Bootstrap

Getting started with Bootstrap

Bootstrap is a *framework* of readily available code that integrates with HTML to create stylized websites that adapt the layout to users' screen sizes. This framework allows us to cut down on the time needed to style a website, simplifies the complexity of how to layout elements, works across multiple browsers, and reduces the frustration of using plain CSS. All it takes to use Bootstrap is a few additional lines in our HTML document.

In this lesson, we'll be working with Bootstrap 4. To incorporate Bootstrap into a project, we have to include two `<meta>` tags and the Bootstrap CSS library.

Intro to the Grid

Bootstrap simplifies the layout of a website using a grid system. For us to fully take advantage of Bootstrap's grid system, we need to understand how to structure our HTML.

At the heart of it, *containers* are the basis of Bootstrap's grid. Inside containers, we nest *rows* as immediate children. Then, nested inside rows are *columns*. Inside columns, we put our actual content. Take a look below at an example of this nesting pattern. Don't worry about the syntax of rows and columns just yet, but do take note of how our HTML is organized:

Bootstrap Containers

Bootstrap uses a grid system based on CSS Flexbox which organizes content in rows or columns.

Using the Bootstrap grid also allows for *responsive design*, in other words, a website's layout can change based on the user's screen size.

Bootstrap uses classes to apply CSS rulesets — these rulesets dictate the layout and styling of the element. To create a container, necessary for Bootstrap's grid, we assign a class of `"container"` to a `<div>` like so:

```
<div class="container"></div>
```

The `<div>` from the example above becomes a Bootstrap container which has a width relative to a user's screen size, becomes horizontally centered, and has a left and right margin.

Knowing how to use documentation is important. We can always check what classes to add using Bootstrap's grid documentation.

Rows

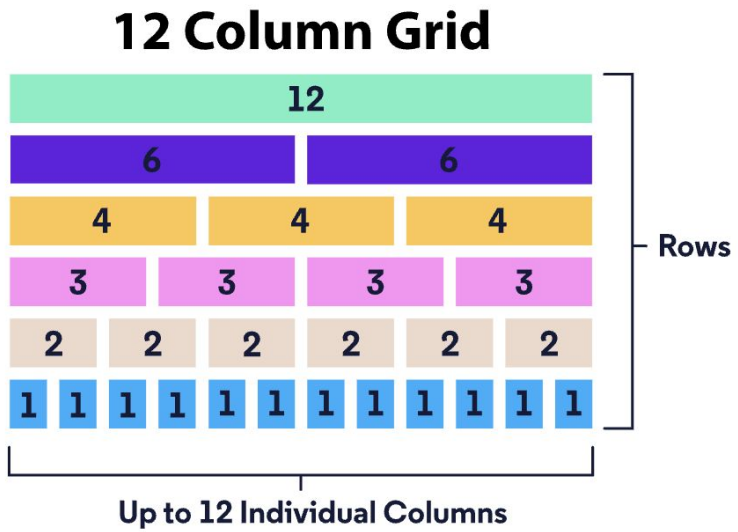
Rows are nested within containers. Also, we can add as many rows as we want inside a container. By default, a row will take up the entire width of its parent container. To create a row we assign a `class` of `"row"` to an element.

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

Columns

Columns are the immediate children of rows and store content. By default, a column will take up the entire width of its parent row. As we add more columns inside a row, the default behavior is for each column's width to be readjusted to fit in the row — each column will have the same width. At most, a row will accommodate 12 columns. Study the diagram to see how column sizing works.

To create a column, we assign an element with the `class` of `"col"`.



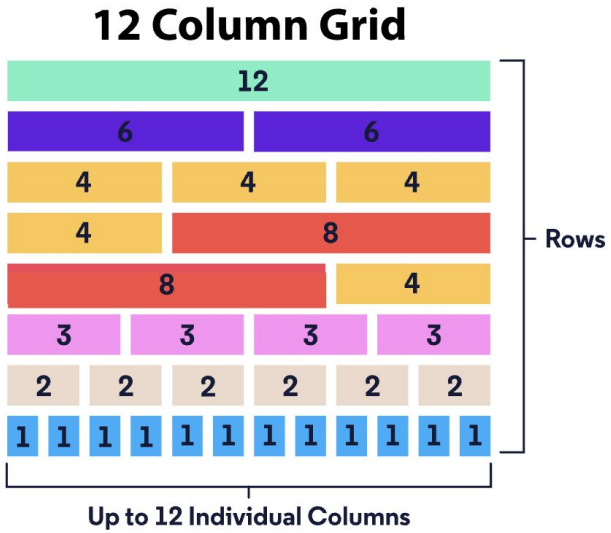

```
<div class="container">
  <div class="row">
    <div class="col">
    </div>
  </div>
</div>
```

Setting Columns Widths

Bootstrap gives us more customization options so that we can make columns of varying widths. Take a look at 4th and 5th row of the diagram for examples of rows containing columns of differing widths.

We can decide the width of a column by appending a hyphen and a number to the `"col"` class like so.

```
<div class="col-8">
  <p>This is the width of 8 columns.</p>
</div>
```



Setting Column Width with Content

Another option we could use to set the width of a column is the content inside the column.

If we want to use the content to set a column's width, we append `"auto"` to the `class` of the column, for example:

```
<div class="col-auto">
  <p>This content determines the width of the
parent column</p>
</div>
```

Bootstrap Breakpoints

One benefit of using Bootstrap is that it incorporates *responsive design*. With responsive design, the layout of the content changes to accommodate a user's screen size.

Bootstrap categorizes screen sizes into 5 categories or as *breakpoints*: extra small, small, medium, large, and extra large. Each breakpoint has a range measured in pixels. The range of these breakpoints and accompanying device types are marked in the following table:

Category	Breakpoint Range	Device Type
Extra small	< 576 px	portrait smartphones
Small	≥ 576 px	landscape smartphones
Medium	≥ 768 px	tablets
Large	≥ 992 px	desktops
Extra Large	≥ 1200 px	large desktop

Breakpoint Naming Convention

Using Bootstrap, we can freely change the layout of our content using grid. We can even customize how our content on the grid changes based on breakpoints (extra small, small, medium, large, extra large). To incorporate these breakpoints into our code, we have to follow Bootstrap's class naming convention.

The naming convention follows a pattern of `"col-{breakpoint}-{width}"`. Let's break this pattern down:

- As seen before `"col"` refers to a column.
- `{breakpoint}` can be `sm`, `md`, `lg`, or `xl`. Notice that there is no extra small or `xs` breakpoint. If we omit `{breakpoint}`, it is by default for extra small screens.
- `{width}` can be set from `1` to `12` and assigns a width to the column.
- When we set a `{breakpoint}-{width}`, it means that we want our column to have that set width for screens that fit in the specified breakpoint range and other larger screens.

Combining Bootstrap Classes

We can add multiple classes to our columns for additional control over the rendering of our content.

```
<div class="col-12 col-md-8 col-xl-6">
</div>
```

Activity