**Introduction to React JS**

* Introduction to React JS

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application. It was created by Jordan Walke, who was a software engineer at Facebook. It was initially developed and maintained by Facebook and was later used in its products like WhatsApp & Instagram. Facebook developed ReactJS in 2011 in its newsfeed section, but it was released to the public in the month of May 2013.

Today, most of the websites are built using MVC (model view controller) architecture. In MVC architecture, React is the 'V' which stands for view, whereas the architecture is provided by the Redux or Flux.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organize these components inside higher level components which define the application structure. For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself. The form components would specify the structure of the form along with elements inside of it.

* History of web development

Web development is the work involved in developing a [website](https://en.wikipedia.org/wiki/Web_site) for the [Internet](https://en.wikipedia.org/wiki/Internet) ([World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web)) or an [intranet](https://en.wikipedia.org/wiki/Intranet) (a private network).[[1]](https://en.wikipedia.org/wiki/Web_development#cite_note-1) Web development can range from developing a simple single [static page](https://en.wikipedia.org/wiki/Static_Web_page) of [plain text](https://en.wikipedia.org/wiki/Plain_text) to complex [web applications](https://en.wikipedia.org/wiki/Web_application), [electronic businesses](https://en.wikipedia.org/wiki/Electronic_business), and [social network services](https://en.wikipedia.org/wiki/Social_network_service). A more comprehensive list of tasks to which Web development commonly refers, may include [Web engineering](https://en.wikipedia.org/wiki/Web_engineering), [Web design](https://en.wikipedia.org/wiki/Web_design), [Web content development](https://en.wikipedia.org/wiki/Web_content_development), client liaison, [client-side](https://en.wikipedia.org/wiki/Client-side_scripting)/[server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting), [Web server](https://en.wikipedia.org/wiki/Web_server) and [network security](https://en.wikipedia.org/wiki/Network_security) configuration, and [e-commerce](https://en.wikipedia.org/wiki/E-commerce) development.

Among Web professionals, "Web development" usually refers to the main non-design aspects of building Web sites: writing [markup](https://en.wikipedia.org/wiki/Markup_language" \o "Markup language) and [coding](https://en.wikipedia.org/wiki/Computer_programming).[[2]](https://en.wikipedia.org/wiki/Web_development#cite_note-2) Web development may use [content management systems](https://en.wikipedia.org/wiki/Content_management_system) (CMS) to make content changes easier and available with basic technical skills.

* History of React

Current version of React.JS is V18.0.0 (April 2022).

Initial Release to the Public (V0.3.0) was in July 2013.

React.JS was first used in 2011 for Facebook's Newsfeed feature.

Facebook Software Engineer, Jordan Walke, created it.

Current version of create-react-app is v5.0.1 (April 2022).

create-react-app includes built tools such as webpack, Babel, and ESLint.

* DOM (Document Object Model)

When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page
* DOM Manipulation

The mechanism of finding a particular node in the document tree is called Querying the DOM. Adding a new node, deleting a node, or updating a node in the document tree is called DOM Manipulation. The result of a DOM manipulation reflects on the web user interface. This process is called rendering.

DOM manipulation is the interaction of the JavaScript DOM API to modify or change the HTML document. With DOM manipulation, you can create, modify, style, or delete elements without a refresh. It also promotes user interactivity with browsers.

You can use different programming languages to manipulate the DOM. This particular article will focus on how to use the JavaScript language to manipulate DOM.

DOM stands for Document Object Model. It is an in-memory representation of HTML and is tree structured. DOM Manipulation is the process by which one can dynamically change the content of the web page. Normally, JavaScript is used while working with the DOM. Methods like getElementById or removeChild are part of the API provided by the HTML DOM.

**The Issue**

DOM Manipulation is the core of modern interactive web pages. So, it is important that working with DOM should be fairly easy and quick. Since, it is always tree-structured, traversing the DOM is easy. But when it comes to the quick part, it is not as quick as it should be.

For example, let us say we want to update the first item of a list containing ten items. Most JavaScript frameworks would update the entire list, just to update only the first item. Most modern web pages have huge DOM structures and this behavior would cost too much, resulting in slower loading pages.

* Virtual DOM

VDOM is the virtual representation of Real DOM.

React update the state changes in Virtual DOM first and then it syncs with Real DOM.

Virtual DOM is just like a blueprint of a machine, can do the changes in the blueprint but those changes will not directly apply to the machine.

Virtual DOM is a programming concept where a virtual representation of a UI is kept in memory synced with “Real DOM ” by a library such as ReactDOM and this process is called reconciliation.

Virtual DOM makes the performance faster, not because the processing itself is done in less time. The reason is the amount of changed information – rather than wasting time on updating the entire page, you can dissect it into small elements and interactions.

Take a look at the following render code representing the React version of the previous JavaScript example:

// ...

const update = () => {

const element = (

<>

<h3>React:</h3>

<form>

<input type="text" />

</form>

<span>Time: {new Date().toLocaleTimeString()}</span>

</>

);

root.render(element);

};

* How does React use Virtual DOM

As we know, [React is a component-based library](https://blog.logrocket.com/build-component-library-react-typescript/). A React component will naturally re-render if there are state or prop changes or if its parent component re-renders.

React cannot afford the cost of repainting all of the DOM nodes after every re-render. To overcome this challenge, React implemented the concept of virtual DOM.

Instead of allowing the browser to redraw all the page elements after every re-render or DOM update, React uses the concept of virtual DOM to figure out what exactly has changed without involving the actual DOM and then ensures that the actual DOM only repaints the necessary data.

* Virtual DOM vs Real DOM

 Now let’s understand the differences between Real Dom and virtual Dom:

|  |  |
| --- | --- |
| Real DOM | Virtual DOM |
| DOM manipulation is very expensive | DOM manipulation is very easy |
| There is too much memory wastage | No memory wastage |
| It updates Slow | It updates fast |
| It can directly update HTML | It can’t update HTML directly |
| Creates a new DOM if the element updates. | Update the JSX if the element update |
| It allows us to directly target any specific node (HTML element) | It can produce about 200,000 Virtual DOM Nodes / Second. |
| It represents the Ul of your application | It is only a virtual representation of the DOM |

* Thinking in React

React community has provided a direction on how to think in React way and build big , fast and scalable applications. React has reached multiple platforms and widely used a popular JavaScript UI interface library.

**Step 1** − Creating a simple mock service

If we need to make a server call and fetch data. We can create a mock service to start with and build a component to fetch and display data.

Here we can include the processing of json in component and evaluating the expected result.

**Step 2** − Break the functionality into smaller components

The first Thing React suggest is to create the smaller understandable design of boxes . These boxes will represent the association between the different components and passing of data flow.

Making of components should be based on principle of single responsibility. That means a single component should handle a single task.

Creating a component hierarchy will make developers task easier.

**Step 3** − If possible start by making a simple static version

With the use of mock service and smaller components we can create a static version and build the app thereon.

Creating a static version should not use state modifications. It should play with passage of props and rendering Ui only. Keeping one way data flow in React makes it simple and modularTo make it more clear the difference and use case of props and state should be understood very well.

**Step 4** − Identifying the Minimal representation of UI state

For making app interactive the state should be able to triggered from the relevant component.

Identifying the required mutable state is necessary to build the correct app.

**Step 5** − Identify where the state should live −

State can be shared between the multiple child components. Lifting the state up can be used to make communication between the multiple components. Using state management libraries like Redux solves the lot of issues arising from state.

React strongly recommends the one way down flow of data to components.

**Step 6** − Add two way data flow if required −

The controlled component in form handling are the example of two way data binding.

* React one - way data binding

Data Binding is the process of connecting the view element or user interface, with the data which populates it.

In ReactJS, components are rendered to the user interface and the component’s logic contains the data to be displayed in the view(UI). The connection between the data to be displayed in the view and the component’s logic is called data binding in ReactJS.

ReactJS uses one-way data binding. In one-way data binding one of the following conditions can be followed:

* **Component to View:**Any change in component data would get reflected in the view.
* **View to Component:**Any change in View would get reflected in the component’s data.

In order to demonstrate the code examples, we have to create a basic React application using the following steps.

**Creating React Application:**

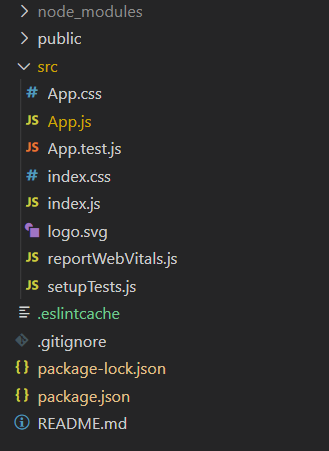
* **Step 1:** Create a React application using the following command:

npx create-react-app foldername

* **Step 2:**After creating your project folder i.e. foldername**,**move to it using the following command:

cd foldername

**Project Structure:**It will look like the following.



**Implementing Component to View Data Binding:**

Write down the following code in the **App.js** file. Here, App is our default component where we have written code.

import React, { Component } from 'react';

class App extends Component {

constructor() {

super();

this.state = {

subject: "ReactJS"

};

}

render() {

return (

<div style={{ textAlign: "center" }}>

<h4 style={{ color: "#68cf48" }}>GeeksForGeeks</h4>

<p><b>{this.state.subject}</b> Tutorial</p>

</div>

)

}

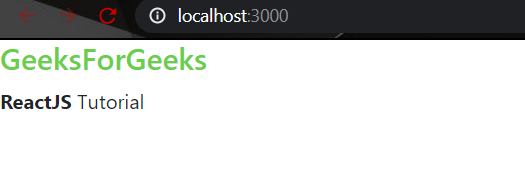
}

export default App;

**Step to Run Application:**Run the application using the following command from the root directory of the project:

npm start

**Output:**Now open your browser and go to***http://localhost:3000/***, you will see the following output:



**Implementing View to Component Data Binding:**

We cannot directly apply View to Component data binding in ReactJS, for this we have to add **event handlers** to the view element.

Write down the following code in the **App.js** file. Here, App is our default component where we have written code.

import React, { Component } from 'react';

class App extends Component {

constructor() {

super();

this.state = {

subject: ""

};

}

handleChange = event => {

this.setState({

subject: event.target.value

})

}

render() {

return (

<div>

<h4 style={{ color: "#68cf48" }}>GeeksForGeeks</h4>

<input placeholder="Enter Subject"

onChange={this.handleChange}></input>

<p><b>{this.state.subject}</b> Tutorial</p>

</div>

)

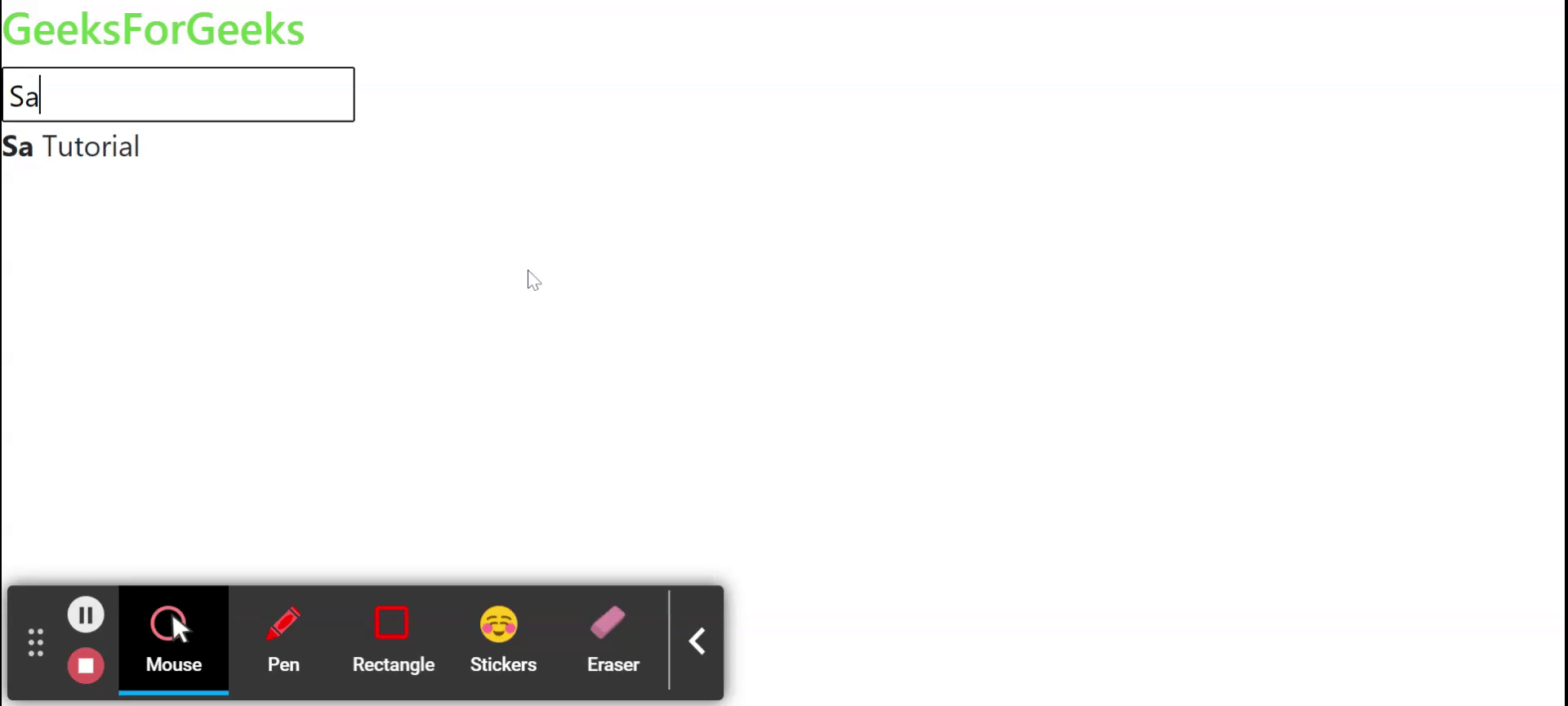
}

}

export default App;

**Step to Run Application:**Run the application using the following command from the root directory of the project: npm start

**Output:**Now open your browser and go to***http://localhost:3000/***, you will see the following output:



* React two - way data binding

Now that we understand both component to view and view to component connections, we can try and implement them together. How? Let’s understand that with an example:

Consider the following component which implements two-way data-binding:

function Component1() {

let [inputValue,setInputValue] = useState('');

  let changeValue = (e) => setInputValue(e.target.value);

return (

   <div>

     <input value={inputValue} onChange={changeValue} />

   </div>

);

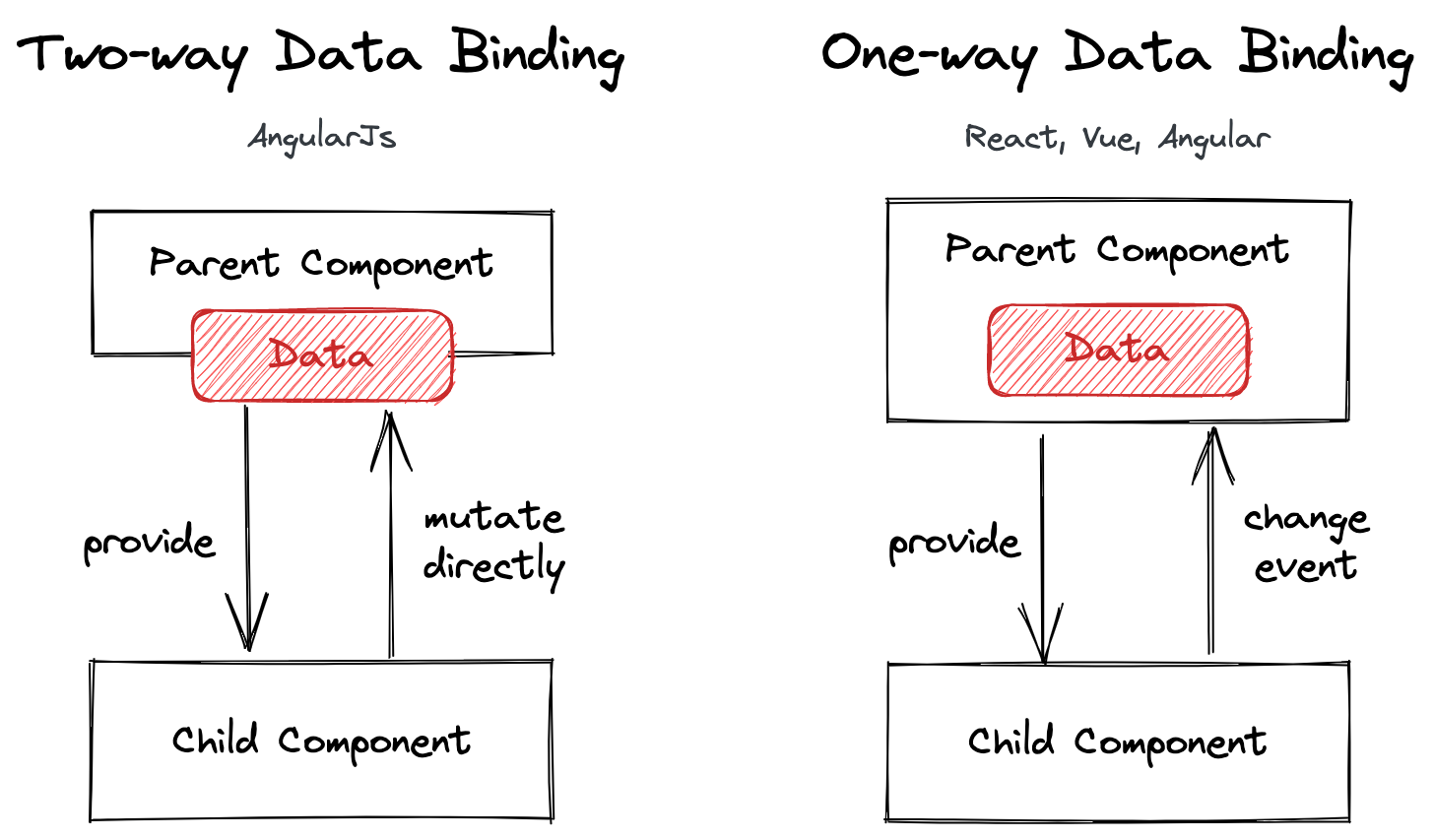
}

We have used our knowledge of both the connections and used them here in the code above. By using component to view connection, we have bound the value of input element to the component variable inputValue and by using view to component connection, we have added the onChange event to the input element. This way, we can perform two-way data binding in React.

* One way vs two-way data binding or Difference b/n one - way data binding & two - way data binding

With two-way data binding the data flows in both directions, from the parent component to the child component and vice versa. Both, the parent and the child, are allowed to mutate data.

With one-way Data Binding the data flows in one direction, from the parent component to the child component. The child component can read the data but is not allowed to update the data directly. Instead, the child component emits an event to the parent and the parent component is responsible for updating the data.



Older frameworks (like AngularJS, which is deprecated) support Two-way Data Binding but modern frameworks like Angular and Vue don't support this model anymore. They both work with One-way Data Binding.

* Server-side Rendering

Server-side rendering with JavaScript libraries like React is where the server returns a ready to render HTML page and the JS scripts required to make the page interactive. The HTML is rendered immediately with all the static elements. In the meantime, the browser downloads and executes the JS code after which the page becomes interactive. The interactions on this page are now handled by the browser on the client-side. For any new content or new data, the browser sends a request to the server through APIs and only the newly required information is fetched.

* Advantages of using server-side rendering
* Fast initial loading of the web page since ready to display HTML is provided to the browser.
* Great user experience even if the user has a bad connection, outdated device or JavaScript disabled in the browser because all the basic content is ready to be rendered.
* The content of the web page is indexed quicker resulting in better SEO ranking.
* A great option for static pages since server-side rendering loads the content promptly and efficiently.
* Advantages of React.JS

**1. Easy to Learn and Use**

ReactJS is much easier to learn and use. It comes with a good supply of documentation, tutorials, and training resources. Any developer who comes from a JavaScript background can easily understand and start creating web apps using React in a few days. It is the V(view part) in the MVC (Model-View-Controller) model, and referred to as ?one of the JavaScript frameworks.? It is not fully featured but has the advantage of open-source JavaScript User Interface(UI) library, which helps to execute the task in a better manner.

**2. Creating Dynamic Web Applications Becomes Easier**

To create a dynamic web application specifically with HTML strings was tricky because it requires a complex coding, but React JS solved that issue and makes it easier. It provides less coding and gives more functionality. It makes use of the JSX(JavaScript Extension), which is a particular syntax letting HTML quotes and HTML tag syntax to render particular subcomponents. It also supports the building of machine-readable codes.

**3. Reusable Components**

A ReactJS web application is made up of multiple components, and each component has its own logic and controls. These components are responsible for outputting a small, reusable piece of HTML code which can be reused wherever you need them. The reusable code helps to make your apps easier to develop and maintain. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

**4. Performance Enhancement**

ReactJS improves performance due to virtual DOM. The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. Most of the developers faced the problem when the DOM was updated, which slowed down the performance of the application. ReactJS solved this problem by introducing virtual DOM. The React Virtual DOM exists entirely in memory and is a representation of the web browser's DOM. Due to this, when we write a React component, we did not write directly to the DOM. Instead, we are writing virtual components that react will turn into the DOM, leading to smoother and faster performance.

**5. The Support of Handy Tools**

React JS has also gained popularity due to the presence of a handy set of tools. These tools make the task of the developers understandable and easier. The React Developer Tools have been designed as Chrome and Firefox dev extension and allow you to inspect the React component hierarchies in the virtual DOM. It also allows you to select particular components and examine and edit their current props and state.

**6. Known to be SEO Friendly**

Traditional JavaScript frameworks have an issue in dealing with SEO. The search engines generally having trouble in reading JavaScript-heavy applications. Many web developers have often complained about this problem. ReactJS overcomes this problem that helps developers to be easily navigated on various search engines. It is because React.js applications can run on the server, and the virtual DOM will be rendering and returning to the browser as a regular web page.

**7. The Benefit of Having JavaScript Library**

Today, ReactJS is choosing by most of the web developers. It is because it is offering a very rich JavaScript library. The JavaScript library provides more flexibility to the web developers to choose the way they want.

**8. Scope for Testing the Codes**

ReactJS applications are extremely easy to test. It offers a scope where the developer can test and debug their codes with the help of native tools.

* When to use React JS? And when to opt for alternatives?
* **When you need a large ecosystem**

If you are building an app that uses a lot of different packages and libraries, React can be a good choice. It currently has one of the largest ecosystems for web development frameworks, with helpful libraries like Material UI and React Spring. In fact, there are more than 75,000 packages on NPM with the React Keyword, which is almost three times the next largest framework, Vue. React also has a large and mature community, with lots of tutorials and guides on various aspects of React.

* **When you need a mature and widely used base**

React is mature, having been used for years in production by many large companies, like Facebook, Netflix, Uber, and more. It is almost guaranteed to be stable, as Facebook uses the latest releases in production on their website and app. If you have an app that is required to be extremely reliable and stable, then React can be a good choice.

* Applications of React.js
* Facebook
* Instagram
* Netflix
* New York Times
* Yahoo! Mail
* Khan Academy
* WhatsApp
* Vivaldi Browser
* Codecademy
* Dropbox