Bootstrap

¿Qué es Bootstrap?

Bootstrap es un framework[1] (marco de trabajo), de código libre, para desarrollar con HTML, CSS y JavaScript. Es un conjunto de herramientas para el diseño de sitios y aplicaciones web, y a diferencia de otros frameworks web, solo se encarga del desarrollo front-end[2].

Originalmente, fue llamado Blueprint, y desarrollado por Mark Otto y Jacob Thornton de Twitter, como un marco de trabajo para mantener la consistencia entre las herramientas internas que utilizaban, ya que el uso de las mismas generó incontables inconsistencias y una gran carga de trabajo en mantenimiento.

Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo LESS que implementan la variedad de componentes de la herramienta. En su última versión, Bootstrap 4, es compatible con la mayoría de los navegadores modernos: Chrome, Firefox, Internet Explorer 10+, Edge, Safari y Opera. Bootstrap 3 es la última versión estable compatible con Internet Explorer 8 e Internet Explorer 9.

¿Por qué usar Bootstrap?

1. **Fácil:** no se necesitan demasiados conocimientos de CSS, ni mucho tiempo para empezar a utilizar los componentes y utilidades que provee este framework.
2. **Rápido:** ahorra el tiempo y esfuerzo que habría que invertir si se quisieran crear ciertos estilos desde cero. Solo deben agregarse los archivos de Bootstrap al proyecto, y comenzar a usarlos. La documentación es muy buena y completa, por lo cual el aprendizaje es relativamente sencillo.
3. **Vistoso:** con tan solo un poco de contenido básico de Bootstrap, pueden crearse sitios que luzcan bien.
4. **Personalizable:** el código fuente de Bootstrap puede descargarse del sitio, y cambiarse para que se adapte a las necesidades del proyecto o gustos personales.

[1] Un framework o marco de trabajo, es un conjunto de conceptos, herramientas y tecnologías que facilitan tareas comunes del desarrollo de aplicaciones, ahorrando tiempo y esfuerzo.
[2] Front-end, hace referencia al desarrollo de las interfaces de usuario, para que los mismos puedan interactuar con el sistema.

Ventajas y Desventajas

Como todo framework, Bootstrap, presenta una serie de ventajas y desventajas, que deberían evaluarse para cada proyecto en el que se planee incorporarlo.

Ventajas

1. Previene la repetición de código y trabajo entre proyectos, porque pueden reusarse los componentes dentro del framework.
2. Ayuda a ahorrar tiempo, no solo en el desarrollo de componentes reusables, sino en la documentación de los mismos. Bootstrap posee una extensa documentación que puede consultarse en línea[3], y una gran comunidad que le da soporte.
3. Utiliza el diseño adaptable (responsive) para permitir que los sitio web se ajusten a distintos tamaños de pantalla, como móvil, escritorio, etc.
4. Añade consistencia al diseño y código entre proyectos, y entre distintos desarrolladores.
5. Asegura la compatibilidad entre distintos navegadores. Disminuye la cantidad de problemas (bugs), que surgen al intentar cubrir los mismos.
6. Cuenta con un excelente sistema de grilla. El mismo, es uno de los aspectos esenciales del framework; es la base sobre la cual toda la disposición (layout) es creada.
7. El núcleo CSS de Bootstrap añade estilos a formularios, tablas, botones, listas e imágenes, como también barras de navegación completamente funcionales; mientras que el núcleo de JavaScript añade código que ayuda a la creación de modales, carruseles, alertas, pop ups, desplegables y acordeones.
8. Es ligero y personalizable.
9. Contiene estructuras y estilos adaptables.
10. Pueden encontrarse una gran cantidad de plantillas y temas profesionales basados en Bootstrap.

Desventajas

1. Ya que los frameworks tienen un conjunto de componentes y códigos estandarizados, restringen el diseño.

[3] Documentación de Bootstrap (Inglés)

2. Los frameworks limitan la creatividad, ya que hay que adaptarse a los requerimientos de los mismos.

3. Puede terminar dedicándose demasiado tiempo a sobre escribir estilos y archivos, para crear un diseño que se aparte del presentado por Bootstrap.

4. Se necesita un gran nivel de personalización del contenido de Bootstrap o todos los sitios terminan luciendo de la misma manera.

5. Puede llegar a ser bastante pesado, ya que debe incluirse el framework dentro del proyecto, traduciéndose en mayores tiempos de carga para el usuario.

Guía de Instalación

Desde el sitio de Bootstrap[4], se pueden descargar los archivos CSS y JS compilados, el código fuente o copiarse el CDN[5]. Además, se presentan instrucciones para realizar la instalación con distintos gestores de paquetes, como *npm*, *Composer*, etc.

1. Archivos CSS y JavaScript compilados: simplemente se descargan, y se incluyen en los HTML del proyecto.

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" type="text/css" href="bootstrap.min.css" />
  </head>
  <body>
    <script src="bootstrap.bundle.min.js"></script>
  </body>
</html>
```

2. Código fuente: se pueden descargar los archivos SASS y JavaScript, y compilarlos con un proceso propio. Esto puede ser de utilidad si se desea modificar o personalizar Bootstrap en gran medida.

3. CDN: sólo deben incluirse los enlaces al CDN que son provistos en el sitio.

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
  </head>
  <body>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
  </body>
</html>
```

[4] Bootstrap Descargas

[5] CDN (Content Delivery Network) es un conjunto de ubicaciones, que se encarga de redistribuir localmente el contenido de los servidores y guardan en caché los archivos que no necesitan actualización permanente. De esta manera, librerías, estilos, fuentes, etc., se cargan en el sitio con una mayor velocidad y rendimiento.

Conceptos Importantes

Diseño Responsive

El diseño web responsive (adaptable), es un concepto de diseño relativamente nuevo, con el objetivo de hacer que una página web se vea bien en todos los dispositivos y distintos tamaños de pantallas; ya sea escritorio, tablet, smartphone, etc.

En pocas palabras, se trata de crear un solo diseño que funcione en todos los dispositivos, en vez de tener que crear un diseño distinto para cada uno. Cuando se logra utilizar todo el poder de HTML5 y CCS3 para achicar, redimensionar, mover, ocultar o agrandar el contenido en un sitio web para que se vea presentable en cualquier tamaño de pantalla, es llamado Diseño Web Responsive.

El diseño responsive, tiene tres características principales:

1. Diseño basado en grillas
2. Imágenes y media flexibles
3. Media queries de CSS

Diseño basado en grillas

El diseño basado en grillas, organiza los elementos de un sitio web de manera que sea fácil de leer para el usuario, y que se visualicen correctamente en cualquier tipo de dispositivo o tamaño de pantalla.

Este diseño, adopta unidades relativas como porcentajes o unidades EM, para definir el tamaño de los elemento en pantalla, en vez de usar unidad absolutas como píxeles o puntos. Una unidad EM, es igual al valor calculado de la propiedad “font-size” para el elemento al cual se está aplicando.

Imágenes y media flexibles

En el diseño web responsive, las imágenes también reciben un tamaño en unidades relativas y son colocadas en una grilla con un ancho (width) máximo de 100%. La propiedad “width”, es usada para especificar el ancho de área de contenido de un dispositivo en CSS. Esto evita que se muestren fuera del elemento en el que fueron colocadas. Por ende, el tamaño de una imagen nunca puede ser mayor al tamaño de la grilla si se utiliza diseño responsive.

Media Queries de CSS

Esta técnica usa las reglas @media de CSS, para definir diferentes reglas de estilo para distintos tamaños y dispositivos. La regla @media, es un condicional de CSS que permite

aplicar distintos estilos dependiendo del tipo general del dispositivo, características específicas o el entorno. Usando las Media Queries, puede especificarse cuándo y dónde deben aplicarse ciertas reglas de CSS, basándose en las características del dispositivo; generalmente, el ancho del navegador.

Mobile First

Además de ser completamente adaptable, Bootstrap utiliza un concepto denominado “Mobile First” (primero móvil), para ayudar en el desarrollo de sitios web responsive y sencillos de usar en dispositivos móviles. La idea de este enfoque, es que los desarrolladores se concentren en construir primero el sitio para tamaños pequeños de pantalla y luego escalen el diseño para pantallas más grandes.

También conocido como “Mejora Progresiva”, es un enfoque de diseño que otorga la habilidad de entender y alcanzar los requerimientos de los usuarios móviles.

Sistema de Grilla

Bootstrap, está equipado con un sistema de grilla de 12 columnas, totalmente responsive, basado en flexbox, que se escala automáticamente dependiendo de la resolución de la pantalla del dispositivo.

Se pueden usar las 12 columnas individualmente o agruparlas para crear diseños responsive de manera sencilla. Además, con las utilidades responsive, se pueden esconder o mostrar ciertas secciones de contenido en un tamaño de pantalla específico.

Algunos ejemplos:

1. Especificando el breakpoint *sm*: se crean las columnas a partir del breakpoint *sm* (dispositivos pequeños), lo que hace que se distribuyan de la misma manera en dispositivos de mayor tamaño.

Una de tres columnas	Una de tres columnas	Una de tres columnas

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      Una de tres columnas
    </div>
    <div class="col-sm">
      Una de tres columnas
    </div>
    <div class="col-sm">
      Una de tres columnas
    </div>
  </div>
</div>
```

2. Sin especificar tamaño: si no se especifican los tamaños de las columnas, Bootstrap calcula la cantidad según las clases que se hayan definido dentro de la fila.

1 de 2	2 de 2

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

3. Especificando tamaño y breakpoint: pueden crearse diferentes configuraciones según los tamaños de los dispositivos. En la primer tabla, se ve la configuración para un dispositivo *md* o superior, mientras en la segunda para un dispositivo *sm*.

1 de 4	2 de 4	3 de 4	4 de 4

1 de 4
2 de 4
3 de 4
4 de 4

```
<div class="container">
  <div class="row">
    <div class="col-md-3 col-sm-12">
      1 de 4
    </div>
    <div class="col-md-3 col-sm-12">
      2 de 4
    </div>
    <div class="col-md-3 col-sm-12">
      3 de 4
    </div>
    <div class="col-md-3 col-sm-12">
      4 de 4
    </div>
  </div>
</div>
```

Flexbox

El diseño flexbox, apunta a proveer una manera más eficiente de mostrar el contenido, alinearlo y distribuir el espacio entre los elementos de un container, incluso cuando su tamaño es desconocido y/o dinámico (de ahí la palabra “flex”).

La idea principal detrás de flexbox, es darle al container la habilidad de alterar el ancho y alto de sus elementos para que ocupen el espacio disponible de la mejor manera posible. Un container flex, expande sus elementos para llenar todo el espacio libre posible, o los achica para prevenir overflow (desborde de los elementos fuera del container).

Ya que flexbox es un módulo, contiene un conjunto completo de propiedades, algunas de las cuales deben agregarse al container, y otras, a los elementos que se encuentran en el mismo. Algunas de las más importantes:

1. En el componente padre:
    - *display*: esto define el tipo de componente; de ser “flex”, crea el contexto de flexbox para todos los hijos del componente.

```
.container {
  display: flex;
}
```

    - *flex-direction*: define cuál será el eje principal del componente, marcando así la dirección del contenido.

```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

    - *flex-wrap*: por defecto, los elementos flex van a intentar acomodarse en una sola línea, lo que puede cambiarse con esta propiedad.

```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

    - *justify-content*: define la alineación a lo largo del eje principal.

```
.container {
  justify-content: flex-start | flex-end | center | space-between |
space-around | space-evenly;
}
```

    - *align-items*: define la alineación a lo largo del eje secundario.

```
.container {
  align-items: flex-start | flex-end | center | baseline | stretch;
}
```

2. En los componentes hijo:
   - *order*: por defecto, los elementos se acomodan tal cual se definen en el código fuente.

```
.item {
  order: <integer>; /* default is 0 */
}
```

   - *flex-grow*: define la habilidad de un elemento flex de crecer si es necesario. Acepta un valor sin unidades que representa una proporción.

```
.item {
  flex-grow: <number>; /* default 0 */
}
```

   - *flex-shrink*: define la habilidad de un elemento flex de achicarse si es necesario.

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

   - *flex*: atajo para combinar *flex-grow*, *flex-shrink*, y una tercera propiedad *flex-basis*.

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

   - *align-self*: permite sobreescribir la alineación para los elementos en forma individual.

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

Visual Breakpoints

Bootstrap divide los diferentes dispositivos en cinco clases predefinidas de grilla, basándose en el ancho de la ventana: xs, sm, md, lg, y xl, desde dispositivos móviles (xs), hasta smartphones (sm), tablets (md), escritorio/notebooks (lg) y escritorio (xl).

Esto elimina la necesidad de definir nuestros propios breakpoints, ya que la mayoría de los dispositivos disponibles hoy, caen bajo uno de estos breakpoints.

Usando una combinación de estas clases, se puede crear diseños responsive dinámicos, flexibles y complejos, fácilmente, y acorde a las necesidades del proyecto.

Configuraciones Globales

HTML5 doctype

Bootstrap requiere el uso del HTML doctype. Sin el mismo, pueden aparecer algunos estilos incompletos.

```
<!DOCTYPE html>
<html lang="en">
  ...
</html>
```

Etiqueta <meta> para responsive

Bootstrap se desarrolla con el concepto de “móvil primero”, estrategia en la que se optimiza el código para dispositivos móviles en primera medida, y luego se escalan los componentes como sea necesario. Para asegurar un renderizado adecuado en todos los dispositivos, debería añadirse la siguiente etiqueta <meta> al <head>:

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

Ejemplos

Navbar

Con Bootstrap, es posible crear barras de navegación totalmente responsive, de manera sencilla. De todas maneras, deben tenerse en cuenta las siguientes consideraciones:

- Las navbars deben envolverse con las clases .navbar y .navbar-expand{-sm|-md|-lg|-xl} para responsive.
- Las navbars y su contenido son fluidos por defecto.
- Hay que usar las clases de spacing y flex para controlar el espacio y alineación dentro de las navbars.
- Se asegura la accesibilidad, usando el elemento <nav>; o si se usa un elemento más genérico como , añadiendo el atributo role=”navigation” a todas las navbars, para que las tecnologías de asistencia puedan identificarlas.

A continuación se presenta un ejemplo de una navbar sencilla, con tres enlaces y un desplegable:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false"
aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavDropdown">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Features</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Pricing</a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Dropdown link
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li>
```

```
    </ul>
  </div>
</nav>
```

Breadcrumb

Indica la ubicación de la página actual dentro de la jerarquía de navegación. En cuanto a la accesibilidad, conviene agregar un aria-label significativo.

```
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Home</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Library</li>
  </ol>
</nav>

<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item"><a href="#">Library</a></li>
    <li class="breadcrumb-item active" aria-current="page">Data</li>
  </ol>
</nav>
```

Forms

Un formulario simple, parecido a un login:

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
aria-describedby="emailHelp" placeholder="Enter email">
    <small id="emailHelp" class="form-text text-muted">We'll never share your email with
anyone else.</small>
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div class="form-group form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Layout

Contiene componentes y opciones para distribuir y presentar el contenido en pantalla.

- *Containers*: elemento más básico del layout de Bootstrap. Son requeridos para utilizar el sistema de grillas.

```
<div class="container">
  <!-- Content here -->
</div>
```

- *Responsive Breakpoints*:

```
// Dispositivos “Extra small” (teléfonos en modo portrait, con menos de 576px)
// No hay media query para `xs` ya que es el valor por defecto en Bootstrap

// Dispositivos “Small” (teléfonos en modo landscape, 576px y más)
@media (min-width: 576px) { ... }

// Dispositivos “Medium” (tablets, 768px y más)
@media (min-width: 768px) { ... }

// Dispositivos “Large” (escritorio, 992px y más)
@media (min-width: 992px) { ... }

// Dispositivos “Extra large” (escritorio grandes, 1200px y más)
@media (min-width: 1200px) { ... }
```

Buttons

Usar color para demostrar significado, solo provee una indicación visual, lo que no es de ayuda para usuarios con tecnologías de asistencia, como lectores de pantalla. Hay que asegurar que la información otorgada por los colores, sea obvia por el contenido del botón, o que sea incluída por medios alternativos, tales como texto adicional escondidos con la clase .sr-only.

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```

Guía de Ejercicios

Todos los ejercicios serán aplicados sobre un archivo HTML adjunto, sin estilo.

1. Incluir Bootstrap en el HTML
2. Crear una navbar con Bootstrap
3. Agregar un carousel
4. Incluir Cards en la página
5. Implementar el sistema de grilla
6. Estilar un formulario
7. Detalles y responsive