React JS

According to Wikipedia

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. 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

Simple Defination

React is a library that let Web Developers easily manage DOM Actions and Events. We could for example, dynamically change entire page without browser reload with React easily.

Notable features

Declarative

React adheres to the declarative programming paradigm. Developers design views for each state of an application, and React updates and renders components when data changes. This is in contrast with imperative programming.

Components

React code is made of entities called components. These components are reusable and must be formed in the SRC folder following the Pascal Case as its naming convention (capitalize camelCase). Components can be rendered to a particular element in

the DOM using the React DOM library. When rendering a component, one can pass the values between components through "props"

```
import React from "react";
import Tool from "./Tool";
const Example = () => {
  return (
    <>
      <div className="app">
        <Tool name="Gulshan" />
      </div>
    </>
  );
};

export default Example;
```

In the above example, the name property with the value "Gulshan" has been passed from the Example component to the Tool component.

Also the return section is wrapped in a tag because there is a limitation in the return function, it can only return a single value. So all JSX elements and components are bound into a single tag.

The two primary ways of declaring components in React are through function components and class-based components.

Functional components

Function components are declared with a function that then returns some JSX

```
const Greeter = () => <div>Hello World</div>;
```

Class-based components

Class-based components are declared using ES6 classes.

```
class ParentComponent extends React.Component {
  state = { color: 'green' };
  render() {
```

```
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Where class components are all about the use of classes and the lifecycle methods, functional components have hooks to deal with state management and other problems which arise when writing code in React.

Virtual DOM

Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This process is called reconciliation. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page.

Lifecycle methods

Lifecycle methods for class-based components use a form of hooking that allows the execution of code at set points during a component's lifetime.

- `shouldComponentUpdate` allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
- `componentDidMount` is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
- `componentWillUnmount` is called immediately before the component is torn down or "unmounted". This is commonly

used to clear resource-demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any `setInterval()` instances that are related to the component, or an "eventListener" set on the "document" because of the presence of the component)
- `render` is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

JSX

JSX, or JavaScript Syntax Extension, is an extension to the JavaScript language syntax.[13] Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

```
class App extends React.Component {
  render() {
    return (
      <div>
        <p>Header</p>
        <p>Content</p>
        <p>Footer</p>
      </div>
    );
  }
}
```

Architecture beyond HTML

The basic architecture of React applies beyond rendering HTML in the browser. For example, Facebook has dynamic charts that render to `<canvas>` tags, and Netflix and PayPal use universal loading to render identical HTML on both the server and client.

React hooks

Hooks are functions that let developers "hook into" React state and lifecycle features from function components. Hooks do not work inside classes — they let you use React without classes.

React provides a few built-in hooks like useState, useContext, useReducer , useMemo and useEffect. Others are documented in the Hooks API Reference. useState and useEffect, which are the most commonly used, are for controlling state and side effects respectively.

Rules of hooks

There are rules of hooks which describe the characteristic code pattern that hooks rely on. It is the modern way to handle state with React.

1. Hooks should only be called at the top level (not inside loops or if statements).
2. Hooks should only be called from React function components and custom hooks, not normal functions or class components. Although these rules can't be enforced at runtime, code analysis tools such as linters can be configured to detect many mistakes during development. The rules apply to both usage of hooks and the implementation of custom hooks, which may call other hooks.

History

React was created by Jordan Walke, a software engineer at Facebook, who released an early prototype of React called "FaxJS". He was influenced by XHP, an HTML component library for PHP. It was first deployed on Facebook's News Feed in 2011 and later on Instagram in 2012. It was open-sourced at JSConf US in May 2013.

React Native, which enables native Android, iOS, and UWP development with React, was announced at Facebook's React Conf in February 2015 and open-sourced in March 2015.

On April 18, 2017, Facebook announced React Fiber, a new set of internal algorithms for rendering, as opposed to React's old rendering algorithm, Stack. React Fiber was to become the foundation of any future improvements and feature development of the React library.The actual syntax for programming with React does not change; only the way that the syntax is executed has changed. React's old rendering system, Stack, was developed at a time when the focus of the system on dynamic change was not understood. Stack was slow to draw complex animation, for example, trying to accomplish all of it in one chunk. Fiber breaks down animation into segments that can be spread out over multiple frames. Likewise, the structure of a page can be broken into segments that may be maintained and updated separately. JavaScript functions and virtual DOM objects are called "fibers", and each can be operated and updated separately, allowing for smoother on-screen rendering.

On September 26, 2017, React 16.0 was released to the public.

On February 16, 2019, React 16.8 was released to the public. The release introduced React Hooks.

On August 10, 2020, the React team announced the first release candidate for React v17.0, notable as the first major release without major changes to the React developer-facing API.