React is a front-end library developed by Facebook. It is used for handling the view layer for web and mobile apps. ReactJS allows us to create reusable UI components. It is currently one of the most popular JavaScript libraries and has a strong foundation and large community behind it.

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −

React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features

* **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
* **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
* **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
* **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

## React Advantages

* Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
* Can be used on client and server side as well as with other frameworks.
* Component and data patterns improve readability, which helps to maintain larger apps.

## React Limitations

* Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
* Uses inline templating and JSX, which might seem awkward to some developers.

The main difference is that for SSR your server’s response to the browser is the HTML of your page that is ready to be rendered, while for CSR the browser gets a pretty empty document with links to your javascript. That means your browser will start rendering the HTML from your server without having to wait for all the JavaScript to be downloaded and executed. In both cases, React will need to be downloaded and go through the same process of building a virtual dom and attaching events to make the page interactive — but for SSR, the user can start viewing the page while all of that is happening. For the CSR world, you need to wait for all of the above to happen and then have the virtual dom moved to the browser dom for the page to be viewable.

Now, there are a few caveats:

* While the page is rendered earlier and the customer can see the page sooner, they can’t really interact with it until react is done executing. If the customer is really fast and clicks a button, the action won’t be executed until React is done executing;
* SSR TTFB(Time To First Byte)is slower than CSR, because your server will have to spend the time to create the HTML for your page instead of just sending out a relatively empty response;

React Directly in HTML

<script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>

<script src="https://unpkg.com

/react-dom@16/umd/react-dom.production.min.js"></script>

<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>

React's goal is in many ways to render HTML in a web page.

React renders HTML to the web page by using a function called ReactDOM.render().

React using NPM

Install create-react-app by running this command in your terminal:

npm install -g create-react-app

to create a React application named myfirstreact:

npx create-react-app myfirstreact

to start app

npm start

JSX

React uses JSX for templating instead of regular JavaScript. It is not necessary to use it, however, following are some pros that come with it.

* It is faster because it performs optimization while compiling code to JavaScript.
* It is also type-safe and most of the errors can be caught during compilation.
* It makes it easier and faster to write templates, if you are familiar with HTML.

## Using JSX

JSX looks like a regular HTML in most cases. We already used it in the Environment Setup chapter. Look at the code from **App.jsx** where we are returning **div**.

JSX stands for JavaScript XML.

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

This funny tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

const myelement = <h1>React is {5 + 5} times better with JSX</h1>;

## React Components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and returns HTML via a render function.

Components come in two types, Class components and Function components,

## Create a Class Component

When creating a React component, the component's name must start with an upper case letter.

The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

## Create a Function Component

A  Function component also returns HTML, and behaves pretty much the same way as a Class component, but Class components have some additions,

States:

**State** is the place where the data comes from. We should always try to make our state as simple as possible and minimize the number of stateful components. If we have, for example, ten components that need data from the state, we should create one container component that will keep the state for all of them.

React components has a built-in state object. The state object is where you store property values that belongs to the component. When the state object changes, the component re-renders.

**Jest** is a JavaScript test runner that lets you access the DOM via jsdom. While jsdom is only an approximation of how the browser works, it is often good enough for testing React components. Jest provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.

**React Testing Library** is a set of helpers that let you test React components without relying on their implementation details. This approach makes refactoring a breeze and also nudges you towards best practices for accessibility. Although it doesn’t provide a way to “shallowly” render a component without its children, a test runner like Jest lets you do this by mocking.