ThePower business school

Variables CSS 📝🎨

Las denominadas **variables CSS** realmente no son variables, sino que solemos denominarlas así por su parecido con la programación y porque es más fácil comprender rápidamente lo que son, pero tienen sus diferencias. Lo adecuado sería llamarlas custom properties.

Estas custom properties son un mecanismo que nos permite dar un valor personalizado a las propiedades y guardarlo o almacenarlo. Esto nos evitará repetir código y recordarlo con un nombre adecuado. Esto además nos facilitará en un futuro los cambios, ya que solo habrá que modificar los valores en un lugar y los cambios se verán reflejados automáticamente en todos los elementos en los que se use la variable.

Para declarar custom properties (variables) usaremos un nombre que comience con dos guiones (`-`), y un valor que puede ser cualquier valor válido de CSS.

```
elemento {
  --main-bg-color: red;
}
```

Aquí hay que tener en cuenta el scope o ámbito de aplicación de esta variable. Lo correcto sería declarar las variables globales de nuestra aplicación en un ámbito o scope global, para poder acceder a ellas desde cualquier punto. Esto se consigue con la pseudo-clase :root

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

Ahora podremos acceder a esta custom property desde cualquier lugar de la app, como por ejemplo en los <p> de la aplicación:

```
p {
  background-color: var(--main-bg-color);
}
```

Además, es conveniente añadir un valor de la propiedad por defecto por si la variable desapareciese o no fuese accesible desde nuestro scope. Imaginemos en el ejemplo anterior que, por lo que sea, la variable de :root desaparece, nuestro ejemplo se quedaría sin color de fondo, para solucionarlo podemos hacer lo siguiente:

```
p {
  background: var(--background-color, red);
}
```

Es importante recalcar que se necesitará introducir la variable dentro de los () precedidos de la palabra clave var que indicará que se trata de una variable. Así, la palabra clave VAR recibirá 2 parámetros, la variable que queremos usar será el primero y un valor a modo de salvavidas por si la variable no funcionase.

Vamos a ver para terminar un ejemplo de ámbitos o scope porque al principio puede que os cueste un poco entenderlo, pero os será de mucha utilidad:

```
<div class="parent">
  <div class="first child">First child</
div>
  <div class="second child">Second child
</div>
</div>
<div class="third child">Third child</di
v>
```


```
.parent {
  --background-color: red;
  color: white;
}

.first {
  --background-color: lightblue;
}

.child {
  background: var(--background-color, vi
olet);
}
```

Aquí podéis ver el resultado, vamos a analizarlo.

- En primer lugar, el primer div será de color lightblue debido a que, aunque en el padre que tiene la clase parent se indica la variable, en el propio div con el texto “First child” se indica otro valor para la variable que es lightblue en la clase first, lo que indica que tiene mayor prioridad y por lo tanto se sobreescribirá.

- El segundo div será de fondo rojo ya que en la clase second no se indica un valor distinto para la variable, por lo que prevalece el valor que tiene inicialmente la clase parent.
- Por último, el tercer div será de color violet debido a que, al no ser hijo del elemento con la clase parent, no puede acceder a ese ámbito o scope y, por lo tanto, no podrá acceder a ningún valor de la variable, con lo que se mostrará el color por defecto violet.

¿Qué hemos aprendido?

1. Qué son las variables CSS y cómo se definen y usan.
2. El ámbito o scope de las variables.