# React

1. React is an open-source frontend JavaScript library which is used for building user interfaces especially for single page applications.

It is used for handling view layer for web and mobile apps.

React was created by Jordan Walke, a software engineer working for Facebook.

React was first deployed on Facebook's News Feed in 2011 and on Instagram in 2012.

1. The major features of React are:

* It uses VirtualDOM instead of RealDOM considering that RealDOM manipulations are expensive.

The Virtual DOM (VDOM) is an in-memory representation of Real DOM. The representation of a UI is kept in memory and synced with the "real" DOM. It's a step that happens between the render function being called and the displaying of elements on the screen. This entire process is called reconciliation.

Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation

Then the difference between the previous DOM representation and the new one is calculated. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.

* Supports **server-side rendering**.
* Follows **Unidirectional** data flow or data binding.
* Uses reusable/composable UI components to develop the view.

1. Create React components using:

* Function components

function Greeting({ message }) {

return <h1>{`Hello, ${message}`}</h1>

}

* Class Components

class Greeting extends React.Component {

render() {

return <h1>{`Hello, ${this.props.message}`}</h1>

}

}

1. State and Props
2. Camel casing for event handling

<button onClick={activateLasers}>

activateLasers = () => {

console.log('Activated Lasers.')

};

1. Bind event handers

* Binding in constructor

class CounterApp extends React.Component {

constructor(props) {

super(props)

this.handleClick = this.handleClick.bind(this)

}

handleClick() {

//handle method

}

render() {

<button onClick={this.handleClick}>clickme</button>

}

}

* Arrow functions

class CounterApp extends React.Component {

constructor(props) {

super(props)

}

handleClick() {

//handle method

}

render() {

<button onClick={(event) => this.handleClick(event)}>clickme</button>

}

}

1. Inline Conditional Expressions

<h1>Hello!</h1>

{

messages.length > 0 && !isLogin?

<h2>

You have {messages.length} unread messages.

</h2>

:

<h2>

You don't have unread messages.

</h2>

}

1. Iterate over list of items to display

let fruits = ['Apple', 'Orange', 'Banana']

const displayFruits = fruits.map((fruit, index) =>

<li key={index}>

{fruit.text}

</li>

)

1. Super constructor with props

class MyComponent extends React.Component {

constructor(props) {

super(props)

console.log(this.props) // prints values in props of parent component

}

render() {

console.log(this.props) // prints values in props of parent component

}

}

class MyComponent extends React.Component {

constructor(props) {

super()

console.log(this.props) // prints undefined

}

render() {

console.log(this.props) // prints values in props of parent component

}

}

### validation on props in React

When the application is running in *development mode*, React will automatically check all props that we set on components to make sure they have *correct type*. If the type is incorrect, React will generate warning messages in the console.

It's disabled in *production mode* due to performance impact. The mandatory props are defined with isRequired.

* PropTypes.number
* PropTypes.string
* PropTypes.array
* Etc.

import React from 'react'

import PropTypes from 'prop-types'

class User extends React.Component {

static propTypes = {

name: PropTypes.string.isRequired,

age: PropTypes.number

}

render() {

return (

<div>

<h1>{`Welcome, ${this.props.name}`}</h1>

<h2>{`Age, ${this.props.age}`}</h2>

</div>

)

}

}