REACT JAVASCRIPT LIBRARY: CONCEPTS & TUTORIALS FOR GETTING STARTED



React, aka React JS, is a JavaScript library designed for building user interfaces in web applications. It focuses on powering the view layer of the MVC (Model-View-Controller) architectural pattern.

React is based on reusable elements called components that help to create fast and scalable user interfaces—something many developers love.

So, in this article, we will:

- Look at the React JavaScript library
- Explore main features
- See building blocks of a React app
- Understand its benefits
- Compare React to other JS frameworks
- Build a sample React application

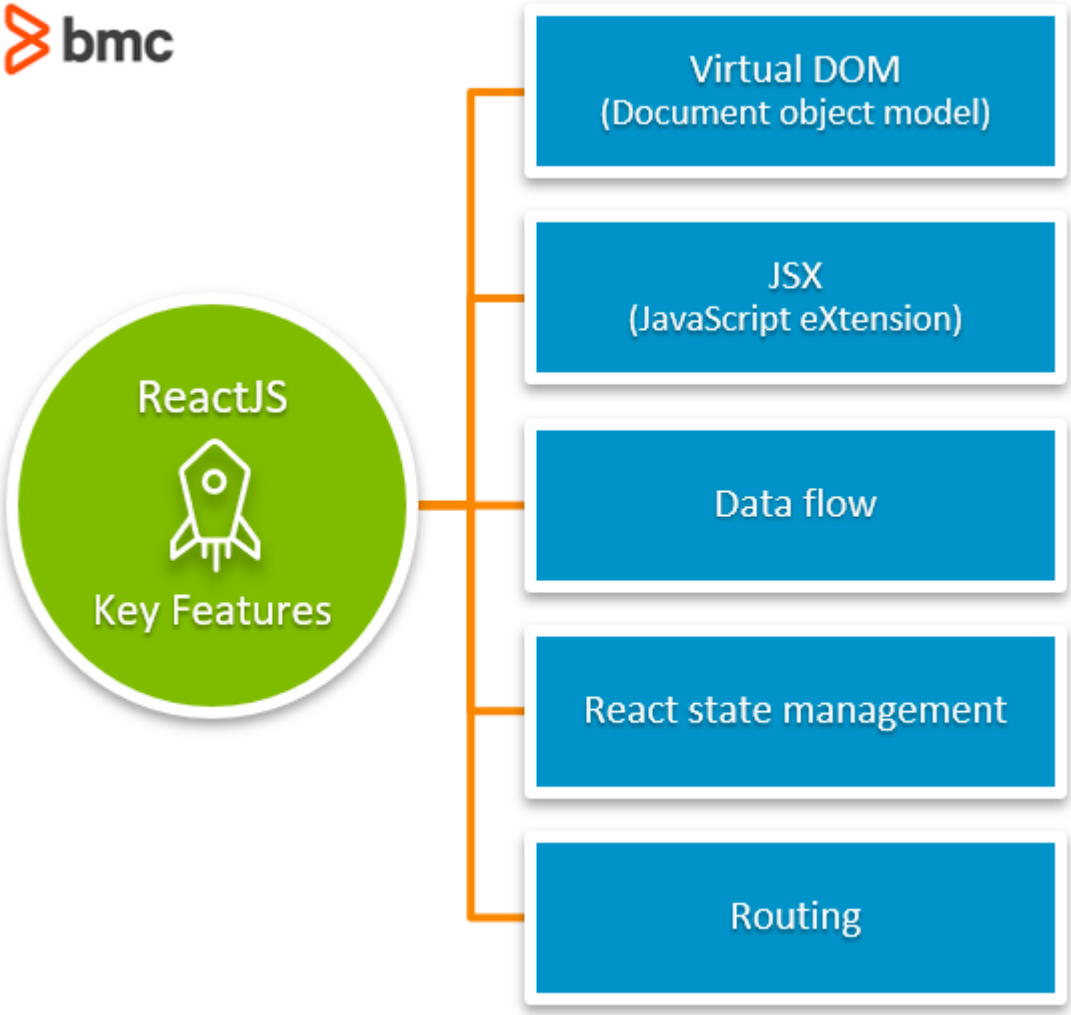
Let's get started!

When to use ReactJS

React is aimed at creating robust and extensible user interfaces (UI) across multiple platforms. One bright spot of React is that it does not limit users to a specific JavaScript framework. React also:

- Supports integration with preexisting JS frameworks to enhance the user experience of an application—without complex modifications to the backend.
- Enables developers to easily utilize other frameworks and platforms designed for React, such as Gatsby or js, to power a React Stack.

React is the ideal choice for creating cross-platform, responsive, versatile, and scalable applications when it comes to application development.



Main features of React

To better understand React, we need to know about the components that power it. So, in this section, we'll go through the core concepts and features of React.

Virtual DOM

The document object model (DOM) is the core part that represents the complete structure of a web page document in any web application. It defines the logical structure of an HTML or XML document and how they are accessed and manipulated, enabling developers to target specific elements and objects with a document directly.

This approach further allows developers to update specific parts of the DOM tree without having to load the complete web page—resulting in:

- Increased performance

- A better user experience

React takes this process one step further by utilizing a virtual DOM, which acts as a virtual representation of the DOM tree stored in memory. At the start of the web application, React will create a copy of the DOM tree and store it in memory. React will update the virtual DOM tree when an update happens, then compare the updated virtual DOM tree with the real DOM tree and update only the necessary elements in the real DOM tree.

This reduces overall load times even further and leads to fast, responsive designs.

JSX

JavaScript eXtension (JSX) is a syntax extension for JavaScript. It enables developers to define HTML structure within the JavaScript code. JSX also simplifies the development process by:

- Increasing the readability of overall code
- Reducing the need for highly complex DOM structures.

While JSX is not mandatory in React, it is widely adopted because it allows developers to define UI elements with JavaScript. JSX is also used for creating React components.

Example syntax:

```
// Example - JSX
const name = "Barry Stevens";
const age = 14;

const message_user = <p>{name} is {age} years old.</p>;

function get_city() {
    city = "New York";
    return city;
}

const message_city = <h1>This is {get_city()}</h1>
```

Data flow

In React, data is handled according to the “properties flow down, actions flow up” principle. React passes a set of immutable values to the components renderer as properties. The component cannot directly modify these properties, and developers can utilize a call-back function to request modifications.

React state management

All React components have built-in states. This state simply refers to the data structure within the component. State management in React provides a way to both:

- Create communications (connections)
- Share data across React components

Every React function or class consists of a state. It's essential to properly manage the state across the web application as the state of a component can be changed by a user action (user interaction

with a UI element).

React offers powerful state management libraries that can be easily integrated into a web application to provide enhanced control over function and class states. Common libraries include:

- Redux
- Recoil

Routing

One of the crucial parts of a good application is the ability to navigate between pages easily.

Usually, developers will use the History API within a browser to facilitate this navigation (routing). However, the feature set and functionality offered by History API are quickly outmatched when dealing with complex applications.

React provides a feature called React Router in its standard library to facilitate routing functionality. React Router is an API that enables developers to create a routing layer for the application. On top of that, it offers additional features such as:

- Dynamic route matching
- Location transition handling
- Lazy code loading
- Etc.

Building blocks of a React application

Next, let's look at the building blocks that go into any React app.

Components

Components are the core building block of a React-based UI. These components are analogous to JavaScript functions. Each component will accept a set of defined inputs and return a React element that will be rendered in the frontend.

This component-based approach allows developers to create isolated components which address specific issues and can be easily reused elsewhere. This reduces repetition, leading to more robust source code.

State

The built-in React object in each component contains data about the component. As state relates to data of a component, any changes to data (state) cause the component to re-render. User interactions or system events will cause changes in the state, and it will directly impact the behavior and visualization of the component.

Stateful component example:

```
// Example - State
class App extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      name: "Hello World",
      description: "State Example"
    }
  }
  render() {
    return (
      <div>
        <h1>{this.state.name}</h1>
        <h2>{this.state.description}</h2>
      </div>
    );
  }
}
export default App;
```

Props

In React, “props” is short for properties, which is a built-in object that stores the attribute values of tags. Props can be analogous to HTML attributes. These props can be passed to components as inputs to facilitate functionality with a component similar to traditional method arguments.

Example of passing username as an argument for the component:

```
class User extends React.Component{
  render(){
    return(
      <div>
        <label>Username: {this.props.username} </label>
      </div>
    );
  }
}

export default User;

const user_details = <User username = "barry"></User>
```

Advantages of React

React offers many benefits to app developers.

Simplicity & ease of use

React is comparatively more approachable than other frameworks and libraries. The only prerequisite to learn React is a basic understanding of HTML, CSS, and JavaScript. Even if React uses the powerful JSX syntax, developers do not have to use it. Instead, they can start using React with plain old JavaScript.

Another point for simplicity: component-based development methodology helps developers to create powerful yet easily manageable and scalable applications.

Performance

React is built with performance as a core tenant, as we see in features like the virtual DOM, data flow, and server-side rendering. React applications render faster by only reloading necessary components of the DOM tree. This drastically reduces the load time on the frontend, promoting a smoother user experience.

Additionally, React ensures that child structure changes will not affect the parents with its single-way data flow. Those changes are encapsulated within the child structures, with states controlling the modification of components. This makes the application more stable, and it increases overall performance.

React has the ability to run on the backend server to render and return DOM to a browser as regular HTML. This helps mitigate search engine optimization (SEO) issues inherent to JavaScript-based applications, thus increasing the application's SEO.

React EcoSystem

React has a robust ecosystem that contains numerous libraries that can be used to easily integrate various functionalities to a web application, such as:

- State management
- Form routing (Reach Router, React Router)
- Visualizations

On top of that, UI components libraries such as KendoReact and Rebass enable developers to add ready-made, robust components to their applications without having to build everything from scratch.

All these libraries are managed through npm, which is the largest software registry.

Developer tool set

React offers a comprehensive toolset to support the development and debugging of web applications. It has many tools designed to simplify the software development lifecycle (SDLC), such as:

- Create React App, a CLI tool to set up a React project with a single command
- React Developer Tools extension to debug react applications
- React-Unit to run unit tests on React core
- Enzyme for performance testing

You can also easily integrate React with tools like VSCode, Atom, and WebStorm to create a proper development environment. Finally, React starter kits like React Static Boilerplate and Rekit help developers to kickstart the development of a React-based application.

React Native

React has evolved beyond just web applications. Now, React Native supports mobile UI development for both Android and iOS. This allows developers to create JavaScript-based UIs for mobile applications that will render with native code.

Additionally, React Native facilitates creating platform-specific components while sharing a single codebase and backend that powers both web and mobile platforms.

React vs Angular vs Vue: Comparing JS frameworks

The following table illustrates a comparison of React with other JavaScript frameworks.

	React	Vue	Angular
Type	JavaScript library	JavaScript framework	JavaScript framework
Architecture	Flexible architecture (View layer that can be added to any model)	Flexible architecture (View layer that can be added to any model)	Model-View-Controller (MVC) architecture
License	MIT License	MIT License	MIT License
Use Cases	Cross-platform applications	Simple, lightweight applications	Large-scale applications
Built-in Features	The library is only aimed at creating UI components and interacting with DOM (Additional features are available through external components - Redux, React Router)	Halfway point between React and Angular. Vue offers some built-in features like routing and state management but lacks features like validations (which are supported through external components)	Largest framework with the most extensive built-in feature set (E.g.: validations, routing, state management, etc.)
Scalability	High (Via component-based approach)	Limited (Due to the template-based syntax)	High (Via modular development composition)
DOM	Virtual DOM	Virtual DOM	Real DOM
Data Binding	Single-Way Data Binding	Two-Way Data Binding	Two-Way Data Binding
Learning Curve	Medium	Low	High

Tutorial: sample React application

The easiest way to get up and running with your React app is to use the Create React App CLI tool. It will create a React project with a simple application structure that can be used to build an application.

First, we'll create a React app using the following command:

```
npx create-react-app simple-app
```

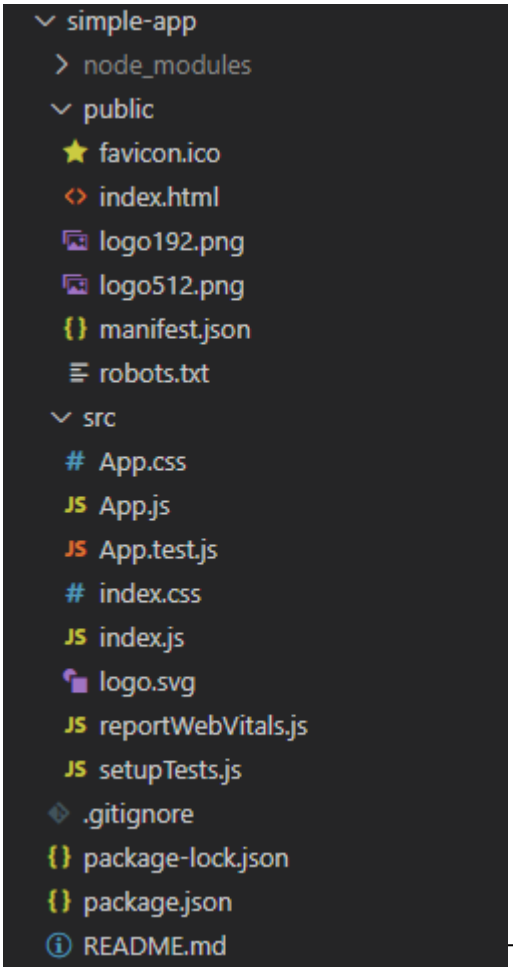
Result:

```
❯ npx create-react-app simple-app
Need to install the following packages:
  create-react-app
Ok to proceed? (y) y

Creating a new React app in G:\Dev - Env\Projects\Articles\react_test\simple-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
```

This will install all the required packages and create a sample folder structure for the React application.



The App.js file is the primary file in the React application. In the sample code block below, we import the logo and CSS classes and then use the **App() function** to render and return the elements.

App.js

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

If we look at the index.js, we can see that it points to the App function and renders it in the frontend.

index.js

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

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

reportWebVitals();
```

Then we can run the application by running the **npm start** command, which will:

1. Compile the application .
2. Start the development build accessible through localhost via port 3000.

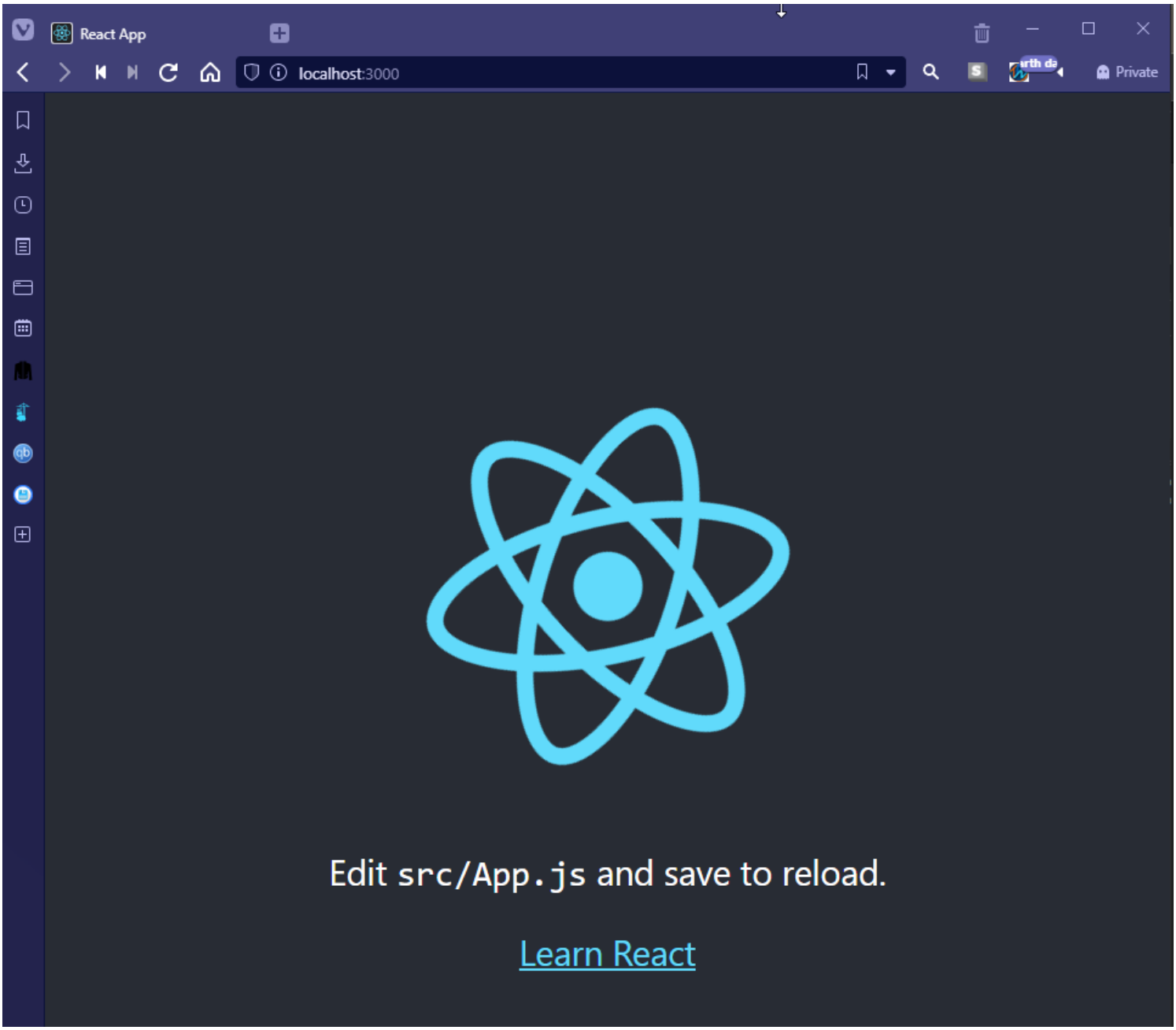
```
Compiled successfully!

You can now view simple-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.6.0.3:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

Sample React app (Browser)



Now, let's modify this application by creating a new file called Home.js and rendering it as the output.

Home.js

```
import "./App.css";

function Home() {
  return (
    <div className="Home">
      <h1>Hello React Test</h1>
      <h2>This will be rendered in the index.js</h2>
    </div>
  );
}

export default Home;
```

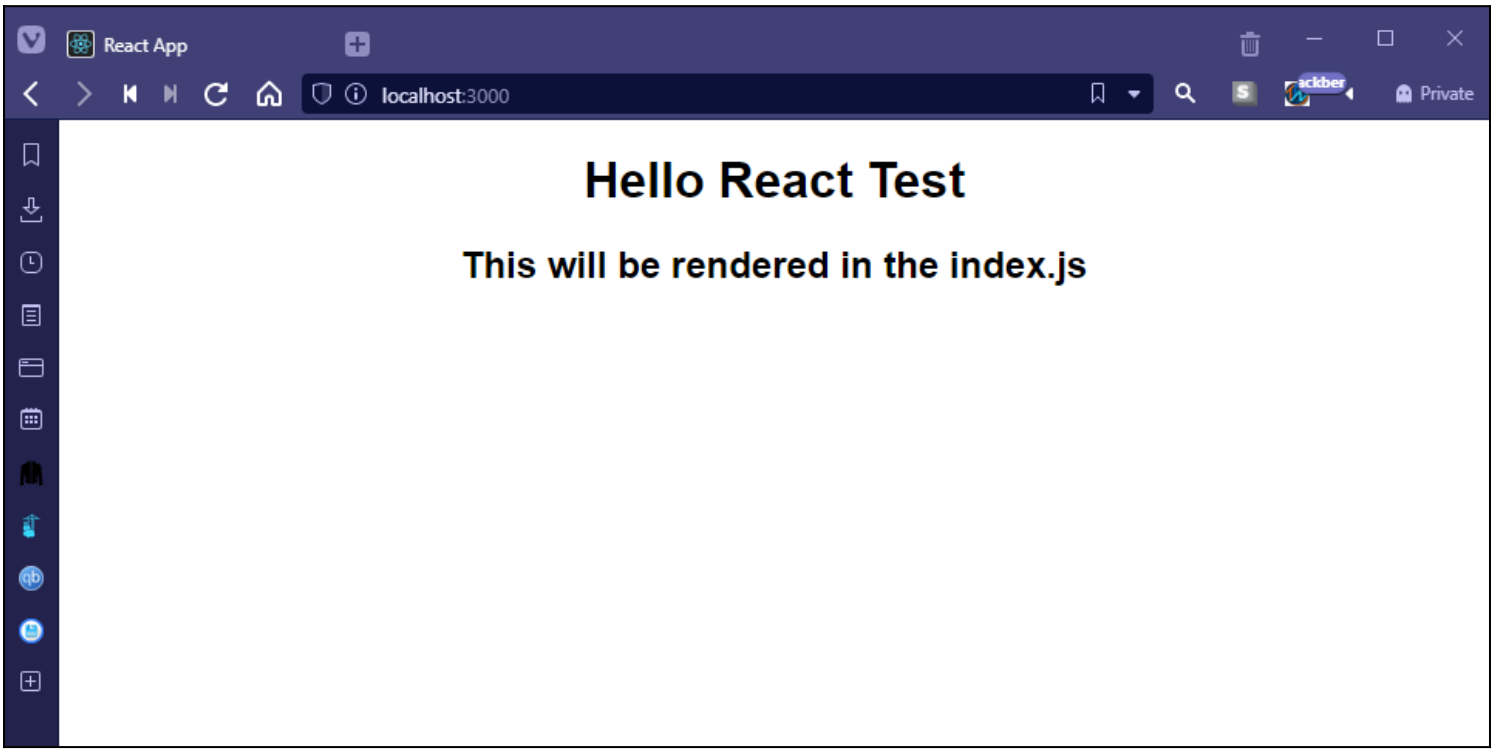
index.js

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

ReactDOM.render(
  <React.StrictMode>
    <Home />
  </React.StrictMode>,
  document.getElementById('root')
);

reportWebVitals();
```

We are now importing the newly created Home.js file and rendering it in the index of the application. The result will show:



That's it! You've just successfully created a simple React application. Now, you can start creating a component based React application using the structure of this sample application.

React is for all developers

Though React isn't something you want to use for every application—indeed, simpler apps are best—developers appreciate React for being easy to learn, use, and integrate within their existing environment.

Related reading

- BMC DevOps Blog
- Best Programming Languages To Learn Today
- The Lua Programming Language Beginner's Guide
- Polymorphism In Programming
- Application Performance Management in DevOps
- Top Conferences for Programming & Software Development