React Basics

written by:
Aya Mostafa
Web Developer

Introduction

Hi friends,

As we talked in the previous article about what is React? and have a good introduction about it. Now, we'll continue learning more about base features in React so let's start 😀

Building our environment

First, we need to install our work react environment from this starter repository sponsored by facebook

https://github.com/facebook/create-react-app

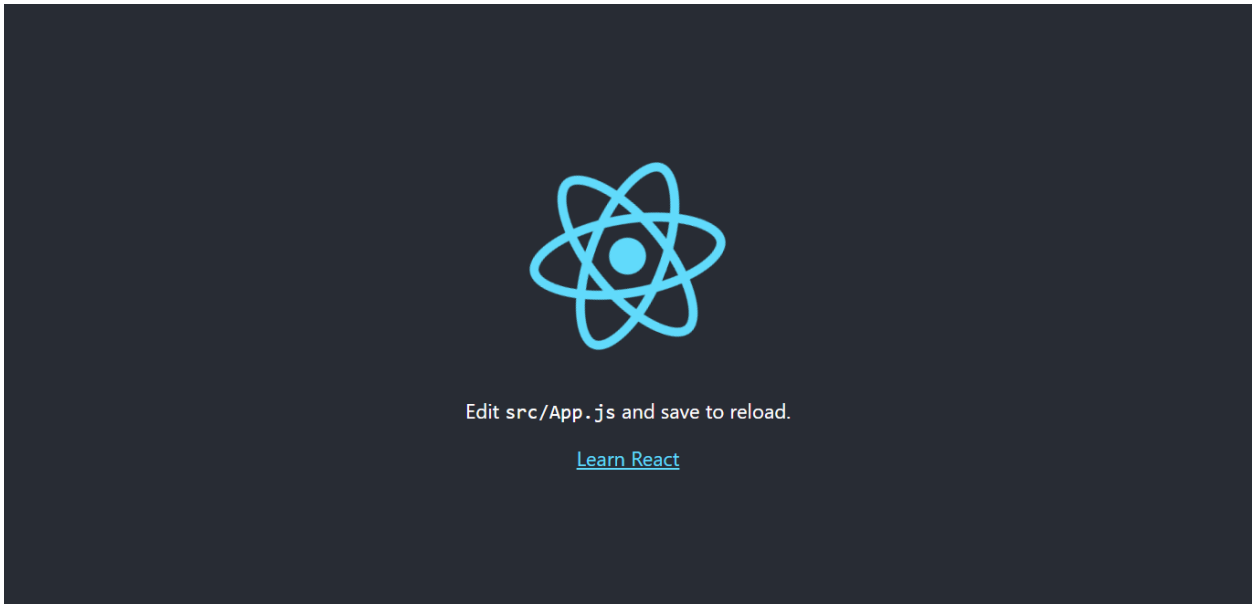
Then navigate to where you want your project to be locally and in your terminal, run this command:-

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

This command will create react app with all needed dependencies to start our react app.

After installing all dependencies, cd your app folder and run “npm run start” in your terminal.

You should see the welcome screen of starter app



Ok, now let’s have a look at our folder structure:-

1. In root path, you can find public which has index.html the main page that will be rendered in the browser when run application and in the body tag there is the following tag

```
<div id="root"></div>  // div with id “root” where the app will be rendered inside it
```

2. Manifest.json file that hold metadata information about the app like app_nam, icons,etc…
3. In src folder at the root path where we will put our app logic so let’s check our files
    a. (Index.js) file is the starting point for the app.you’ll see the following code which calls the component (App) that will be responsible for all other components that will be rendered further in the app.

```
import App from './App';
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root') // root is the id of div in index.html
);
```

    b. (App.js) file where we see our first component in the app. This app component has a function that return html that will be rendered in the dom. Let’s check the code in return it looks like html but in fact it’s not, it’s called JSX.

What is JSX?

JSX stands for JavaScript XML. JSX allows us to write HTML in React and makes it easier to write and add HTML in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
You are not required to use JSX while you code in React, but JSX makes it easier to write React applications.
let’s see how with two examples, the first uses JSX and the second does not 👇

```
Inside App.js in the return the following code uses jsx syntax:-
return (
<div className="App">
     <p>This is React App</p>
```

```
 </div>
);

// we can do the same output without JSX like that
  return React.createElement(
    "Div",   //the root div in dom
    { className: "App" },   // any html attributes for this node
     React.createElement("h1", null, "This is React App") // the children nodes
);
```

JSX is just syntactic sugar for JavaScript, allowing you to write HTML code instead of nested React.createElement(...) calls.

Creating components

you have the choice between two different ways:

- **Functional components** (also referred to as "presentational", "dumb" or "stateless" components)

```
const cmp = (props) => {  //props any arguments will be sent when calling this component
return <div>some JSX {props.propName}</div> // single curly braces for any dynamic content inside html // propName is the name of properties when data is passed into this component like that <cmp title=”...” /> title is the name of prop
 }
*note for any text sended by component it can be accessed by this reserved word inside props like that
<cmp >Any text here...</cmp>
This text can be accessed like that
{props.children}
```

- **class-based components** (also referred to as "containers", "smart" or "stateful" components)

```
class Cmp extends Component
{
   render ()
   {
     return <div>some JSX {this.props.argName}</div>
   }
}
```

State

While props allow you to pass data down the component tree (and hence trigger an UI update), state is used to change the component, well, state from within. Changes to state also trigger an UI update.

```
class NewPost extends Component { // state can only be accessed in class-based components as a class property and word is reserved (can't be changed).
    state = {
        counter: 1
    };
    this.setState({counter: 2})  // to change the states values
    render () { // Needs to be implemented in class-based components.
        return (
            <div>{this.state.counter}</div>
        );
    }
}
```

The difference between states and props is that this happens within one and the same component - you don't receive new data from outside like props.

State in Functional Components

We can manage state of data inside functional component but in different way from class component like that

```
Import  { useState } from 'react';
const NewPost = (props) =>  {
    const [counterState, setCounterState]=    useState({
        counter: 1
    });
 //useState() returns an array with exactly two elements:1) Your current state value 2) A method to update your state value
//if you want to change state value then call the return function
    setCounterState({counter: 2})  // function returned from destruction of useState as the second parms that will be responsible of changing the states values further, please note that in functional component (NO automatic merging)  like state in class component so if our state contains multiple objects, you need to copy every object
```

```
in the new state to the function but in class component it's already merged in the
new state with no need of your code.
      return (
         <div>{counterState.counter}</div>
      );
  }
```

Events

Just like HTML, React can perform actions based on user events. React has the same events as HTML: click, change, mouseover etc.

React events are written in camelCase syntax: onClick instead of onclick.

React event handlers are written inside curly braces: onClick={shoot} instead of onClick="shoot()".

```
class Football extends React.Component {
  shootHandler() {
    alert("Great Shot!");
  }
  render() {
    return (
      <button onClick={this.shootHandler}>Take the shot!</button>
    );
  }
}
```

Styling React Components Using CSS

There are many ways to style React with CSS, we will take a closer look at **inline styling,** and **CSS stylesheet.**

CSS Stylesheet

You can write your CSS styling in a separate file, just save the file with the .css file extension, and import it in your application like that => import './App.css';

Inline Styling

```
const mystyle = {
      color: "white",
      backgroundColor: "DodgerBlue",
      padding: "10px",
      fontFamily: "Arial"
```

```
    };

 render() {
    return (
      <div>
      <h1 style={{ backgroundColor:blue; color: "red"}}>...!</h1>
      <h2 style={mystyle}></h2> //You can also create an object with styling
information, and refer to it in the style attribute
      </div>
    );
  }
```

*Since the inline CSS is written in a JavaScript object, properties with two names, like background-color, must be written with camel case syntax:

If you want to give your app powerful styling capabilities without CSS, there are two packages that gives you a set of tools to manage inline styles on React elements and improved experience:-

1. radium
2. styled-components

So have a look at them 😀

CSS Modules with create-react-app

If you want a create-react-app to treat a CSS file as a CSS Module file, you need to use a specific naming convention when naming your CSS file. You need to name your CSS files with (fileName.module.css) extension. This way create-react-app knows that the file should be treated as CSS Module then you can import the CSS file normally and use the class names scoped to a given component like that:-

```
// In App.module.css File
.App {
    color: red;
}

// In App.js File
import styles from "./App.module.css"; // then you can access any class from this
file dynamically
 return (
      <div className={styles.App}>
);

// By the way, if you somehow also want to define a global class name
, you can prefix the selector with :global like that
```

8

```
:global .App {
   color: red;
}
```

CSS Modules are a relatively new concept (you can dive super-deep into them here: https://github.com/css-modules/css-modules).

Conclusion

By the end of this article, hope you enjoyed knowing more about react features. You can find the source code of this article at https://gitlab.com/ayamostafa/learn-react/-/tree/base

Good Bye,
aya mostafa