flex-

basis: Establece cuanto tiene que ocupar el item antes de encojer o extenderse. Flex basis prevalece ante width si el axis es horizontal o ante height si el main-axis es vertical. El valor es un entero desde 0 hasta n

Los valores de flex son:

Por defecto es 0 1 auto

flex:auto; equivale a 1 1 auto;

flex:none; equivale a 0 0 auto;

flex: flex-grow(obligatorio) flex-shrink(opcional) flex-basis(opcional);

Order: Establece qué orden ocupará el elemento en el contenedor flex. El valor es el "peso" respecto a los valores

Extra:

Existe un shorthand para flex-direction y flex-wrap. flex-flow: flex-direction flex-wrap. flex-flow: column wrap;

Práctica con flexbox

Sección 13: Grid

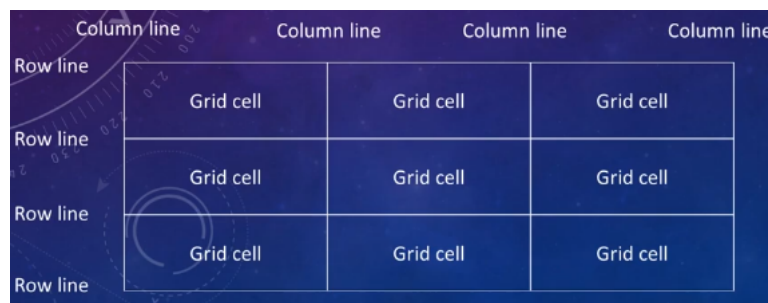
Fundamentos de grid

- Es layout que permite crear grillas o cuadrículas dinámicas. Es un **valor** de la propiedad **display**
- Tenemos **display: grid** y **display inline-grid**
- Se necesita el **grid-container** y su hijo **grid-item**
- Las **celdas** son los **items**, el **contenedor** es el **contexto** al que pertenecen los items

A	B	C	D
E	F	G	H
I	J	K	L

Terminología

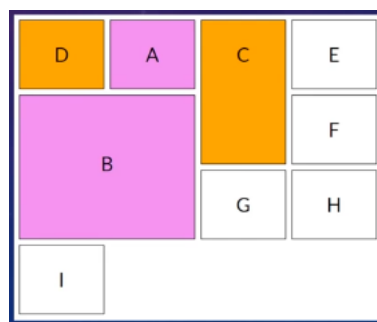
- **grid column:**
- **grid row:**
- **grid cell:**
- **grid gap:** separación entre las celdas.
- **grid line:** existen **column lines** y **row lines**, son las líneas que delimitan cada **columna/fila** respectivamente.



NOTA: si hay **N** filas entonces hay **N+1 ROW LINE** o **M** columnas entonces hay **M+1 COLUMN LINE**

Ventajas

- Cada celda es dinámica y el resto de la cuadrícula se adapta
- Podemos decir donde empieza cada celda y donde acaba, tanto en columna como en fila
- Posibilidades son inmensas.
- Ejemplo de cuadrícula:



Propiedades

- **grid-template-columns:** cantidad de columnas
- **grid-template-rows:** cantidad de filas
- **grid-column-gap:** distancia entre column
- **grid-row-gap:** distancia entre filas

grid-column y grid-row

grid-column-start: establece desde que column-line empezará el elemento

grid-column-end: establece hasta que column-line llega el elemento

grid-row-start: establece desde que row-line empezará el elemento

grid-row-end: establece hasta que row-line llega el elemento

Existe un shorthand que engloba las 2 propiedades

grid-column: start / end

grid-row: start / end

tanto start como end admiten valores positivos, negativos y la palabra span

Valor positivo: Empieza a contar las column-lines o las row-lines de izquierda a derecha

Valor negativo: Empieza a contar las column-lines o las row-lines de derecha a izquierda

span: establecemos cuantas columnas o filas ha de ocupar, span 3 es como decirle que ocupe 3 columnas

tip: Si ponemos en el valor end -1 llegará hasta el final

medidas y repeat()

Medidas de la cuadrícula

Se pueden dar medidas con cualquiera de las unidades que ya conocemos y dos más que vienen con grid, fr y auto

fr: Equivale a n fracciones del espacio disponible después de establecer las medidas fijas.

auto: Equivale al espacio que quede después de repartir todos elementos, es el último que se reparte. El tamaño mínimo del item será el espacio del contenido + el padding si lo tuviera

función repeat()

Con la función repeat podemos establecer repeticiones de medidas o patrones.

En el caso "simple" recibiría 2 parámetros (valores):

repeat(nColumnas o nFilas, medida)

grid-template-columns: repeat(4,100px) es lo mismo que grid-template-columns: 100px 100px 100px 100px;

Si como segundo valor añadimos más de una medida, construiremos un patrón, no hay límite de valores.

grid-template-columns: repeat(2, 100px 50px...) es lo mismo que grid-template-columns: 100px 50px 50px 100px;

implicit grid y explicit grid

Explicit grid:

Es el grid que declaramos, tanto con grid-template-columns como con grid-template-rows.

Implicit grid:

Es el grid que no declaramos, los items que quedan fuera del explicit grid.

grid-auto-columns:

Establece qué hacer con las columnas no definidas.

grid-auto-rows:

Establece qué hacer con las filas no definidas.

grid-auto-flow:

Establece la dirección en la cual se va a pintar el implicit grid. Admite 3 valores:

row(default): Se crearán filas adicionales.

column: Se crearán columnas adicionales.

dense: Establece qué hacer con los huecos que queden.

min-max(), auto-fill y auto-fit

minmax():

Recibe dos parámetros (valores) para establecer el mínimo y el máximo que pueden tener los items.

grid-template-columns: repeat(2, minmax(100px, 1fr));

min-content: mínimo necesario en función del contenido

max-content: máximo necesario en función del contenido

auto-fill: Crea tantos grid-items vacios como quepan en el viewport respetando las medidas.

auto-fit: Elimina los grid-items vacios que no se estén ocupando.

Alineamiento y order

justify-items: Alinea los elementos horizontalmente respecto a la celda

align-items: Alinea los elementos verticalmente respecto a la celda

Admiten los valores:

start

end

center

stretch (default)

place-items: Engloba las propiedades justify-items y align-items

place-items: align-items justify-items

justify-content: Alinea los elementos horizontalmente respecto al contenedor

align-content: Alinea los elementos verticalmente respecto al contenedor

Admiten los valores:

start

end

center

stretch (default)

space-around

space-between

space-evenly

place-content: align-content justify-content

Tenemos las mismas propiedades con self para los items:

justify-self

align-self

place-self

Admite los valores:

start
end
center
stretch

order: Funciona igual que en flexbox, por defecto todos los items tienen order:0

Grid template areas

establecer columnas y filas con nombres

grid-template-areas:

"header header header"

"aside article article"

"footer footer footer"

NOTA: agregar un punto (.) en vez de un nombre para dejar ese espacio vacío

Grid lines

```
.grid-container {  
  padding: 1rem;  
  background-color: #fff;  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    [first-column-line]  
    100px  
    [second-column-line]  
    100px  
    [third-column-line]  
    100px  
    [fourth-column-line]  
    100px  
    [last-column-line];  
  grid-template-rows:  
    [first-row-line]  
    100px  
    [second-row-line]  
    100px  
    [third-row-line]  
    100px  
    [fourth-row-line]  
    100px  
    [last-row-line];  
}
```

Grid animados

Grid puede existir dentro de otro Grid

Shorthands y grid track

grid: (sin soporte total)

grid-template: Engloba **grid-template-columns**, **grid-template-rows** y **grid-template-areas**

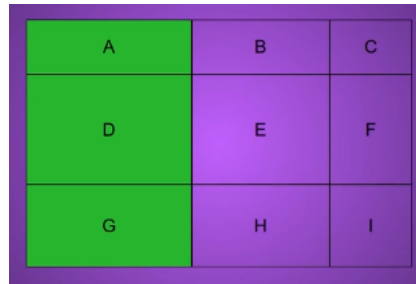
grid-gap: engloba grid-column-gap y grid-row-gap

grid-column: engloba grid-column-start/end

grid-row: engloba grid-row-start/end

GRID TRACK

Celdas seleccionadas consecutivas.



Práctica con grid

Sección 14: Responsive

Responsive Web Design

- No es un estándar, no es CSS, es un conjunto de patrones de diseño
- Objetivo es que cualquier página se vea bien independientemente del dispositivo
- Frameworks resuelven ese problema, pero es un error utilizarlos para evitar aprender lo que hay por detrás
- 3 Conceptos fundamentales
 - Columnas fluidas
 - imágenes flexibles: `max-width: 100%`
 - media queries

Columnas Fluidas

- aplicar **anchos en %**
- **NO** hay que asignar todos los anchos en %, solo se aplica a los **elementos que necesiten cambiar su tamaño**

Media queries

- Condicionales para que el navegador sepa cómo actuar en función de la condición
- **all**: apto para todos los dispositivos
- **print**: Destinado a material impreso y visualización del modo de vista previa de impresión
- **screen**: Destinado principalmente a las pantallas
- **speech**: Destinado a sintetizadores de voz

SINTAXIS

@media "tipo de medio" and | or

Condiciones más utilizadas son **min-width, max-width y orientation**

- @media screen and (min-width: 360px)
- @media screen and (max-width: 1024px) and (orientation: landscape)
 - **LANDSCAPE**: significa que el **width > height**
- @media screen and (max-width: 480px) or orientation: portrait
 - **PORTRAIT**: height > width

Metodologías

- **Moble-first**: consiste en hacer primero el diseño móvil e ir subiendo de tamaño (**recomendado**)
- **Desktop-first**: Consiste en hacer primero el diseño de escritorio e ir bajando de tamaño

- **Content-first:** Consiste en saber primero todo el contenido de la web y después se hace el diseño (**marketing, etc**)

Responsive sin breakpoints ni media queries

```
/*
banner con unidades de viewport
footer siempre abajo
responsive con flexbox
responsive con grid
*/
body {
  margin: 0;
  color: #eee;
  font-family: sans-serif;
  background-image: radial-gradient(circle, #b94bf8, #3a1957);
  background-repeat: no-repeat;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

/* .flexbox{
  display: flex;
  flex-wrap: wrap;
}

.flexbox__item{
  margin: 2rem;
  min-width: 320px;
  flex: 1 1 320px;
} */

.grid{
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(320px, 1fr));
  grid-gap: 2rem;
  padding: 1rem;
}

.banner{
  background-color: royalblue;
  padding: 15vmin;
  text-align: center;
  text-transform: uppercase;
  font-size: 1.5rem;
}

.footer{
  background-color: red;
  padding: 1em;
  text-align: center;
  text-transform: uppercase;
  font-size: 1.5rem;
  margin-top: auto;
}
```

```
<!DOCTYPE html>
<html lang="es">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grid -Shorthands</title>
    <link rel="stylesheet" href="styles.css">
  </head>

  <body>
    <header class="header">
      <div class="banner">Banner</div>
    </header>
    <div class="flexbox grid">
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
      <p class="flexbox__item grid__item">Lorem ipsum dolor sit amet consectetur adipisicing elit. In minus fugiat volupt
    </div>
    <footer class="footer">
      footer
  </body>
</html>
```

```

        </footer>
    </body>

</html>

```

Videos responsive

```

/*
    Vídeos responsive:
        La forma más común es hacer un padding-bottom de 56.25%, éste número es el resultado de la operación (9 / 16) es decir
*/

.container{
    width: 80%;
    margin-left: auto;
    margin-right: auto;
    min-width: 320px;
    max-width: 1400px;
}

.video{
    height: 0;
    padding-bottom: 56.25%;
    position: relative;
}

.video iframe{
    position: absolute;
    width: 100%;
    height: 100%;
}

```

```

<!DOCTYPE html>
<html lang="es">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Videos responsive</title>
        <link rel="stylesheet" href="styles.css">
    </head>

    <body>
        <div class="container">
            <div class="video">
                <iframe width="560" height="315" src="https://www.youtube.com/embed/Dym1Q1gr0FI" frameborder="0" allow="acceler
            </div>
        </div>
    </body>

</html>

```

Breakpoints

```

/*
    Breakpoints: Existen dos tipos de breakpoints, los major breakpoints y los minor breakpoints
    Los major breakpoints son los que cambian la apariencia de la web de forma significativa
    Los minor breakpoints son los que cambian una cosa específica para un elemento en concreto

    Medidas más comunes:
        0 - 320px - 360px: Móviles portrait
        480px - 640px - 768px: Móviles landscape - tablet
        1024px - 1200px - 1400px - 1440px: tablets grandes, portátiles no HD
        1920 - hacia arriba: HD, 2k, 4k...
*/

.menu{
    list-style: none;
    margin-top: 0;
    margin-bottom: 0;
    padding-left: 0;
    font-size: 3rem;
}

body{
    background-color: royalblue;
}

```

```
@media screen and (min-width:1400px){
  body{
    background-color: green;
  }

  .menu{
    display: flex;
    justify-content: space-around;
  }
}
```

```
<!DOCTYPE html>
<html lang="es">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Breakpoints</title>
    <link rel="stylesheet" href="styles.css">
  </head>

  <body>
    <ul class="menu">
      <li class="menu__item">Item 1</li>
      <li class="menu__item">Item 2</li>
      <li class="menu__item">Item 3</li>
      <li class="menu__item">Item 4</li>
    </ul>
  </body>

</html>
```