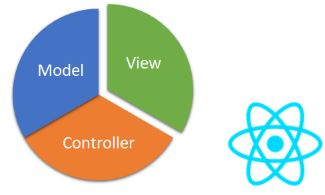
**What is React ?**

React is a JavaScript library created for building fast and interactive user interfaces for web and mobile applications. It is an open-source, component-based, front-end library responsible only for the application’s view layer. In Model View Controller (MVC) architecture, the view layer is responsible for how the app looks and feels. React was created by Jordan Walke, a software engineer at Facebook.

****

**Why React?**

React’s popularity today has eclipsed that of all other front-end development frameworks. Here is why:

**Easy creation of dynamic applications**: React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly.

**Improved performance**: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM compares the components’ previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.

**Reusable components**: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their logic and controls, and they can be reused throughout the application, which in turn dramatically reduces the application’s development time.

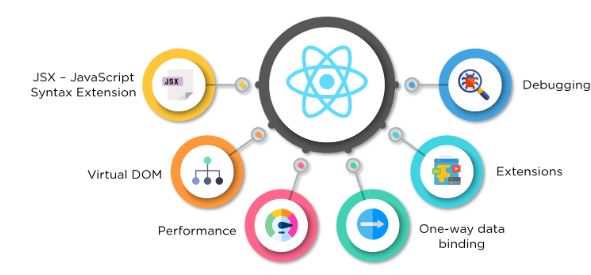
**Unidirectional data flow**: React follows a unidirectional data flow. This means that when designing a React app, developers often nest child components within parent components. Since the data flows in a single direction, it becomes easier to debug errors and know where a problem occurs in an application at the moment in question.

**Small learning curve**: React is easy to learn, as it mostly combines basic HTML and JavaScript concepts with some beneficial additions. Still, as is the case with other tools and frameworks, you have to spend some time to get a proper understanding of React’s library.

**It can be used for the development of both web and mobile apps**: We already know that React is used for the development of web applications, but that’s not all it can do. There is a framework called React Native, derived from React itself, that is hugely popular and is used for creating beautiful mobile applications. So, in reality, React can be used for making both web and mobile applications.

**Dedicated tools for easy debugging**: Facebook has released a Chrome extension that can be used to debug React applications. This makes the process of debugging React web applications faster and easier.

**Features of React:**



**JSX - JavaScript Syntax Extension**

JSX is a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code. This makes the code easier to understand and debug, as it avoids the usage of complex JavaScript DOM structures.

jsx



const name = 'Simplilearn';

const greet = <h1>Hello, {name}</h1>;

The above code shows how JSX is implemented in React. It is neither a string nor HTML. Instead, it embeds HTML into JavaScript code.

**Virtual DOM**

React keeps a lightweight representation of the “real” DOM in the memory, and that is known as the “virtual” DOM (VDOM). Manipulating real DOM is much slower than manipulating VDOM because nothing gets drawn on the screen. When the state of an object changes, VDOM changes only that object in the real DOM instead of updating all of the objects.

It may all seem a bit overwhelming for now, so let’s first understand what DOM is, and then we’ll go through how VDOM and real DOM interact with each other.

What is the Document Object Model (DOM)?

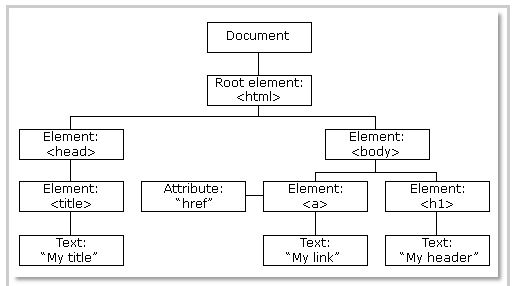


Fig: DOM of a Webpage

DOM (Document Object Model) treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

**How do Virtual DOM and React DOM interact with each other?**



Fig: Virtual DOM

When the state of an object changes in a React application, VDOM gets updated. It then compares its previous state and then updates only those objects in the real DOM instead of updating all of the objects. This makes things move fast, especially when compared to other front-end technologies that have to update each object even if only a single object changes in the web application.

**Performance**

React uses VDOM, which makes the web applications run much faster than those developed with alternate front-end frameworks. React breaks a complex user interface into individual components, allowing multiple users to work on each component simultaneously, thereby speeding up the development time.

**Extensions**

React goes beyond simple UI design and has many extensions that offer complete application architecture support. It provides server-side rendering, which entails rendering a normally client-side only web application on the server, and then sends a fully rendered page to the client. It also employs Flux and Redux extensively in web application development. Finally, there is React Native, a popular framework derived from React, used to create cross-compatible mobile applications.

**One-way Data Binding**

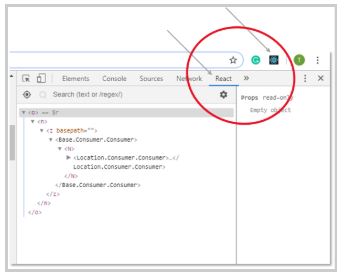
React’s one-way data binding keeps everything modular and fast. A unidirectional data flow means that when a developer designs a React app, they often nest child components within parent components. This way, a developer knows where and when an error occurs, giving them better control of the whole web application.



**Fig:** One-way data binding

**Debugging**

React applications are easy to test due to a large developer community. Facebook even provides a small browser extension that makes React debugging faster and easier.



This extension, for example, adds a React tab in the developer tools option within the Chrome web browser. The tab makes it easy to inspect React components directly.

**What is JSX ?**

JSX is a custom syntax extension to JavaScript which is used for creating markup with React. It might remind you a bit of a templating language, but with JSX you can use the full power of JavaScript. However, remember that JSX will not work directly in browsers and requires a build step to convert JSX markup into React.createElement function calls.

**Why use JSX?**

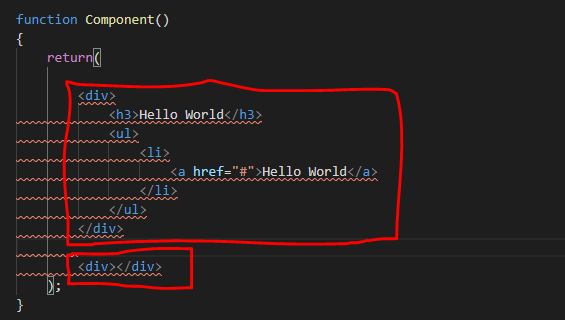
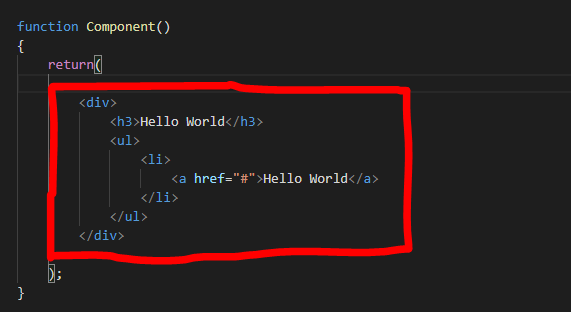
* It is faster in comparison to JavaScript as it performs optimization during translation of the code to JavaScript.
* Instead, the separation of technologies by putting markup and logic in separate files, React has the components that contain both.
* It is type-safe because most of the errors are found at compile time.
* It makes template creation easy.

[**React JSX rules must be follow while using React.**](https://medium.com/nerd-for-tech/jsx-rules-in-react-a-javascript-framework-4b0ab66fdbf9)

* Return Single Element.
* div/section/article or React Fragment.
* Use camelCase for HTML Attribute.
* className Instead of Class.
* Close Every Element.
* Formatting.
* Tags are elements.

**Return Single Element:**

In JavaScript we use to return JSX elements to DOM through components. We return HTML elements or tags to render it later. So one of the rule is to return single element. Wait I don’t mean that you can just return single tag. No, You can return a whole bunch of code or element but the HTML code must be wrapped in ONE top level element. So if you like to write two “div”, you must put them inside a parent element, like a “div” element.



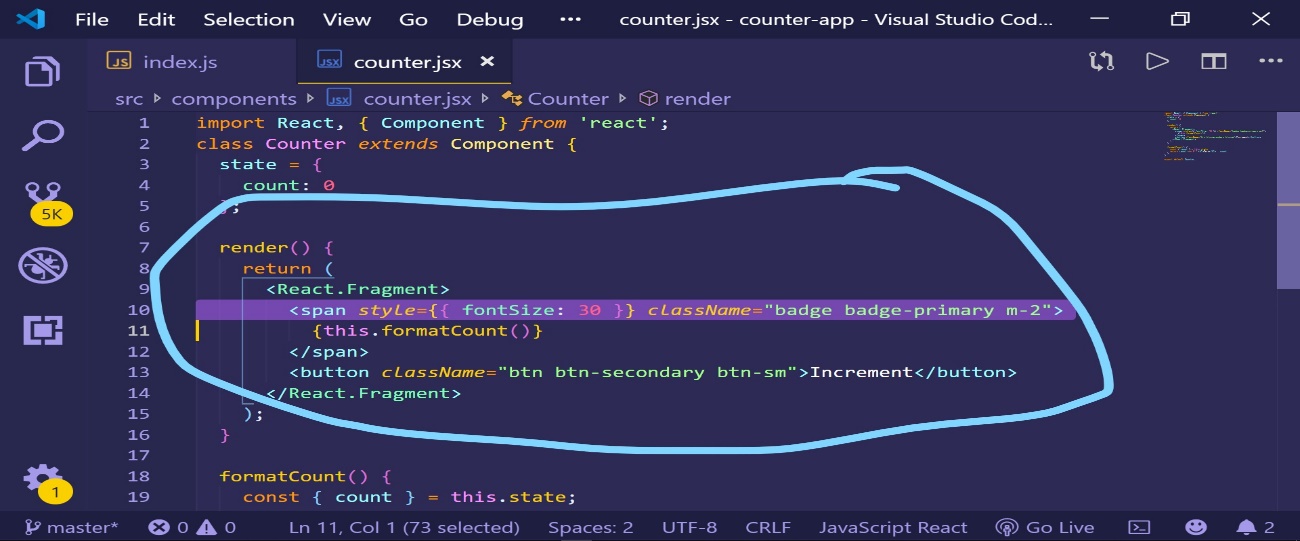
**div/section/article or React Fragment:**

While working with the JSX one should keep in mind the HTML semantics. In the above rule I make it clear that there should a parent element to all the HTML elements. For these elements we can use “div” but we shouldn’t use it for everything. There is no rule against it to “div” everywhere but it is the best practice to follow the HTML semantics. An empty tags can also be used for the top level element.



**Use camelCase for HTML Attribute:**

In JavaScript we used to write inline attributes or event Listeners e.g. “onclick()”. But in case of React we can’t write attributes like this. “onclick” attribute of JavaScript must be capitalized e.g. camelCase. If you don’t know about camelCasing, here you can read more about camelCase.



**className Instead of Class:**

In HTML we use class attribute for specific element. As we are using React, a JavaScript framework so we can’t use class keyword because in ES6 it is a reserve keyword to define a class. If we use class for HTML attribute it will gives an error, instead we should use className.

**Close Every Element:**

In HTML almost all the tags have starting and closing tags, rather than a few e.g. <img>, <input>, <br>.

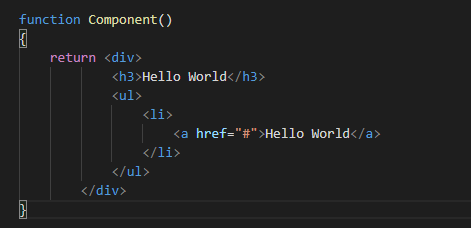
These few tags have no closing tags. We use it simply as the way they are defined in the example above. In React JSX every tag must be close even those which have no closing tags e.g.

<img src = “ “ alt = “ “ />

**Formatting:**

While returning JSX we should have parenthesis so the HTML code must be wrapped inside.

Don’t worry you have a choice to use parenthesis or not. if you don’t want to use it, your parent element starting tag must be in the same line where the return keyword exist.



**Tags are elements:**

JSX Tags map to calls to React.createElement().

* Use lowercase tags <lowercase/> when you need a DOM elements
* Capitalized tags <Capitalized/> for component elements.

**Looping In JSX ?**

**MAP**

mport React from ‘react’

const RenderList = props => {

const animals = ["Dog", "Bird", "Cat", "Mouse", "Horse"];

return (

<ul>

{animals.map((animal, index) => (

<li key={index}>{animal}</li>

))}

</ul>

);

};

**FOR-OF**

const getAnimalsContent = animals => {

let content = [];

for (let item of animals) {

content.push(<li key={item.id}>{item.animal}</li>);

}

return content;

};

return <ul>{getAnimalsContent(animals)}</ul>;

**FOR-IN**

const getAnimalsContent = animals => {

let content = [];

for (let idx in animals) {

const item = animals[idx];

content.push(<li key={item.id}>{item.animal}</li>);

}

return content;

};

return <ul>{getAnimalsContent(animals)}</ul>;