

Bootcamp Full Stack

CSS avanzado



Custom properties



¿Qué es una custom property?

Siguiendo con el uso de elementos, funciones y demás que se veían sólo en preprocesadores, **CSS incorpora el uso de *custom properties* que no son ni más ni menos que variables.**

```
h2 {  
    --fuente: 1.5rem;  
    margin-bottom: var(--fuente);  
    text-transform: capitalize;  
    font-size: var(--fuente);  
}
```

Lo más interesante es la posibilidad de no repetir millones de veces los mismos elementos e incluso podemos usar estas propiedades personalizadas de formas más complejas como en este ejemplo:

```
h2 {  
    --r: 100;  
    --g: 200;  
    --b: 150;  
    --a: 1;  
    background: rgba(var(--r), var(--g), var(--b), var(--a));  
}
```

A partir de esto, las posibilidades a explorar son infinitas, dado que de forma más simple que **Sass** y el uso de *mixins*, nos permite reutilizar y cambiar un parámetro en una función como vemos en el siguiente ejemplo:

```
h2 {  
    --r: 100;  
    --g: 200;  
    --b: 150;  
    --a: 1;  
    background: rgba(var(--r), var(--g), var(--b), var(--a));  
}  
  
h2:hover {  
    --g: 50;  
}
```



En el trabajo con **fondos múltiples o elementos que son similares pero deben ir cambiando**, sin demasiadas complejidades como declaración de variables o importaciones, *mixins* y otras funciones que tenemos en preprocesadores **nos ayudan a trabajar de forma rápida y eficaz.**

```
body {  
    --fondo1: url("./img/imagen1.svg");  
    --fondo2: url("./img/imagen2.svg");  
  
    background-image: var(--fondo1), var(--fondo2);  
}
```

Usos avanzados de custom properties

Se pueden trabajar las *custom properties* para indicar, como en los *mixins* de Sass, una parte específica del valor **total**, veamos un ejemplo con una sombra donde podemos modificar dinámicamente la extensión de la misma.

```
▽ button {  
    width: 100px;  
    height: 50px;  
    --spread: 2px;  
    box-shadow: 0 0 20px var(--spread) □rgba(0,0,0,0.5);  
}  
▽ button:hover {  
    --spread: 5px;  
}
```

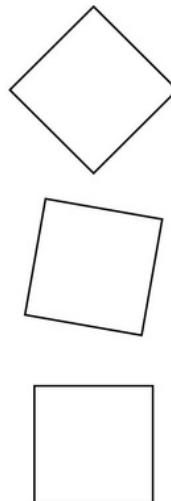
Luego, podremos fácilmente agregar más características, donde el botón se verá de manera distinta sea que hacemos *click*, *hover* o simplemente lo vemos sin ningún estado en particular.

Botón grande

Botón grande

```
index.html X # estilos.css X
css > # estilos.css > button:active
38     width: 100px;
39     height: 50px;
40     --spread: 2px;
41     --color: #rgba(0,0,0,0.5);
42     box-shadow: 0 0 20px var(--spread) var(--color);
43 }
44
45
46 button:hover {
47     --spread:3px;
48
49     --color: #rgba(0,0,0,0.5);
50 }
51
52
53 button:active {
54     --spread: 5px;
55     --color: #rgba(255,0,255,0.5);
56 }
57
58
59
```

En un elemento que va a transformarse de forma distinta, por ejemplo, una rotación con diferentes grados se puede integrar el uso de *custom properties* para facilitar el trabajo.



```
<> index.html | # estilos.css X  
css > # estilos.css > div.segundo  
1  div {  
2      width: 100px;  
3      height: 100px;  
4      --grados: 45deg;  
5      transform: rotate(var(--grados));  
6      border: 2px solid black;  
7  }  
8  
9  
10 div.primero {  
11     --grados: 10deg;  
12 }  
13  
14  
15 div.segundo {  
16     --grados: 180deg;  
17 }
```

Uso de cascada en custom properties

Las **custom properties** usan el concepto de **cascada** eso significa, de forma simple que en el siguiente HTML tenemos al elemento **main** más cercano al **section**, por tanto el color del section será gray, porque ese es el color del elemento más cercano. En base a eso, **podemos configurar diversos elementos sin repetir valores teniendo en cuenta esta cascada o jerarquía.**

Veamos un ejemplo en ésta y la siguiente pantalla.

```
<body>
  <main>
    <section>Mi texto </section>
  </main>
</body>
```

Veamos el **.css** del bloque HTML de la pantalla anterior y su resultado:

```
index.html X # estilos.css X
css > # estilos.css > section
1 body {
2   --background: red;
3 }
4 main {
5   --background: gray;
6 }
7 section {
8   background: var(--background);
9 }
```



¿Por qué, a veces, se usa :root?

Muchas veces vemos que **las custom properties son declaradas en el elemento :root**. La realidad es que es lo mismo que declararlas en el elemento **html** o **body**, pero quizás en cuanto a prolijidad y orden puede que **organice mejor la estructura** y no está mal inicializar las variables en ese elemento al comienzo de todo.

Nota: es importante recordar que el uso de cascada en custom properties es una gran herramienta, por tanto, no siempre declarar todo al comienzo o no sobreescribir algunos valores es un uso común o real de las variables.

```
index.html # estilos.css ...
css > # estilos.css > ...
1  :root {
2  |   --color: red;
3  }
4
5  /*es lo mismo que*/
6
7
8  body {
9  |   --color: red;
10 }
11
```

