ReactJs

Course plan

- ReactJS Overview & Architecture
- ReactJS Installation
- ReactJS and React Native
- React JSX
- ReactJS Components & Life Cycle
- ReactJS Routing
- ReactJS Styling
- ReactJs Properties(Props)
- ReactJS State Management
- ReactJS Event Management
- ReactJS - Form Programming
- ReactJS - Http Client Programming
- React Hooks
- ReactJs Building and Deployment

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 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 news feed section, but it was released to the public in the month of **May 2013.**

Introduction

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.

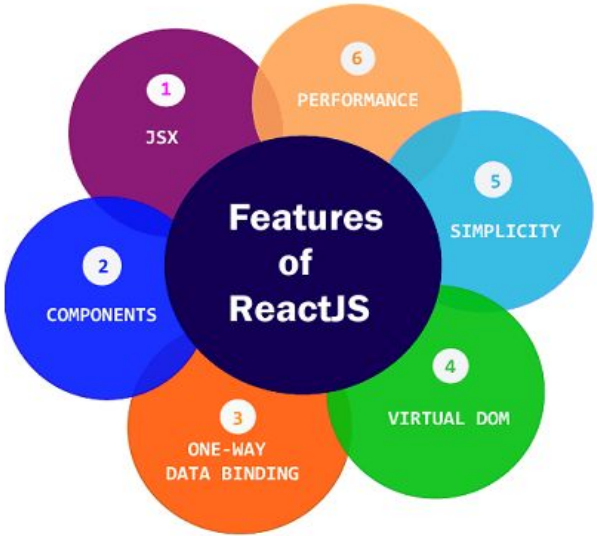
These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM.

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

Features

The important features of ReactJS are as following.

- JSX
- Components
- One-way Data Binding
- Virtual DOM
- Simplicity
- Performance



JSX

JSX stands for JavaScript XML.

It is a JavaScript syntax extension.

Its 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.

It is not necessary to use JSX, but it is recommended to use in ReactJS.

Components

ReactJS is all about components.

ReactJS application is made up of multiple components, and each component has its own logic and controls.

These 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.

If the data flow is in another direction, then it requires additional features.

It is because components are supposed to be immutable and the data within them cannot be changed.

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.

This makes it simple to use and learn.

Performance

ReactJS is known to be a great performer.

This feature makes it much better than other frameworks out there today.

The reason behind this is that it manages a virtual DOM.

The DOM is a cross-platform and programming API which deals with HTML, XML or XHTML. The DOM exists entirely in memory.

Due to this, when we create a component, we did not write directly to the DOM.

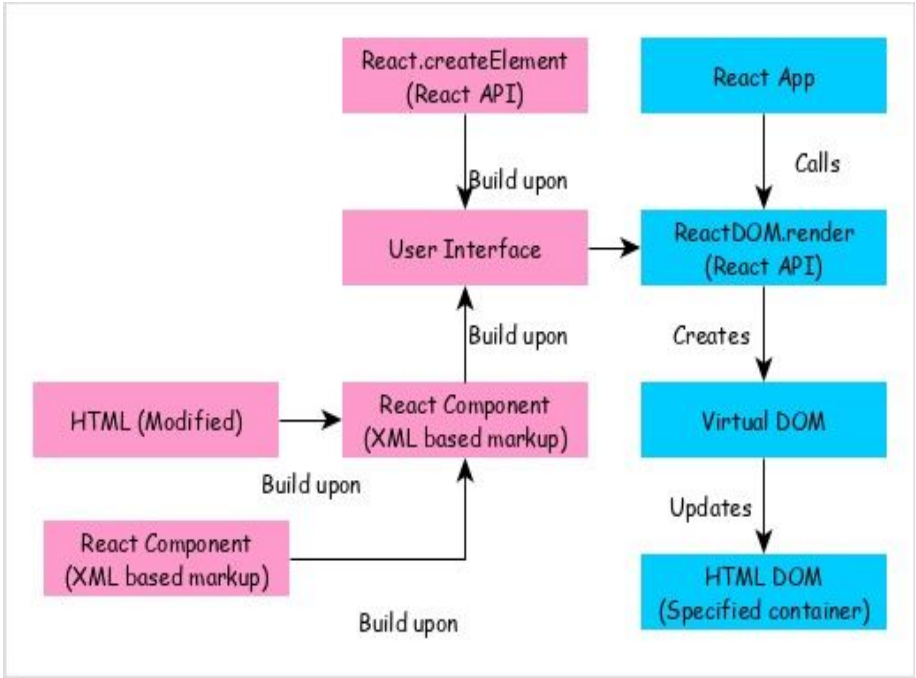
Instead, we are writing virtual components that will turn into the DOM leading to smoother and faster performance.

Workflow of a React application

React app calls ReactDOM.render method by passing the user interface created using React component (coded in either JSX or React element format) and the container to render the user interface.

ReactDOM.render processes the JSX or React element and emits Virtual DOM.

Virtual DOM will be merged and rendered into the container.



Workflow of a React application

- React app starts with a single root component.
- Root component is build using one or more component.
- Each component can be nested with other component to any level.
- Composition is one of the core concepts of React library. So, each component is built by composing smaller components instead of inheriting one component from another component.
- Most of the components are user interface components.
- React app can include third party component for specific purpose such as routing, animation, state management, etc.