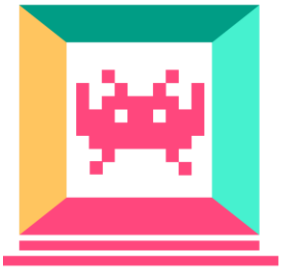

The Complete Beginner's Guide to React

By Kristen Dyrr

Software Engineer and Web Developer

Table of Contents

Chapter 1: Beginner's Guide to React.js, With Examples

React.js is a JavaScript library that was created by Facebook. It is often thought of as the "view" in a model-view-controller (MVC) user interface. This makes sense when you consider the fact that the only function that must be implemented in React is the "render" function. The render function provides the output that the user sees (the "view").

Let's take a look at why you may want to use React and how to set up a basic interface.

Download the source code

You can download all of the files associated with this tutorial from here.

Learn React online

If you are keen to learn React from the ground-up feel free to check Learn and Understand React JS on Zenva Academy which covers all the basics + lots of bonus topics like React Router and Flux.

Tutorial requirements

- This tutorial assumes that you have at least a beginner's grasp of HTML and JavaScript.
- You will need to download the React library if you want to go beyond the testing phase. We show you how to get around this during testing.
- You will need a text editor of some sort. Notepad++ is popular for those on Windows machines, and TextMate is popular on a Mac. Editors that highlight code are preferable.
- Normally, you would incorporate React into a larger application. If you want to test the basic code with external data files at the end of the tutorial, you will need to use a local or remote web server to get the page to work. MAMP is popular on Mac, and WAMP is most common on Windows machines. You can also use a lightweight Mongoose web server, or Python's HTTP server. Many people use React with Node.js, so you can also use a Node.js server. The React library download page (above) also includes a server and other options.

Downloading React and getting started

There are many options for downloading and installing React and its various add-ons, but the fastest way to get started and begin playing around with React is to simply serve the JavaScript files directly from the CDN (as described on the React GitHub page… the most common CDN options are listed there):

```
<!-- The core React library -->
<script src="https://fb.me/react-0.14.1.js"></script>
<!-- The ReactDOM Library -->
<script src="https://fb.me/react-dom-0.14.1.js"></script>
```

Both the download pages go into detail on the various ways to download, install, and serve React in various formats, but we're going to stick with this most basic option so we can focus on learning how to code with the React library itself. It's a good idea to have the React API open while you work for reference.

From there, we create an index.html file, and a main.js file. I've also included a css file for basic styling:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Learn Game Development at ZENVA.com</title>
    <!-- Just for basic styling. -->
    <link rel="stylesheet" href="assets/css/base.css" />
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.0/react.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react/0.14.0/react-dom.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.23/browser.min.js"></script>
  </head>
  <body>
    <div id="content"></div>
    <script type="text/babel" src="main.js"></script>
  </body>
</html>
```

In order to get around using a server while testing, I'm calling the React.js, react-dom.js, and the browser.min.js babel-core files from the CDN. You wouldn't want to do this in production. The babel-core file allows us to use JSX, and the script type must be "text/babel" in order for it to work properly. Our main JavaScript code goes in main.js, and that's where it all starts.

Why React is better with JSX

Most React implementations make use of JSX, which allows you to put XML-like syntax right within JavaScript. Since React displays output as it's main function, we will be using HTML in just about every component. JSX simplifies the code that would normally have to be written in JavaScript, which makes your code much more readable and simplified.

JSX is not required, but consider the difference between two very simple statements. The following statement is created without JSX:

```
var element = React.createElement('div', { className: 'whatever' }, 'Some text');
```

The following is with JSX:

```
var element = <div className="whatever">
   Some text
</div>
```

As you can see, the JSX code is much easier to read. Now that we have a basic understanding of what our output syntax will look like, let's dig in and learn the building blocks of React.

Understanding React components

React is based on components and states. This is what makes React such a popular library. When you want to create an application, you usually break it into simpler parts. When programming with React, you will want to break your interface into its most basic parts, and those will be your React components.

Components are wonderful because they are modular and reusable. You can take a basic component used in one area of an application and reuse it in others without having to duplicate code. This helps to speed up development.

Components can be nested, so that the most basic components can be grouped into a parent component. For example, if you were to create a home listing interface with React, the top level component would be the home list itself. Within the list, you would have a description of a single home. Within the home component, you would have the address of the home, as well as other small components such as a photo, perhaps a favorite or save button, and a link to view details and a map.