# **Tema 4:** Bootstrap

Semana 13: Personalizar Bootstrap con Sass

# **BLOQUE 3:** Bootstrap

**Módulo:** Desarrollo de Interfaces Web
**Ciclo:** Desarrollo de Aplicaciones Web
**Curso:** 2023 – 2024
**Profesora:** Rosa Medina

# Índice

# 1. Introducción

En la versión de Bootstrap (v3), la creación de temas se debió en parte gracias a las modificaciones de las variables de Bootstrap con LESS, CSS personalizado y una hoja de estilo de temas separada que está incluida hoy en día en el directorio dist. Con bastante trabajo podíamos rediseñar de forma completa el estilo de Bootstrap 3 sin tocar los archivos centrales, por tanto, desde Bootstrap 4 se ha proporcionado un enfoque familiar y ligeramente diferente.

El tema que nos ofrecen se logra mediante variables en Sass, mapas y CSS personalizados. Ahora, no encontramos hojas de estilo personalizadas de forma que podemos habilitar mucho más fácil los temas incorporados, añadir sombras, degradados, y mucho más.

# 2. Sass

Desde Boostrap 4 por tanto, se nos permite utilizar los archivos de Sass para aventajarnos y modificar sus variables, mapas, mixin, etc. (utilizad la última versión de Bootstrap). Estos archivos se encuentran en el directorio bootstrap-x.x.x/scss

## 2.1. Estructura

Siempre que sea posible, debemos evitar modificar los archivos troncales de Bootstrap, eso significa que debemos crear nuestras propias hojas de estilos, de modo que importemos Bootstrap, modifiquemos y extendamos los estilos ahí. Asumiendo que usamos la librería de Bootstrap que hemos descargado, la estructura que este tiene es:

```
nuestro-proyecto/
├── scss
│   └── custom.scss
└── node_modules/
    └── bootstrap
        ├── js
        └── scss
```

En nuestro custom.scss, debemos importar los archivos Sass de Bootstrap. Para ello tenemos dos opciones: incluir todo Bootstrap o coger sólo las partes que necesitamos teniendo en cuenta las dependencias y requerimientos que pueden tener. Además, de incluir el JS correspondiente para el plugins que necesitemos.

Ejemplo A:

```
// Custom.scss
// Option A: Incluimos todo Bootstrap

// Incluimos cualquier variable por defecto a sobreescribir aquí

@import "../node_modules/bootstrap/scss/bootstrap";

// Añade el resto del código customizado aquí
```

Ejemplo B:

```
// Custom.scss
// Option B: Incluye partes de Bootstrap aquí

// 1. Incluye primero las funciones (así puedes manipular colores, SVG, calc, etc.)
@import "../node_modules/bootstrap/scss/functions";

// 2. Incluye cualquier variable por defecto que quieras sobreescribir aquí

// 3. Incluye el resto de hojas de estilo de Bootstrap
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/variables-dark";

// 4. Incluye cualquier mapa para sobreescribirlo aquí

// 5. Incluye el resto de estilos requeridos
@import "../node_modules/bootstrap/scss/maps";
@import "../node_modules/bootstrap/scss/mixins";
@import "../node_modules/bootstrap/scss/root";

// 6. Opcionalmente incluye el resto que necesites
@import "../node_modules/bootstrap/scss/utilities";
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
@import "../node_modules/bootstrap/scss/images";
@import "../node_modules/bootstrap/scss/containers";
@import "../node_modules/bootstrap/scss/grid";
@import "../node_modules/bootstrap/scss/helpers";

// 7. Opcionalmente incluye las últimas APIS para generar clases basadas en el mapa de Sass en `_utilities.scss`
@import "../node_modules/bootstrap/scss/utilities/api";

// 8. Añade el código opcional aquí
```

Con esta configuración, podemos modificar cualquier variable con Sass y mapear nuestro custom.scss. Podemos añadir las partes de Bootstrap que necesitemos debajo de la sección dada, aunque el equipo de Bootstrap nos recomienda que añadamos Bootstrap.scss completo como punto de partida.

## 2.2. Variables por defecto

Todas y cada una de las variables de Bootstrap incluyen la posibilidad de utilizar ! default para poder sobrescribir cualquier valor en tu código de Sass sin necesidad de tener que modificar el código original de Bootstrap. Podemos por tanto copiar y pegar todas las variables que necesitemos y modificar así posteriormente sus valores quitando el valor de !default. Si una variable ha sido ya asignada, a esta variable no se va a reasignar por los valores que tenga por defecto en Bootstrap.

La lista completa de las variables de Bootstrap la tenemos en: scss/_variables.scss. Algunas variables están establecidas a null, estas variables no generarán la propiedad a menos que lo establezcamos nosotros en nuestra configuración.

Veamos un ejemplo de cómo modificar el color de fondo del body importando y compilando Bootstrap vía npm:

```
// Required
@import "../node_modules/bootstrap/scss/functions";

// Default variable overrides
$body-bg: #000;
$body-color: #111;

// Required
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/variables-dark";
@import "../node_modules/bootstrap/scss/maps";
@import "../node_modules/bootstrap/scss/mixins";
@import "../node_modules/bootstrap/scss/root";

// Optional Bootstrap components here
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
// etc
```

Esto lo debemos repetir tantas veces como nos sea necesario para sobrescribir aquellas variables de Bootstrap que queramos modificar.

## 2.3. Maps y loops

Bootstrap incluye un gran número de maps en Sass (para colores, grid breakpoints, etc.) y pares de clave-valor para facilitar la generación de nuestros CSS. Al igual que las variables de Sass, todos los maps incluyen el valor !default que se puede sobreescribir y extender.

Por ejemplo, para modificar un color del mapa $theme-colors, lo que haremos será modificar el color que queramos:

```
$primary: #0074d9;
$danger: #ff4136;
```

Y después estas variables que hemos modificado se cambiarán en el map $theme-colors de Bootstrap:

```
$theme-colors: (
  "primary": $primary,
  "danger": $danger
);
```

Para añadir nuevos colores a $theme-colors o a otro map creando un nuevo map de Sass, lo que haremos será crear nuestras variables o mapa directamente y hacer un merge:

```
// Creamos nuestro map
$custom-colors: (
  "custom-color": #900
);

// Merge el mapa de tema de colores anteriores con nuestro color añadido
$theme-colors: map-merge($theme-colors, $custom-colors);
```

Si por el contrario lo que queremos hacer es eliminar un color, deberemos hacer un map-remove. Para ello, debemos asegurarnos que está tras los scss requerido.

```
// Required
@import "../node_modules/bootstrap/scss/functions";
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/variables-dark";

$theme-colors: map-remove($theme-colors, "info", "light", "dark");

@import "../node_modules/bootstrap/scss/maps";
@import "../node_modules/bootstrap/scss/mixins";
@import "../node_modules/bootstrap/scss/root";

// Optional
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
// etc
```

# 3. Funciones

Bootstrap utiliza numerosas funciones de Sass, por ejemplo para los colores podemos oscurecer/aclarar un color utilizando las funciones tint-color() y shade-color(), estas funciones mezclan los colores con blanco o negro como hace Sass con las funciones lighten() y darken():

```
// Tint a color: mix a color with white
@function tint-color($color, $weight) {
  @return mix(white, $color, $weight);
}

// Shade a color: mix a color with black
@function shade-color($color, $weight) {
  @return mix(black, $color, $weight);
}
```

```
// Shade the color if the weight is positive, else tint it
@function shift-color($color, $weight) {
  @return if($weight > 0, shade-color($color, $weight), tint-color($color, -$weight));
}
```

En la práctica, podemos llamar a la función y pasarle el color y % como parámetros:

```
.custom-element {
  color: tint-color($primary, 10%);
}

.custom-element-2 {
  color: shade-color($danger, 30%);
}
```

Más info aquí.

# 4. Color contrast

Una funcionalidad extra incluida en Bootstrap es la función color-contrast que utiliza WCAG contrast ratio algorithm para calcular el contraste en base a la luminosidad relativa en sRGB. Esta función es muy útil para mixins o loops donde estamos generando múltiples clases.

Por ejemplo, para generar muestras del mapa $theme-colors:

```
@each $color, $value in $theme-colors {
  .swatch-#{$color} {
    color: color-contrast($value);
  }
}

.custom-element {
  color: color-contrast(#000); // returns `color: #fff`
}

.custom-element {
  color: color-contrast($dark); // returns `color: #fff`
}
```

# 5. Opciones con Sass

Personalizar Bootstrap con Sass es bastante fácil con el archivo de variables personalizadas incorporado, de forma que sólo tendríamos que cambiar las preferencias globales de CSS con las nuevas variables $enable-* con esto sobreescribimos el valor de una variable, y recompilaríamos (npm run test).

Podemos encontrar y personalizar las variables para las opciones globales en el archivo: scss/_variables.scss.

| Variable | Valor | Descripción |
|---|---|---|
| $spacer | 1rem (por defecto), o cualquier valor >0 | Especifica el espacio por defecto que generará nuestro spacer utilities. |
| $enable-dark-mode | true (por defecto) o false | Habilita la compatibilidad con el modo oscuro integrado en el proyecto y componentes |
| $enable-rounded | true (por defecto) o false | Por defecto en varios componentes está habilitado el border-radius |
| $enable-shadows | true o false (por defecto) | Por defecto en varios componentes está habilitado el box-shadow |
| $enable-gradients | true o false (por defecto) | Habilita los gradientes vía background-image en varios compoentes |
| $enable-transitions | true (por defecto) o false | Habilita las transiciones en varios componentes |
| $enable-reduced-motion | true (por defecto) o false | Habilita las preferencias de los media queries sobre animación/transición. |
| $enable-grid-classes | true (por defecto) o false | Habilita la generación de las clases CSS para el grid (Por ejemplo: .container, .row, .col-md-1,...) |
| $enable-container-classes | true (por defecto) o false | Habilita la generación de las clases CSS para los contenedores |
| $enable-caret | true (por defecto) o false | Habilita el pseudo elemento caret en la clase .dropdown-toggle |
| $enable-button-pointers | true (por defecto) o false | Añade el cursor pointer a elementos de botón que no está deshabilitado |
| $enable-rfs | true (por defecto) o false | Habilita RFS. |
| $enable-validation-icons | true (por defecto) o false | Habilita background-image iconos con inputs y algunos formularios con validaciones |
| $enable-negative-margins | true (por defecto) o false | Habilita la generación de las herramientas de márgenes negativos |

| | | |
|---|---|---|
| **$enable-deprecation-messages** | true (por defecto) o false | Esablece a falso para ocultar los warnings cuando usamos algún mixin deprecated o función que está planeada para ser eliminada en la versión 6. |
| **$enable-important-utilities** | true (por defecto) o false | Habilita !important |
| **$enable-smooth-scroll** | true (por defecto) o false | Aplica scroll-behavior: smooth |

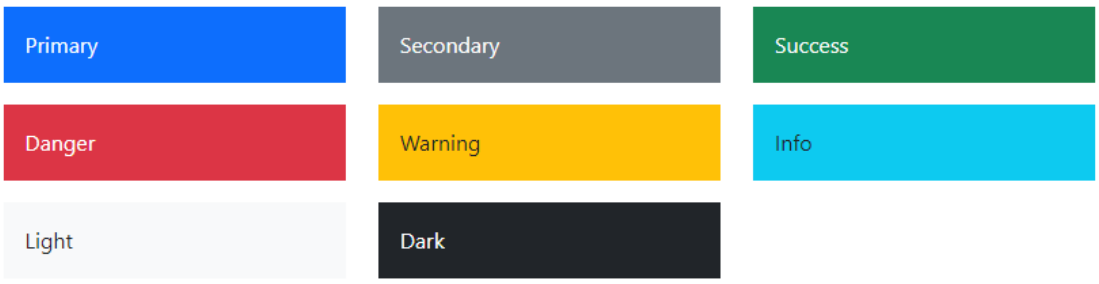
```
<link rel="stylesheet" href="shetland.css" type="text/css" media="screen and (max-device-width: 480px)">
```

# 6. Colores

Muchos de los componentes construidos en Bootstrapt tienen una serie de colores definidas en el map de Sass($theme-colors).

```
$theme-colors: (
  "primary":    $primary,
  "secondary":  $secondary,
  "success":    $success,
  "info":       $info,
  "warning":    $warning,
  "danger":     $danger,
  "light":      $light,
  "dark":       $dark
);
```

La definición de los colores se encuentra en el archivo scss/_variables.scss.




Estos son los colores básicos que tiene, los cuales los modifican mediante el valor (blanco/negro/peso), creando una paleta de colores.

## 6.1. Color sass maps

Dentro del archivo scss/_variables.scss, encontramos 3 maps de colores que nos facilitan y sus valores hexadecimales.

- $colors: Lista todos los colores
- $theme-colors: Lista los colores semánticamente nombrados
- $grays: Lista todos los colores de tonos de grises

Por ejemplo, la variable $colors: contiene:

```
$colors: (
  "blue":      $blue,
  "indigo":    $indigo,
  "purple":    $purple,
  "pink":      $pink,
  "red":       $red,
  "orange":    $orange,
  "yellow":    $yellow,
  "green":     $green,
  "teal":      $teal,
  "cyan":      $cyan,
  "white":     $white,
  "gray":      $gray-600,
  "gray-dark": $gray-800
);
```

Añadir, modificar o eliminar valores de este mapa de colores no es el más acertado, ya que muchos componentes no lo utilizan (por ahora). Hasta entonces tendremos que usar la variable ${color} y ese map de Sass.

## 6.2. Generating utilities

Bootstrap no tiene color ni background-color para cada variable color, pero podemos generárnosla mediante utility API. Para ello, los pasos a seguir son:

1. Nos debemos de asegurar que hemos importado las funciones, variables, mixins y utilities
2. Usamos la fusión para hacer un merge map-merge-multiple() creando un nuevo maps
3. Hacemos un merge para combiar cualquier utility con un nombre de clase {color}-{level}.

Por ejemplo, si queremos generar .text-purple-500:

```
@import "bootstrap/scss/functions";
@import "bootstrap/scss/variables";
@import "bootstrap/scss/variables-dark";
@import "bootstrap/scss/maps";
```

```
@import "bootstrap/scss/mixins";
@import "bootstrap/scss/utilities";

$all-colors: map-merge-multiple($blues, $indigos, $purples, $pinks, $reds, $oranges, $yellows,
$greens, $teals, $cyans);

$utilities: map-merge(
  $utilities,
  (
    "color": map-merge(
      map-get($utilities, "color"),
      (
        values: map-merge(
          map-get(map-get($utilities, "color"), "values"),
          (
            $all-colors
          ),
        ),
      ),
    ),
  )
);

@import "bootstrap/scss/utilities/api";
```

Esto nos genera un nuevo .text-{color}-{level} utilities para cada color y nivel.

# 7. Componentes

## 7.1. Modificadores

Muchos componentes de Bootstrap han sido construidos con el objetivo de poder ser modificados posteriormente mediante unas clases modificadoras. Esto significa que la mayor parte del estilo está contenido en una clase base (por ejemplo .btn) mientras que las variables de estilo están confinadas a las clases modificadoras (por ejemplo: .btn-danger). Estas clases se crean a partir del mapa $theme-color para personalizar el número y el nombre de nuestras clases modificadoras.

Por ejemplo, para generar a partir de los modificadores de los componentes .alert y .list-group.

```
// Generate contextual modifier classes for colorizing the alert.

@each $state, $value in $theme-colors {
  $alert-background: shift-color($value, $alert-bg-scale);
  $alert-border: shift-color($value, $alert-border-scale);
  $alert-color: shift-color($value, $alert-color-scale);
  @if (contrast-ratio($alert-background, $alert-color) < $min-contrast-ratio) {
```

```
    $alert-color: mix($value, color-contrast($alert-background), abs($alert-color-scale));
  }
  .alert-#{$state} {
    @include alert-variant($alert-background, $alert-border, $alert-color);
  }
}

// List group contextual variants
//
// Add modifier classes to change text and background color on individual items.
// Organizationally, this must come after the `:hover` states.

@each $state, $value in $theme-colors {
  $list-group-variant-bg: shift-color($value, $list-group-item-bg-scale);
  $list-group-variant-color: shift-color($value, $list-group-item-color-scale);
  @if (contrast-ratio($list-group-variant-bg, $list-group-variant-color) < $min-contrast-ratio) {
    $list-group-variant-color: mix($value, color-contrast($list-group-variant-bg), abs($list-group-item-color-scale));
  }

  @include list-group-item-variant($state, $list-group-variant-bg, $list-group-variant-color);
}
```

## 7.2. Responsive

Podemos modificar los responsive por ejemplo de nuestros componentes o utilities. Por ejemplo, si queremos modificar la alineación del texto donde tenemos un mix @each para cada $grid-breakpoints en Sass podríamos hacer:

```
// We deliberately hardcode the `bs-` prefix because we check
// this custom property in JS to determine Popper's positioning

@each $breakpoint in map-keys($grid-breakpoints) {
  @include media-breakpoint-up($breakpoint) {
    $infix: breakpoint-infix($breakpoint, $grid-breakpoints);

    .dropdown-menu#{$infix}-start {
      --bs-position: start;

      &[data-bs-popper] {
        right: auto;
        left: 0;
      }
    }

    .dropdown-menu#{$infix}-end {
      --bs-position: end;

      &[data-bs-popper] {
```

```
      right: 0;
      left: auto;
    }
   }
  }
}
```

Debemos tener en cuenta que necesitamos modificar nuestro $grid-breakpoints, esto se cambiará en todos los ítems de dicho mapa.

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);
```