Sanchit Aggarwal et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 5, Issue 1, March 2018, pp. 133-137

Modern Web-Development using ReactJS

Sanchit Aggarwal

Department of Information Technology, Northern India Engineering College, New Delhi, India

sanchit.niec@gmail.com

***Abstract* — ReactJS is a component based library which is deployed for the development of interactive user interfaces. Currently it is the most popular front-end JS library. It incorporates the view (V) layer in M-V-C (Model View Controller) pattern. It is supported by Facebook, Instagram and a community of individual developers and organisations. React basically enables development of large and complex web based applications which can change its data without subsequent page refreshes. It targets to provide better user experiences and with blazing fast and robust web apps development. ReactJS can also integrated with other JavaScript libraries or frameworks in MVC, such as AngularJS.**

***Keywords* — Application development, React, ReactJS, MVC, virtual DOM, Front-end development.**

I. INTRODUCTION

ReactJS is JavaScript library which is deployed to develop reusable user interface (UI) components. According to React official documentation, following is the definition React is a library for building modular user interfaces [9]. React basically enables development of large and complex web based applications which can change its data without subsequent page refreshes. It is used as the View (V) in the Model-View-Controller(MVC). React abstracts the Document Object Model (DOM), thus offering a simple, performing and robust application development experience. React mostly renders on server side using NodeJS, and support for native mobile apps is offered using React Native. React implements unidirectional data flow thus simplifying the boilerplate and hence proves to be much easier than traditional data binding.

II. FEATURES

A) Lightweight DOM(Document Object Model) For Better Performance

React provides a much efficient and light weight document object model. It does not interact with the DOM generated by the browser but reacts to the document object model stored in the memory. This results in a blazing fast and robust performance of the application. In most of the other web development frameworks, direct interaction with the browser DOM is made which results in direct entire DOM tree manipulation on each and every page triggering event. As a result, when a large chunk of data is to be modified, the performance gets intensively affected. On the contrary, ReactJS uses something known as a virtual DOM. The working of it is quite simple. Comparisons to virtual DOM and actual DOM are made using an diff() algorithm and only the nodes changes the thus reflected back into the document object model tree[11].

B) Easy Learning Curve

The easy and non-complex nature of ReactJS enables one to quickly get comfortable with the framework. The learning curve is extremely easy and gets one along without any complications. The architecture is intensively easy the idea of using JSX feels to be an entirely natural and pleasing phenomenon that a developer gets along with the framework very easily. Initial levels of expertise in the framework can easily be achieved without any hindrances or complications.

C) JSX

JSX is a language which is very similar to the tech giant XML. It is not at all mandatory to use JSX while developing a react based application but it is highly popular between the developers as it is a short hand that makes development easy, whenever they are writing mark-ups for components and the corresponding binding events. It is the tendency of human nature to opt of pleasing and non complex techniques that makes JSX intensively popular.

D) Performance

ReactJS is known to be a highly efficient performer. This is one of the key factors that makes the frameworks stand out of dozens of frameworks out there in the competitive world. The reason for highly efficient performance of the framework is essentially is the virtual DOM feature of the framework. What happens is that ReactJS maintains a virtual document object model inside the memory. Whenever a change is to be reflected to the currently displayed webpage, instead of instantly updating the browse DOM, first changes to virtual DOM are made. After changes to virtual DOM are made, a diff() algorithm is applied which compares the tow, the virtual DOM and the browse DOM and only relevant and desired nodes of browser DOM tree are updated, which results in blazing fast performance of the application.

E) *One way data flow*

ReactJS id designed in such a way that unidirectional data flow that is downstream is allowed and supported. Id bidirectional data flow is required, that additional features needs to be implemented. This was done as the components need to be immutable and data within them must not change under any circumstances. Thus, listen to data coming in one direction only is made, not the other. Henceforth React.js is well-known for the generation of canonical data sources that remains synchronized across the components that which pay attention to it. As a result, it proves to be one of the best framework to develop interactive web based applications. If a certain change is to be made on the upstream data, the components using that data will automatically re-render in hope to reflect the changes. This is the reason why they have to be in synchronisation with the data that is flowing downstream. Similar style of data binding is provided by Flux in React.js, which is an alternative to the common model view controller (MVC).

F) Virtual DOM

Another key feature of ReactJS is the virtual DOM (Virtual document object model) of ReactJS. It is similar to the document object model generated by the browser but with a difference that it is stored it memory. The working of virtual DOM is quite simple. Whenever a request for changing the page content is made, the changes are reflected to the memory residing virtual DOM first. After that a diff() algorithm compares the two i.e. the virtual DOM and the browser DOM an then the required changes only are reflected to the browser DOM, instead of re-rendering the entire DOM. This provides a gigantic boost to the performance of application, essentially when thousands of data changes are to be made.

III. WORKING

The model view controller or M-V-C design paradigm is popular and fundamental to the user interface development, not only in web apps but in front-end applications running on any platform. DOM represents physical View in case of web-applications. The DOM is made via a HTML template which itself is fetched from a different file, script block or a precompiled template function. The textual template is given life as a DOM by the View entity itself. It plays a key role of handling the Events and manipulating the document object model tree as a part of its life cycle. A view can only be useful if and only if it makes the user interaction possible as well as display the required data.

Data is an entity that is brought from some Data-Store, which could be a Database ,a web service or a Local Store. Frameworks provide a way to bind view to the data store in order to make sure that changes made to the database are automatically reflected back. This process of automatic data updates pushing is commonly referred to as Data-Binding. There are many application program interfaces or API's which make this process merely a cakewalk.

As shown in figure 1 [1-2], the M-V-C paradigm is completed by the C component i.e. the Controller which engages the rest two components i.e. the model and the view and enables the data model flow into the View and user events out of View, eventually leading to changes in the Model.

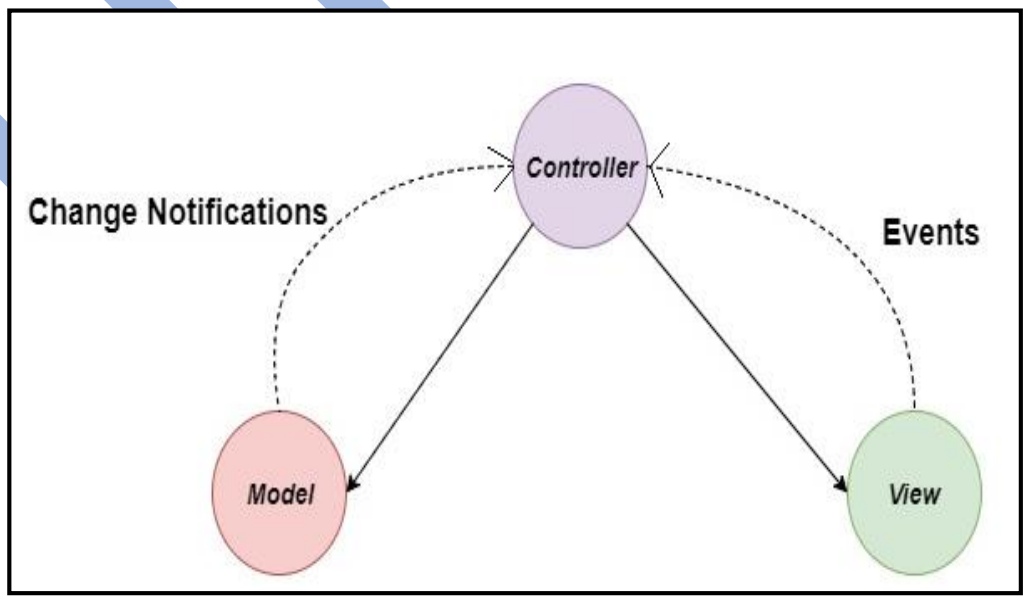

Fig. 1 Interaction between M-V-C components

In order to understand how React manages all of these tasks, a much deeper understanding of components is required, starting with the Component. The Component is the major building-block in React. The entire user interface can be designed by assembling a tree of multiple components. An intermediate DOM is generated by

the render() method present in each of the React components. As shown in figure 2, a Call to the React.renderComponent() method on the root Component leads to recursively going down the Component-tree and generating the intermediate-DOM. Then afterwards this intermediate DOM is converted to the actual HTML DOM.

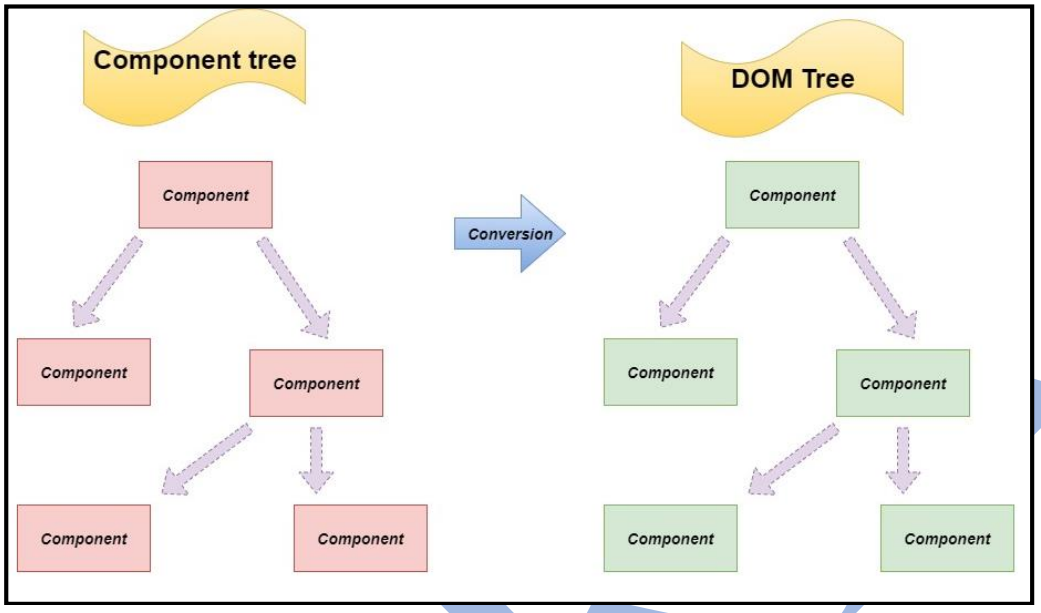

Fig. 2 Conversion of intermediate DOM to HTML DOM

React makes use of a convenient XML-based extension to JavaScript, known as the JSX, to generate the component tree as a combination of multiple XML nodes. This makes DOM visualization and understanding easy and much more convenient. JSX also plays a key role in simplifying the association of the event handlers and properties as XML attributes. The final JavaScript is generated using a command line and in browser tool. A Component is directly mapped by a JSX XML node. It is important to note that React works independent of JSX and the using JSX only plays the task of simplifying the task of generating intermediate DOM.

Component Life Cycle

All of the Components in the ReactJS framework have a very particular lifecycle and contains a state-machine that has three distinct states.

The Mounting process gives life to a Component. When Mounting is passed through a render-pass the component tree or the Intermediate DOM is generated. This tree is then converted and placed into a container-node of the real document object model. All of this process happens when React.renderComponent() method is called.

As shown in figure 3, after mounting has been done, the component remains in the Update state. Any component gets updated when state is changed using the setState() method or when properties are changed using setProps() method. This is followed by a call to the render () method, which synchronizes the document object model with the data i.e. props and state. React calculates the delta between the previous component-tree and the newly generated tree between subsequent updates.

This step is highly optimized and a flagship feature that minimizes the real DOM manipulation. The final state is then Un-mounted. This happens automatically if a component that tends to be a child is no longer generated in a render() call. Most often developers don't have to worry about this and just allows React do the required.

Now it would have been a big remiss, if React didn't informed when it switched the places between the Mounted-Update-Un-mounted states. But that is not the case and hooks are provided which can be over-ridden in order to notify whenever a state change happens.

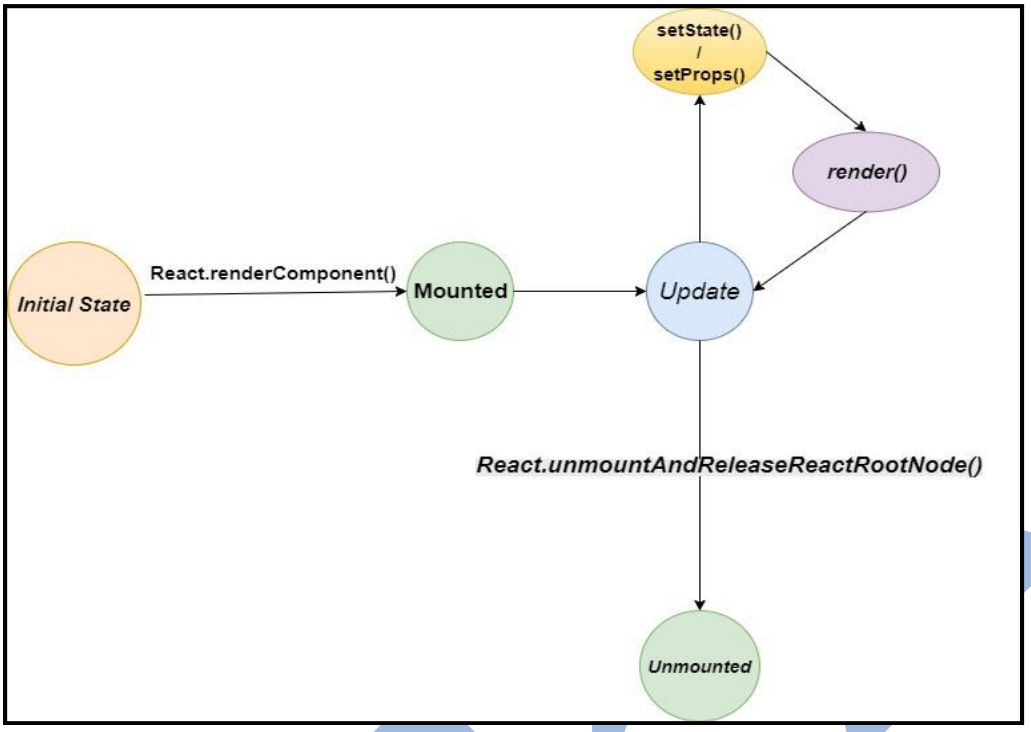

Fig. 3 The React Component life cycle

IV. THE DIFFERENCE

With React, out of the Model, View and Controller in MVC paradigm, the View-part has prominence and is packed into an entity called the Component. An immutable property set called props and a state that represents the user-driven state of the User interface is maintained by the Component. It is the view generation part of the component in React that makes it intensively interesting and pleasing and makes it stand out of all of the frameworks present in the market. Instead of directly interacting with the browser DOM , a virtual DOM is maintained inside the memory which after comparison with the actual browser DOM pushes changes into the real DOM.

Event handling and data binding is performed as an integral part of the intermediate DOM generation. Similar strategies are adopted by language runtimes (aka Virtual Machines) in case of interpreted languages. The JavaScript runtime at the very first generates an intermediate representation before the native code is presented. React cleverly generates an intermediate document object model before generating the final HTML DOM. The intermediate document object model is nothing but a JavaScript object graph and is not rendered directly. The real document object model is generated after a translation process. This is the technique that enable blazing fast DOM manipulations and make react stand out of other frameworks in the market.

V. LIMITATIONS

React has a few limitations too that must be considered before React is chosen for any project development. These are:

- Only the View entity in the mobile view controller or MVC is handled by the react. Thus additional tooling is required in order to complete the project development.
- Use of inline templates and JSX sometimes proves to be an intensively complex and tiring task for a few set of developers.
- Also, in case of ReactJS, failures happen at compile time instead of runtime as in case of other languages and frameworks, which can sometimes be very frustrating and tiring.

VI. CONCLUSIONS

Despite of a few minor disadvantages that that ReactJS has, it is definitely a sure shot game changer. Modern web is day to day becoming more dynamic and user interactive. User experience design trends are continuously changing and evolving. The client scripts now makes sure that only necessary and essential data is pushed, and a seamless and pleasing experience is maintained across the entire web. It is today's world's demand for ease, efficiency, and greater accessibility. ReactJS has intense power and features to meet requirements of today's trends. In a nutshell, it can said that ReactJS is definitely going to impact the way apps are written for the web.

ACKNOWLEDGMENT

This work would not have been possible without the support of Mrs. Jyoti Verma, faculty of Information Technology , Northern India Engineering College, New Delhi who have been intensively supportive not only during this research as but also of my career goals and worked actively to provide me with the protected academic time to pursue those goals. As my teacher and mentor, she has taught me more than I could ever give her credit for here. She has shown me, by her example, what a good teacher (and person) should be. Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models.

REFERENCES

[1] Wikipedia.org, 'React (JavaScript Library)'. [Online]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Accessed: Feb- 2018]
[2] MongoDB.com, MEAN and MERN stacks'. [Online]. Available: https://www.mongodb.com/blog/post/the-modern-application-stackpart-1-introducing-the-mean-stack [Accessed: Feb- 2018].
[3] Google trends,'Google Trends',[Online]. Available : https://www.trends.google.com . [Accessed: Feb- 2018]
[4] Quora.com, 'MEAN VS. MERN'. [Online]. Available: https://www.quora.com/How-is-MERN-stack-compared-to-MEANstack. [Accessed: Feb- 2018]
[5] Angular.io, Angular Documentation'. [Online]. Available: https://angular.io. [Accessed: Feb- 2018]
[6] MongoDB.com, 'MongoDB official'. [Online]. Available: https://www.mongodb.com/. [Accessed: Feb- 2018]
[7] ExpressJS.com, 'Express official'. [Online]. Available: http://expressjs.com. [Accessed: Feb- 2018]
[8] NodeJS.org, 'NodeJs official'. [Online]. Available: http://nodejs.org. [Accessed: Feb- 2018]
[9] ReactJS.org,' ReactJS official'. [Online]. Available: http://www.ReactJs.org. [Accessed: Feb- 2018]
[10] Github.com, 'React Documentation'. [Online]. Available: https://github.com/facebook/react. [Accessed: Feb- 2018]
[11] Draw.io,"Flowchart Making Tool".[Online].Available: https://www.draw.io/#. [Accessed: Feb- 2018]
[12] React:Up and Running: Building Web Applications,Book by Stoyan Stefanov
[13] Getting MEAN With Mongo, Express, Angular, and Node Book by Simon Holmes
[14] Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React,NODE Book by Vasan Subramanian

