Selectores Css *2*

Modelo de Caja (CSS Box Model) *5*

Flexbox *8*

Float y clear *10*

Posicionamiento en CSS *10*

Listas *12*

Display Grid *14*

Animaciones CSS *16*

Column-count *17*

Responsive *18*

Overflow *19*

Sombras *20*

Filter *21*

Variables en Css *26*

Selectores Css

Los selectores CSS constituyen la piedra angular para aplicar estilos a los elementos HTML, ya que nos permiten seleccionar estos elementos para modificar su presentación visual. Existen diversas categorías de selectores:

1. **Selectores Simples:**

- Tipo: Dirigen los estilos a elementos basados en su nombre de etiqueta, ej., p, div, h1, etc.
- Clase: Seleccionan elementos en función del atributo de clase, utilizando un punto (.) como prefijo, ej., .clase.
- ID: Apuntan a un elemento único según su atributo de ID, utilizando una almohadilla (#) como prefijo, ej., #idUnico.

2. **Selectores Compuestos:**

- Descendente: Estilizan elementos anidados dentro de otros, identificados por un espacio entre ellos, ej., div p.
- Hijo Directo: Emplean el signo (>) para seleccionar únicamente los hijos directos, ej., div > p.
- Hermano Adyacente: Utilizan el signo (+) para seleccionar el elemento situado inmediatamente después de otro, ej., h2 + p.
- Hermanos: Emplean el signo (~) para seleccionar todos los hermanos subsecuentes de un elemento, ej., h2 ~ p.

3. **Selectores Pseudo-clases y Pseudo-elementos:**

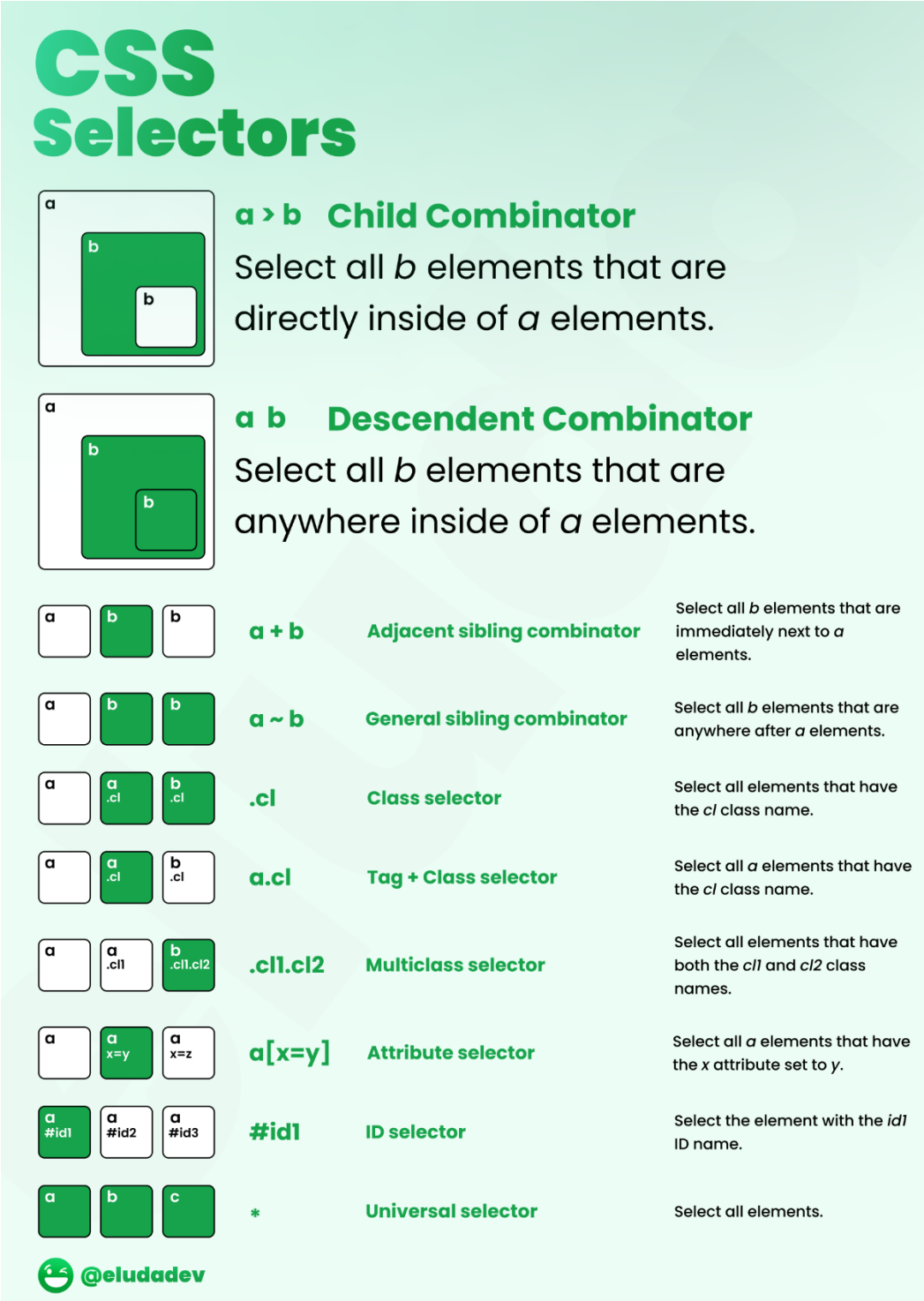
- Pseudo-clases: Eligen elementos en un estado particular, ej., :hover, :focus, :first-child, etc.
- Pseudo-elementos: Escogen partes específicas de un elemento, ej., ::before, ::after, ::first-line, etc.

4. **Selectores de Atributo:**

- Dirigen estilos a elementos con un atributo específico o valor de atributo, ej., [type="text"], [href^="https"], etc.

Los selectores pueden combinarse y anidarse para lograr una selección más detallada y específica, brindando una amplia flexibilidad y control en la estilización de páginas web. Comprender y dominar los selectores CSS es esencial para desarrollar hojas de estilos efectivas y eficientes.

Imagen Selectores



Web Selectores avanzados

En esta web podréis encontrar todos los selectores posibles y sus ejemplos: https://fffuel.co/css-selectors/

Juego Selectores Css

Si necesitas practica o recordar un poco los selectores este juego puede ayudaros:

https://flukeout.github.io/

Modelo de Caja (CSS Box Model)

El Modelo de Caja CSS es un concepto fundamental en el diseño web, ya que define cómo los elementos son estructurados y visualizados en la página. Cada elemento en una página web se considera como una "caja" con diferentes capas que detallan su composición y presentación.

Contenido (Content):

Es el núcleo del modelo de caja, que incluye el contenido real del elemento, como texto, imágenes o video. Su tamaño puede ser controlado usando las propiedades width y height.

Relleno (Padding):

Es el espacio entre el contenido del elemento y su borde. Aunque es transparente, el relleno puede afectar el tamaño total del elemento y se puede ajustar con la propiedad padding.

Borde (Border):

Es la línea que envuelve el relleno y el contenido. Puede ser estilizada en términos de tamaño, estilo y color con la propiedad border.

Margen (Margin):

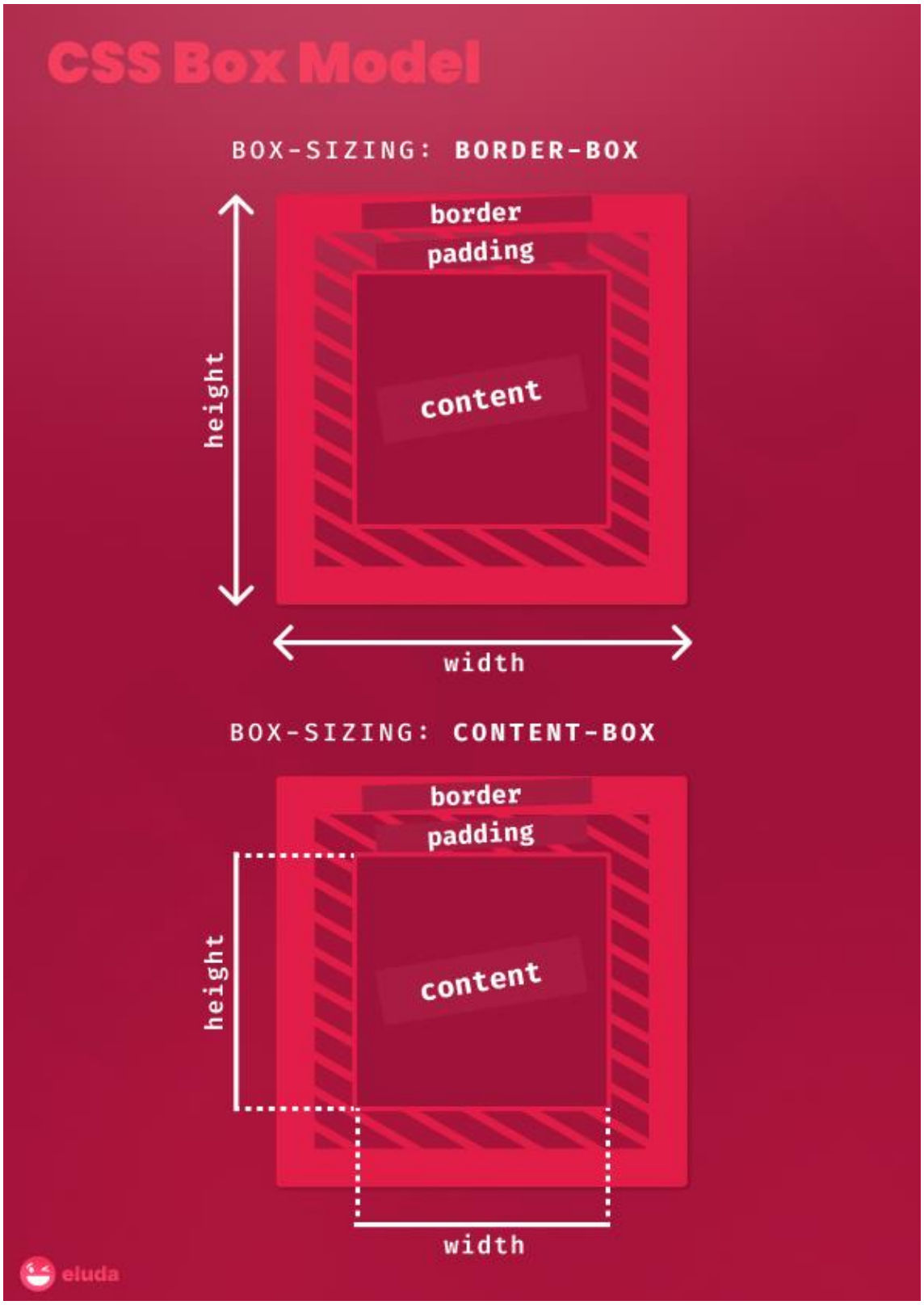
Es el espacio exterior al borde, utilizado para separar el elemento de otros elementos en la página. Es transparente y su tamaño se puede controlar con la propiedad margin.

El modelo de caja es crucial para entender cómo los elementos afectan y son afectados por su entorno. Es importante tener en cuenta que las dimensiones totales de un elemento son la suma de sus contenidos, relleno, borde y margen.

Box-sizing

La propiedad box-sizing permite alterar cómo se calcula el tamaño de la caja, siendo content-box el valor por defecto, y border-box un valor comúnmente utilizado para incluir el relleno y el borde en el tamaño del elemento.

Comprender y dominar el Modelo de Caja CSS es esencial para el diseño web efectivo, ya que afecta la disposición, tamaño y espaciado de los elementos en la página.


CSS Box Model
BOX-SIZING: BORDER-BOX
border
padding
content
height
width
BOX-SIZING: CONTENT-BOX
border
padding
content
height
width
eluda

Flexbox

Flexbox, o modelo de caja flexible, es una técnica de diseño en CSS que simplifica la disposición de elementos en un contenedor, haciendo que el espacio entre los elementos sea distribuido y los elementos mismos se alineen de manera más predecible, incluso cuando sus tamaños son desconocidos o dinámicos.

Contenedor Flex (Flex Container):

- Para crear un contenedor flex, aplicamos **display: flex; o display: inline-flex;** a un elemento padre. Esto activa el contexto flex para todos sus hijos directos.
- Propiedades importantes incluyen **flex-direction** (define la dirección principal del flujo de elementos), **justify-content** (alinea elementos a lo largo del eje principal), **align-items** (alinea elementos a lo largo del eje transversal), y **flex-wrap** (permite que los elementos se pasen a la línea siguiente).

Elementos Flex (Flex Items):

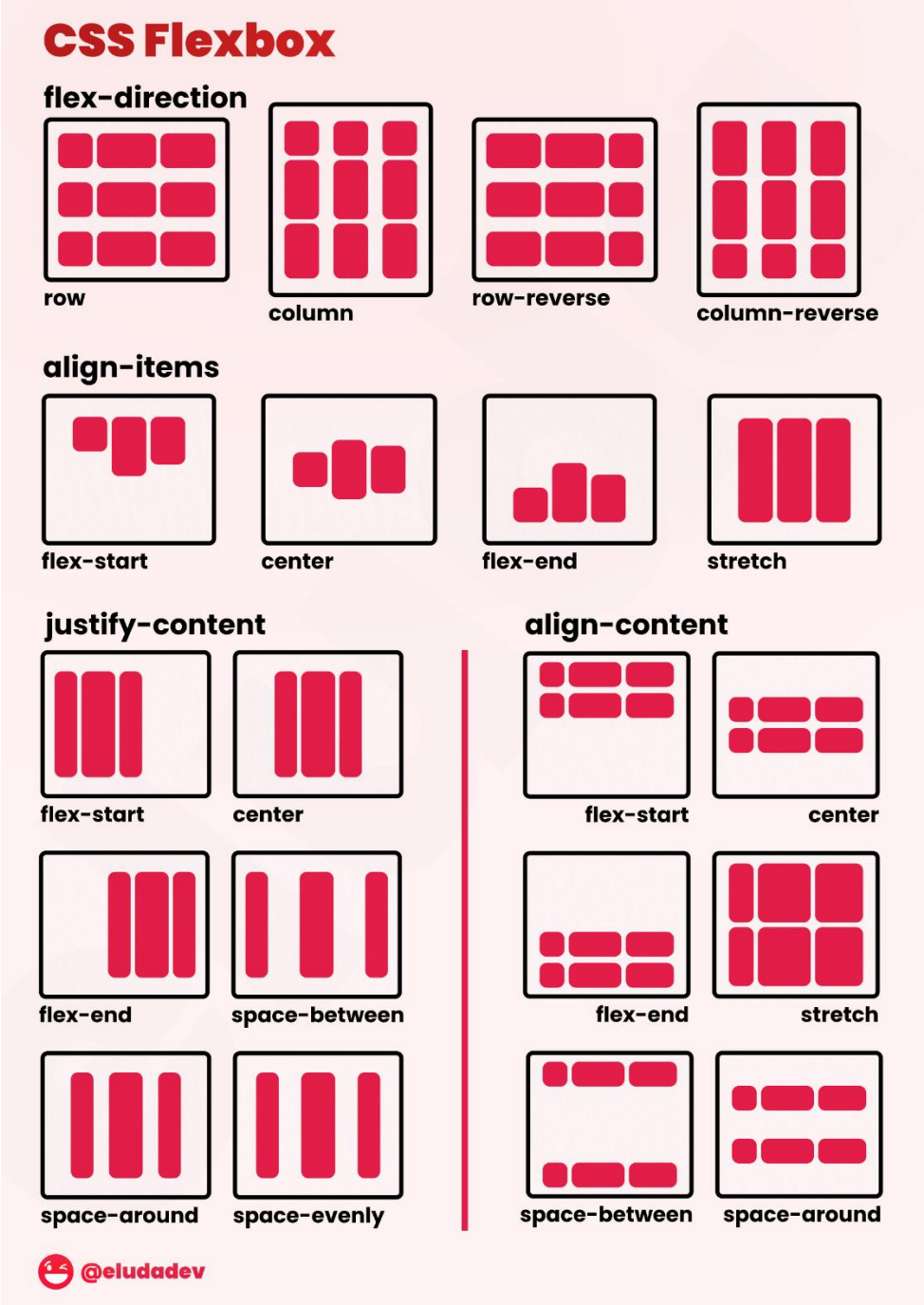
- Los hijos directos de un contenedor flex se convierten en elementos flex.
- Propiedades importantes para los elementos flex incluyen **flex-grow** (define la capacidad de un elemento para crecer si es necesario), **flex-shrink** (define cómo se reducirá un elemento), **flex-basis** (define el tamaño base de un elemento), y **align-self** (permite la alineación por defecto del contenedor flex para ser anulada para elementos individuales).

Ejes Flex:

- El eje principal es definido por **flex-direction**, siendo row la opción por defecto.
- El eje transversal es perpendicular al eje principal y es donde **align-items** y **align-self** tienen su efecto.

Flexbox es especialmente útil cuando la disposición de nuestros elementos debe ser sensible al contenido y al tamaño de la pantalla (responsive). Permite el control del espacio y alineación, tratando los problemas comunes de diseño en la interfaz de usuario de una manera eficiente.

Imagen Flexbox



Juegos Flexbox

https://flexboxfroggy.com/#es

http://www.flexboxdefense.com/

Float y clear

Estas dos propiedades, **“float” y “clear”** sirven para posicionar y formatear el contenido de un elemento o varios de una página web.

Con float lo que hacemos es establecer cómo debería “flotar” un elemento en un contenido. Por ejemplo para situar una imagen a la derecha o la izquierda de un párrafo.

Con clear lo que hacemos es establecer que el siguiente elemento flotante (float) no se pueda poner a la derecha o a la izquierda (o ambos).

Posicionamiento en CSS

El posicionamiento en CSS3 es una característica fundamental que permite a los desarrolladores controlar precisamente dónde se colocan los elementos en la página web. Dependiendo del método de posicionamiento aplicado, los elementos pueden ser manipulados de manera diferente en el espacio de diseño.

Static:

- Valor por defecto para todos los elementos.
- No es afectado por las propiedades top, right, bottom, o left.
- Sigue el flujo normal del documento.

Relative:

- Posiciona un elemento en relación con su posición original en el flujo normal.

- Las propiedades top, right, bottom, y left afectan su posición final.

Fixed:

- El elemento se posiciona en relación con la ventana del navegador.
- No se mueve cuando se hace scroll en la página.
- Afectado por las propiedades top, right, bottom, y left.

Absolute:

- El elemento se posiciona en relación con su ancestro posicionado más cercano o, si no existe, en relación con el bloque contenedor inicial.
- No sigue el flujo normal del documento.
- Afectado por las propiedades top, right, bottom, y left.

Inherit:

- Hereda el valor de la propiedad position de su elemento padre.

Initial:

- Resetea el valor de la propiedad al valor inicial, es decir, static.

Sticky:

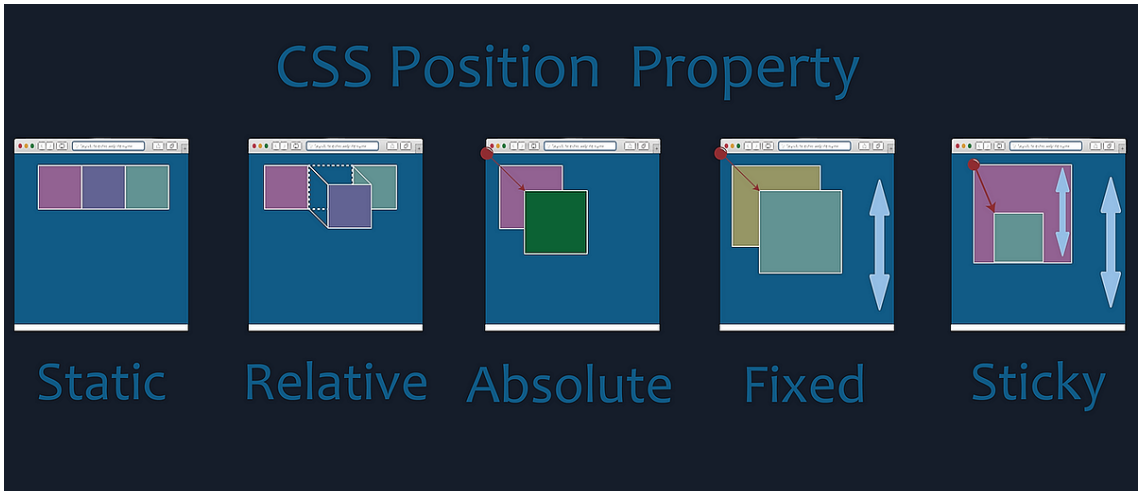
- Combina los comportamientos de los posicionamientos relative y fixed.
- El elemento se mueve con el scroll hasta que alcanza un punto específico, después se "pega" en el lugar.

Z-index:

- Controla el orden de apilamiento de los elementos a lo largo del eje Z.
- Solo funciona en elementos posicionados (no static).
- Los valores más altos se sitúan encima de los valores más bajos.

Es importante entender que el uso correcto del posicionamiento y del z-index puede influir significativamente en la usabilidad y la apariencia de una página

web. La propiedad position y sus valores correspondientes ofrecen a los desarrolladores una herramienta poderosa para crear diseños complejos y visualmente atractivos.



Listas

List-style-type

La propiedad list-style-type nos permite establecer el tipo de estilo que queremos para los marcadores de la lista, existen varios valores que puede tomar y los conoceremos a continuación.

- **disc** - indica un marcador en forma de disco.
- **Circle** - marcador con forma de círculo.
- **Square** - tipo de estilo cuadrado.
- **Decimal** - Este valor enumera la lista empezando por 1.
- **Decimal-leading-zero** - Empieza la numeración con un 01.
- **lower-roman** - lista con números romanos en minúscula (i, ii, iii, iv, v, etc.).
- **upper-roman** - números romanos en mayúscula (I, II, III, IV…).
- **lower-greek** - letras griegas en minúscula alfa/α, beta/β, gamma/γ, …
- **lower-latin** - indica letras ASCII en minúscula (a, b, c, … z).
- **upper-latin** - letras ASCII en mayúscula (A, B, C, … Z).

- **armenian** - aplica numeración armenia tradicional (ayb/ayp, ben/pen, gim/keem, …).
- **Georgian** - numeración georgiana tradicional (an, ban, gan, …, he, tan, in, in-an, …).
- **None** - no aplica ningún estilo.

list-style-image

La propiedad list-style-image de CSS nos permite establecer una imagen como viñeta para los elementos de una lista.

```
ul.conimagen {
        list-style-image:url("sonrisa.jpg");
}
```

list-style-position

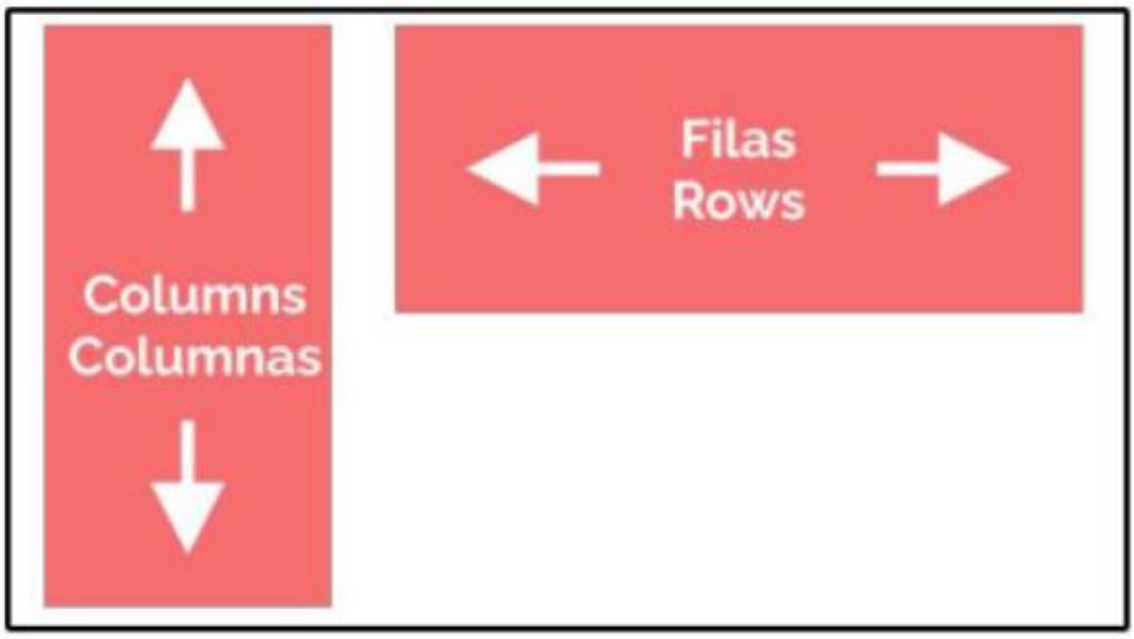
La propiedad list-style-position nos permite establecer la posición de la caja de viñetas y puede tomar los valores:

- **Inside** - indica que la lista está adentro del flujo de contenido.
- **Outside** - este es el valor por defecto e indica que la lista está fuera del flujo de contenido.

Display Grid

https://www.w3schools.com/cssref/pr_grid.php

Grid es un modelo de diseño bidimensional que permite diseñar estructuras complejas y alinear contenido en columnas y filas.



Básicos

Contenedor Grid

```
.contenedor-grid {
    display: grid;
}
```

Definición de columnas y filas:

```
.contenedor-grid {
    grid-template-columns: 1fr 2fr 1fr;  /* Tres columnas: la del medio es el doble de ancho que las otras dos */

    grid-template-rows: auto 100px;      /* Dos filas: la primera se adapta al contenido y la segunda tiene 100px */
}
```

Colocación de ítems

Posición específica

```
.item1 {
    grid-column: 1 / 3;  /* Desde la línea 1 hasta la línea 3 */
    grid-row: 1;         /* En la primera fila */
}
```

Espaciado entre ítems

```
.contenedor-grid {
    grid-gap: 20px; /* Espaciado uniforme de 20px entre columnas y
filas */
}
```

O define espaciados distintos para columnas y filas:

```
.contenedor-grid {
    grid-row-gap: 10px;     /* Espaciado entre filas */
    grid-column-gap: 20px;  /* Espaciado entre columnas */
}
```

Áreas del Grid

Puedes definir áreas y colocar ítems en esas áreas:

```
.contenedor-grid {
    grid-template-areas:
        "header header header"
        "sidebar content aside"
        "footer footer footer";
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.content { grid-area: content; }
.aside { grid-area: aside; }
.footer { grid-area: footer; }
```

Responsive Design con Grid

Usando Media Queries, puedes definir diferentes layouts según el tamaño de la pantalla:

```
/* Mobile layout por defecto */

.contenedor-grid {
    grid-template-areas:
        "header"
        "content"
        "sidebar"
        "aside"
        "footer";
}

/* Layout para tablets y desktops */
```

```
@media (min-width: 768px) {
    .contenedor-grid {
        grid-template-areas:
            "header header header"
            "sidebar content aside"
            "footer footer footer";
    }
}
```

Animaciones CSS

http://canianimate.com

Las animaciones nos permiten, al igual que las transiciones, pasar un elemento de un estado a otro, permitiendo definir un número variable de estados intermedios. Es decir, nos ofrecen un mayor control sobre el cambio de estado.

Propiedades

Propiedades	Descripción	Valor
animation-name	Nombre de la animación a aplicar.	none | *nombre*
animation-duration	Duración de la animación.	0 | TIME
animation-timing-function	Ritmo de la animación.	Ver funciones de tiempo
animation-delay	Retardo en iniciar la animación.	0 | TIME
animation-iteration-count	Número de veces que se repetirá.	1 | infinite | NUMBER
animation-direction	Dirección de la animación.	normal | reverse | alternate | alternate-reverse
animation-fill-mode	Como se quedará la animación al terminar.	none | forwards | backwards | both
animation-play-state	Estado de la animación.	running | paused

Funciones de Tiempo

Valor	Inicio	Transcurso	Final	Equivalente en cubic-bezier
ease	Lento	Rápido	Lento	cubic-bezier(0.25, 0.1, 0.25, 1)
linear	Normal	Normal	Normal	cubic-bezier(0, 0, 1, 1)
ease-in	Lento	Normal	Normal	cubic-bezier(0.42, 0, 1, 1)
ease-out	Normal	Normal	Lento	cubic-bezier(0, 0, 0.58, 1)
ease-in-out	Lento	Normal	Lento	cubic-bezier(0.42, 0, 0.58, 1)
cubic-bezier(*A, B, C, D*)	-	-	-	Transición personalizada

Atajo

animation: <name> <duration> <timing-function> <delay> <iteration-count> <direction> <fill-mode> <play-state>

animation: change-color 5s linear 0.5s 4 normal forwards running;

Keyframes

Aquí es donde realizaremos la animación que pretendemos hacer.

```
@keyframes change-color {
  from { background: red; }
  to { background: green; }
}
```

```
@keyframes change-color {
  0% {
    background: red;
  }
  50% {
    background: yellow;
    width: 400px;
  }
  100% {
    background: green;
  }
}
```

Column-count

Con Flex ya somos capaces de hacer diseños por columnas y tablas, pero CSS3 introdujo recientemente el sistema column-count.

- Column-count es una propiedad única de CSS que permite dar un número de columnas a un contenido.
- Aplicable a contenedores de elementos hijos.
- Su valor se da en números enteros.

Column-count: indica en cuántas columnas debemos dividir el contenido.

column-gap: indica el espacio entre columnas.

column-rule: indica el borde entre columnas, funciona igual que la propiedad border.

Responsive

Breakpoints

Lo llamado Breakpoint es el punto en el que "pasan cosas", en contexto responsive, el tamaño de pantalla en el que los elementos se empiezan a colocar de otra manera.

Por ejemplo, en escritorio mostraremos la información en cuatro columnas, en Tablet en 3 columnas, desplazando el resto de la información a la siguiente columna. En movil lo mostraremos a dos columnas.

Los puntos que hacen que cambie la rejilla son los breakpoints

Media queries

Los media queries plantean una condición, si nuestro navegador no cumple esa condición no leera el css que contiene.

```
@media screen and (*condición*) {
  /* reglas CSS */
  /* reglas CSS */
}
```

Media queries

```
@media screen and (max-width: 640px) {
 #banner {
  background: blue;
 }
}
```

```
@media screen and (min-width: 640px) and (max-width: 1280px) {
  #banner {
   background: red;
 }
}
```

```
@media screen and (min-width: 1280px) {
#banner{
   background: green;
 }
}
```

Overflow

La propiedad overflow de CSS especifica qué hacer si el contenido de un contenedor supera sus dimensiones.

- **visible**: Es el valor por defecto. El contenido que excede las dimensiones del contenedor se muestra y desborda el contenedor.

- **hidden**: El contenido que excede las dimensiones del contenedor se oculta.

- **scroll**: Se añaden barras de desplazamiento al contenedor, independientemente de si el contenido excede las dimensiones o no.

- **auto**: Las barras de desplazamiento se añaden solo si el contenido excede las dimensiones del contenedor.

overflow-x y overflow-y

Puedes controlar el comportamiento de desbordamiento en los ejes horizontal y vertical de manera independiente:

- **overflow-x**: Controla el desbordamiento del contenido en el eje horizontal (izquierda/derecha).
- **overflow-y**: Controla el desbordamiento del contenido en el eje vertical (arriba/abajo).

Sombras

box-shadow

La propiedad box-shadow se usa para añadir sombras a los elementos.

box-shadow: [desplazamiento horizontal] [desplazamiento vertical] [difuminado] [extensión] [color];

- **Desplazamiento horizontal y vertical**: Indican la distancia en la que la sombra se moverá en relación con el elemento.
- **Difuminado:** Define cuánto se difuminará la sombra.
- **Extensión**: Aumenta o disminuye el tamaño de la sombra.
- **Color:** Especifica el color de la sombra.

Text-shadow

La propiedad text-shadow se utiliza para añadir sombras a los textos.

text-shadow: [desplazamiento horizontal] [desplazamiento vertical] [difuminado] [color];

- **Desplazamiento horizontal y vertical**: Indican la distancia en la que la sombra se moverá en relación con el texto.
- **Difuminado:** Define cuánto se difuminará la sombra.
- **Color**: Especifica el color de la sombra.

Filter

Los filtros nos dan la capacidad de alterar los colores a partir de un algoritmo determinado. Lo podemos aplicar a elementos o imágenes.

Difuminar

```css
.elemento {
  filter: blur(5px);
}
```

Brillo

```css
.elemento {
  filter: brightness(2);
}
```

Original Brightness

Contraste

```css
.elemento {
  filter: contrast(200%);
}
```

Original Contrast

Sombra de la imagen

Solo funciona con SVGs.

posición x, posición y, suavizado y color.

```css
.elemento {
  filter: drop-shadow(30px 10px 4px green);
}
```

Original Drop-shadow

Escala de grises o transformar a blanco y negro

```css
.elemento {
  filter: grayscale(100%);
}
```

Original Grayscale

Transparencia

```css
.elemento {
  filter: opacity(50%);
}
```

Original Opacity

Saturación

```css
.elemento {
  filter: saturate(50%);
}
```

Original Saturate

Sepia

```css
.elemento {
  filter: sepia(100%);
}
```

Original Sepia

Múltiples filtros

```css
filter: contrast(175%) brightness(3%);
```

Variables en Css

Las variables en CSS te permiten asignar valores específicos a nombres definidos por el usuario, y luego referenciar esos nombres en lugar de los valores repetidamente a lo largo de tus estilos. Esto hace que el mantenimiento y la actualización de los estilos sean más fáciles.

Para aplicar unas variables que se apliquen a todo el documento debemos declararlas dentro de :root.

```
:root {
    --color-amarillo: #ECD078;
    --color-naranja: #D95B43;
    --color-rojo: #C02942;
    --fuente-titulo: 'Arial';
}

h1 {
    color: var(--color-naranja);
    font-family: var(--fuente-titulo);
}
```