1. **REACT.JS:**

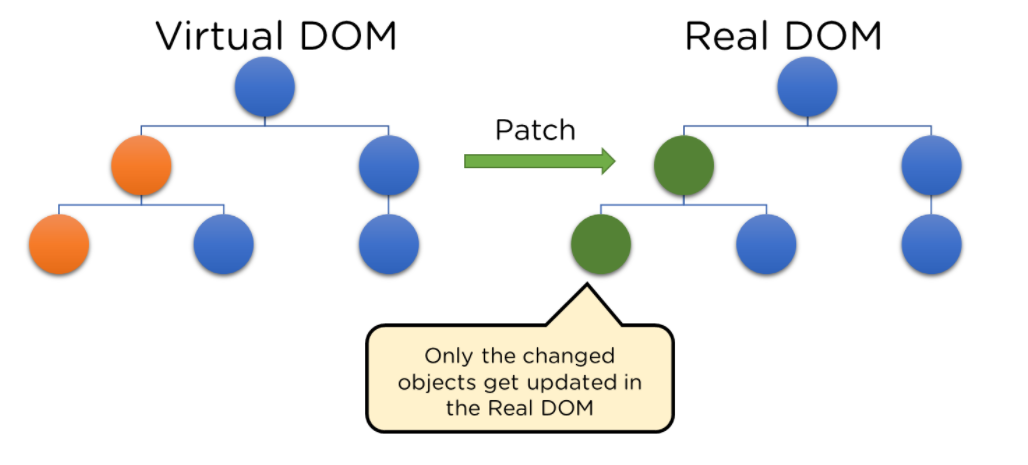
React is a JavaScript-based UI development library. Facebook and an open-source developer community run it. Although React is a library rather than a language, it is widely used in web development. The library first appeared in May 2013 and is now one of the most commonly used frontend libraries for web development. React offers various extensions for entire application architectural support, such as Flux and React Native, beyond mere UI. 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](https://www.simplilearn.com/tutorials/html-tutorial/what-is-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.

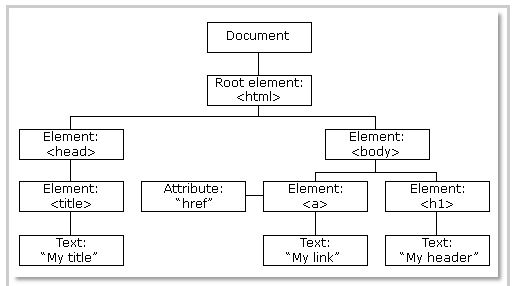
**JSX:** JSX is a JavaScript syntactic extension. It's a term used in React to describe how the user interface should seem.



**Virtual Document Object Model (DOM):** The Virtual DOM is React's a lightweight version of the Real DOM. Real DOM manipulation is substantially slower than virtual DOM manipulation. When an object's state changes, Virtual DOM updates only that object in the real DOM rather than all of them.



* **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?**

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.

### Architecture

In a **Model View Controller(MVC) architecture**, React is the 'View' responsible for how the app looks and feels.

MVC is an architectural pattern that splits the application layer into Model, View, and Controller. The model relates to all data-related logic; the view is used for the UI logic of the application, and the controller is an interface between the Model and View.

### Extensions: React goes beyond just being a UI framework; it contains many extensions that cover the entire application architecture. It helps the building of mobile apps and provides server-side rendering. Flux and Redux, among other things, can extend React.

### Data Binding: Since React employs one-way data binding, all activities stay modular and quick. Moreover, the unidirectional data flow means that it's common to nest child components within parent components when developing a React project.



Fig: One-way data binding

### Debugging: Since a broad developer community exists, React applications are straightforward and easy to test. Facebook provides a browser extension that simplifies and expedites React debugging.

## Components in React: Components are the building blocks that comprise a React application representing a part of the user interface. React separates the user interface into numerous components, making debugging more accessible, and each component has its own set of properties and functions. Here are some of the features of Components -

* Re-usability - A component used in one area of the application can be reused in another area. This helps speed up the development process.
* Nested Components - A component can contain several other components.
* Render method - In its minimal form, a component must define a render method that specifies how the component renders to the DOM.
* Passing properties - A component can also receive props. These are properties passed by its parent to specify values.

## Props in React

Props, short for Properties in React Props, short for properties, allow the user to pass arguments or data to components. These props help make the components more dynamic. Props in a component are read-only and cannot be changed.

## State in React

A state is an object that stores properties values for those attributed to a component that could change over some time.

* A state can be changed as a result of a user's action or changes in the network.
* React re-renders the component to the browser whenever the state of an object changes.
* The function Object() { [native code] } is where the state object is created.
* Multiple properties can be stored in the state object.
* this.
* setState() is used to alter the state object's value.
* The setState() function merges the new and prior states shallowly.

**Props vs. State in React**

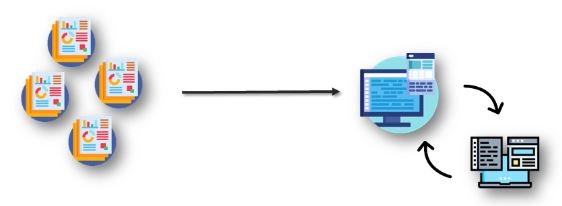
|  |  |
| --- | --- |
| Props | State |
| Props are used to send data and event handlers to a component's children. | The data of the components that must be presented to it store the view in the state. |
| Props are immutable — they can't be modified after they've been set. | The data is stored in the state, which might change over time. |
| Both functional and class components can benefit from the use of props. | Only class components can use the state. |
| The parent component sets props for the children components. | Event handlers are typically responsible for updating the state. |

1. b) REDUX : **Redux**

Redux is a predictable state container for [JavaScript](https://www.simplilearn.com/reasons-to-learn-javascript-article) apps. As an application grows, it becomes difficult to keep it organized and maintain data flow. That is where Redux comes to the rescue.

**Why Redux?**

State transfer between components is pretty messy in React since it is hard to keep track of which component the data is coming from. It becomes really complicated if users are working with a large number of states within an application.



Redux solves the state transfer problem by storing all of the states in a single place called a store. So, managing and transferring states becomes easier as all the states are stored in the same convenient store. Every component in the application can then directly access the required state from that store.

What is Redux?

Redux is a predictable state container for JavaScript applications. It helps you write apps that behave consistently, run in different environments (client, server, and native), and are easy to test. Redux manages an application’s state with a single global object called Store.

Redux is a state management tool

Redux can be used with any JavaScript framework or library

Redux stores the state of the application, and the components can access the state from a state store

**Principles of Redux**

The three most important Redux principles are:

* **Single Source of Truth**

The state of your whole application is stored in an object tree within a single-store.

|  |  |
| --- | --- |
| state-tree | A single state tree makes it easier to debug or inspect an application  It gives you a faster development cycle by enabling you to persist in your app's navigation state |

* **State Is Read-Only**

The only way to change the state is to initiate an action, an object describing what happened.

|  |  |
| --- | --- |
| read | This feature ensures that no events like network callbacks or views can change the state. They can only express an intent to change the state  Actions are just plain objects; they can be logged, serialized, stored, and later replayed for debugging or testing purposes |

* **Changes are Made with Pure Functions**

To specify how actions transform the state tree, you need to write pure reducers.

|  |  |
| --- | --- |
| user | * The user can return new state objects instead of mutating the previous state * The user can start with a single reducer, and, as the app grows, can split it off into smaller reducers that manage specific parts of the state tree * Because reducers are just functions, it’s easy to control the order in which they are called, pass additional data, or even make reusable reducers |

**Pillars of Redux**

These are Redux’s main pillars:

1. Store

A store is an object that holds the application's state tree. There should only be a single store in a Redux app, as the composition happens at the reducer level.

getState() returns the current state of the store.



dispatch() dispatches an action. It is the only way to update the application state.



subscribe() subscribes a change listener to the state.

unsubscribe() is useful when you no longer want to call your listener method when the state changes.



* **Action**

An action is a plain object that represents an intention to change the state. They must have a property to indicate the type of action to be carried out.

* Actions are payloads of information that send data from your application to your store.
* Any data, whether from UI events or network callbacks, needs to eventually be dispatched as actions.
* Actions must have a type field, indicating the type of action being performed.
* Reducers

Reducers are pure functions that specify how the application's state changes in response to actions sent to the store.

* Actions only describe what happened, not how the application's state changes.
* A reducer is a function that accepts the current state and action, and returns a new state with the action performed.
* combineReducers() utility can be used to combine all the reducers in the app into a single index reducer which makes maintainability much easier.

**Pros and Cons of Redux**

Pros

There is always one source of truth, the store. Thus, there is no confusion about how to sync the current state with actions and other parts of the application.

The code is easier to maintain because it has a predictable outcome and strict structure.

Redux makes coding more consistent due to more stringent code organization procedures

It’s very useful, especially during the initial render, making for better user experience and search engine optimization.

Developers can track everything going on in the app in real-time—from actions to state changes

Cons

Since it has no encapsulation, any component can access data, which may potentially cause security issues

Some parts of the code are just boilerplate. However, these parts have to be incorporated with no alteration, and this restricts the design

As the state is immutable in Redux, the reducer updates the state by returning a new state every time which can cause excessive use of memory

1. JAVASCRIPT :

**JavaScript** is a lightweight, interpreted **programming** language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. **JavaScript** is very easy to implement because it is integrated with HTML. It is open and cross-platform.

## Why to Learn Javascript

**Javascript** is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Javascript:

* Javascript is the most popular **programming language** in the world and that makes it a programmer’s great choice. Once you learnt Javascript, it helps you developing great front-end as well as back-end softwares using different Javascript based frameworks like jQuery, Node.JS etc.
* Javascript is everywhere, it comes installed on every modern web browser and so to learn Javascript you really do not need any special environment setup. For example Chrome, Mozilla Firefox , Safari and every browser you know as of today, supports Javascript.
* Javascript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience.
* JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as Javascript Programmer.
* Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market.
* Great thing about Javascript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

There could be 1000s of good reasons to learn Javascript Programming. But one thing for sure, to learn any **programming language**, not only Javascript, you just need to code, and code and finally code until you become expert.

## Hello World using Javascript

Just to give you a little excitement about **Javascript programming**, I'm going to give you a small conventional Javascript Hello World program, You can try it using Demo link

[Live Demo](http://tpcg.io/aMKLYm)

<html>

<body>

<script language = "javascript" type = "text/javascript">

<!--

document.write("Hello World!")

//-->

</script>

</body>

</html>

There are many useful **Javascript frameworks** and libraries available:

* Angular
* React
* jQuery
* Vue.js
* Ext.js
* Ember.js
* Meteor
* Mithril
* Node.js
* Polymer
* Aurelia
* Backbone.js

It is really impossible to give a complete list of all the available Javascript frameworks and libraries. The Javascript world is just too large and too much new is happening.

## Applications of Javascript Programming

As mentioned before, **Javascript** is one of the most widely used **programming languages** (Front-end as well as Back-end). It has it's presence in almost every area of software development. I'm going to list few of them here:

* **Client side validation** - This is really important to verify any user input before submitting it to the server and Javascript plays an important role in validting those inputs at front-end itself.
* **Manipulating HTML Pages** - Javascript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using javascript and modify your HTML to change its look and feel based on different devices and requirements.
* **User Notifications** - You can use Javascript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
* **Back-end Data Loading** - Javascript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.
* **Presentations** - JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides RevealJS and BespokeJS libraries to build a web-based slide presentations.
* **Server Applications** - Node JS is built on Chrome's Javascript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

This list goes on, there are various areas where millions of software developers are happily using Javascript to develop great websites and others softwares.

4) Your email ID & Phone Number to connect .