

## Easy to set up

There are multiple ways to include Bootstrap in a project. The easiest way to import Bootstrap is using a Content Delivery Network (CDN). There's a great introduction in the official documentation on how to import using a BootstrapCDN [here](#).

You can also Install Bootstrap as project dependency using **npm**, if you're using a JavaScript library like React or a framework like Angular.

```
npm install --save bootstrap
```

Once you have installed Bootstrap, you can proceed with importing the CSS file.

- For React, include Bootstrap in your app entry file, usually **index.js**.
- For Angular 2+, include Bootstrap in the configuration file. This should be your **angular.json** file.

Also include jQuery, Popper and the Bootstrap minified bundle, as they are required for certain components to function properly.

### React example

Import Bootstrap in your **index.js** file.

```
import React from 'react';  
  
import ReactDOM from 'react-dom';  
  
import './index.css';  
  
import App from './App';  
  
import * as serviceWorker from './serviceWorker';  
  
import 'bootstrap/dist/css/bootstrap.min.css';  
  
import $ from 'jquery';  
  
import Popper from 'popper.js';  
  
import 'bootstrap/dist/js/bootstrap.min.js';
```

## Angular 2+ example

Import Bootstrap in your **angular.json** file.

```
"build": {
  "builder": "@angular-devkit/build-angular:browser",
  "options": {
    "outputPath": "dist/output",
    "index": "src/index.html",
    "main": "src/main.ts",
    "polyfills": "src/polyfills.ts",
    "tsConfig": "src/tsconfig.app.json",
    "assets": [
      "src/favicon.ico",
      "src/assets"
    ],
    "styles": [
      "node_modules/bootstrap/dist/css/bootstrap.min.css"
    ],
    "scripts": [
      "node_modules/jquery/dist/jquery.min.js",
      "node_modules/popper.js/dist/umd/popper.min.js",
      "node_modules/bootstrap/dist/js/bootstrap.min.js"
    ]
  }
}
```

Alternatively, you can download Bootstrap source code [here](#) and use it locally.

## Responsiveness

The responsive feature Bootstrap is so famous for is the **12-Column Grid System** mentioned above. The grid is divided into 3 important set of classes: the **container**, the **row** and the **column**.

- **Container** is the wrapper for the rows and columns.
- **Row** houses the columns.
- **Columns** contain the main content and must be inside rows.

To build a container you can use the **container** class or **container-fluid** class. The main difference between these two **responsive** containers is that the **container** class specifies a fixed-width box for each device screen or viewport size, while **container-fluid** class specifies a container that will always fill all the available width.

Bearing this idea in mind, we can start by creating our container. In this example, we have a fixed-width container with 2 rows inside. The rows will be our wrappers for the columns. You can create has many rows as you need with different column sizes.

```
<div class="container">

  <div class="row">. . .</div>

  <div class="row">. . .</div>

</div>
```

After creating a row, you can use the 12-column system provided by Bootstrap to place flexible columns of different sizes inside the rows you've created. The columns will host the main content and must be inside the rows. The nomenclature to build columns is as follows:

**col-<breakpoint>-<size-of-column>**

Breakpoint applies a size to the column based on the device screen or viewport size and are used in following order:

**xs > sm > md > lg > xl**

- **xs** – for extra small devices, like portrait phones (< 576px); **(No longer supported in Bootstrap 4, replaced with col-<size-of-column>)**
- **sm** – for small devices, like landscape phones (>=576px);
- **md** – for medium-size devices, like tablets (>=768px);
- **lg** – for large devices, like laptops (>=992px);
- **xl** – for extra-large devices, like large desktops (>=1200px);

It's important to note that Bootstrap is a collection of JavaScript and CSS files, so, if you ever need to add new a breakpoint or modify the current ones, you can easily do this by overriding the original files. An example will be shown at the end of this topic.

As a row can contain up to 12 columns, column size goes from **1 to 12**. The column size is represented in percentages and always relative to its parent, which is the row itself. Following the example above, we can start using some columns in our rows.

```
<div class="container">

  <div class="row">

    <div class="col-sm-4" style="background: lightgreen">. . .</div>

    <div class="col-sm-8" style="background: aqua">. . .</div>

  </div>

  <div class="row">

    <div class="col-lg-4" style="background: lime">. . .</div>

    <div class="col-lg-4" style="background: lightgray">. . .</div>

    <div class="col-lg-4" style="background: lightblue">. . .</div>

  </div>

</div>
```

This will generate the following output:



In this example the first row has 2 columns that have a ratio of 1:2. The first column will have a width of 33% and the second column will have a width of 66%, as columns widths are set in percentages and relative to the row width.

These columns will apply the style to small devices (**col-sm-\***) and maintain its proportions on medium-size devices (**col-md-\***) right up to extra-large (**col-xl-\***) devices. This happens because **sm** breakpoint is based on a media query that applies the style to devices having a screen size larger than or equal to 576px if, at least, the **col-md-\*** class is not present.

The second row has 3 columns of the same size, each column taking up to 1/3 of the container width. These columns will maintain their proportions on large devices (**col-lg-\***) right up to extra-large (**col-xl-\***) devices. As we do not specify a class for small devices (**col-sm-\***) and medium-size devices (**col-md-\***), the second row will have the 3 columns separated in new lines. This is great for devices with small screens, as the user experience will be pleasanter. The second row can also be written like this:

```
<div class="row">
  <div class="col-lg" style="background: lime">. . .</div>
  <div class="col-lg" style="background: lightgray">. . .</div>
  <div class="col-lg" style="background: lightblue">. . .</div>
</div>
```

If we do not specify the column size, Bootstrap will assume a row with equal-width columns. Note that the sum of the columns should be equal to 12 if you want to keep the columns in one line and side by side. In first row, there are two columns with 4 + 8 and in second row, we have 4 + 4 + 4, which both add up to 12.

You can, of course, create more flexible rows, which change the column orientation according to the device screen. The following example contains 5 components retrieved from an internal project developed in the company. These components are aligned horizontally for bigger resolutions and start to decompose into new lines accordingly.



Without Bootstrap, you would need to write something like this to achieve the result above.

```
<div class="main-content">

  <div class="item">. . .</div>

  <div class="item">. . .</div>

  <div class="item">. . .</div>

  <div class="item">. . .</div>

  <div class="item">. . .</div>

</div>
```

```
.main-content {
  position: relative;
  display: flex;
  max-width: 100%;
  padding: 15px;

  .item {
    position: relative;
    display: flex;
    flex-direction: column;
    margin: 15px;
    min-width: 150px;
    max-width: 300px;
    width: 100%;
    height: 400px;
  }
}
```

```

@media only screen and (min-width: 768px) {
  .main-content {
    flex-flow: row wrap;
    justify-content: center;
  }
}

@media only screen and (min-width: 1200px) {
  .main-content {
    flex-wrap: nowrap;
    justify-content: space-evenly;
  }
}

```

With Bootstrap, you only need to write the following.

```

<div class="container-fluid">
  <div class="row">
    <div class="col-sm-12 col-md-6 col-lg"> . . </div>
    <div class="col-sm-12 col-md-6 col-lg"> . . </div>
  </div>
</div>

```

We can break this last example into small parts:

- We define a fluid-width container with one row;
- The row has just 5 columns that change size and orientation according to the device screen.
- For small devices (**col-sm-\***), each column will be separated into a new line, as the size of the column for this breakpoint is 12, which corresponds to the whole width of the row.
- For medium devices (**col-md-\***), we can see that the sum of the sizes for this breakpoint is equal to 30. This implies a separation of the row into three lines, with each line containing up to two equal-width columns.
- For large devices (**col-lg-\***) to extra-large devices(**col-xl-\***), the size of the columns is omitted, meaning that Bootstrap will render a row with equal-width columns.

## Customise and add new breakpoints

If you want to add a new breakpoint or modify current ones, you must import the necessary bootstrap files into a main CSS file and override the default configurations. In the example below, we change the default **xl** breakpoint from 1200px to 1024px and add a new **xxl** breakpoint, which starts at 1440px.

**Note:** In this example, Bootstrap was installed as a local dependency.

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

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

$grid-breakpoints: (
  xl: 1024px,
  xxl: 1440px
);

$container-max-widths: (
  xl: 1024px,
  xxl: 1440px
);

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

## Summary

In this brief introduction to Bootstrap we've covered some of the most useful features this framework provides. To summarise, these are some of the major advantages of using this framework and also the reasons why it's so popular:

- It's **easy to use** and doesn't require a deep knowledge of CSS or JavaScript. With a quick search through the official documentation you can build the website you want, to look and focus more on the functionality itself;
- **It saves a lot of time and effort** spent writing CSS, thanks to its pre-styled classes with responsive features;
- **Provides a ton of pre-built components**, all of which are highly customisable and easy to implement;
- **It's supported across all popular browsers**, like Chrome, Firefox, Edge, Safari, etc;
- **It's open source** and completely free to use.