**Let’s talk about ReactJS -** [Chandur Dissanayake](https://medium.com/@chandur.d?source=post_page-----723d70645eb0--------------------------------) May 13, 2022

In this article you will learn about one of the top UI frameworks out there in the world. We will be talking a lot on React so I will try to keep the information concise and as clear as possible. So lets get started with React and I hope you learn and gain a fundamental knowledge on React.

**Table of Contents**

1. [Introduction](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#23c4)
2. [Why React?](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#e633)
3. [Features of React](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#4da1)
4. [Industry trends](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#2007)
5. [Key Concepts of React](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#62d8)
6. [Component Lifecycle](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#e897)
7. [React Hooks](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#c0cc)
8. [Conclusion](https://medium.com/@chandur.d/lets-talk-about-reactjs-723d70645eb0#dcd2)

**Introduction**

React is one of the most popular JavaScript frameworks used for development of frontend. It is also worth noting that React is maintained by Meta (Facebook) and also open source community developers and companies. Unlike other frameworks React is a library rather than a language. React also offers extension and options to developers for entire application architectural support, such as Flux and React Native, beyond the scope of mere UI.

**Why React?**

What is the answer you must be wondering and what made React more popular than other frontend frameworks such as Angular and Vue?

* It is very convenient to create a dynamic web application in React as it requires less coding and provides more functionality when compared to plain JavaScript, where coding of a dynamic application can get very complex and confusing over time.
* React has enhanced performance because it uses Virtual DOM hence creating web applications would be more efficient. What Virtual DOM does it that it compares the component's previous sate and updates **only**the items in the Real DOM that have been changed. So in simple terms it only renders the items which have been changed, not the entire component like other typical web applications do.
* In React Components are like the building blocks of the application in which an application typically contains many components. These components can be reused for different other functionalities consisting of similar logic and controls. Therefore significantly reducing the application development time.
* Debugging of errors in React is much easier due to that fact that data flow in React is **unidirectional**. So for example in a scenario where there are nested child components within a parent component it would be easier to pin point the error as data moves at only one direction.
* If you already know JavaScript and HTML, React would be very easy to learn because React only has a small leaning curve.
* React can be used to build both web or mobile applications. using React Native we can build beautiful mobile applications, so learning React would never be a waste due to its dual operability.
* To make debugging even more convenient, Meta released a Chrome extension which can debug a React application when it is run, which I'm sure developers getting into React would appreciate and find it useful at tough times.

**Industry Trends**

Graphical user interface, chart, line chart

Description automatically generated

As you can see in the above graph, React is widely adopted compared to other frameworks or libraries making it very beneficial to learn and have a knowledgeable understanding.

**Features of React**

Some of the features React offers are as follows, which made then widely popular and most adopted library for frontend.

**JSX**

Icon

Description automatically generated with medium confidence

JSX is a JavaScript syntactic extension (fileName.jsx) which is used to create UI structures. With JSX we can write both JavaScript as well as HTML in the same code. In the below example we can see HTML is embed into the JavaScript code.

Graphical user interface, text, application

Description automatically generated

Example of JavaScript and HTML

**Virtual DOM (Virtual Document Object Model)**

Diagram

Description automatically generated

A self explanatory image

As we can see in the below image when there is a change in state of an object, the Virtual DOM updates only that specific object in the Real DOM instead of updating everything. This makes the website feel fast and very responsive.

Chart, bubble chart

Description automatically generated

But what is a DOM you may ask. DOM or Document Object Model configures XML or HTML documents as a tree structure where each node is an object of the document. An image is given below to visually illustrate how the DOM works and structure.

Diagram

Description automatically generated

DOM of a generic webpage

**Architecture**

Diagram

Description automatically generated

Architecture

React perfectly aligns with the Model View Controller (MVC) architecture where React would be the “View”. The “View” is responsible for the UI logic of the application.

**React Extensions**

Icon

Description automatically generated with medium confidence

React is more than just a UI framework because it covers the complete application architecture. For example, React aids in creating mobile apps and also provides server-side rendering. Features such as Flux, Redux and etc can greatly extend React.

**Data Binding**

Data in React move unidirectionally meaning that it is common to have nested child components inside of parent components in React projects. In addition, since React uses one-way binding, all the activities would have high modularity and speed.

Graphical user interface

Description automatically generated with medium confidence

One-way data binding

**Debugging in React**

Because there is a large community in React, it is easy to test and debug the code. To aid with this Facebook provides a browser extension that make debugging easier and avoids the need to use third part debugging tools as well.

A picture containing graphical user interface

Description automatically generated

React debugging extension

**Key Concepts of React**

**React Components**

In React components act as the building blocks in the application user interface.

Graphical user interface

Description automatically generated

Different components making the UI

React divides the user interface into different components making it easier to design, debug and write more reusable and organized code. Each component has its own set of properties and functions. Note : there are 2 types of components, functional or class components. These types achieve the same goal but in different ways of doing so.

*Characteristics of components*

* Re-usable — The same component can be reused in another section reducing development time
* Nested Components — Components can contain many other components
* Render method — A component must declare a render method specifying how the component renders to the DOM
* Passing properties — Component can receive props. Props or properties passed by the parent component specifying some values.

*Demo example :*

Text

Description automatically generated

Functional Component

A screenshot of a computer

Description automatically generated with medium confidence

Class Component

Notice that in class component there is a render() function where as in functional component we don't. This is why typically developers code in the format of functional components as code is much simpler and easier to understand. export default is used to export only one object from the file which can be a function, variable or class. export default can only be used once per file.

When we export these components we can import them to our App.js

Text

Description automatically generated

App.js

Then we can run the application and we would get the following inside the browser.

Graphical user interface, text, application

Description automatically generated

Browser Output

**React Props**

Props or properties allows the user to pass arguments or some data to components making them more dynamic. Props inside of a component are read-only so they cannot be manipulated.

*Demo example :*

Text

Description automatically generated

classProps.js

As you can see we have 2 properties called “name” and “place” and we can utilize these props by defining them in our parent component (App.js) to have a dynamic effect.

Text

Description automatically generated

App.js

Below shows the browser output where as you can see the same component is called 3 times with passing of 3 different values for the same property name

Graphical user interface, text, application

Description automatically generated

Browser Output

**React State**

Simply a state is a object that stores properties values of a component that may potentially change over time. There are some facts on state;

* State can be changed by user actions or changes in network
* React re-renders the component when the state of an object is changed
* The function Object (){} is where state object is created
* Many properties can be stored in the state object
* setState() can be used to change the values of the state object
* setState() merges the newly modified and prior states shallowly

*Demo example :*

As you can see the <h3> tag outputs the value of “message” which is a state object

Text

Description automatically generated

State.js

Text

Description automatically generated

App.js

Graphical user interface, text, application, email

Description automatically generated

Bowser Output

By using setState() a state can be updated to even handlers, server responses or even prop changes.

this.setState({ quantity: value })

setState() adds to the queue of all the updates which is made to the component state and then tells React to re-render the component along with their children to have the updated state.

Text

Description automatically generated

State.js

We can create another state object called sub for the button. When the button is clicked the method ChangeMessage() is called and uses the setState() method to update the values of message and sub and then finally re-renders the output.

Graphical user interface, text, application

Description automatically generated

Browser Output : before clicking button

Graphical user interface, application, Word

Description automatically generated

Browser Output : after clicking button

Note: The above demo can be achieved using “props” however using “state” is much more efficient. Next lets discuss the differences between props and state.

**Props vs State**

Table

Description automatically generated

Differences between props and state

**Component Lifecycle**

There is a component lifecycle to protect the state of the component. The state of the component must not be modified while React is creating the component. So what happens is, component goes into a known state, creates and then opens up the lifecycle for effects, state updates as well as events.

React has 2 lifecycles as given below;

Diagram

Description automatically generated

Lifecycle of React

And within the update lifecycle we have more phases;

Graphical user interface, text, application, chat or text message

Description automatically generated

Update cycle of React

**Render —**the render function must be deterministic and have no side effects. We can consider it as a pure function from props to JSX.

**Pre-Commit**— We can read data from the DOM using getSnapShotBeforeUpdate() method. This would be useful if we want to read data such as scroll position or rendered size of an element before the DOM re-renders.

**Commit —**In this phase, React updates the DOM and refs. We can get access into it using componentDidUpdate or the useEffect hook. Now we can run effects, schedule updates, use the DOM etc.

Diagram

Description automatically generated

React Component Lifecycle Diagram by Dan Abramov ([Source](https://twitter.com/dan_abramov/status/981712092611989509))

**React Hooks**

A picture containing shape

Description automatically generated

React Hooks are basically functions which permits you to access into the React component lifecycle without the usage of class syntax or calling of lifecycle methods directly. Instead of declaring a class we code a render function.

With the use of useEffect()1 we can queue effects and run the effects at any time in the lifecycle of the component. We can archeive this after component mount phase using componentDidMount() , during the commit phase using componentDidUpdate() or before the compoent unmounts using componentWillUnmount() .

With useEffect() we can hook something a component needs when a component mounts and then update it when the component re-draws and also clean up before the component unmounts to prevent memory leaks, all using one function call instead of 3 different method calls.

*Hooks enables :*

* Coding as functional components rather than class components.
* Have clean organized code which is readable.
* Share reusable logic between different components to decrease development time.
* Compose hooks to create custom hooks of your choice by calling a hook from inside of another hook.

React Hooks will be useful as many developers prefer functional components over class components because there is less code, more readable and organized look, reusable and convenient testing.

Timeline

Description automatically generated

React Hooks Lifecyle

**Conclusion**

*Hope you have learnt and gained something from this article about ReactJS and have gain a fundamental knowledge about this increasing popular technology.*