Bootstrap tutorial

Rui Couto José C. Campos

Sistemas Interactivos
Mestrado Integrado em Engenharia Informática
DI/UMinho
March 9, 2020

Contents

1 Introduction

In the last tutorial you created an HTML page from scratch, using only HTML and CSS (and Sass!). The last step was to add some responsive features to the page. In this tutorial, it is proposed to recreate the same page, but resorting to a responsive framework. The chosen framework is Bootstrap[1]. The principle behind Bootstrap is the same as for other frameworks (e.g. Foundation, Skeleton), and Bootstrap is currently one of the most popular ones. This tutorial assumes that Bootstrap version 4 is being used.

2 Base template

Although it is possible to either add Bootstrap manually to an existing page, or download existing examples, for this tutorial a base template is provided. In this case, it is composed of the framework files (in the `bootstrap-4.3.1` folder), a base CSS (`starter-template.css` in the `style` folder) and the `index.html` file. Apart from the `starter-template.css`, you are not expected to modify any of the framework files. Instead, if you need to add something (CSS for instance), create new files.

The provided HTML page is as follows:

```
1  <!doctype html>
2  <html lang="en">
3    <head>
4      <!-- Required meta tags -->
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7      <!-- Bootstrap CSS -->
8      <link href="bootstrap-4.3.1/css/bootstrap.css" rel="stylesheet">
9      <title>Template</title>
10   </head>
11   <body>
12     <div class="container">
13
14     </div><!-- /.container -->
15
16     <!-- Bootstrap core JavaScript -->
17     <!-- Placed at the end of the document so the pages load faster -->
18     <script src="bootstrap-4.3.1/js/jquery-3.3.1.js"></script>
19     <script src="bootstrap-4.3.1/js/popper.min.js"></script>
20     <script src="bootstrap-4.3.1/js/bootstrap.js"></script>
21   </body>
22 </html>
```

In the template, lines 8 and 18 to 20 are the definition of the libraries and files required by Bootstrap. Line 13 is where the page content should be placed.

[1] `http://getbootstrap.com/`

Further details can be read in the included comments. After extraction of the template, it is possible to start building the page.

3 Grid system

Bootstrap relies in a grid system to define the layout of the pages (as opposed to the box model of CSS[2]). Each page is organised in a grid of *rows* and *columns*. The number of rows depends on the page's contents. The grid system has 12 base columns that divide the available horizontal space evenly. They can can be combined to create the desired number of columns in each row (hence, the width of a column is expressed in terms of the number of base columns that it spans).

3.1 Grid system basics

Figure 1 illustrates the usage of the grid system. The first row has 12 columns of width 1. Each of the columns in the row takes up one base column of the grid system. The second row has 2 columns, with the first measuring 8 base columns in width and the second 4 base columns in width. Hence, the first will measure twice the width of the second. The third row has 3 columns of width 4; and the fourth, 2 columns of width 6.

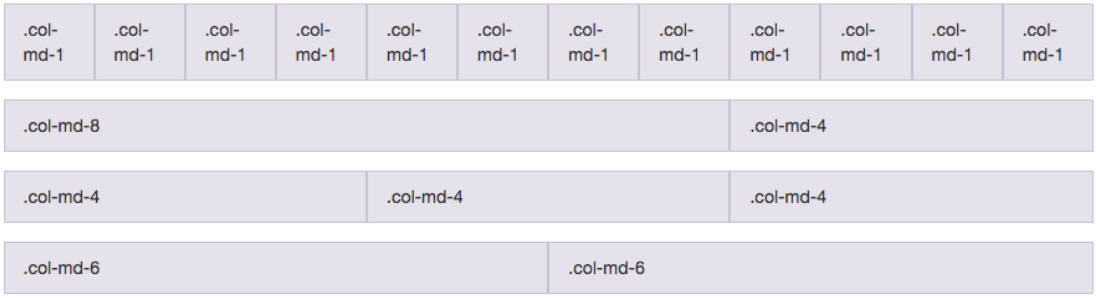

Figure 1: Bootstrap grid system.

Columns must be placed inside rows, and rows inside a special Bootstrap container. The container, columns and rows are declared using the HTML class attribute. Thus, in order to create a page resorting to the grid system, the page should contain:

- a `<div>` with the class `container`, i.e., `<div class="container">`
- inside this `<div>` each row is declared, i.e., `<div class="row">`

[2]See `https://www.w3schools.com/css/css_boxmodel.asp`, last visited April 1, 2019.

- and inside each row, the columns, i.e., `<div class="col">`

Note that the framework supports nesting.

Bootstrap is a responsive framework. Columns automatically adjust in size, allowing the page layout to adapt to different screen (or browser's window) sizes. By default, the horizontal space is divided evenly between the columns in a row. However, as mentioned above, it is possible to define the width of a column. This is done by using classes `col-*`, where `*` can be from 1 to 12 (representing the number of base columns to span) or `auto` to fit the column width to the content.

Bootstrap supports adjusting the layout to different screen sizes[3]. It might happen that, as the screen size decreases, it becomes impossible to fit all columns in the screen (in a useful manner). Hence, columns widths are usually defined for specific screen size classes (and they apply to for screen widths in that size class and above). This is done using classes `col-BP-S"`, where `BP` corresponds to the breakpoint size class (see below), and `S` corresponds to the column width to be used.

The following four size classes are defined:

.col-sm- for small devices ($\geq$576px) — phones

.col-md- for medium devices ($\geq$768px) — tablets

.col-lg- for large devices ($\geq$992px) — desktop displays

.col-xl- for extra large devices ($\geq$1200px) — desktop displays

Note that a fifth class exists (**.col-**), which corresponds to all screen sizes (i.e. including screen widths $<$576px).

It is possible to assign different widths for different size classes to a column. The declaration `<div class="col-sm-6 col-lg-3">` will make the column's width 6 (half the screen) for small and medium sizes and 3 (a quarter of the screen) for large ones. If the screen size is below any of the defined size classes, then columns stack vertically within the row (i.e. each takes all available horizontal space).

An example of a layout with three columns of widths 2, 3 and 7, for medium devices and up (for smaller sizes the rows will become vertically stacked) is presented in Figure 2:

3.2 Applying the grid system

In order to use Bootstrap, the layout of the web page must be defined in terms of rows and columns. The contents of the page is then placed inside the identified cells.

[3] In fact, different viewport sizes. We will refer to screen sizes for simplicity sake.

```html
<div class="container">
  <div class="row">
    <div class="col-md-2">
      ...
    </div>
    <div class="col-md-3">
      ...
    </div>
    <div class="col-md-7">
      ...
    </div>
  </div>
</div>
```

Figure 2: Three columns layout example.

Looking at the mockup proposed in the previous tutorial, several rows and columns can be identified. Figure 3 presents the proposed organisation of the layout.

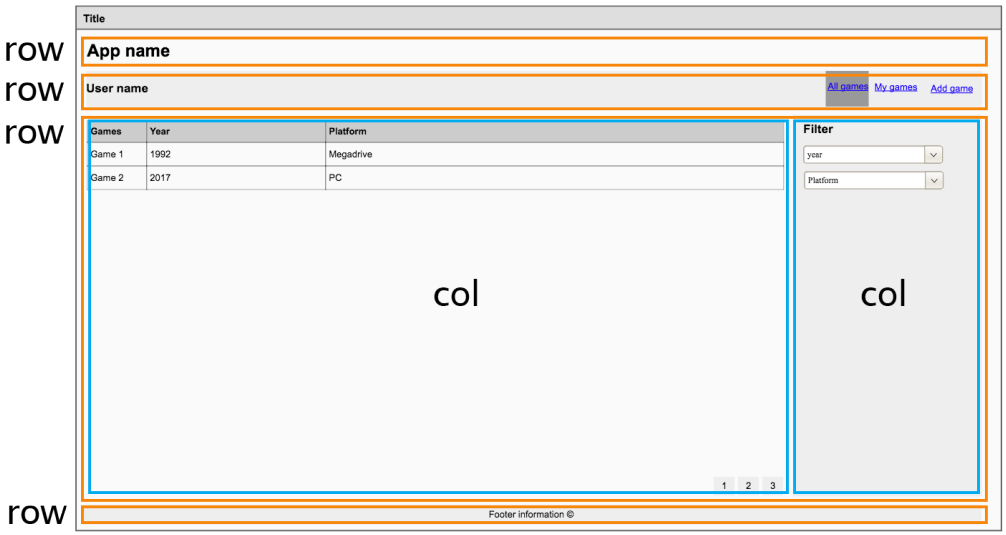

Figure 3: Rows and columns of the mockup.

Tasks

1. Using Bootstrap's grid system, create the basic page structure as illustrated in Figure 3. Adjust the size of the columns to match the mockup's proportion (e.g. start with 8 + 4). Do not add the content just yet!

The expected result is a page similar to the presented in Figure 4.

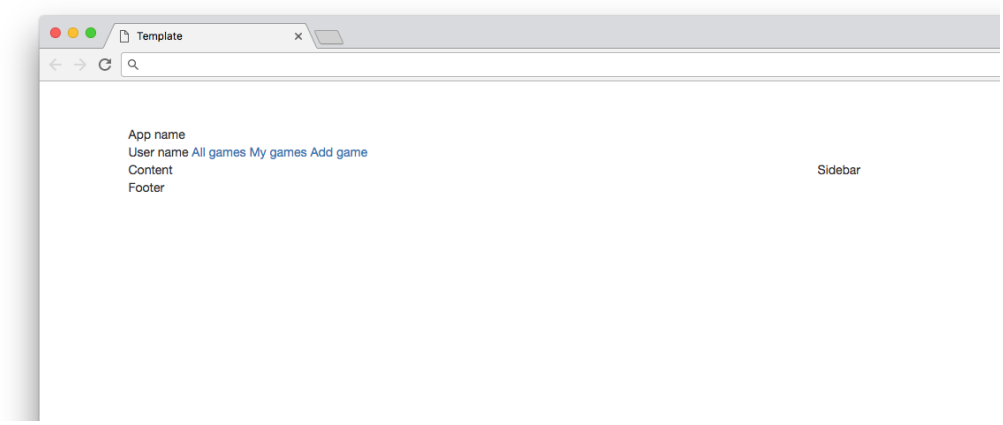

Figure 4: Page in the grid system.

4 Content placement

Now that the page structure is created, the page content can be added in order to match the mockup. Since we are creating a page in HTML, the elements have the usual syntax (e.g. `h1` , `select` , `input`).

Tasks

During the tasks below, reload the page as you make changes, and resize the browser, in order to view how the page reacts to changes.

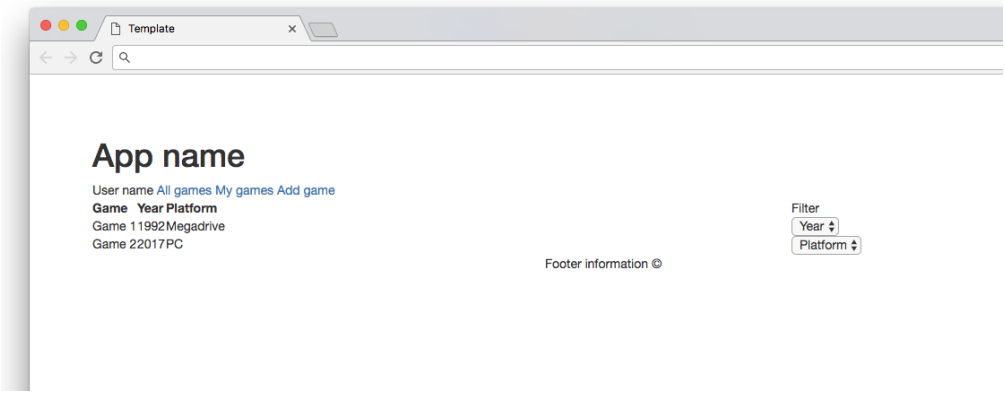

Figure 5: Page and content.

2. Align the footer content to the center.

   Bootstrap provides standard means to align content. Explore the text alignment classes[4] in order to understand the available options.

3. Place the table and filter form on the page. The expected result is a page similar to the one presented in Figure 5.

5 Bootstrap components

Another of the advantages of using Bootstrap, is the set of predefined components[5] available to create webpages. In the case of the presented mockup, it is possible to make use of the **Nav** or **Navbar** components. These components support the definition of the navigation bar, which eases the implementation of the page itself.

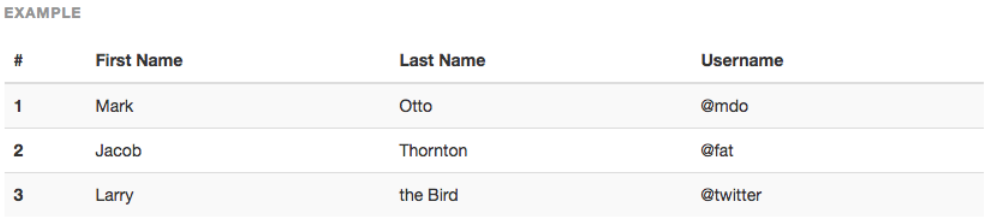

Figure 6: Bootstrap table example.

Another relevant component is **Forms**[6]. Bootstrap provides several powerful means to create interesting forms. An example is presented in Figure 7.

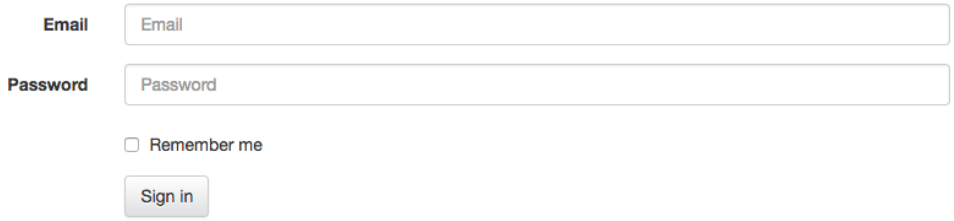

Figure 7: A Bootstrap form.

Finally, tables are not a component but can be created resorting to predefined classes. These classes provide several features, as is the case of highlighting odd lines, c.f. Figure 6.

[4] `https://getbootstrap.com/docs/4.3/utilities/text/`
[5] List of Bootstrap components: `http://getbootstrap.com/components/`
[6] `https://getbootstrap.com/docs/4.3/components/forms/`

Tasks

In order to perform these tasks, you may either copy&paste the elements from the provided URLs, and adjust to your needs, or create the elements manually. Either way, resorting to Bootstrap components, do the following:

4. Add a navigation bar.

   Explore the `Nav` component docs[7], using the provided examples to your needs (see also the the `active` and `d-flex` [8] classes). Alternatively, explore the `Navbar` component[9], which is more powerful and flexible.

5. Format the content table and add the pagination links at the bottom.

   Explore the table related classes[10] and the Pagination[11] component. To add the pagination at the end of the table, in then right position, consider using flex layout.

6. Format the sidebar content as a form.

   Explore the Forms docs to do it.

The final result (depending on the added components), will be similar to the one presented in Figure 5

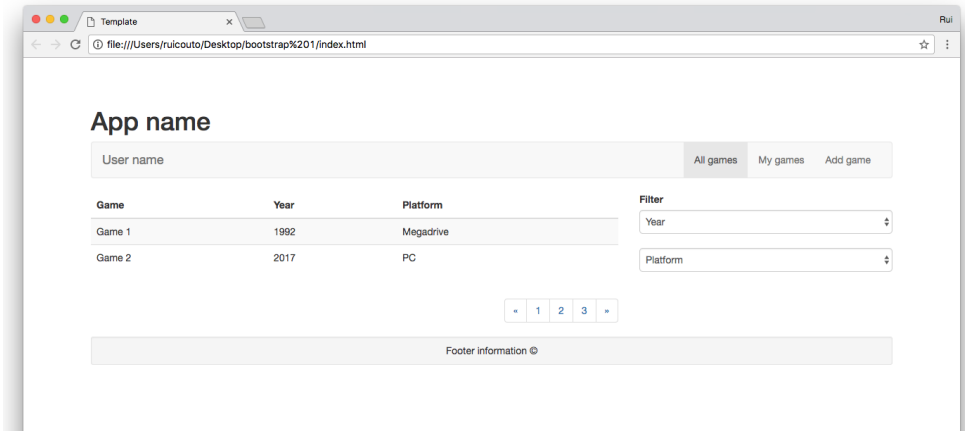

Figure 8: Resulting page.

[7] `https://getbootstrap.com/docs/4.3/components/navs/`
[8] `https://getbootstrap.com/docs/4.0/utilities/flex/`
[9] `https://getbootstrap.com/docs/4.3/components/navbar/`
[10] `https://getbootstrap.com/docs/4.3/content/tables/`
[11] `https://getbootstrap.com/docs/4.3/components/pagination/`

6 Semantic Markup with Bootstrap

One problem commonly mentioned about Bootstrap is that the separation between layout and contents is lost. Compare the size of your CSS file from the previous tutorial with the one from this tutorial. Where has all the layout and styling information go? It is in the HTML code! To mitigate this, it is possible to use Sass mixins to define rows and columns in CSS instead of in the HTML[12].

If you have completed the tutorial above, explore the use of these mixins to remove layout and styling markup from the HTML as much as possible.

[12] See `https://github.com/twbs/bootstrap-sass/blob/master/assets/stylesheets/bootstrap/mixins/`