International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 04 | April -2020 ISSN: 2582-3930

Web Development Using ReactJS

Neethushree V[1]
PG Scholar, Dept of MCA, DSCE
Bangalore, India

Vibha M B[2]
Assistant Professor, Dept of MCA, DSCE
Bangalore, India

Abstract

ReactJS is a component-based JavaScript library that is sent for the improvement of intuitive UIs. Presently it is the most drifting front-end JS library. It consolidates the View (V) layer in M-V-C (Model View Controller) design. It is upheld by Facebook, Instagram and a network of individual engineers and associations. ReactJS fundamentally empowers the improvement of huge and complex electronic applications that can change its information without resulting page revives. It focuses to furnish better client encounters and with bursting quick and powerful web applications advancement. ReactJS can likewise be coordinated with other JavaScript libraries or structures in MVC, for example, AngularJS.

Keywords: ReactJS, Single Page Application, MVC, Virtual DOM, Component Life Cycle.

I. Introduction

ReactJS is one of the JavaScript libraries that is used to create reusable (UI) components. As per ReactJS official documentation, the following is the definition ReactJS is a library for building particular UIs. ReactJS essentially empowers the improvement of enormous and complex electronic applications that can change its information without ensuing page revives. It is utilized as the View (V) in the Model-View-Controller (MVC). It abstracts the Document Object Model (DOM), accordingly offering a straightforward, performing and vigorous application improvement experience. ReactJS for the most part renders on the server-side utilizing NodeJS, and backing for local portable applications is offered utilizing React Native. ReactJS executes unidirectional information stream in this way streamlining the standard and subsequently ends up being a lot simpler than conventional information official.

Single Page Application

ReactJS is a Single Page Application consists of individual components which can be updated/replaced independently, without reloading/refreshing entire page. Whole page is divided into smaller components that interact with each other. A component of a page updated with any changes on user requests.

II. Characteristics

1. Lightweight Document Object Model)

ReactJS gives a much proficient and lightweight DOM. It doesn't connect with the DOM created by the browser yet responds to the DOM that is put away in the memory. This outcomes in a blasting quick and hearty execution of the application. In other web development frameworks, direct connection with the browser DOM is made which brings about direct whole DOM tree control on each page activating events. Accordingly, when an enormous lump of information is to be changed, the presentation gets seriously influenced. Despite what might be expected, ReactJS utilizes something known as a virtual DOM. Its working is very straightforward. Correlations with virtual DOM and real DOM are made utilizing diff() algorithm and just the node changes the along these lines reflected in the DOM.

2. JSX

JavaScript eXtension is a language that is fundamentally same as the tech monster XML. It isn't at all compulsory to utilize JSX while building up a ReactJS based application yet it is exceptionally well-known between the engineers as it is a shorthand that makes development simple, at whatever point they are composing mark-ups for components and the corresponding binding events. It is the inclination of human instinct to decide on satisfying and non-complex strategies that makes JSX seriously well known.

3. Optimizing Performance

Internally, ReactJS utilizes a few astute techniques to limit the quantity of expensive DOM activities required to refresh the UI. For some, applications, utilizing React will prompt a quick UI without accomplishing a lot of work to explicitly upgrade for execution. ReactJS maintains a Virtual DOM inside

the memory. At whatever point a change is to be reflected to the presently showed website page, rather than in a flash refreshing the browser DOM, first changes to virtual DOM are made. After changes to virtual DOM are made, a diff() algorithm is applied which analyses the two, the virtual DOM and the browse DOM and just pertinent and wanted nodes of browser DOM tree are refreshed, which brings about blasting quick execution of the application.

4. One-way Binding

ReactJS is one-way data flow (also called one-way binding) which permits and supports downstream and keeps everything modular and fast. In the event that the bidirectional information stream is required, that extra features should be executed. This will be done as the components should be unchanging and information inside them must not change under any conditions. In this manner, tune in to information coming one way just is made, not the other. Consequently, ReactJS is notable for the age of authoritative information sources that remaining parts synchronized over the components that focus on it. Therefore, it ends up being probably the best framework to create an interactive web app. On the off chance that a specific change is to be made on the upstream information, the parts utilizing that information will consequently re-render in want to mirror the changes. This is the motivation behind why they must be in synchronization with the information that is streaming downstream. A relative way of data binding is given by Flux in React.js, which is an option in contrast to the basic Model View Controller (MVC).

5. Virtual DOM

The VDOM of ReactJS is like the DOM produced by the browser however with a distinction that it is put away in memory. At whatever point a solicitation for changing the page content is made, the progressions are reflected in the memory living virtual DOM first. After that a diff() algorithm analyzes the two for example the virtual DOM and the browser DOM and afterward the necessary changes just are reflected in the browser DOM, rather than re-rendering the whole DOM. This gives a tremendous lift to the presentation of the application, basically when a huge number of information changes are to be made.

III. Working

The model view controller (MVC) design pattern is widely used and fundamental to the UI development in web applications and front-end applications running on any platform. In case of web applications physical View is represented by DOM. The formation of is DOM via a HTML template which itself is yield from a script block, different file or a precompiled template function. The View entity itself gives life to textual template as a DOM. It plays a key role of handling the Events and manipulating the DOM tree as a neighborhood of its life cycle. A view can only be useful if and as long as it makes the user interaction possible also as display the specified data.

Data is an entity that's brought from some Data-Store, which might be a Database, a web service or an area Store. Frameworks provide how to bind view to the info store so as to form sure that changes made to the database are automatically reflected back. This process of automatic data updates pushing is usually mentioned as Data-Binding. There are many application interfaces or API's which make this process merely a cakewalk.

As demonstrated in figure 1, the M-V-C paradigm is completed by the C component i.e. the Controller which engages the remainder two components i.e. the Model and therefore the View and enables the info Model flow into the View and user events out of View, eventually resulting in changes within the Model.

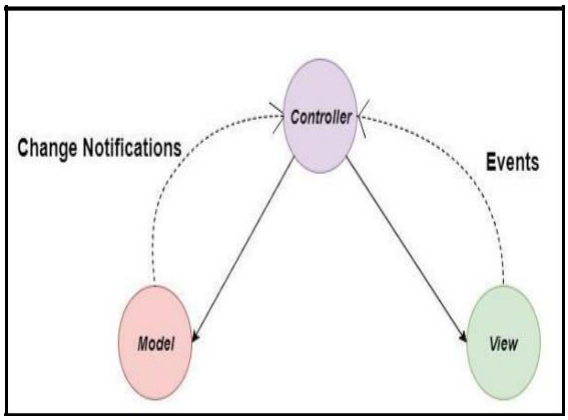

Fig.1 Interaction between M-V-C Component

So as to see how React deals with these assignments, a lot further comprehension of parts is required, beginning with the Component. In ReactJS the major building-block is Components. The whole UI can be planned by amassing a tree of various components. An intermediate DOM is produced by the render() function present in every one of the
React component. A Call to the React.renderComponent() function on the root Component prompts recursively going down the Component-tree and producing the intermediate DOM. At that point a while later this intermediate DOM is changed over to the genuine HTML DOM.

ReactJS utilizes an advantageous XML-based expansion to JavaScript, known as the JSX, to create the component tree as a mix of different XML nodes. This makes DOM perception and seeing simple and substantially more advantageous. JSX additionally assumes a key function in streamlining the

relationship of the properties and event handlers as XML attributes. Using a in browser tool and command line final JavaScript will be generated. A Component is straightforwardly mapped by a JSX XML node. It is imperative to take note of that ReactJS works independent of JSX and the utilizing JSX just plays the undertaking of streamlining the assignment of creating intermediate DOM.

Component Life Cycle

Each Component in the ReactJS framework contains a state-machine that has three states and have an exceptionally specific lifecycle. The Mounting procedure offers life to a Component. When Mounting is gone through a render-pass the Intermediate DOM or component tree is called. This tree is then changed over and put into a container
node of the real DOM. When React.renderComponent() function is called all the above process takes place.

As demonstrated in fig 2, the Component stays in the Update state once the mounting has been done. If the changes are done in the state using setState() or props using setProps() method then the component gets updated. Then the render() method is called which synchronizes the DOM with the information for example props and state. RecatJS computes the delta between the past component tree and the recently created tree between consequent updates.

This step is a flagship feature and highly optimized that limits the real DOM. The last state is then Un-mounted. This results directly if a component that will in general be a child is never again produced in a render() method call. Usually developers don't need to stress over this and just permit ReactJS to do the required.

Now it might be an enormous remiss, if ReactJS did not notified when it is switched between the Mounted, Update and Un-mounted states. Yet, that isn't the situation and hooks are given which can be replaced so as to inform at whatever point a state change occurs.

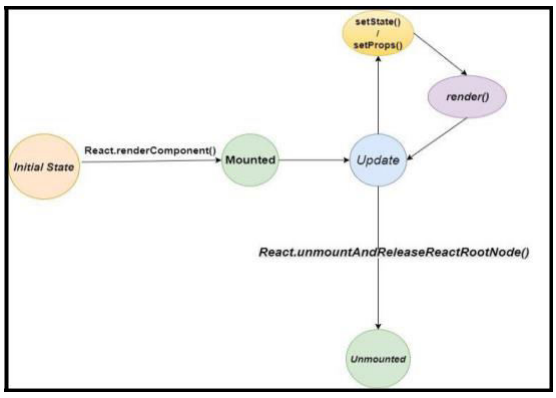

Fig.2 React Component Life Cycle

A. Implementation of Sample ReactJS Application

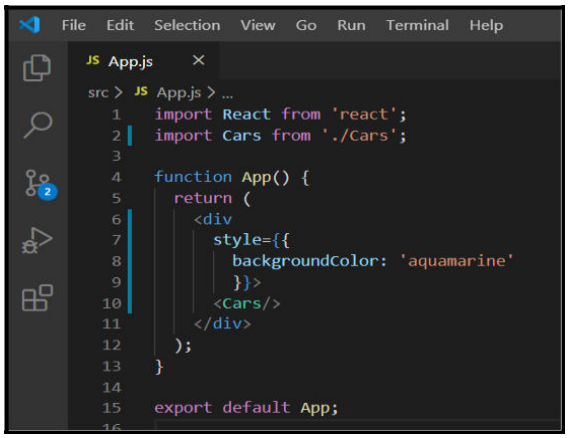

Fig.3 App.js (View Component that controls entire app/page)

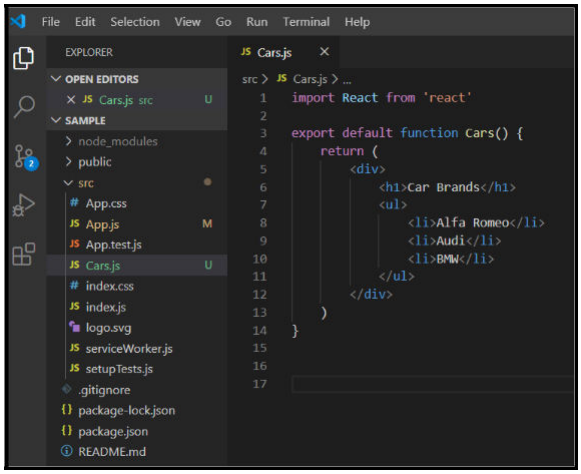

Fig.4 Cars.js (Functional Component is just a plain JS function)

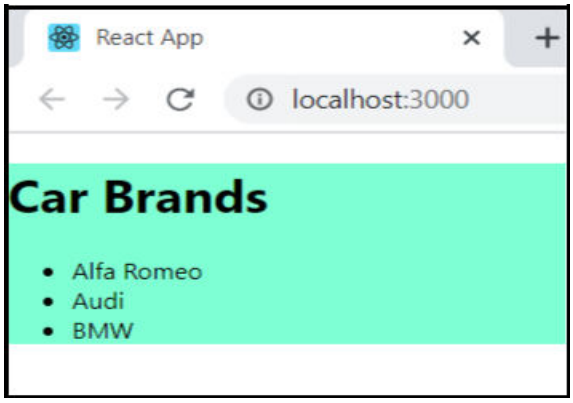

Fig.5 Output of "Sample" ReactJS Application

IV. The Difference

Unlike the MVC paradigm (Model, View, Controller), ReactJS has more prominence to the View part and it is stuffed into an entity called the Component. An unchangeable property(props) and a state that expresses the user-driven state of the UI is managed by the Component.

The View is the component in ReactJS making it more fascinating and satisfying and makes it stand apart from the all the frameworks like AngularJS, Ember.js. Rather than legitimately interacting with the browser DOM, Virtual DOM is kept up inside the memory which after examination with the real browser DOM drives changes into the actual DOM.

The integral part of the intermediate DOM generation is data binding and event handling. In the case of interpreted languages, relative techniques are used by language runtimes (otherwise known as Virtual Machines). Initially JavaScript runtime creates an intermediate design before the native code is used exhibited. ReactJS produces an intermediate DOM before creating the final HTML DOM. The intermediate DOM is only a JavaScript object graph and isn't rendered instantly. The actual DOM is produced after an interpretation procedure. This is the method that empowers blasting quick DOM manipulation and makes ReactJS stand apart of different frameworks in the market.

V. Limitations

ReactJS has a couple of restrictions too that must be considered before ReactJS is picked for any web development. These are:

- Only the View element in the portable view controller or MVC is taken care of by the ReactJS. Subsequently extra tooling is required so as to finish the project development.
- The utilization of JSX and inline templates some of the time ends up being a seriously mind-boggling and tiring undertaking for a couple of set of engineers.
- Also, on account of ReactJS, errors occur at compilation time rather than runtime as on account of different frameworks and languages, which can some of the time be exceptionally baffling and tiring.

VI. Conclusion

Despite a couple of minor disadvantages that the ReactJS has, it's definitely a sure shot game changer. The latest web is trending with more user interactive and dynamic. User experience design patterns are consistently changing and developing. The client scripts now make sure that only necessary and essential data is pushed, and a seamless and pleasing experience is maintained across the whole web. It's today's world's demand for greater accessibility, ease and efficiency. ReactJS has an exceptional force and highlights to meet the necessities of the present trends. During a nutshell, it can say that ReactJS is certainly getting to impact the way apps are written for the web.

References

1. ReactJS Official Website [Online] Available: https://reactjs.org/
2. Wikipedia.org, 'React (JavaScript Library)'. [Online]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library)
3. Quora.com, 'MEAN VS. MERN'. [Online]. Available: https://www.quora.com/How-is-MERN-stack-compared-to-MEAN-stack
4. Angular.io, Angular Documentation'. [Online]. Available: https://angular.io
5. MongoDB.com,'MongoDBofficial'. [Online]. Available: https://www.mongodb.com/.
6. ExpressJS.com, 'Express official'. [Online]. Available: http://expressjs.com
7. NodeJS.org, 'NodeJs official'. [Online]. Available: http://nodejs.org.
8. Github.com,'ReactDocumentation'. [Online]. Available: https://github.com/facebook/react
9. Draw.io," Flowchart Making Tool". [Online]. Available: https://www.draw.io/#.
10. React: Up and Running: Building Web Applications, Book by Stoyan Stefanov