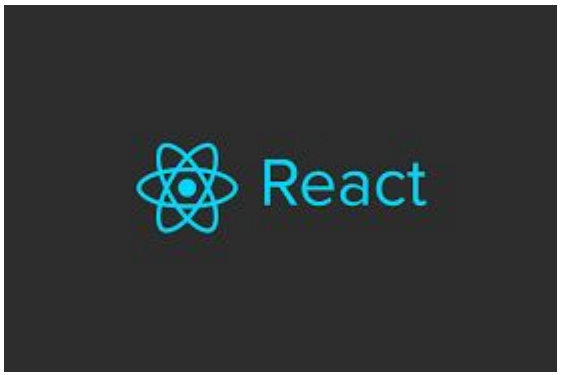
React

By Dr. Ashfaq Shaikh Ph.D Computer Engg.
Assit. Professor MHSS Mumbai



Outline

- Introduction React.
- Why React.
- JSX (Java Script Extension).
- Component.
- Component Life Cycle.
- State and Props.
- Router and single page applications.

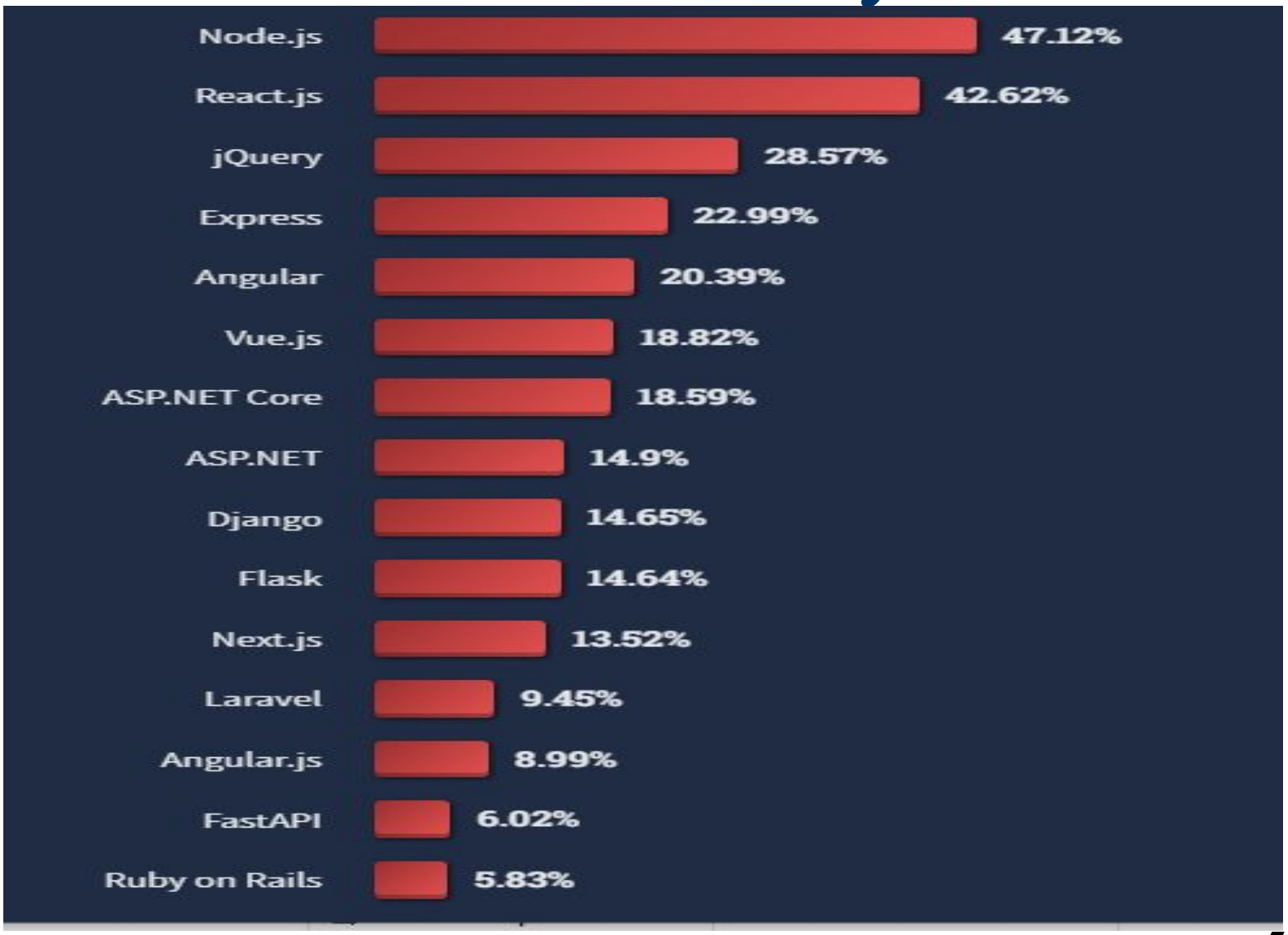
React Introduction

- 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.

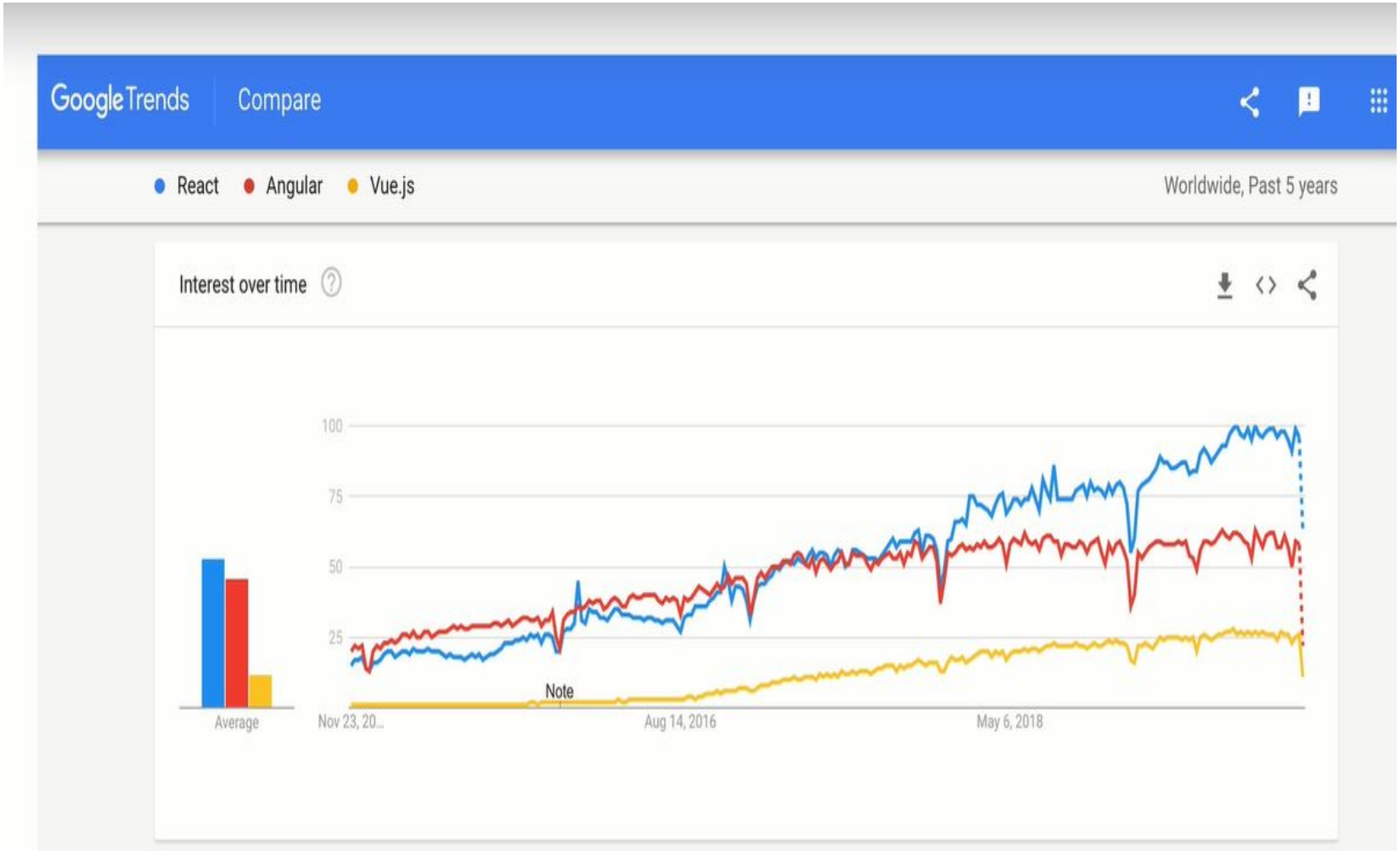
React Introduction...

- It was created by Jordan Walke, who was a software engineer at Facebook.
- It was initially developed and maintained by Facebook
- Facebook developed ReactJS in 2011 in its newsfeed section,
- Released to the public in the month of May 2013.
- React offers various extensions for entire application architectural support.
- Flux and React Native, beyond mere UI.

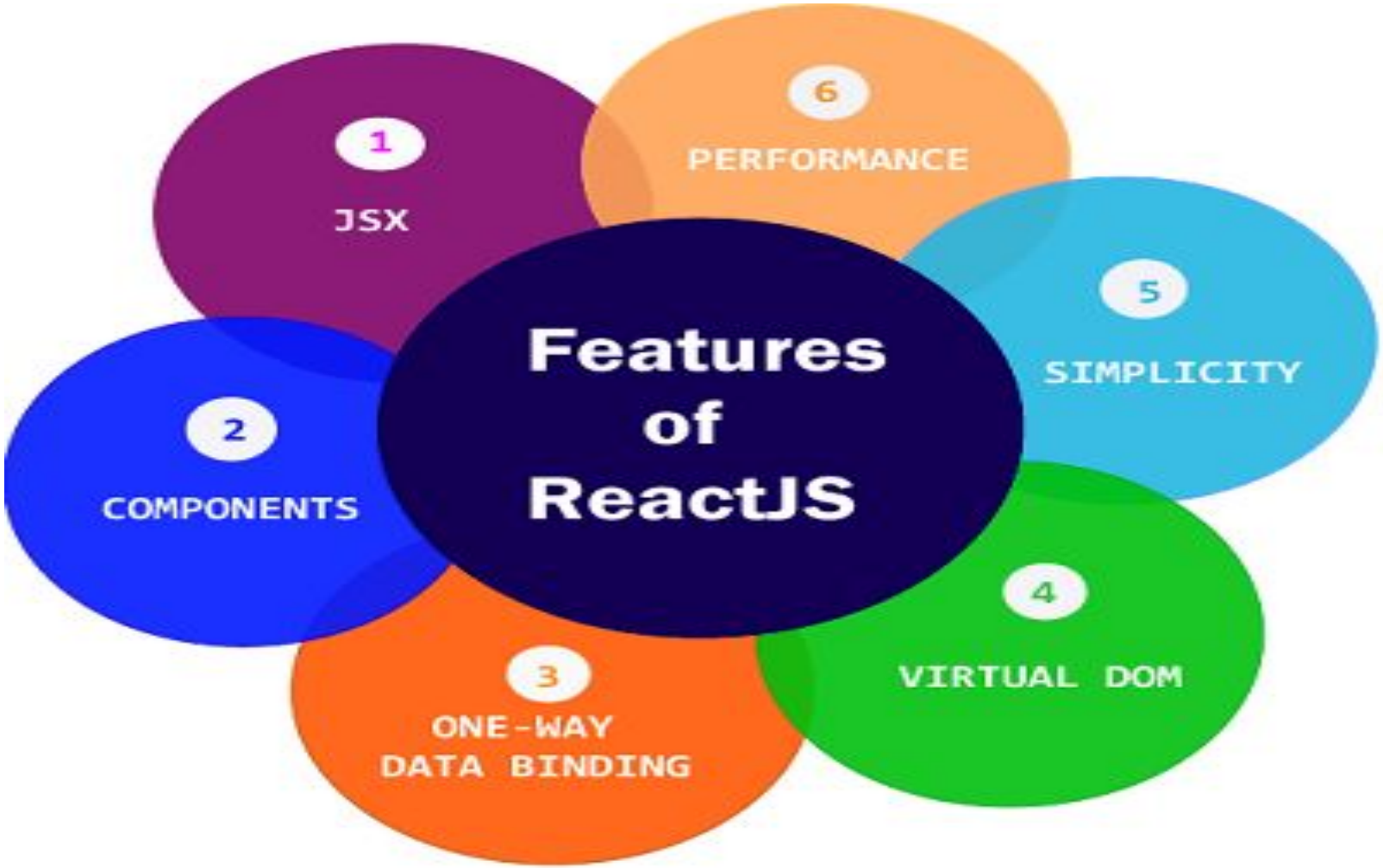
Stack Overflow Survey



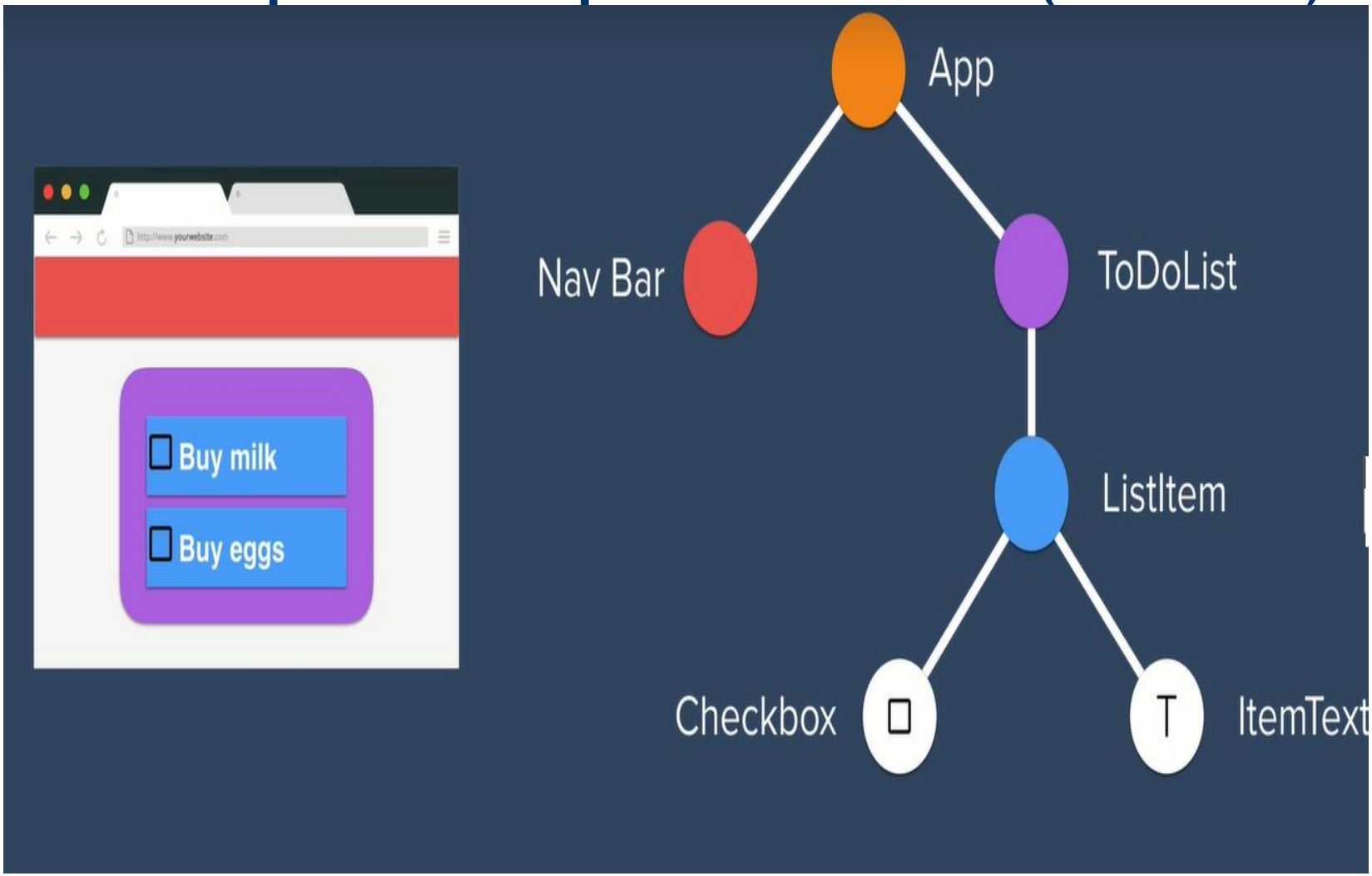
Google Trends



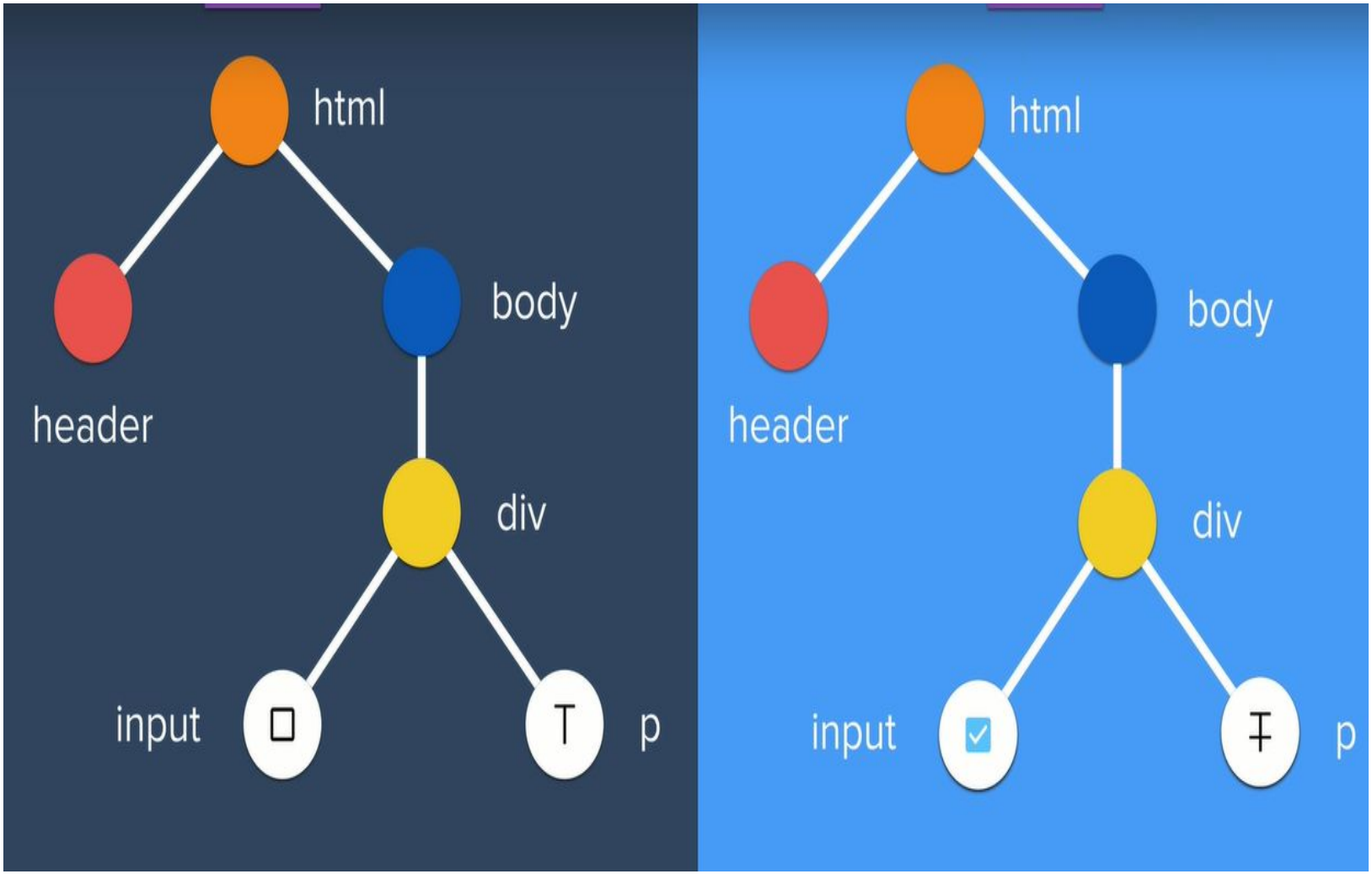
Features of React



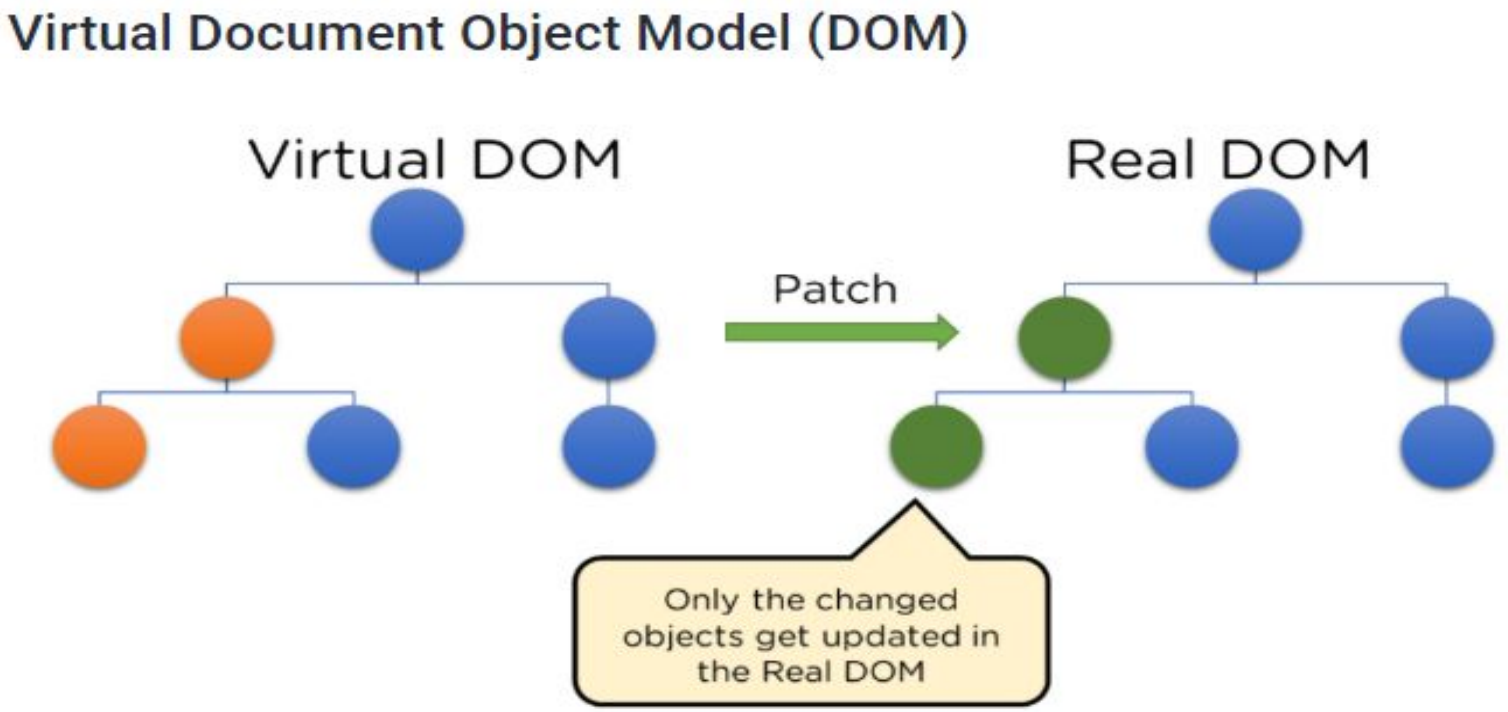
React Break Complex structure to simple component tree (DOM)



Old DOM New DOM



React Features (Virtual DOM)



The Virtual DOM is React's lightweight version of the Real DOM. Real DOM manipulation is substantially slower than virtual DOM manipulation. When an object's state changes, Virtual DOM updates only that object in the real DOM rather than all of them.

JSX

JSX

JSX is a JavaScript syntactic extension. It's a term used in React to describe how the user interface should seem. You can write HTML structures in the same file as JavaScript code by utilizing JSX.

React JSX Programming Steps

- Create a new project either using

>npx create-react-app myapp

To Start Server
>npm start

This will start working server on port no 3000

JSX code always index.js

Import React as “react”;

Import ReactDOM as “react-dom”;

ReactDOM.render(what to display, where to display)

Ex.
ReactDOM.render(<h1>Hello World</h1>,
Document.getElementById(“root”));

To Test output
  http://localhost:3000/index.js

JSX Codes

Import React as “react”;

Import ReactDOM as “react-dom”;

 const name= “mhss”

ReactDOM.render(<h1>Hello {name}</h1>,
Document.getElementById(“root”));

To Test output
  http://localhost:3000/index.js

JSX Codes

```
Import  React as “react”;

Import ReactDOM as “react-dom”;

 const fname= “mhss”;
Const lname=“admin”;

ReactDOM.render(<h1>Hello {fname} {lname}</h1>,
Document.getElementById(“root”));
```

JSX Codes

```
Import  React as “react”;

Import ReactDOM as “react-dom”;

 const fname= “mhss”;
Const lname=“admin”;

ReactDOM.render(<h1>Hello {fname . “ ” .
   Lname}</h1>,
Document.getElementById(“root”));
```

JSX Arithmetic Operations

```
Import  React as “react”;

Import ReactDOM as “react-dom”;

Let a,b,c;
A=100;b=200,c=a+b;

ReactDOM.render(<h1>Addition= { c }</h1>,
Document.getElementById(“root”));
```

JSX Display More then one HTML

```
Import  React as “react”;

Import ReactDOM as “react-dom”;

 const fname= “mhss”;
Const lname=“admin”;

ReactDOM.render(
<div>
<h1>Hello {fname . “ ” . Lname}</h1>
<p>welcome to MHSS</p>
<div>,
Document.getElementById(“root”));
```

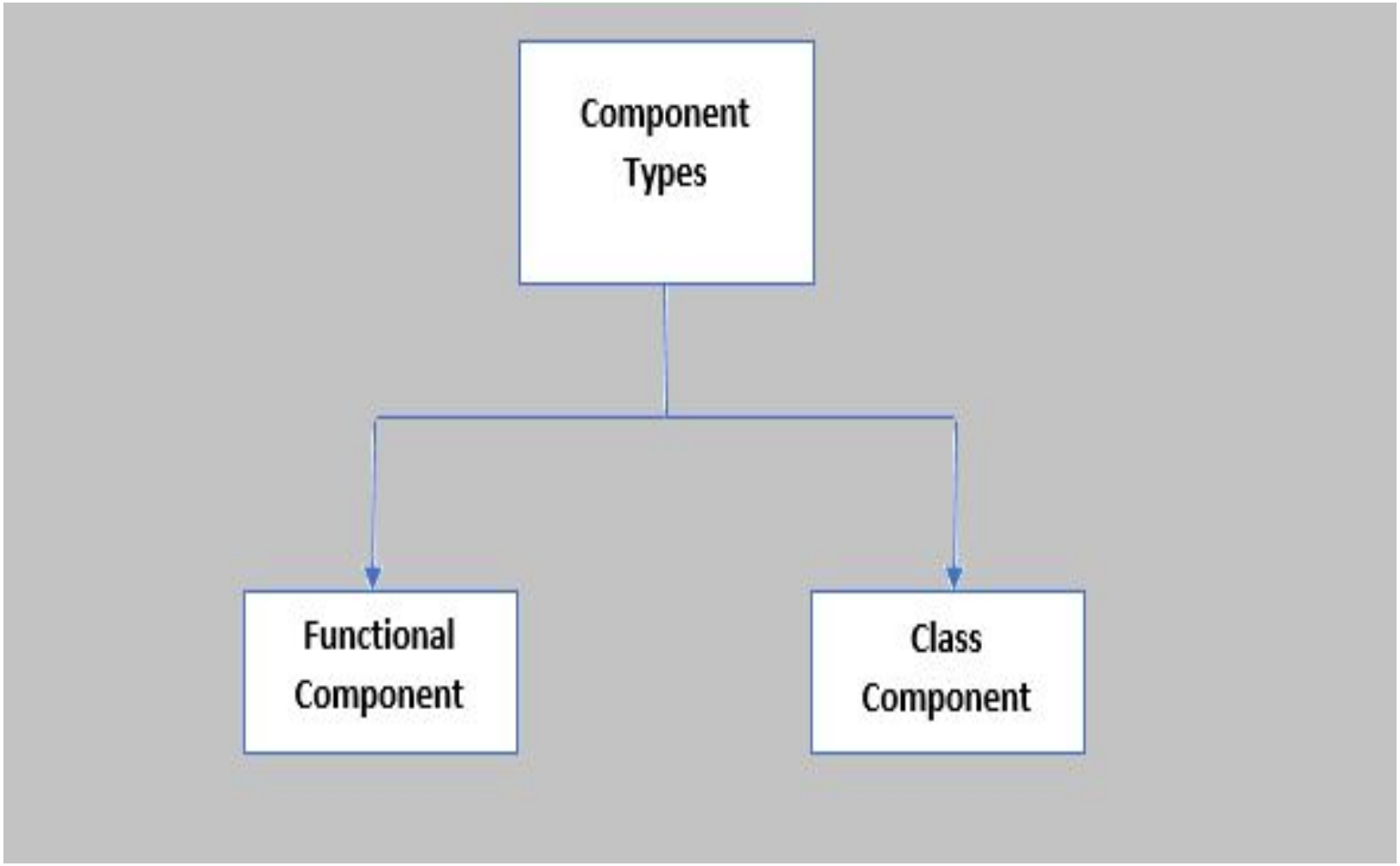
Component in React

- React component is the building block of a React application.
- A React component represents a small chunk of user interface in a webpage.
- The primary job of a React component is to render its user interface and update it whenever its internal state is changed. In addition to rendering the UI,

Components

- React component accomplish these feature using three concepts −
- Properties − Enables the component to receive input.
- Events − Enable the component to manage DOM events and end-user interaction.
- State − Enable the component to stay stateful. Stateful component updates its UI with respect to its state.

Types



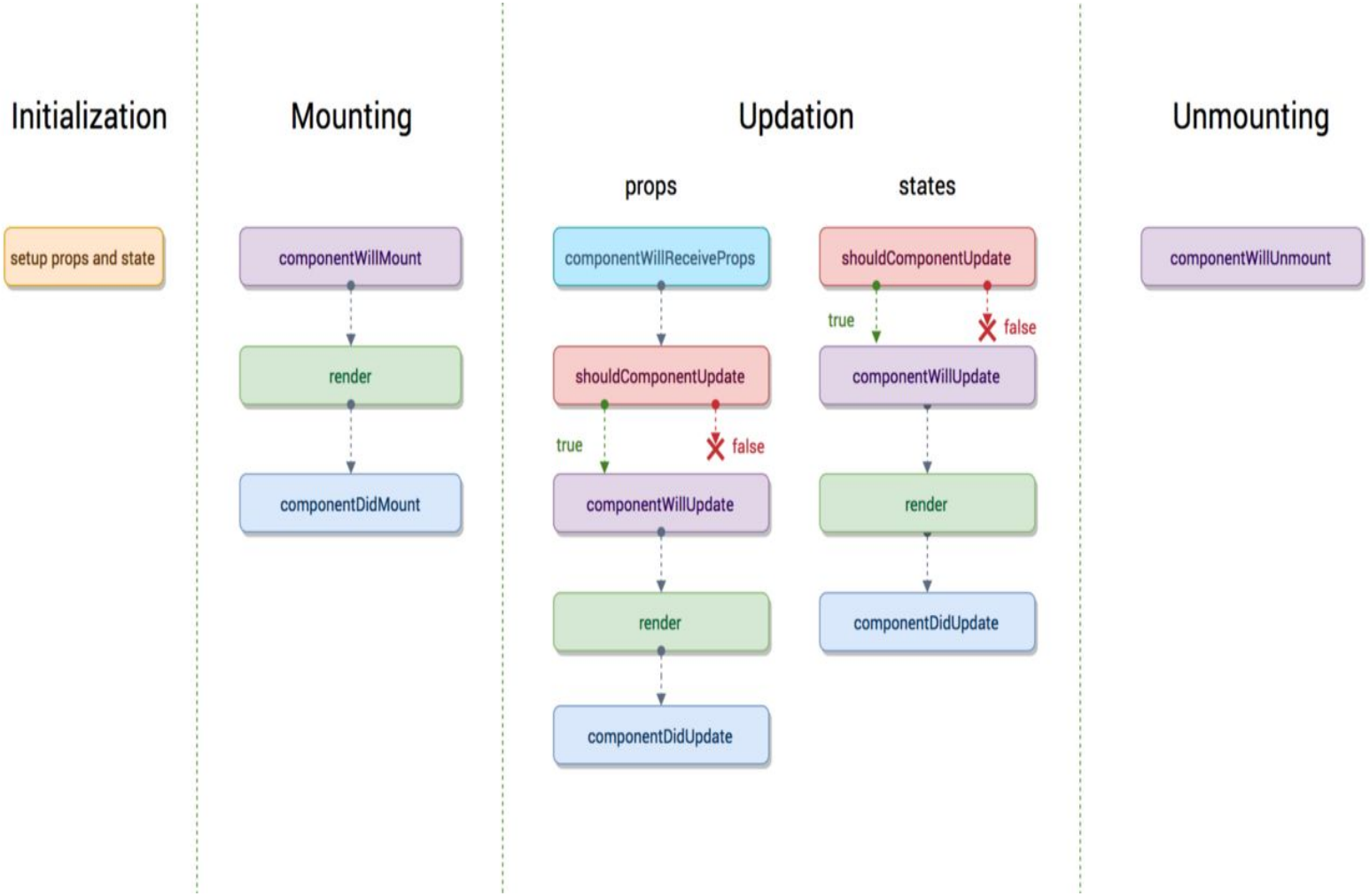
Function vs class components

Functional Component	Class Component
• Functional Components is a plain JavaScript, you do not have a choice to set the state in functional component. • There is no render function we are using in functional components. • Functional components only accept the props as an argument. • Functional components are sometimes called stateless components.	• Class components we have a feature to set the set state in component. • In class components, we have a render function which is use to return the react elements. • In class components, we have both options use the props and set the state also. • Class components are sometimes called stateful components.

Summary

- Functional components take only props as an argument and return react elements.
- Functional components are state less components and no life cycle methods.
- Class components extends by react base class name is **Component**. It help to set the state and implement the life cycle methods.
- Class components we have render method which is used for return the react elements.

Life cycle


Initialization
setup props and state
Mounting
componentWillMount
render
componentDidMount
Updation
props
componentWillReceiveProps
shouldComponentUpdate
true
false
componentWillUpdate
render
componentDidUpdate
states
shouldComponentUpdate
true
false
componentWillUpdate
render
componentDidUpdate
Unmounting
componentWillUnmount

Examples

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Car() {
  return <h2>Hi, I am a Car!</h2>;
}

const root =
    ReactDOM.createRoot(document.getElementByI
    d('root'));
root.render(<Car />);
```

Function Component with props

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Car(props) {
  return <h2>I am a {props.color} Car!</h2>;
}

const root =
    ReactDOM.createRoot(document.getElementById('r
    oot'));
root.render(<Car color="red"/>);
```

Nesting components

```
function Car() {
  return <h2>I am a Car!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car />
    </>
  );
}

const root =
    ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

Function from file

This is the new file, we named it "Car.js":

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}

export default Car;

//from index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import Car from './Car.js';

const root
    =ReactDOM.createRoot(document.getElementById('root')
    );
root.render(<Car />);
```

Class components

```
class ExpenseEntryItem extends React.Component {
   render() {
      return (
         <div>Hello</div>
      );
   }
}
```

Class Components

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  render() {
    return <h2>I am a {this.state.color} Car!</h2>;
  }
}

const root =
    ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

Class components

```
import React from 'react';
import './ExpenseEntryItem.css';

class ExpenseEntryItem extends React.Component {
   render() {
      return (
         <div>
            <div><b>Item:</b> <em>Mango Juice</em></div>
            <div><b>Amount:</b> <em>30.00</em></div>
            <div><b>Spend Date:</b>
    <em>2020-10-10</em></div>
            <div><b>Category:</b> <em>Food</em></div>
         </div>
      );
   }
}
export default ExpenseEntryItem;
```

Props

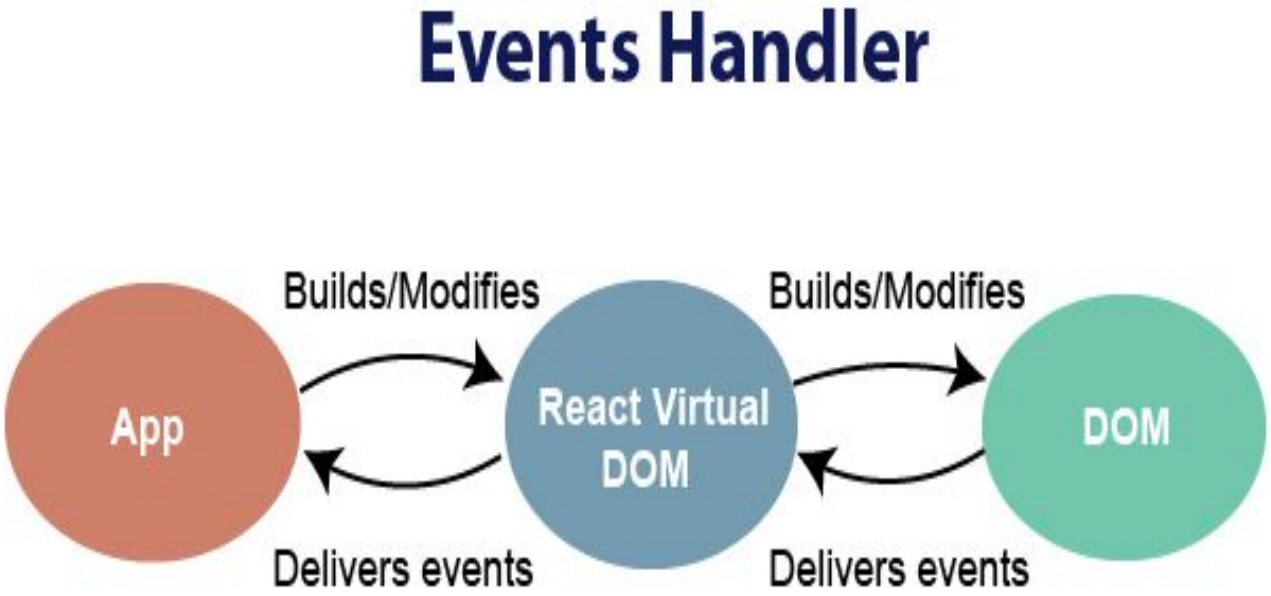
```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  render() {
    return <h2>I am a {this.props.color} Car!</h2>;
  }
}

ReactDOM.render(<Car color="red"/>,
    document.getElementById('root'));
```

React Events

- Just like HTML DOM events, React can perform actions based on user events.
- An event is an action that could be triggered as a result of the user action or system generated event.
- For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.
- React has its own event handling system which is very similar to handling events on DOM elements.
- The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.

React Event Handler

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.



Handling events with react have some syntactic differences from handling events on DOM. These are:

Example

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );   }

const root =
    ReactDOM.createRoot(document.getElementById('r
    oot'));
root.render(<Football />);
```

Passing Args in event

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );
}
const root =
    ReactDOM.createRoot(document.getElementById('r
    oot'));
root.render(<Football />);
```

React State

- React components has a built-in state object.
- The state object is where you store property values that belongs to the component.
- When the state object changes, the component re-renders.

React State

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {brand: "Ford"};
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

State..

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    ); }}
```

React Form

- Handling forms is about how you handle the data when it changes value or gets submitted.
- In HTML, form data is usually handled by the DOM.
- In React, form data is usually handled by the components.
- When the data is handled by the components, all the data is stored in the component state.
- You can control changes by adding event handlers in the onChange attribute.

React Form

- React uses forms to allow users to interact with the web page.
- In React, form data is usually handled by the components.
- When the data is handled by the components, all the data is stored in the component state.
- You can control changes by adding event handlers in the onChange attribute and that event handler will be used to update the state of the variable.

```
function MyForm() {
  const [name, setName] = useState("");

  const handleSubmit = (event) => {
    event.preventDefault();
    alert(`The name you entered was: ${name}`);
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
      <input type="submit" />
    </form>
  )}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyForm />);
```

React Routes

- React Router is a standard library for routing in React.
- It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.
- Step 1: cd into your project directory i.e geeks.
- Step 2: To install the React Router use the following command:
- npm install – -save react-router-dom@5 or npm i react-router-dom
- (its a updated command)
- Step 3: Start building react routes

Example React Routes

```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from
    "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import Blogs from "./pages/Blogs";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";

export default function App() {
  return (
    <BrowserRouter>
```

Example...

```
<Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}
const root =
    ReactDOM.createRoot(document.getElementById('r
    oot'));
root.render(<App />);
```