Hyperiondev

TASK

React - Overview

Visit our website

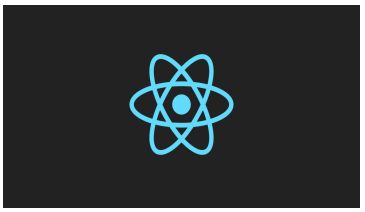
Introduction

WELCOME TO THE REACT - OVERVIEW TASK!

Web developers often use web development frameworks and libraries to create web applications more quickly and efficiently. React.js or ReactJS (commonly known as React) is a JavaScript library for building front-end web applications.

In this task, you will learn what React is and how to use a React starter kit to start developing a React application with minimal configuration. You will also learn to create and render elements using React.

WHAT IS REACT?



React is used to generate user interfaces (UIs). All web UIs must ultimately be converted to HTML, as web browsers render HTML. So why can't we just use HTML and CSS to create user interfaces for the web? We could, but HTML and CSS alone can only display static pages.

If we only used static pages, we cannot implement logic (validating user input, looping, performing calculations etc.) with HTML. JavaScript was developed to make web pages dynamic, and this is where React comes in. With React, we *use JavaScript to write HTML*!

React is a JavaScript library for building UIs (note that React is a library, not a framework). It was created by **Jordan Walke**, a software engineer at Meta (formerly known as Facebook). It was first deployed on Facebook's Feed (formerly known as Facebook's News Feed) in 2011 and later on Instagram in 2012. The initial public release was on 29 May 2013. Meta and a community of developers and organisations maintain React.

With React, we use JavaScript to describe what we want our UI to look like, and React makes it happen. In other words, React is declarative. This means we tell React **what** we want to do, **not how** to do it.

A note from the
HyperionDev Team

Why use React over other libraries or frameworks?

React is one of the most popular libraries for front-end web development. Vue.js and Angular are, however, also popular. This **blog post by HyperionDev** compares React, Vue, and Angular.

Before we get started with React, there are a few concepts that we need to understand. Each of these core concepts is discussed under the subsequent headings in this task. Some code examples used in this task are from **React's official documentation**.

COMMON REACT TERMS

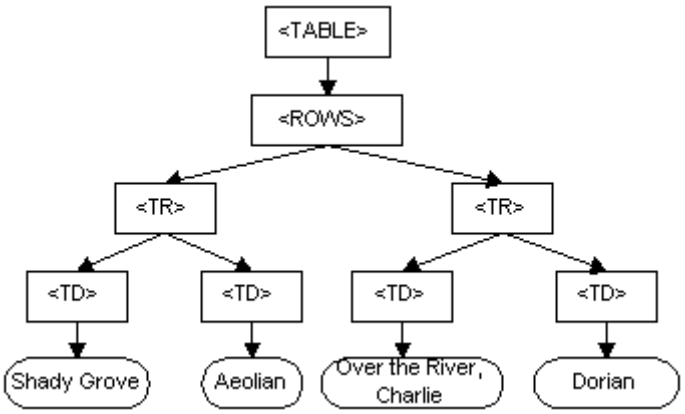
JSX: stands for JavaScript XML. JSX is a JavaScript Extension Syntax used in React to write HTML and JavaScript together easily.
React Elements:: the building blocks of React applications.
React Components:: small, reusable pieces of code that return a React element to be rendered to the page.
Props:: inputs to a React component. They are data passed down from a parent component to a child component.
State:: an object that holds data and information related to a React component. It can be used to store, manage, and update data within the application.
Hooks: these let you use different React features from your components. There are several types of hooks in modern React, including but not limited to:

- **State Hooks:** State lets a component "remember" information like user input.
- **Context Hooks:** Context lets a component receive information from distant parents without passing it as props..
- **Ref Hooks:** Refs let a component hold some information that isn't used for rendering.
- **Effect Hooks:** these let a component connect to and synchronise with external systems. This includes dealing with a network, browser DOM,

animations, widgets written using a different UI library, and other non-React code.

THE VIRTUAL DOM

The image below, from the **World Wide Web Consortium** (W3C), visualises a Document Object Model (DOM) for an HTML table. The DOM is a programming API for HTML and XML documents that defines the logical structure of documents and how they are accessed and manipulated.



DOM representation of an HTML table. (**Source**)

As web developers, we use the DOM to manipulate HTML documents. We can add, delete, or even change HTML elements using the DOM; in fact, you have already been doing so by making use of JavaScript! For example:

```
let div = document.getElementById("container");
let imgProfile = document.createElement("img");
imgProfile.src = "./pictures/profilePic.png";
div.appendChild(imgProfile);
let paragraph = document.createElement("p");
paragraph.innerHTML = "Dynamically adding a paragraph to HTML";
div.appendChild(paragraph);
```

However, rewriting the DOM every time the HTML document changes can significantly slow down your web application or site. To address this problem, React uses a virtual DOM.

A virtual DOM is a virtual representation of the HTML document in memory. React works by taking a *snapshot* of the DOM before changes are made. As changes are made, the virtual DOM is updated. After the changes are complete, the virtual DOM and the snapshot of the DOM (which is also saved in memory) are compared

to see where changes were made. Instead of rewriting the entire DOM, the real DOM is only updated with the changes that were made.

REACT STARTER KITS

React relies heavily on other libraries to function properly. **Babel** is a JavaScript compiler used for writing next-generation JavaScript. It is particularly relevant to developers who work with React because it can convert JSX syntax into a backward-compatible version of JavaScript that can operate on earlier browsers. Babel's goal is to ensure that developers can use the most recent JavaScript features without worrying about compatibility concerns with older browsers.

React works best when used with **Webpack**. When you create apps using React, many modules and other resources, like images and CSS files, are created. Webpack is a module bundler for modern JavaScript applications. It creates a dependency graph that is used to handle all the resources you need for your app.

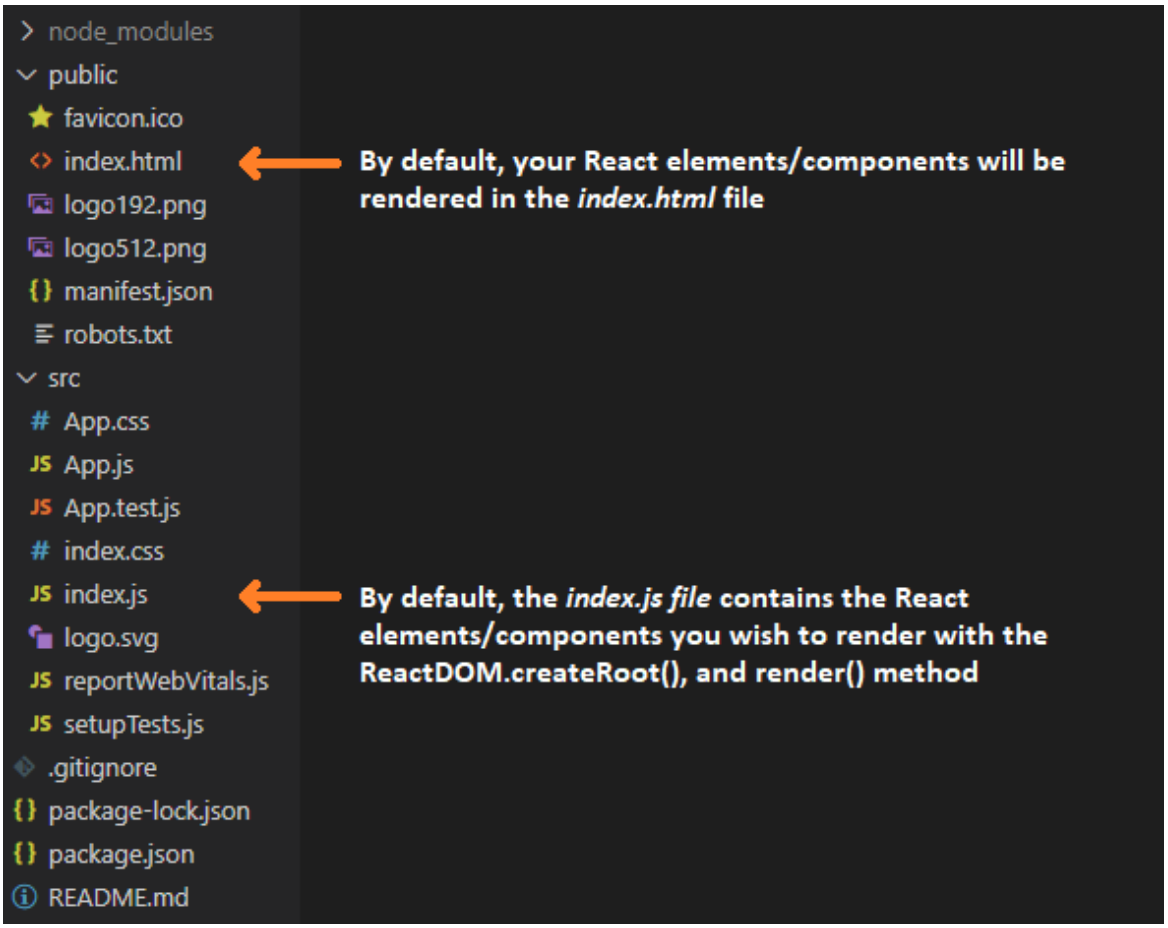
Extra resource

The intricate details and inner workings surrounding Webpack are beyond the scope of this Bootcamp.

Learn more about Webpack using the **Webpack documentation**.

Although you could manually install and configure React and all its dependent libraries separately, Meta (the creators of React), recommend that you make use of a starter kit instead. A starter kit automatically instals all the libraries and modules needed to create a basic React app. It also does some basic configuration so that you are able to start creating a React app immediately. Although there are several starter kits available for you to use, in this Bootcamp we will be using one of the official starter kits recommended by Meta, namely **Create React App** (CRA), which is used to create single-page applications.

When you create a React app using CRA, a project folder with a specific directory structure will be created, for example:



By default, the React elements you create will be rendered in the **index.js** file that has been created for you by CRA. To find this file, open the project directory that was created with CRA. In this directory, you will see a directory called **src** which contains the file **index.js** file. In this file, you will see an instruction to render (`root.render()`) any React elements or components, as shown in the example below:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
     <App />
  </React.StrictMode>
```

```
 );
 // If you want to start measuring performance in your app, pass a
function
 // to log results (for example: reportWebVitals(console.log))
 // or send to an analytics endpoint. Learn more:
https://bit.ly/CRA-vitals
  reportWebVitals();
```

To understand where the React element or component you created will be rendered, open the **public** directory. You should find a file called **index.html**. This HTML file contains a **`div`** HTML element that has an **`id`** attribute with a value of **`'root'`** assigned to it. When the render() method is executed within the **index.js** file, we are specifying that the React element/component you create will become a child of this **`div`** (within the **index.html** file).

A note from the
HyperionDev Team

Are you unable to create a React app with CRA?

As you are aware, technology constantly improves, therefore existing software also needs to improve. The instructions as per **CRA's official documentation** are valid and correct, however, if you experience errors while trying to use CRA, one of the main causes could be that the version of NPM on your local machine is trying to install an earlier (older) version of CRA. You can force NPM to install the latest version of CRA, by making use of the **`@latest`** flag, for example: **`npx create-react-app@latest my-app`**.

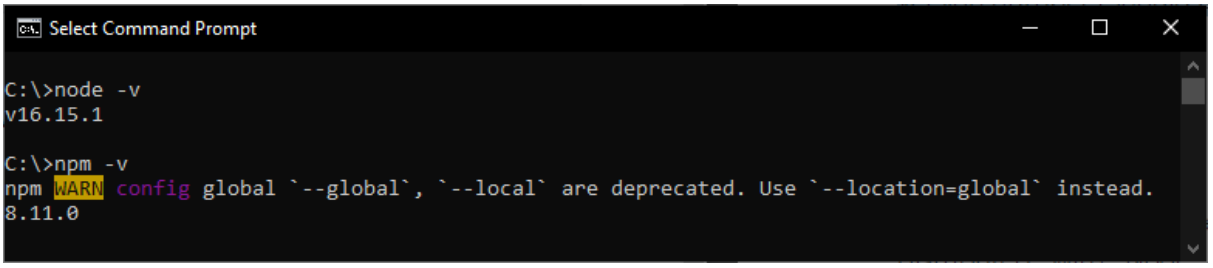
DEBUGGING A REACT APP

The developers of React have created various tools to assist with debugging React apps. Access the **React Documentation** and the official **React Developer Tools** for information and tools for debugging your apps.

Alternatively, you can install additional browser extensions to help with debugging a React App. **Firefox** and **Chrome** have browser extensions available for debugging. You can also use **CodeSandbox** for simple debugging.

RUN REACT ON YOUR LOCAL MACHINE

To use React on your local machine, **Node** and **NPM** need to be installed. To see if you already have Node and NPM installed, open the command line interface or terminal and try to see which versions of Node and NPM are installed. If correctly installed, the version numbers should be displayed, for example:



Take note!

Setting up the example for this task

As mentioned before; React relies heavily on other libraries to function properly, most of which are contained within the **node_modules** folder (which contains *a lot* of files). To run the React example on your local machine, we need the **node_modules** folder.

Please read through, and execute, the instructions contained within the **README.md** file, which can be found in the root folder of the example.

Compulsory Task 1

Follow these steps:

- Use **Create-React-App** to create a React Application.
- Create a new folder and open a command line terminal in the folder.
- To create a project called `my-app`, run this command in the command line in the empty folder we just created:

```
npx create-react-app my-app
```

- To run the application, navigate to the **my-app** folder where you created the application and run the following script:

```
npm run start
```

- Take a screenshot of the React Application running in your browser and upload it to the Dropbox task folder.

- Once you are ready to have your code reviewed, delete the node_modules folder. Please note that this folder typically contains hundreds of files which have the potential to slow down Dropbox sync and possibly your computer if you're working directly from your Dropbox folder. Therefore, you may prefer to work in a different folder on your local machine and move it to Dropbox when you have completed the lab.

- Please compress (e.g. .zip) your project folder before you submit it via the relevant task folder in Dropbox.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

CodeSandbox (2022). Documentation - CodeSandbox Documentation. Retrieved on 26 June 2022, from **https://codesandbox.io/docs**

CodeSandbox (2022). React - CodeSandbox. Retrieved on 26 June 2022, from **https://codesandbox.io/s/new**

Create React App (2022). Adding a Stylesheet | Create React App. Retrieved on 26 June 2022 from **https://create-react-app.dev/docs/adding-a-stylesheet/**

React.js. (2023). Quick Start – React. Retrieved on 11 July 2023, from **https://react.dev/learn**

React.js (2022). Team - React. Retrieved on 11 July 2023, from https://react.dev/community/team

React.js (2022). Glossary of React Terms - React. Retrieved on 26 June 2022, from **https://reactjs.org/docs/glossary.html** (The reactjs.org website has been depreciated, but the glossary is useful for looking up terms)

React.js (2022). Strict Mode - React. Retrieved on 11 July 2023, from **https://react.dev/reference/react/StrictMode**

React.js (2022). ReactDOMClient - React. Retrieved on 11 July 2023, from **https://react.dev/reference/react-dom/client**

React.js (2022). Create a New React App – React. Retrieved on 11 July 2023, from **https://react.dev/learn/start-a-new-react-project#create-react-app**

React.js (2022). react/packages/react-devtools at main · facebook/react · GitHub. Retrieved on 26 June 2022, from **https://github.com/facebook/react/tree/main/packages/react-devtools**

React.js logo (2022). Retrieved on 26 June 2022, from **https://reactjs.org**

React-Bootstrap (2022). React-Bootstrap · React-Bootstrap Documentation. Retrieved on 26

June 2022 from **https://react-bootstrap.netlify.app/**

Reactstrap (2022). Home/Installation - Page · Reactstrap. Retrieved on 26 June 2022 from **https://reactstrap.github.io/**

MDN (2022). Expressions and operators - JavaScript | MDN. Retrieved on 26 June 2022, from **https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators#expressions**

MDN (2022). Strings - JavaScript | MDN. Retrieved on 26 June 2022, from **https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String**

MDN (2022). Document.getElementById() - Web APIs | MDN. Retrieved on 26 June 2022, from **https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById**

Webpack (2022). Concepts | Webpack. Retrieved on 26 June 2022 from **https://webpack.js.org/concepts/**