Práctica Universitaria: Introducción y Aplicación de SASS en Desarrollo Web

Duración estimada: 2-3 horas

Nivel: Intermedio (requiere conocimientos básicos de CSS y HTML)

Objetivo General:

Aprender a utilizar las principales características de SASS en proyectos web reales, mejorando la eficiencia y organización del código CSS.

1. Variables

 Explicación: Las variables permiten guardar valores reutilizables como colores, fuentes o medidas.

 Ejercicio:

```
// _variables.scss
$primary-color: #3498db;
$secondary-color: #2ecc71;
$font-stack: 'Arial, sans-serif';
$text-size: 16px;

// main.scss
@import 'variables';

.profile-card {
  background-color: $primary-color;
  color: white;
  font-family: $font-stack;
  font-size: $text-size;
  padding: 1rem;
  border-radius: 8px;
}
```

 Tarea del estudiante: Modificar los valores de las variables y observar cómo cambia el estilo en toda la tarjeta.

2. Anidación (Nesting)

 Explicación: SASS permite anidar reglas CSS dentro de otras, imitando la estructura HTML.

 Ejercicio:

```
.navbar {
  background-color: $secondary-color;

  ul {
    list-style: none;
    padding: 0;

    li {
      display: inline-block;
      margin-right: 10px;

      a {
        color: white;
        text-decoration: none;

        &:hover {
          text-decoration: underline;
        }
      }
    }
  }
}
```

 Tarea del estudiante: Agregar una clase `.active` con estilos diferentes al enlace seleccionado.

3. Mixins

 Explicación: Un mixin permite definir estilos reutilizables que pueden aceptar argumentos.

 Ejercicio:

```
@mixin button-style($bg-color, $padding) {
  background-color: $bg-color;
  padding: $padding;
  border: none;
  color: white;
  border-radius: 4px;
  cursor: pointer;
}

.btn-primary {
  @include button-style(#e74c3c, 10px 20px);
}
```

```
.btn-secondary {
  @include button-style(#8e44ad, 5px 15px);
}
```

 Tarea del estudiante: Agregar un nuevo botón `.btn-success` usando el mixin.

4. Funciones

 Explicación: Las funciones en SASS permiten devolver valores y realizar operaciones como cálculos.

 Ejercicio:

```
@function px-to-rem($px, $base: 16) {
  @return #{$px / $base}rem;
}

.text {
  font-size: px-to-rem(24);
}
```

 Tarea del estudiante: Usar la función para establecer el tamaño de diferentes elementos (`h1`, `p`, `.small-text`).

5. Herencia (@extend)

 Explicación: `@extend` permite heredar estilos de otra clase.

 Ejercicio:

```
.box {
  border: 1px solid #ccc;
  padding: 1rem;
  border-radius: 5px;
}

.alert {
  @extend .box;
  background-color: #f2dede;
}

.card {
  @extend .box;
  background-color: #ecf0f1;
}
```

 Tarea del estudiante: Agregar una clase `.info-box` que herede de `.box` pero con diferente color de fondo.

6. Partials y @import / @use

 Explicación: SASS permite dividir el código en archivos parciales (`_archivo.scss`) y luego importarlos o usarlos en un archivo principal.

 Ejercicio:

```
// main.scss
@use 'variables';
@use 'mixins';
@use 'buttons';
```

 Tarea del estudiante: Separar cada sección del código que ha creado en su correspondiente archivo parcial.

7. Condicionales y Bucles

 Explicación: SASS permite usar `@if`, `@for`, `@each` para generar código dinámicamente.

 Ejercicio:

```
@for $i from 1 through 12 {
 .col-#{$i} {
   width: (100% / 12) * $i;
 }
}
```

 Tarea del estudiante: Agregar media queries para que las columnas se ajusten en pantallas pequeñas.

Entregables del Estudiante

1. Proyecto con estructura SASS organizada en parciales.

2. Código fuente (`.scss`) y compilado (`.css`).

3. Capturas o demo funcional mostrando la implementación de cada tema.

Recomendaciones para el Profesor

- Usar una herramienta como Live Sass Compiler (VSCode) o CodePen con soporte SASS.

- Evaluar claridad del código, organización de archivos, uso adecuado de SASS.

- Dar retroalimentación sobre cómo podrían optimizar aún más el CSS generado.