CSS DOCUMENTATION

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

Documentation

align-content	Specifies the alignment between the lines inside a flexible container when the items do not use all available space
align-items	Specifies the alignment for items inside a flexible container
align-self	Specifies the alignment for selected items inside a flexible container
background	A shorthand property for all the *background-** properties
background-position	Specifies the position of a background image
background-size	Specifies the size of the background images

border	A shorthand property for *border-width, border-style* and *border-color*
box-shadow	Attaches one or more shadows to an element
box-sizing	Defines how the width and height of an element are calculated: should they include padding and borders, or not
cursor	Specifies the mouse cursor to be displayed when pointing over an element
display	Specifies how a certain HTML element should be displayed
flex	A shorthand property for the *flex-grow, flex-shrink*, and the *flex-basis* properties
font-family	Specifies the font family for text

font-size	Specifies the font size of text
grid	A shorthand property for the *grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns*, and the *grid-auto-flow* properties
height	Sets the height of an element
justify-content	Specifies the alignment between the items inside a flexible container when the items do not use all available space
justify-items	Is set on the grid container. Specifies the alignment of grid items in the inline direction
justify-self	Is set on the grid item. Specifies the alignment of the grid item in the inline direction
max-height	Sets the maximum height of an element. Min-height also.

max-width	Sets the maximum width of an element. Min-width also.
@media	Sets the style rules for different media types/devices/sizes
opacity	Sets the opacity level for an element
outline	A shorthand property for the *outline-width, outline-style*, and the *outline-color* properties
overflow	Specifies what happens if content overflows an element's box
padding	A shorthand property for all the *padding-** properties
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)
rotate	Specifies the rotation of an element

text-align	Specifies the horizontal alignment of text
text-justify	Specifies the justification method used when text-align is "justify"
text-shadow	Adds shadow to text
top	Specifies the top position of a positioned element
transition	A shorthand property for all the *transition-** properties
transform	Applies a 2D or 3D transformation to an element
width	Sets the width of an element
z-index	Sets the stack order of a positioned element

Pro-Tips

INVESTIGAR PÁGINAS GRANDES E IR COPIANDO, SER BUEN FRONTEND

caja:hover > .hijo{} Cuando toque la caja, cambia al .hijo (funciona con hijos)
Usar variables en colores para cambiar muchos colores con una línea; color: var(--nombredelavariable). Luego solo cambio –nombredelavariable = #aff.
Tener la redirección 303 de la https.
Rangos en posicionamiento web.
White-space: nowrap; así no se va hacia abajo y mejora al cambiar resolución.
Ojo cuidado al agrandar la página.

SEO: en los links ir a mi misma página. <a href=”https:www.mipagina.com/x”>
SEO: las imágenes deben tener alt.

text-align: justify; justificar texto.

CANONICAL

line-height: separación entre filas de texto.

- No permitir contraseñas débiles.

Particles JS: interactivity: true;

Certificado SSL

vh es viewport height y vw es viewport width

Posicionamiento de webs y buenas prácticas:

Pseudoelementos

::first-line -> (no en inline) Siempre detecta la primera línea de texto, no importa cuál sea el tamaño, siempre tomará la primera.

::first-letter -> (no en inline) Lo mismo pero para la primera letra del texto.

::placeholder -> Lo que aparece dentro del input.

::selection -> Cuando selecciono algo, se suele ver azul, lo puedo modificar.

::after -> (inline, lleva content=””) Coloca un contenido texto antes del elemento.

::before -> (inline, lleva content="") Coloca un contenido texto después del elemento.

Pseudoclases

:hover -> Mouse encima.

:link -> Enlace no visitado.

:visited -> Enlace que ya ha sido visitado.

:active -> Solamente cuando se da click, si se deja de apretar, se va.

:focus -> Para los inputs, cuando selecciono el elemento se cambia.

:lang ->

Link edit

- Link (no visitado): El estado predeterminado que presenta un enlace cuando no está en ningún otro estado. Se puede especificar usando la pseudoclase `:link`.
- Visited: Un enlace cuando ya se ha visitado (está grabado en el historial del navegador); se le aplica otro formato con la pseudoclase `:visited`.
- Hover: Un enlace cuando se le pasa el cursor por encima; se le aplica otro formato con la pseudoclase `:hover`.
- Focus: Un enlace cuando tiene el foco (por ejemplo, se salta a este con la tecla Tab del teclado o se le da el foco mediante programación usando `HTMLElement.focus()`); se le aplica un formato diferente con la pseudoclase `:focus`.
- Active: Un enlace cuando se activa (por ejemplo, se hace clic encima); se le aplica un formato diferente con la pseudoclase `:active`.

```
body {
  width: 300px;
  margin: 0 auto;
  font-size: 1.2rem;
  font-family: sans-serif;
}

p {
  line-height: 1.4;
}

a {
```

```
  outline: none;
  text-decoration: none;
  padding: 2px 1px 0;
}

a:link {
  color: #265301;
}

a:visited {
  color: #437A16;
}

a:focus {
  border-bottom: 1px solid;
  background: #BAE498;
}

a:hover {
  border-bottom: 1px solid;
  background: #CDFEAA;
}

a:active {
  background: #265301;
  color: #CDFEAA;
}
```

Images Object Fict

Se aplica al <img>, tenemos *object-fit*:

- Contain: por defecto, se ajusta a su resolución y deja espacios en blanco.
- Cover: se ajusta al contenedor, se corta la imagen.

También tenemos *object-position* que modifica dónde está la imagen (sirve rem).

Position

Afecta flujo HTML: orden en que ponemos los elementos.

El padre debe estar posicionado para que sus hijos tomen como referencia a él.

Static: Valor por defecto. Top, left, right, bottom no tienen efecto. Podemos utilizar el margin auto aquí.

Relative: Se le da posición, pero el espacio se sigue manteniendo. Adquiere también el z-index. Podemos usar el margin auto para centrar horizontalmente.

Para poner al hijo por debajo del padre, debe el hijo solamente tener un z-index con valor de -1, el padre no debe tener z-index.

Absolute: No ocupa espacio, es como que flota, no altera a los demás. Se ajusta al tamaño de su contenido al aplicarlo. Si el padre es absolute y se mueve, todos sus hijos se mueven con él. Lo puedo dejar en medio dándole a top left right bottom un valor de 0, además del margin auto.

Fixed: Es lo mismo que absolute, pero no se mueve, queda fijo ahí. Sirve para que la publicidad siga fija.

Sticky: Combinación de fixed y relative, pero empieza a quedarse fijo cuando scroll. Le damos position sticky y un top de 0.

Overflow

Qué haremos con el contenido que se sale del contenedor -> overflow x,y

- auto; si se sale, permite hacer scroll en el contenedor.
- hidden; oculta lo que sale

Flex

flex-direction: column, row, row-inverse, column-reverse

flex-wrap: wrap; si se disminuye, bajan. No les des height al contenedor para que se adapte al contenido. Existe un wrap reverse, donde los que no caben irán hacia arriba.

flex-flow: [flex-direction] [flex-wrap]

justify-content: para el main axis (horizontal). Con flex-start va a la izquierda, flex-end a la derecha, center al centro, space-around separa los elementos y también con el contenedor, space-between separa lo máximo entre elementos. Space-evenly distribuye el mismo espacio para todos lados, incluyendo contenedor.

align-items: para el cross axis (vertical). Mismos que el anterior, pero con baseline podemos alinear según el texto, a la misma altura.

align-content: para el cross axis.

flex-grow: Si damos el valor de 1 a un solo elemento, intentará ocupar todo el espacio posible, los demás se contraerán. Esto hace efecto cuando aplicamos números diferentes a otros elementos. No son medidas como px o rem.

flex-shrink: Mientras mayor es su número, más se contrae (cuando se cambian las dimensiones de la pantalla), todos tienen por defecto valor de 1.

flex-basis: Mejorando el min-width y max-width, le pongo el tamaño (ahora sí uso rem, px) y siempre intentará llegar a ese tamaño, pero si algo cambia, no se desbordará, intentará ocupar todo lo máximo que se pueda. Cuando llegue a su espacio, se detiene, si se contrae, lo hace también.

flex: [flex-grow] [flex-shrink] [flex-basis]

align-self: Se alinea verticalmente (aplica al elemento, mismo que justify content, en sus opciones.

Transition

- *transition-property:* Qué propiedad se modificará (no usar all)
- *transition-duration:* Lo que dura la transición (1s)
- *transition-delay:* Lo que se demora en aplicar la transición, por ejemplo en el hover, se aplica delay luego de colocar el mouse encima (3s)
- *transition-timing-function:* Velocidad de la transición en función del tiempo:

linear = los cambios se ven a la misma velocidad
ease = arranca con todo y termina despacito (más intenso)
ease-in = arranca con poco y termina con todo
ease-out = arranca fuerte y termina lento
ease-in-out = arranca despacio, acelera, termina despacio.

Animaciones @Keyframes

.

Transform

Translate(50px,50%) depende del tamaño del contenedor. (TranslateX, TranslateY). Translate es la más eficiente y consume menos recursos.
Scale(2) Si le damos 2, es el doble de grande. También hay scaleX, scaleY. Dentro de una propiedad transform , podemos dar espacio y juntar translate y scale.
Skew(30deg) se deforma diagonalmente, si queremos que el texto no se deforme, a este le damos skew(-30deg) para la sumatoria de cero. Pero tenemos clip path.

Hay un montón de otras más.

Hay generadores para clip path(forma específica de algo) o sombras.

Box-Shadow

box-shadow: horizontal, vertical, blur, color

Box-sizing

```
*{
    scroll-behavior: smooth;
    user-select: none; /*no puede seleccionar*/
    box-sizing: border-box;
}
```

Controlamos el cálculo que se hace en margin, padding, border. Estos añaden tamaño a la caja, por lo que es un problema, mide más.
Por defecto es *content-box* y *border-box* calcula el tamaño del elemento incluyendo lo demás. Restamos tamaño de contenido para mantener el tamaño que indico con width y height.

Background

Background-image: Agrego la imagen al contenedor.

Background-size: 100% y le damos el 100% de la imagen. Con el *cover* se ajusta la imagen al fondo. Con *contain* se ajusta al contenedor. Damos medidas.

Background-repeat: No-repeat; (se rellena lo que falta con background el color)

Background-clip: Desde donde se empieza a ver la imagen, la recorta. Algo parecido con el *background-origin* que es donde empieza y termina, sin recortarla.

Background-position: Movemos la imagen de posición.

Background-attachment: La imagen se queda quieta y genera como un hoyo, la cual se mueve al hacer scroll y la imagen de fondo queda quieta.

Variables

```css
:root {
    --color-rojo-claro: #f44;
    --color-azul-claro: #66f;
}

div {
    padding: 30px;
    height: 150px;
    width: 150px;
    margin: 30px;
}

.container {
    background: var(--color-rojo-claro)
}

.container-2 {
    background: var(--azul-claro)
}
```

Filtros

Filter: diferentes opciones.

- blur(10px) funciona con px o rem solamente. Es desenfoque.
- brightness(números)
- contrast(números)
- drop-shadow(10px 10px 10px black) da sombra a imágenes sin fondo
- opacity(0.5)
- saturate(%)

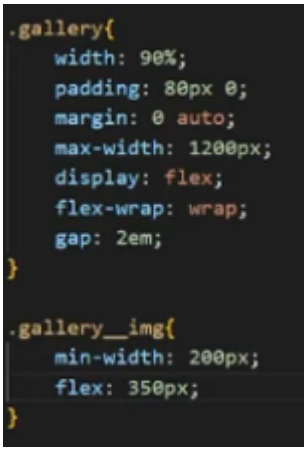
Otras propiedades

- Direction: cambia la posición del texto; rtl, ltr
- Letter spacing: separación entre letras (px)
- Scroll-behavior: lo que se mueve el scroll automáticamente, se pone smooth, así cuando me muevo de un punto a otro es suave.
- User-selected: no seleccionar la letra de la página.

Responsive Design (con y sin media querys)

container: display flex, flex-wrap: wrap, align-content flex-start y al item le damos un flex-grow de 1 y flex-basis de 100px eso puede cambiar. Se adapta.

Galería de fotos para dispositivos móviles:



```
.gallery{
    width: 90%;
    padding: 80px 0;
    margin: 0 auto;
    max-width: 1200px;
    display: flex;
    flex-wrap: wrap;
    gap: 2em;
}

.gallery__img{
    min-width: 200px;
    flex: 350px;
}
```

Tips (centrar, mantener footer abajo, etc etc)

imágenes de código, por si se me olvida cómo hacerlo, en vez de buscarlo por internet lo busco por aquí porque ya lo hice, este es mi internet y torpedo

Flex con margin 0 auto

```css
.title{
    padding: 1rem 0;
    border: 1px solid black;
    display: flex;
}

.span-title{
    color: aqua;
    margin: 0 auto;
    border: 1px solid black;
}
```

Mantener footer abajo (1)

- Body con *flex* y *direction* column, dar un *justifty* de space-between.