International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 5 Issue 4, May-June 2021 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Review on React JS

Bhupati Venkat Sai Indla[1], Yogeshchandra Puranik[2]

[1]PG Student, [2]Professor,
[1,2]MCA, P.E.S. Modern College of Engineering, Pune, Maharashtra, India

ABSTRACT
This is a review on react js. Its introduction, how to use it, why to use it. Its uses in the front-end development world and its effectiveness and advantages as well.

KEYWORDS: *React, React js, web development, frontend web dev*

How to cite this paper: Bhupati Venkat Sai Indla | Yogeshchandra Puranik "Review on React JS" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-4, June 2021, pp.1137-1139, URL: www.ijtsrd.com/papers/ijtsrd42490.pdf

IJTSRD42490

INTRODUCTION:

React (also known as React.js or ReactJS) is an open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.

Web before Reactjs

Let's take a walk down to the technology space before 2015 when web development was all about scripting and rendering. The time when languages like HTML, CSS ruled the frontend, and PHP ruled the backend.

Web development was so easy back then. All we needed to do was to put static HTML pages in some folders and render them using PHP. Although that's not a unique and intuitive way to develop websites, you were still able to establish a two-way connection between client and server. All the credit goes to Server-Side Rendering (SSR). We've been building web applications this way for decades, but what we didn't see coming is the revolution of websites after Javascript libraries like Reactjs.

The dawn of Single page apps (SPA), Javascript, and Reactjs

Ever since the Javascript revolution took over, you can do a lot more with Javascript than you could ten years ago.

So what brings the change?
The answer is: writing web apps with client-side Javascript.

Yes, we are referring to the development of Single Page Apps (SPA) using Javascript. While many Javascript frameworks let you write client-side javascript, Angular was the only one that promoted this approach.

Imagine being able to fetch some data via Javascript, add some attributes to your markup, and voila!: you have built a dynamic website without messing up with PHP and servers.

But, no matter how popularized this approach seemed to be, DOM manipulations (a way to render several components) remained not so fast.

Enter Reactjs!

The Isomorphic javascript library, introduced in 2015, enabled developers to build dynamic web applications with blazing speed.

React was primarily used to render views in web or mobile applications. It allowed developers to create reusable components that are independent of each other. So when any critical feature of a web application broke, they were still better off with the remaining elements. Also, React brought this fantastic feature called Virtual DOM that enabled developers to implement SSR without needing to update the whole view each time during an update.

What's so revolutionary about this, you ask?

For instance, while building a dynamic front-end or a SPA, you also want client-side routing to ensure a quick navigational experience for the end-user. Now, navigation is not something you would want to lose when you implement SSR. Thankfully, you can still use Reactjs on the client-side and implement navigation. What this means is that the initial render uses SSR, and the subsequent navigations behave like a SPA. Also, with React, you are not moving away with SSR; you are just utilizing it whenever needed.

To sum this up: Reactjs shines in building dynamic and engaging web interfaces and triumphs over other javascript frameworks (such as Angular, Ember). The reason is: Virtual DOM facilitates updating components whenever a user does any interaction without affecting other parts of the interface.

Why Do JavaScript Developers Use React JS?
React is a JavaScript library that specializes in helping developers build user interfaces, or UIs. In terms of websites and web applications, UIs are the collection of on-screen menus, search bars, buttons, and anything else someone interacts with to USE a website or app.

Before React JS, developers were stuck building UIs by hand with "vanilla JavaScript" (developers speak for the raw JavaScript language on its own) or with less UI-focused React predecessors like jQuery. That meant longer development times and plenty of opportunities for errors and bugs. So, in 2011, Facebook engineer Jordan Walke created React JS specifically to improve UI development.

In addition to providing reusable React library code (saving development time and cutting down on the chance for coding errors), React comes with two key features that add to its appeal for JavaScript developers:
JSX
Virtual DOM

JSX
At the heart of any basic website are HTML documents. Web browsers read these documents and display them on your computer, tablet, or phone as web pages. During this process, browsers create something called a Document Object Model (DOM), a representational tree of how the web page is arranged. Developers can then add dynamic content to their projects by modifying the DOM with languages like JavaScript.

JSX (short for JavaScript eXtension) is a React extension that makes it easy for web developers to modify their DOM by using simple, HTML-style code. And—since React JS browser support extends to all modern web browsers—JSX is compatible with any browser platform you might be working with.

This isn't just a matter of convenience, though—using JSX to update a DOM leads to significant site performance improvements and development efficiency.

Virtual DOM
If you're not using React JS (and JSX), your website will use HTML to update its DOM (the process that makes things "change" on screen without a user having to manually refresh a page). This works fine for simple, static websites, but for dynamic websites that involve heavy user interaction it can become a problem (since the entire DOM needs to reload every time the user clicks a feature calling for a page refresh).

However, if a developer uses JSX to manipulate and update its DOM, React JS creates something called a Virtual DOM. The Virtual DOM (like the name implies) is a copy of the site's DOM, and React JS uses this copy to see what parts of the actual DOM need to change when an event happens (like a user clicking a button).

Let's say a user enters a comment in a blog post form and pushes the "Comment" button. Without using React JS, the entire DOM would have to update to reflect this change (using the time and processing power it takes to make this update). React, on the other hand, scans the Virtual DOM to see what changed after a user action (in this case, a comment being added) and selectively updates that section of the DOM only.

This kind of selective updating takes less computing power and less loading time, which might not sound like much when you're talking about a single blog comment, but—when you start to think about all the dynamics and updates associated with even a slightly complex website—you'll realize it adds up to a lot.

Why use React? – React usage benefits
Now that you found out the origination of this ground-breaking library, let's find out the benefits of React and why should you use it for your web application projects:

It's Easier to Learn for Developers : One of the main concerns developers have is choosing a framework (or library) that is easier to learn and implement. React is easy to grasp for developers who are familiar with Javascript. So if you have a team of developers that are very well-versed with Javascript, Reactjs should be your best bet. However, even if developers don't know Javascript, React can be the right place to start. Unlike Angular, React holds a smooth learning curve.

React enables developers to reuse components: In React, your application comprises components. Ideally, you start with building small components like buttons, checkboxes, dropdowns, menus, etc. and create wrapper components around these smaller components. And as you go on writing the higher level wrapper components, you will have a single root component and several hierarchical components. Now, here's a no-brainer: each component in React has its own logic. So if you want to re-use the button component through your app, you are good to go. I am pretty much confident everybody wants reusability in their project.

It provides a unique Abstraction Layer: Another lesser-known business-related benefit with React is that it allows for a good abstraction layer, which means an end-user can't access the complex internals. Your developer only needs to be aware of some basics, and they'd be better off knowing the internal functionalities. Moreover, it does not dictate any architectural patterns like MVC, MVP, and MVVM. Your developer is free to design an app's architecture in any way he sees fit.

It's well established with a vibrant ecosystem of Developer tools: React consists of a rich and vibrant ecosystem. Developers can find dozens of ready-made and customizable charts, graphics, documentation tools, and other components that allow them to build a web app in less time without reinventing the wheel. There's this awesome collection of Reactjs dev tools and tutorials that help developers to build awesome stuff.

Single-page applications catering to multiple industries
Reactjs can be used to build a Single page application catering to any industry. A single-page app is different from the traditional multi-page app that you see everywhere. When a user navigates on a SPA, he will keep interacting with the same page without interacting with an entirely new page. Instead, the web pages (also known as views in this context) typically load inline within the same page itself.

An app like Trello is the best example of single page navigation. Technically, such a type of navigation can be implemented by a technique called routing. The good news is: React offers a library called React-router, which provides routing capabilities in SPAs.

Cross-platform Mobile Apps (React Native)
Using Reactjs in your project comes with a bonus: React Native. Yes, you can build cross-platform apps for Android and iOS using React Native.

For instance, suppose you have built a website for your bakery business. After some-time, you can also build a mobile app using React Native to support it. Of course, you or your developer will not be able to re-use the same code you wrote for the web. Better still, you will be able to use the same architecture and methodology for building the mobile app. Sounds cool, doesn't it?

Where else should you use Reactjs?
The list is endless, but here's a taste of some example web apps where you can use Reactjs:

- Blogs (Gatsby)
- Business websites
- Portfolios
- Forums
- Rating websites
- Membership sites
- eLearning modules
- Galleries
- Personal websites for self-promotion
- Job boards
- Business directories
- Q&A websites like Quora
- Non-profit websites for collecting donations
- Wikis and knowledge bases
- Media-centric sites like YouTube
- Auction and coupon sites

Conclusion
Reactjs is an excellent addition to the projects that need component reusability, impressive user interactions, or crazy animations. That said, it's a robust UI library to build projects that cater to small, medium, and even large-scale organizations. That's why so many companies rely heavily on React for their long-term business goals. Considering React js pros and cons, it can be easily summed up in three words: non-risky, responsive and advanced. The main idea behind this particular library is: "to build large-scale applications with data that changes repeatedly over time" and it tackles the challenge well. It provides developers with the capability of working with a virtual browser (DOM) that is much faster and user-friendly, than the real one. Apart from that, it offers the easier creation of interactive UIs, JSX support, component-based structure and much more. The combination of the above-mentioned factors makes it a reasonable choice for both startups and enterprises.

References:
[1] https://en.wikipedia.org/wiki/React_(JavaScript_library)

[2] https://reactjs.org/

[3] https://www.simform.com/why-use-react/