Hyperiondev

React I: Introduction to React

Visit our website

Introduction

WELCOME TO THE INTRODUCTION TO REACT TASK!

As you saw in previous tasks, web developers often make use of web development frameworks and libraries to create web applications more quickly and efficiently. React is such a JavaScript library for building front-end web applications. In this task, you will learn more about what React is and how to use a React starter kit to start React development with minimal configuration. You will also learn to create and render elements using React.

Get in touch
Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

WHAT IS REACT?

React is used to generate user interfaces (UIs). We already know that all browsers render HTML and that all web UIs ultimately need to be converted to HTML. So why can't we just use HTML and CSS to create user interfaces for the web? We can, but HTML and CSS alone can only display static pages. As you know, you can't implement logic (testing conditions, looping, etc.) with HTML. That is why we started using JavaScript in the first place. JavaScript can be used to make web pages dynamic. This is where React comes in. With React, *we use JavaScript to write HTML*.
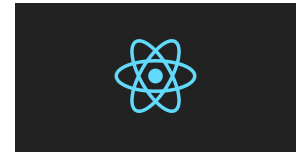
React is a JavaScript library for building UIs. React is a library, not a framework. With React, we use JavaScript to describe what we want our UI to look like and React makes it happen. In other words, React is declarative. This means we tell React what we want to do, not how to do it.

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 of the code examples that will be used in this task are from React's official documentation, which can be found **here**.

A note from the

HyperionDev Team

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

THE VIRTUAL DOM

The image below, from the **World Wide Web Consortium**, visualises a Document Object Model (DOM) for an HTML table. As you already know, the DOM is a programming API for HTML and XML documents that defines the logical structure of documents and the way they are accessed and manipulated.
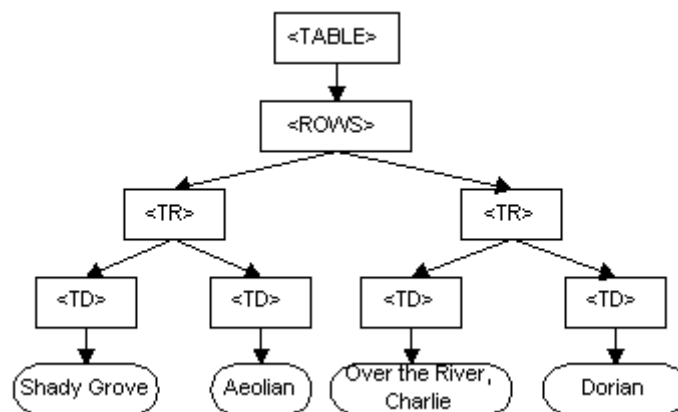
As web developers, we use the DOM to manipulate HTML documents. We can add, delete or change HTML elements using the DOM. You have already been doing this using JavaScript (`e.g.  let imgProfile = document.createElement("img"); div.appendChild(imgProfile);`). However, rewriting the DOM every time the HTML document changes can slow down your web app significantly. React uses a virtual DOM to address this problem. 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, the virtual DOM and the snapshot of the DOM before the changes (also saved in memory) are compared to see where the changes are. Instead of rewriting the entire DOM, the real DOM is then updated with only the changes that were made.

REACT ELEMENTS

React elements are similar to DOM elements (like DIV, BODY, HEAD). You should already be comfortable creating static HTML pages using various DOM elements. Like HTML, with React, you can use React elements to build user interfaces (UIs).

The **official definition** of React elements is as follows:

"React elements are the building blocks of React applications. An element describes what you want to see on the screen. React elements are immutable (they cannot be modified after they are created).

```
const element = <h1>Hello, world</h1>;
```

Typically, elements are not used directly, but get returned from components."

We will learn more about how to create components and the relationship between elements and components in the next task.

CREATING REACT ELEMENTS

There are 3 basic steps that you will use to create any React elements:

1. Create React and ReactDom objects by importing React and ReactDOM.
2. Create the React Element using either JavaScript or JSX.
3. Render the element that you have created to the DOM using the `render()` method.

Create React and ReactDom objects by importing React and ReactDOM

Before creating any React elements, you need to import the React library and the React-dom package. The react-dom package provides DOM-specific methods including the render method(). You import these as shown below.
Example A - Keep at the top

```
import React from "react";
import ReactDOM from "react-dom";
```

The ***import statement*** *may be new to you. It is used to import functions, objects or variables which are exported by another module. You will learn more about this later but for now, know that importing from React returns an object that we can use for creating React apps.*

Create the React Element using either JavaScript or JSX

There are various ways in which you can create a React element. You could simply declare a variable that stores HTML or JSX, or your variable could consist of more than one element.

JSX

JSX is a syntax extension for JavaScript that can make creating React applications a lot quicker and easier. JSX can be thought of as a mix of JavaScript and XML. Like XML, JSX tags have a tag name, attributes and children. With JSX, if an attribute

value is enclosed in quotes, it is a string and *if the value is wrapped in braces ({ }), it is an enclosed JavaScript expression.* An example of JSX can be seen below:

Example B: Add to Example A

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Dave',
  lastName: 'Hyperion',
  profileImg: './images/12345.gif'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Babel is a tool that can be used to convert JSX to JavaScript. Go to the **Babel REPL** website and copy and paste the code above into the 'Write code here' space provided so that you can see the difference between this React code (using JSX) and code written using just JavaScript.

Look again at the code example above. There are some rules that you should be aware of when using JSX:

- As with HTML, with JSX you can specify various elements that can have different attributes.
- With JSX you can use normal HTML elements like **div**, **img**, or **h1**. You can also create user-specified elements.
- With JSX, if an attribute value is enclosed in quotes, it is a string. If the value is wrapped in braces, it is an enclosed JavaScript expression. Attributes can be assigned using either:
  - String literals e.g. `<img src="./images/logo.gif" />`
    String literals should be enclosed by quotation marks, e.g. "./images/logo.gif"

- ○ Or JavaScript expressions, e.g. `<img src={user.profileImg} />` JavaScript expressions should always be enclosed by curly braces instead of quotation marks, e.g. `{user.profileImg}`

Your variable could consist of more than one element

You can do this if you wrap them in a parent element (`<div>` in the example below) and enclose the elements with a brackets as shown:

Example C: Add to Example A

```
const element = (<div>
  <h1>I'm learning React with Hyperion</h1>
  <img
src="http://hyperiondev.com/static/moocadmin/assets/img/hyperionLogo.png"/>
  </div>
);
ReactDOM.render(element, document.getElementById('root'));
```

Note: *Each element can only return one parent element. However, that element can contain many child elements.*

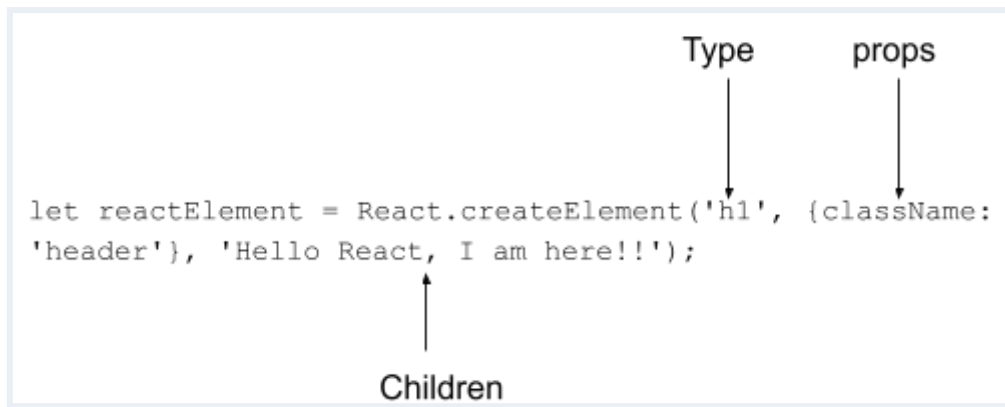You could use the `createElement()` method to create React elements. For example:

Example D: Add to Example A

```
const reactElement = React.createElement('h1', {}, 'Hello World!');
ReactDOM.render(reactElement, document.getElementById('root'));
```

The `createElement()` method takes three arguments:

- **Type:** this could be an HTML tag name (such as `div` or `h1`) or a ReactClass (which you will learn more about later). It describes what type of element you are creating.

- **Props:** props are properties that can be passed from parent to child elements. Props that can be passed include descriptions of HTML attributes (like class or style).

- **Children:** used to pass the child elements that this element should have. This could be a string, another React Element, or even an array of elements.

  For example:

```
                                              Type         props
                                               │            │
                                               ↓            ↓
let reactElement = React.createElement('h1', {className:
'header'}, 'Hello React, I am here!!');
                                  ↑
                                  │
                              Children
```

**Render the element that you have created to the DOM using the `render()` method**

Once you have created an element, you still have to render it so that it is visible to the user. Notice that each of the code examples above already contain a render method. For example:

```
ReactDOM.render(reactElement, document.getElementById('root'));
```

The first argument passed to the render method is the React Element which you have created. The second argument passed to the render() method is the DOM element to which you want to append your react elements.

Try this:
1. Open **CodeSandBox,** an online code editor that allows you to easily create and test basic React apps.
2. Make sure that you have **index.js** selected.
3. Delete everything in **index.js** except for the first two import statements. Remember, you'll need these import statements every time you create React elements. If one of the statements (you can see them in the example below) is missing, add it in.
4. Now copy and paste each of the code examples that we have considered above (code examples a, b, c and d) into CodeSandBox one at a time. For each element, study the code and the output in CodeSandBox. Be sure you understand the code in each code example.
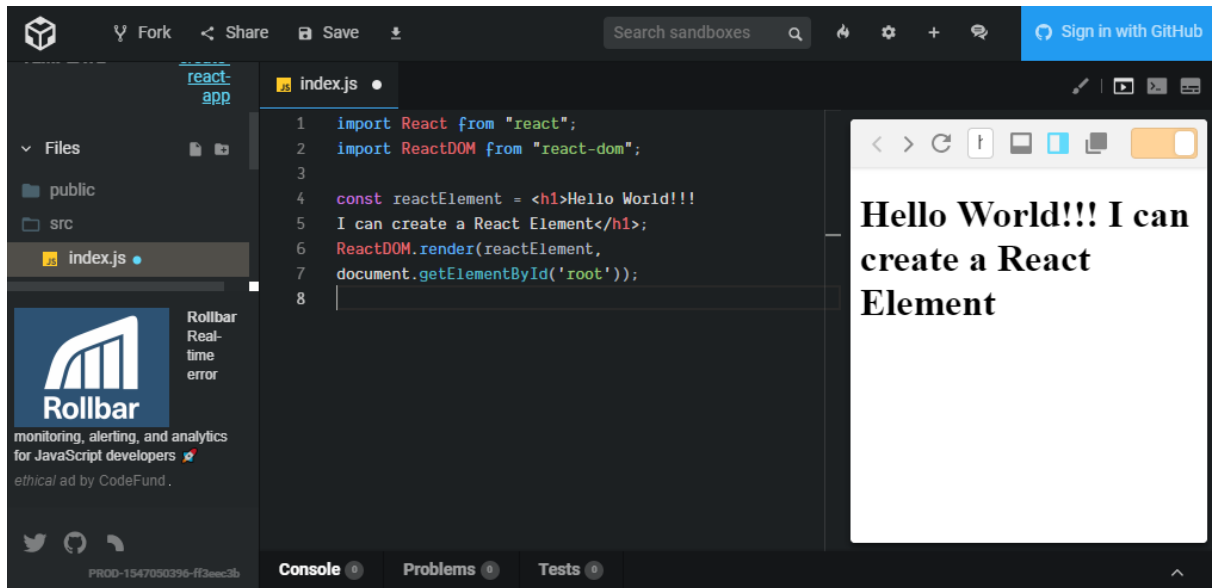
SPOT CHECK 1

Let's see what you can remember from this section.

1. What does React allow us to do?

2. What are the three steps of creating React elements?

STYLING REACT ELEMENTS

As you have already learned, using CSS effectively to create attractive UIs is an indispensable skill. It is easy to style elements with React. As with creating HTML pages, you can create your own custom stylesheets. To apply the rules in the stylesheet to the elements, simply import the .css file to the JavaScript file where you are creating the element. E.g. `import './css/custom.css';`. See **here** for more information about stylesheets with React.

We also don't have to start styling from scratch when creating React components. There are several CSS libraries that can make creating components quicker and easier by reusing and modifying existing code. Let's consider just two of these:

- **React-Bootstrap**: React-Bootstrap is a complete re-implementation of the Bootstrap components using React.
  Using React Bootstrap is similar to using normal Bootstrap. To use React-Bootstrap:
  1. Install React-Bootstrap using NPM. Navigate to the project directory for your react app and type: npm install react-bootstrap bootstrap

2. Import the React-Bootstrap component you want to use. E.g. if you want to use the React-Bootstrap Button component you would need to add the following line of code: import Button from 'react-bootstrap/Button';

3. Include the latest styles from the React-Bootstrap CDN. Get the link from the official **React-Bootstrap website**. It should look something like:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/css/
bootstrap.min.css"
integrity="sha384-GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMD
m4iYMj70gZWKYbI706tWS" crossorigin="anonymous"/>
```

- **Reactstrap**: Reactstrap has fewer features and is a good choice for projects that need smaller builds.
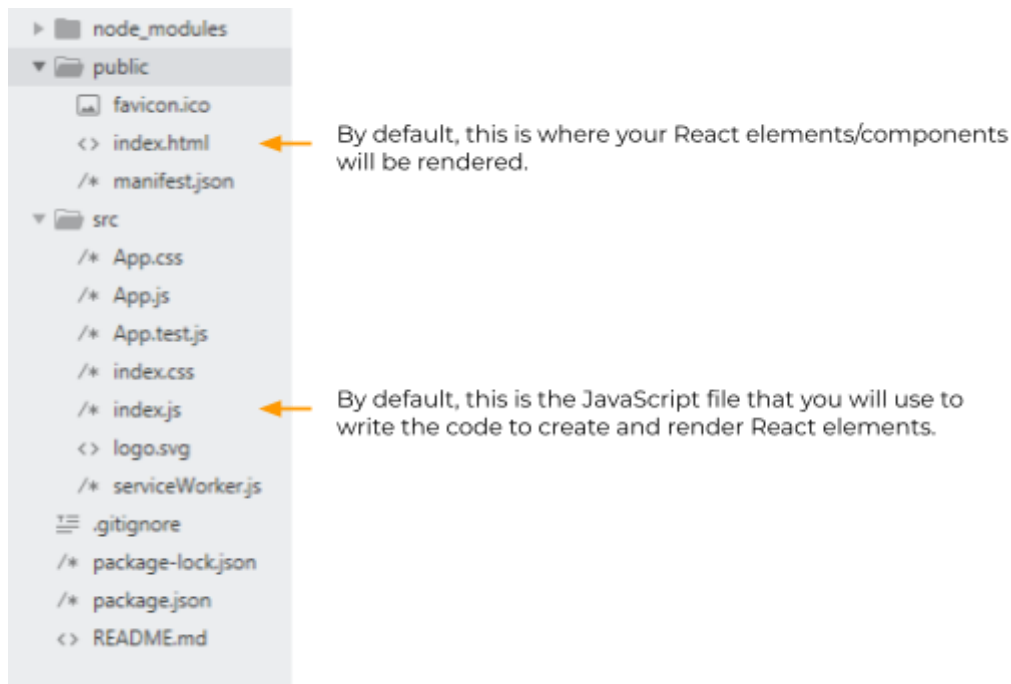
REACT STARTER KITS

React relies heavily on other libraries to work. As we have already seen, **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.

React also 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. It creates a dependency graph that is used to handle all the resources you need for your app. We will discuss Webpack in more detail later in this course.

You could manually install and configure React and all its dependent libraries separately, but Facebook (the creators of React) recommend that you use a starter kit instead. A starter kit automatically installs 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 that are available for you to use, we will be using two official starter kits recommended by Facebook in this course: we will use **Create React App** to create single-page applications and (later in this level) **Next.js** for creating static and server-rendered applications.

When you create a React app using Create React App (CRA), a project folder with a specific directory structure will be created.

By default, this is where your React elements/components will be rendered.

By default, this is the JavaScript file that you will use to write the code to create and render React elements.

By default, the React elements you create will be rendered in the **index.js** file that is created. 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**. In this file, you will see an instruction to render any React elements or components.

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App/>, document.getElementID('root'));
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note that this comes with pitfalls.
// Learn more about service workers at http://bit.ly/CRA-PWA
serviceWorker.unregister();
```
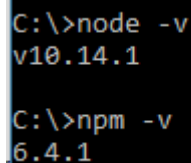
To understand where the element you created will be rendered, open the public directory; you should find a file called **index.html**. This HTML file contains a div element with the ID 'root' (`<div id="root"></div>`). When you use the render method shown in the image above, you are specifying that the element you create will become a child of this div.

DEBUGGING A REACT APP

The developers of React have created various tools to assist with debugging React apps. To find out more, see the debugging tools explained **here**. Click on the appropriate **Firefox** or **Chrome** links to access browser extensions to help with debugging. As you have already seen, you can also use **CodeSandBox** for simple debugging.

REACTING LOCALLY

To use React on your machine, Node and NPM should already be installed. To see if you already have Node and NPM installed, open the command line interface and check which versions of Node and NPM are installed as shown below. If correctly installed, the version numbers should be displayed.

If Node is not yet installed, follow the instructions **here** to install Node. NPM is distributed with Node which means that when you download Node, you automatically get NPM installed on your computer.

> **SPOT CHECK 2**
> Let's see what you can remember from this section.
>
> 1. How do you import a CSS file to style your React elements?
>
> 2. What do you need to import into your JavaScript file to use React?

Instructions

- Copy the 'example' directory that accompanies this task to your local computer. Follow the instructions in the readme.md file before attempting this compulsory task.
- Make sure that you have installed and set up all programs correctly.

Compulsory Task 1

Follow these steps:

- Use **CodeSandBox** to create the following elements. Copy and paste the code for each element you create from CodeSandBox into a file called **my-React-elements.txt**.
  - Write the JSX needed to create and render a button element that links to **https://www.hyperiondev.com/** when clicked.
  - Write the JSX needed to display a div that contains the current date and time. Help **here**.
  - Write the JSX to display a list of the technologies in the MERN stack.
- Make sure that your **my-React_elements.txt** file is saved to the Dropbox folder for this task.

Compulsory Task 2

Follow these steps:
- Create a folder called 'L2Task2' on your local machine.
- Open the command line interface and `cd` to the folder you have created above.
- Follow the instructions found **here** to install React using the Create React App Starter Kit. Call your app "react-hello."
- Once you have started your front-end server (`npm start`), test the default React app you have just created by navigating to http://localhost:3000/ in your browser.
- Modify **src/index.js** to:
  - Create a JavaScript object called 'user' that stores all the details for a particular user of your app. This object should have at least the following properties: name, surname, date_of_birth, country, email, telephone, company, profile_picture, interests. The interests property should be used to store an array of the user's interests.
  - Create and render an element that displays all the information about the user in an attractive way. This element should:
    - Include at least 1 **React-Bootstrap** component.

- - Apply a custom stylesheet that you have created.
- 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, if you're working directly from Dropbox, has the potential to **slow down Dropbox sync and possibly your computer**), compress your project folder and add it to the relevant task folder in Dropbox.

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Completed the task(s)?

Ask an expert to review your work!

Review work

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.

Reference:

React.js. (2020). Getting Started – React. Retrieved 6 August 2020, from https://reactjs.org/docs/getting-started.html

SPOT CHECK 1 ANSWERS

1. React allows us to use JavaScript to write HTML.
2.

   a. Create React and ReactDom objects by importing React and ReactDOM.
   b. Create the React Element using either JavaScript or JSX.
   c. Render the element that you have created to the DOM using the `render()` method.

SPOT CHECK 2 ANSWERS

1. You can import the .css file to the JavaScript file where you are creating the element. E.g. `import './css/custom.css';`
2. ```
   import React from 'react';
   import ReactDOM from 'react-dom';
   import App from './App';
   import * as serviceWorker from './serviceWorker';
   ```