

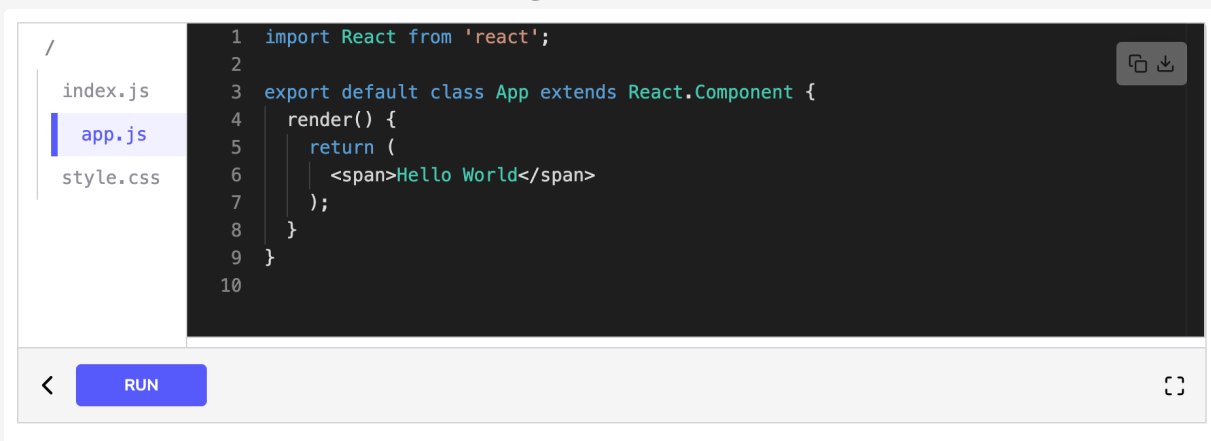
# Requirements

This lesson will help you learn the pre-requisites prior to learning React.

To follow this course you'll need to be familiar with the basics of web development, i.e., how to use HTML, CSS, and JavaScript. It also helps to understand [APIs](#), as they will be covered thoroughly.

## Editor and Terminal

For the entire course, you do not need a IDE on your local machine, you can use the **Educative SPA widget** to see the live execution of codes.

A screenshot of the Educative SPA widget interface. On the left, there is a file explorer showing three files: 'index.js', 'app.js' (which is selected and highlighted with a blue bar), and 'style.css'. The main area is a code editor with a dark background, displaying the following JavaScript code:

```
1 import React from 'react';
2
3 export default class App extends React.Component {
4   render() {
5     return (
6       <span>Hello World</span>
7     );
8   }
9 }
10
```

At the bottom of the widget, there is a light gray bar containing a left-pointing arrow, a blue button labeled 'RUN', and a right-pointing arrow.

While Educative SPA widget is a great tool for sharing code online, local machine setup is a better tool for learning different ways to create a web application. Also, if you want to develop applications professionally, a local setup will be required.

## Set-up on local machine

I have provided [a setup guide](#) to get you started with general web development. For this learning experience, you will need a text editor (e.g., Sublime Text) and a command-line tool (e.g., iTerm) or an IDE (e.g. Visual Studio Code). I recommend Visual Studio Code (also called VS Code) for beginners, as it's an all-in-one solution that provides an advanced editor with an integrated command-line tool, and because it's a popular choice among

web developers.

Throughout this learning experience I will use the term *command-line*, which will be used synonymously for the terms *command-line tool*, *terminal*, and *integrated terminal*. The same applies to the terms *editor*, *text editor*, and *IDE*, depending on what you decided to use for your setup.

Optionally, I recommend managing projects in GitHub while we conduct the exercises in this course, and I've provided a [short guide](#) on how to use these tools. Github has excellent version control, so you can see what changes were made if you make a mistake or just want a more direct way to follow along. It's also a great way to share your code later with other people.

## Node and NPM

Before we can begin, we'll need to have [node and npm](#) installed. Both are used to manage libraries (node packages) you will need along the way. These node packages can be libraries or whole frameworks. We'll install external node packages via npm (node package manager).

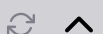
You can verify node and npm versions in the command line using the `node --version` command. If you don't receive output in the terminal indicating which version is installed, you need to install node and npm:

```
node --version
*vXX.YY.ZZ
npm --version
*vXX.YY.ZZ
```



You can check the version of node and npm by typing the commands above in the terminal below:

● Terminal



If you have already installed Node and npm, make sure that your installation is the most recent version. If you're new to npm or need a refresher, this [npm crash course](#) I created will get you up to speed.

---

The next lesson will help you set up a react project.

