PERSONALIZAR BOOTSTRAP

ANA ROSA HOYOS TERAN

[NOMBRE DE LA EMPRESA] [Dirección de la compañía]

Contenido

Personalizar Bootstrap

Aprende a crear temas, personalizar y ampliar Bootstrap con Sass, un montón de opciones globales, un amplio sistema de colores y más.

- **Sass**: Utiliza nuestros archivos fuente Sass para aprovechar variables, mapas, mixins y funciones.
- **Options**: Personaliza Bootstrap con variables integradas para alternar fácilmente las preferencias globales de CSS.
- **Color**: Conoce y personaliza los sistemas de colores utilizados en todo el toolkit.
- **Components**: Aprende cómo construimos casi todos nuestros componentes de manera responsive y con clases y modificadoras base.
- **CSS variables**: Utiliza las propiedades personalizadas de CSS de Bootstrap para un diseño y desarrollo rápidos y con visión de futuro.
- **Optimize**: Manten tus proyectos ágiles, responsive y fáciles de mantener para que puedas ofrecer la mejor experiencia.

Descripción general

Hay varias formas de personalizar Bootstrap. Tu mejor camino puede depender de tu proyecto, la complejidad de tus herramientas de compilación, la versión de Bootstrap que estás usando, la compatibilidad con el navegador y más.

Nuestros dos métodos preferidos son:

1. Usar Bootstrap a través del administrador de paquetes para que puedas usar y ampliar nuestros archivos fuente.
2. Usar los archivos de distribución compilados de Bootstrap o jsDelivr para que puedas agregar o sobrescribir los estilos de Bootstrap.

Si bien no podemos entrar en detalles aquí sobre cómo usar cada administrador de paquetes, podemos brindarte orientación sobre el uso de Bootstrap con su propio compilador Sass.

Para aquellos que quieran usar los archivos de distribución, revisen la página de inicio para saber cómo incluir esos archivos y una página HTML de ejemplo. A partir de ahí, consulta los documentos para conocer el diseño, los componentes y los comportamientos que te gustaría usar.

A medida que te familiarices con Bootstrap, continúa explorando esta sección para obtener más detalles sobre cómo utilizar nuestras opciones globales, cómo cambiar nuestro sistema de color, cómo construimos nuestros componentes, cómo usar nuestra creciente lista de propiedades personalizadas de CSS y cómo optimizar tu código al construir con Bootstrap.

CSP y SVG integrados

Varios componentes de Bootstrap incluyen SVG incrustados en nuestro CSS para diseñar componentes de manera consistente y sencilla en todos los navegadores y dispositivos. **Para organizaciones con configuraciones de CSP más estrictas**, hemos documentado todas las instancias de nuestros SVG integrados (todos los cuales se aplican a través de `background-image`) para que puedas revisar más a fondo sus opciones.

- Accordion
- Botón de cierre (usado en alerts y modals)
- Checkboxes y botones radio de formulario
- Switches de formulario
- Iconos de validación de formulario
- Menús de selección
- Controles de Carousel
- Botones de Navbar

Según la conversación de la comunidad, algunas opciones para abordar esto en tu propio código base incluyen reemplazar las URL con assets alojados localmente, eliminar las imágenes y usar imágenes en línea (no es posible en todos los componentes), y modificando tu CSP. Nuestra recomendación es revisar cuidadosamente tus propias políticas de seguridad y decidir cuál es el mejor camino a seguir, si es necesario.

Personalizar Sass de Bootstrap

Utiliza nuestros archivos fuente Sass para aprovechar variables, mapas, mixins y funciones para ayudarte a construir más rápido y personalizar tu proyecto.

Utiliza nuestros archivos fuente Sass para aprovechar variables, mapas, mixins y más.

Estructura de archivos

Siempre que sea posible, evita modificar los archivos del núcleo de Bootstrap. Para Sass, eso significa crear tu propia hoja de estilos que importe Bootstrap para que puedas modificarla y ampliarla. Suponiendo que estás utilizando un administrador de paquetes como npm, tendrás una estructura de archivos que se ve así:

```
your-project/
├── scss
│   └── custom.scss
└── node_modules/
    └── bootstrap
        ├── js
        └── scss
```

Si descargaste nuestros archivos fuente y no estás utilizando un administrador de paquetes, querrás configurar manualmente algo similar a esa estructura, manteniendo los archivos fuente de Bootstrap separados de los tuyos.

```
your-project/
├── scss
│   └── custom.scss
└── bootstrap/
    ├── js
    └── scss
```

Importación

En tu `custom.scss`, importarás los archivos fuente Sass de Bootstrap. Tienes dos opciones: incluir todo Bootstrap o elegir las partes que necesitas. Recomendamos lo último, aunque ten en cuenta que existen algunos requisitos y dependencias entre nuestros componentes. También deberás incluir algo de JavaScript para nuestros complementos.

```scss
// Custom.scss
// Opción A: Incluir todo Bootstrap

// Incluye cualquier sobrescritura de variable predeterminada aquí (aunque las funciones no estarán disponibles)

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

// Luego agrega código personalizado adicional aquí
// Custom.scss
// Opción B: Incluir partes de Bootstrap

// 1. Incluye las funciones primero (para que puedas manipular colores, SVG, calc, etc.)
@import "../node_modules/bootstrap/scss/functions";
```

```scss
// 2. Incluye cualquier sobrescritura de variable predeterminada aquí

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

// 4. Incluye cualquier sobrescritura de mapa predeterminado aquí

// 5. Incluye el resto de las partes requeridas
@import "../node_modules/bootstrap/scss/maps";
@import "../node_modules/bootstrap/scss/mixins";
@import "../node_modules/bootstrap/scss/root";

// 6. Opcionalmente, incluye otras partes según sea necesario
@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 la API de utilidades en último lugar para
generar clases basadas en el mapa Sass en `_utilities.scss`
@import "../node_modules/bootstrap/scss/utilities/api";

// 8. Agrega código personalizado adicional aquí
```

Con esta configuración, puedes comenzar a modificar cualquiera de las variables y mapas de Sass en tu `custom.scss`. También puedes comenzar a agregar partes de Bootstrap en las secciones `// Opcionalmente` según sea necesario. Sugerimos usar la pila de importación completa de nuestro archivo `bootstrap.scss` como punto de partida.

Valores predeterminados de variables

Cada variable de Sass en Bootstrap incluye el indicador `!default` que te permite sobrescribir el valor predeterminado de la variable en tu propio Sass sin modificar el código fuente de Bootstrap. Copia y pega las variables según sea necesario, modifica tus valores y elimina el indicador `!default`. Si ya se asignó una variable, no se reasignará con los valores predeterminados en Bootstrap.

Encontrarás la lista completa de variables de Bootstrap en `scss/_variables.scss`. Algunas variables se establecen en `null`, estas variables no generan la propiedad a menos que se sobrescriban en tu configuración.

Las sobrescrituras de variables deben realizarse después de importar nuestras funciones, pero antes del resto de las importaciones.

Aquí hay un ejemplo que cambia `background-color` y `color` para `<body>` al importar y compilar Bootstrap a través de npm:

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

// Sobrescritura del valor por defecto de las variables
$body-bg: #000;
$body-color: #111;
```

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

// Componentes opcionales de Bootstrap aquí
@import "../node_modules/bootstrap/scss/reboot";
@import "../node_modules/bootstrap/scss/type";
// etc
```

Repite según sea necesario para cualquier variable en Bootstrap, incluidas las opciones globales a continuación.

¡Comienza con Bootstrap a través de npm con nuestro proyecto de inicio! Diríjete al repositorio de plantillas twbs/bootstrap-npm-starter para ver cómo construir y personalizar Bootstrap en tu propio proyecto npm. Incluye el compilador Sass, Autoprefixer, Stylelint, PurgeCSS y Bootstrap Icons.

Mapas y bucles

Bootstrap incluye un puñado de mapas Sass, pares de clave valor que facilitan la generación de familias de CSS relacionadas. Usamos mapas Sass para nuestros colores, breakpoints de cuadrícula (grid) y más. Al igual que las variables de Sass, todos los mapas de Sass incluyen el indicador `!default` y se pueden sobrescribir y ampliar.

Algunos de nuestros mapas Sass se fusionan en mapas vacíos de forma predeterminada. Esto se hace para permitir una fácil expansión de un mapa Sass determinado, pero tiene el costo de hacer *removing* de elementos de un mapa sea un poco más difícil.

Modificar mapa

Todas las variables en el mapa `$theme-colors` se definen como variables independientes. Para modificar un color existente en nuestro mapa `$theme-colors`, agrega lo siguiente a tu archivo Sass personalizado:

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

Posteriormente, estas variables se configuran en el mapa `$theme-colors` de Bootstrap:

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

Añadir al mapa

Agrega nuevos colores a `$theme-colors`, o cualquier otro mapa, creando un nuevo mapa Sass con sus valores personalizados y combinándolo con el mapa original. En este caso, crearemos un nuevo mapa `$custom-colors` y lo fusionaremos con `$theme-colors`.

```scss
// Crea tu propio mapa
$custom-colors: (
  "custom-color": #900
);
```

```scss
// Fusionar los mapas
$theme-colors: map-merge($theme-colors, $custom-colors);
```

Quitar del mapa

Para eliminar colores de `$theme-colors`, o cualquier otro mapa, usa `map-remove`. Ten en cuenta que debes insertarlo entre nuestros requisitos y opciones:

```scss
// Requerido
@import "../node_modules/bootstrap/scss/functions";
@import "../node_modules/bootstrap/scss/variables";
@import "../node_modules/bootstrap/scss/maps";
@import "../node_modules/bootstrap/scss/mixins";
@import "../node_modules/bootstrap/scss/root";

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

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

Claves requeridas

Bootstrap asume la presencia de algunas claves específicas dentro de los mapas de Sass tal como las usamos y las ampliamos nosotros mismos. A medida que personalizas los mapas incluidos, es posible que encuentres errores cuando se utiliza la clave de un mapa Sass específico.

Por ejemplo, usamos las teclas `primary`, `success` y `danger` de `$theme-colors` para enlaces, botones y estados de formulario. Reemplazar los valores de estas claves no debería presentar problemas, pero eliminarlas puede causar problemas de compilación de Sass. En estos casos, deberás modificar el código Sass que utiliza esos valores.

Funciones

Colores

Además de los mapas Sass que tenemos, los colores del tema también se pueden usar como variables independientes, como `$primary`.

```scss
.custom-element {
  color: $gray-100;
  background-color: $dark;
}
```

Puedes aclarar u oscurecer los colores con las funciones `tint-color()` y `shade-color()` de Bootstrap. Estas funciones mezclarán colores con negro o blanco, a diferencia de las funciones `lighten()` y `darken()` nativas de Sass, que cambiarán la luminosidad en una cantidad fija, lo que a menudo no produce el efecto deseado.

```scss
// 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, llamarías a la función y pasarías los parámetros de color y peso.

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

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

Contraste de color

Para cumplir con los estándares de accesibilidad WCAG 2.0 para contraste de color, los autores **deben** proporcionar una relación de contraste de al menos 4,5:1, con muy pocas excepciones.

Una función adicional que incluimos en Bootstrap es la función de contraste de color, `color-contrast`. Utiliza el algoritmo WCAG 2.0 para calcular umbrales de contraste basados en luminancia relativa en un espacio de color `sRGB` para devolver automáticamente un color de contraste claro (`#fff`), oscuro (`#212529`) o negro (`#000`) basado en el color base especificado. Esta función es especialmente útil para mixins o bucles en los que generas varias clases.

Por ejemplo, para generar muestras de color a partir de nuestro mapa `$theme-colors`:

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

También se puede utilizar para necesidades puntuales de contraste:

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

También puedes especificar un color base con nuestras funciones de mapa de colores:

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

Escapar SVG

Usamos la función `escape-svg` para escapar de los caracteres <, > y # para las imágenes de fondo SVG. Al usar la función `escape-svg`, se deben citar los URI de datos.

Sumar y restar funciones

Usamos las funciones `add` y `subtract` para envolver la función `calc` de CSS. El objetivo principal de estas funciones es evitar errores cuando se pasa un valor `0` "sin unidades" a una expresión `calc`. Expresiones como `calc(10px - 0)` devolverán un error en todos los navegadores, a pesar de ser matemáticamente correctas.

Ejemplo donde el cálculo es válido:

```scss
$border-radius: .25rem;
$border-width: 1px;

.element {
  // El cálculo de salida (.25rem - 1px) es válido
  border-radius: calc($border-radius - $border-width);
}

.element {
  // Salida de la misma calc(.25rem - 1px) como arriba
  border-radius: subtract($border-radius, $border-width);
}
```

Ejemplo donde el cálculo no es válido:

```scss
$border-radius: .25rem;
$border-width: 0;

.element {
  // El cálculo de salida (.25rem - 0) no es válido
  border-radius: calc($border-radius - $border-width);
}

.element {
  // Salida .25rem
  border-radius: subtract($border-radius, $border-width);
}
```

Mixins

Nuestro directorio `scss/mixins/` tiene una tonelada de mixins que potencian partes de Bootstrap y también se pueden usar en tu propio proyecto.

Esquemas de color

Un mixin de atajo para la media query `prefers-color-scheme` está disponible con soporte para esquemas de color `light`, `dark` y personalizado.

```scss
@mixin color-scheme($name) {
  @media (prefers-color-scheme: #{$name}) {
    @content;
  }
}
.custom-element {
  @include color-scheme(dark) {
    // Inserta estilos de modo oscuro aquí
  }

  @include color-scheme(custom-named-scheme) {
    // Inserta estilos de esquemas de colores personalizados aquí
  }}
```

Opciones para personalizar Bootstrap

Personaliza rápidamente Bootstrap con variables integradas para alternar fácilmente las preferencias globales de CSS para controlar el estilo y el comportamiento.

Personaliza Bootstrap con nuestro archivo de variables personalizadas incorporado y alterna fácilmente las preferencias globales de CSS con las nuevas variables `$enable-*` Sass. Sobrescribe el valor de una variable y vuelve a compilar con `npm run test` según sea necesario.

Puedes encontrar y personalizar estas variables para opciones globales clave en el archivo `scss/_variables.scss` de Bootstrap.

Variable	Values	Description
`$spacer`	`1rem` (default), o cualquier valor > 0	Especifica el valor de espaciador predeterminado para generar mediante programación nuestras utilidades de espaciador.
`$enable-rounded`	`true` (default) o `false`	Habilita estilos `border-radius` predefinidos en varios componentes.
`$enable-shadows`	`true` o `false` (default)	Habilita estilos decorativos predefinidos de `box-shadow` en varios componentes. No afecta a los `box-shadow` que se utilizan para los estados de enfoque.
`$enable-gradients`	`true` o `false` (default)	Habilita gradientes predefinidos a través de estilos de `background-image` en varios componentes.
`$enable-transitions`	`true` (default) o `false`	Habilita `transitions` predefinidas en varios componentes.
`$enable-reduced-motion`	`true` (default) o `false`	Habilita las `prefers-reduced-motion` media query, que suprime ciertas animaciones/transiciones basadas en las preferencias del usuario en el navegador/sistema operativo.
`$enable-grid-classes`	`true` (default) o `false`	Habilita la generación de clases CSS para el sistema de cuadrícula (por ejemplo, `.row`, `.col-md-1`, etc.).

Variable	Values	Description
`$enable-container-classes`	`true` (default) o `false`	Habilita la generación de clases CSS para contenedores de diseño. (Nuevo en v5.2.0)
`$enable-caret`	`true` (default) o `false`	Habilita el icono caret del pseudo elemento en `.dropdown-toggle`.
`$enable-button-pointers`	`true` (default) o `false`	Agregue el cursor de “mano” a los elementos de botón no deshabilitados.
`$enable-rfs`	`true` (default) o `false`	Habilita globalmente RFS.
`$enable-validation-icons`	`true` (default) o `false`	Habilita los íconos de `background-image` dentro de los inputs de texto y algunos formularios personalizados para los estados de validación.
`$enable-negative-margins`	`true` o `false` (default)	Habilita la generación de utilidades de margen negativo.
`$enable-deprecation-messages`	`true` (default) o `false`	Establécelo en `false` para ocultar las advertencias al usar cualquiera de los mixins y funciones en desuso que se planea eliminar en v6.
`$enable-important-utilities`	`true` (default) o `false`	Habilita el sufijo `!important` en las clases de utilidad.
`$enable-smooth-scroll`	`true` (default) o `false`	Aplica `scroll-behavior: smooth` globalmente, excepto para los usuarios que solicitan un movimiento reducido a través de la media query `prefers-reduced-motion`

Los colores de Bootstrap

Bootstrap está respaldado por un extenso sistema de colores que tematiza nuestros estilos y componentes. Esto permite una personalización y extensión más completas para cualquier proyecto.

Colores del tema

Usamos un subconjunto de todos los colores para crear una paleta de colores más pequeña para generar esquemas de color, también disponible como variables Sass y un mapa Sass en el archivo `scss/_variables.scss` de Bootstrap.

```
<div class="row">
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-primary text-white">Primary</div>
    </div>
    <div class="col-md-4">
```

```
      <div class="p-3 mb-3 bg-secondary text-white">Secondary</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-success text-white">Success</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-danger text-white">Danger</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-warning text-dark">Warning</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-info text-dark">Info</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-light text-dark">Light</div>
    </div>
    <div class="col-md-4">
      <div class="p-3 mb-3 bg-dark text-white">Dark</div>
    </div>
</div>
```

Todos estos colores están disponibles como mapa Sass, `$theme-colors`.

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

Consulta nuestros documentos de mapas y bucles de Sass para saber cómo modificar estos colores.

Todos los colores

Todos los colores de Bootstrap están disponibles como variables Sass y un mapa Sass en el archivo `scss/_variables.scss`. Para evitar aumentar el tamaño de los archivos, no creamos clases de color de texto o de fondo para cada una de estas variables. En su lugar, elegimos un subconjunto de estos colores para una paleta de temas.

Asegúrate de controlar las relaciones de contraste a medida que personalizas los colores. Como se muestra a continuación, hemos agregado tres proporciones de contraste a cada uno de los colores principales: una para los colores actuales de la muestra, una contra el blanco y otra contra el negro.

$blue #0d6efd

$blue-100

$blue-200

$blue-300

$blue-400

$blue-500

$blue-600

$blue-700

$blue-800

$blue-900

indigo #6610f2

$indigo-100

$indigo-200

$indigo-300

$indigo-400

$indigo-500

$indigo-600

$indigo-700

$indigo-800

$indigo-900

purple #6f42c1

$purple-100

$purple-200

$purple-300

$purple-400

$purple-500

$purple-600

$purple-700

$purple-800

$purple-900

pink #d63384

$pink-100

$pink-200

$pink-300

$pink-400

$pink-500

$pink-600

$pink-700

$pink-800

$pink-900

red #dc3545

$red-100

$red-200

$red-300

$red-400

$red-500

$red-600

$red-700

$red-800

$red-900

orange #fd7e14

$orange-100

$orange-200

$orange-300

$orange-400

$orange-500

$orange-600

$orange-700

$orange-800

$orange-900

yellow #ffc107

$yellow-100

$yellow-200

$yellow-300

$yellow-400

$yellow-500

$yellow-600

$yellow-700

$yellow-800

$yellow-900

green #198754

$green-100

$green-200

$green-300

$green-400

$green-500

$green-600

$green-700

$green-800

$green-900

teal #20c997

$teal-100

$teal-200

$teal-300

$teal-400

$teal-500

$teal-600

$teal-700

$teal-800

$teal-900

cyan #0dcaf0

$cyan-100

$cyan-200

$cyan-300

$cyan-400

$cyan-500

$cyan-600

$cyan-700

$cyan-800

$cyan-900

gray-500 #adb5bd

$gray-100

$gray-200

$gray-300

$gray-400

$gray-500

$gray-600

$gray-700

$gray-800

$gray-900

black #000

white #fff

Notas sobre Sass

Sass no puede generar variables programáticamente, por lo que creamos variables manualmente para cada tinte y sombreado nosotros mismos. Especificamos el valor del punto medio (por ejemplo, `$blue-500`) y usamos funciones de color personalizadas para teñir (aclarar) o sombrear (oscurecer) nuestros colores a través de la función de color `mix()` de Sass.

Usar `mix()` no es lo mismo que `lighten()` y `darken()`: el primero combina el color especificado con blanco o negro, mientras que el último solo ajusta el valor de luminosidad de cada color. El resultado es un conjunto de colores mucho más completo, como se muestra en esta demostración de CodePen.

Nuestras funciones `tint-color()` y `shade-color()` usan `mix()` junto con nuestra variable `$theme-color-interval`, que especifica un valor porcentual escalonado para cada color mezclado que producimos. Consulta los archivos `scss/_functions.scss` y `scss/_variables.scss` para obtener el código fuente completo.

Mapas Sass de colores

Los archivos fuente de Sass de Bootstrap incluyen tres mapas para ayudarte a recorrer rápida y fácilmente una lista de colores y sus valores hexadecimales.

- `$colors` enumera todos nuestros colores base disponibles (`500`)
- `$theme-colors` enumera todos los colores del tema con nombres semánticos (que se muestran a continuación)
- `$grays` enumera todos los tintes y tonos de gris

Dentro de `scss/_variables.scss`, encontrarás las variables de color de Bootstrap y el mapa de Sass. Aquí hay un ejemplo del mapa Sass `$colors`:

```
$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
);
```

Agrega, elimina o modifica valores dentro del mapa para actualizar cómo se usan en muchos otros componentes. Desafortunadamente, en este momento, no todos los componentes utilizan este mapa de Sass. Las actualizaciones futuras se esforzarán por mejorar esto. Hasta entonces, planea hacer uso de las variables `${color}` y este mapa de Sass.

Ejemplo

Así es como puedes usarlos en tu Sass:

```
.alpha { color: $purple; }
.beta {
  color: $yellow-300;
  background-color: $indigo-900;
}
```

Las clases de utilidad Color y background también están disponibles para configurar `color` y `background-color` utilizando los valores de color `500`.

Generación de utilidades

Añadido en v5.1.0

Bootstrap no incluye las utilidades `color` y `background-color` para cada variable de color, pero puedes generarlas tú mismo con nuestra utility API y nuestra mapas Sass extendido añadido en v5.1.0.

1. Para comenzar, asegúrate de haber importado nuestras funciones, variables, mixins y utilidades.
2. Utiliza nuestra función `map-merge-multiple()` para fusionar rápidamente múltiples mapas Sass en un nuevo mapa.
3. Fusiona este nuevo mapa combinado para ampliar cualquier utilidad con un nombre de clase `{color}-{level}`.

Aquí hay un ejemplo que genera utilidades de color de texto (por ejemplo, `.text-purple-500`) siguiendo los pasos anteriores.

```
@import "bootstrap/scss/functions";
@import "bootstrap/scss/variables";
@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 generará nuevas utilidades `.text-{color}-{level}` para cada color y nivel. También puedes hacer lo mismo con cualquier otra utilidad y propiedad.

Sobre los componentes de Bootstrap

Aprende cómo y por qué construimos casi todos nuestros componentes de forma responsive y con clases y modificadoras base.

Clases base

Los componentes de Bootstrap se construyen en gran medida con una nomenclatura de modificador base. Agrupamos tantas propiedades compartidas como sea posible en una clase base, como `.btn`, y luego agrupamos estilos individuales para cada variante en clases de modificadores, como `.btn-primary` o `.btn-success`.

Para construir nuestras clases de modificadores, usamos los bucles `@each` de Sass para iterar sobre un mapa de Sass. Esto es especialmente útil para generar variantes de un componente mediante nuestros `$theme-colors` y crear variantes responsive para cada breakpoint. A medida que personalizas estos mapas de Sass y los vuelves a compilar, verás automáticamente tus cambios reflejados en estos bucles.

Consulta nuestros documentos de mapas y bucles de Sass para saber cómo personalizar estos bucles y extender el enfoque modificador base de Bootstrap a tu propio código.

Modificadores

Muchos de los componentes de Bootstrap están construidos con un enfoque de clase de modificador base. Esto significa que la mayor parte del estilo está contenido en una clase base (p. ej., `.btn`), mientras que las variaciones de estilo se limitan a las clases de modificadores (p. ej., `.btn-danger`). Estas clases de modificadores se crean a partir del mapa `$theme-colors` para personalizar el número y el nombre de nuestras clases de modificadores.

Aquí hay dos ejemplos de cómo recorremos el mapa `$theme-colors` para generar modificadores para 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);
}
```

Responsive

Estos bucles de Sass tampoco se limitan a mapas de colores. También puedes generar variaciones sensibles de tus componentes. Tomemos como ejemplo nuestra alineación responsive de los menús desplegables donde mezclamos un bucle `@each` para el mapa Sass `$grid-breakpoints` con una media query incluida.

```
// 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;
      }
    }
  }
}
```

Si modificas tus `$grid-breakpoints`, tus cambios se aplicarán a todos los bucles que iteren sobre ese mapa.

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

Para obtener más información y ejemplos sobre cómo modificar nuestros mapas y variables Sass, consulta la sección Sass de la documentación de Grid.

Creando el tuyo propio

Te alentamos a que adoptes estas pautas cuando construya con Bootstrap para crear tus propios componentes. Nosotros mismos hemos ampliado este enfoque a los componentes personalizados en nuestra documentación y ejemplos. Los componentes como nuestros *callouts* se construyen al igual que nuestros componentes provistos con clases base y modificadoras.

```
  <div class="bd-callout my-0">
    <strong>Esto es un callout (texto destacado).</strong> Lo creamos
personalizado para nuestros documentos para que nuestros mensajes se
destaquen. Tiene tres variantes a través de clases de modificadoras.
  </div>
<div class="callout">...</div>
```

En tu CSS, tendrías algo como lo siguiente, donde la mayor parte del estilo se realiza a través de `.callout`. Luego, los estilos únicos entre cada variante se controlan a través de la clase modificadora.

```
// Clase base
.callout {}

// Clases modificadoras
.callout-info {}
.callout-warning {}
.callout-danger {}
```

Para los *callouts*, ese estilo único es simplemente un `border-left-color`. Cuando combinas esa clase base con una de las clases modificadoras, obtienes tu familia de componentes completa:

```
<div class="bd-callout bd-callout-info">
  <strong>Esto es un info callout.</strong> Texto de ejemplo para mostrarlo
en acción.
</div>
<div class="bd-callout bd-callout-warning">
  <strong>Esto es un warning callout.</strong> Texto de ejemplo para
mostrarlo en acción.
```

```
</div>
<div class="bd-callout bd-callout-danger">
  <strong>Esto es un danger callout.</strong> Texto de ejemplo para mostrarlo en acción.
</div>
```

Variables CSS de Bootstrap

Utiliza las propiedades personalizadas de CSS de Bootstrap para un diseño y desarrollo rápidos y con visión de futuro.

Bootstrap incluye muchas propiedades personalizadas de CSS (variables) en su CSS compilado para la personalización en tiempo real sin necesidad de volver a compilar Sass. Estas brindan un fácil acceso a los valores de uso común, como nuestros colores de tema, breakpoints y pilas de fuentes primarias, cuando trabajas en el inspector de tu navegador, un sandbox de código o en la creación de prototipos en general.

Todas nuestras propiedades personalizadas tienen el prefijo `bs-` para evitar conflictos con CSS de terceros.

Variables root

Estas son las variables que incluimos (ten en cuenta que se requiere `:root`), a las que se puede acceder desde cualquier lugar donde se cargue el CSS de Bootstrap. Están ubicados en nuestro archivo `_root.scss` e incluidos en nuestros archivos dist compilados.

```
:root {
  --bs-blue: #0d6efd;
  --bs-indigo: #6610f2;
  --bs-purple: #6f42c1;
  --bs-pink: #d63384;
  --bs-red: #dc3545;
  --bs-orange: #fd7e14;
  --bs-yellow: #ffc107;
  --bs-green: #198754;
  --bs-teal: #20c997;
  --bs-cyan: #0dcaf0;
  --bs-white: #fff;
  --bs-gray: #6c757d;
  --bs-gray-dark: #343a40;
  --bs-gray-100: #f8f9fa;
  --bs-gray-200: #e9ecef;
  --bs-gray-300: #dee2e6;
  --bs-gray-400: #ced4da;
  --bs-gray-500: #adb5bd;
  --bs-gray-600: #6c757d;
  --bs-gray-700: #495057;
  --bs-gray-800: #343a40;
  --bs-gray-900: #212529;
  --bs-primary: #0d6efd;
  --bs-secondary: #6c757d;
  --bs-success: #198754;
  --bs-info: #0dcaf0;
  --bs-warning: #ffc107;
  --bs-danger: #dc3545;
  --bs-light: #f8f9fa;
  --bs-dark: #212529;
```

```
  --bs-primary-rgb: 13, 110, 253;
  --bs-secondary-rgb: 108, 117, 125;
  --bs-success-rgb: 25, 135, 84;
  --bs-info-rgb: 13, 202, 240;
  --bs-warning-rgb: 255, 193, 7;
  --bs-danger-rgb: 220, 53, 69;
  --bs-light-rgb: 248, 249, 250;
  --bs-dark-rgb: 33, 37, 41;
  --bs-white-rgb: 255, 255, 255;
  --bs-black-rgb: 0, 0, 0;
  --bs-body-color-rgb: 33, 37, 41;
  --bs-body-bg-rgb: 255, 255, 255;
  --bs-font-sans-serif: system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial, "Noto Sans", "Liberation Sans", sans-serif, "Apple
Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";
  --bs-font-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation
Mono", "Courier New", monospace;
  --bs-gradient: linear-gradient(180deg, rgba(255, 255, 255, 0.15), rgba(255,
255, 255, 0));
  --bs-body-font-family: var(--bs-font-sans-serif);
  --bs-body-font-size: 1rem;
  --bs-body-font-weight: 400;
  --bs-body-line-height: 1.5;
  --bs-body-color: #212529;
  --bs-body-bg: #fff;
}
```

Component variables

También estamos comenzando a utilizar propiedades personalizadas como variables locales para varios componentes. De esta manera, podemos reducir nuestro CSS compilado, asegurarnos de que los estilos no se hereden en lugares como tablas anidadas y permitir algunos cambios de estilo básicos y la ampliación de los componentes de Bootstrap después de la compilación de Sass.

Echa un vistazo a nuestra documentación de tabla para obtener algunos conocimientos sobre cómo usamos las variables CSS.

También estamos usando variables CSS en nuestras cuadrículas (grid), principalmente para gutters, con más uso de componentes en el futuro.

Ejemplos

Las variables de CSS ofrecen una flexibilidad similar a las variables de Sass, pero sin necesidad de compilación antes de ser enviadas al navegador. Por ejemplo, aquí estamos restableciendo la fuente de nuestra página y los estilos de enlaces con variables CSS.

```
body {
  font: 1rem/1.5 var(--bs-font-sans-serif);
}
a {
  color: var(--bs-blue);
}
```

Grid breakpoints

Si bien incluimos nuestros breakpoints de cuadrícula como variables CSS (excepto `xs`), ten en cuenta que **las variables CSS no funcionan en las media queries**. Esto es por diseño en la especificación CSS para variables, pero puede cambiar en los próximos años con soporte para variables `env()`. Consulta esta respuesta de Stack Overflow para obtener algunos enlaces útiles. Mientras tanto, puedes usar estas variables en otras situaciones de CSS, así como en tu JavaScript.

Formas para optimizar Bootstrap

Mantén tus proyectos ágiles, responsive y mantenibles para que puedas brindar la mejor experiencia y concentrarte en trabajos más importantes.

Importaciones Sass ligeras

Cuando utilices Sass en tus assets, asegúrate de optimizar Bootstrap solo importando (`@import`) los componentes que necesitas. Tus optimizaciones más grandes probablemente provendrán de la sección `Layout & Components` de nuestro `bootstrap.scss`.

```scss
// Configuration
@import "functions";
@import "variables";
@import "maps";
@import "mixins";
@import "utilities";

// Layout & components
@import "root";
@import "reboot";
@import "type";
@import "images";
@import "containers";
@import "grid";
@import "tables";
@import "forms";
@import "buttons";
@import "transitions";
@import "dropdown";
@import "button-group";
@import "nav";
@import "navbar";
@import "card";
@import "accordion";
@import "breadcrumb";
@import "pagination";
@import "badge";
@import "alert";
@import "progress";
@import "list-group";
@import "close";
@import "toasts";
@import "modal";
@import "tooltip";
@import "popover";
@import "carousel";
@import "spinners";
@import "offcanvas";
```

```
@import "placeholders";

// Helpers
@import "helpers";

// Utilities
@import "utilities/api";
```

Si no estás utilizando un componente, coméntalo o elimínalo por completo. Por ejemplo, si no estás utilizando el *carousel*, elimina esa importación para ahorrar algo de tamaño de archivo en tu CSS compilado. Ten en cuenta que existen algunas dependencias en las importaciones de Sass que pueden dificultar la omisión de un archivo.

JavaScript ligero

El JavaScript de Bootstrap incluye todos los componentes de nuestros archivos de distribución principales (`bootstrap.js` y `bootstrap.min.js`), e incluso nuestra dependencia principal (Popper) con nuestros archivos de paquete (`bootstrap.bundle.js` y `bootstrap.bundle.min.js`). Mientras personalizas a través de Sass, asegúrate de eliminar el JavaScript relacionado.

Por ejemplo, suponiendo que estás usando tu propio paquete de JavaScript como Webpack o Rollup, solo importarás el JavaScript que planeas usar. En el siguiente ejemplo, mostramos cómo incluir solamente nuestro JavaScript modal:

```
// Import just what we need

// import 'bootstrap/js/dist/alert';
// import 'bootstrap/js/dist/button';
// import 'bootstrap/js/dist/carousel';
// import 'bootstrap/js/dist/collapse';
// import 'bootstrap/js/dist/dropdown';
import 'bootstrap/js/dist/modal';
// import 'bootstrap/js/dist/offcanvas';
// import 'bootstrap/js/dist/popover';
// import 'bootstrap/js/dist/scrollspy';
// import 'bootstrap/js/dist/tab';
// import 'bootstrap/js/dist/toast';
// import 'bootstrap/js/dist/tooltip';
```

De esta forma, no incluyes ningún JavaScript que no pretendas utilizar para componentes como botones, carruseles e tooltips. Si estás importando menús desplegables, tooltips o ventanas emergentes (popovers), asegúrate de incluir la dependencia de Popper en tu archivo `package.json`.

Default Exports

Los archivos en `bootstrap/js/dist` usan **default export** (exportación predeterminada), por lo que si deseas usar uno de ellos, debes hacer lo siguiente:

```
import Modal from 'bootstrap/js/dist/modal'

const modal = new Modal(document.getElementById('myModal'))
```

Autoprefixer .browserslistrc

Bootstrap depende de Autoprefixer para agregar automáticamente prefijos de navegador a ciertas propiedades de CSS. Los prefijos están dictados por nuestro archivo `.browserslistrc`, que se encuentra en la raíz del repositorio de Bootstrap. Personalizar esta lista de navegadores y volver a compilar el Sass eliminará automáticamente algunos CSS de tu CSS compilado si hay prefijos de proveedores exclusivos para ese navegador o versión.

CSS sin usar

Se necesita ayuda con esta sección, considere abrir un PR. ¡Gracias!

Si bien no tenemos un ejemplo preconstruido para usar PurgeCSS con Bootstrap, hay algunos artículos y tutoriales útiles que ha escrito la comunidad. Aquí hay algunas opciones:

- https://medium.com/dwarves-foundation/remove-unused-css-styles-from-bootstrap-using-purgecss-88395a2c5772
- https://lukelowrey.com/automatically-removeunused-css-from-bootstrap-or-other-frameworks/

Por último, este artículo de CSS Tricks sobre CSS sin usar muestra cómo usar PurgeCSS y otras herramientas similares.

Minificar y gzip

Siempre que sea posible, asegúrate de comprimir todo el código que ofreces a tus visitantes. Si estás utilizando archivos dist de Bootstrap, intenta inclinarte por las versiones minificadas (indicadas por las extensiones `.min.css` y `.min.js`). Si estás compilando Bootstrap desde la fuente con tu propio sistema de compilación, asegúrate de implementar tus propios minificadores para HTML, CSS y JS.

Archivos sin bloqueo

Si bien minimizar y usar la compresión puede parecer suficiente, hacer que tus archivos no bloqueen también es un gran paso para que tu sitio esté bien optimizado y sea lo suficientemente rápido.

Si estás utilizando un complemento Lighthouse en Google Chrome, es posible que hayas tropezado con FCP. La métrica The First Contentful Paint mide el tiempo desde que la página comienza a cargarse hasta que cualquier parte del contenido de la página se representa en la pantalla.

Puedes mejorar FCP aplazando JavaScript o CSS no críticos. ¿Qué significa eso? Simplemente, JavaScript o las hojas de estilo que no necesitan estar presentes en la primera pintura de tu página deben marcarse con los atributos `async` o `defer`.

Esto asegura que los recursos menos importantes se carguen más tarde y no bloqueen la primera pintura. Por otro lado, los recursos críticos se pueden incluir como scripts o estilos en línea.

Si deseas obtener más información sobre esto, ya hay muchos artículos excelentes al respecto:

- https://web.dev/render-blocking-resources/
- https://web.dev/defer-non-critical-css/

Utiliza siempre HTTPS

Tu sitio web solo debe estar disponible a través de conexiones HTTPS en producción. HTTPS mejora la seguridad, la privacidad y la disponibilidad de todos los sitios, y no existe el tráfico web no confidencial. Los pasos para configurar tu sitio web para que se sirva exclusivamente a través de HTTPS varían ampliamente según tu arquitectura y proveedor de alojamiento web y, por lo tanto, están más allá del alcance de estos documentos.

Los sitios servidos a través de HTTPS también deben acceder a todas las hojas de estilo, scripts y otros activos a través de conexiones HTTPS. De lo contrario, enviarás a los usuarios contenido activo mixto, lo que generará vulnerabilidades potenciales en las que un sitio puede verse comprometido al alterar una dependencia. Esto puede generar problemas de seguridad y advertencias en el navegador que se muestran a los usuarios. Ya sea que estés obteniendo Bootstrap de un CDN o sirviéndolo tú mismo, asegúrate de acceder solo a través de conexiones HTTPS.

Uso de los Breakpoints de Bootstrap

Los breakpoints son anchos personalizables que determinan cómo se comporta tu diseño responsive en los tamaños de dispositivo o viewport en Bootstrap.

Conceptos básicos

- **Los breakpoints son los componentes básicos del diseño responsive.** Úsalos para controlar cuándo se puede adaptar tu diseño a un viewport particular o tamaño de dispositivo.
- **Utiliza media queries para diseñar tu CSS por breakpoint.** Las media queries son una función de CSS que te permite aplicar estilos de forma condicional en función de un conjunto de parámetros del navegador y del sistema operativo. Comúnmente usamos `min-width` en nuestras media queries.
- **Mobile first, el diseño responsive es el objetivo.** El CSS de Bootstrap tiene como objetivo aplicar el mínimo de estilos para hacer que un diseño funcione en el breakpoint más pequeño, y luego capas de estilos para ajustar ese diseño para dispositivos más grandes. Esto optimiza tu CSS, mejora el tiempo de renderizado y proporciona una gran experiencia para tus visitantes.

Breakpoints disponibles

Bootstrap incluye seis breakpoints predeterminados, a veces denominados *niveles de cuadrícula*, para compilar de forma responsive. Estos breakpoints se pueden personalizar si estás utilizando nuestros archivos fuente Sass.

Breakpoint	Infijo de clase	Dimensiones
X-Small	*None*	<576px
Small	`Sm`	≥576px
Medium	`Md`	≥768px

Breakpoint	Infijo de clase	Dimensiones
Large	Lg	≥992px
Extra large	Xl	≥1200px
Extra extra large	Xxl	≥1400px

Cada breakpoint se eligió para contener cómodamente contenedores cuyos anchos son múltiplos de 12. Los breakpoints también son representativos de un subconjunto de tamaños de dispositivos comunes y dimensiones de viewports; no se dirigen específicamente a cada caso de uso o dispositivo. En cambio, los rangos brindan una base sólida y consistente para desarrollar casi cualquier dispositivo.

Estos breakpoints se pueden personalizar a través de Sass; los encontrarás en un mapa de Sass en nuestra hoja de estilo `_variables.scss`.

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

Para obtener más información y ejemplos sobre cómo modificar nuestros mapas y variables Sass, consulta la sección Sass de la documentación de Grid.

Media queries

Dado que Bootstrap está desarrollado para ser Mobile first, usamos un puñado de media queries para crear breakpoints sensibles para nuestros diseños e interfaces. Estos breakpoints se basan principalmente en anchos mínimos de viewports y nos permiten aumentar la escala de los elementos a medida que cambia la ventana gráfica.

Min-width

Bootstrap utiliza principalmente los siguientes rangos de media queries, o breakpoints, en nuestros archivos fuente Sass para nuestro diseño, sistema de cuadrícula y componentes.

```
// Source mixins

// No es necesaria una media query para el breakpoint xs, ya que es
efectivamente `@media (min-width: 0) { ... }`
@include media-breakpoint-up(sm) { ... }
@include media-breakpoint-up(md) { ... }
@include media-breakpoint-up(lg) { ... }
@include media-breakpoint-up(xl) { ... }
@include media-breakpoint-up(xxl) { ... }

// Uso

// Ejemplo: Ocultar comenzando en `min-width: 0`, y luego mostrar en el
breakpoint `sm`
```

```scss
.custom-class {
  display: none;
}
@include media-breakpoint-up(sm) {
  .custom-class {
    display: block;
  }
}
```

Estos mixins de Sass se traducen en nuestro CSS compilado utilizando los valores declarados en nuestras variables de Sass. Por ejemplo:

```scss
// Dispositivos X-Small (teléfonos verticales, menos de 576px)
// No hay media queries para `xs` ya que este es el valor predeterminado en Bootstrap

// Dispositivos pequeños (teléfonos horizontales, 576px y más)
@media (min-width: 576px) { ... }

// Dispositivos medianos (tabletas, 768px y más)
@media (min-width: 768px) { ... }

// Dispositivos grandes (escritorios, 992px y más)
@media (min-width: 992px) { ... }

// Dispositivos X-Large (escritorios grandes, 1200px y más)
@media (min-width: 1200px) { ... }

// Dispositivos XX-Large (escritorios más grandes, 1400px y más)
@media (min-width: 1400px) { ... }
```

Max-width

Ocasionalmente usamos media queries que van en la otra dirección (el tamaño de pantalla dado *o más pequeño*):

```scss
// No es necesaria una media query para el punto de interrupción xs, ya que es efectivamente `@media (max-width: 0) { ... }`
@include media-breakpoint-down(sm) { ... }
@include media-breakpoint-down(md) { ... }
@include media-breakpoint-down(lg) { ... }
@include media-breakpoint-down(xl) { ... }
@include media-breakpoint-down(xxl) { ... }

// Ejemplo: Estilo desde el breakpoint medium y hacia abajo
@include media-breakpoint-down(md) {
  .custom-class {
    display: block;
  }
}
```

Estos mixins toman esos breakpoints declarados, restan `.02px` de ellos y los usan como nuestros valores de `max-width`. Por ejemplo:

```scss
// `xs` devuelve solo un conjunto de reglas y ninguna media query
// ... { ... }

// `sm` se aplica a dispositivos x-small (teléfonos verticales, menos de 576px)
@media (max-width: 575.98px) { ... }
```

```scss
// `md` se aplica a dispositivos small (teléfonos horizontales, menos de 768px)
@media (max-width: 767.98px) { ... }

// `lg` se aplica a dispositivos medium (tabletas, menos de 992px)
@media (max-width: 991.98px) { ... }

// `xl` se aplica a dispositivos large (escritorios, menos de 1200px)
@media (max-width: 1199.98px) { ... }

// `xxl` se aplica a dispositivos x-large (escritorios grandes, menos de 1400px)
@media (max-width: 1399.98px) { ... }
```

¿Por qué restar .02px? Actualmente, los navegadores no admiten consultas de contexto de rango, por lo que solucionamos las limitaciones de prefijos `min-` y `max-` y viewports con anchos fraccionarios (que pueden ocurrir bajo ciertas condiciones en dispositivos de dpi altos, por ejemplo) utilizando valores con mayor precisión.

Single breakpoint

También hay media queries y mixins para apuntar a un solo segmento de tamaños de pantalla utilizando los anchos de breakpoint mínimo y máximo.

```scss
@include media-breakpoint-only(xs) { ... }
@include media-breakpoint-only(sm) { ... }
@include media-breakpoint-only(md) { ... }
@include media-breakpoint-only(lg) { ... }
@include media-breakpoint-only(xl) { ... }
@include media-breakpoint-only(xxl) { ... }
```

Por ejemplo, `@include media-breakpoint-only(md) { ... }` dará como resultado:

```scss
@media (min-width: 768px) and (max-width: 991.98px) { ... }
```

Entre breakpoints

De manera similar, las media queries pueden abarcar múltiples anchos de breakpoint:

```scss
@include media-breakpoint-between(md, xl) { ... }
```

Lo que resulta en:

```scss
// Ejemplo
// Aplicar estilos a partir de dispositivos medianos y hasta dispositivos extra grandes
@media (min-width: 768px) and (max-width: 1199.98px) { ... }
```

Uso de Contenedores en Bootstrap

Los contenedores son un bloque de construcción fundamental de Bootstrap que contienen, rellenan y alinean tu contenido dentro de un dispositivo o viewport determinado.

Cómo trabajan

Los contenedores son el elemento de diseño más básico en Bootstrap y son **requeridos cuando se usa nuestro sistema de cuadrícula predeterminado**. Los contenedores se utilizan para contener, rellenar y (a veces) centrar el contenido dentro de ellos. Si bien los

contenedores *pueden* anidarse, la mayoría de los diseños no requieren un contenedor anidado.

Bootstrap viene con tres contenedores diferentes:

- `.container`, que establece un `max-width` en cada breakpoint responsive
- `.container-fluid`, que es `width: 100%` en todos los breakpoints
- `.container-{breakpoint}`, que es `width: 100%` hasta el breakpoint especificado

La siguiente tabla ilustra cómo se compara el `max-width` de cada contenedor con el `.container` y el `.container-fluid` originales en cada breakpoint.

Puedes verlo en acción y compararlos en nuestro ejemplo de cuadrícula.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

Contenedor predeterminado

Nuestra clase `.container` predeterminada es un contenedor sensible de ancho fijo, lo que significa que su `max-width` cambia en cada breakpoint.

```
<div class="container">
  <!-- Contenido aquí -->
</div>
```

Contenedores responsive

Los contenedores responsive te permiten especificar una clase que tiene un 100% de ancho hasta que se alcanza el breakpoint especificado, después de lo cual aplicamos `max-width`s para cada uno de los breakpoints más altos. Por ejemplo, `.container-sm` tiene un 100% de ancho al principio hasta que se alcanza el breakpoint `sm`, donde se escalará con `md`, `lg`, `xl` y `xxl`.

```
<div class="container-sm">100% de ancho hasta el small breakpoint</div>
<div class="container-md">100% de ancho hasta el medium breakpoint</div>
<div class="container-lg">100% de ancho hasta el large breakpoint</div>
<div class="container-xl">100% de ancho hasta el extra large breakpoint</div>
<div class="container-xxl">100% de ancho hasta el extra extra large
breakpoint</div>
```

Contenedores fluidos

Usa `.container-fluid` para un contenedor de ancho completo, que abarque todo el ancho del viewport.

```
<div class="container-fluid">
  ...
</div>
```

Sass

Como se muestra arriba, Bootstrap genera una serie de clases de contenedores predefinidas para ayudarte a crear los diseños que deseas. Puedes personalizar estas clases de contenedor predefinidas modificando el mapa de Sass (que se encuentra en `_variables.scss`):

```
$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
  xl: 1140px,
  xxl: 1320px
);
```

Además de personalizar el Sass, también puedes crear tus propios contenedores con nuestro mixin Sass.

```
// Source mixin
@mixin make-container($padding-x: $container-padding-x) {
  width: 100%;
  padding-right: $padding-x;
  padding-left: $padding-x;
  margin-right: auto;
  margin-left: auto;
}

// Uso
.custom-container {
  @include make-container();
}
```

Para obtener más información y ejemplos sobre cómo modificar nuestros mapas y variables Sass, consulta la sección Sass de la documentación de Grid.

Sistema de cuadrícula (Grid) en Bootstrap

Utiliza nuestra potente cuadrícula flexbox mobile-first para crear diseños de todas las formas y tamaños gracias a un sistema de doce columnas, seis niveles responsive (breakpoints) predeterminados, variables Sass y mixins, y docenas de clases predefinidas.

Ejemplo

El sistema de cuadrícula (grid) de Bootstrap utiliza una serie de contenedores, filas y columnas para diseñar y alinear el contenido. Está construido con flexbox y es completamente responsive. A continuación se muestra un ejemplo y una explicación detallada de cómo se compone el sistema de cuadrículo.

¿Eres nuevo o no estás familiarizado con flexbox? Lee esta guía de flexbox de CSS Tricks para conocer los antecedentes, terminología, directrices y fragmentos de código.

```
<div class="container">
  <div class="row">
    <div class="col">
      Columna
    </div>
    <div class="col">
      Columna
    </div>
    <div class="col">
      Columna
    </div>
  </div>
</div>
```

El ejemplo anterior crea tres columnas de igual ancho en todos los dispositivos y viewports utilizando nuestras clases de cuadrícula predefinidas. Esas columnas están centradas en la página con el `.container` principal.

¿Cómo funciona el sistema de cuadrícula (Grid)?

Desglosándolo, así es como se compone el sistema de cuadrícula:

- **Nuestra cuadrícula admite seis breakpoints responsive.** Los breakpoints se basan en media queries `min-width`, lo que significa que afectan ese breakpoint y todos los anteriores (por ejemplo, `.col-sm-4` se aplica a `sm`, `md`, `lg`, `xl` y `xxl`). Esto significa que puedes controlar el tamaño y el comportamiento del contenedor y la columna en cada breakpoint.
- **Los contenedores centran y rellenan horizontalmente su contenido.** Usa `.container` para un ancho de píxel responsive, `.container-fluid` para `width: 100%` en todos los viewports y dispositivos, o un contenedor responsive (p. ej., `.container-md`) para una combinación entre fluido y anchos en píxeles.
- **Las filas son envoltorios para las columnas.** Cada columna tiene un `padding` horizontal (llamado *gutter*) para controlar el espacio entre ellas. Este `padding` luego se contrarresta en las filas con márgenes negativos para garantizar que el contenido de tus columnas esté alineado visualmente en el lado izquierdo. Las filas también admiten clases modificadoras para aplicar uniformemente el tamaño de columna y clases de *gutter* para cambiar el espaciado de tu contenido.
- **Las columnas son increíblemente flexibles.** Hay 12 columnas de plantilla disponibles por fila, lo que te permite crear diferentes combinaciones de elementos que abarcan cualquier número de columnas. Las clases de columna indican el número de columnas de plantilla que abarcan (por ejemplo, `col-4` abarca cuatro). `width`s se establecen en porcentajes para que siempre tengas el mismo tamaño relativo.
- **Los gutters también son responsive y personalizables.** Las clases *Gutter* están disponibles en todos los breakpoints, con los mismos tamaños que nuestro espaciado de margin y padding. Cambia los gutters horizontales con las clases `.gx-*`, los gutters verticales

con `.gy-*` o todos los gutters con las clases `.g-*`. `.g-0` también está disponible para eliminar gutters.
- **Las variables Sass, los mapas y los mixins alimentan la cuadrícula.** Si no deseas usar las clases de cuadrícula predefinidas en Bootstrap, puedes usar nuestra código fuente Sass de cuedrículas para crear el tuyo propio con más marcado semántico. También incluimos algunas propiedades personalizadas de CSS para consumir estas variables de Sass para una mayor flexibilidad.

Ten en cuenta las limitaciones y los errores relacionados con flexbox, como la incapacidad de usar algunos elementos HTML como contenedores flexibles.

Opciones de cuadrícula

El sistema de cuadrícula de Bootstrap puede adaptarse a los seis breakpoints predeterminados y a cualquier breakpoint que personalices. Los seis niveles de cuadrícula predeterminados son los siguientes:

- Extra small (xs)
- Small (sm)
- Medium (md)
- Large (lg)
- Extra large (xl)
- Extra extra large (xxl)

Como se indicó anteriormente, cada uno de estos breakpoints tienen su propio contenedor, prefijo de clase único y modificadores. Así es como cambia la cuadrícula en estos breakpoints:

	xs <576px	**sm ≥576px**	**md ≥768px**	**lg ≥992px**	**xl ≥1200px**	**xxl ≥1400px**
Contenedor `max-width`	None (auto)	540px	720px	960px	1140px	1320px
Prefijo de clase	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`
# de columnas	12					
Ancho de gutter	1.5rem (.75rem a izquierda y derecha)					
Gutters personalizados	Si					
Anidable	Si					
Ordenamiento de columnas	Si					

Columnas con auto-layout

Utiliza clases de columnas de breakpoints específicos para facilitar la asignación de tamaño de las columnas sin una clase numerada explícita como `.col-sm-6`.

Anchos iguales

Por ejemplo, aquí hay dos diseños de cuadrícula que se aplican a cada dispositivo y viewport, desde `xs` hasta `xxl`. Agrega cualquier número de clases sin unidades para cada breakpoint que necesites y cada columna tendrá el mismo ancho.

```
<div class="container">
  <div class="row">
    <div class="col">
      1 de 2
    </div>
    <div class="col">
      2 de 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 de 3
    </div>
    <div class="col">
      2 de 3
    </div>
    <div class="col">
      3 de 3
    </div>
  </div>
</div>
```

Establecer un ancho de columna

El auto-layout para las columnas de cuadrícula de flexbox también significa que puedes establecer el ancho de una columna y hacer que las columnas hermanas cambien de tamaño automáticamente a su alrededor. Puedes usar clases de cuadrícula predefinidas (como se muestra a continuación), combinaciones de cuadrícula o anchos en línea. Ten en cuenta que las otras columnas cambiarán de tamaño sin importar el ancho de la columna central.

```
<div class="container">
  <div class="row">
    <div class="col">
      1 de 3
    </div>
    <div class="col-6">
      2 de 3 (más ancho)
    </div>
    <div class="col">
      3 de 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 de 3
    </div>
    <div class="col-5">
      2 de 3 (más ancho)
    </div>
    <div class="col">
      3 de 3
    </div>
```

```
  </div>
</div>
```

Contenido de ancho variable

Usa las clases `col-{breakpoint}-auto` para dimensionar las columnas en función del ancho natural de su contenido.

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 de 3
    </div>
    <div class="col-md-auto">
      Contenido de ancho variable
    </div>
    <div class="col col-lg-2">
      3 de 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 de 3
    </div>
    <div class="col-md-auto">
      Contenido de ancho variable
    </div>
    <div class="col col-lg-2">
      3 de 3
    </div>
  </div>
</div>
```

Clases responsive

La cuadrícula de Bootstrap incluye seis niveles de clases predefinidas para crear diseños responsive complejos. Personaliza el tamaño de tus columnas en dispositivos extra pequeños, pequeños, medianos, grandes o extra grandes como mejor te parezca.

Todos los breakpoints

Para cuadrículas que son iguales desde el dispositivo más pequeño hasta el más grande, usa las clases `.col` y `.col-*`. Especifica una clase numerada cuando necesites una columna de tamaño particular; de lo contrario, siéntate libre de apegarte a `.col`.

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
  </div>
  <div class="row">
    <div class="col-8">col-8</div>
    <div class="col-4">col-4</div>
  </div>
</div>
```

De apilado a horizontal

Usando un solo conjunto de clases `.col-sm-*`, puedes crear un sistema de cuadrícula básico que comienza apilado y se vuelve horizontal en el breakpoint pequeño (`sm`).

```
<div class="container">
  <div class="row">
    <div class="col-sm-8">col-sm-8</div>
    <div class="col-sm-4">col-sm-4</div>
  </div>
  <div class="row">
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
    <div class="col-sm">col-sm</div>
  </div>
</div>
```

Mezclar y combinar

¿No quieres que tus columnas simplemente se apilen en algunos niveles de cuadrícula? Usa una combinación de diferentes clases para cada nivel según sea necesario. Mira el ejemplo a continuación para tener una mejor idea de cómo funciona todo.

```
<div class="container">
  <!-- Apila las columnas en dispositivos móviles haciendo una de ancho completo y la otra de ancho medio -->
  <div class="row">
    <div class="col-md-8">.col-md-8</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Las columnas comienzan con un 50% de ancho en dispositivos móviles y cambian hasta un 33,3% de ancho en computadoras de escritorio -->
  <div class="row">
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
    <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  </div>

  <!-- Las columnas siempre tienen un 50% de ancho, en dispositivos móviles y de escritorio -->
  <div class="row">
    <div class="col-6">.col-6</div>
    <div class="col-6">.col-6</div>
  </div>
</div>
```

Columnas de fila

Usa las clases responsive `.row-cols-*` para establecer rápidamente la cantidad de columnas que mejor representen tu contenido y diseño. Mientras que las clases `.col-*` normales se aplican a las columnas individuales (p. ej., `.col-md-4`), las clases de *columnas de fila* se establecen en el `.row` principal como acceso directo. Con `.row-cols-auto` puedes dar a las columnas su ancho natural.

Utiliza estas clases de filas y columnas para crear rápidamente diseños de cuadrícula básicos o para controlar los diseños de tus tarjetas.

```
<div class="container">
  <div class="row row-cols-2">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
<div class="container">
  <div class="row row-cols-3">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
<div class="container">
  <div class="row row-cols-auto">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
<div class="container">
  <div class="row row-cols-4">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col-6">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
<div class="container">
  <div class="row row-cols-1 row-cols-sm-2 row-cols-md-4">
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
    <div class="col">Columna</div>
  </div>
</div>
```

También puedes usar el mixin de Sass que lo acompaña, `row-cols()`:

```
.element {
  // Tres columnas al inicio
  @include row-cols(3);

  // Cinco columnas desde el breakpoint medium hacia arriba
  @include media-breakpoint-up(md) {
    @include row-cols(5);
  }
}
```

Anidamiento

Para anidar tu contenido con la cuadrícula predeterminada, agrega un nuevo `.row` y un conjunto de columnas `.col-sm-*` dentro de una columna `.col-sm-*` existente. Las filas anidadas deben incluir un conjunto de columnas que suman 12 o menos (no es necesario que uses las 12 columnas disponibles).

```
<div class="container">
  <div class="row">
    <div class="col-sm-3">
      Nivel 1: .col-sm-3
    </div>
    <div class="col-sm-9">
      <div class="row">
        <div class="col-8 col-sm-6">
          Nivel 2: .col-8 .col-sm-6
        </div>
        <div class="col-4 col-sm-6">
          Nivel 2: .col-4 .col-sm-6
        </div>
      </div>
    </div>
  </div>
</div>
```

Sass

Al usar los archivos fuente Sass de Bootstrap, tienes la opción de usar variables Sass y mixins para crear diseños de página personalizados, semánticos y responsive. Nuestras clases de cuadrícula predefinidas utilizan estas mismas variables y mixins para proporcionar un conjunto completo de clases listas para usar para rápidos diseños responsive.

Variables

Las variables y los mapas determinan el número de columnas, el ancho del gutter y el punto de media query en el que comienzan las columnas flotantes. Los usamos para generar las clases de cuadrícula predefinidas documentadas anteriormente, así como para los mixins personalizados que se enumeran a continuación.

```
$grid-columns:      12;
$grid-gutter-width: 1.5rem;
```

```
$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);
$container-max-widths: (
  sm: 540px,
  md: 720px,
  lg: 960px,
```

```
  xl: 1140px,
  xxl: 1320px
);
```

Mixins

Los mixins se utilizan junto con las variables de cuadrícula para generar CSS semántico para columnas de cuadrícula individuales.

```
// Crea un contenedor para una serie de columnas
@include make-row();

// Hacer que el elemento esté listo para la cuadrícula (aplicando todo menos el ancho)
@include make-col-ready();

// Sin valores de tamaño opcionales, el mixin creará columnas iguales (similar a usar .col)
@include make-col();
@include make-col($size, $columns: $grid-columns);

// Compensar con márgenes
@include make-col-offset($size, $columns: $grid-columns);
```

Ejemplo de uso

Puedes modificar las variables a tus propios valores personalizados, o simplemente usar los mixins con sus valores predeterminados. Este es un ejemplo del uso de la configuración predeterminada para crear un diseño de dos columnas con un espacio entre ellas.

```
.example-container {
  @include make-container();
  // Asegúrate de definir este ancho después del mixin para sobrescribir
  // `width: 100%` generado por `make-container()`
  width: 800px;
}

.example-row {
  @include make-row();
}

.example-content-main {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(8);
  }
}

.example-content-secondary {
  @include make-col-ready();

  @include media-breakpoint-up(sm) {
    @include make-col(6);
  }
  @include media-breakpoint-up(lg) {
    @include make-col(4);
  }
```

```
}
<div class="example-container">
  <div class="example-row">
    <div class="example-content-main">Contenido principal</div>
    <div class="example-content-secondary">Contenido secundario</div>
  </div>
</div>
```

Personalización de la cuadrícula

Utilizando nuestras variables y mapas Sass de cuadrícula incorporados, es posible personalizar completamente las clases de cuadrícula predefinidas. Cambia la cantidad de niveles, las dimensiones de las media queries y los anchos de los contenedores, luego vuelve a compilar.

Columnas y gutters

El número de columnas de la cuadrícula se puede modificar mediante variables Sass. `$grid-columns` se utiliza para generar los anchos (en porcentaje) de cada columna individual, mientras que `$grid-gutter-width` establece el ancho de los gutters (espacios de separación) de las columnas.

```
$grid-columns: 12 !default;
$grid-gutter-width: 1.5rem !default;
```

Niveles de cuadrícula

Yendo más allá de las columnas en sí, también puedes personalizar la cantidad de niveles de cuadrícula. Si quisieras solo cuatro niveles de cuadrícula, actualizarías `$grid-breakpoints` y `$container-max-widths` a algo como esto:

```
$grid-breakpoints: (
  xs: 0,
  sm: 480px,
  md: 768px,
  lg: 1024px
);

$container-max-widths: (
  sm: 420px,
  md: 720px,
  lg: 960px
);
```

Al realizar cambios en las variables o mapas de Sass, deberás guardar los cambios y volver a compilar. Al hacerlo, se generará un nuevo conjunto de clases de cuadrícula predefinidas para anchos de columna, compensaciones y ordenación. Las utilidades de visibilidad responsive también se actualizarán para usar los breakpoints personalizados. Asegúrate de establecer los valores de cuadrícula en `px` (no en `rem`, `em` o `%`).

Columnas para el diseño en Bootstrap

Aprende a modificar columnas con un puñado de opciones de alineación, orden y compensación gracias a nuestro sistema de cuadrícula flexbox. Además, conoce cómo usar clases de columna para administrar anchos de elementos que no son de cuadrícula.

Importante! Asegúrate de leer la página de cuadrícula antes de sumergirte en cómo modificar y personalizar las columnas de tu cuadrícula.

¿Cómo funcionan las Columnas?

- **Las columnas se basan en la arquitectura flexbox de la cuadrícula.** Flexbox significa que tenemos opciones para cambiar columnas individuales y modificar grupos de columnas en el nivel de fila. Tú eliges cómo crecen, se reducen o cambian las columnas.
- **Al crear diseños de cuadrícula, todo el contenido va en columnas.** La jerarquía de la cuadrícula de Bootstrap va desde container de fila a columna hasta tu contenido. En raras ocasiones, puedes combinar contenido y columna, pero ten en cuenta que puede haber consecuencias no deseadas.
- **Bootstrap incluye clases predefinidas para crear diseños rápidos y responsive.** Con seis breakpoints y una docena de columnas en cada nivel de cuadrícula, tenemos docenas de clases ya construidas para que puedas crear tus diseños deseados. Esto se puede desactivar a través de Sass si lo deseas.

Alineación

Usa las utilidades de alineación de flexbox para alinear columnas vertical y horizontalmente.

Alineamiento vertical

```
<div class="container">
  <div class="row align-items-start">
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
      Una de tres columnas
    </div>
  </div>
  <div class="row align-items-center">
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
      Una de tres columnas
    </div>
  </div>
  <div class="row align-items-end">
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
      Una de tres columnas
    </div>
    <div class="col">
```

```
      Una de tres columnas
    </div>
  </div>
</div>
<div class="container">
  <div class="row">
    <div class="col align-self-start">
      Una de tres columnas
    </div>
    <div class="col align-self-center">
      Una de tres columnas
    </div>
    <div class="col align-self-end">
      Una de tres columnas
    </div>
  </div>
</div>
```

Horizontal alignment

```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
```

```
  <div class="row justify-content-evenly">
    <div class="col-4">
      Una de dos columnas
    </div>
    <div class="col-4">
      Una de dos columnas
    </div>
  </div>
</div>
```

Envoltura de columna

Si se colocan más de 12 columnas dentro de una sola fila, cada grupo de columnas adicionales se ajustará, como una unidad, a una nueva línea.

```
<div class="container">
  <div class="row">
    <div class="col-9">.col-9</div>
    <div class="col-4">.col-4<br>Ya que 9 + 4 = 13 &gt; 12, este div 4-column-wide se envuelve en una nueva línea como una unidad contigua.</div>
    <div class="col-6">.col-6<br>Subsequent columns continue along the new line.</div>
  </div>
</div>
```

Saltos de columna

Dividir columnas en una nueva línea en flexbox requiere un pequeño truco: agrega un elemento con `width: 100%` donde quieras envolver tus columnas en una nueva línea. Normalmente esto se logra con múltiples `.row`s, pero no todos los métodos de implementación pueden dar cuenta de esto.

```
<div class="container">
  <div class="row">
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>

    <!-- Obligar a las siguientes columnas a pasar a una nueva línea -->
    <div class="w-100"></div>

    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>
  </div>
</div>
```

También puedes aplicar esta interrupción en breakpoints específicos con nuestras utilidades de visualización responsive.

```
<div class="container">
  <div class="row">
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>

    <!-- Obligar a las siguientes columnas a pasar a una nueva línea en el breakpoint md y hacia arriba -->
    <div class="w-100 d-none d-md-block"></div>

    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  </div>
</div>
```

Reordenando

Ordenar clases

Usa las clases `.order-` para controlar el **orden visual** de tu contenido. Estas clases son responsive, por lo que puedes establecer el `order` por breakpoint (por ejemplo, `.order-1.order-md-2`). Incluye soporte para `1` a `5` en los seis niveles de cuadrícula.

```
<div class="container">
  <div class="row">
    <div class="col">
      Primero en el DOM, sin ordenamiento aplicado
    </div>
    <div class="col order-5">
      Segundo en el DOM, con un orden más grande
    </div>
    <div class="col order-1">
      Tercero en el DOM, con un orden de 1
    </div>
  </div>
</div>
```

También hay clases responsive `.order-first` y `.order-last` que cambian el `order` de un elemento aplicando `order: -1` y `order: 6`, respectivamente. Estas clases también se pueden mezclar con las clases numeradas `.order-*` según sea necesario.

```
<div class="container">
  <div class="row">
    <div class="col order-last">
      Primero en el DOM, ordenado último
    </div>
    <div class="col">
      Segundo en el DOM, sin ordenamiento
    </div>
    <div class="col order-first">
      Tercero en el DOM, ordenado primero
    </div>
  </div>
</div>
```

Columnas de compensación

Puedes compensar las columnas de la cuadrícula de dos maneras: nuestras clases responsive de cuadrícula `.offset-` y nuestras utilidades de margen. Las clases de cuadrícula se dimensionan para que coincidan con las columnas, mientras que los márgenes son más útiles para diseños rápidos donde el ancho del desplazamiento es variable.

Clases de compensación

Mueve las columnas a la derecha usando las clases `.offset-md-*`. Estas clases aumentan el margen izquierdo de una columna en columnas `*`. Por ejemplo, `.offset-md-4` mueve `.col-md-4` cuatro columnas.

```
<div class="container">
  <div class="row">
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
  </div>
```

```
  <div class="row">
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  </div>
  <div class="row">
    <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
  </div>
</div>
```

Además de borrar las columnas en los breakpoints responsive, es posible que debas restablecer las compensaciones. Observa esto en acción en el ejemplo de cuadrícula.

```
<div class="container">
  <div class="row">
    <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
    <div class="col-sm-5 offset-sm-2 col-md-6 offset-md-0">.col-sm-5 .offset-
sm-2 .col-md-6 .offset-md-0</div>
  </div>
  <div class="row">
    <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-
6</div>
    <div class="col-sm-6 col-md-5 offset-md-2 col-lg-6 offset-lg-0">.col-sm-6
.col-md-5 .offset-md-2 .col-lg-6 .offset-lg-0</div>
  </div>
</div>
```

Utilidades de margen

Con el cambio a flexbox en v4, puedes usar utilidades de margen como `.me-auto` para forzar que las columnas hermanas se separen entre sí.

```
<div class="container">
  <div class="row">
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4 ms-auto">.col-md-4 .ms-auto</div>
  </div>
  <div class="row">
    <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
    <div class="col-md-3 ms-md-auto">.col-md-3 .ms-md-auto</div>
  </div>
  <div class="row">
    <div class="col-auto me-auto">.col-auto .me-auto</div>
    <div class="col-auto">.col-auto</div>
  </div>
</div>
```

Clases de columna independientes

Las clases `.col-*` también se pueden usar fuera de una `.row` para dar a un elemento un ancho específico. Siempre que las clases de columna se utilicen como elementos hijos no directos de una fila, se omiten los rellenos.

```
<div class="col-3 bg-light p-3 border">
  .col-3: width of 25%
</div>
<div class="col-sm-9 bg-light p-3 border">
  .col-sm-9: ancho de 75% por encima del breakpoint sm
```

```
</div>
```

Las clases se pueden usar junto con las utilidades para crear imágenes flotantes responsive. Asegúrate de envolver el contenido en un envoltorio `.clearfix` para borrar el flotante si el texto es más corto.

```
<div class="clearfix">
  <img src="..." class="col-md-6 float-md-end mb-3 ms-md-3" alt="...">

  <p>
    Un párrafo de texto de marcador de posición. Lo estamos usando aquí para mostrar el uso de la clase clearfix. Estamos agregando bastantes frases sin sentido aquí para demostrar cómo las columnas interactúan aquí con la imagen flotante.
  </p>

  <p>
    Como puedes ver, los párrafos se envuelven con gracia alrededor de la imagen flotante. Ahora imaginate cómo se vería esto con algún contenido real aquí, en lugar de solo este aburrido texto de marcador de posición que sigue y sigue, pero que en realidad no transmite información tangible. Simplemente ocupa espacio y en realidad no debe leerse.
  </p>

  <p>
    Y, sin embargo, aquí estás, aún perseverando en la lectura de este texto de marcador de posición, con la esperanza de obtener más información o algún huevo de pascua oculto de contenido. Una broma, tal vez. Desafortunadamente, no hay nada de eso aquí.
  </p>
</div>
```

Los Gutters para el diseño en Bootstrap

Los gutters son el padding entre las columnas, que se utilizan para espaciar y alinear de manera responsive el contenido en el sistema de cuadrícula de Bootstrap.

¿Cómo funcionan los Gutters?

- **Los gutters son los espacios entre el columnas, creados con `padding` horizontal.** Establecemos `padding-right` y `padding-left` en cada columna, y usamos un `margin` negativo para compensar eso al principio y final de cada fila para alinear el contenido.
- **Los gutters comienzan en `1.5rem` (`24px`) de ancho.** Esto nos permite hacer coincidir nuestra cuadrícula con la escala de espaciadores padding y margin.
- **Los gutters se pueden ajustar de forma responsive.** Utiliza clases gutter de breakpoints específicos para modificar gutters horizontales, gutters verticales y todos los gutters.

Gutters horizontales

Las clases `.gx-*` se pueden usar para controlar los anchos de los gutters horizontales. Es posible que sea necesario ajustar el `.container` o `.container-`

`fluid` padre si también se usan gutters más grandes para evitar el desbordamiento no deseado, usando una utilidad de padding coincidente. Por ejemplo, en el siguiente ejemplo hemos aumentado el padding con `.px-4`:

```
<div class="container px-4">
  <div class="row gx-5">
    <div class="col">
     <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
  </div>
</div>
```

Una solución alternativa es agregar un contenedor alrededor de `.row` con la clase `.overflow-hidden`:

```
<div class="container overflow-hidden">
  <div class="row gx-5">
    <div class="col">
     <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
  </div>
</div>
```

Gutters verticales

Las clases `.gy-*` se pueden usar para controlar los anchos de los gutters verticales. Al igual que los gutters horizontales, los gutters verticales pueden provocar un desbordamiento debajo de la `.row` al final de una página. Si esto ocurre, agrega un contenedor alrededor de `.row` con la clase `.overflow-hidden`:

```
<div class="container overflow-hidden">
  <div class="row gy-5">
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
  </div>
</div>
```

Gutters horizontales & verticales

Las clases `.g-*` se pueden usar para controlar los anchos de los gutters horizontales, para el siguiente ejemplo usamos un ancho de gutter más pequeño, por lo que no será necesario agregar la clase contenedora `.overflow-hidden`.

```
<div class="container">
  <div class="row g-2">
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
    <div class="col-6">
      <div class="p-3 border bg-light">Padding de columna personalizado</div>
    </div>
  </div>
</div>
```

Gutters columnas de filas

Las clases de gutters también se pueden agregar a columnas de fila. En el siguiente ejemplo, usamos columnas de fila responsive y clases de gutters responsive.

```
<div class="container">
  <div class="row row-cols-2 row-cols-lg-5 g-2 g-lg-3">
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
    <div class="col">
      <div class="p-3 border bg-light">Columna de fila</div>
    </div>
  </div>
</div>
```

Sin gutters

Los gutters entre columnas en nuestras clases de cuadrícula predefinidas se pueden eliminar con `.g-0`. Esto elimina los `margins` negativos de `.row` y el `padding` horizontal de todas las columnas secundarias inmediatas.

¿Necesitas un diseño de borde a borde? Deja el `.container` o `.container-fluid` principal y agrega `.mx-0` a `.row` para evitar el desbordamiento.

En la práctica, así es como se ve. Ten en cuenta que puedes continuar usándolo con todas las demás clases de cuadrícula predefinidas (incluidos los anchos de columna, los niveles responsive, los reordenamientos y más).

```
<div class="row g-0">
  <div class="col-sm-6 col-md-8">.col-sm-6 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
```

Cambiar los gutters

Las clases se crean a partir del mapa Sass `$gutters` que se hereda del mapa Sass `$spacers`.

```
$grid-gutter-width: 1.5rem;
$gutters: (
  0: 0,
  1: $spacer * .25,
  2: $spacer * .5,
  3: $spacer,
  4: $spacer * 1.5,
  5: $spacer * 3,
);
```

Utilidades para el diseño en Bootstrap

Para un desarrollo responsive y listo para dispositivos móviles, Bootstrap incluye docenas de clases de utilidades para mostrar, ocultar, alinear y espaciar el contenido.

Cambiar el `display`

Utiliza nuestras utilidades de visualización para alternar de manera responsive los valores comunes de la propiedad `display`. Mézclalo con nuestro sistema de cuadrícula, contenido o componentes para mostrarlos u ocultarlos en viewports específicos.

Opciones Flexbox

Bootstrap está construido con flexbox, pero no se ha cambiado la propiedad `display` de todos los elementos a `display: flex`, ya que esto agregaría muchas sobrescrituras innecesarias y cambiaría inesperadamente los comportamientos clave del navegador. La mayoría de nuestros componentes están construidos con flexbox habilitado.

Si necesitas agregar `display: flex` a un elemento, hazlo con `.d-flex` o una de las variantes responsive (por ejemplo, `.d-sm-flex`). Necesitarás esta clase o el valor `display` para permitir el uso de nuestras utilidades flexbox adicionales para ajustar el tamaño, la alineación, el espaciado y más.

Margin y padding

Usa `margin` y `padding` utilidades de espaciado para controlar cómo se espacian y dimensionan los elementos y componentes. Bootstrap incluye una escala de seis niveles para las utilidades de espaciado, basada en una variable `$spacer` predeterminada de valor `1rem`. Elije valores para todos los viewports (p. ej., `.me-3` para `margin-right: 1rem` en LTR) o elije variantes responsive para apuntar a viewports específicos (p. ej., `.me-md-3` para `margin-right: 1rem` —en LTR— comenzando en el punto de interrupción `md`).

Alternar `visibility`

Cuando no es necesario alternar `display`, puedes alternar la propiedad `visibility` de un elemento con nuestras utilidades de visibilidad. Los elementos invisibles seguirán afectando el diseño de la página, pero están visualmente ocultos para los visitantes.

Uso de Z-index en Bootstrap

Si bien no forman parte del sistema de cuadrícula de Bootstrap, los z-index juegan un papel importante en la forma en que nuestros componentes se superponen e interactúan entre sí.

Varios componentes de Bootstrap utilizan `z-index`, la propiedad CSS que ayuda a controlar el diseño al proporcionar un tercer eje para organizar el contenido. Utilizamos una escala de z-index predeterminada en Bootstrap que ha sido diseñada para capas de navegación,tooltips y popovers, modals y más.

Estos valores más altos comienzan en un número arbitrario, lo suficientemente alto y específico para evitar idealmente conflictos. Necesitamos un conjunto estándar de estos en todos nuestros componentes en capas (tooltips, popovers, navbars, dropdowns, modals) para que podamos ser razonablemente consistentes en los comportamientos. No hay razón por la que no pudiéramos haber usado `100+` o `500+`.

No fomentamos la personalización de estos valores individuales; si cambias uno, es probable que necesites cambiarlos todos.

```
$zindex-dropdown:                   1000;
$zindex-sticky:                     1020;
$zindex-fixed:                      1030;
$zindex-offcanvas-backdrop:         1040;
$zindex-offcanvas:                  1045;
$zindex-modal-backdrop:             1050;
$zindex-modal:                      1055;
$zindex-popover:                    1070;
$zindex-tooltip:                    1080;
```

Para manejar los bordes superpuestos dentro de los componentes (por ejemplo, buttons e inputs en input groups), usamos valores bajos de z-index de un solo dígito de `1`, `2`, y `3` para default, hover, y estdos active. En

hover/focus/active, traemos un elemento particular al frente con un valor de `z-index` más alto para mostrar su borde sobre los elementos hermanos.

Cuadrícula CSS en Bootstrap

Aprende a habilitar, usar y personalizar nuestro sistema de diseño alternativo basado en CSS Grid con ejemplos y fragmentos de código.

El sistema de cuadrícula predeterminado de Bootstrap representa la culminación de más de una década de técnicas de diseño de CSS, probadas por millones de personas. Pero también se creó sin muchas de las funciones y técnicas modernas de CSS que vemos en navegadores como el nuevo CSS Grid.

Aviso: ¡nuestro sistema CSS Grid es experimental y está habilitado a partir de la versión 5.1.0! Lo incluimos en el CSS de nuestra documentación para demostrártelo, pero está deshabilitado de manera predeterminada. Sigue leyendo para aprender cómo habilitarlo en tus proyectos.

¿Cómo funciona las cuadrículas CSS?

Con Bootstrap 5, agregamos la opción para habilitar un sistema de cuadrícula separado que se basa en CSS Grid, pero con un toque de Bootstrap. Todavía recibe clases que puedes aplicar por capricho para crear diseños responsive, pero con un enfoque diferente bajo el capó.

- **CSS Grid es opcional.** Desactiva el sistema de cuadrícula predeterminado configurando `$enable-grid-classes: false` y habilita CSS Grid configurando `$enable-cssgrid: true`. Luego, vuelve a compilar tu Sass.
- **Reemplaza las instancias de `.row` con `.grid`.** La clase `.grid` establece `display: grid` y crea un `grid-template` que se construye con tu HTML.
- **Reemplaza las clases `.col-*` con las clases `.g-col-*`.** Esto se debe a que nuestras columnas CSS Grid usan la propiedad `grid-column` en lugar de `width`.
- **Los tamaños de columnas y gutters se establecen a través de variables CSS.** Establécelos en el `.grid` principal y personalízalos como lo desees, en línea o en una hoja de estilo, con `--bs-columns` y `--bs-gap`.

En el futuro, es probable que Bootstrap cambie a una solución híbrida, ya que la propiedad `gap` ha logrado una compatibilidad casi completa del navegador con flexbox.

Diferencias clave

En comparación con el sistema de cuadrícula predeterminado:

- Las utilidades de Flex no afectan las columnas de CSS Grid de la misma manera.
- Los huecos (gaps) reemplazan a los gutters. La propiedad `gap` reemplaza el `padding` horizontal de nuestro sistema de cuadrícula predeterminado y funciona más como `margin`.

- Como tal, a diferencia de `.row`s, `.grid`s no tienen márgenes negativos y las utilidades de margen no se pueden usar para cambiar los gutters de la cuadrícula. Los espacios de cuadrícula (gaps) se aplican horizontal y verticalmente de forma predeterminada. Consulta la sección de personalización para obtener más detalles.
- Los estilos en línea y personalizados deben verse como reemplazos de las clases de modificadores (por ejemplo, `style="--bs-columns: 3;"` vs `class="row-cols-3"`).
- El anidamiento funciona de manera similar, pero puede requerir que restablezcas los recuentos de columnas en cada instancia de un `.grid` anidado. Consulta la sección de anidamiento para obtener más información.

Ejemplos

Tres columnas

Se pueden crear tres columnas de igual ancho en todas los viewports y dispositivos usando las clases `.g-col-4`. Agrega clases responsive para cambiar el diseño según el tamaño del viewport.

```
<div class="grid">
  <div class="g-col-4">.g-col-4</div>
  <div class="g-col-4">.g-col-4</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```

Responsive

Usa clases responsive para ajustar tu diseño en los viewports. Aquí comenzamos con dos columnas en los viewports más estrechos y luego crecemos a tres columnas en los viewports medianos y superiores.

```
<div class="grid">
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
  <div class="g-col-6 g-col-md-4">.g-col-6 .g-col-md-4</div>
</div>
```

Compara eso con este diseño de dos columnas en todas los viewports.

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Wrapping

Los elementos de la cuadrícula se ajustan automáticamente a la siguiente línea cuando no hay más espacio horizontalmente. Ten en cuenta que el `gap` se aplica a los espacios horizontales y verticales entre los elementos de la cuadrícula.

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Starts

Las clases *start* tienen como objetivo reemplazar las clases de desplazamiento (offset) de nuestra cuadrícula predeterminada, pero no son del todo iguales. CSS Grid crea una plantilla de cuadrícula a través de estilos que le indican a los navegadores que "comiencen en esta columna" y "finalicen en esta columna". Esas propiedades son `grid-column-start` y `grid-column-end`. Las clases de inicio son una forma abreviada de lo primero. Combínalos con las clases de columnas para dimensionar y alinear tus columnas como lo necesites. Las clases de inicio comienzan en `1` ya que `0` es un valor no válido para estas propiedades.

```
<div class="grid">
  <div class="g-col-3 g-start-2">.g-col-3 .g-start-2</div>
  <div class="g-col-4 g-start-6">.g-col-4 .g-start-6</div>
</div>
```

Auto columnas

Cuando no hay clases en los elementos de la cuadrícula (los elementos hijos inmediatos de un `.grid`), cada elemento de la cuadrícula se dimensionará automáticamente en una columna.

```
<div class="grid">
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
</div>
```

Este comportamiento se puede combinar con clases de columna de cuadrícula.

```
<div class="grid">
  <div class="g-col-6">.g-col-6</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
  <div>1</div>
</div>
```

Anidamiento

Similar a nuestro sistema de cuadrícula predeterminado, nuestra CSS Grid permite anidar fácilmente `.grid`s. Sin embargo, a diferencia de la predeterminada, esta cuadrícula hereda los cambios en las filas, columnas y espacios. Considera el siguiente ejemplo:

- Sobrescribimos el número predeterminado de columnas con una variable CSS local: `--bs-columns: 3`.

- En la primera columna automática, el recuento de columnas se hereda y cada columna es un tercio del ancho disponible.
- En la segunda columna automática, hemos restablecido el recuento de columnas en el `.grid` anidado a 12 (nuestro valor predeterminado).
- La tercera columna automática no tiene contenido anidado.

En la práctica, esto permite diseños más complejos y personalizados en comparación con nuestro sistema de cuadrícula predeterminado.

```
<div class="grid" style="--bs-columns: 3;">
  <div>
    Primera columna automática
    <div class="grid">
      <div>Columna automática</div>
      <div>Columna automática</div>
    </div>
  </div>
  <div>
    Segunda columna automática
    <div class="grid" style="--bs-columns: 12;">
      <div class="g-col-6">6 de 12</div>
      <div class="g-col-4">4 de 12</div>
      <div class="g-col-2">2 de 12</div>
    </div>
  </div>
  <div>Tercera columna automática</div>
</div>
```

Personalización

Personaliza el número de columnas, el número de filas y el ancho de los espacios con variables CSS locales.

Variable	Valor Fallback	Descripción
`--bs-rows`	1	El número de filas en tu plantilla de cuadrícula
`--bs-columns`	12	El número de columnas en tu plantilla de cuadrícula
`--bs-gap`	`1.5rem`	El tamaño del espacio entre columnas (vertical y horizontal)

Estas variables CSS no tienen un valor predeterminado; en su lugar, aplican valores alternativos (fallback) que se usan *hasta* que se proporcione una instancia local. Por ejemplo, usamos `var(--bs-rows, 1)` para nuestras filas de CSS Grid, que ignora `--bs-rows` porque aún no se ha configurado en ninguna parte. Una vez que lo esté, la instancia `.grid` usará ese valor en lugar del valor alternativo de `1`.

Sin clases de cuadrícula

Los elementos hijos inmediatos de `.grid` son elementos de cuadrícula, por lo que se dimensionarán sin agregar explícitamente una clase `.g-col`.

```
<div class="grid" style="--bs-columns: 3;">
  <div>Columna automática</div>
  <div>Columna automática</div>
```

```
  <div>Columna automática</div>
</div>
```

Columnas y espacios (gaps)

Ajusta el número de columnas y el espacio.

```
<div class="grid" style="--bs-columns: 4; --bs-gap: 5rem;">
  <div class="g-col-2">.g-col-2</div>
  <div class="g-col-2">.g-col-2</div>
</div>
<div class="grid" style="--bs-columns: 10; --bs-gap: 1rem;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```

cambiar la ubicación de las columnas:

```
<div class="grid" style="--bs-rows: 3; --bs-columns: 3;">
  <div>Columna automática</div>
  <div class="g-start-2" style="grid-row: 2">Columna automática</div>
  <div class="g-start-3" style="grid-row: 3">Columna automática</div>
</div>
```

Gaps

Cambia los espacios verticales solo modificando `row-gap`. Ten en cuenta que usamos `gap` en `.grids`, pero `row-gap` y `column-gap` se pueden modificar según sea necesario.

```
<div class="grid" style="row-gap: 0;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Por eso, puedes tener diferentes `gaps` verticales y horizontales, que pueden tomar un solo valor (todos los lados) o un par de valores (vertical y horizontal). Esto se puede aplicar con un estilo en línea para `gap`, o con nuestra variable CSS `--bs-gap`.

```
<div class="grid" style="--bs-gap: .25rem 1rem;">
  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>

  <div class="g-col-6">.g-col-6</div>
  <div class="g-col-6">.g-col-6</div>
</div>
```

Sass

Una limitación de CSS Grid es que nuestras clases predeterminadas aún son generadas por dos variables Sass, `$grid-columns` y `$grid-gutter-width`. Esto predetermina efectivamente el número de clases generadas en nuestro CSS compilado. Tienes dos opciones aquí:

- Modifica esas variables Sass predeterminadas y vuelve a compilar tu CSS.
- Usa estilos en línea o personalizados para aumentar las clases proporcionadas.

Por ejemplo, puedes aumentar el número de columnas y cambiar el tamaño del espacio, y luego dimensionar sus “columnas” con una combinación de estilos en línea y clases de columna de CSS Grid predefinidas (por ejemplo, `.g-col-4`).

```
<div class="grid" style="--bs-columns: 18; --bs-gap: .5rem;">
  <div style="grid-column: span 14;">14 columnas</div>
  <div class="g-col-4">.g-col-4</div>
</div>
```