React-Js Simplified

React-Js in Depth

Sunil Yadav

Table of Content

Page Left Intentionally

Introduction

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application.

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(Fast Processing) based mechanism to fill data in HTML DOM.

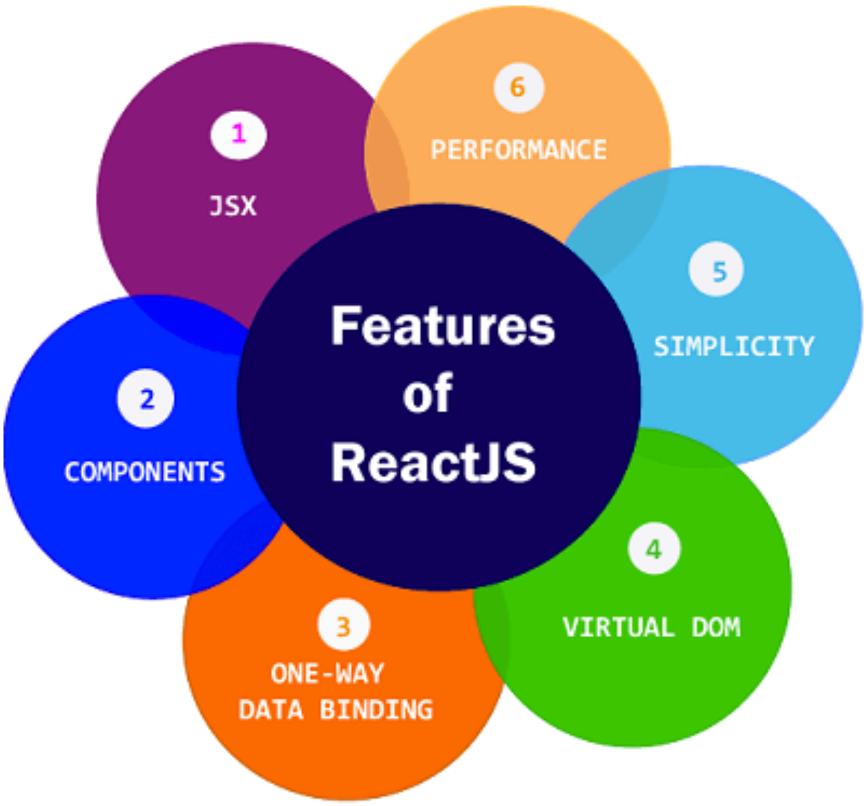
The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organise these components inside higher level components which define the application structure.

Rather than manipulating the document in a browser after changes to our data like in traditional web apps, React resolves changes on a DOM built and run entirely in memory. After the virtual DOM has been updated, React determines what changes made to the actual browser's DOM.

Features of ReactJs

It is playing an essential role in the front-end ecosystem. The important features of ReactJS are as following.



JSX

JSX stands for JavaScript XML. It is a JavaScript syntax extension. It’s an XML or HTML like syntax used by ReactJS. This syntax is processed into JavaScript calls of React Framework. It extends the ES6 so that HTML like text can co-exist with JavaScript react code.

Components

Chunks of self-functional code, that aggregates together to make entire project. Components can be reusable which help you to maintain the code when working on larger scale projects

One-way Data Binding

ReactJS is designed in such a manner that follows unidirectional data flow or one-way data binding. The benefits of one-way data binding give you better control throughout the application.

Flux is a pattern that helps to keep your data unidirectional. This makes the application more flexible that leads to increase efficiency.

Virtual DOM

A virtual DOM object is a representation of the original DOM object. It works like a one-way data binding. Whenever any modifications happen in the web application, the entire UI is re-rendered in virtual DOM representation. Then it checks the difference between the previous DOM representation and new DOM. Once it has done, the real DOM will update only the things that have actually changed. This makes the application faster, and there is no wastage of memory.

Simplicity

ReactJS uses JSX file which makes the application simple and to code as well as understand. We know that ReactJS is a component-based approach which makes the code reusable as your need.

Performance

The DOM exists entirely in memory. Due to this, when we create a component, we did not write directly to the DOM. Instead, we are writing virtual components that will turn into the DOM leading to smoother and faster performance.

Pros and Cons of React-Js

Pros -

- *Easy to Learn and Use*
- *Creating Dynamic Web Applications Becomes Easier*
- *Reusable Component*
- *Performance Enhancement*
- *Scope of Testing Code.*

Cons -

- *The high pace of development*
- *Poor Documentation*
- *Only for UI*
- *JSX as a barrier.*

AngularJS and ReactJS

Angular Js -

AngularJS is an open-source JavaScript framework used to build a dynamic web application. It was developed and maintained by google. It is based on HTML and JavaScript and mostly used for building a Single Page Application.

It can be included to an HTML page with a <script> tag. It extends HTML by adding built-in attributes with the directive and binds data to HTML with Expressions.

Feature of Angular Js -

- *Data-binding (2-Way)*
- *POJO Model*
- *MVC Framework*
- *Services*
- *UI Interface with HTML*
- *Dependency Injection*
- *Routing - Changing Content onto the Same page with URL routing.*

Angular vs React

Attribute	Angular - Js	React - Js
Author	Google	Facebook Community
Developer	Misko Hevery	Jordan Walke
Initial Release	October 2010	March 2013
Language	JavaScript, HTML	JSX
Type	Open Source MVC Framework	Open Source JS Framework
Rendering	Client-Side	Server-Side
Packaging	Weak	Strong
Data-Binding	Bi-directional	Uni-directional
DOM	Regular DOM	Virtual DOM
App Architecture	MVC	Flux
Dependencies	It manages dependencies automatically.	It requires additional tools to manage dependencies.
Performance	Slow	Fast, due to virtual DOM.
Best For	It is best for single page applications that update a single view at a time.	It is best for single page applications that update multiple views at a time.

React Native

React Native is an open-source JavaScript framework used for developing a mobile application for iOS Android, and Windows. It uses only JavaScript to build a cross-platform mobile app.

React Native is same as React, but it uses native components instead of using web components as building blocks. It targets mobile platforms rather than the browser.

React Native was initially developed for the iOS application. However, recently, it also supports the Android operating system.

Advantages of React Native

- Cross-Platform Usage
- Improving with Time
- Javascript
- Community
- Native Components

Advantages of React Native

- Tough to Learn.
- React Native is New and Immature.
- It take time to initialise.
- It is not recommended for Confidential type of application development.

React vs React Native

React - Js	React - Native
The ReactJS initial release was in 2013.	The React Native initial release was in 2015.
It is used for developing web applications.	It is used for developing mobile applications.
It can be executed on all platforms.	It is not platform independent. It takes more effort to be executed on all platforms.
It uses a JavaScript library and CSS for animations.	It comes with built-in animation libraries.
It uses React-router for navigating web pages.	It has built-in Navigator library for navigating mobile applications.
It uses HTML tags.	It does not use HTML tags.
It can use code components, which saves a lot of valuable time.	It can reuse React Native UI components & modules which allow hybrid apps to render natively.
It provides high security.	It provides low security in comparison to ReactJS.
In this, the Virtual DOM renders the browser code.	In this, Native uses its API to render code for mobile applications.

React vs Vue

React and Vue have a lot of common things like the component-based architecture, usage of virtual DOM, usage of props, chrome Dev tools for debugging, and many more.

Attribute	React	Vue
Preferred Language	JavaScript/ JavaScript XML	HTML/JavaScript
Size	The size of the React library is 100 kilobytes (approx.).	The size of the Vue library is 60 kilobytes (approx.).
Performance	Its performance is slow as compared to Vue.	Its performance is fast as compared to React.
Flexibility	React provides great flexibility to support third-party libraries.	Vue provides limited flexibility as compared to React.
Coding Style	JSX Coding Style	It separates HTML, JS, and CSS.
Data Binding	React supports one-way data binding.	Vue supports both one-way and two-way data binding.
Current Version	React 16.8.6 on March 27, 2019	Vue 2.6.10 on March 20, 2019.
Long Term Support	It is suitable for long term supports.	It is not suitable for long term support.

React JSX

JSX is an HTML-like syntax used by React that extends ECMAScript so that **HTML-like** syntax can co-exist with JavaScript/React code. The syntax is used by **preprocessors** (i.e., transpilers like babel) to transform HTML-like syntax into standard JavaScript objects that a JavaScript engine will parse.

```
JSX :: <div>Hello World</div>
```

Has corresponding Output for React as -

```
React.createElement("div", null, "Hello World");
```

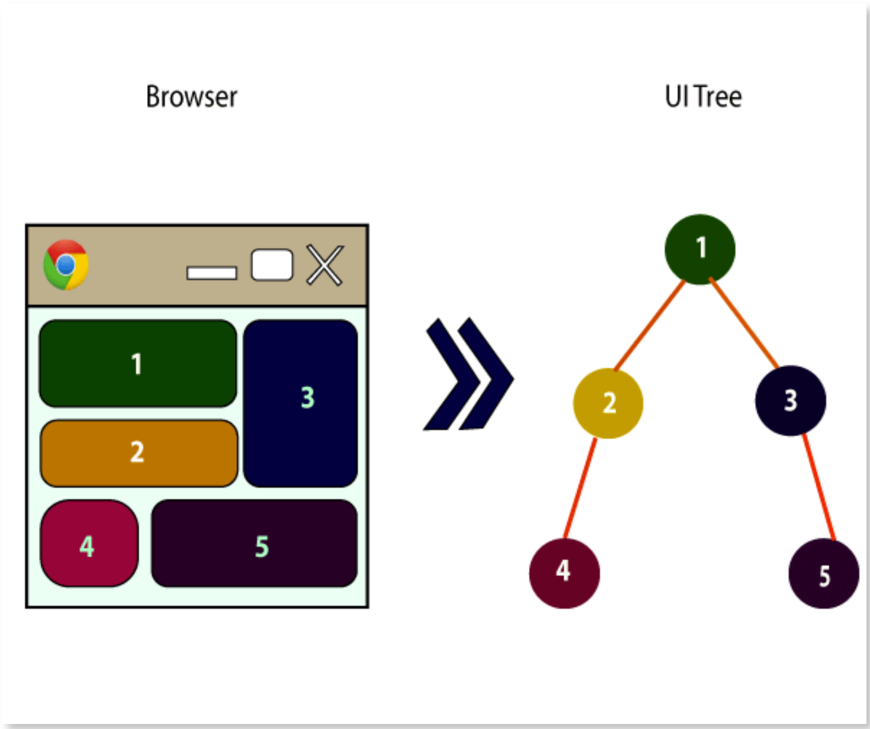
To use more than one element, you need to wrap it with one container element.

JSX use attributes with the HTML elements same as regular HTML. JSX uses **camelcase** naming convention for attributes rather than standard naming convention of HTML such as a class in HTML becomes **className** in JSX because the class is the reserved keyword in JavaScript.

React Component

A Component is considered as the core building blocks of a React application. It makes the task of building UIs much easier. Each component exists in the same space, but they work independently from one another and merge all in a parent component, which will be the final UI of your application.

Every React component have their own structure, methods as well as APIs. They can be reusable as per your need.



Components and their Connection

In ReactJS, we have mainly two types of components. They are

1. Functional Components
2. Class Components

Function Components

FC are a way to write components that only contain a render method and don't have their own state.

The functional component is also known as a stateless component because they do not hold or manage state.

Class components

CC are more complex than functional components. It requires you to extend from React. Component and create a render function which returns a React element.

The class component is also known as a stateful component because they can hold or manage local state.

React State

They are also responsible for making a component dynamic and interactive.

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time. The change in state over time can happen as a response to user action or system event. A component with the state is known as stateful components (Class Component).

It can be set by using the **setState()** method and calling setState() method triggers UI updates.

To set an initial state before any interaction occurs, we need to use the **getInitialState()** method.

To do this, add a class constructor which assigns an initial state using this.state.

The '**this.state**' property can be rendered inside **render()** method.

To set the state, it is required to call the super() method in the constructor. It is because this.state is uninitialised before the super() method has been called.

```
this.toggleDisplayBio = this.toggleDisplayBio.bind(this);
```

React Props

It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are **immutable** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as **this.props** and can be used to render dynamic data in our render method.

It is not necessary to always add props in the reactDom.render() element.

It is possible to combine both state and props in your app. You can set the state in the parent component and pass it in the child component using props

React Props Validation

Props validation is a tool that will help the developers to avoid future bugs and problems. It is a useful way to force the correct usage of your components. It makes your code more readable. React components used special property **PropTypes** that help you to catch bugs by validating data types of values passed through props, although it is not necessary to define components with propTypes.

```
import PropTypes from 'prop-types'

Component.propTypes = { /*Definition */};
```

ReactJS Custom Validators

ReactJS allows creating a custom validation function to perform custom validation. The following argument is used to create a custom validation function.

- **props:** It should be the first argument in the component.
- **propName:** It is the propName that is going to validate.
- **componentName:** It is the componentName that are going to validated again.

Difference between State and Props

Props are read-only.	State changes can be asynchronous.
Props are immutable.	State is mutable.
Props allow you to pass data from one component to other components as an argument.	State holds information about the components.
Props can be accessed by the child component.	State cannot be accessed by child components.
Props are used to communicate between components.	States can be used for rendering dynamic changes with the component.
Stateless component can have Props.	Stateless components cannot have State.
Props make components reusable.	State cannot make components reusable.
Props are external and controlled by whatever renders the component.	The State is internal and controlled by the React Component itself.

React Constructor

The constructor in a React component is called before the component is mounted. When you implement the constructor for a React component, you need to call **super(props)** method before any other statement.

In React, constructors are mainly used for two purposes:

1. It used for initialising the local state of the component by assigning an object to this.state.
2. It used for binding event handler methods that occur in your component.

React Component API

ReactJS component is a top-level API. It makes the code completely individual and reusable in the application. It includes various methods for:

- Creating elements
- Transforming elements
- Fragments

Here, we are going to explain the three most important methods available in the React component API.

1. setState()
2. forceUpdate()
3. findDOMNode()

React Component Lifecycle

In ReactJS, every component creation process involves various lifecycle methods. These lifecycle methods are termed as component's lifecycle.

The lifecycle of the component is divided into **four phases**. They are:

1. Initial Phase
2. Mounting Phase
3. Updating Phase
4. Unmounting Phase

Initial Phase –

getDefaultProps()

getInitialState()

Mounting Phase –

componentWillMount()
componentDidMount()
render()

Updating Phase –

componentWillRecieveProps()

shouldComponentUpdate()

componentWillUpdate()

render()

componentDidUpdate()

Unmount Phase —

componentWillUnmount()

Controlled Vs UnControlled Component

Controlled Component :

A controlled component is bound to a value, and its changes will be handled in code by using **event-based callbacks**. Here, the input form element is handled by the react itself rather than the DOM. In this, the mutable state is kept in the state property and will be updated only with setState() method.

UnControlled Component :

It maintains their own state and will be updated when the input value changes.

Controlled	Uncontrolled
It does not maintain its internal state.	It maintains its internal states.
Here, data is controlled by the parent component.	Here, data is controlled by the DOM itself.
It accepts its current value as a prop.	It uses a ref for their current values.
It allows validation control.	It does not allow validation control.
It has better control over the form elements and data.	It has limited control over the form elements and data.

React Conditional Rending

There is more than one way to do conditional rendering in React. They are given below.

- if
- ternary operator
- logical && operator
- switch case operator
- Conditional Rendering with enums

React Router

Routing is a process in which a user is directed to different pages based on their action or request. ReactJS Router is mainly used for developing Single Page Web Applications. React Router is used to define multiple routes in the application. When a user types a specific URL into the browser, and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.

What is Route?

It is used to define and render component based on the specified path. It will accept components and render to define what should be rendered.

What is < Link> component?

This component is used to create links which allow to **navigate** on different **URLs** and render its content without reloading the webpage.

React Router <Redirect>

A <Redirect> component is used to redirect to another route in our application to maintain the old URLs. It can be placed anywhere in the route hierarchy.

Benefits Of React Router

The benefits of React Router is given below:

- In this, it is not necessary to set the browser history manually.
- Link uses to navigate the internal links in the application. It is similar to the anchor tag.
- It uses Switch feature for rendering.
- The Router needs only a Single Child element.
- In this, every component is specified in .

React CSS

CSS in React is used to style the React App or Component. The **style** attribute is the most used attribute for styling in React applications, which adds dynamically-computed styles at render time. It accepts a JavaScript object in **camelCased** properties rather than a CSS string.

four ways to style React Components, which are given below:

1. Inline Styling
2. CSS Stylesheet
3. CSS Module
4. Styled Components

React Animation

we can add animation using an explicit group of components known as the **React Transition Group**.

React Transition group has mainly **two APIs** to create transitions. These are:

1. **ReactTransitionGroup**
2. **ReactCSSTransitionGroup**

React High Order Component

HOC is an advanced technique for reusing component logic. It is a function that takes a component and returns a new component

A higher order component function accepts another function as an argument.

React Context

Context allows passing data through the component tree without passing props down manually at every level.

How to use Context

There are two main steps to use the React context into the React application:

1. Setup a context provider and define the data which you want to store.
2. Use a context consumer whenever you need the data from the store

React Context API

The React Context API is a component structure, which allows us to share data across all levels of the application. The main aim of Context API is to solve the problem of prop drilling (also called "Threading"). The Context API in React are given below.

1. React.createContext
2. Context.provider
3. Context.Consumer
4. Class.contextType

```
static contextType = createdContext;
```

React Hooks

Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

When to use a Hooks

If you write a function component, and then you want to add some state to it, previously you do this by converting it to a class. But, now you can do it by using a Hook inside the existing function component.

Hooks State

Hook state is the new way of declaring a state in React app. Hook uses useState() functional component for setting and retrieving state.

Hook Effect

Effects Hooks are equivalent to componentDidMount(), componentDidUpdate(), and componentWillUnmount() lifecycle methods, in a functional component.

Side effects have common features which the most web applications need to perform, such as:

- Updating the DOM,
- Fetching and consuming data from a server API,
- Setting up a subscription, etc.

In React component, there are two types of side effects:

1. Effects Without Cleanup
2. Effects With Cleanup

Built-in Hooks

Here, we describe the APIs for the built-in Hooks in React. The built-in Hooks can be divided into two parts, which are given below.

Basic Hooks

- useState
- useEffect
- useContext

Additional Hooks

- useReducer
- useCallback
- useMemo
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue