**Let’s learn some ReactJS –**

[Disni Jayawickrama](https://medium.com/@Disni_Jayawickrama?source=post_page-----f2c6a23e5c42--------------------------------) May 14, 2022

Today from this blog article, I would like to go through some basic pieces of information that you must know when you start developing with ReactJS.

First of all, you might be wondering while you are using those confusing words in the programs, isn’t the main thing we should concern is, what is this programming language? What is the history?

Here, let me fill you up.

**What is ReactJS?**

Graphical user interface, diagram

Description automatically generated

React is a free and open-source JavaScript front-end library for creating user interfaces using UI components. It is also known as ReactJS or ReactJS.

The previous Facebook (Meta version) and a few other individual developers and businesses are supporting ReactJS. It allows you to create complicated user interfaces using “components,” which are small, independent pieces of code. ReactJS is a JavaScript library for creating reusable user interface components.

Platform: A web-based platform

React is a JavaScript library for creating user interfaces that are declarative, efficient, and customizable. That also indicates that react is not a framework, but a library.

**React History**

Timeline

Description automatically generated

Jordan Walke was the original creator of React. React now has over 1,000 open-source contributors. We’d like to thank a few folks who have contributed to React and its documentation in the past, as well as worked to maintain them throughout time: This is not an exhaustive list. React Native has a unique backstory: it began as a Facebook internal hackathon project in the summer of 2013 and has since grown into one of the most popular frameworks. React. js hosted the first public preview in January of 2015.

Their code has to be updated immediately in order to become more efficient. They had the proper model, but they needed to improve the user experience. As a result, Jordan Walke created a prototype that made the process more efficient and React. js was born.

Now, we have heard React and there is another word… react native? What is the difference between these two? Wanna know?

**React vs React native**

Graphical user interface, application

Description automatically generated

**DOM provides blazing speed.**

One of ReactJS’ USPs from a business standpoint is its ability to deliver blazing fast speed for applications and websites built with the DOM or Document Object Model.

React builds an in-memory data structure cache, which computes the differences, and then seamlessly changes the browser’s displayed DOM. The programmer writes the code as if each change renders the full page, whereas in reality, only the altered sub-components are rendered. This provides ReactJS an advantage over other platforms for designing mobile apps, as speed is one of the most critical components in creating a successful app.

**Time is saved using component-based architecture.**

ReactJS introduced the concept of component-based design, which enforces component reusability and saves a significant amount of time. Individual components of a broader user interface are converted into self-contained micro-systems using this component-based design. If any component in the Adset tab of Facebook Ads, for example, requires considerable changes, only that component will be reused and re-defined, rather than the entire UI of Facebook Ads.

**SEO Gets a Boost with ReactJS**

Because ReactJS supports server-side rendering, it improves a webpage’s or app’s SEO and draws more organic visitors. Indexing and caching of content increase faster when a Google bot accesses a server that has previously produced the content and graphics. It provides a significant boost to the website’s entire SEO strategy. Furthermore, if the Google bot indexes the content directly from the server, the page load time is reduced. This quick visibility of web pages provides a better user experience for end-users, resulting in a win-win situation.

**Developer Tools Ecosystem Is Massive**

ReactJS is fortunate to have a large developer community that has produced an ecosystem of tools, component libraries, IDEs, extensions for code editors, web browsers, and more. The availability of a diverse set of third-party tools and extensions opens up new avenues for innovation while also saving time and resources. ReactJS is one of the most popular solutions for building innovative, ground-breaking apps and websites because of community-driven innovation.

Virtual DOM is utilized in ReactJS to render browser code, but native APIs are used in React Native to render components on mobile.

ReactJS apps render HTML in the UI, whereas React Native apps render JSX, which is nothing more than JavaScript.

In ReactJS, styling is done with CSS, however, in React Native, styling is done with a stylesheet.

Animation is available in ReactJS using CSS, exactly like in web programming, however, in React Native, an animated API is used to instigate animation across many components of the React Native application.

ReactJS is the best option for building a high-performing, dynamic, and responsive UI for web interfaces, while React is the best option for building a high-performing, dynamic, and responsive UI for web interfaces.

ReactJS is the greatest option for creating a high-performing, dynamic, and responsive user experience for online interfaces, while React Native is the best option for giving mobile apps a fully native feel.

Both ReactJS and React Native are crucial pillars for app and online development, and they’re gaining traction with each passing day because to their flexible features and growing ecosystem of libraries.

While ReactJS is essentially a JavaScript library and React Native is the whole framework, the former is at the heart of the latter and the two operate well together.

If ReactJS is best for generating high-functioning apps with complicated computations, then React Native is best for giving your mobile apps a native feel.

Every developer technology or framework has restrictions, and ReactJS and React Native are no exceptions. As a result, it is recommended that you map the benefits and drawbacks of each of these technologies before making an informed decision based on your priorities and projected outcomes.

Every programing language has a life cycle just like a human being. React also has a component life cycle. Excited to check out? Here we go!

React components are React extensions. The parent class of a component inherits a collection of methods from the react **lifecycle**.

· Render and Commit state

· Controlled and Uncontrolled components.

· Managing side effects.

**Component life cycle**

Diagram

Description automatically generated

● constructor

● render

● componentDidMount

● componentDidUpdate

● componentWillUnmount

● componentDidCatch

Lifecycle of Components Each component in React has a lifecycle that you can monitor and manipulate during its three main phases.

The three phases are Mounting, Updating, and Unmounting.

Diagram

Description automatically generated

**Mounting**

Mounting means putting elements into the DOM. React has four built-in methods that gets called, in this order, when mounting a component:

constructor ()

getDerivedStateFromProps()

render ()

componentDidMount()

**render()**

the method is required and will always be called; the others are optional and will be called if you define them.

**constructor ()**

The constructor () method is called before anything else when the component is initiated, and it is the natural place to set up the initial state and other initial values.

**getDerivedStateFromProps**()

Before rendering the element(s) in the DOM, the getDerivedStateFromProps() method is invoked. Setting the state object depending on the initial props makes sense here.It accepts state as an argument and returns an object containing state changes.

**componentDidMount()**

The componentDidMount() method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

**Updating**

When a component is updated, the lifecycle moves on to the next step. When the state or props of a component change, the component is updated. When a component is changed, React calls the following five built-in methods in this order:

getDerivedStateFromProps()

shouldComponentUpdate()

render()

getSnapshotBeforeUpdate()

componentDidUpdate()

The render() method is necessary and will always be called; however, the other methods are optional and will only be invoked if they are defined.

**getDerivedStateFromProps**

The getDerivedStateFromProps function is also invoked during updates. When a component is modified, the first method that is called is this one.

The natural place to set the state object based on the initial props is still here.

**shouldComponentUpdate**

developer can use the shouldComponentUpdate() method to return a Boolean result that indicates whether or not React should continue rendering.

True is the default value.

**render()**

When a component is modified, the render() method is called because it needs to re-render the HTML to the DOM with the new changes.

**getSnapshotBeforeUpdate()**

developer may access the props and state before the update using the getSnapshotBeforeUpdate() method, which means you can check what the values were before the update even after it has occurred.

developer must include the componentDidUpdate() function if the getSnapshotBeforeUpdate() method is present; otherwise, an error will occur.

**componentDidUpdate()**

The componentDidUpdate method is called after the component is updated in the DOM.

**Unmounting**

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

* componentWillUnmount()

**componentWillUnmount()**

The method is called when the component is about to be removed from the DOM.

I believe a good background explanation is a must to understand all those code parts one by one.