CHAPTER 1

Getting Started

Creating responsive websites and applications is becoming a hard requirement now that many people have started using mobile phones and tablets. Twitter Bootstrap is here to give you all the necessary tools, so that the work you create is equally well viewed on various display sizes.

This chapter starts with the basics of Twitter Bootstrap, mainly its grid system.

You will learn about the 12-column grid system, so that you can divide your page in multiple rows and columns. Figure 1-1 shows an example of the Twitter Bootstrap grid system.

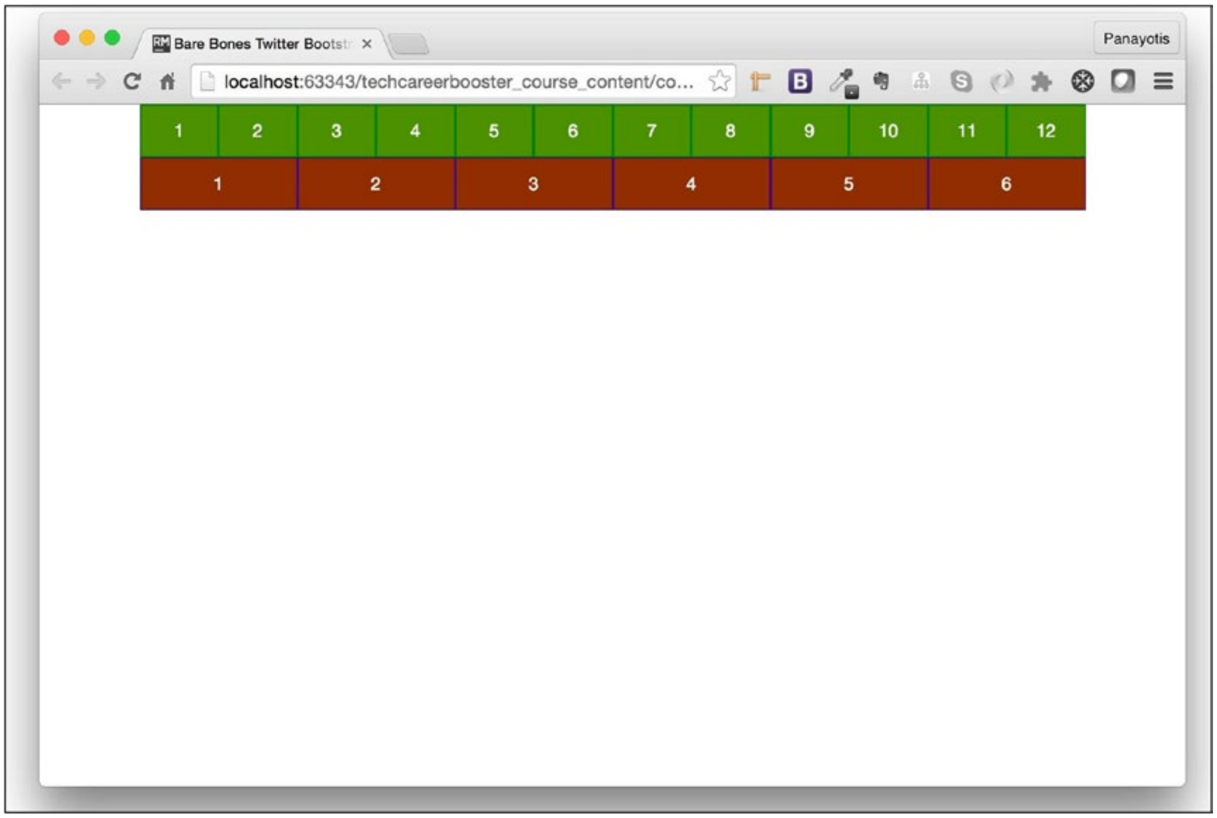

Bare Bones Twitter Bootstr
localhost:63343/techcareerbooster_course_content/co...
Panayotis
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4 5 6

Figure 1-1. A Demo Page That Shows the Twitter Bootstrap Grid System

© Panos Matsinopoulos 2020
P. Matsinopoulos, *Practical Bootstrap*, https://doi.org/10.1007/978-1-4842-6071-5_1

You will make sure that you have a page as shown in Figure 1-2.

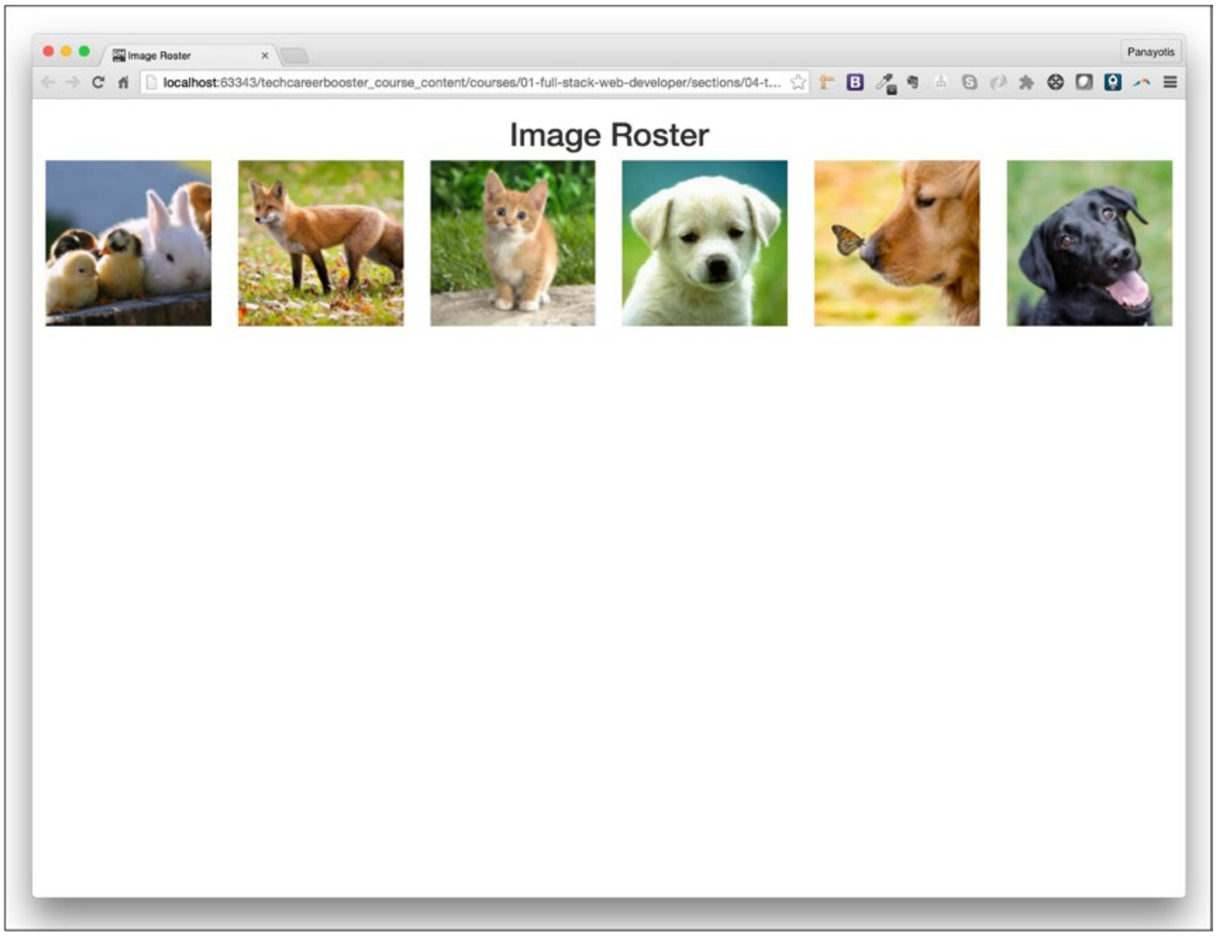


Image Roster
localhost:63343/techcareerbooster_course_content/courses/01-full-stack-web-developer/sections/04-t...
Panayotis
Image Roster

Figure 1-2. *Page as Viewed on Large Displays*

It can automatically be transformed to look like Figure 1-3 on smaller devices.

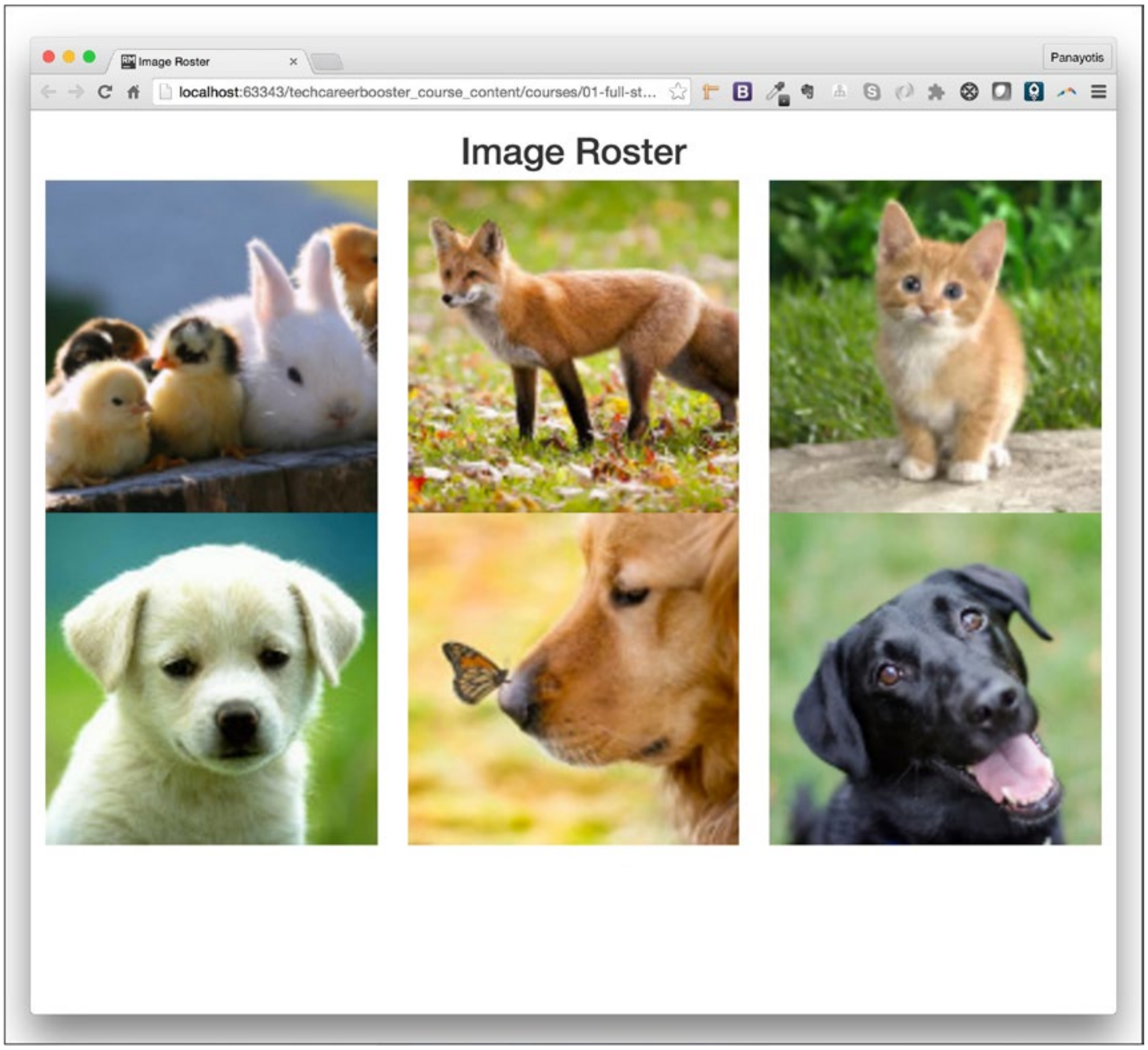


Image Roster
localhost:63343/techcareerbooster_course_content/courses/01-full-st...
Panayotis
Image Roster

Figure 1-3. *Page Adapts Its Layout on Smaller Devices*

Learning Goals

1. Learn what Twitter Bootstrap is.
2. Learn how to include Twitter Bootstrap CSS and JavaScript libraries in your project.
3. Learn about the two main containers available and their differences.
4. Learn about how the grid system of Twitter Bootstrap can help you organize the content of your page and generate a responsive layout.

Bootstrap Defined

Twitter Bootstrap is a set of CSS and JavaScript code that you can base your own HTML, CSS, and JavaScript code on, in order to quickly have a website or web application that is responsive, with a mobile-first design approach.

In other words, it provides you with some CSS classes and JavaScript code that you can just go and use. You apply the classes, and you use the JavaScript modules and quickly have your website and your web application have features that would have taken you a lot of effort to develop from scratch.

The Bare-Bones Twitter Bootstrap Page

Let's start by creating the first Twitter Bootstrap page. Create a new file `index.html` and put the following content inside (Listing 1-1).

Listing 1-1. Bare-Bones Twitter Bootstrap Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <title>Bare Bones Twitter Bootstrap Page</title>
</head>
<body>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

This is a page that does not have any valuable content. Its body element is empty. However

1. It references the Bootstrap CSS library.
2. It references the `jQuery` library, which is a JavaScript library necessary for Twitter Bootstrap JavaScript modules.
3. It references the `popper` library. This is a JavaScript library that Bootstrap is using to position tooltips and popovers.
4. Finally, it references the Bootstrap JavaScript library too.

If you load the preceding page on your browser, you will see nothing. But all the Twitter Bootstrap features would be loaded.

Containers

When you start your page layout, you need to decide on the container style that would include your page content. There are three types of containers:

1. `container`, which sets a `max-width` at each responsive breakpoint.
2. `container-fluid`, which is `width: 100%` at all breakpoints.
3. `container-{breakpoint}`, which is `width: 100%` until the specified breakpoint. After the specified breakpoint (including), it uses the `max-width` as if it were a `container`.

Note `max-width` defines the maximum width an element can have. This means that the element tries to occupy as much width as possible up to the `max-width` after which it expands its height in order to fit the content. This of course means that the actual width of the element cannot be more than `max-width`, even if the `width` property may have a larger value.

Tip Twitter Bootstrap defines five breakpoints, that is, pixel widths, after which the grid layout changes. For example, a small breakpoint has the value 576px and is referring to displays with width between 576px and 767px. I will talk about breakpoints later on in this chapter.

You apply one of them by setting the corresponding `div` element (or equivalent block element) to have a class equal to the name of the container.

Enhance the content of your page so that it is like the following (in bold, you see the code that has been added - Listing 1-2).

Listing 1-2. Add Two Containers

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>Bare Bones Twitter Bootstrap Page</title>
</head>
<body>
    <div class="container">
        This is a container
    </div>
    <div class="container-fluid">
        This is a container fluid
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

1. I have added a reference to a local CSS file named `stylesheets/index.css`. I have put that exactly after the Twitter Bootstrap CSS reference. I will put inside this file the web page–specific CSS rules. This file needs to be after the Twitter Bootstrap CSS reference, because it will be relying on it.

2. I have created two `div`s that would function as the page containers. The first `div` is a `container`, whereas the second `div` is a `container-fluid`.

   Create the file `index.css` inside the folder `stylesheets` and add the following content (Listing 1-3).

Listing 1-3. CSS for the index.css File

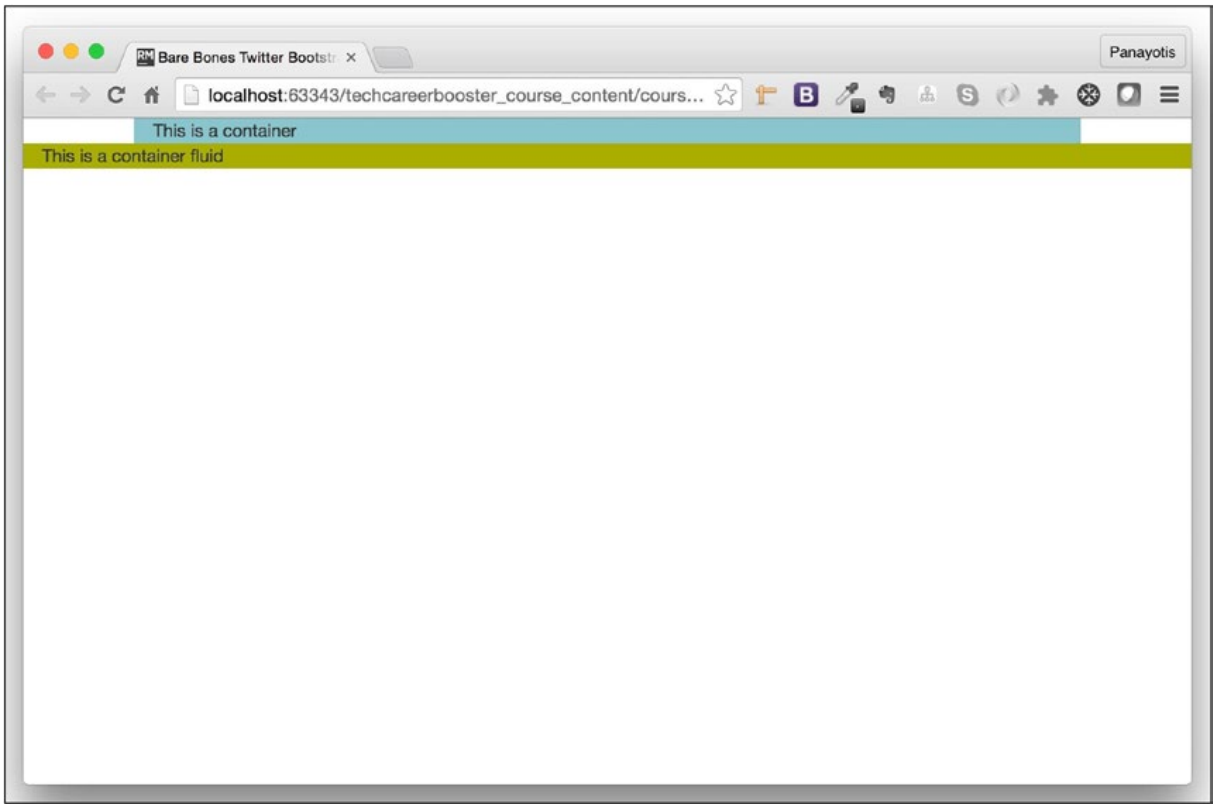
```
.container {
    background-color: #8DC5CC;
}

.container-fluid {
    background-color: #aaad21;
}
```

Caution Now that you are using Twitter Bootstrap, you will not need to write CSS reset rules like `* {box-sizing: border-box}`. This is because Twitter Bootstrap itself has a series of defaults that make very good sense, and they are aligned with the rest of the Twitter Bootstrap classes.

As you can imagine, this is used to apply specific, different colors to the two containers so that I can see how they are put on the page.

Save all files and load the `index.html` page on your browser. What you will see should be something like Figure 1-4.


Bare Bones Twitter Bootstr
Panayotis
localhost:63343/techcareerbooster_course_content/cours...

This is a container
This is a container fluid

Figure 1-4. *container vs. container-fluid*

As you can see in Figure 1-4, the `container` has left and right margins, whereas the `container-fluid` does not have margins and covers the whole width of the page. `container` margins are not of fixed width. Their width changes depending on the viewport width.

On the other hand, both containers have left and right paddings, equal to `15px`. You can see that the text "This is a container ..." does not start at the edge of the left border, but it has some free blank space.

Note that even if you use a `container`, when you shrink the width of your browser window, so that it becomes less than `576px` wide, the `container` loses its left and right margins and, essentially, behaves like the `container-fluid`. In other words, this page on an Apple iPhone X device would be displayed like what you see in Figure 1-5.

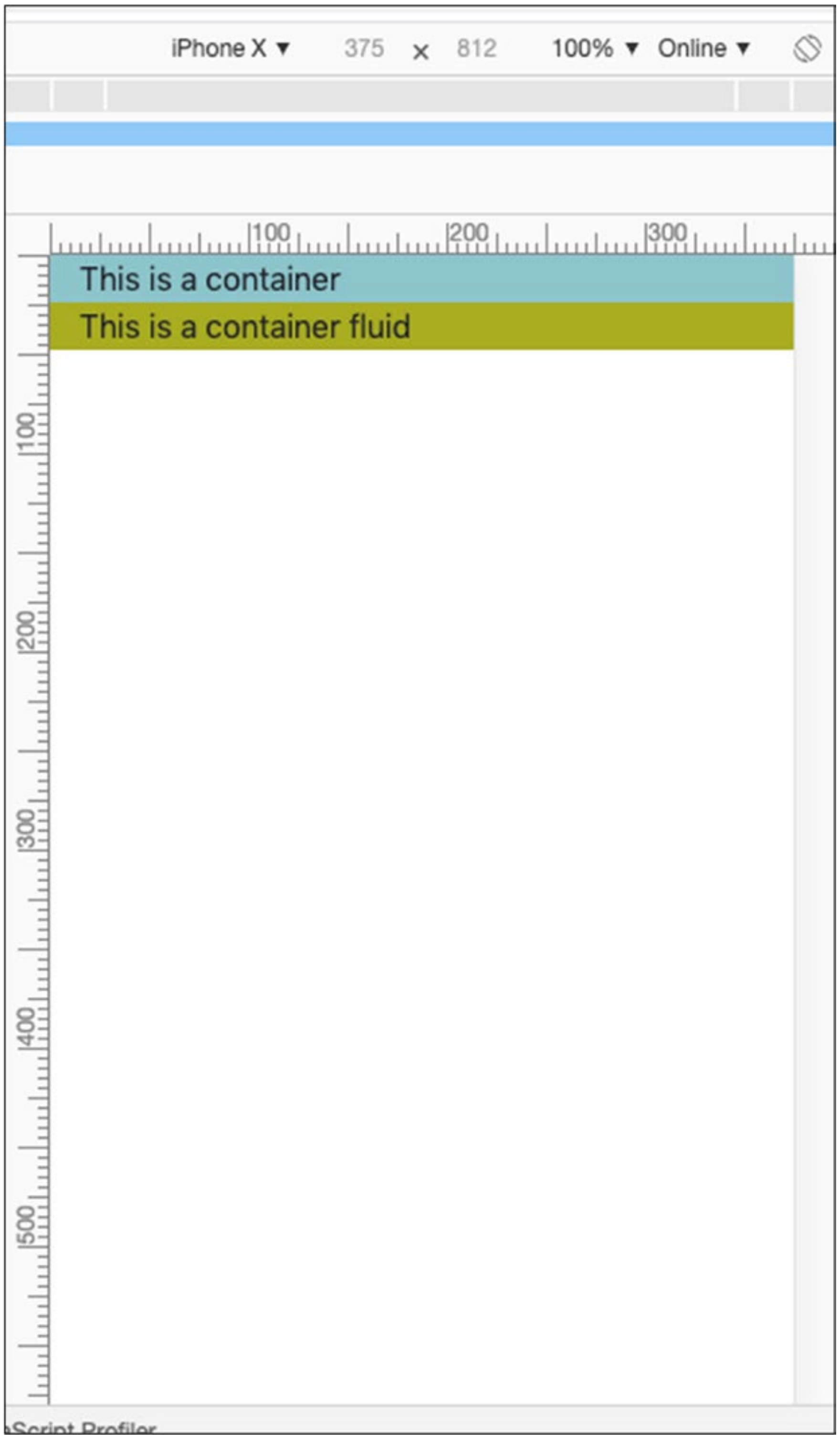


Figure 1-5. *`container` Behaves like `container-fluid` on iPhone X*

Hence, `container` and `container-fluid` in that case would be the same.

Grid System

The grid system, that is, dividing your layout into rows and columns, is a good practice to design the layout of your page. Twitter Bootstrap comes with handy classes that will allow you to scale your layout up to 12 columns, as the device or viewport size increases.

Hence, whenever you want to divide your page into a series of columns, you have to

1. Decide on the container type.
2. Inside the container, add a `row` class `div`.
3. Inside the `row` class `div`, add one or more `col-*` class `div` elements.

I will explain what a `col-*` class is using some examples.

Example: One Row with 12 Columns

Let's create a page with a one-row and 12-column layout as in Figure 1-6.

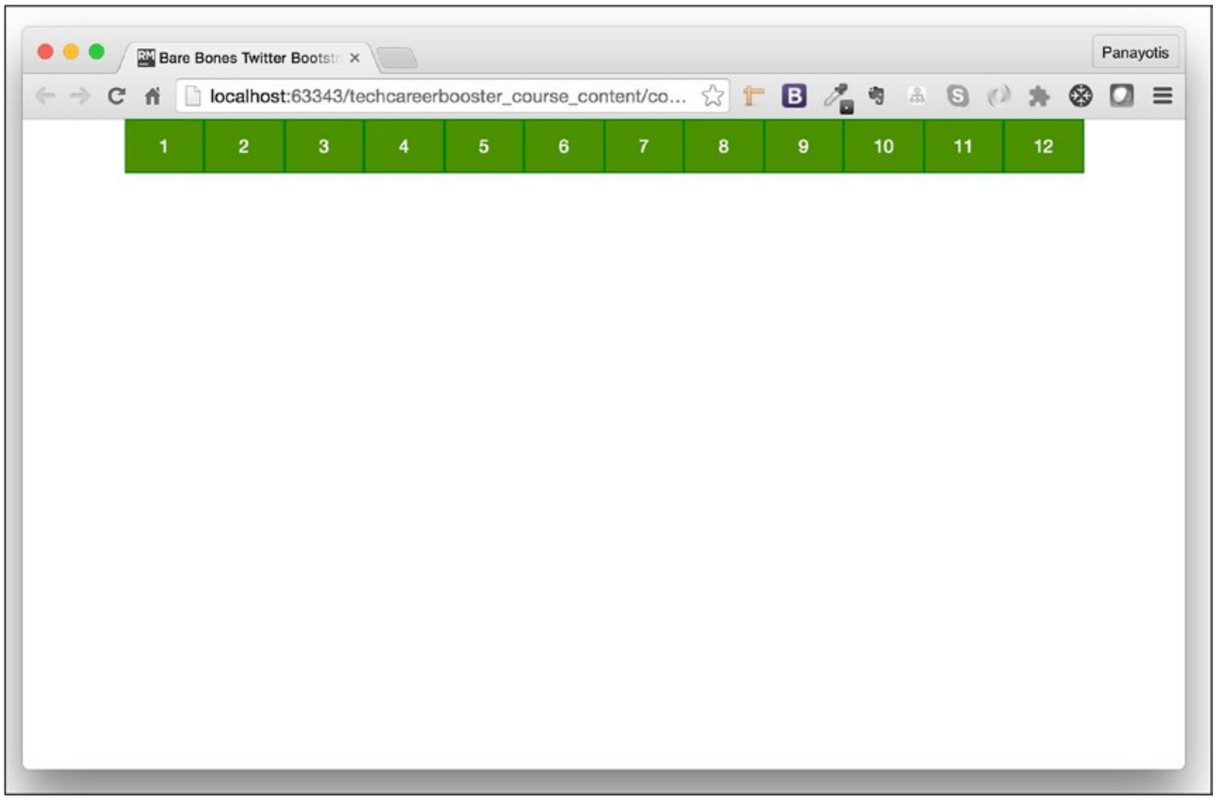

Bare Bones Twitter Bootstr
Panayotis
localhost:63343/techcareerbooster_course_content/co...
1
2
3
4
5
6
7
8
9
10
11
12

Figure 1-6. *One-Row 12-Column Grid*

One way you can create this page is shown in the following (Listing 1-4).

Listing 1-4. One-Row 12-Column Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>1 row 12 columns</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
            <div class="col">
                7
            </div>
            <div class="col">
                8
            </div>
            <div class="col">
                9
            </div>
            <div class="col">
                10
            </div>
            <div class="col">
                11
            </div>
            <div class="col">
                12
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
```

```
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

The CSS file (named `index.css`) is given in Listing 1-5.

Listing 1-5. CSS for the One-Row 12-Column Layout

```
.container {
    background-color: #8DC5CC;
}

.container .row div {
    background-color: #4e9013;
    padding: 10px 0;
    border: 1px solid green;
    color: white;
    text-align: center;
}
```

As you can see in the HTML file, `index.html`, I have used a `container` and inside it I have introduced a `row` class `div`. Then for each one of the 12 columns, I have used a `col` class `div.`

Again, I wanted 12 columns, and I added 12 `divs` with class `col`. All 12 `divs` have been added inside a single `row div`. The `row div` is used to lay out columns, and only columns can be immediate children of rows.

Example: Two-Row Grid, the Second with Six Columns

Let's add one more row to our grid page. But this time, let's assume that we want six columns on the second row, as in Figure 1-7.

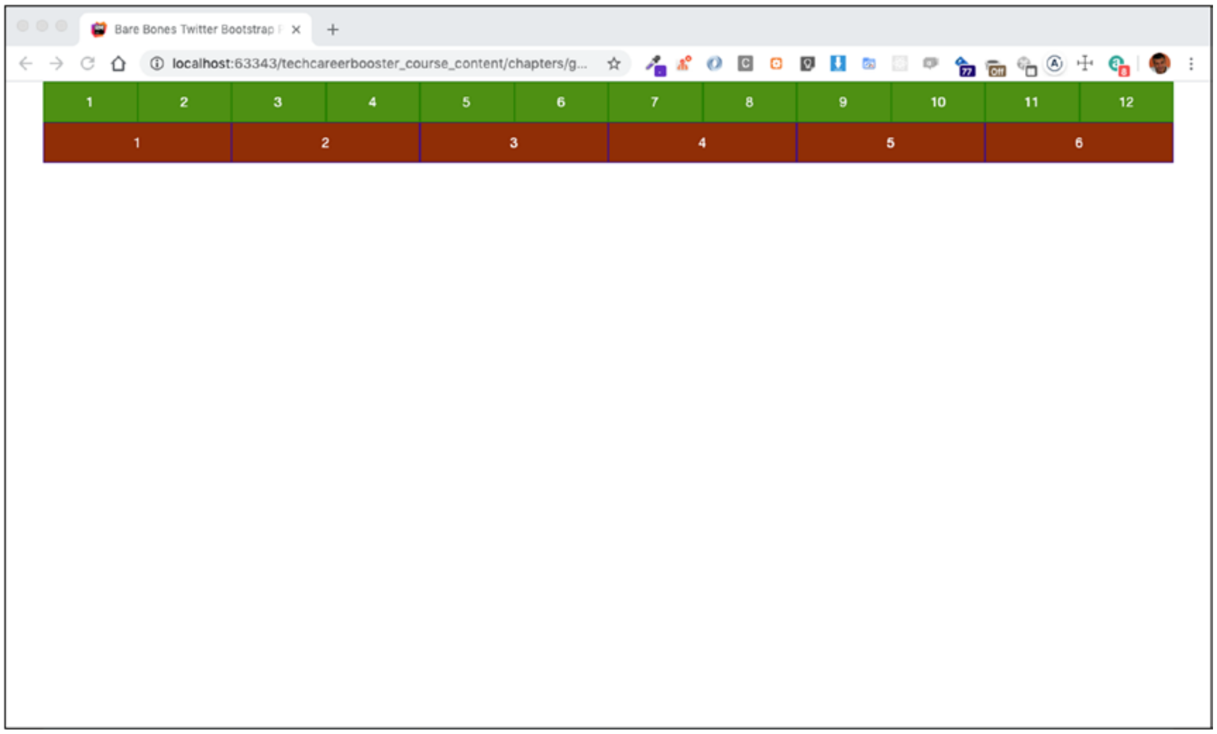

Bare Bones Twitter Bootstrap
localhost:63343/techcareerbooster_course_content/chapters/g...
1
2
3
4
5
6
7
8
9
10
11
12
1
2
3
4
5
6

Figure 1-7. *Two Rows, the Second Row with Six Columns*

As you can see in Figure 1-7, each one of the columns of the second row occupies double the column width of the columns of the first row.

The full HTML code is in Listing 1-6.

Listing 1-6. Two Rows, the Second Row Having Six Columns, HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>2 rows 2nd 6 columns</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
            <div class="col">
                7
            </div>
            <div class="col">
                8
            </div>
            <div class="col">
                9
            </div>
            <div class="col">
                10
            </div>
            <div class="col">
                11
            </div>
            <div class="col">
                12
            </div>
        </div>
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPoOiEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
```

```
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

Listing 1-7 is the full CSS code (which is not critical to the creation of the grid, but it is only there to help me visualize the columns of the rows of the grid on my page).

Listing 1-7. CSS for the Page Having Two Rows, the Second Row Having Six Columns

```
.container {
    background-color: #8DC5CC;
}

.container .row:nth-of-type(1) div {
    background-color: #4e9013;
    padding: 10px 0;
    border: 1px solid green;
    color: white;
    text-align: center;
}

.container .row:nth-of-type(2) div {
    background-color: #902e07;
    padding: 10px 0;
    border: 1px solid #441180;
    color: white;
    text-align: center;
}
```

As you can see, it is very easy to have six columns by only having six `div`s with class `col` as children of the `div` with class `row`.

Example: Two-Row Grid, the Second with Five Columns

Let’s say that you want one row to have five columns, instead of six, with the first column being double the size of the others. For example, see Figure 1-8.

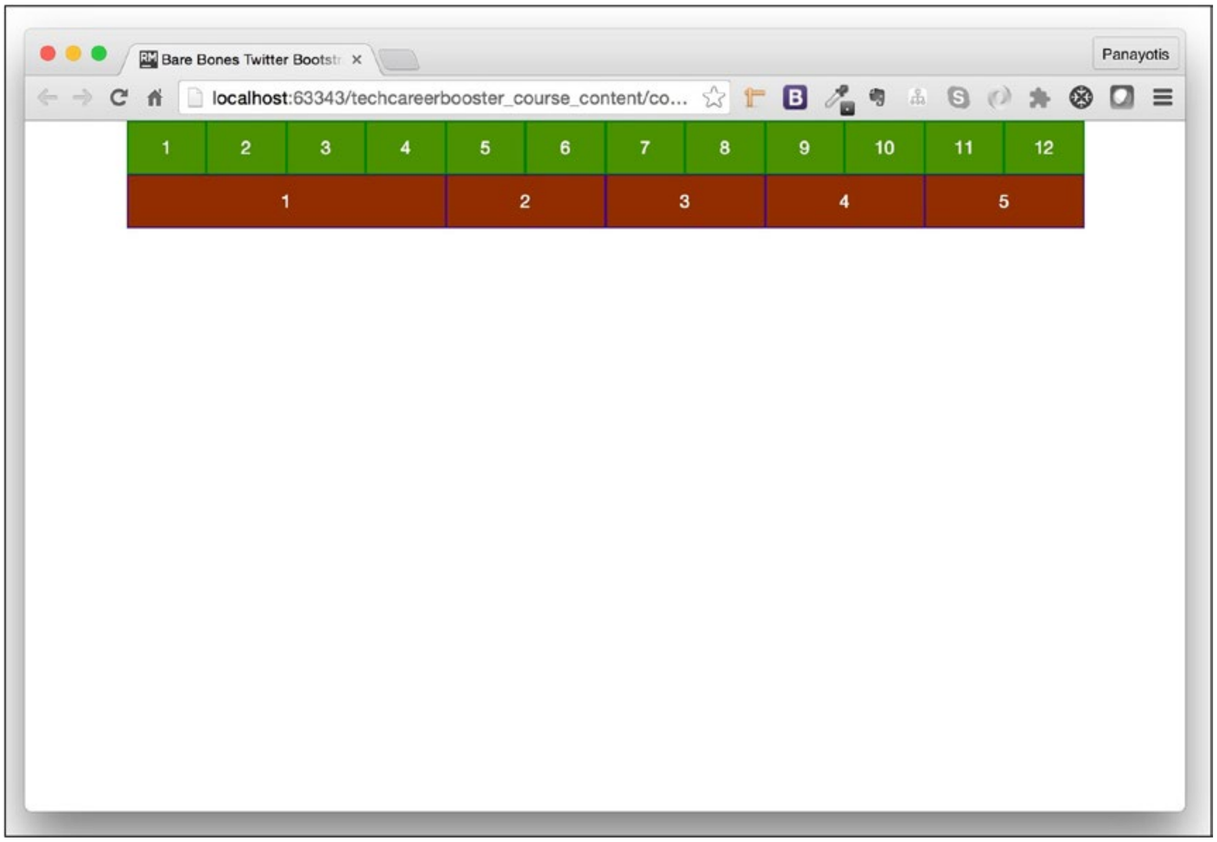

Bare Bones Twitter Bootstr
Panayotis
localhost:63343/techcareerbooster_course_content/co...
1
2
3
4
5
6
7
8
9
10
11
12
1
2
3
4
5

Figure 1-8. *Second Row Having Five Columns, First Column Being Wider*

How would you do that? If you just changed your code to have five divs, inside the second `row div`, what would you get? You would get the page on Figure 1-9.

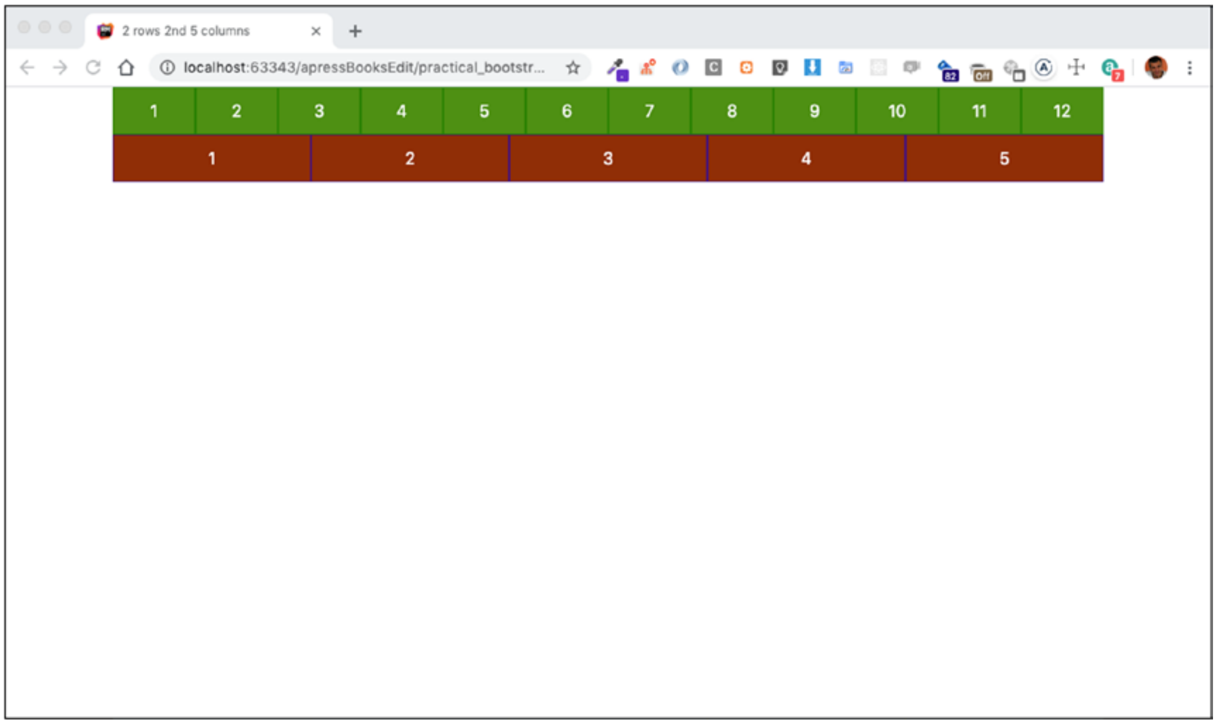


Figure 1-9. *Second Row Having Five Columns of Equal Size*

This is not exactly what you want. If you just put five `divs`, the columns generated have equal width. But what you want is the first column to be double the width of the others. Also, if you look more carefully, you will see that the first column occupies four out of the 12 available standard columns, whereas the other columns occupy two. Again, this is what you want to achieve (Figure 1-10).

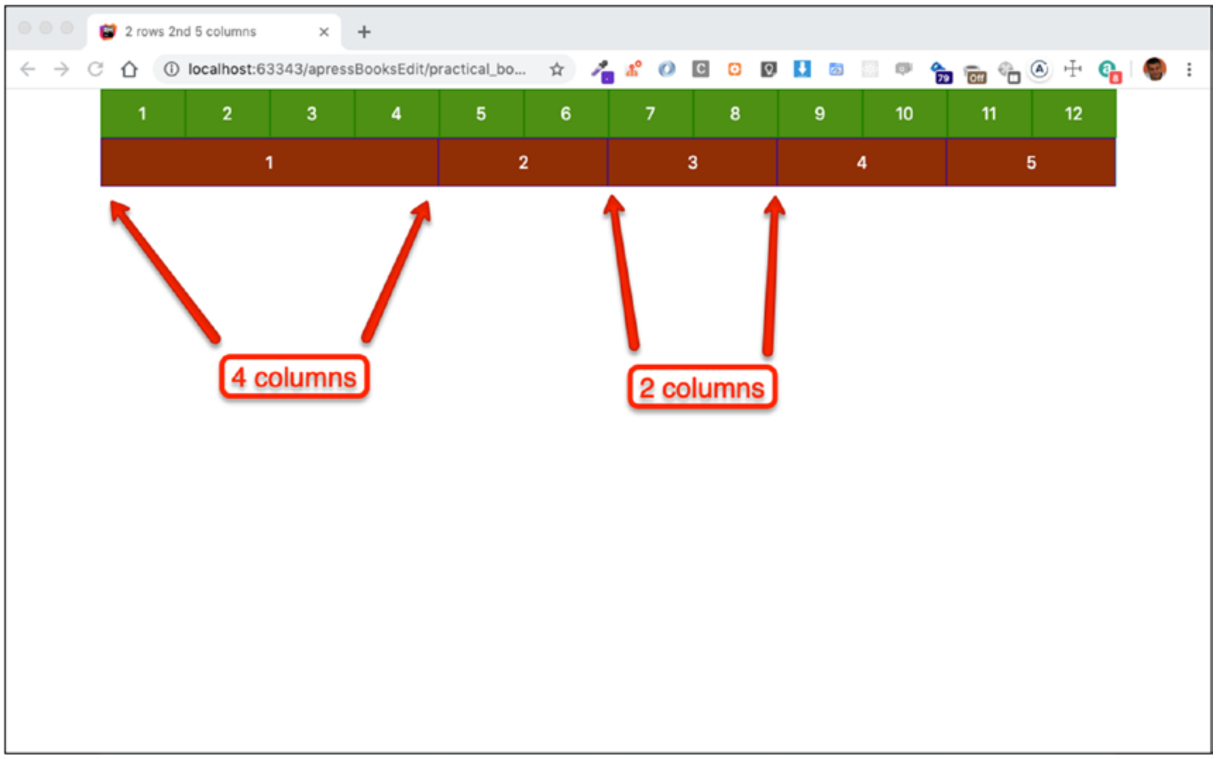


Figure 1-10. *What You Want to Achieve*

In order to do that, you have to be more precise on the class of each column `div`. You have to tell exactly how many standard columns each column should occupy.

Hence, for the first column, you have to use the class `col-4`, whereas for the other columns, you have to use the class `col-2`. The correct HTML code is in Listing 1-8.

Listing 1-8. Using col-4 for the First Column and col-2 for the Others

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>2 rows 2nd 5 columns</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
            <div class="col">
                7
            </div>
            <div class="col">
                8
            </div>
            <div class="col">
                9
            </div>
            <div class="col">
                10
            </div>
            <div class="col">
                11
            </div>
            <div class="col">
                12
            </div>
        </div>
        <div class="row">
            <div class="col-4">
                1
            </div>
            <div class="col-2">
                2
            </div>
            <div class="col-2">
                3
            </div>
            <div class="col-2">
                4
            </div>
            <div class="col-2">
                5
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
```

```
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

Another way you could divide your second row into five columns is shown in Listing 1-9.

Listing 1-9. Another Way to Divide into Five Columns

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>2 rows 2nd 5 columns</title>
</head>
```

```
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
            <div class="col">
                7
            </div>
            <div class="col">
                8
            </div>
            <div class="col">
                9
            </div>
            <div class="col">
                10
            </div>
            <div class="col">
                11
            </div>
            <div class="col">
                12
            </div>
        </div>
        <div class="row">
            <div class="col-3">
                1
            </div>
            <div class="col-2">
                2
            </div>
            <div class="col-3">
                3
            </div>
            <div class="col-2">
                4
            </div>
            <div class="col-2">
                5
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

If you save and load this page on your browser, you will see what is depicted in Figure 1-11.

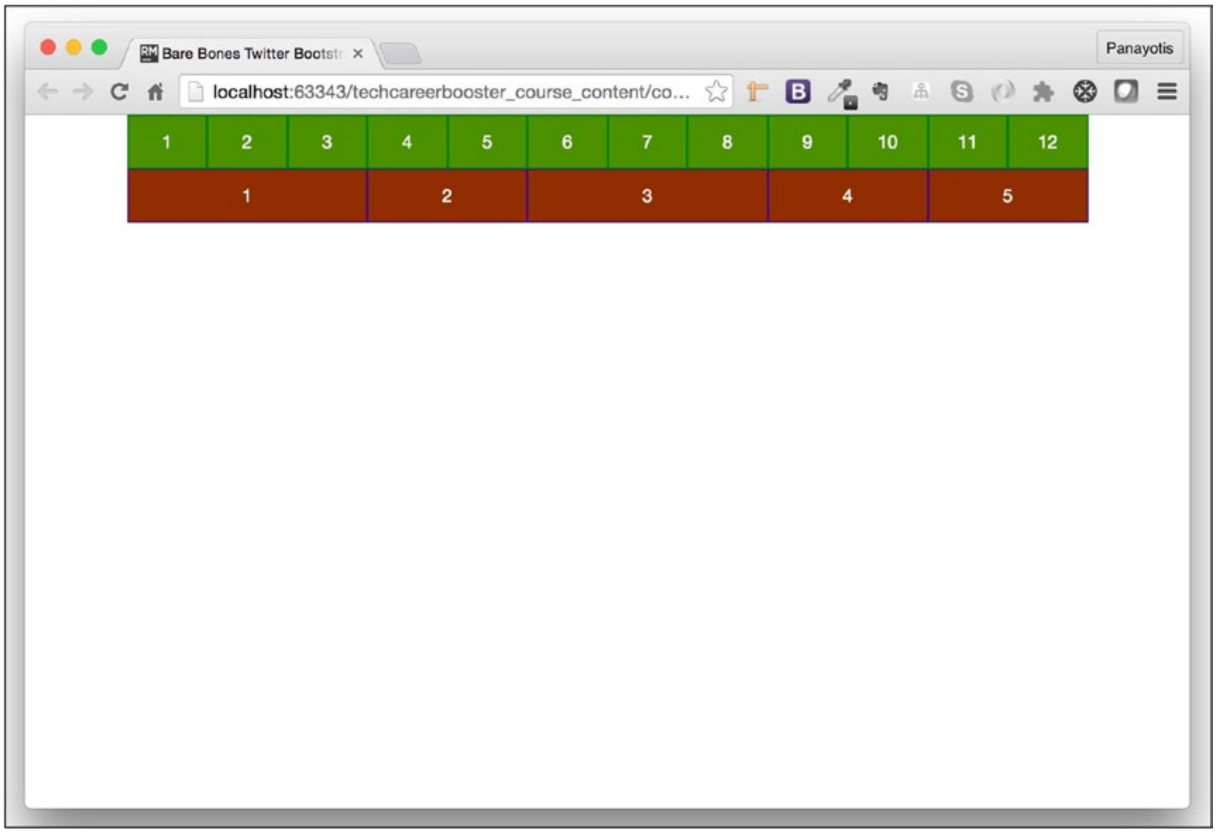

Bare Bones Twitter Bootstr
Panayotis
localhost:63343/techcareerbooster_course_content/co...
1
2
3
4
5
6
7
8
9
10
11
12
1
2
3
4
5

Figure 1-11. *Another Way to Divide into Five Columns*

As you can see in Figure 1-11, the second row is, again, with five columns, but with a different size for the first and third columns. These two are of size `3`, whereas all the other columns are of size `2`. However, even in this case, the total column sizes add up to 12: `3 + 2 + 3 + 2 + 2`.

Example: Two-Row Grid, the Second Having Three Columns, with the Middle Wider

But you don't always have to be precise about the width of all the columns. Sometimes, you want to be specific only about the width of one column, and you want the others to have a width automatically calculated. See, for example, Figure 1-12.

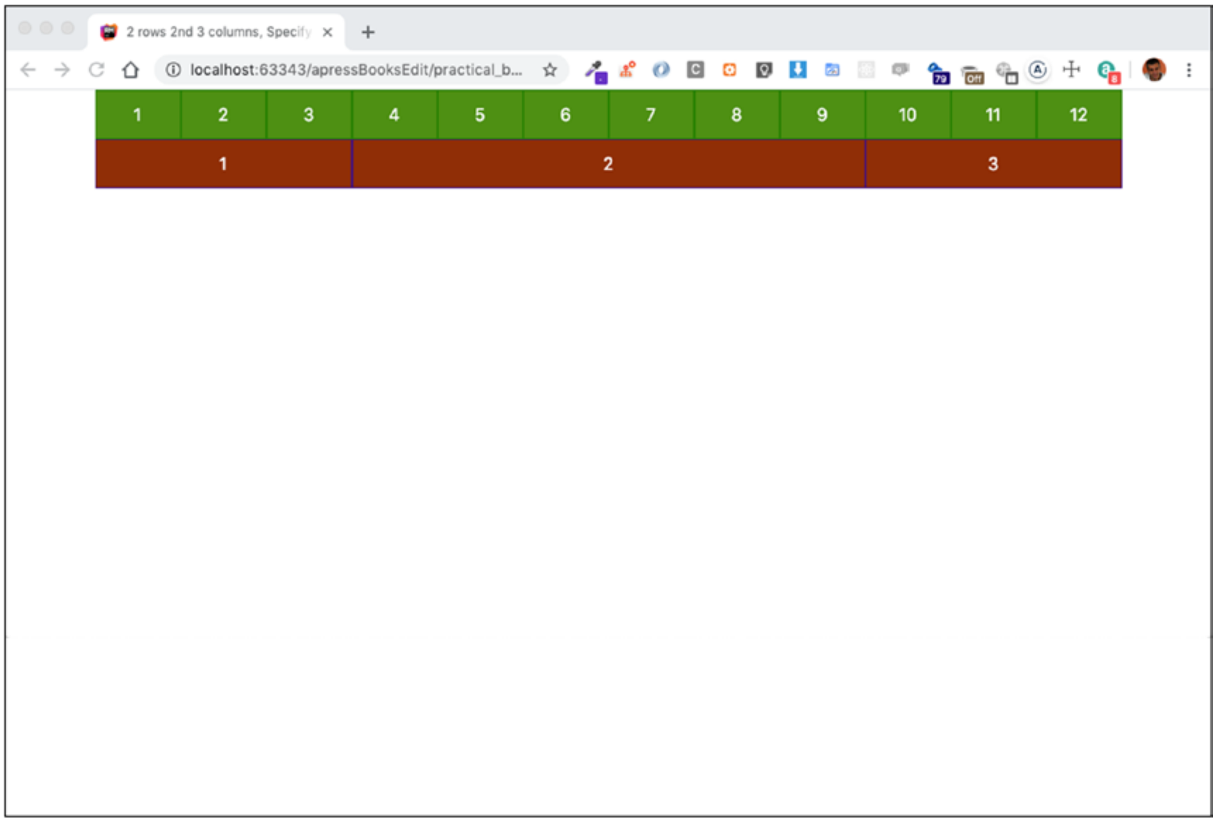

2 rows 2nd 3 columns, Specify
localhost:63343/apressBooksEdit/practical_b...
1
2
3
4
5
6
7
8
9
10
11
12
1
2
3

Figure 1-12. *Middle Column to Be Half of Available Width*

You want the middle column to be occupying half of the available width and leave the rest of the space shared between the first and third columns.

You could do that by giving `col-3` to the first and third `div`s and `col-6` to the second `div`, but it appears that you can do that by giving `col` to the first and third `div`s and `col-6` to the second `div`, as in Listing 1-10.

Listing 1-10. Specifying the Width of the Middle Column Only

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
    bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>Specifying the width of the middle column only</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col">
                2
            </div>
            <div class="col">
                3
            </div>
            <div class="col">
                4
            </div>
            <div class="col">
                5
            </div>
            <div class="col">
                6
            </div>
            <div class="col">
                7
            </div>
            <div class="col">
                8
            </div>
            <div class="col">
                9
            </div>
            <div class="col">
                10
            </div>
            <div class="col">
                11
            </div>
            <div class="col">
                12
            </div>
        </div>
        <div class="row">
            <div class="col">
                1
            </div>
            <div class="col-6">
                2
            </div>
            <div class="col">
                3
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

Different Display Widths

Twitter Bootstrap allows you to use different classes for your grid layout (and for other properties of your web page style), according to the different display widths that your web page will be displayed on. In particular, there are five key width breakpoints as listed in Table 1-1.

Table 1-1. *Twitter Bootstrap Responsive Breakpoints*

Name	Symbol	Display Width
Extrasmall	`xs`	<576px
Small	`sm`	>=576px
Medium	`md`	>=768px
Large	`lg`	>=992px
Extralarge	`xl`	>=1200px

You are referring to them by their corresponding symbol, as part of the class name that you are using.

So, if you are using, for example, the class `col-lg-5`, this means that this class refers to the breakpoint `lg`, for devices with width >=992px. However, further interpretation of each particular class depends on the class case itself. We will learn the most important classes in this section.

The Twitter Bootstrap grid classes that work based on the preceding five key breakpoints are

- `col-1`, `col-2`, and so on, up to `col-12`, for the extrasmall devices
- `col-sm-1`, `col-sm-2`, and so on, up to `col-sm-12`, for small devices
- `col-md-1`, `col-md-2`, and so on, up to `col-md-12`, for medium devices
- `col-lg-1`, `col-lg-2`, and so on, up to `col-lg-12`, for large devices
- `col-xl-1, col-xl-2,` and so on, `up to col-xl-12`, for extralarge devices

But what is the difference between `col-sm-2` and `col-md-2`, for example?

They both occupy two columns. `col-sm-2` will occupy two columns for any display area width that is >=576px, but `col-md-2` will occupy two columns only for display widths that are >= 768px. `col-md-2`, for display areas that are <768px, will not occupy two columns, but will stack one column above the other, as if they were two rows, essentially each column occupying the whole 12-column available width.

Let's see an example here. You are going to create a page with a row that has six columns, each one occupying two-column width. Write the following code into an HTML document (Listing 1-11).

Listing 1-11. Six Columns with the `sm` Breakpoint

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>Listing 1-11</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-sm-2">
                1
            </div>
            <div class="col-sm-2">
                2
            </div>
            <div class="col-sm-2">
                3
            </div>
            <div class="col-sm-2">
                4
            </div>
            <div class="col-sm-2">
                5
            </div>
            <div class="col-sm-2">
                6
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

Write the following (Listing 1-12) CSS in the corresponding `index.css` file inside the `stylesheets` folder.

Listing 1-12. CSS for Listing 1-11

```
.container {
    background-color: #8DC5CC;
}

.container .row:nth-of-type(1) div {
    background-color: #4e9013;
    padding: 10px 0;
    border: 1px solid green;
    color: white;
    text-align: center;
}
```

If you save the preceding code and then load the HTML page on your browser, you will see the following (Figure 1-13).

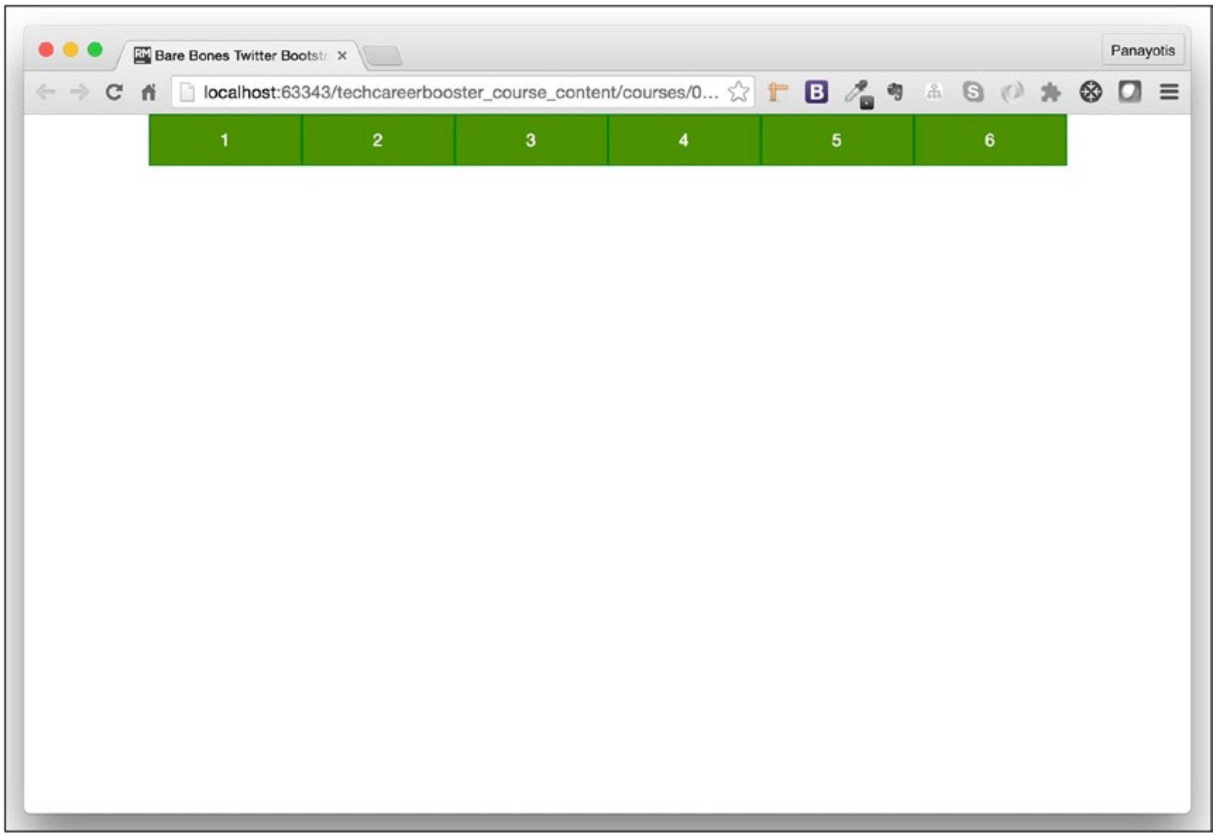

Bare Bones Twitter Bootstr
Panayotis
localhost:63343/techcareerbooster_course_content/courses/0...
1
2
3
4
5
6

Figure 1-13. *One-Row Six-Column Layout—Each One Occupying Two Columns*

As you can see in the HTML code, I am using the breakpoint `sm`, and I instruct that each column occupies two of the available 12 columns. This will result in the row being divided into six columns, but only for display areas with width >=576px.

Enable the developer tools and shrink the browser window so that it has width exactly 576px. You will not see any significant difference on how your page is displayed. All six columns will be there, horizontally arranged (Figure 1-14).

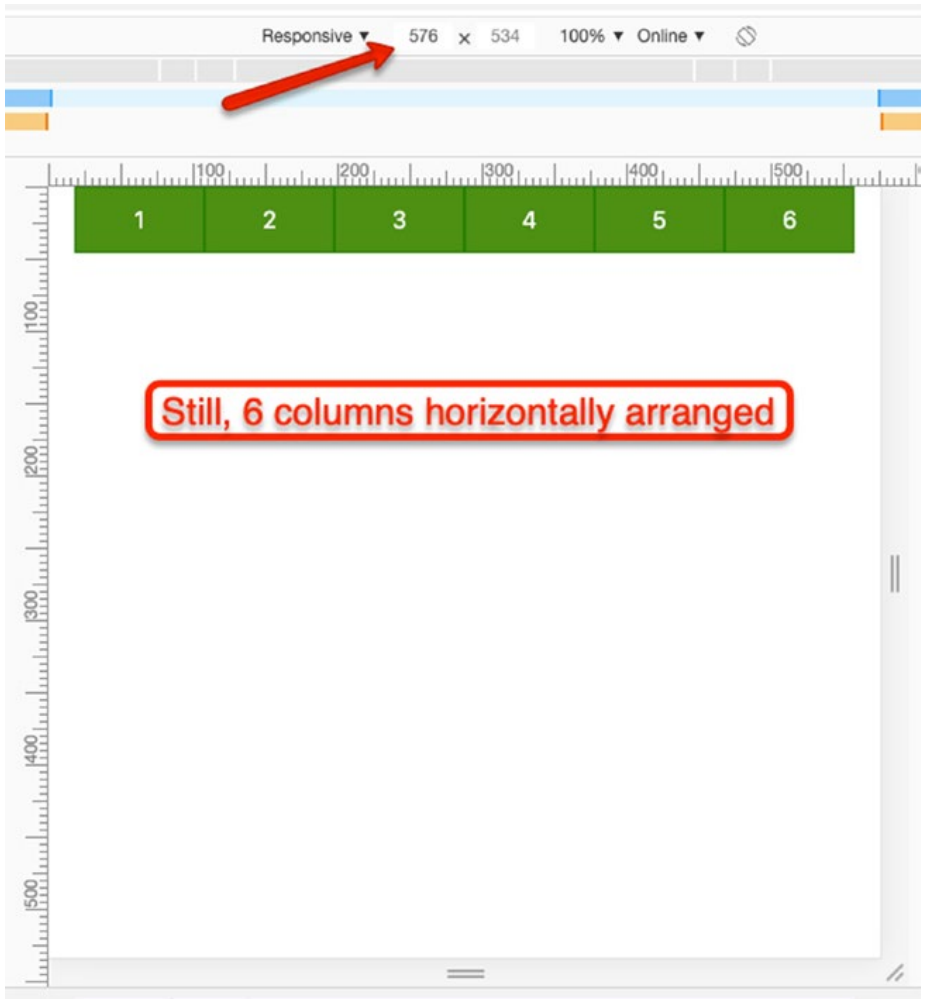


Figure 1-14. *Exactly 576px Width—Still Six Columns in Width*

Things will change when you shrink 1 pixel more, to 575px. You will see the following (Figure 1-15).

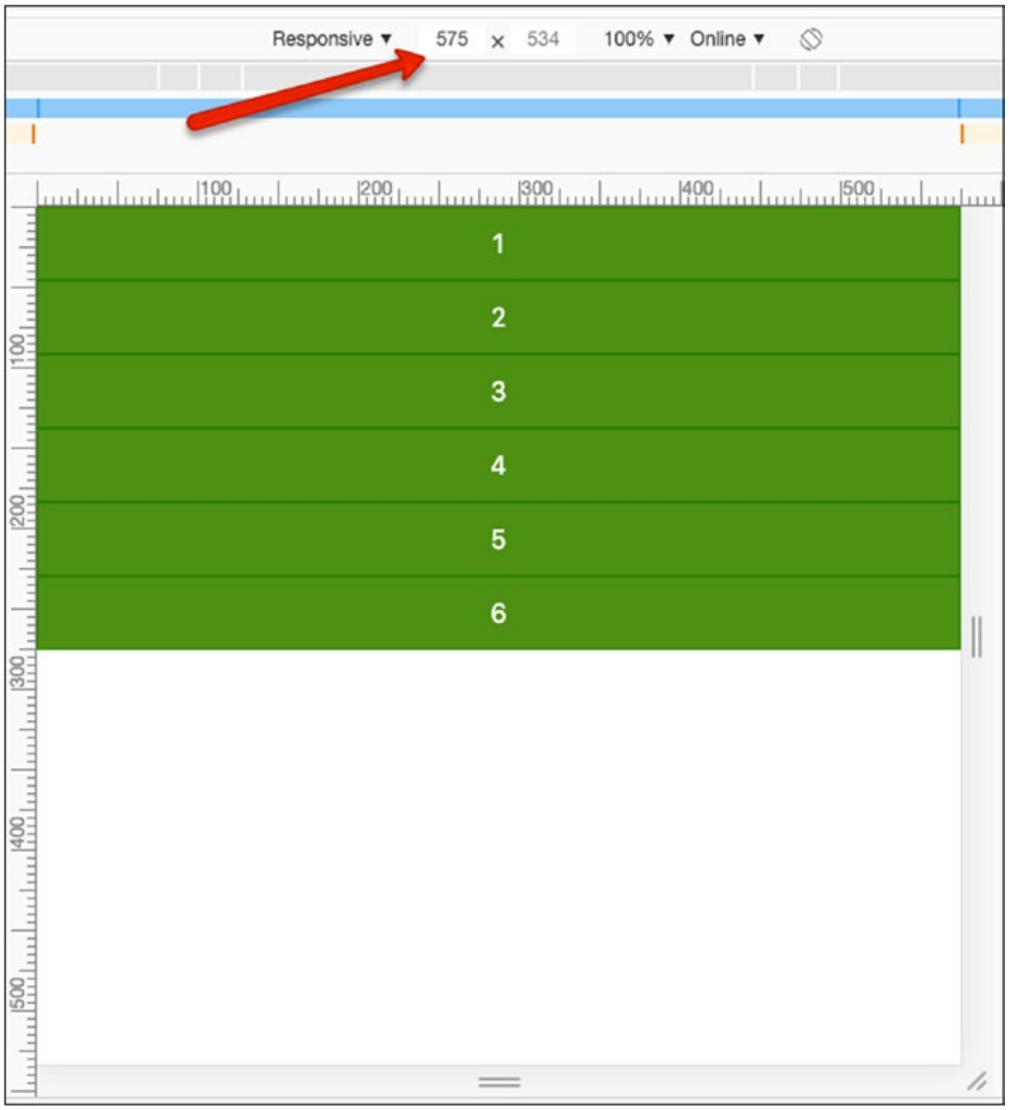


Figure 1-15. *Six Columns Stacked When Below the `sm` Breakpoint*

This happened because I have used the class `col-sm-2`. Again, this class stacks the columns one on top of the other for each display width less than 576px and keeps the columns horizontal for any width greater than or equal to 576px.

As you can see from the preceding picture, the initially two-column-wide columns are now 12-column-wide, occupying the whole row.

The nice thing with Bootstrap is that you can apply multiple classes of different breakpoints on the same `div`, in order to adapt the grid layout for different devices.

See the following example in order to understand what I mean. Let's suppose that I want, for small devices and above, to have six columns, but for smaller devices (less than 576px), I do not want one column per row, but two columns per row.

Hence

1. For small devices or larger, I want six columns. This means that I need to use the `col-sm-2` class for the `divs` (`sm` for small and `2` for six columns, since `12/6` gives `2`).
2. For extrasmall devices, I want two columns per row. Hence, I need to use the `col-6` class for the `divs` (for extrasmall, I don't interpolate the symbol of the device width into the class name, since the default value is `xs` and `6` for two columns, since `12/2` gives `6`).

Let's try that. Change the content of the preceding HTML page, in order to be like that in Listing 1-13.

Listing 1-13. Two Columns on Extrasmall Displays, but Six on Small Displays and Wider

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.
    com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
    Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>Listing 1-12</title>
</head>
```

```
<body>
    <div class="container">
        <div class="row">
            <div class="col-6 col-sm-2">
                1
            </div>
            <div class="col-6 col-sm-2">
                2
            </div>
            <div class="col-6 col-sm-2">
                3
            </div>
            <div class="col-6 col-sm-2">
                4
            </div>
            <div class="col-6 col-sm-2">
                5
            </div>
            <div class="col-6 col-sm-2">
                6
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

If you save the preceding code and reload your page, while it is on 575px width (or smaller), you will see that, now, your page has two columns per row (Figure 1-16).

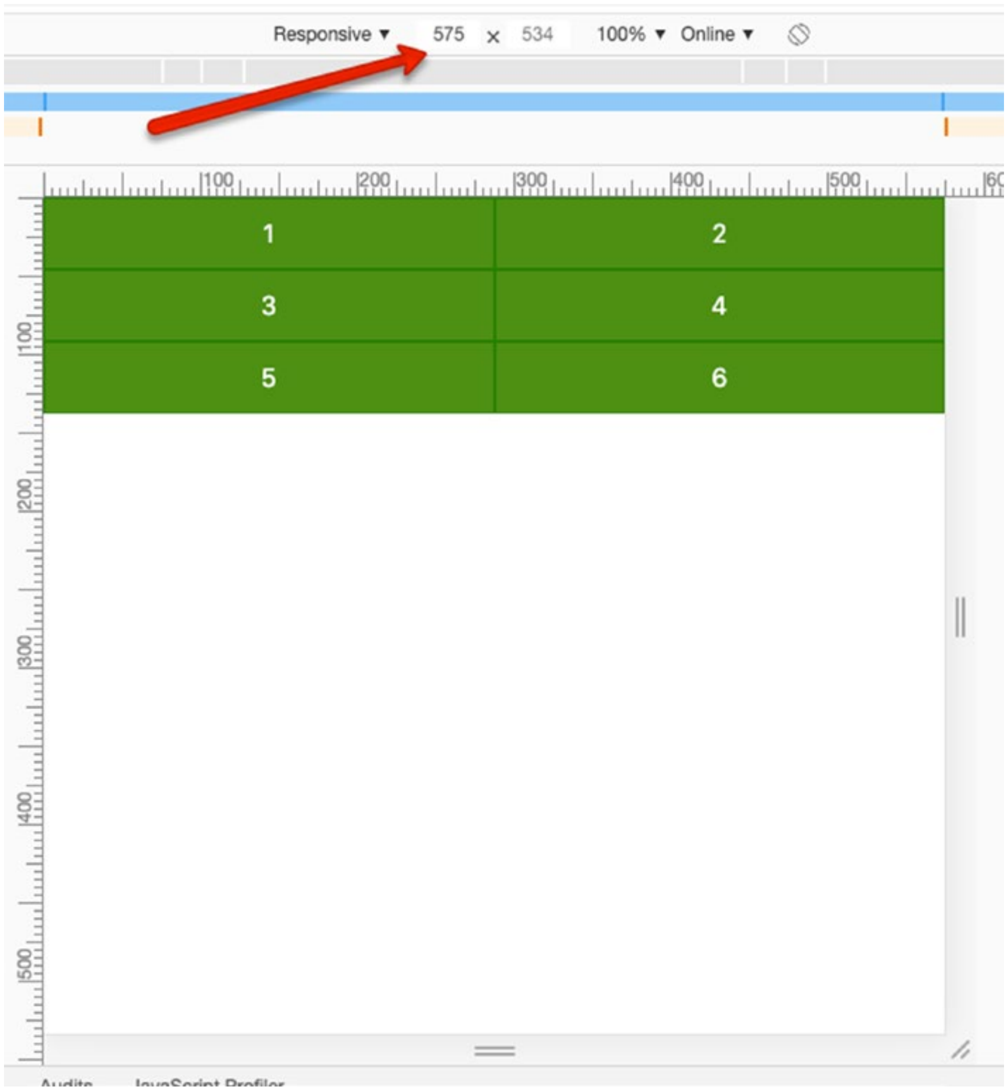


Figure 1-16. *Two Columns on Extrasmall Devices*

However, if you enlarge your display area to have a width greater than or equal to 576px, you will see that your page has six columns per row.

This is how the responsiveness of your grid is achieved.

Note One might say, "Hold on! When on extrasmall devices, the columns inside the `div have a number` suffix (`-6`) that adds up to 36, not to 12." You need to know here that Twitter Bootstrap will automatically wrap the columns above the 12th to a new row.

The previous web page had one grid layout breakpoint actually—the 576px, whichever display width below 576px vs. whichever display width equal to or greater than 576px. You can apply the same concept to a second breakpoint in the same grid. For example, you may want the number of columns to be three for display areas greater than 576px (small) and six for display areas greater than or equal to 768px (medium or wider).

So what I want is the following:

a. For extrasmall devices (<576px), two columns per row (hence, `col-6`)

b. For small devices (>=576px, <768px), three columns per row (hence, `col-sm-4`)

c. For medium or wider devices (>=768px), six columns per row (hence, `col-md-2`)

Now that I know what I want to achieve, it is pretty simple. I will apply the preceding classes to my column `div`s, and I am done (Listing 1-14).

Listing 1-14. Two for `xs`, Three for `sm`, and Six for Wider Displays

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
    bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGs
    O3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
    crossorigin="anonymous">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="stylesheets/index.css" type="text/css">

    <title>Listing 1-14</title>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-6 col-sm-4 col-md-2">
                1
            </div>
            <div class="col-6 col-sm-4 col-md-2">
                2
            </div>
            <div class="col-6 col-sm-4 col-md-2">
                3
            </div>
            <div class="col-6 col-sm-4 col-md-2">
                4
            </div>
            <div class="col-6 col-sm-4 col-md-2">
                5
            </div>
            <div class="col-6 col-sm-4 col-md-2">
                6
            </div>
        </div>
    </div>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
    integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1
    yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/
    popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3
    zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/
    bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7Yw
    aYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
</body>
</html>
```

Now, if you scale your browser width to be, for example, 760px, then you will see three columns on each row. See Figure 1-17.

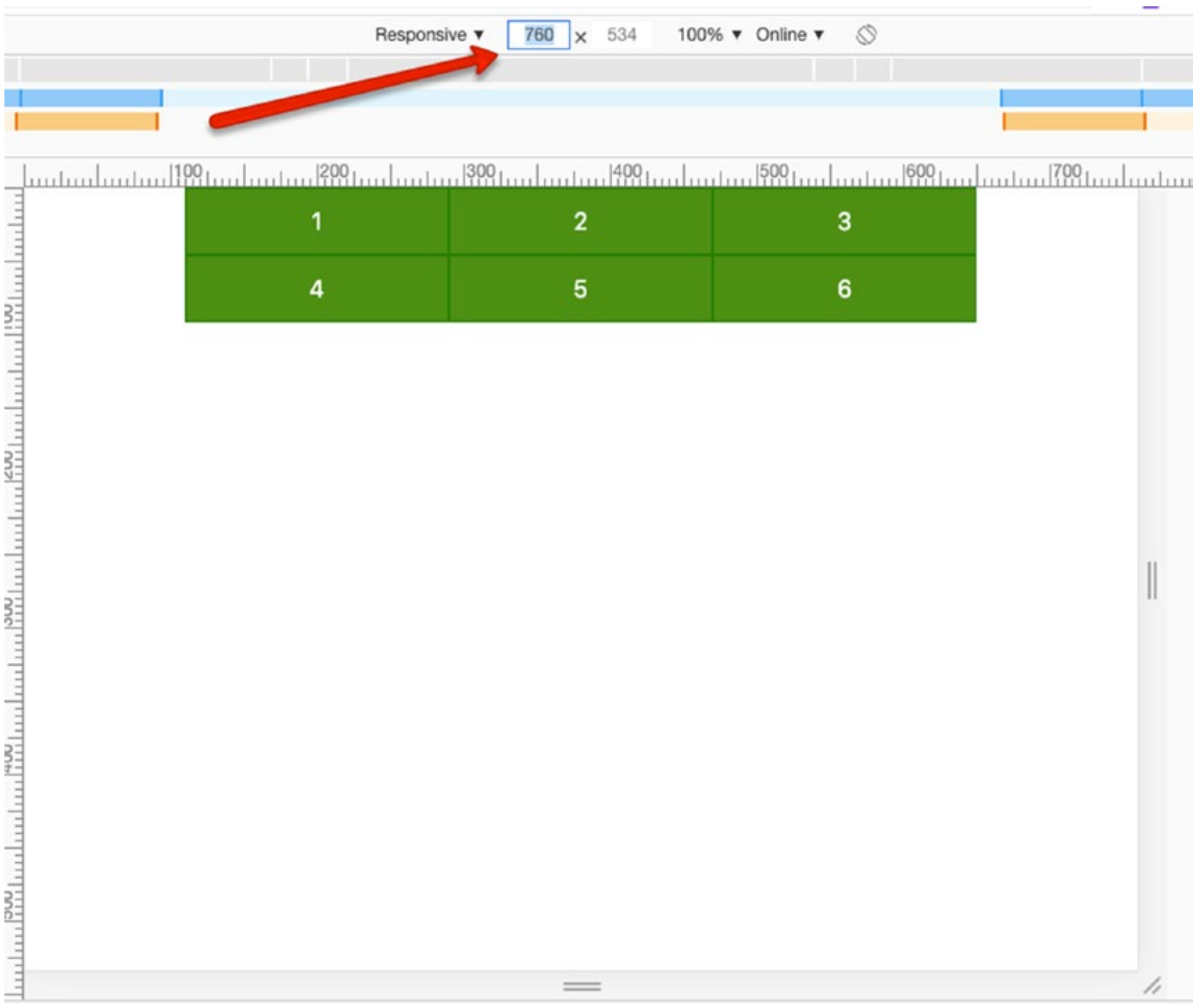


Figure 1-17. *Three Columns on Small-Size Displays*

But if you scale it even more, to be wider than 768px, it will display six columns per row (Figure 1-18).

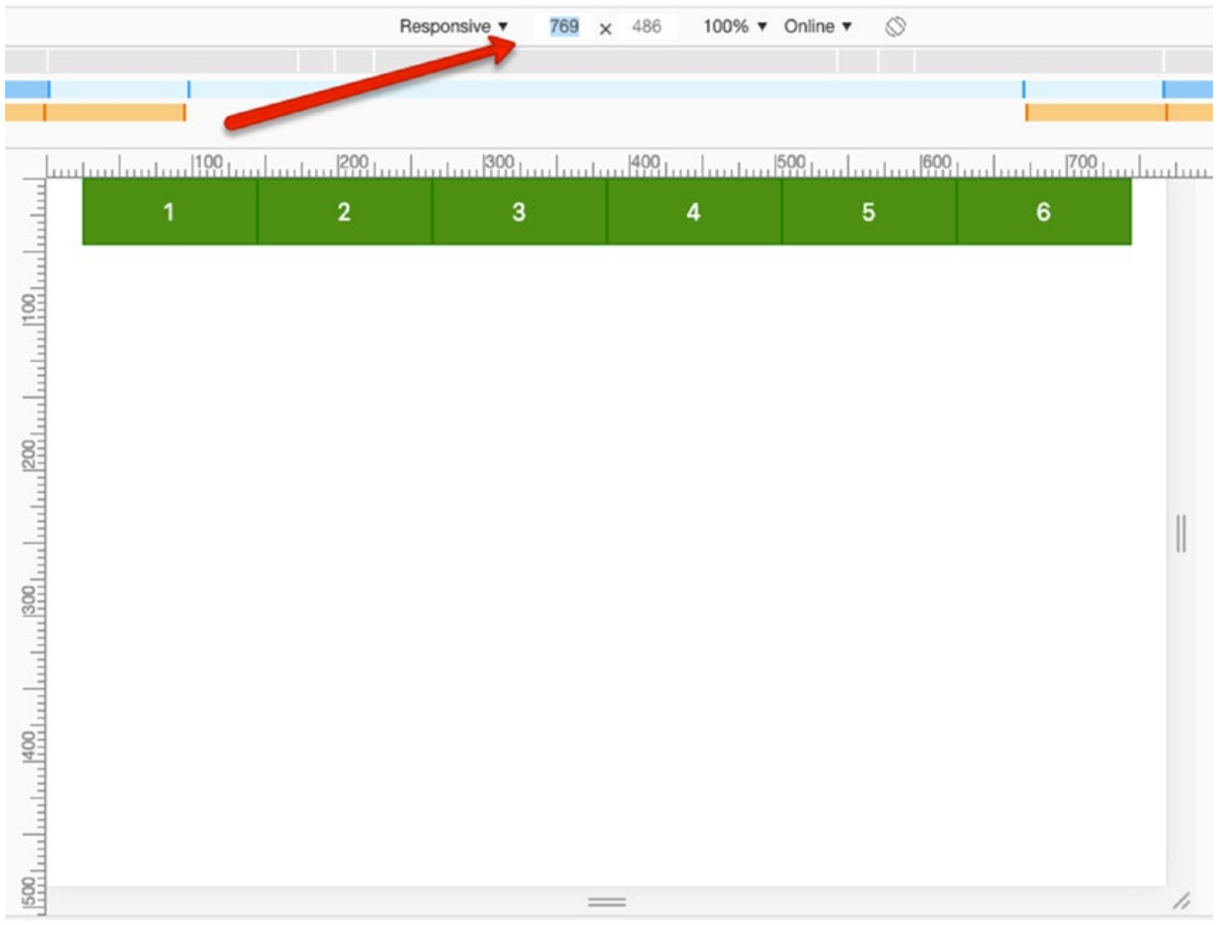


Figure 1-18. *Six Columns on Medium-Size Displays*

Tasks and Quizzes

TASK DETAILS

1. You need to implement a multicolumn layout like the ones presented in the chapter. Here are the requirements of the grid depending on the display width:

    1. For extrasmall displays, it should display one column per row.

    2. For small displays, it should display two columns per row.

3. For medium displays, it should display three columns per row.
4. For large displays, it should display four columns per row.
5. For extralarge displays, it should display six columns per row.

Good luck!

Key Takeaways

Congratulations! You have just finished the first chapter, and you already know so many things about how you make your web pages responsive. Here is a list of the key things you learned:

- The bare-minimum Twitter Bootstrap page and what it needs to include
- The containers
- The row divs
- The col divs as children of the row divs
- How to break the row into several columns of equal width or different widths
- The different Bootstrap breakpoints
- How to use different classes to define the layout of your page of different display widths

In the following chapter, you will use some more advanced techniques around the Twitter Bootstrap grid system.