

# sass Scss curso

---

## comando para transformar sass a css

tenemos que ejecutar este comando obligatoriamente para que pueda leer el archivo scss y convertirlo a css que lo entienden los navegadores sin esto nuestros archivos scss no funcionarían como hemos dicho antes los archivos sass no los entiende el navegador por eso hay que referenciar a un archivo css tendremos que poner la ruta completa

```
sass --watch src/assets:public/css
```

por supuesto que en nuestro archivo tendremos que poner el link que hace referencia a nuestra hoja de estilos css que el navegador si que entiende

```
<link rel="stylesheet" href="/css/style.css" />
```

## asi podemos definir variables

```
$color-primary: red;

h1 {
  color: $color-primary; // se define con $nombre variable
  background-color: black;
}
```

si la ponemos al principio del documento sera una variable global

## modularizar nuestro codigo

Podemos crear archivos sin que sean compilados a css para tener un código mas organizado para ello debemos crear un archivo de **scss** con este formato **\_variables.scss** despues en nuestro archivo sass principal debemos usar la palabra clave **use**

```
@use; _nombre de archivo sin extension;
```

## veamos otro ejemplo

Tenemos un caso especial imagina que definimos un archivo con todas las variables **\_variables.scss** nosotros usamos el **use \_nombre del archivo** y colocamos las variables donde nosotros queramos pero hay un problema no se aplicaran las variables porque nuestro modulo es de variables para poder utilizarlo usamos la siguiente sintaxis

```
@use "_variables"; // usamos el archivo de variables

h1 {
  color: variables.$color-text; // para acceder a la variable ponemos el nombre
del fichero.$variable
  background-color: variables.$color-primary;
}
```

veamos un paso a poso

1. creamos nuestro archivo con esta sitaxis **\_nombreArchivo.scss**
- 2.ponemos lo que queremos en el archivo
2. despues nos vamos al archivo principal y usamos la palabra clave @use '\_nombreArchivo.scss'
3. por otra parte si hacemos un archivo de variables **\_variables.scss** debemos de utilizar el @use **nombreArchivo.scss** para que funcione debemos usar el siguiente codigo

```
h1 {
  color: variables.$color-text; // para acceder a la variable ponemos el nombre
del fichero.$variable
  background-color: variables.$color-primary;
}
```

## Nesting

Imaginemos que tenemos un menu de navegacion y queremos acceder al elemento **a** esta seria la forma sin nesting

```
nav ul li a {
  text-decoration: none; // manera tradicional
}
```

### Usando nesting

```
nav {
  background-color: blue;
  padding: 1em;
  ul {
    color: black;
  }
  li {
    list-style: none;
  }
  a {
    text-decoration: none;
    color: red;
  }
}
```

```
}  
}
```

Ahora le decimos que el **nav** tendra un color de fondo con un padding dentro del nav esta el elemento **ul** con un color negro

dentro de ese un elemnto **li** le quitamos el estilo de lista y dentro del el esta el elemento **a** con un text-decoration y un color rojo

de esta manera nuestro scss quedara de una manera mucho mas limpia

veamos como lo transforma a css

```
nav {  
  color: blue;  
}  
  
nav {  
  background-color: blue;  
  padding: 1em;  
}  
nav ul {  
  color: black;  
}  
nav li {  
  list-style: none;  
}  
nav a {  
  text-decoration: none;  
  color: red;  
}
```

Como podemos ver con el nesting se entiende y se ve el codigo mucho mas rapido y es mas facil de leer pero esta forma no es la adecuada

Usanndo el selector padre &

### Un ejemplo

```
<template>  
  <nav class="nav__container">  
    <a href="ejemplo">ejemplo</a>  
    <a href="ejemplo2">ejemplo</a>  
    <a href="ejemplo3">ejemplo</a>  
    <a href="ejemplo4">ejemplo</a>  
  </nav>  
</template>
```

Cundo nosotros estamos utilizando este selector **&** en nuestro archivo scss es una forma corta de escribir

```
.nav
```

⚠ **Es muy importante que nuestro elemento sea una clase o un id si no nunca funcionará** ⚠

Esta sería la mejor manera de escribirlo

```
.nav {  
  background-color: cadetblue;  
  
  &__container {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    gap: 1em;  
    width: 100%;  
    height: 80px;  
    background-color: chocolate;  
  }  
}
```

asi quedaria compilado en css

```
.nav {  
  background-color: cadetblue;  
}  
.nav__container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  gap: 1em;  
  width: 100%;  
  height: 80px;  
  background-color: chocolate;  
}
```

como podemos observar **&** hace referencia a **.nav**

Agrengando media query

```
.nav {  
  background-color: cadetblue;  
  
  &__container {  
    display: flex;
```

```
        justify-content: center;
        align-items: center;
        gap: 1em;
        width: 100%;
        height: 80px;
        background-color: rgb(84, 84, 220);
    }
    &__item {
        text-decoration: none;
        color: rgb(29, 29, 35);
        font-size: 1em;
    }

    &__item:hover {
        color: white;
    }

    @media only screen and (min-width: 800px) {
        &__container {
            display: flex;
            justify-content: end;
        }
    }
}
```