Frontend I
Clase 23
SASS II

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Temas

1 Introducción

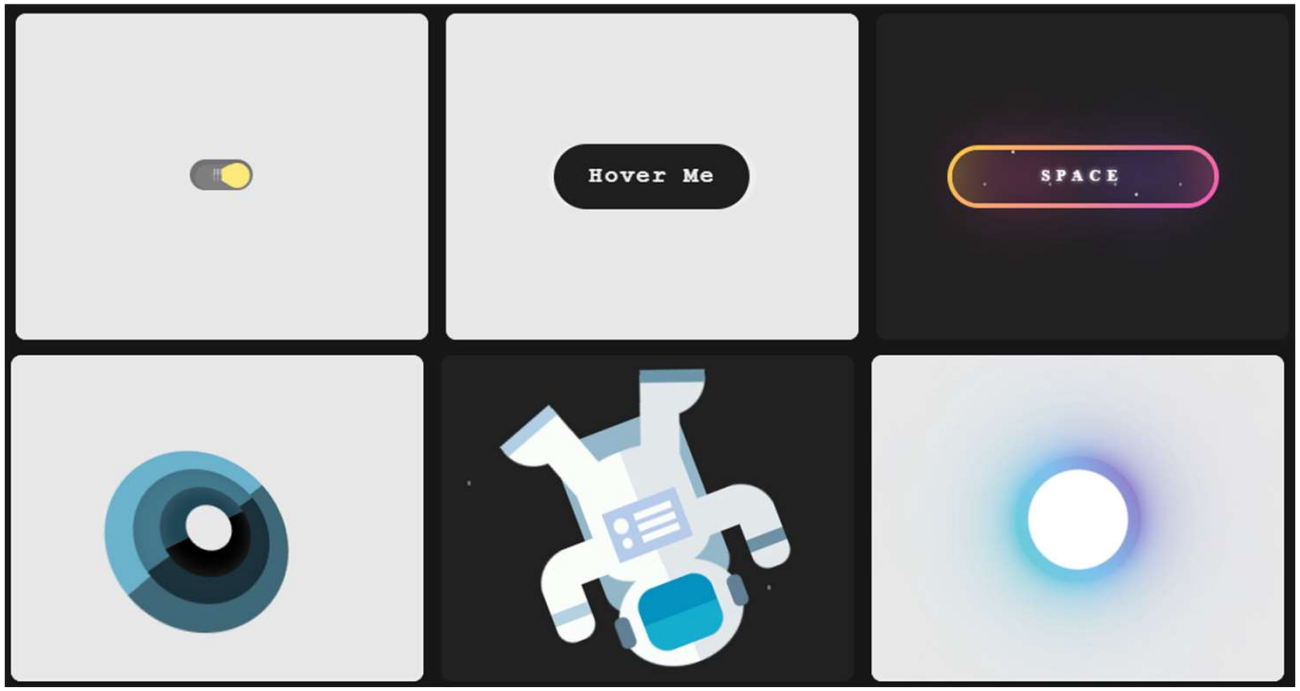
2 Repaso

3 Cierre

4 Actividad

Tip

https://uiverse.io/all



SASS - Partials

- Los "partials" en Sass son archivos que contienen **estilos parciales reutilizables** que pueden ser importados en otros archivos Sass.

- Estos archivos **se crean con un guión bajo (_) al comienzo de su nombre** de archivo para indicar que no deben ser compilados a archivos CSS independientes, sino que están destinados a ser importados en otros archivos Sass.

```scss
// _variables.scss

// Definición de variables
$primary-color: #ff0000;
$secondary-color: #00ff00;
```

```scss
// main.scss

// Importar el partial sin guión bajo
@import 'variables';

// Usar las variables en estilos
body {
  background-color: $primary-color;
  color: $secondary-color;
}
```

SASS - Mixin

- Un mixin es una función que define un **conjunto de estilos** en Sass.
- Puede recibir **argumentos** y generar estilos basados en esos argumentos.
- Los mixins se pueden **reutilizar en múltiples lugares**, lo que permite tener un código más modular y mantener los estilos de manera más eficiente.

```scss
// Definir un mixin para estilos de botón
@mixin button-style($background, $color) {
  background-color: $background;
  color: $color;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}
```

```scss
// Usar el mixin en estilos de botón
.button-primary {
  @include button-style(#ff0000, #ffffff);
}

.button-secondary {
  @include button-style(#00ff00, #000000);
}
```

SASS - Extend

- El "extend" en Sass permite crear una **relación de herencia entre estilos**, donde una clase puede heredar los estilos de otra clase. Esto evita la duplicación de código y permite reutilizar estilos de manera eficiente.

```scss
// Definir un estilo base
.button-base {
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}
```

```scss
// Utilizar extend para heredar estilos
.button-primary {
  @extend .button-base;
  background-color: #ff0000;
  color: #ffffff;
}

.button-secondary {
  @extend .button-base;
  background-color: #00ff00;
  color: #000000;
}
```

SASS - Diferencias entre mixin y extend

En Sass, tanto los mixins como el "extend" son mecanismos para reutilizar estilos, pero tienen algunas diferencias importantes.

Los mixins son funciones que generan estilos y permiten la reutilización de código con argumentos, mientras que el "extend" permite heredar estilos de una clase base sin duplicar código.

Ambos enfoques tienen sus **ventajas** y se pueden utilizar en diferentes situaciones dependiendo de las necesidades específicas del proyecto.

SASS - Diferencias entre mixin y extend

Mixin	Extend
Es una función que define un conjunto de estilos y **puede recibir argumentos**.	Permite heredar estilos de una clase a otra sin duplicar código.
Se puede llamar a un mixin con la directiva **"@include"** y pasarle valores de argumentos específicos.	Se puede utilizar la directiva **"@extend"** para heredar estilos de una clase base a una clase que la extienda.
Los mixins **se compilan en los estilos generados**, y el resultado final incluye los estilos del mixin y de la clase que lo llamó.	Los estilos **extendidos se copian literalmente en la clase** que los extiende, sin modificaciones ni compilaciones adicionales.
Los mixins pueden ser utilizados en **múltiples lugares** y permiten una mayor flexibilidad en la generación de estilos.	Los estilos extendidos **solo se aplican a las clases que las extienden** y no se comparten con otras clases, lo que puede resultar en un código más limpio y eficiente.

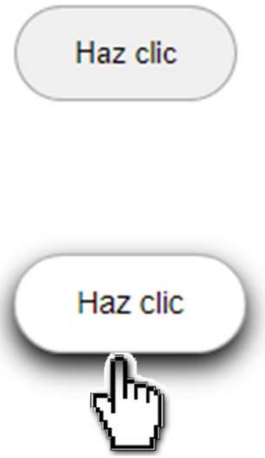
SASS - Extend

SASS - Extend

SASS - Extend

```scss
.button {
  border-radius: 50px;
  border: 1px solid rgba(0, 0, 0, 0.3);
  display: inline-block;
  font-size: 12px;
  padding: 12px 24px;
  transition: all 0.3s ease-in-out;

  &:hover {
    background-color: $primaryColor;
    box-shadow: 0 5px 10px $secondaryColor;
    cursor: pointer;
    transform: scale(1.05);
  }
}
```



SASS - Extend

```scss
.success {
    @extend .button;
    color: $successColor;
    font-size: 18px;
    font-weight: bold;
}

.warning {
    @extend .button;
    color: $warningColor;
    font-size: 22px;
    font-weight: bold;
}

.error {
    @extend .button;
    color: $dangerColor;
    font-size: 26px;
    font-weight: bold;
}
```

SASS - Mixin

```scss
@mixin myButton ($fcolor, $fsize: 32px, $fweight: bold, $cursor: help) {
    border-radius: 50px;
    border: 1px solid rgba(0, 0, 0, 0.3);
    color: $fcolor;
    display: inline-block;
    font-size: $fsize;
    font-weight: $fweight;
    padding: 12px 24px;
    transition: all 0.3s ease-in-out;

    &:hover {
        background-color: $primaryColor;
        box-shadow: 0 5px 10px $secondaryColor;
        cursor: $cursor;
        transform: scale(1.05);
    }
}
```

SASS - Mixin

```
.button {
    @include myButton (darkgrey);
}

.success {
    @include myButton (rgb(221, 160, 221), 18px, bold);
}

.warning {
    @include myButton (darkgoldenrod, 22px, bold);
}

.error {
    @include myButton (darkred, 26px, bold);
}
```

Actividad

https://drive.google.com/file/d/1s18lrc-sOkTqLNwiOhxTn2lzgbneQeml/view?usp=sharing

https://drive.google.com/file/d/17A1ZeHTTofSdddxemGY5hO35e3-vpjva/view?usp=sharing

Conclusiones

Sass es un preprocesador CSS que permite escribir estilos más eficientes y mantenibles, gracias a su capacidad para definir variables, anidar estilos y reutilizar código. Así, puede ayudarnos a mejorar la calidad y la eficiencia del código CSS en proyectos web.

DigitalHouse>
Coding School