Lexpin

¿Qué es CSS?

CSS (Cascading Style Sheets / Hojas de Estilos en Cascada) es un lenguaje de estilizado usado para elaborar la presentación visual de un documento HTML. Con CSS podemos definir la apariencia y el diseño de una página web.

¿Qué debes saber antes de utilizar CSS?

Antes de comenzar a usar CSS debes saber que son los selectores, y es que, en CSS, los selectores son utilizados para elegir los elementos HTML que se desean estilizar, estos apuntan a diferentes elementos HTML, más adelante veras como utilizarlos.

Sintaxis

Podemos Escribir CSS de diversas maneras, estas son:

- **Directo en un elemento HTML**: Se puede escribir CSS en un elemento HTML mediante la propiedad style.
  Ejemplo: <p style= “color: red; ”>Texto de prueba</p>
- **Mediante una etiqueta HTML**: También puede ser agregado también dentro de la etiqueta style.
  Ejemplo: <style> p{ color: red; } </style>

- **En un archivo CSS (archivo dedicado):** Podemos escribir CSS en un archivo de extensión dedicada, para esto primeramente debemos seguir 2 simples pasos:
- Crear un archivo con extensión CSS ejemplo: style.css
- Una vez creado se debe llamar al archivo CSS en el head de tu documento HTML a través de la siguiente etiqueta: <link rel="stylesheet" href="style.css">

¿Cómo utilizan los Selectores?

Los Selectores básicos en CSS son:

- **Selector HTML (nombre de etiqueta)**: este selector aplica el estilo a todos los elementos que coincidan con el elemento indicado.
  Ejemplo:

```
p {
  color: blue;
}
```

  Modificará todos los elementos de tipo “p” (párrafo).
- **Selector de clase (.)** : este selector aplica el estilo a todos los elementos que tienen la propiedad class con el valor “mi-clase”, es decir class=“mi-clase”.
  Ejemplo:

```
.mi-clase {
  font-size: 16px;
}
```

- **Selector de ID (#)** : este selector aplica el estilo a todos los elementos que tienen la propiedad id con el valor “mi-id”, es decir id=“mi-id”.
  Ejemplo:

```
#mi-id {
  background-color: green;
}
```

Los Selectores avanzados son:

- **Selector descendente (contenedor elemento)**: este selector aplica el estilo a los elementos dentro de un elemento principal.
  Ejemplo:

```
div p {
  color: gray;
}
```

  En este ejemplo se afectarán todos los elementos de tipo “p” (párrafo) dentro de los div.

- **Selector de hijo directo (>)** : este selector aplica el estilo solo a los hijos directos de un elemento principal.
  Ejemplo:

```
ul > li {
  list-style-type: disc;
}
```

  Aquí se afectan los hijos directos de tipo li de una lista desorganizada (ul).

- **Selector de atributo (elemento[atrib])** : este selector aplica el estilo a los elementos que tienen un atributo específico.
  Ejemplo:
  ```
  input[type="text"] {
    color: yellow;
  }
  ```
- **Selector de pseudo-clase (:)**: este selector aplica el estilo a un elemento en un estado específico (***se habla más a fondo en la sección: Pseudo-Clases***).
  Ejemplo:
  ```
  a:hover {
    color: blue;
  }
  ```
- **Selector de pseudo-elemento (::)** : este selector aplica el estilo a partes específicas de un elemento (***se habla más a fondo en la sección: Pseudo-Elementos***).
  Ejemplo:
  ```
  p::first-line {
    color: green;
  }
  ```

Estilos básicos

En un documento .css, los estilos se aplican a los elementos HTML seleccionados por los selectores, estos pueden incluir propiedades como el tamaño de fuente, color de fuente, margen y la ubicación del elemento, por mencionar algunos.

A continuación, veras algunos de los estilos básicos en CSS y que hacen:

- **color**: Se utiliza para aplicar color al texto de un elemento.
  Ejemplo:
  ```
  p{
    color: red;
  }
  ```

  dará color rojo al texto de los elementos tipo p (Párrafo)
- **background-color**: Se utiliza para dar un color de fondo a un elemento.
  Ejemplo:
  ```
  div{
    background-color: green;
  }
  ```

dará color verde al fondo de los elementos tipo div

- **margin**: Se utiliza para dar margen (espaciado exterior).
  Ejemplo:
  ```
  .caja{
    margin: 10px;
  }
  ```
  dará margen (espaciado exterior) de 10 pixeles a los elementos con la clase caja.
- **padding**: Se utiliza para espaciado interior.
  Ejemplo:
  ```
  #contenedor{
    padding:15px;
  }
  ```

  dará espaciado interno de 15 pixeles al elemento con el ID contenedor

Hasta este momento has visto como dar estilos básicos con diferentes selectores, pero existe una gran cantidad de estilos que puedes aplicar, seguiremos con una lista menos detallada, y si deseas ver más estilos puedes consultar la siguiente documentación

- **font-family**: establece el tipo de fuente que se utilizará en el texto.
- **font-size**: establece el tamaño del texto.
- **font-weight**: establece el grosor de la fuente.
- **text-align**: establece la alineación del texto dentro de su contenedor.
- **text-decoration**: establece la decoración del texto, como subrayado o tachado.
- **line-height**: establece la altura de línea del texto.
- **display**: establece el comportamiento de visualización del elemento, como block, flex o grid.
- **border**: establece el borde del elemento, incluyendo el ancho, estilo y color.
- **border-radius**: establece la curvatura de las esquinas del borde.
- **width**: establece el ancho del elemento.
- **height**: establece la altura del elemento.
- **background-image**: establece la imagen de fondo del elemento.
- **background-size**: establece el tamaño de la imagen de fondo.
- **box-shadow**: establece la sombra del elemento.
- **text-transform**: establece la transformación del texto, como mayúsculas o minúsculas.
- **text-shadow**: establece la sombra del texto.
- **opacity**: establece la opacidad del elemento.
- **cursor**: establece el estilo del cursor cuando se pasa por encima del elemento.
- **overflow**: establece cómo se maneja el contenido que desborda el elemento.
- **z-index**: establece la capa de profundidad del elemento en relación con otros elementos.

Unidades de medida

Existen varias unidades de medida que se pueden utilizar para definir tamaños y distancias. Las unidades de medida se dividen en dos categorías: unidades absolutas y unidades relativas.

Las unidades absolutas: son medidas fijas y no cambian en función del contexto o del tamaño de la pantalla.

Estas unidades son:

- **px**: píxeles, una unidad de medida absoluta que define un tamaño fijo en píxeles.
- **pt**: puntos, una unidad de medida absoluta que se utiliza principalmente en la impresión.
- **cm**: centímetros, una unidad de medida absoluta que se utiliza principalmente en la impresión.
- **in**: pulgadas, una unidad de medida absoluta que se utiliza principalmente en la impresión.

Las unidades relativas: son medidas que se definen en relación con otras medidas, como el tamaño de la pantalla o del elemento contenedor.

Estas unidades son:

- **em**: se define en relación con el tamaño de la fuente del elemento padre. Por ejemplo: si el tamaño de la fuente del elemento padre es de 16px, 1em será igual a 16px.
- **rem**: se define en relación con el tamaño de la fuente del elemento raíz (normalmente el elemento html). Esto significa que rem es una medida más estable y fácil de usar para tamaños de fuente y diseños escalables.
- **%:** porcentaje, se define en relación con el tamaño del elemento padre. Por ejemplo: si el ancho del elemento padre es de 200px, 50% será igual a 100px.
- **vw/vh**: se definen en relación con el ancho o la altura de la ventana gráfica del navegador. Por ejemplo: 1vw será igual al 1% del ancho de la ventana gráfica.
- **vmin/vmax**: se definen en relación con el ancho o la altura de la ventana gráfica, pero utilizan el valor más pequeño (vmin) o el valor más grande (vmax) para calcular la medida.

Flexbox y Grid

Flexbox y Grid son dos tecnologías que permiten crear diseños responsivos y flexibles.

Flexbox: Permite crear diseños flexibles y adaptables a diferentes tamaños de pantalla. En Flexbox, un contenedor flexible (display: flex) tiene elementos hijos flexibles (flex items) que se pueden ajustar y distribuir dentro del contenedor. Flexbox proporciona propiedades para controlar la dirección, el tamaño, la alineación y el orden de los elementos hijos.

Algunas de sus propiedades son:

- **display: flex** : Convierte el elemento en un elemento flexible, se suele usar en contenedores (div,aside,section,main,nav).
- **flex-direction**: Es una propiedad que define la dirección principal en la que se colocan los elementos dentro de un contenedor flexible. Es decir, esta propiedad determina en qué dirección se distribuyen los elementos flexibles dentro del contenedor.

  Hay cuatro posibles valores para esta propiedad:

  - **row**: los elementos se distribuyen horizontalmente en una sola fila. El valor por defecto es row.
  - **column**: los elementos se distribuyen verticalmente en una sola columna.
  - **row-reverse**: los elementos se distribuyen horizontalmente en una sola fila, pero en orden inverso al que se escriben en el código HTML.
  - **column-reverse**: los elementos se distribuyen verticalmente en una sola columna, pero en orden inverso al que se escriben en el código HTML.
- **justify-content**: Es una propiedad que se utiliza para **alinear horizontalmente** los elementos dentro de un contenedor flexible a lo largo del eje principal cuando se trate de **una fila** (flex-direction: row) **y verticalmente** cuando se trate de **una columna** (flex-direction: columna). Es decir, esta propiedad define cómo se distribuyen y se espacian los elementos dentro del contenedor a lo largo del eje principal.

  Hay seis posibles valores para la propiedad justify-content:

  - **flex-start**: los elementos se alinean al inicio del contenedor flexible.
  - **flex-en**d: los elementos se alinean al final del contenedor flexible.
  - **center**: los elementos se alinean en el centro del contenedor flexible.
  - **space-between**: los elementos se distribuyen con un espacio uniforme entre ellos, pero sin espacio en los extremos.
  - **space-around**: los elementos se distribuyen con un espacio uniforme alrededor de ellos, incluyendo el espacio en los extremos.
  - **space-evenly**: los elementos se distribuyen con un espacio uniforme entre ellos y en los extremos del contenedor.
- **align-items**: Es una propiedad que se utiliza para **alinear verticalmente** los elementos dentro de un contenedor flexible a lo largo del eje secundario cuando se trate de **una fila**

(flex-direction: row) **y horizontalmente** cuando se trate de **una columna** (flex-direction: columna). Es decir, esta propiedad define cómo se distribuyen y se espacian los elementos dentro del contenedor a lo largo del eje secundario.

Hay cinco posibles valores para la propiedad align-itemsent:

- **flex-start**: los elementos se alinean en el inicio del contenedor flexible.
- **flex-end**: los elementos se alinean en el final del contenedor flexible
- **center**: los elementos se alinean en el centro del contenedor flexible.
- **baseline**: los elementos se alinean en su línea base.
- **stretch**: los elementos se estiran para llenar el contenedor flexible.

- **flex-wrap**: Es una propiedad de Flexbox en CSS que determina si los elementos flexibles dentro de un contenedor flexible deben ajustarse a una sola línea o si se deben envolver en múltiples líneas para ajustarse al contenedor.

  Hay tres posibles valores para la propiedad flex-wrap:

  - **nowrap**: los elementos flexibles no se envolverán y se ajustarán a una sola línea, incluso si los elementos se desbordan del contenedor.
  - **wrap**: los elementos flexibles se envolverán automáticamente en múltiples líneas si no pueden ajustarse en una sola línea, y se distribuirán en varias filas.
  - **wrap-reverse**: los elementos flexibles se envolverán automáticamente en múltiples líneas si no pueden ajustarse en una sola línea, pero se distribuirán en filas invertidas en lugar de en orden normal.

Pseudo-Clases

Las pseudo-clases son palabras clave que se agregan a los selectores para indicar que se está seleccionando un elemento en un estado o condición particular, esto permite aplicar estilos a elementos en función de eventos o interacciones del usuario.

Se agregan al selector después del nombre del elemento y se escriben con dos puntos (:). Ejemplo: button:hover{…}

Algunas de las pseudo-clases más comunes son:

- **:hover**: selecciona el elemento cuando el usuario pasa el cursor sobre él.
- **:active**: selecciona el elemento cuando el usuario hace clic en él.
- **:focus**: selecciona el elemento cuando está enfocado.
- **:nth-child**(n): selecciona el elemento que es el "n-ésimo" hijo de su padre.
- **:first-child**: selecciona el primer hijo de su padre.
- **:last-child**: selecciona el último hijo de su padre.
- **:checked**: selecciona el elemento que está seleccionado en un conjunto de elementos (como un checkbox o un input tipo radio).

- **:disabled**: selecciona el elemento que está deshabilitado.

Todas estas pseudo-clases permiten aplicar estilos cuando los elementos reciban cierta interacción o se encuentren en algún estado en específico.

Pseudo-Elementos

Los pseudo-elementos son palabras clave que se agregan a un selector para seleccionar y estilizar partes específicas de un elemento. permiten agregar contenido o estilos a elementos que no existen en el documento HTML, como la primera letra o la selección de texto de un párrafo.

Se escriben con dos puntos (::) al inicio y se agregan al final del selector después del nombre del elemento o clase. Ejemplo: p::first-letter{…}

Algunos de los pseudo-elementos más comunes son:

- ::before: permite insertar contenido antes del contenido de un elemento.
- ::after: permite insertar contenido después del contenido de un elemento.
- ::first-letter: selecciona la primera letra de un elemento.
- ::first-line: selecciona la primera línea de un elemento.
- ::selection: selecciona el texto seleccionado por el usuario.

Transformaciones

Son un conjunto de propiedades que permiten modificar la apariencia de un elemento en una página web, se utilizan con la propiedad transform.

transform: Es la base de todas las transformaciones. Puede usarse para aplicar una o varias transformaciones a un elemento. Las transformaciones se especifican mediante una serie de funciones separadas por espacios.

Por ejemplo: transform: rotate(45deg) scale(1.5).

- **rotación(rotate)**: Esta propiedad permite girar un elemento en su lugar. Se especifica el ángulo de giro en grados (deg).
  Por ejemplo: rotate(45deg) gira un elemento 45 grados en sentido del reloj.
- **escalado(scale)**: Esta propiedad permite cambiar el tamaño de un elemento en base al factor de escala indicado.
  Por ejemplo: scale(1.5) aumenta el tamaño de un elemento 1.5 veces el tamaño original, es decir, un 50%. Lo mismo para disminuir scale:(0.5), disminuirá el tamaño del elemento a 0.5 del tamaño original, es decir, recude un 50%.
- **traslación(translate)**: Esta propiedad permite mover un elemento en el plano horizontal y vertical. Se especifica la distancia de traslación en píxeles o porcentajes.

Por ejemplo: translate(50px, 20%) mueve un elemento 50 píxeles hacia la derecha y un 20% hacia abajo. También se pueden usar valores negativos.

- **deformación(skew)**: Esta propiedad permite deformar un elemento en el plano horizontal y vertical. Se especifica el ángulo de deformación en grados.
  Por ejemplo: skew(30deg, 20deg) deforma un elemento 30 grados en sentido horario en el eje horizontal y 20 grados en sentido antihorario en el eje vertical.

Overflow

La propiedad **overflow** se utiliza para controlar cómo se maneja el contenido que sobresale de un contenedor con límites definidos.

Hay cuatro valores posibles para la propiedad overflow:

- **visible**: Es el valor por defecto. El contenido que sobresale del contenedor se muestra fuera de los límites del contenedor.
- **hidden**: El contenido que sobresale del contenedor se recorta y no se muestra.
- **scroll**: Se agrega una barra de desplazamiento al contenedor, permitiendo al usuario desplazarse hacia arriba y hacia abajo para ver el contenido que sobresale
- **auto**: Se agrega una barra de desplazamiento al contenedor solo cuando es necesario, es decir, cuando el contenido sobresale del contenedor.

Transiciones

Las transiciones permiten cambiar gradualmente los valores de las propiedades de un elemento en un período de tiempo determinado. Con las transiciones, podemos crear animaciones sutiles y elegantes en nuestras páginas web sin necesidad de usar JavaScript o frameworks externos.

Sintaxis:

```
selector {
  propiedad: valor-inicial;
  transition: propiedad duración [tipo-de-timing-function];
}

selector:hover {
  propiedad: valor-final;
}
```

```
p{
  color: gray;
  transition: color 2s linear;
}

p:hover {
  color: black;
}
```

Es decir:

Esta transicion llevara el texto del parrafo desde el color gris hasta el color negro, durara 2 segundos y se realizara de manera linear, cuando el parrafo tenga el cursor arriba, es decir cuando se encuentre en estado :hover.

En la propiedad **transition** se especifica forma en la que se va a realizar el efecto animado, aquí se define la duración de la animación y, opcionalmente, el tipo de timing function que se usará para la animación.

Hay cuatro valores que se pueden usar en la propiedad transition:

- **transition-property**: especifica la propiedad CSS que se va a animar.
- **transition-duration**: especifica la duración de la animación.
- **transition-timing-function**: especifica la función de tiempo que se usará para la animación. Los valores comunes son ease, linear, ease-in, ease-out, y ease-in-out.
- **transition-delay**: especifica un retraso antes de que comience la animación.

También se pueden agregar múltiples animaciones separando cada una con una coma.

Nota: Es importante recalcar que para que surtan efecto la propiedad a animar debe tener un valor iniciar y un valor final.

```
selector {
    propiedad1: valor-inicial;
    propiedad2: valor-inicial;
    propiedad3: valor-inicial;
    transition:
        propiedad1 duración [tipo-de-timing-function],
        propiedad2 duración [tipo-de-timing-function],
        propiedad3 duración [tipo-de-timing-function],;
} <- #1-9 selector

selector:hover {
    propiedad1: valor-final;
    propiedad2: valor-final;
    propiedad3: valor-final;
} <- #11-15 selector:hover
```

```
div{
    color: red;
    width: 100px;
    height: 100px;
    transition:
        color 2s linear,
        width 1500ms ease,
        height 2000ms ease-in-out;
} <- #63-71 div

div:hover {
    color: blue;
    width: 200px;
    height: 150px;
} <- #73-77 div:hover
```

Animaciones

Las animaciones permiten crear transiciones más complejas y efectos de animación en elementos HTML. Una animación se define utilizando la regla @keyframes y se aplica a un elemento utilizando la propiedad **animation**.

Algunas de las propiedades que se pueden definir utilizando la propiedad **animation**:

- **animation-name**: especifica el nombre de la animación definida en @keyframes.
- **animation-duration**: especifica la duración de la animación, en segundos o milisegundos (s, ms).
- **animation-timing-function**: especifica la función de tiempo que se utilizará para controlar la velocidad de la animación.
    1. **linear**: La velocidad de la animación es constante a lo largo del tiempo.
    2. **ease**: La animación comienza lentamente, acelera a medida que se mueve a través de la mitad del recorrido y desacelera al final.
    3. **ease-in**: La animación comienza lentamente y se acelera rápidamente hacia el final.
    4. **ease-out**: La animación comienza rápidamente y se desacelera hacia el final.
    5. **ease-in-out**: La animación comienza lentamente, se acelera a medida que se mueve a través de la mitad del recorrido y luego se desacelera de nuevo hacia el final.
    6. **cubic-bezier(n,n,n,n)**: Permite personalizar una función de tiempo utilizando valores numéricos que controlan la curva de aceleración y desaceleración. Se pueden especificar cuatro valores entre 0 y 1, que definen los puntos de control de la curva.
- **animation-delay**: especifica el tiempo de espera antes de que comience la animación. (s, ms)
- **animation-iteration-count**: especifica el número de veces que se repetirá la animación.
- **animation-direction**: especifica la dirección de la animación
    1. **normal**: La animación se reproduce en la dirección predeterminada, desde el inicio hasta el final.
    2. **reverse**: La animación se reproduce en dirección inversa, desde el final hasta el inicio.
    3. **alternate**: La animación se reproduce de ida y vuelta, es decir, se reproduce en la dirección normal en la primera iteración, en la dirección inversa en la segunda, y así sucesivamente.
    4. **alternate-reverse**: La animación se reproduce de ida y vuelta, comenzando desde la dirección inversa en la primera iteración.
- **animation-play-state**: especifica si la animación está en pausa o en reproducción.
    1. **paused**: La animación se detiene en su lugar.
    2. **running**: La animación se reproduce normalmente.

Sintaxis:

```css
.elemento {
    animation-name: mover;
    animation-duration: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
    animation-timing-function: ease;
    animation-delay: 1.5s;
} <- #64-71 .elemento

.elemento:hover{
    animation-play-state: paused;
}

@keyframes mover {
    from {
        transform: translate(0px 0px);
    }

    to {
        transform: translate(100px 0px);
    }
} <- #77-85 @keyframes mover
```

En este ejemplo, se define una animación llamada mover utilizando la regla @keyframes. La animación mueve un elemento hacia la derecha utilizando la propiedad transform: translate().

Comienza en la posición original del elemento con translate(0px 0px) y se mueve 100px hacia la derecha con translate(100px 0px).

Se aplica al elemento con la clase .elemento utilizando la propiedad animation.

Se especifica el nombre de la animación (en este caso se llama mover), la duración de la animación (2s) y el número de veces que se repetirá la animación (infinite), la dirección de la animación (alternate), la función de tiempo (ease) y un retraso de inicio desde que se carga la página (1.5s).

Adicionalmente, cuando el elemento con la clase .elemento este en el estado :hover, se pausara la animación en el punto que se llegue a encontrar y se reanudara cuando el cursor salga del elemento.