**Chapter 7. Laying out pages with Flex Layout**

*This chapter covers*

* Implementing responsive web design using the Flex Layout library
* Using the ObservableMedia service
* Changing the layout based on the viewport size

When it comes to developing a web app, you need to decide whether you’ll have separate apps for desktop and mobile versions or reuse the same code on all devices. The former approach allows you to use the native controls on mobile devices so the UI looks more natural, but you need to maintain separate versions of the code for each app. The latter approach is to use a single code base and implement *responsive web design* (RWD) so the UI layout will adapt to the device screen size.

**NOTE**

Rkq mkrt YMK wsc deonci by Vngzr Weatcrto jn kry eiatlcr “Ynssvpoeei Mky Qgsein,” ilbaaaevl rs <http://alistapart.com/article/responsive-web-design>.

There’s a third approach: in addition to your web application that works on desktops, develop a *hybrid* application, which is a web application that works inside the mobile browser but can invoke the native API of the mobile device too.

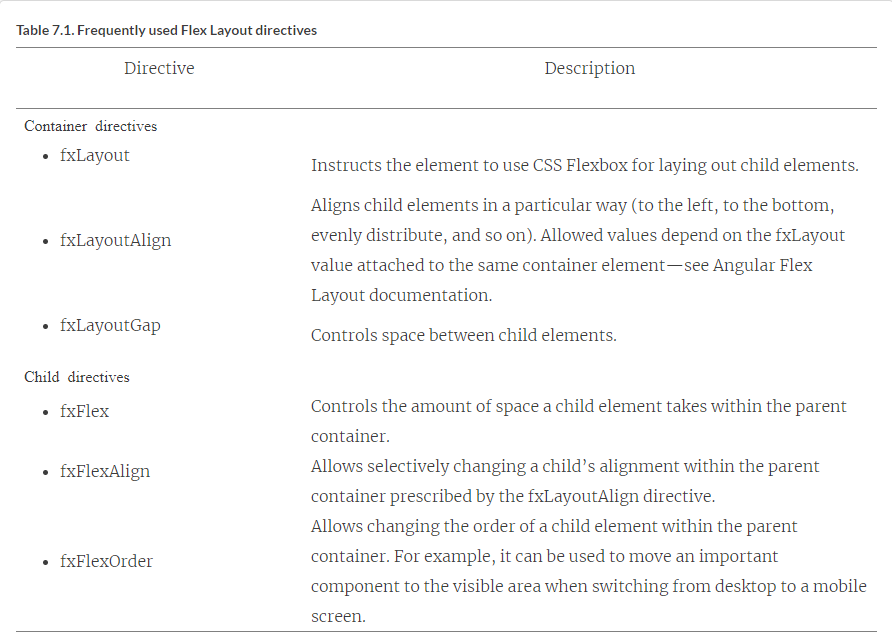
In this chapter, you’ll see how to make your app look good and be functional on large and small screens using the RWD approach. [Chapter 6](https://livebook.manning.com/book/angular-development-with-typescript-second-edition/chapter-7/5#!/book/angular-development-with-typescript-second-edition/chapter-6/ch06) covered observables that can push notifications when certain important events happen in your app. Let’s see if you can use observables to let you know if the user’s screen size changes and change the UI layout based on the width of the viewport of the user’s device. Users with smartphones and users with large monitors should see different layouts of the same app.

We’ll show you how to use the Flex Layout library for implementing RWD and how to use its ObservableMedia service to spare you from writing lots of CSS.

Finally, you’ll start rewriting the ngAuction app, illustrating many of the techniques you’ve learned, with the main goal to remove Bootstrap from the app, using only the Angular Material and Flex Layout libraries.

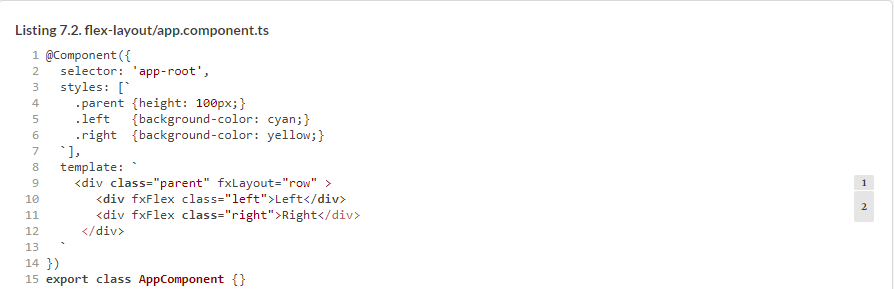
### 7.1. Flex Layout and ObservableMedia

#### 7.1.1. Using Flex Layout directives



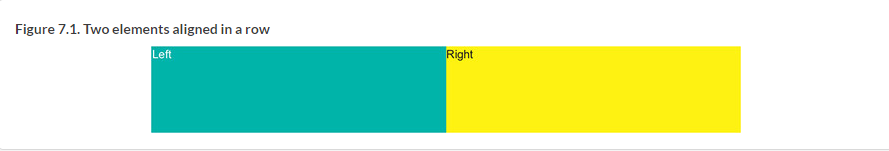


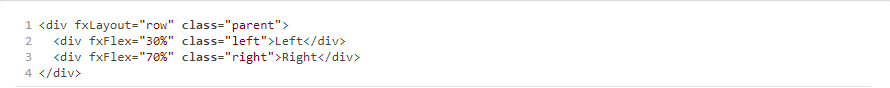




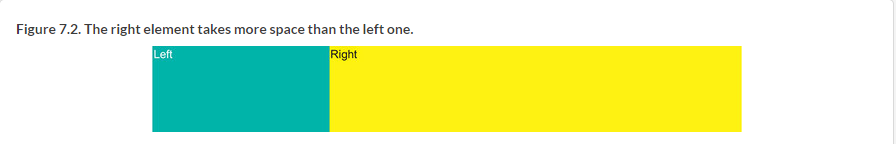
To see this application in action, run the following command:





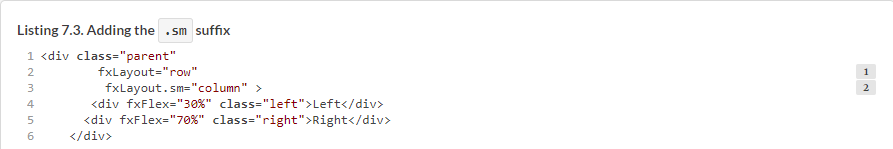


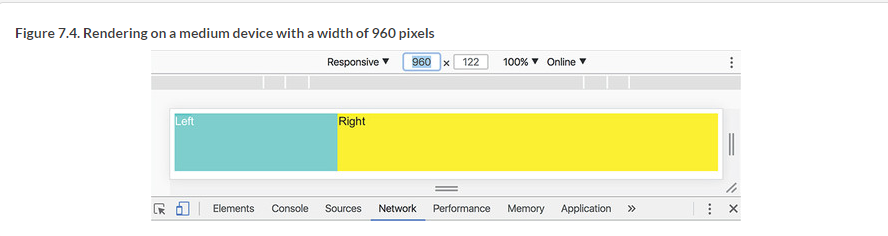
Now the UI is rendered as shown in [figure 7.2](https://livebook.manning.com/book/angular-development-with-typescript-second-edition/chapter-7/ch07fig02).

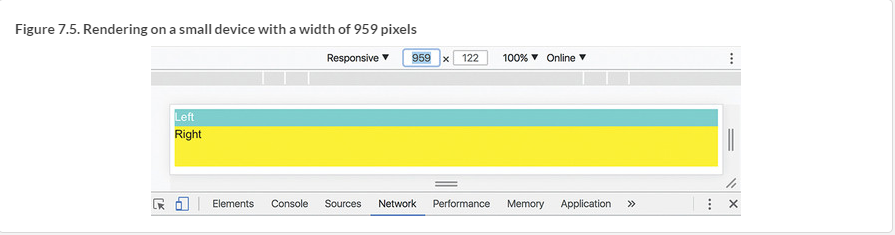
















#### 7.1.2. ObservableMedia service

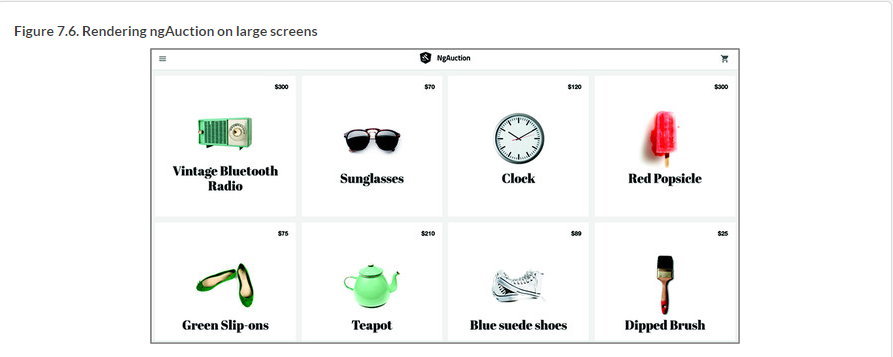


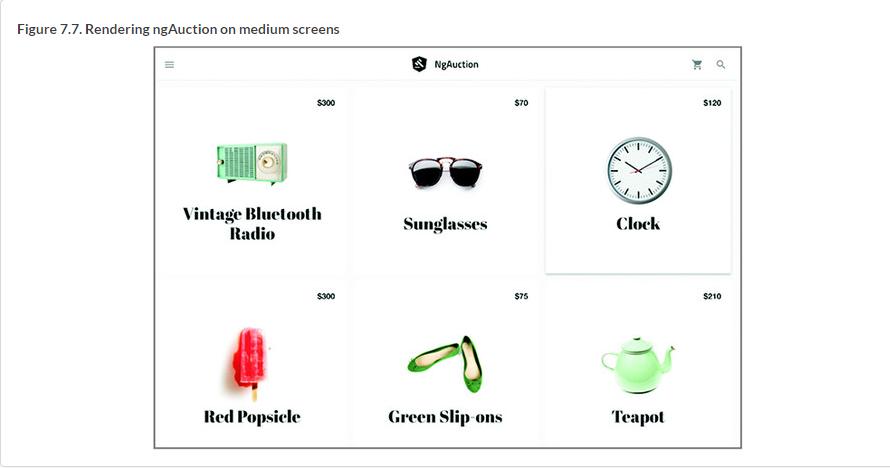


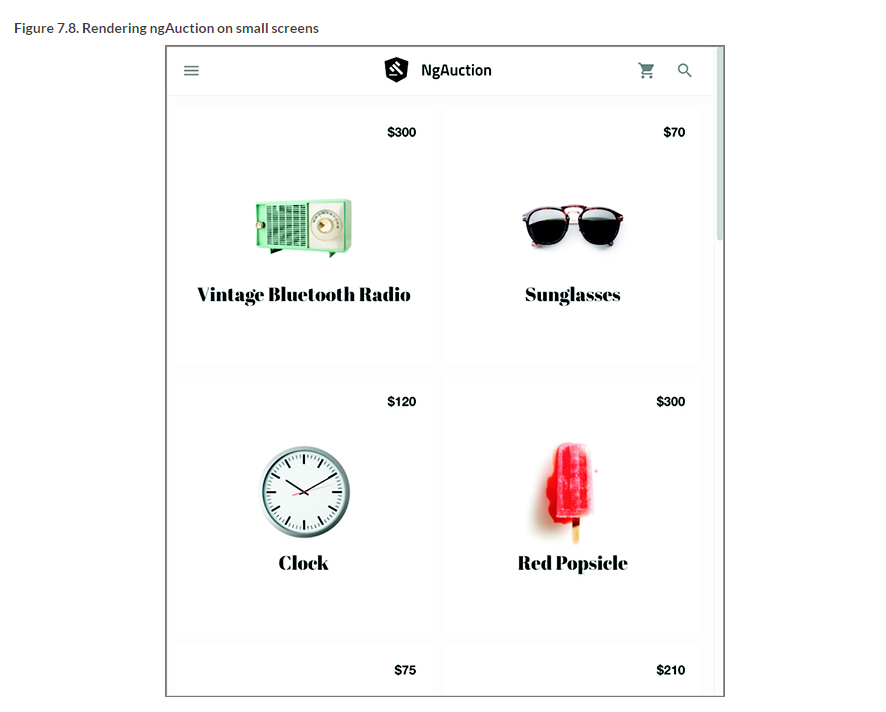


### 7.2. Hands-on: Rewriting ngAuction

#### 7.2.1. Why rewrite the ngAuction app from scratch?



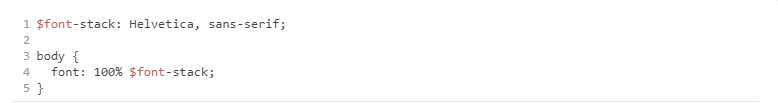




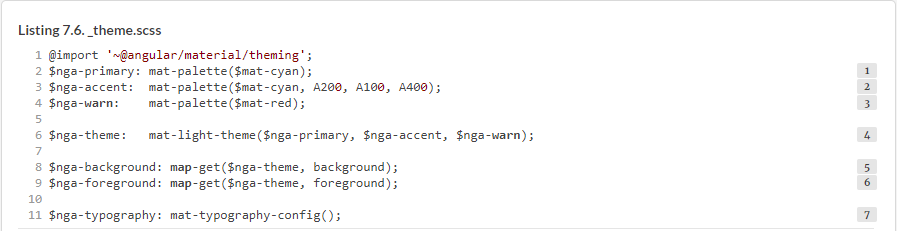
#### 7.2.2. Generating a new ngAuction app



#### 7.2.3. Creating a custom Angular Material theme with Sass





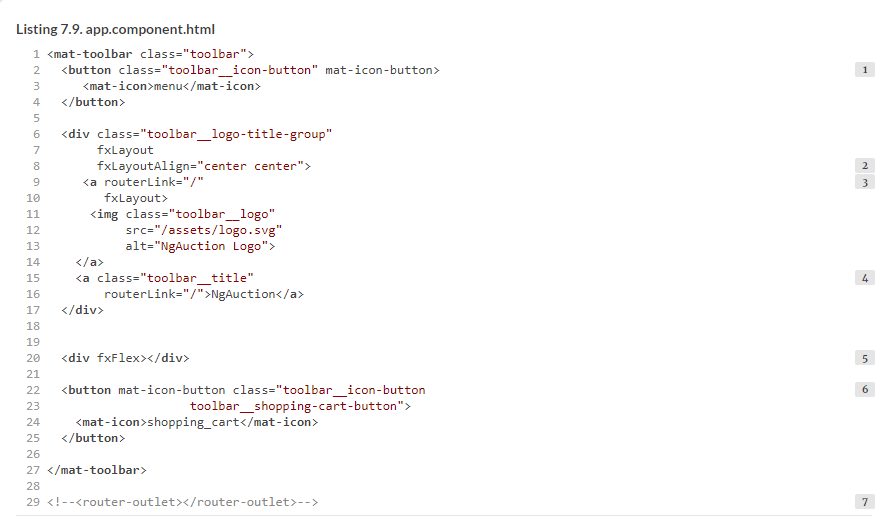




#### 7.2.4. Adding a toolbar to the top-level component

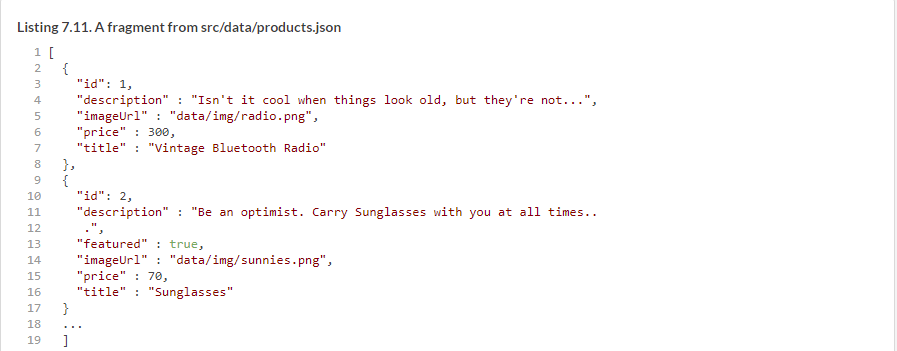






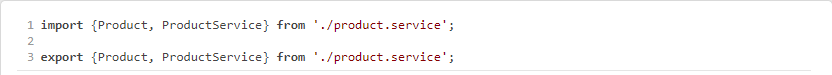


#### 7.2.5. Creating the product service







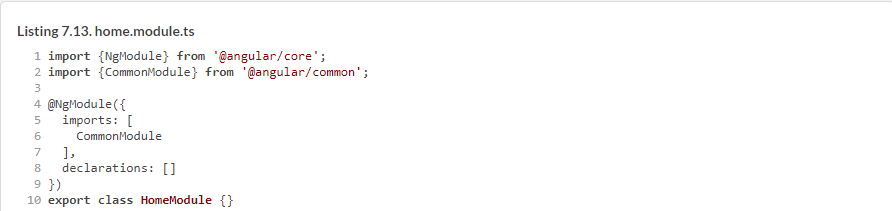








#### 7.2.6. Creating the home module



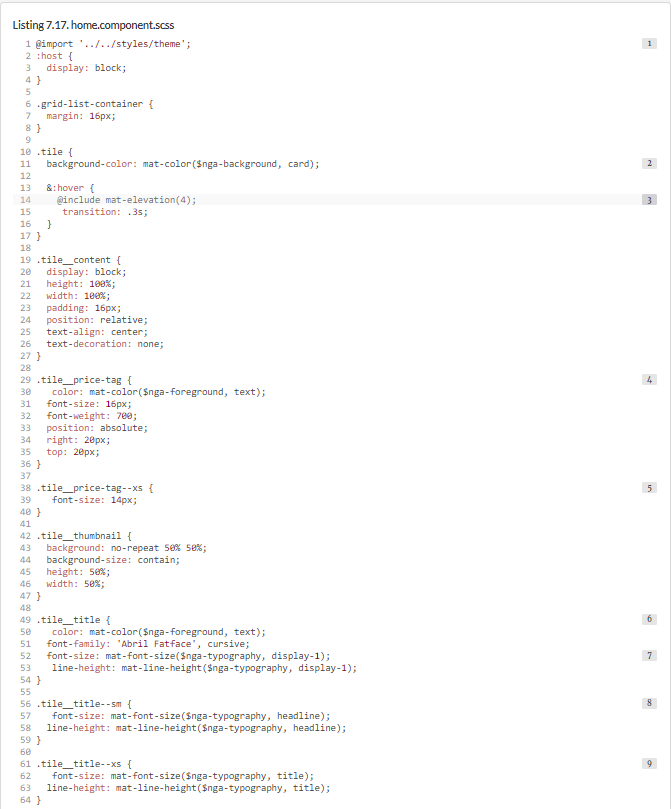






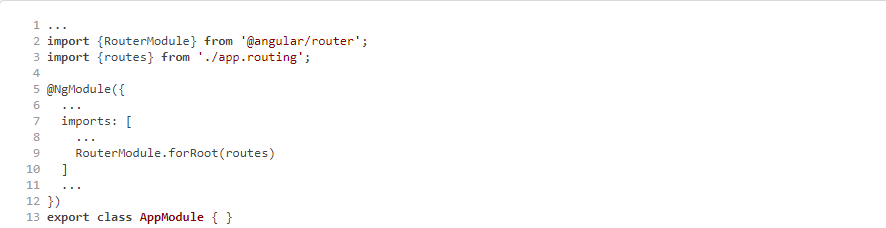






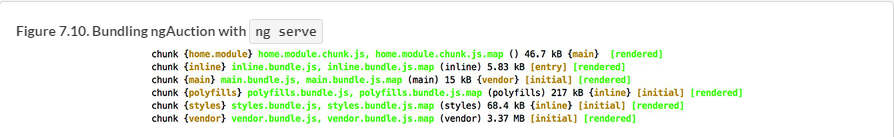
#### 7.2.7. Configuring routes

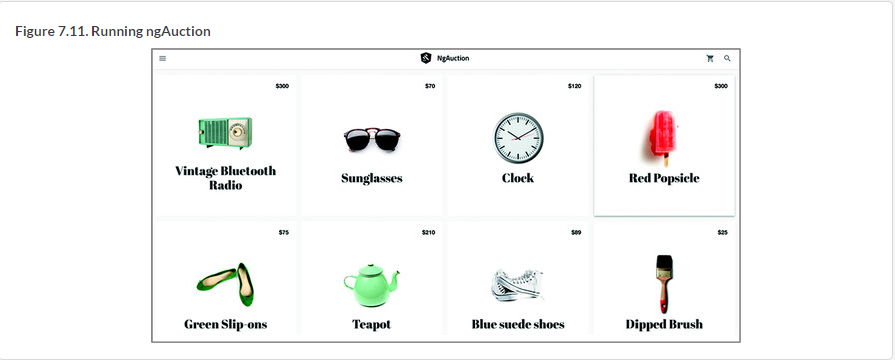






#### 7.2.8. Running ngAuction





### Summary