Funciones adicionales

Módulo 03

Implementación SASS

Implementación

SASS

Este preprocesador posee una función super interesante llamada @extend.

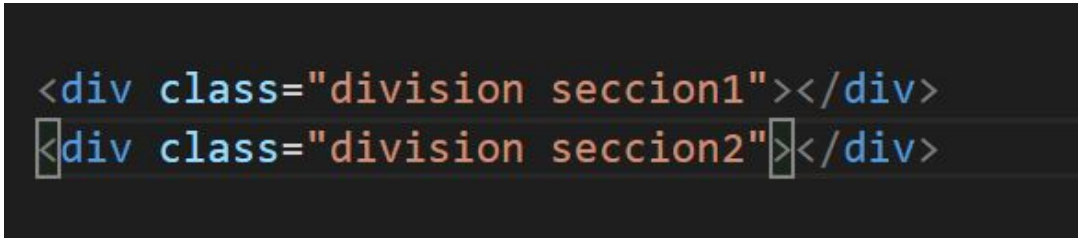
Esta función permite **compartir propiedades de un selector a otro,**

```
.caja { display: flex; justify-content: center; align-items:center;}

.seccion1 {background-color: blue;  display: flex; justify-content: center;
align-items:center;}

.seccion2 { background-color: red; display: flex; justify-content: center;
align-items:center;}
```

Implementación

SASS

Sin embargo en el paso anterior la típica solución sería asignarle la clase **.division en el .html** a todos **aquellos elementos que la necesiten,**



De esta manera, podemos utilizar esas características pero termina muchas veces siendo molesto de utilizar, por eso ofrecemos otra solución,

Implementación

SASS

De esta manera utilizamos **@extend para trabajar,**

```
.caja { display: flex; justify-content: center; align-items:center;}

.seccion1 {background-color: blue; @extend .caja; }

.seccion2 { background-color: red;  @extend .caja;}
```

Implementación

SASS

El resultado del paso anterior, luego de compilado será

```
.caja, .seccion1, .seccion2 {
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-pack: center;
        -ms-flex-pack: center;
            justify-content: center;
    -webkit-box-align: center;
        -ms-flex-align: center;
            align-items: center;
}
```

Implementación

SASS

Si bien el código anterior funcionará perfectamente **SASS** en su última versión **ha cambiado un poco su sintaxis en la clases,**

index.html estilos.scss

css > estilos.scss > ...

```scss
%encabezados{
    background-color: rgb(233, 196, 141);
    color: rgb(236, 97, 16);
    font-family:Verdana, Geneva, Tahoma, sans-serif;
}


h1 {
  @extend %encabezados;
  font-size: 3em;
}

h2 {
  @extend %encabezados;
  font-size: 1.7em;
}
```

Implementación

SASS

En el ejemplo anterior, vemos que se han sumado algunos elementos adicionales.

Es que en la compilación se agregan algunos hacks para evitar problemas en algunas versiones de algunos navegador.

Si bien hoy el uso no es tan necesario, no afectará en nada el hecho de dejarlos, **utilizando la ventaja de que ni siquiera tuvimos que pensarlos o escribirlos nosotros, ya que el propio compilador lo hizo.**

Implementación

SASS

También podemos trabajar **con pseudo clases o pseudo elementos** de forma dinámica, por ejemplo **si tenemos un botón,**

```
$ancho : 200px;
$alto : 30px;
$color: blue;
.button {

width: $ancho;
height: $alto;
background-color: $color;}
```

Implementación

SASS

El código anterior podemos trabajarlo desde **sass** no dará el siguiente resultado luego de compilarlo,

```
.button {
  width: 200px;
  height: 30px;
  background-color: blue;
}
/*# sourceMappingURL=estilos.css.map */
```

Implementación

¿Cómo empezar a trabajar?

Ahora vamos a implementar la pseudo **clase :hover,**

```
$ancho : 200px;
$alto : 30px;
$color: blue;
.button {

width: $ancho;
height: $alto;
background-color: $color;

&:hover {

background-color: green;


}

    }
```

Implementación

¿Cómo empezar a trabajar?

El paso anterior luego de compilado será el siguiente en el .css

```css
.button {
  width: 200px;
  height: 30px;
  background-color: blue;
}

.button:hover {
  background-color: green;
}
```

Implementación

sass

De la misma forma podríamos **haber trabajado con un pseudo elemento,**

```
.button {

  width: $ancho;
  height: $alto;
  background-color: $color;

&:hover {

  background-color: green;


  }

&::before {
  content: '.'

  }

      }
```

Implementación

sass

El resultado del código anterior, luego de compilado será el siguiente,

```
.button {
  width: 200px;
  height: 30px;
  background-color: blue;
}

.button:hover {
  background-color: green;
}

.button::before {
  content: '.';
}
```

Implementación

sass

También podemos añadir **@media** a nuestros estilos desde sass,

```
$ancho : 200px;
$alto : 30px;
$color: blue;
.button {

width: $ancho;
height: $alto;
background-color: $color;


@media (max-width: 765px){width: $ancho / 2}
&:hover {background-color: green;}

&::before {content: '.'}

}
```

Implementación

sass

El resultado del código anterior, luego de compilado será el siguiente,

```
@media (max-width: 765px) {
  .button {
    width: 100px;
  }
}
```

Evidentemente esta forma de implementación de **@media es más corta y más sencilla que la que usualmente debemos realizar**

Implementación

sass

Podemos también en SASS trabajar con funciones con retorno,

```
@function multiplicar(x, y) {
  @return x * y;
}

p {
  margin-top: multiplicar(10px, 2);
}
```

Evidentemente sass nos permite movernos como si estuviésemos trabajando **con JS sin ningún tipo de diferencia o desventaja.**

Implementación

sass

Es importante saber qué **SASS** permite trabajar con @function para funciones más complejas como por ejemplo la siguiente tomada de la página oficial de **SASS.** Este formato de función es más similar al que encontramos en lenguajes de programación con **JS** siendo posible el trabajo con return en este caso **@return. El ejemplo debajo incluye una estructura de control compleja, pero sirve a modo de ejemplo para el alumno de las potencialidades de SASS.**

SCSS Sass

```scss
@function pow($base, $exponent) {
  $result: 1;
  @for $_ from 1 through $exponent {
    $result: $result * $base;
  }
  @return $result;
}

.sidebar {
  float: left;
  margin-left: pow(4, 3) * 1px;
}
```

CSS

```css
.sidebar {
  float: left;
  margin-left: 64px;
}
```

Revisión

- Repasar los conceptos vistos de **sass.**
- Trabajar con el **bonus track** debajo de todo para practicar los **elementos individualmente.**
- Ver todos los videos y materiales necesarios antes de continuar

¡Muchas gracias!

¡Sigamos trabajando!