PROTECO

CSS AVANZÁDO

VARIABLES

DECLARACIÓN

PROTECO

- Conocidas como **propiedades personalizadas**, se establecen mediante la notación de propiedades personalizadas, por ejemplo:

```
--colorAzul: blue;
```

- Se acceden mediante la función **var()**, por ejemplo:

```
color: var(--colorAzul);
```

ÁMBITO (SCOPE)

PROTECO

- Ten en cuenta que el selector que usemos para las reglas de estilo define el ámbito **(scope)** en el que podremos usar la **variable**.

- Una buena práctica común es declarar variables en la pseudo-clase **:root** y así aplicarlas globalmente al documento HTML:

```
:root {
    --main-bg-color: brown;
}
```

HERENCIA

- Las variables pueden **heredarse** si no se establece **ningún valor** en un elemento dado, por lo cual se utiliza el valor de su elemento padre:

```
<div class="uno">
    <div class="dos"><div>
    <div class="tres"><div>
<div>
```

```
.uno {
    --test: 10px;
}
```

```
.dos {
    --test: 2em;
}
```

- El elemento con clase **uno** cuenta con un **--test** de **10px**
- El elemento con clase **dos** cuenta con un **--test** de **2em**
- El elemento con clase **tres** cuenta con un **--test** de **10px**

VALORES DE SUSTITUCIÓN (FALLBACK)

PROTECO

- Utilizando **var()** podemos definir múltiples valores de sustitución que se usarán cuando la variable dada **no se encuentre definida** aún.

- El primer argumento a la función es el nombre de la variable que se va a sustituir mientras que el segundo argumento es un valor de reserva, ejemplo:

```
.titulo {
    color: var(--my-var, red);
}
```

VALORES DE SUSTITUCIÓN (FALLBACK)

PROTECO

- Otra forma es colocar como segundo parámetro otra variable y de igual forma colocar el valor de sustitución:

```
.titulo {
    color: var(--my-var, var(--my-background, red) );
}
```

CONSIDERACIÓN

PROTECO

* Los valores de sustitución no se usan para arreglar problemas de compatibilidad del navegador. Es simplemente un respaldo para que aquellos navegadores que sí soportan variables de CSS puedan elegir un valor diferente en caso de que la variable no se haya definido o contenga un valor no válido.

PSEUDO CLASES

¿QUÉ ES?

PROTECO

- Es una **palabra clave** que se añade a los selectores y que especifica un **estado especial** del elemento seleccionado.

- Estas permiten **aplicar un estilo a un elemento** no sólo en relación con el contenido del árbol de documento, sino también en relación a **factores externos** como el historial del navegador, el estado de su contenido, o la posición del ratón.

ESTRUCTURA

PROTECO

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

TIPOS

- :active
- :checked
- :default
- :dir()
- :disabled
- :empty
- :enabled
- :first
- :first-child
- :link
- :not()
- :nth-child()
- :nth-last-child()
- :nth-last-of-type()
- :nth-of-type()
- :only-child
- :only-of-type
- :optional
- :out-of-range
- :first-of-type
- :fullscreen
- :focus
- :hover
- :indeterminate
- :in-range
- :invalid
- :lang()
- :last-child
- :last-of-type
- :left
- :read-only
- :read-write
- :required
- :right
- :root
- :scope (en-US)
- :target
- :valid
- :visited

PSEUDO ELEMENTOS

¿QUÉ SON?

PROTECO

- Son similares a las pseudoclases con la diferencia de que actúa como si se **agregara un elemento HTML** en lugar de haberse aplicado una clase nueva a los elementos presentes.

- No describen un estado especial sino que permiten añadir estilos a una **parte concreta** del documento

TIPOS

- ::after
- ::before
- ::first-letter
- ::first-line
- ::selection
- ::backdrop
- ::placeholder
- ::marker
- ::spelling-error
- ::grammar-error (en-US)

Experimentales:
Su uso no es del todo funcional en la mayoría de navegadores por lo que queda bajo responsabilidad del desarrollador su implementación

ESTRUCTURA

PROTECO

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

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

PROTECO

INTRODUCCIÓN A SASS

¿QUÉ ES SASS?

PROTECO

- Syntactically Awesome Style Sheets **(SASS)** es un **preprocesador** de código CSS que nos permite generar de manera automática, hojas de estilo, añadiendoles características que no tiene CSS, y que son propias de los **lenguajes de programación**.

- Las principales características con las que cuenta son:
  - **Variables**
  - **Nesting (Anidación)**
  - **Parent Selector (Selector Principal)**
  - **Mixins**
  - **Funciones**
  - **Herencia**

CARACTERÍSTICAS BÁSICAS

PROTECO

- SASS dispone de 2 formatos diferentes para sintaxis, traduciéndose en 2 tipos de archivos diferentes:
  - **.sass (Con una sintaxis sin llaves o puntuaciones)**
  - **.scss (Con una sintaxis identica a CSS)**

- Tanto la sintaxis de **.sass** como la de **.scss** no pueden ser interpretadas directamente por los navegadores, por lo debe ser compilada para traducir el archivo SASS en uno de tipo CSS.

INSTALACIÓN

PROTECO

- Existen varias formas de poder instalar Sass en nuestro sistema operativo, tanto de forma por interfaz gráfica como también por línea de comandos mediante un **gestor de paquetes**, como se muestra a continuación:

  - **Utilizando npm:** npm install -g sass
  - **Utilizando choco:** choco install sass
  - **Utilizando brew:** brew install sass/sass/sass
  - **Utilizando gem:** gem install sass **(obsoleta)**

DESISTALACIÓN

PROTECO

- De igual forma es posible desinstalar Sass de nuestros equipos, siendo los casos más recurrentes cuando necesitemos otra versión o simplemente no deseemos trabajar más con el:

  - **Utilizando npm:** npm uninstall -g sass
  - **Utilizando choco:** choco uninstall sass
  - **Utilizando brew:** brew uninstall sass/sass/sass
  - **Utilizando gem:** gem uninstall sass

VARIABLES

PROTECO

- Su declaración es de la forma:

```
$nombre: valor o propiedad
```

- Para acceder a la variable desde el mismo archivo basta con nombrarla, pero si se desea utilizar desde otro archivo es necesario importar el archivo con **@use, @forward (o @import)** colocando después lo siguiente:

```
propiedad: nombreArchivo.$variable;
```

@USE

PROTECO

- Cuando se deseen utilizar el contenido de un archivo en otros archivos, se deberá utilizar la palabra reservada **@use** seguido de la **ubicación y nombre del archivo entre comillas**, ejemplo:

```
@use ‘rutaRelativa/variables’;
```

ALIAS Y SELECTOR UNIVERSAL

PROTECO

- Si se desea colocar un **alias** al **nombre del archivo** a importar, se puede colocar seguido del nombre de la siguiente forma:

```
@use ‘rutaRelativa/archivo as alias;
```

- En dado caso de no querer utilizar un alias y solamente llamar al contenido por como fue hecho se utiliza *:

```
@use ‘rutaRelativa/archivo’ *;
```

@FORWARD

PROTECO

- De forma similar podemos utilizar las variables o inclusive funciones, mixins o entends de un archivo que las importe a otro, utilizando **@forward** como se muestra:

Archivo: componentes.scss

```
@forward 'rutaRelativa/colores';

$color--rojo: red;
```

Archivo: styles.scss

```
@use 'rutaRelativa/componentes' as cp;

selector{
    background-color: cp.$color--rojo;
}
```

- Y de igual forma que **@use** es posible colocarle un **alias**.

NESTING

PROTECO

- Cuando se necesite utilizar **especificidad** para aplicar un estilo se puede utilizar la **anidación**, donde cada **selector descendiente** deberá colocarse dentro de cada **selector superior** donde se encuentre, contando con una estructura:

```
selectorPadre{
    ...
    selectorHijo{
        ...
        selectorNieto{
            ...
        }
    }
}
```

PARENT SELECTOR

PROTECO

- Se usa en selectores anidados para referirse al **selector externo**. Cuando esto sucede se reemplaza con el selector externo correspondiente.
- Para utilizarlo se debe colocar & como anidación:

```
selectorPadre{
    ...
    &selectoranidado{
        ...
    }
}
```

- Puede utilizarse con pseudoclases o pseudoelementos.

MIXINS

PROTECO

- Permiten definir **propiedades o estilos** en selectores que pueden ser **reutilizables** para ciertas partes de nuestros estilos, contando con la siguiente estructura:

```
@mixin nombreMixin(argumento: valorPredeterminado) {
    propiedades CSS
}

...

selector {
    @include nombreMixin();
    propiedades CSS
}
```

HERENCIA (CLASES PLACEHOLDER CON EXTENDS)

- Son similares a los mixins, con la diferencia de que **no es posible pasar parámetros** a los selectores o clases:

```
%nombreExtend {
    propiedades CSS
}

...

selector {
    @extend %nombreExtend;
}
```

- Esto nos permite **heredar** estructuras de código a otras.

FUNCTIONS

PROTECO

- Las funciones son similares a los Mixins ya que ambos pueden acceder a **variables globales** que estén declaradas en el archivo y pueden aceptar argumentos, sin embargo es necesario siempre **retornar el argumento** como se muestra:

```
@function nombreFuncion(argumento: valorPredeterminado) {
    propiedades CSS
    @return argumento;
}

selector {
    propiedad: nombreFuncion(argumento);
    propiedades CSS
}
```

NO UTILIZAR @IMPORTANT

PROTECO

- Es posible usar **@important** en lugar de **@use** y **@forward** para obtener la **información** de todos los archivos, sin embargo al hacerlo trae como consecuencias:

  - Todas las hojas de estilos se **compilan**.
  - Todas las variables, mixins, extends y funciones se convertirán en **globales**.

  Por lo que su uso **no es recomendable**.