REACT JS 18X

- React upto 17 are different
- Components, Hooks
- API
- State Management
- Redux
- Material UI
- React Native
- End to End Application

MERN Stack

MongoDB	Database
Express	Middleware
React	Front End
Node	Server Side

1. What is React ?
A. It is a JavaScript library used to build UI.

2. What is difference between React and Angular?
A. React is a library.
Angular is a framework.
If your project is about build application with lot of server side frameworks and you want just your UI be more effects then "React" is good.
If your project is about building app with minimum use of frameworks in backend and maximum is expected to handle client side then better "Angular".

Netflix, Amazon, FaceBook, Instagram => React
OS Window Mobile - Angular, PWA

3. Is there any difference between React and React JS?
A. No Difference

4. Is there any difference between Angular and Angular JS?
A. Yes

5. Why we need Angular or React like library and framework to build UI as we already have jQuery and JavaScript?

6. What are the challenges in modern web development?
- Unified UX
- Fluid UX
- Loosely coupled and extensible

7. What is solution?
A. Better build SPA [Single Page Applications] - Twitter

8. How to build SPA? Can we use JS, JQ?

A. Yes.
- JS DOM
- Logic, Coding funcitons
- Ajax

9. What is solution?
A. React, Angular, Vue, Knockout, Ember, BackBone etc..

Summary
- What is React?
- What is difference between a library and framework?
- What is difference between React and Angular?
- Why we need React and Angular?
- Challenges in modern web development
- SPA

Features of React

1. It is component based.
    - It is faster in designing complex UI
    - It enables reusability
    - It is easy to extend
    - It is easy to test

2. It uses Virtual DOM

Browser Architecture
- UI : It describes the browser interface, which includes title bar, shortcut buttons, addressbar, scroll bar.

- Browser Engine : It translates your HTML.

- Rendering Engine : It presents your markup in browser.

- JavaScript Interpreter : It translated JavaScript - JIT [Just-in-Time]

- Network : It tracks the performance of page

- UI Backend : It defines the backend API's or extentions for browser.

- Data Persistance : It includes storage for browser
    a) Local Storage
    b) Cookies
    c) Session Storage
    d) Shared Storage
- Various browser engines & rendering

v8, chakra, edgeHTML, webkit, spider monkey, gecko etc..

What is DOM?
- It is a hierarchy of presenting elements.
- It refers to Document Object Model.
- It comprises of parent and child nodes.
- Nodes are converted from Tokens
- Tokens are from chars
- Chars are from Bytes
- Bytes are from markup.

What is Shadow DOM?
- It is a hierarchy of nodes in a component.

What is Virutal DOM?
- It is a duplicate copy of actual DOM.

Features of React JS
1. It is component based
2. It uses Virtual DOM
    a) DOM
    b) Shadow DOM
    c) Virtual DOM
3. It is faster.
4. It is modular
    - It uses only the library that is required for application.
    - It is not legacy.
    - Application specific library
    - Application light weight
    - It uses less memory
5. It uses built-in Async methods [Implicitly Asynchronous]
6. You can easily plugin any 3rd party

Issues with react
1. React is not developed for what you are using.
2. Lot of GAP's.
3. Fill the GAP by using 3rd party library
4. Lot of 3rd party libraries are required
5. Pace of development and poor documentation.

Setup Environment for React

1. Download and Install "NodeJS" on your PC

NodeJS	version 18x
NPM	version 8x

-NPM is package manager.

-Package Manager is a tool used for installing, updating and removing library.
[NPM, Yarn, Bower, NuGet ...]

https://nodejs.org/en/

```
C:\> npm   -v
C:\> node  -v
```

2. Download and Install "Visual Studio Code" editor

https://code.visualstudio.com/

3. Open VS Code and Install extention
"Live Server"

Using React in existing web application:

1. Create a new folder for Web Application

D:\web-app

2. Open folder in Visual Studio code

3. Open Terminal

```
> npm init  -y
```
[It generates package.json]

4. Add folders

- public	: It is for static resources , HTML, Images, documents
- src	: It is for dynamic resources, css, scss, ts, js etc..

5. Add startup page

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Index</title>
</head>
<body>
    <h1>Welcome to React - 18</h1>
    <p>This page is not using react.</p>
    <p>React is in Home Page</p>
    <a href="public/home.html">Home</a>
</body>
</html>
```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
</head>
<body>
    <noscript>Please enable javascript on your browser.</noscript>
    <h1></h1>
</body>
</html>
```

- You can use react library from CDN, react in HTML page requires 3 libraries

a) react	: It is the core library for React.
b) react-dom	: It is the library to handle virtual DOM
c) babel	: It is JavaScript compiler used by react.

reactjs.org => docs => CDN Links => Development

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script&gt;
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script&gt;
```

https://babeljs.io/

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script&gt;
```

Ex: Upto React 17

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script&gt;
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script&gt;
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script&gt;
    <script type="text/babel">
          ReactDOM.render("Hello ! React JS", document.querySelector("h1"));
    </script>
</head>
<body>
    <noscript>Please enable javascript on your browser.</noscript>
    <h1></h1>
</body>
</html>
```

Ex: React 18

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script&gt;
    <script crossorigin src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script&gt;
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script&gt;
    <script type="text/babel">
          const root = ReactDOM.createRoot(document.getElementById("root"));
          root.render("Hello ! Welcome to React 18");
    </script>
</head>
<body>
    <noscript>Please enable javascript on your browser.</noscript>
    <div id="root"></div>
</body>
</html>
```

Features of React
Issues with React
Setup Environment for React
- Node JS [NPM]
- Visual Studio Code
- Created a web application
- Use react in existing web application with CDN links.

- How to use react in existing web application using offline library ?
    a) React

b) React DOM
c) Babel

1. Open Terminal and Install following libraries

```
>npm install  react    --save
>npm install  react-dom  --save
>npm install  @babel/standalone
```

2. Any library installed using NPM package manager copies its files into a folder "node_modules"

Note: Modules of JavaScript uses various module systems
a) Common JS
b) UMD (Universal Module Distribution)
c) AMD (Asynchronous Module Distribution)
d) ES [ECMAScript2022]

3. Link the files for React and React DOM from UMD system

```
umd / react.development.js
umd / react-dom.development.js
@babel/ babel.js
```

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Home</title>
   <script src="../node_modules/react/umd/react.development.js"></script>
   <script src="../node_modules/react-dom/umd/react-dom.development.js"></script>
   <script src="../node_modules/@babel/standalone/babel.js"></script>
   <script type="text/babel">
       const root = ReactDOM.createRoot(document.getElementById("root"));
       root.render("Hello ! React 18");
   </script>
</head>
<body>
   <noscript>Please enable javascript on your browser.</noscript>
   <div id="root"></div>
</body>
</html>
```

Note: Babel is a compiler used for JavaScript and JSX.
JSX is JavaScript Extention Library

JavaScript DOM methods can handle browser DOM not Virtual DOM.

JSX methods can handle virtual DOM not browser DOM.

JSX rules:

- JSX will not allow individual elements in UI.
- JSX requires a container or a fragment to handle elements.

```
<h1>Welcome</h1>        // invalid
<p> React 18 </p>

<div>
    <h1> Welcome </h1>
    <p> React 18 </p>
</div>
```

- JSX don't have support for void elements.

```
<img> </img>        or    <img />
<input type="text"> </input>
<input type="text" />
```

Ex:
home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <script src="../node_modules/react/umd/react.development.js"></script>
    <script src="../node_modules/react-dom/umd/react-dom.development.js"></script>
    <script src="../node_modules/@babel/standalone/babel.js"></script>
    <script type="text/jsx">
         function Login(){
            return(
                <div>
                    <h2>User Login</h2>
                    <dl>
                        <dt>User Name</dt>
                        <dd><input type="text"></input></dd>
                        <dt>Password</dt>
                        <dd><input type="password"/></dd>
                    </dl>
                    <button>Login</button>
                </div>
            )
         }
```

```
        function MainComponent(){
          return(
              <div>
                  <h2>Netflix - Home </h2>
                  <Login />
                  <hr />
                  <Login/>
              </div>
          )
        }
        const root = ReactDOM.createRoot(document.getElementById("root"));
        root.render(<MainComponent/>);
    </script>
</head>
<body>
    <noscript>Please enable javascript on your browser.</noscript>
    <div id="root"></div>

</body>
</html>
```

Netflix Application Design

1. Create a new folder for Netflix Project

```
E:\react-netflix
```

2. Open project folder in VS Code

3. Add following folder into project

    - public
    - src

4. Run following commands in Terminal

```
> npm  init  -y
> npm  install  react  --save
> npm  install  react-dom  --save
> npm  install  @babel/standalone --save
> npm  install   bootstrap --save
> npm  install   bootstrap-icons --save
> npm  install   jquery --save

[node_modules]
```

5. Add a startup page for project

    index.html

- JSX can't use HTML attributes, it uses JavaScript properties.

```
<img  src=""  width=""   height=""   class="">        // not valid in JSX

img.src
img.width
img.height
img.className

<img src=""  width=""  height=""   className=""  />   // valid in JSX
```

Ex:
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Netflix</title>
    <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../node_modules/bootstrap-icons/font/bootstrap-icons.css">
    <style>
        .netflix-back {
            background-image: url("netflixbanner.jpg");
            background-size: cover;
            color:white;
            height: 768px;
        }
        section {
            background-color: rgba(0,0,0,0.5);
            height: 768px;
        }
        .brand-title {
            font-size: 35px;
            color:red;
            font-weight: bold;
        }
        main {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 500px;
            text-align: center;
        }
        main h1 {
            font-size: 50px;
        }
        main p {
            font-size: 30px;
        }
        .register-box {
            width: 800px;
        }
        .register-box p {
            font-size: 20px;
        }
    </style>

    <script src="../node_modules/react/umd/react.development.js"></script>
    <script src="../node_modules/react-dom/umd/react-dom.development.js"></script>
    <script src="../node_modules/@babel/standalone/babel.js"></script>
    <script type="text/jsx">
        function HeaderComponent(){
            return(
                <header className="p-3 d-flex justify-content-between">
                    <div className="brand-title">NETFLIX</div>
                    <div className="d-flex">
                        <div className="dropdown me-3">
                            <button className="btn btn-dark dropdown-toggle">
                                <span className="bi bi-globe"></span>
                                Language
                            </button>
                        </div>
                        <div>
                            <button className="btn btn-danger">
                                Signin
                            </button>
                        </div>
                    </div>
                </header>
            )
        }

        function MainComponent(){
            return(
                <main>
                    <div>
                        <h1>Unlimited movies, TV shows and more.</h1>
                        <p>Watch anywhere. Cancel anytime.</p>
                        <RegisterComponent />
                    </div>
                </main>
            )
        }

        function RegisterComponent(){
            return(
                <div className="register-box">
                    <p>Ready to watch? Enter your email to create or restart your membership.</p>
                    <div className="input-group input-group-lg">
                        <input type="email" placeholder="Your email address" className="form-control"
    />
                        <button className="btn btn-danger">
                            Get Started <span className="bi bi-chevron-right"></span>
                        </button>
                    </div>
                </div>
            )
        }

        function IndexComponent(){
            return(
                <div className="netflix-back">
                  <section>
                     <HeaderComponent />
                     <MainComponent />
                  </section>
                </div>
            )
        }
        const root = ReactDOM.createRoot(document.getElementById("root"));
        root.render(<IndexComponent />);
    </script>
    </head>
    <body>
    <noscript>Please enable JavaScript on your browser.</noscript>
    <div id="root"></div>
    <script src="../node_modules/jquery/dist/jquery.js"></script>
    <script src="../node_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>
    </body>
    </html>
```

React Application Components

- JavaScript Module System is used to build React Application
- Module is a set of classes, functions and variables.
- Every JavaScript file is considered as a Module.

```
            HomeComponent.js        => HomeComponent

        export function HomeComponent()
        {
        }
```

- You can import the library and use in any page

```
      import   { HomeComponent  }  from  "HomeComponent.js"

      root.render(<HomeComponent />);
```

Create a new React Application:

1. Open any location on your PC in command prompt

```
E:\>
```

2. Run the command

```
E:\>npx  create-react-app   react-shopping
```

3. Open react-shopping folder in VS Code

4. You can run project

```
E:\react-shopping> npm  start
```

Server is listening on : http://localhost:3000
http://localhost:5500

File System of React Application

1. node_modules : It comprises of all library files.

2. public : It comprises of all static resources html, images, documents..

3. src : It comprises of all dynamic resources css, js, jsx, ts, scss ..

4. .gitignore : It configures the resources not to includes while publishing on GIT.

5. package.json : It comprises of project meta data.

6. package-lock.json : It is used to install and update dependencies.

```
>npm  install
```

7. readme.md : It is help document for developers

Components and Data Binding

React Components
- A component comprises of
    a) Presentation
    b) Styles
    c) Logic
- Presentation is done by using HTML
- Styles are defined by using CSS
- Logic is defined by using JavaScript | JSX | TypeScript
- It is a reusable design with functionality.
- It provides customizable designs.
- Components are building blocks for application.
- React component can be
    a) A function
    b) A class

Creating a component using functions

- Every function used for creating component must start with Uppercase letter.

```
function  Login()
{
}
```

- Every component function must return a presentation. It can't be void.

```
      function  Login()
      {
        return (<markup></markup>);
      }
```

- Component function can be parameterless or parameterized.

- Component parameters are reffered as properties.

- Properties are used to customize a component.

```
    function  Login(props)
    {
       <input type=props.controlType>
    }

    Login(props.email);
    Login(props.text);
    Login(props.date);
```

- Component function can't use immutable types.

- It is not recommended to used immutable types in component.

```
    function Login()
    {
        var username;            // not good
    }
```

- According to project tradition every component at high level can have 3 files

```
        .js          [markup and logic]
        .css        [styles]
        .spec.js    [testing]
        .test.js       [testing]
```

- Component file can be created with 2 types of extentions

```
        .js       ]       JavaScript
        .jsx       ]

        .ts       ]       TypeScript
        .tsx    ]
```

- A component function in module must be defined with "export"

```
        export  function Login() { }                import  { Login } from  "Login";
        export  default function Login() { }        import  Login from "Login";

                    (or)

        function Login() { }
        export  default Login;

                    (or)

        const  Login = () => { }

                    (or)

        const  Login = function() { }
```

Ex:
 1. Add a new folder
      "components"

2. Add a folder for component "login"

3. Add following files

      "login.jsx"

```
import "./login.css";

export function Login()
{
    return(
        <div className="container-fluid">
            <form>
                <h2>User Login</h2>
                <dl>
                    <dt>User Name</dt>
                    <dd><input type="text" className="form-control"/></dd>
                    <dt>Password</dt>
                    <dd><input type="password" className="form-control"/></dd>
                </dl>
                <button className="btn btn-primary w-100">Login</button>
            </form>
        </div>
    )
}

        login.css

.container-fluid {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 500px;
}
form {
    padding: 20px;
    border:2px solid gray;
}
```

Data Binding in Component

- It is the process of accessing data from source and binding to UI.
- JavaScript uses lot of DOM methods.
- React uses data binding expression " { } "

```
            var name = "Admin Login";

            <h2> { name } </h2>
```

Data Binding in React

Data Binding in React
- It is the process of accessing data from source and binding to UI.
- JavaScript and jQuery can handle data binding using DOM methods and jQuery functions.

```
                var  username = "John";

                <p> </p>

                document.querySelector("p").innerHTML = username
```

- React uses "Data Binding Expression" => { }

```
                 var  username = "John";

                <p> {username} </p>
```

- React requires various techniques to bind data which is complex.
    a) Primitive Types
        number, string, boolean, null, undefined, symbol

b) Non Primitive types
    array, object, map

Ex: Primitive Types

data-binding.jsx

```
export function DataBinding()
{
    var id = 1;
    var name = "Samsung TV";
    var price = 45000.44;
    var stock = true;
    return(
      <div className="container-fluid">
         <h2>Product Details</h2>
         <dl>
            <dt>Product Id</dt>
            <dd>{id}</dd>
            <dt>Name</dt>
            <dd>{name}</dd>
            <dt>Price</dt>
            <dd>{price}</dd>
            <dt>Stock</dt>
            <dd>{(stock==true)?"Available":"Out of Stock"}</dd>
         </dl>
      </div>
    )
}
```

Ex: Object - JSON

data-binding.jsx

```
export function DataBinding()
{
    var product = {
       "Id" : 1,
       "Name": "Samsung LED TV",
       "Price": 145000.44,
       "Stock": true
    }
    return(
      <div className="container-fluid">
```

```
            <h2>Product Details</h2>
            <dl>
                <dt>Product Id</dt>
                <dd>{product.Id}</dd>
                <dt>Name</dt>
                <dd>{product.Name}</dd>
                <dt>Price</dt>
                <dd>{product.Price}</dd>
                <dt>Stock</dt>
                <dd>{(product.Stock==true)?"Available":"Out of Stock"}</dd>
            </dl>
        </div>
    )
}
```

Ex: Array
- To present array and bind its data to UI you need various array methods

```
            toString()
            join()
            slice()
            map()
            find()
            filter() etc...
```

Syntax:

```
            collection.map((item)=> <p> { item } </p>)
```

data-binding.jsx

```
export function DataBinding()
{
    var categories = ["Electronics", "Footwear", "Fashion"];
    return(
        <div className="container-fluid">
            <h2>Categories</h2>
            <ul className="list-unstyled">
                {
                    categories.map((category)=>
                     <li><a href="#">{category}</a></li>
                    )
                }
            </ul>
            <div>
                {
                    categories.map((category)=>
                     <button className="btn btn-danger mb-2 d-block w-
25">{category}</button>
                    )
                }
            </div>
            <ul className="list-unstyled">
                {
                    categories.map((category)=>
                     <li><input type="checkbox"/> {category}</li>
                    )
                }
            </ul>
            <ol>
                {
                    categories.map((category)=>
                        <li>{category}</li>
                    )
                }
            </ol>
            <select>
                {
                    categories.map((category)=>
                      <option>{category}</option>
                    )
                }
            </select>
            <table className="table table-hover">
                <thead>
                    <tr>
                        <th>Categories</th>
                    </tr>
                </thead>
                <tbody>
                    {
                        categories.map((category)=>
                          <tr>
                            <td>{category}</td>
                          </tr>
                        )
                    }
                </tbody>
            </table>
        </div>
     )
}
```

Note: In virtual DOM if any element is repeating then every repeating element must have a unique identification key.

```
                        collection.map((item)=> <li key={item}> {item} </li>)
```

Ex:

```
export function DataBinding()
{
     var categories = ["Electronics", "Footwear", "Fashion"];
     return(
        <div className="container-fluid">
            <h2>Categories</h2>
            <ul className="list-unstyled">
                {
                    categories.map((category)=>
                     <li key={category}><a href="#">{category}</a></li>
                    )
                }
            </ul>
            <div>
                {
                    categories.map((category)=>
                     <button key={category} className="btn btn-danger mb-2 d-block w-
25">{category}</button>
                    )
                }
            </div>
            <ul className="list-unstyled">
                {
                    categories.map((category)=>
                     <li key={category}><input type="checkbox"/> {category}</li>
                    )
                }
            </ul>
            <ol>
                {
                    categories.map((category)=>
                        <li key={category}>{category}</li>
                    )
                }
            </ol>
            <select>
                {
                    categories.map((category)=>
                      <option key={category}>{category}</option>
                    )
                }
            </select>
            <table className="table table-hover">
                <thead>
                    <tr>
                        <th>Categories</th>
                    </tr>
                </thead>
                <tbody>
                    {
                        categories.map((category)=>
                          <tr key={category}>
                            <td>{category}</td>
                          </tr>
                        )
                    }
                </tbody>
            </table>
        </div>
     )
}
```

Ex: Array of Objects

data-binding.jsx

```
export function DataBinding()
{
    var data = [
       {Name: "Samsung TV", Price: 45000.44, Stock: true},
       {Name: "Nike Casuals", Price: 6000.55, Stock: true},
       {Name: "Watch", Price: 3400.33, Stock: false}
    ]
    return(
      <div className="container-fluid">
        <h2>Products Table</h2>
        <table className="table table-hover">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
                    <th>Stock</th>
                </tr>
            </thead>
```

```
                <tbody>
                    {
                        data.map((item)=>
                           <tr key={item.Name}>
                              <td>{item.Name}</td>
                              <td>{item.Price}</td>
                              <td>{(item.Stock==true)?"Available":"Out of Stock"}</td>
                              <td>
                                <button className="btn btn-danger">
                                    <span className="bi bi-trash-fill"></span>
                                </button>
                              </td>
                           </tr>
                        )
                    }
                </tbody>
            </table>
        </div>
     )
}
```

Ex: Nested Iterations

data-binding.jsx

```
export function DataBinding()
{
     var menu = [
          {Category: "Electronics", Products: ["Mobiles", "Televisions"]},
          {Category: "Footwear", Products: ["Casuals", "Boots"]}
     ]
     return(
        <div className="container-fluid">
            <h2>Menu</h2>
            <ol>
                {
                    menu.map((item)=>
                      <li key={item.Category}>
                         {item.Category}
                         <ul>
                            {
                                item.Products.map((product)=>
                                 <li key={product}>{product} <button className="btn btn-
warning"><span className="bi bi-pen-fill"></span></button> </li>
                                )
                            }
                    </ul>
                  </li>
                )
              }
            </ol>
            <hr />
            <select>
              {
                menu.map((item)=>
                  <optgroup label={item.Category}>
                    {
                      item.Products.map((product)=>
                        <option>{product}</option>
                      )
                    }
                  </optgroup>
                )
              }
            </select>
          </div>
        )
}
```

Note: React supports only "One-Way-Binding". There no "Two-Way-Binding".

Two Way Binding

Data Binding in React
- One Way Binding
  Binding Primitive Types : number, string, boolean, null, undefined..
  Binding Non Primitive Types : Array, Object, JSON

Note: Don't use variables for storing data in component.
    Variables are immutable.

State in React

- Web Applications use "http | https" as protocols.
- Http is a state less protocol.
- It can't remember information between requests.
- State less nature of application is good for server, as it manages memory.
- It is not good for contineous operations.
- You can handle this issue by implementing various state management techniques
    a) Application State
    b) Session State

        c) Cookies
        d) QueryString etc..

- React component can manage state for application
- State was available in React upto 17 version only with Class Components.
- React 18+ version introduced state for function components.
- React function component can handle state by using a Hook [method] called "useState()".
- useState is not for class component, it is only for function component.
- State is mutable.

Syntax:

```
    import   { useState }  from  "react";

    const [getter, setter] = useState();            => any type
    const [name]            = useState("John");    => string
    const [products]        = useState([]);        => Array
    const [product]        = useState({});        => Object
```

Ex:
data-binding.jsx

```
import  { useState } from "react";

export function DataBinding()
{
    const [userName]   = useState("John");
    const [categories] = useState(["Electronics", "Footwear", "Fashion"]);
    const [products] = useState([{Name:"TV", Price:45000.33}, {Name:"Nike Casuals",
Price:3200.44}]);

    return(
      <div className="container-fluid">
        <h2>State in React Function Component</h2>
        <p>Hello ! {userName}</p>
        <h2>Categories</h2>
        <ol>
          {
            categories.map((category)=>
             <li key={category}> {category} </li>
            )
          }
        </ol>
        <h2>Products Table</h2>
        <table className="table table-hover">
           <thead>
            <tr>
```

```
              <th>Name</th>
              <th>Price</th>
            </tr>
           </thead>
           <tbody>
              {
                products.map((product)=>
                 <tr key={product.Name}>
                   <td>{product.Name}</td>
                   <td>{product.Price}</td>
                 </tr>
                )
              }
           </tbody>
        </table>
      </div>
    )
}
```

Two Way Binding

- It is the process of accessign data from source and binding to UI and identifying the changes in UI and updating back to source.

- React is not supported with default two way binding.

- You have to manually manage by using Events.

Syntax:

```
        const [name, setname] = useState("john");

        <input type="text"  value={name}  onChange={handleNameChange}>

        function handleNameChange(event) {
            setname(event.target.value);
        }

        Hello ! {name}
```

Note: Two way binding uses only "onChange" as default event that identified the changes in UI.

Ex: Two Way Binding

```
import  { useState } from "react";
```

```
export function DataBinding()
{
    const [product, setProduct] = useState({Name:'', Price:0, City: '', Stock:false});

    function handleNameChange(e){
       setProduct({
           Name: e.target.value,
           Price: product.Price,
           City: product.City,
           Stock: product.Stock
       })
    }
    function handlePriceChange(e){
      setProduct({
        Name: product.Name,
        Price: e.target.value,
        City: product.City,
        Stock: product.Stock
    })
    }
    function handleCityChange(e){
      setProduct({
        Name: product.Name,
        Price: product.Price,
        City: e.target.value,
        Stock: product.Stock
    })
    }
    function handleStockChange(e){
      setProduct({
        Name: product.Name,
        Price: product.Price,
        City: product.City,
        Stock: e.target.checked
    })
    }


    return(
      <div className="container-fluid">
         <div className="row">
            <nav className="col-3">
                <h2>Register Product</h2>
                <dl>
                  <dt>Name</dt>
                  <dd><input type="text" onChange={handleNameChange}
className="form-control"/></dd>
                  <dt>Price</dt>
                  <dd><input type="number" onChange={handlePriceChange}
className="form-control"/></dd>
                  <dt>City</dt>
                  <dd>
                    <select className="form-select" onChange={handleCityChange}>
                       <option>Delhi</option>
                       <option>Hyd</option>
                    </select>
                  </dd>
                  <dt>Stock</dt>
                  <dd className="form-switch">
                    <input type="checkbox" onChange={handleStockChange}
className="form-check-input"/> Available
                  </dd>
                </dl>
            </nav>
            <main className="col-9">
              <h2>Details</h2>
              <dl>
                <dt>Name</dt>
                <dd>{product.Name}</dd>
                <dt>Price</dt>
                <dd>{product.Price}</dd>
                <dt>City</dt>
                <dd>{product.City}</dd>
                <dt>Stock</dt>
                <dd>{(product.Stock==true)?"Available":"Out of Stock"}</dd>
              </dl>
            </main>
         </div>
      </div>
    )
}
```

Connecting with API

Accessing Data from API
- Application Programming Interface [API]
- It refers to distributed computing architecture.
- Distributed computing architecture allows two different applications running or 2 different machines to share information.

(or)

Two different applications running on two different process of same machine can share information.

- There are various distributed computing architectures

    a) CORBA
    b) DCOM
    c) RMI
    d) EJB
    e) Web Services
    f) Remoting

- Service Specifications are 3 types

    a) SOAP
        Consumer => XML request
        Provider => XML response
    b) REST
        Consumer => Query request
        Provider => XML response [JSON]
    c) JSON
        Consumer => JSON request | Query
        Provider => JSON response

JavaScript Fetch Promise:

```
fetch("url").then(response=>binary=>json).then(use json).catch()
```

Ex:

```
import { useEffect, useState } from "react";

export function NasaComponent()
{
   const [mars, setMars] = useState({photos:[]});

   useEffect(()=>{
      fetch("https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY&quot;)
      .then(response=> response.json())
      .then(data=> {
         setMars(data);
      })
   },[])

   return(
      <div className="container-fluid">
         <h2>Mars Rover Photos  </h2>
         <table className="table table-hover">
```

```
                <thead>
                    <tr>
                        <th>Photo Id</th>
                        <th>Preview</th>
                        <th>Camera Name</th>
                        <th>Rover Name</th>
                    </tr>
                </thead>
                <tbody>
                    {
                        mars.photos.map(item=>
                             <tr key={item.id}>
                                 <td>{item.id}</td>
                                 <td>
                                    <a href={item.img_src} target="_blank"><img width="200" 
height="200" src={item.img_src} /></a>
                                 </td>
                                 <td>
                                    {item.camera.full_name}
                                 </td>
                                 <td>
                                    {item.rover.name}
                                 </td>
                             </tr>
                            )
                    }
                </tbody>
            </table>
        </div>
    )
}
```

React API - Fakestore

- useState
- useEffect
- fetch()

useEffect

- It is a react hook that provides following phases

    a) Mount
    b) Unmount

```
            useEffect(()=>{
                  // actions on mount
                 return() {
```

```
                //actions on unmount
            }
        }, [ dependencies ])
```

fetch()

- It is a promise of JavaScript.
- It is used to handle communication with API.

Syntax:

```
        fetch("url")
        .then(function(response){
           return response.json();
        })
        .then(function(data){
           ....present data..
        })
        .catch(function(ex){
           ...exception...
        })
```

Ex:
nasa.component.css

```
.card {
   width: 200px;
}
.card:nth-child(odd) {
   background-color: aquamarine;
}
.card:nth-child(even) {
   background-color: lightgray;
}
```

nasa.component.jsx

```
import { useEffect, useState } from "react";
import "./nasa.component.css";

export function NasaComponent()
{
   const [mars, setMars] = useState({photos:[]});

   useEffect(()=>{
      fetch("https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY&quot;)
```

```
        .then(response=> response.json())
        .then(data=> {
            setMars(data);
        })
    },[])

    return(
        <div className="container-fluid">
            <h2>Mars Rover Photos  </h2>
            <div className="d-flex flex-wrap">
                {
                    mars.photos.map((item)=>
                      <div key={item.id} className="card m-2 p-2">
                         <img src={item.img_src} height="200" className="card-img-top" />
                         <div className="card-body">
                            <dl>
                                <dt>Camera Name</dt>
                                <dd>{item.camera.full_name}</dd>
                                <dt>Rover Name</dt>
                                <dd>{item.rover.name}</dd>
                            </dl>
                         </div>
                      </div>
                    )
                }
            </div>
        </div>
    )
}
```

http://fakestoreapi.com

GET   http://fakestoreapi.com/products   [{ }, { }]
GET   http://fakestoreapi.com/products/1   { }
GET   http://fakestoreapi.com/products/category/electronics   [{ }] electronics
GET   http://fakestoreapi.com/products/categories   [" "]

```
                products.find((product)=> product.id==1)
                products.filter((product)=> product.category=="electronics")
```

Ex:
fakestore.component.css

```
.card-header {
    height: 130px;
}
main {
    height: 500px;
    overflow: auto;
}
```

fakestore.component.jsx

```
import { useState, useEffect } from "react";
import "./fakestore.component.css";

export function FakestoreComponent()
{
    const [categories, setCategories] = useState([]);
    const [products, setProducts] = useState([]);

    function LoadCategories(){
        fetch("http://fakestoreapi.com/products/categories&quot;)
        .then((response)=> response.json())
        .then((data)=> {
            data.unshift("all");
            setCategories(data);
        })
    }

    function LoadProducts(url){
        fetch(url)
        .then((response)=> response.json())
        .then((data)=>{
             setProducts(data);
        });
    }

    useEffect(()=>{
        LoadCategories();
        LoadProducts("http://fakestoreapi.com/products&quot;);
    },[]);

    function handleCategoryChange(event){
         if(event.target.value=="all"){
             LoadProducts("http://fakestoreapi.com/products&quot;);
         } else {
            LoadProducts(`http://fakestoreapi.com/products/category/${event.target.value
}`);
         }
    }
    return(
        <div className="container-fluid">
            <header className="d-flex justify-content-between p-2 bg-dark text-white mt-
2">
                <div><h2>Fakestore</h2></div>
                <div>
                    <span className="me-4">Home</span>
                    <span className="me-4">Electronics</span>
                    <span className="me-4">Jewelery</span>
                    <span className="me-4">Men's Fashion</span>
                    <span className="me-4">Women's Fashion</span>
                </div>
                <div>
                    <span className="bi bi-search me-3"></span>
                    <span className="bi bi-heart me-3"></span>
                    <span className="bi bi-person me-3"></span>
                    <span className="bi bi-cart me-3"></span>
                </div>
            </header>
            <section className="mt-4 row">
                <nav className="col-2">
                   <div>
                      <label className="form-label">Select Category</label>
                      <div>
                        <select onChange={handleCategoryChange} className="form-
select">
                            {
                                categories.map(category=>
                                     <option key={category} value={category}>
{category.toUpperCase()} </option>
                                    )
                            }
                        </select>
                      </div>
                   </div>
                </nav>
                <main className="col-10 d-flex flex-wrap">
                   {
                      products.map(product=>
                         <div key={product.id} className="card m-2 p-2">
                            <img src={product.image} height="150" className="card-img-top"
/>
                            <div className="card-header">
                                <p className="card-title">{product.title}</p>
                            </div>
                            <div className="card-body">
                                <dl>
                                    <dt>Price</dt>
                                    <dd>{product.price}</dd>
                                    <dt>Rating</dt>
                                    <dd> <span className="bi bi-star-fill text-success"></span>
{product.rating.rate} [{product.rating.count}]</dd>
                                </dl>
                            </div>
                            <div className="card-footer">
                                <button className="btn btn-danger w-100"> <span
className="bi bi-cart4"></span> Add to Cart</button>
                            </div>
                         </div>
                        )
                   }
                </main>
            </section>
        </div>
    )
}
```

React API with jQuery and Shopping Example

Fakestore Shopping API

Ex:
fakestore-component.css

```
.card-header {
    height: 100px;
}
main {
    height: 500px;
    overflow: auto;
}
```

fakestore.component.jsx

```
import { useState, useEffect } from "react";
import "./fakestore.component.css";

export function FakestoreComponent()
```

```
{
    const [categories, setCategories] = useState([]);
    const [products, setProducts] = useState([]);
    const [cartItems, setCartItems] = useState([]);
    const [cartCount, setCartCount] = useState(0);

    function GetCartCount(){
        setCartCount(cartItems.length);
    }

    function LoadCategories(){
        fetch("http://fakestoreapi.com/products/categories&quot;)
        .then((response)=> response.json())
        .then((data)=> {
            data.unshift("all");
            setCategories(data);
        })
    }

    function LoadProducts(url){
        fetch(url)
        .then((response)=> response.json())
        .then((data)=>{
             setProducts(data);
        });
    }

    useEffect(()=>{
        LoadCategories();
        LoadProducts("http://fakestoreapi.com/products&quot;);
        GetCartCount();
    },[]);

    function handleCategoryChange(event){
         if(event.target.value=="all"){
             LoadProducts("http://fakestoreapi.com/products&quot;);
         } else {
            LoadProducts(`http://fakestoreapi.com/products/category/${event.target.value
}`);
         }
    }

    function handleAddToCartClick(e){
        fetch(`http://fakestoreapi.com/products/${e.target.id}`)
        .then(response=> response.json())
        .then(data=>{
            cartItems.push(data);
            GetCartCount();
            alert(`${data.title}\nAdded to Cart`);
        })
    }

    function handleHomeClick(){
        LoadProducts("http://fakestoreapi.com/products&quot;);
    }

    return(
        <div className="container-fluid">
            <header className="d-flex justify-content-between p-2 bg-dark text-white mt-
2">
                <div><h2>Fakestore</h2></div>
                <div>
                    <span className="me-4"><button onClick={handleHomeClick}
className="btn text-white">Home</button></span>
                    <span className="me-4">Electronics</span>
                    <span className="me-4">Jewelery</span>
                    <span className="me-4">Men's Fashion</span>
                    <span className="me-4">Women's Fashion</span>
                </div>
                <div>
                    <span className="bi bi-search me-3"></span>
                    <span className="bi bi-heart me-3"></span>
                    <span className="bi bi-person me-3"></span>
                    <button data-bs-target="#cart" data-bs-toggle="modal" className="btn
btn-light position-relative">
                        <span className="bi bi-cart me-3"></span>
                        <span className="badge rounded-circle bg-danger position-
absolute"> {cartCount}</span>
                    </button>
                    <div className="modal fade" id="cart">
                        <div className="modal-dialog">
                            <div className="modal-content">
                                <div className="modal-header">
                                    <h2 className="text-primary">Your Cart Items</h2>
                                    <button data-bs-dismiss="modal" className="btn-
close"></button>
                                </div>
                                <div className="modal-body">
                                    <table className="table table-hover">
                                        <thead>
                                            <tr>
                                                <th>Title</th>
```

```
                                      <th>Preview</th>
                                      <th>Price</th>
                                  </tr>
                              </thead>
                              <tbody>
                                  {
                                      cartItems.map(item=>
                                          <tr>
                                              <td>{item.title}</td>
                                              <td><img src={item.image} width="50"
height="50"/></td>
                                              <td>{item.price}</td>
                                              <td>
                                                  <button className="btn btn-danger">
                                                      <span className="bi bi-trash-fill"></span>
                                                  </button>
                                              </td>
                                          </tr>
                                          )
                                  }
                              </tbody>
                          </table>
                      </div>
                  </div>
              </div>
          </div>
      </div>
  </header>
  <section className="mt-4 row">
      <nav className="col-2">
         <div>
            <label className="form-label">Select Category</label>
            <div>
              <select onChange={handleCategoryChange} className="form-
select">
                  {
                      categories.map(category=>
                           <option key={category} value={category}>
{category.toUpperCase()} </option>
                          )
                  }
              </select>
            </div>
         </div>
      </nav>
      <main className="col-10 d-flex flex-wrap">
```

```
                    {
                       products.map(product=>
                          <div key={product.id} className="card m-2 p-2">
                             <img src={product.image} height="150" className="card-img-top"
/>
                             <div className="card-header">
                                 <p className="card-title">{product.title}</p>
                             </div>
                             <div className="card-body">
                                 <dl>
                                     <dt>Price</dt>
                                     <dd>{product.price}</dd>
                                     <dt>Rating</dt>
                                     <dd> <span className="bi bi-star-fill text-success"></span>
{product.rating.rate} [{product.rating.count}]</dd>
                                 </dl>
                             </div>
                             <div className="card-footer">
                                 <button id={product.id} onClick={handleAddToCartClick}
className="btn btn-danger w-100"> <span className="bi bi-cart4"></span> Add to
Cart</button>
                             </div>
                          </div>
                         )
                    }
                </main>
            </section>
        </div>
    )
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { NetflixIndex } from './netflix/netflix-index';
import '../node_modules/bootstrap/dist/css/bootstrap.css';
import '../node_modules/bootstrap-icons/font/bootstrap-icons.css';
import "../node_modules/bootstrap/dist/js/bootstrap.bundle.js";

import { Login } from './components/login/login';
import { DataBinding } from './components/data-binding/data-binding';
import { NasaComponent } from './components/nasa-api/nasa.component';
import { FakestoreComponent } from './components/fakestore-
api/fakestore.component';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <FakestoreComponent />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

FAQ: What are the issues with Fetch promise?
Ans :
- It returns data in binary format.
- You have to parse the data into JSON format.
- Browser may block COM marshling [Common Object Model]
  [object - to - binary & vice versa]
- CORS issues [Cross Origin Resource Sharing]
- Fetch promise is using JavaScript "catch()" which is poor in error handling.

jQuery Ajax with React

- jQuery provides Ajax methods.
- They default return data in JSON
- No conversions required.
- Handle CORS issues
- It reduces the browser compatibility issues.
- It provides lot of Ajax life cycle methods, which can track the ajax process and report the errors.

```
ajaxStart()
ajaxStop()
ajaxSuccess()
ajaxError()
ajaxComplete()
```

Syntax:

```
$.ajax({
    method : "get | post | put | delete",
    url: "path",
    data: post data,
    success: () => { }
    error: () => { }
})
```

Ex:
1. Install jQuery library for project

```
>npm install jquery --save
```

2. Import jquery library into component

```
import  $  from  "jquery";
```

3. Implement Ajax

```
$.ajax({ })
```

fakestore.component.jsx

```
import { useState, useEffect } from "react";
import "./fakestore.component.css";
import $ from "jquery";

export function FakestoreComponent()
{
    const [categories, setCategories] = useState([]);
    const [products, setProducts] = useState([]);
    const [cartItems, setCartItems] = useState([]);
    const [cartCount, setCartCount] = useState(0);

    function GetCartCount(){
        setCartCount(cartItems.length);
    }

    function LoadCategories(){

        $.ajax({
            method: "get",
            url: "http://fakestoreapi.com/products/categories&quot;,
            success:(response) => {
                response.unshift("all");
                setCategories(response);
            },
            error: (response) => {
                console.log(response);
            }
        })

       /* fetch("http://fakestoreapi.com/products/categories&quot;)
```

```
        .then((response)=> response.json())
        .then((data)=> {
            data.unshift("all");
            setCategories(data);
        }) */
    }

    function LoadProducts(url){
        fetch(url)
        .then((response)=> response.json())
        .then((data)=>{
             setProducts(data);
        });
    }

    useEffect(()=>{
        LoadCategories();
        LoadProducts("http://fakestoreapi.com/products&quot;);
        GetCartCount();
    },[]);

    function handleCategoryChange(event){
         if(event.target.value=="all"){
             LoadProducts("http://fakestoreapi.com/products&quot;);
         } else {
            LoadProducts(`http://fakestoreapi.com/products/category/${event.target.value
}`);
         }
    }

    function handleAddToCartClick(e){
        fetch(`http://fakestoreapi.com/products/${e.target.id}`)
        .then(response=> response.json())
        .then(data=>{
            cartItems.push(data);
            GetCartCount();
            alert(`${data.title}\nAdded to Cart`);
        })
    }

    function handleHomeClick(){
        LoadProducts("http://fakestoreapi.com/products&quot;);
    }

    return(
        <div className="container-fluid">
```

```
         <header className="d-flex justify-content-between p-2 bg-dark text-white mt-
2">
            <div><h2>Fakestore</h2></div>
            <div>
               <span className="me-4"><button onClick={handleHomeClick}
className="btn text-white">Home</button></span>
               <span className="me-4">Electronics</span>
               <span className="me-4">Jewelery</span>
               <span className="me-4">Men's Fashion</span>
               <span className="me-4">Women's Fashion</span>
            </div>
            <div>
               <span className="bi bi-search me-3"></span>
               <span className="bi bi-heart me-3"></span>
               <span className="bi bi-person me-3"></span>
               <button data-bs-target="#cart" data-bs-toggle="modal" className="btn
btn-light position-relative">
                  <span className="bi bi-cart me-3"></span>
                  <span className="badge rounded-circle bg-danger position-
absolute"> {cartCount}</span>
               </button>
               <div className="modal fade" id="cart">
                  <div className="modal-dialog">
                     <div className="modal-content">
                        <div className="modal-header">
                           <h2 className="text-primary">Your Cart Items</h2>
                           <button data-bs-dismiss="modal" className="btn-
close"></button>
                        </div>
                        <div className="modal-body">
                           <table className="table table-hover">
                              <thead>
                                 <tr>
                                    <th>Title</th>
                                    <th>Preview</th>
                                    <th>Price</th>
                                 </tr>
                              </thead>
                              <tbody>
                                 {
                                    cartItems.map(item=>
                                       <tr>
                                          <td>{item.title}</td>
                                          <td><img src={item.image} width="50"
height="50"/></td>
                                          <td>{item.price}</td>
```

```
                                            <td>
                                                <button className="btn btn-danger">
                                                    <span className="bi bi-trash-fill"></span>
                                                </button>
                                            </td>
                                        </tr>
                                        )
                                }
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    </div>
</header>
<section className="mt-4 row">
    <nav className="col-2">
       <div>
          <label className="form-label">Select Category</label>
          <div>
            <select onChange={handleCategoryChange} className="form-
select">
                {
                    categories.map(category=>
                         <option key={category} value={category}>
{category.toUpperCase()} </option>
                        )
                }
            </select>
          </div>
       </div>
    </nav>
    <main className="col-10 d-flex flex-wrap">
       {
          products.map(product=>
             <div key={product.id} className="card m-2 p-2">
                <img src={product.image} height="150" className="card-img-top"
/>
                <div className="card-header">
                    <p className="card-title">{product.title}</p>
                </div>
                <div className="card-body">
                    <dl>
                        <dt>Price</dt>
                        <dd>{product.price}</dd>
```

```
                                 <dt>Rating</dt>
                                 <dd> <span className="bi bi-star-fill text-success"></span>
{product.rating.rate} [{product.rating.count}]</dd>
                              </dl>
                          </div>
                          <div className="card-footer">
                              <button id={product.id} onClick={handleAddToCartClick}
className="btn btn-danger w-100"> <span className="bi bi-cart4"></span> Add to
Cart</button>
                          </div>
                       </div>
                      )
                  }
               </main>
           </section>
       </div>
   )
}
```

React Axios and Style Binding

axios
- It returns data in JSON
- It good in error handling , more features than jQuery
- It is more secured.
   a) Prevent CORS attacks
   b) Prevent XSS attacks
   c) Cross Page Posting etc..
- It can handle multiple requests simultaneously at the same time.
- It is async, without blocking current request it can process another request.
- It is faster.

https://www.npmjs.com/

Syntax:

```
> npm  install  axios --save

import   axios  from "axios";

axios.get("url").then(()=>{on success}).catch(()=>{on failure})
```

Ex:
fakestore.component.jsx

```
import { useState, useEffect } from "react";
import "./fakestore.component.css";
import $ from "jquery";
import axios from "axios";

export function FakestoreComponent()
{
    const [categories, setCategories] = useState([]);
    const [products, setProducts] = useState([]);
    const [cartItems, setCartItems] = useState([]);
    const [cartCount, setCartCount] = useState(0);

    function GetCartCount(){
        setCartCount(cartItems.length);
    }

    function LoadCategories(){

        axios.get("http://fakestoreapi.com/products/categories&quot;)
        .then((response)=> {
            response.data.unshift("all");
            setCategories(response.data);
        })
        .catch((err)=> {
            console.log(err);
        })

      /*  $.ajax({
            method: "get",
            url: "http://fakestoreapi.com/products/categories&quot;,
            success:(response) => {
                response.unshift("all");
                setCategories(response);
            },
            error: (response) => {
                console.log(response);
            }
        }) */

       /* fetch("http://fakestoreapi.com/products/categories&quot;)
        .then((response)=> response.json())
        .then((data)=> {
            data.unshift("all");
            setCategories(data);
        }) */
    }

    function LoadProducts(url){
        fetch(url)
        .then((response)=> response.json())
        .then((data)=>{
             setProducts(data);
        });
    }

    useEffect(()=>{
        LoadCategories();
        LoadProducts("http://fakestoreapi.com/products&quot;);
        GetCartCount();
    },[]);

    function handleCategoryChange(event){
         if(event.target.value=="all"){
             LoadProducts("http://fakestoreapi.com/products&quot;);
         } else {
            LoadProducts(`http://fakestoreapi.com/products/category/${event.target.value}`);
         }
    }

    function handleAddToCartClick(e){
        fetch(`http://fakestoreapi.com/products/${e.target.id}`)
        .then(response=> response.json())
        .then(data=>{
            cartItems.push(data);
            GetCartCount();
            alert(`${data.title}\nAdded to Cart`);
        })
    }

    function handleHomeClick(){
        LoadProducts("http://fakestoreapi.com/products&quot;);
    }

    return(
        <div className="container-fluid">
            <header className="d-flex justify-content-between p-2 bg-dark text-white mt-2">
                <div><h2>Fakestore</h2></div>
                <div>
                    <span className="me-4"><button onClick={handleHomeClick} className="btn
text-white">Home</button></span>
                    <span className="me-4">Electronics</span>
                    <span className="me-4">Jewelery</span>
                    <span className="me-4">Men's Fashion</span>
                    <span className="me-4">Women's Fashion</span>
                </div>
                <div>
                    <span className="bi bi-search me-3"></span>
                    <span className="bi bi-heart me-3"></span>
                    <span className="bi bi-person me-3"></span>
                    <button data-bs-target="#cart" data-bs-toggle="modal" className="btn btn-light
position-relative">
                    <span className="bi bi-cart me-3"></span>
                    <span className="badge rounded-circle bg-danger position-absolute">
{cartCount}</span>
                    </button>
                    <div className="modal fade" id="cart">
                    <div className="modal-dialog">
                        <div className="modal-content">
                            <div className="modal-header">
                                <h2 className="text-primary">Your Cart Items</h2>
                                <button data-bs-dismiss="modal" className="btn-close"></button>
                            </div>
                            <div className="modal-body">
                                <table className="table table-hover">
                                    <thead>
                                        <tr>
                                            <th>Title</th>
                                            <th>Preview</th>
                                            <th>Price</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        {
                                            cartItems.map(item=>
                                                <tr>
                                                    <td>{item.title}</td>
                                                    <td><img src={item.image} width="50"
height="50"/></td>
                                                    <td>{item.price}</td>
                                                    <td>
                                                        <button className="btn btn-danger">
                                                            <span className="bi bi-trash-fill"></span>
                                                        </button>
                                                    </td>
                                                </tr>
                                                )
                                        }
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                    </div>
                </div>
            </header>
            <section className="mt-4 row">
                <nav className="col-2">
                   <div>
                      <label className="form-label">Select Category</label>
                      <div>
                        <select onChange={handleCategoryChange} className="form-select">
                            {
                                categories.map(category=>
                                    <option key={category} value={category}> {category.toUpperCase()}
</option>
                                 )
                            }
                        </select>
                      </div>
                   </div>
                </nav>
                <main className="col-10 d-flex flex-wrap">
                   {
                      products.map(product=>
                         <div key={product.id} className="card m-2 p-2">
                            <img src={product.image} height="150" className="card-img-top" />
                            <div className="card-header">
                                <p className="card-title">{product.title}</p>
                            </div>
                            <div className="card-body">
                                <dl>
                                    <dt>Price</dt>
                                    <dd>{product.price}</dd>
                                    <dt>Rating</dt>
                                    <dd> <span className="bi bi-star-fill text-success"></span>
{product.rating.rate} [{product.rating.count}]</dd>
                                </dl>
                            </div>
                            <div className="card-footer">
                                <button id={product.id} onClick={handleAddToCartClick}
className="btn btn-danger w-100"> <span className="bi bi-cart4"></span> Add to
Cart</button>
                            </div>
                         </div>
                        )
                   }
                </main>
            </section>
        </div>
    )
```

}

FAQ: Which JavaScript object is used by Browser to handle AJAX calls?
Ans: XMLHttpRequest object

Component
DataBinding
a) One Way
b) Two Way
Connecting with API
a) fetch()
b) jquery Ajax
c) axios

Style and ClassBinding

- React will not allow to bind inline styles exactly as you defined in DOM.
- React uses virtual DOM and the actual DOM syntax is not allowed.

Syntax:
```
        <element  style={ {styleAttribute:'value', styleAttribute:'value' } }>
```

Ex:
```
       <h1 style={{color:'red', fontFamily: 'cursive', fontStyle:'italic'}}>Welcome</h1>
```

- Style binding allows to change the element appearence dynamicaly.

Ex:
```
import { useState } from "react";

export function StyleBinding(){

    const [userName, setUserName] = useState('');
    const [styleObject, setStyleObject] = useState({'border':'','boxShadow':''});

    function VerifyName(e){
        setUserName(e.target.value);
        if(userName=="") {
            setStyleObject({
                border: '2px solid red',
                boxShadow: '2px 2px 2px red'
            })
        } else {
            setStyleObject({
                border: '2px solid green',
                boxShadow: '2px 2px 2px green'
            })
        }
    }
    return(
        <div className="container-fluid">
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" style={styleObject} onBlur={VerifyName} /></dd>
                <dd></dd>
            </dl>
        </div>
    )
}
```

Class and Style Binding

Data Binding
Style Binding
        style={ {attributeName:value, attributeName:value} }

Ex:
style.binding.jsx

```
import { useState } from "react";

export function StyleBinding(){

    const [styles, setStyles] = useState({'fontSize':'10px', 'color':'black'})

    function handleFontSizeChange(e){
        setStyles({
            fontSize: e.target.value + "px",
            color: styles.color
        })
    }
    function handleColorChange(e){
        setStyles({
            fontSize: styles.fontSize,
            color: e.target.value
        })
    }
    return(
        <div className="container-fluid">
            <h2>Style Binding</h2>
            <dl>
                <dt>Font Size</dt>
                <dd><input type="range" onChange={handleFontSizeChange}
min="10"  max="100"/></dd>
```

```
                <dt>Font Color</dt>
                <dd><input type="color" onChange={handleColorChange}/></dd>
            </dl>
            <p className="text-center" style={styles}>Sample Text</p>
        </div>
    )
}
```

Class Binding in React

- You can define a CSS class dynamically to any element.

Syntax:

```
        <div  className={ referenceName }> </div>

        referenceName = "string";
```

Ex:
class.demo.jsx

```
import { useState } from "react";

export function ClassDemoComponent(){
    const [cities] = useState(["Delhi","Hyd","Mumbai","Chennai","Bangalore"]);
    const [buttonClass, setButtonClass] = useState('bi bi-sort-alpha-down');

    function handleSortClick(e){
        if(e.target.className=="bi bi-sort-alpha-down") {
            setButtonClass('bi bi-sort-alpha-up');
            cities.sort();
        } else {
            setButtonClass('bi bi-sort-alpha-down');
            cities.sort();
            cities.reverse();
        }
    }

    return(
        <div className="container-fluid mb-3">
            <h2 className="mt-3">Cities List <button onClick={handleSortClick}
className="btn btn-success"> <span className={buttonClass}></span> </button> </h2>
            <ol>
                {
                    cities.map(city=>
                        <li key={city}>{city}</li>
                        )
                }
            </ol>
        </div>
    )
}
```

Ex: Theme

```
import { useState } from "react";

export function ClassBindingComponent()
{
    const [theme, setTheme] = useState('');
    const [buttonTheme, setButtonTheme] = useState('btn btn-dark w-100');

    function handleThemeChange(e){
         if(e.target.checked) {
            setTheme('bg-dark text-white p-3');
            setButtonTheme('btn btn-light w-100');
         } else {
            setTheme('');
            setButtonTheme('btn btn-dark w-100');
         }
    }

    function handleNameChange(e){
         if(e.target.value=="") {
            setTheme('bg-danger text-white p-2');
         } else {
            setTheme('bg-success text-white p-2');
         }
    }

    return(
        <div className="container-fluid">
            <div className="d-flex justify-content-center align-items-center"
style={{height:'500px'}}>
                <form className={theme}>
                    <div className="mt-4 mb-4 form-switch">
                        <input type="checkbox" onChange={handleThemeChange}
className="form-check-input"/> Dark Mode
                    </div>
                    <h2><span className="bi bi-person-fill"></span> User Login</h2>
                    <dl>
                        <dt>User Name</dt>
                        <dd><input type="text" onChange={handleNameChange} className="form-
control"/></dd>
                        <dt>Password</dt>
                        <dd><input type="password" className="form-control" /></dd>
                    </dl>
                    <button className={buttonTheme}>Login</button>
                </form>
            </div>
        </div>
    )
}
```

EVENT BINDING

1. What is Event?
A. Event is a message sent by sender to its subscriber in order to notify change.

2. Which software design pattern Event uses?
A. Event follows a software design pattern called "Observer".

3. What is the mechanism of event?
A. Event uses "Delegate" mechanism, which is function pointer.

4. All React Events a of type "Synthetic Events".

Browser Event	SyntheticEvent
onclick	onClick

Syntax:

```
function InsertClick()              => Subscriber
{
}

<button  onClick={InsertClick}>       => Sender
```

- Sender sends notification to subscriber.
- Subscriber comprises of actions to perform when notified.
- React Events are derived from "SyntheticEvents" base.

Note: Events are related to "Browser" object. React library can't use browser events directly. React Events are configured in a class called "SynthenticEvents", which maps to Browser Events.

React library => SynthenticEvent => onClick => BrowserEvent => onclick

- Every event have 2 parts

```
onClick={handleClick}
```

a) onClick   Event
b) onClick={handleClick}   Event Handler

- Event Event handler have one default argument in React.
    a) event

- JavaScript Event handler have 2 default arguments
    a) this   : sends information about current object
    b) event   : sends information about current event

```
<button  onclick="InsertClick(this, event)">

function InsertClick(obj, e) {
        obj.id, name, class, value, src, alt, etc..
        e.clientX, clientY, shiftKey, ctrlKey, altKey etc..
}
```

- React SyntheticEvent can access both object and event related details by using single "event" argument.

```
<button  onClick={InsertClick}>

 function InsertClick(event) {
        event.EventDetails  => clientX, clientY, keyCode, which, shiftKey..
        event.target.objectDetails => id, name, class etc..
 }
```

Ex:

```
export function EventBinding()
{
    function InsertClick(e){
        document.write(`
            Button Id          : ${e.target.id} <br>
            Button Name    : ${e.target.name} <br>
            Button Value    : ${e.target.value} <br>
            X Position        : ${e.clientX} <br>
            Ctrl Key           : ${e.ctrlKey}
        `);
    }
    return(
        <div className="container-fluid">
            <h2>Events</h2>
            <button id="btnInsert" onClick={InsertClick} name="InsertButton"
value="Insert">Insert</button>
        </div>
    )
}
```

Ex

```
export function EventBinding()
{
    function InsertClick(e){
         for(var property in e.target){
            document.write(`${property} : ${typeof e[property]}<br>`);
         }
    }
    return(
        <div className="container-fluid">
            <h2>Events</h2>
            <button id="btnInsert" onClick={InsertClick} name="InsertButton"
value="Insert">Insert</button>
        </div>
    )
}
```

Note: You can assign event handler or you can call the event function.

```
        <button  onClick={InsertClick}>        Assignment of function
        <button onClick={()=>InsertClick()}>    Calling a function
```

Function calling allows to discard the default arguments and pass custom arguments.

Ex:

```
export function EventBinding()
{
    function InsertClick(msg, product){
         document.write(`
             ${msg} <br>
             ${product.Name} <br>
         `);

    }
    return(
        <div className="container-fluid">
            <h2>Events</h2>
            <button id="btnInsert" onClick={()=>{InsertClick('Welcome',{'Name':'TV'})}}
name="InsertButton" value="Insert">Insert</button>
        </div>
    )
}
```

Ex: You can use "Rest parameters"

```
export function EventBinding()
{
    function InsertClick(...args){
        var [msg, product] = args;
        document.write(`
            ${msg} <br>
            ${product.Name}
        `);
    }
    return(
        <div className="container-fluid">
            <h2>Events</h2>
            <button id="btnInsert" onClick={()=>{InsertClick('Welcome',{'Name':'TV'})}}
name="InsertButton" value="Insert">Insert</button>
        </div>
    )
}
```

Ex: Spread Syntax

```
export function EventBinding()
{
    function InsertClick(msg, product){
       document.write(`
          ${msg} <br>
          ${product.Name}
       `);
    }
    return(
        <div className="container-fluid">
            <h2>Events</h2>
            <button id="btnInsert" onClick={()=>{InsertClick(...['Welcome', {'Name':'Samsung'}])}}
name="InsertButton" value="Insert">Insert</button>
        </div>
    )
}
```

React Events

What is Event
What is Event Handler
What are Event Arguments
Defalut Arguments
Custom Arguments
Event Assignment
Event Calling
SyntheticEvent Base

Function with Rest Parameters
Function using spread operator.

Various Types of Events
1. Mouse Events
    - onMouseover
    - onMouseout
    - onMousedown
    - onMouseup
    - onMousemove
2. Keyboard Events
    - onKeyup
    - onKeydown
    - onKeypress
3. Button Events
    - onClick
    - onDblClick
    - onContextMenu
4. Element State Events
    - onChange
    - onBlur
    - onFocus
    - onSelectstart
5. Clipboard Events
    - onCut
    - onCopy
    - onPaste
6. FormEvents
    - onSubmit
    - onReset
7. TouchEvents
    - onTouchStart
    - onTouchEnd
    - onTouchMove
8. Timer Events
    - setInterval()
    - clearInterval()
    - setTimeout()
    - clearTimeout()

MouseEvents

- onMouseOver
- onMouseOut
- onMouseUp
- onMouseDown
- onMouseMove

Ex:
 mouse-demo.css

```
img:hover{
    cursor: grab;
}
.preview-image:hover {
    transform: scale(1.5);
    transition: 2s;
}
.preview-image {
    transition: 2s;
}
```

mouse-demo.jsx

```
import { useState } from "react";
import "./mouse-demo.css";

export function MouseDemo(){

    const [photos] =
useState(["images/m1.jpg","images/m2.jpg","images/m3.jpg","images/m4.jpg","images/m5.jpg"]);
    const [preview, setPreview] = useState("");

    function HandleMouseOver(e){
        setPreview(e.target.src);
    }

    return(
        <div className="container-fluid">
            <section className="row mt-2">
                <nav className="col-2">
                    {
                        photos.map(photo=>
                            <div className="mb-2 p-1 border border-2 border-primary"
style={{width:'60px'}} key={photo}>
                                <img src={photo}  onMouseOver={HandleMouseOver} width="50"
height="50" />
                            </div>
                            )
                    }
                </nav>
                <main className="col-10">
                    <img width="300" className="preview-image" height="300" src={preview} />
                </main>
            </section>
        </div>
    )
}
```

Ex: over and out

mouse-demo.css

```
marquee {
    padding: 50px;
}
img:hover{
    transform: scale(1.5);
    box-shadow: 4px 4px 3px black;
    transition: 2s;
}
img {
    transition: 2s;
}
```

mouse-demo.jsx

```
import { useState } from "react";
import "./mouse-demo.css";

export function MouseDemo(){

    const [photos] =
useState(["images/m1.jpg","images/m2.jpg","images/m3.jpg","images/m4.jpg","images/m5.jpg"]);

    function handleMouseOver(e){
        e.currentTarget.stop();
    }
    function handleMouseOut(e){
        e.currentTarget.start();
    }
    return(
        <div className="container-fluid">
           <div className="mt-4">
              <marquee scrollamount="10" onMouseOver={handleMouseOver}
onMouseOut={handleMouseOut} >
                {
                    photos.map(photo=>
                         <img width="100" key={photo} className="me-2" height="100"  src={photo}/>
                        )
                }
              </marquee>
           </div>
        </div>
    )
}
```

EX: mouse move

mouse-demo.jsx

```
import { useState } from "react";
import "./mouse-demo.css";

export function MouseDemo(){

    const [styleObject, setStyleObject] = useState({'position':'','top':'', 'left':''});

    function handleMouseMove(e) {
        setStyleObject({
            'position': 'fixed',
            'left': e.clientX + 'px',
            'top': e.clientY + 'px'
        })
    }

    return(
        <div onMouseMove={handleMouseMove}>
            <div style={{height:'1000px'}}>
                <p>move your mouse pointer to test.
                    <br/>
                    X = {styleObject.left}  Y = {styleObject.top}
                </p>
            </div>
            <img src="images/flag.gif" style={styleObject} width="50" height="50"/>
        </div>
    )
}
```

Touch Events

- onTouchStart
- onTouchEnd
- onTouchMove

EX:

```
import { useState } from "react";
import "./mouse-demo.css";

export function MouseDemo(){

    const [styleObj, setStyleObj] = useState({'position':'', 'left':'', 'top':''})
    function handleTouch(e){
        setStyleObj({
            'position': 'fixed',
            'left': e.touches[0].clientX + 'px',
            'top': e.touches[0].clientY + 'px'
        })
    }

    return(
        <div className="container-fluid">
            <img src="images/m1.jpg" style={styleObj} onTouchMove={handleTouch} width="100"
height="100" />
        </div>
    )
}
```

Events Examples

Event Binding
- Mouse Events
- Touch Events

Keyboard Events

onKeyUp	]	good for handling the chars that you key-in
onKeyDown	]	
onKeyPress	]	good for handling the code of character.

Key Event Properties:

keyCode
charCode
which
shiftKey
ctrlKey
altKey

Ex:
keydemo.jsx

```
import { useState } from "react";

export function KeyDemo()
{
    const [users] = useState([{"UserName":"john"}, {"UserName":"john12"},
{"UserName":"david"}]);
    const [userError, setUserError] = useState('');
    const [errorClass, setErrorClass] = useState('');
    const [passwordWarning, setPasswarning] = useState({'display':'none'});

    function VerifyUserName(e){
        for(var user of users){
            if(user.UserName==e.target.value) {
                setUserError('User Name Taken - Try Another');
                setErrorClass('text-danger');
                break;
            } else {
                setUserError('User Name Available');
                setErrorClass('text-success');
            }
        }
    }

    function VerifyPassword(e){
        console.log(`
             Key Code : ${e.keyCode} \n
             Char Code : ${e.charCode} \n
             Which  : ${e.which}
        `);
        if(e.which>=65 && e.which<=90) {
            setPasswarning({'display':'block'});
        } else {
            setPasswarning({'display':'none'});
        }
    }

    return(
        <div className="container-fluid">
            <h3>Register User</h3>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" onKeyUp={VerifyUserName} /></dd>
                <dd className={errorClass}>{userError}</dd>
                <dt>Password</dt>
                <dd><input type="password" onKeyPress={VerifyPassword} /></dd>
                <dd className="text-warning" style={passwordWarning}>
                    <span className="bi bi-exclamation-triangle"></span> Warning : Caps ON
                </dd>
            </dl>
        </div>
    )
```

}

Assignment: Design Password that shows password strength while typing password.

Condition => Password must be 4 to 15 chars alpha numeric with atleast 1 uppercase letter.

Password length < 4 => status "Poor Password"
Password between 4 to 15 => status "Weak Password"
[without uppercase]

Password between 4 to 15 => status "Strong Password"
with uppercase letter

Element State Events

onBlur
onFocus
onChange
onSelectStart

Ex:

```
import { useState } from "react";

export function KeyDemo()
{
    const [userName, setUsername] = useState('john');
    const [tip, setTip] = useState('');

    function ChangeCase(){
        setUsername(userName.toUpperCase());
        setTip('');
    }
    function handleUserChange(e){
        setUsername(e.target.value)
    }
    function ShowTip(){
        setTip('Name in Block Letter Only');
    }

    return(
        <div className="container-fluid">
            <h3>Register User</h3>
            <dl>
               <dt>User Name</dt>
               <dd><input type="text" onFocus={ShowTip} onChange={handleUserChange}
value={userName} onBlur={ChangeCase} placeholder="Name in block letters" /></dd>
```

```
              <dd>{tip}</dd>
           </dl>
        </div>
    )
}
```

Ex:
keydemo.jsx

```
import { useState } from "react";

export function KeyDemo()
{
    const [userName, setUsername] = useState('john');
    const [tip, setTip] = useState('');

    function ChangeCase(){
        setUsername(userName.toUpperCase());
        if(userName==""){
            setTip('User Name Required');
        } else {
            setTip('');
        }
    }
    function handleUserChange(e){
        setUsername(e.target.value)
    }
    function ShowTip(){
        setTip('Name in Block Letter Only');
    }

    return(
        <div className="container-fluid">
            <h3>Register User</h3>
            <dl>
               <dt>User Name</dt>
               <dd><input type="text" onFocus={ShowTip} onChange={handleUserChange}
value={userName} onBlur={ChangeCase} placeholder="Name in block letters" /></dd>
               <dd>{tip}</dd>
            </dl>
        </div>
    )
}
```

Ex: Element State Events
keydemo.jspx

```
import { useState } from "react";

export function KeyDemo()
{
    const [country, setCountry] = useState('');
    const [tip, setTip] = useState('');
    const [mobile, setMobile] = useState('');
    const [regExp, setRegExp] = useState(/ /);
    const [mobileError, setMobileError] = useState('');

    function handleCountryChange(e){
        setCountry(e.target.value);
        switch(e.target.value){
            case "India":
            setTip('+91 10 digits number');
            setRegExp(/\+91\d{10}/);

            break;
            case "US":
            setTip('+(1)(20)(460) 345 3210');
            setRegExp(/\+\(1\)\(\d{2}\)\(\d{3}\)\s\d{3}\d{4}/);

            break;
            case "UK":
            setTip('+(44)(230) 4670 3450');
            setRegExp(/\+\(44\)\(\d{3}\)\s\d{4}\s\d{4}/);

            break;
        }
    }

    function handleMobileChange(e){
        setMobile(e.target.value);
    }

    function handleSubmitClick(){
        if(mobile.match(regExp)){
            document.write("<h2>Verified Successfully..</h2>");
        } else {
            setMobileError(`Invalid Mobile - ${tip}`);
        }
    }

    return(
        <div className="container-fluid">
            <h3>Register User</h3>
            <dl>
              <dt>Your Country</dt>
              <dd>
               <select onChange={handleCountryChange}>
                   <option>Select Country</option>
                   <option>India</option>
                   <option>US</option>
                   <option>UK</option>
               </select>
              </dd>
              <dt>Mobile</dt>
              <dd>
               <input type="text" onChange={handleMobileChange} placeholder={tip} />
              </dd>
              <dd className="text-danger">{mobileError}</dd>
              <dd>
               <button onClick={handleSubmitClick} className="btn btn-
primary">Submit</button>
              </dd>
            </dl>
        </div>
    )
}
```

Timer Events

- Mouse Events
- Keyboard Events
- Element State Events

Button Events

onClick	Single Click of left button
onDoubleClick	Double Click of left button (ondblclick)
onContextMenu	Right Click of mouse button

Ex:

```
export function ButtonDemo()
{
    function OpenWindow(){
        window.open("images/m1.jpg","Mobile","width=400 height=400");
    }
    return(
        <div className="container-fluid">
            <h2>Double Click</h2>
            <img onDoubleClick={OpenWindow} src="images/m1.jpg"  width="100"
height="100"/>
        </div>
    )
}
```

Clipboard Events

onCut
onCopy
onPaste

Ex:

```
import { useState } from "react";


export function ButtonDemo()
{
    const [msg, setMsg] = useState('');

    function OpenWindow(){
        window.open("images/m1.jpg","Mobile","width=400 height=400");
    }
    function handleCut(){
        setMsg('Removed - Copied to clipboard');
    }
    function handleCopy(){
        setMsg('Copied to Clipboard');
    }
    function handlePaste(){
        setMsg('Inserted from Clipboard');
    }
    return(
        <div className="container-fluid">
            <h2>Double Click</h2>
            <img onDoubleClick={OpenWindow} src="images/m1.jpg"  width="100"
height="100"/>
             <br /> <br/>
            <textarea rows="4" cols="40" onCut={handleCut} onCopy={handleCopy}
onPaste={handlePaste}>

            </textarea>
            <p>{msg}</p>
        </div>
    )
}
```

Timer Events

setInterval()	It can execute specified function at regular time intervals
clearInterval()	It can clear the function form memory, which is set using timer.
setTimeout()	It delays the specified function by given time interval.
clearTimeout()	It clears from memory.

Ex:

```
import { useEffect, useState } from "react";

export function ButtonDemo()
{
    const [msg, setMsg] = useState('');

    function Msg1(){
        setMsg("Hello !");
    }
    function Msg2(){
        setMsg("How are you?");
    }
    function Msg3(){
        setMsg("Welcome to React");
    }
    useEffect(()=>{
        setTimeout(Msg1,3000);
        setTimeout(Msg2,5000);
        setTimeout(Msg3,10000);
    },[]);

    return(
        <div className="container-fluid">
            <h1 className="text-center text-primary">{msg}</h1>
        </div>
    )
}
```

Ex:

```
import { useEffect, useState } from "react";

export function ButtonDemo()
{
    const [msg, setMsg] = useState('');

    function ShowTime(){
        var now = new Date();
        setMsg(now.toLocaleTimeString());
    }

    useEffect(()=>{
       setInterval(ShowTime, 1000);
    },[]);

    return(
        <div className="container-fluid">
            <h1 className="text-center text-primary">{msg}</h1>
        </div>
    )
}
```

Ex:

```
import { useEffect, useState } from "react";

export function ButtonDemo()
{
    const [images, setImages] = useState(["images/m1.jpg",
"images/m2.jpg","images/m3.jpg","images/m4.jpg","images/m5.jpg"]);
    const [image, setImage] = useState('');

    var count = 0;
    function LoadImages(){

        count++;
        if(count==images.length) {
            count = 0;
        } else {
            console.log(images[count]);
            setImage(images[count]);
        }
    }

    useEffect(()=>{
        setInterval(LoadImages,1000);
    },[]);

    return(
        <div className="container-fluid">
            <h2>Realme Mobile</h2>
            <img  width="200" height="300" src={image}/>
        </div>
    )
```

}

Ex: Spinner

```
import { useEffect, useState } from "react";

export function ButtonDemo()
{
    const [status, setStatus] = useState(1);
    const [textContainer, setTextContainer] = useState({display:'none'});
    const [imgContainer, setImgContainer] = useState({display:'none'});


    var count = 1;
    function Loading(){
        if(count==100){
            setImgContainer({
                display:'block'
            });
            setTextContainer({
                display: 'none'
            })
        } else {
            count++;
            setStatus(count);
        }
    }

    function handleLoadClick(){
        setTextContainer({
            display: 'block'
        })
        setInterval(Loading, 200);
    }

    useEffect(()=>{

    },[]);

    return(
        <div className="container-fluid">
           <button className="btn mt-2 btn-primary" onClick={handleLoadClick}>Load
Image</button>
           <div className="d-flex justify-content-center align-items-center"
style={{height:'500px'}}>
                <div className="text-center" style={textContainer}>
                    <div className="spinner-border"></div>
                    <p>{status} %</p>
                    <p>Loading</p>
                </div>
                <div style={imgContainer}>
                    <img src="images/m1.jpg" width="300" height="300"/>
                </div>
           </div>
        </div>
    )
}
```

Ex: Progress

```
import { useEffect, useState } from "react";

export function ButtonDemo()
{
   const [status, setStatus] = useState(1);
   const [textContainer, setTextContainer] = useState({display:'none'});
   const [imgContainer, setImgContainer] = useState({display:'none'});
   const [progressValue, setProgressValue] = useState(1);


   var count = 1;
   function Loading(){
      if(count==100){
         setImgContainer({
            display:'block'
         });
         setTextContainer({
            display: 'none'
         })
      } else {
         count++;
         setProgressValue(count);
         setStatus(count);
      }
   }

   function handleLoadClick(){
      setTextContainer({
         display: 'block'
      })
      setInterval(Loading, 200);
   }
```

```
    useEffect(()=>{

    },[]);

    return(
        <div className="container-fluid">
          <button className="btn mt-2 btn-primary" onClick={handleLoadClick}>Load
Image</button>
          <div className="d-flex justify-content-center align-items-center"
style={{height:'500px'}}>
              <div className="text-center" style={textContainer}>
                  <progress min="1" max="100" value={progressValue}></progress>
                  <p>{status} %</p>
                  <p>Loading</p>
              </div>
              <div style={imgContainer}>
                  <img src="images/m1.jpg" width="300" height="300"/>
              </div>
          </div>
        </div>
    )
}
```

Component Properties

Component Properties

- Components a building blocks for application.
- Component comprises of presentation, styles and logic.
- You can re-use any component.
- A component can be customized according to requirements and can be re-used at various locations.
- Component can be modified or customized by using Properties.
- Properities are a set of parameters which can modify a function.
- In React function component the parameter is considered as properties, which is an object type.

```
        export function Name(props)            => props = { Key:'value', Key:'value' }
        {
            props.Key
        }

        <Name  key="value">
```

- Note: Props keys are dynamic type.

Ex:
1. Add a new folder
       "customized-components"

2. Add a new file
        "login-component.jsx"

```
export function LoginComponent(props){
    return(
        <div className="container-fluid">
            <form className="w-25 border border-2 border-primary p-2">
                <h3><span className="bi bi-person-fill"></span> {props.title} </h3>
                <dl>
                    <dt>{props.login_id}</dt>
                    <dd><input type={props.login_type} className="form-control" /></dd>
                    <dt>Password</dt>
                    <dd><input type="password" className="form-control"/></dd>
                </dl>
                <button className="btn btn-primary w-100">
                     {props.button_text}
                </button>
            </form>
        </div>
    )
}
```

3. Add a new HomeComponent

home-component.jsx

```
import { LoginComponent } from "../../customized-components/login/login-component"
export function HomeComponent(){
    return(
        <div className="container-fluid">

            <h2>Admin Portal</h2>
            <LoginComponent title="Admin Login" login_id="Your Email" login_type="email"
button_text="Admin Login" />

            <h2>Customer Portal</h2>
            <LoginComponent title="Customer Login" login_id="Your Mobile" login_type="text"
button_text="Customer Signin" />

        </div>
    )
}
```

Ex:
grid-component.jsx

```
export function GridComponent(props){
    return(
        <div className="container-fluid">
            <table className="table table-hover table-dark caption-top">
                <caption>{props.caption}</caption>
                <thead>
                    <tr>
                        {
                            props.fields.map((field)=>
                              <th key={field}>{field}</th>
                            )
                        }
                    </tr>
                </thead>
                <tbody>
                    {
                        props.data.map(item=>
                             <tr key={item}>
                                {
                                    Object.keys(item).map(key=>
                                        <td key={item[key]}>{item[key]}</td>
                                        )
                                }
                             </tr>
                            )
                    }
                </tbody>
            </table>
        </div>
    )
}
```

home-component.jsx

```
import { useState } from "react"
import { GridComponent } from "../../customized-components/grid/grid-component"

export function HomeComponent(){
    const [products] = useState([{Name:"TV", Price:66000.33},{Name:"Mobile", 
Price:42000.33}]);
```

```
    return(
        <div className="container-fluid">
             <GridComponent caption="Employee Table" fields={["First Name", "Last Name", 
"Salary"]} data={[{'FirstName':'Raj', 'LastName':'Kumar', 
'Salary':56000.44},{'FirstName':'Tom', 'LastName':'Hanks','Salary':67000.44}]} />
             <hr />
             <GridComponent caption="Products List" fields={["Name","Price"]} data={products} 
/>
        </div>
    )
}
```

Classes in JavaScript

JavaScript Classes

1. What is a class ?
A. Class is a program template.
A program template comprises of data and logic, which you can implement and customize according to requirements.

2. Class Behaviours
- If a class is mapping to business requrements then it is known as "Entity".
- If a class is mapping to data requirements then it is known as "Model".
- A class is used as a blue print for creating various instances.

3. JavaScript class can be configured in 2 ways

a) Class Expression
b) Class Declaration

```
    const   Employee = class {                    => class expression

    }

    class Employee                                => class declaration
    {
    }

    const  HomeComponent = function(){ }

    function HomeComponent() { }
```

Ex:
```
<script>
```

```
    var action;
    var password = prompt("Enter Password");
    if(password=="admin") {
      action = function(){
         document.write("Login Succes..");
      }
    } else {
      action = function(){
         document.write("Invalid Password");
      }
    }
    action();
</script>
```

- Class expression allocate memory for class, which can change the set of members according state and situation.

- Class Declaration allows only specific set of members in allocated memory.

4. Class Members
- A JavaScript class member can be

  a) Property
  b) Accessor
  c) Method
  d) Constructor

FAQ: Can we define variable as class member?
Ans: No. You can define a variable in class but not as class member.
It is not possible, as variables are immutable types, and class can have only mutable types.

FAQ: can we define function as class member?
Ans: No.

Property

- Data is stored in property.
- JavaScript property can handle any type of data, primitive or non-primitive.
- Properties are not strongly typed.
- Properties don't have any access modifiers.

Syntax:

```
            class  className
            {

              propertyName = value;
```

```
        }
```

- The properties of a class are accessed by using instance of class.

```
        var instance  = new className();
        instance.propertyName;
```

Ex:

```
<script>
    class Product
    {
       Name = "Samsung TV";
       Price = 40000.44;
       Stock = true;
       Cities = ["Delhi", "Hyd"];
       Rating = {"Rate":4.2, "Count":3500}
    }
    let tv = new Product();
    document.write(`
       Name = ${tv.Name}
     `);
</script>
```

Accessors

- Accessors will provide a fine grained control over property.
- They can control read and write operations on property.
- JavaScript class provides accessors
    a) get()
    b) set()

- get() is used to read and return a value from property.
- set() is used to write a new data into property.

Ex:

```
<script>
    var username = prompt("Enter User Name");
    var role = prompt("Enter Your Role","Admin|Manager|Customer");
    var productname = prompt("Enter New Product Name");

    class Product
    {
       _productName;

       get ProductName(){
          return this._productName;
       }

       set ProductName(newValue){
            if(role=="Admin"){
               this._productName = newValue;
            } else {
               document.write(`Hello ! ${username} - You are not authorized to set Product
Name`);
            }
        }
    }
    let obj = new Product();
    obj.ProductName = productname;
    if(obj.ProductName){
       document.write(`Product Name = ${obj.ProductName}`);
    }
</script>
```

Ex:

```
<script>
    class Product
    {
        Name = "Samsung TV";
        Rating = {
             "VendorRating": {
                 "Rate": 3.5,
                 "Count": 34
             },
             "CustomerRating": {
                "Rate": 3.1,
                "Count": 5000
             }
        }
        get CustomerRating(){
            return this.Rating.CustomerRating.Rate;
        }
    }
    let obj = new Product();
    document.write(obj.CustomerRating);
</script>
```

Ex:

```
<script>
    class Product
    {
        Name = "Samsung TV";
        Price = 45000.44;
        Qty = 2;
        Total(){
            return this.Qty * this.Price;
        }
        Print(){
            document.write(`Name=${this.Name}<br>Price=${this.Price}<br>Qty=${this.Qty}<br>
Total=${this.Total()}`);
        }
    }
    let obj = new Product();
    obj.Name = prompt("Enter Name");
    obj.Price = parseFloat(prompt("Enter Price"));
    obj.Qty = parseInt(prompt("Enter Qty"));
    obj.Print();
</script>
```

React Class Components

Polymorphism
- Poly stands for Many
- Morphos stands for Forms
- Configuring an object for various functionalities is know as Polymorphism.
- A object can handle various situations and change behaviour according to state and situation.
- Technically polymorphism allows a single base class object to use the memory of multiple derived classes.

Ex:

```
<script>
    class Employee
    {
        FirstName;
        LastName;
        Designation;
        Print(){
            document.write(`${this.FirstName} ${this.LastName} - ${this.Designation} <br>`);
        }
    }
    class Developer extends Employee
    {
        FirstName = "Raj";
        LastName = "Kumar";
        Designation = "Developer";
        Role = "Developer Role : Build , Debug, Test, Deploy";
        Print(){
            super.Print();
            document.write(this.Role);
        }
    }
    class Admin extends Employee
    {
        FirstName = "Kiran";
        LastName = "Rao";
        Designation = "Admin";
        Role = "Admin Role : Authorizations";
        Print(){
            super.Print();
            document.write(this.Role);
        }
    }
    class Manager extends Employee
    {
        FirstName = "Tom";
        LastName = "Hanks";
        Designation = "Manager";
        Role = "Manager Role : Approvals";
        Print(){
            super.Print();
            document.write(this.Role);
        }
    }

    var Employees = new Array(new Developer(), new Admin(), new Manager());
    var designation = prompt("Enter Designation");
    for(var employee of Employees){
        if(employee.Designation==designation){
            employee.Print();
        }
    }
</script>
```

React Class Components

- React Introduced components with JavaScript classes from its early versions.

```
        class   Login
        {

        }
```

- To give component behaviour every component class in React must extend "React.Component" base class.

```
        class   Login extents  React.Component
        {

        }
```

- Every class component must render markup by using "render()".
- render() is virtual DOM method used to add a react fragment into virtual DOM.
  Fragment = Container
- render() used "return" statement to return any markup.
- render() adds the returned markup into virtual DOM.

Syntax:

```
        export  class Login  extends  React.Component
        {
             render(){
                 return(
                    <React.Fragment> </React.Fragment>
                 )
             }
        }
```

Ex:

LoginClass.component.jsx

```
import React from "react";

export class LoginClassComponent extends React.Component
{
      render(){
        return(
            <React.Fragment>
               <div className="container-fluid">
               <h2>Login Class Component</h2>
               </div>
            </React.Fragment>
        )
      }
}
```

State for Class Component

- React class components are also known as Statefull components.
- React.Component base class provides a state in built for every component.
- You can't use "useState()" hook in class component.

Note: React Hooks are only for function component.

- In a class component state is configured at the time of creating component object.
- You can't configure state in any class method.
- It must be defined only in Constructor.

Syntax:

```
    export class  Login  extends React.Component
    {
          constructor(){
             super();
            this.state = { key:value }
        }
    }

    <div> {this.state.key } </div>
```

Ex:

```
import React from "react";

export class LoginClassComponent extends React.Component
{
      constructor(){
         super();
         this.state = {
            title : "Categories List",
            categories : ["Electronics", "Footwear", "Fashion"]
         }
      }
      render(){
        return(
            <React.Fragment>
               <div className="container-fluid">
               <h2>{this.state.title}</h2>
               <ol>
                  {
                    this.state.categories.map((category)=>
                     <li key={category}>{category}</li>
                    )
                  }
               </ol>
               </div>
            </React.Fragment>
        )
      }
}
```

- To Set state dynamically react component provides
  "this.setState()"

```
 this.setState({
       title : newValue
 })
```

- Data Binding
- Event Binding
- Class Binding
- Style Binding
- Life Cycle Hooks
  mount()
  unmount()

Class Components - Event Binding

Data Binding
Class Binding
Style Binding

Event Binding in Class Component

- React class component uses methods for events.
- Method of class is configured as "Subscriber".

Syntax:

```
   class   LoginComponent  extends React.Component
   {

       handleLoginClick() {

       }

       render() {
           return() {
              <button onClick={this.handleLoginClick}> </button>
           }
       }
```

```
    }
```

- All Event Args are same as function component.
- All Synthetic Events are same as function component.

Ex:
```
import React from "react";


export class LoginClassComponent extends React.Component
{
      constructor(){
         super();
      }
      handleLoginClick(e){
         alert(`
            Button Id :  ${e.target.id} \n
            Button Name: ${e.target.name} \n
            X Position : ${e.clientX}
         `);
      }
      render(){
        return(
            <React.Fragment>
               <div className="container-fluid">
                   <button onClick={this.handleLoginClick} id="btnLogin"
name="Login">Login</button>
               </div>
            </React.Fragment>
        )
      }
}
```

Note: If class component events are using "state" object then it is mandatory that you have to bind every event with current class.

a) In constructor your configure the bind()
b) You can define directly in event handler

Syntax:
```
        constructor(){
            super();
            this.handleLoginClick = this.handleLoginClick.bind(this);
         }
```

Syntax:

```
    <button onClick={this.handleLoginClick.bind(this)}>
```

Ex:

```
import React from "react";

export class LoginClassComponent extends React.Component
{
    constructor(){
      super();
      this.state = {
        Name : '',
        Price: 0,
        City: ''
      }
      this.handleNameChange = this.handleNameChange.bind(this);
      this.handlePriceChange = this.handlePriceChange.bind(this);
    }

    handleNameChange(e){
      this.setState({
        Name: e.target.value,
        Price: this.state.Price,
        City: this.state.City
      })
    }

    handlePriceChange(e){
      this.setState({
        Name: this.state.Name,
        Price: e.target.value,
        City: this.state.City
      })
    }
    handleCityChange(e){
      this.setState({
        Name: this.state.Name,
        Price: this.state.Price,
        City: e.target.value
      })
    }

    handleRegisterClick(){
      alert(JSON.stringify(this.state));
    }

    render(){
        return(
            <React.Fragment>
               <div className="container-fluid">
                   <div className="mb-2 mt-2">
                     <dl>
                        <dt>Name</dt>
                        <dd><input type="text" onChange={this.handleNameChange} /></dd>
                        <dt>Price</dt>
                        <dd><input type="number" onChange={this.handlePriceChange}/></dd>
                        <dt>City</dt>
                        <dd>
                           <select onChange={this.handleCityChange.bind(this)}>
                              <option>Delhi</option>
                              <option>Hyd</option>
                           </select>
                        </dd>
                        <button onClick={this.handleRegisterClick.bind(this)}>Register</button>
                     </dl>
                   </div>
                   <div>
                     <h2>Details</h2>
                     <dl>
                        <dt>Name</dt>
                        <dd>{this.state.Name}</dd>
                        <dt>Price</dt>
                        <dd>{this.state.Price}</dd>
                        <dt>City</dt>
                        <dd>{this.state.City}</dd>
                     </dl>
                   </div>
               </div>
            </React.Fragment>
        )
      }
}
```

Managing Mount and Unmount

- React class component provides Life Cycle Methods, which are known as Life Cycle hooks.
- The Life Cycle hooks can manage the mount and unmount phase of component.

```
        componentWillMount()
        componentDidMount()
        componentWillUnmount()
```

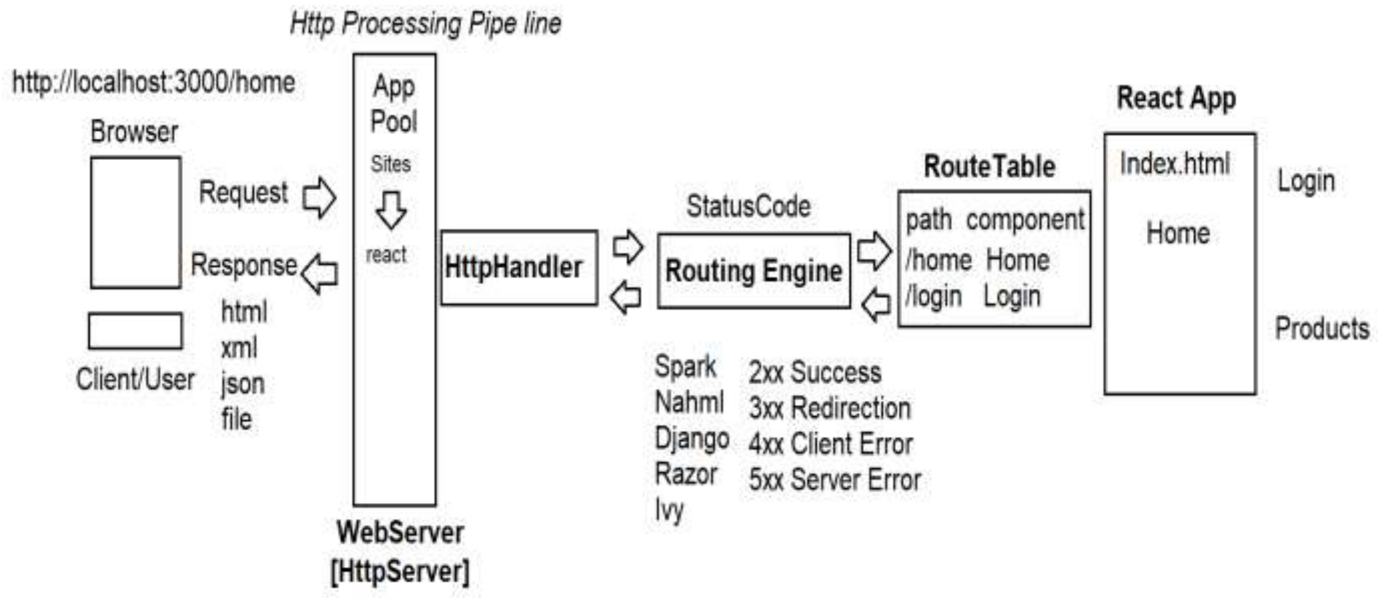
Ex:

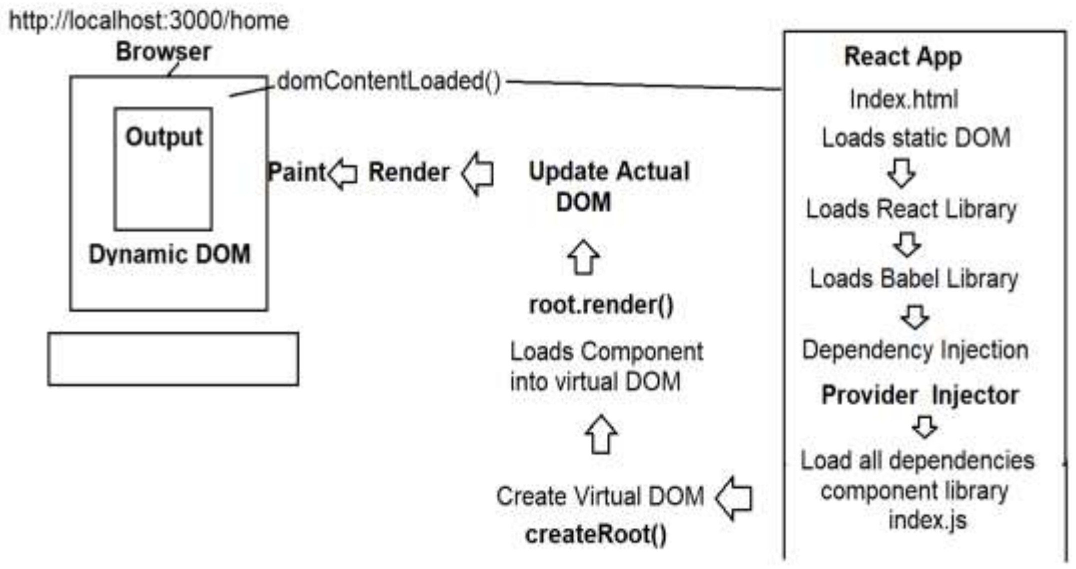
```
import React from "react";


export class FakestoreClass extends React.Component
{
    constructor(){
      super();
      this.state = {
         products: []
      }

    }

    componentDidMount(){
      fetch("http://fakestoreapi.com/products&quot;)
      .then(res=> res.json())
      .then(data=> {
         this.setState({
            products : data
         })
      })
    }


    render(){
      return(
         <div className="container-fluid">
            <h2>Fakestore Products</h2>
            <ol>
               {
                  this.state.products.map(product=>
                     <li key={product.id}>{product.title}</li>
                     )
               }
            </ol>
         </div>
      )
    }
}
```

React Component Life Cycle

Component Life Cycle Hooks

- Life Cycle hooks are a set of methods used to track the process of a class component from start to end.
- So that is easy to handle interactions at various phases of component.





React Life Cycle Methods

React Request Flow
React Application Bootstrap Process [Static DOM => Dynamic DOM]

Component Life Cycle

- It defines various phases of component from the time of creating a component and destroying the component.
- A component is created when user requests the component.
- A component is destroyed when another component is requested or when you close the appication.
- Every component have 3 major phases

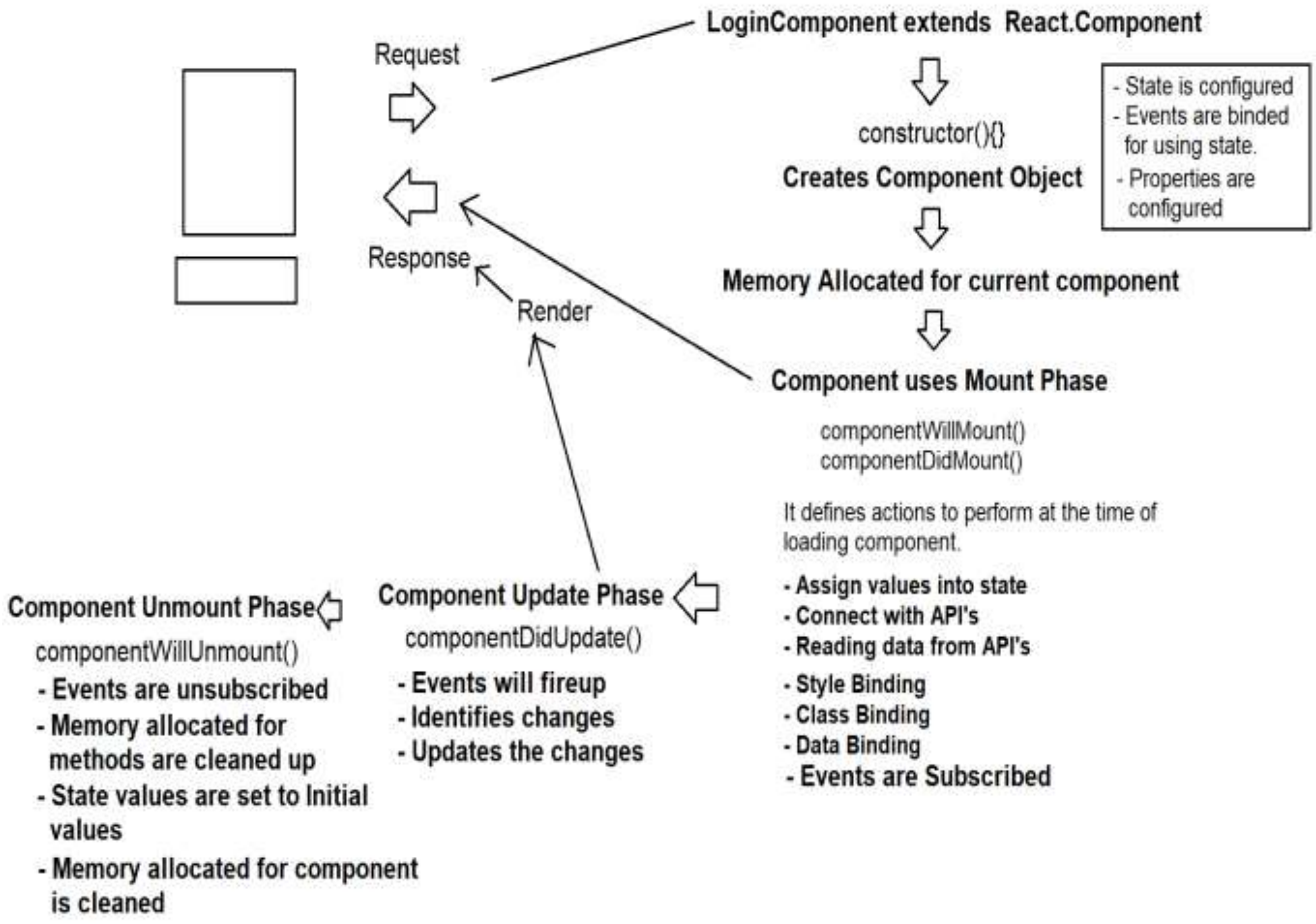
    a) Mount
    b) Update
    c) Unmount

- Mount
    componentWillMount()
    componentDidMount()
- Update
    componentWillUpdate()
- UnMount
    componentWillUnmount()

Note: The life span of every component is from component request to component destrory, which is trigged when any another component is requested.

Ex:

```
import React from "react";

export class LoginComponent extends React.Component
{
   componentDidMount(){
      alert(`Login Component Requested and Will mount`);
   }
   componentWillUnmount(){
      alert(`Login Component will unmount`);
   }
   render(){
      return(
         <div>
            <h2>User Login</h2>
            </div>
        )
    }
}

export class RegisterComponent extends React.Component
{
    componentDidMount(){
        alert(`Register Component Requested and Will mount`);
    }
    componentWillUnmount(){
        alert(`Register Component will unmount`);
    }
    render(){
        return(
            <div>
                <h2>Register User</h2>
            </div>
        )
    }
}

export class LifeCycleDemo extends React.Component
{
    constructor(){
        super();
        this.state = {
            View: ''
        }
    }
    handleLoginClick(){
        this.setState({
            View: <LoginComponent/>
        })
    }
    handleRegisterClick(){
        this.setState({
            View: <RegisterComponent/>
        })
    }
    render(){
        return(
            <div className="container-fluid">
                <h2>Life Cycle Demo</h2>
                <button onClick={this.handleLoginClick.bind(this)}>Login</button>
                <button onClick={this.handleRegisterClick.bind(this)}>Register</button>
                <hr />
            {this.state.View}
         </div>
      )
   }
}
```


LoginComponent extends React.Component
Request
Response
Render
constructor(){}
Creates Component Object
- State is configured
- Events are binded for using state.
- Properties are configured
Memory Allocated for current component
Component uses Mount Phase
componentWillMount()
componentDidMount()
It defines actions to perform at the time of loading component.
- Assign values into state
- Connect with API's
- Reading data from API's
- Style Binding
- Class Binding
- Data Binding
- Events are Subscribed
Component Update Phase
componentDidUpdate()
- Events will fireup
- Identifies changes
- Updates the changes
Component Unmount Phase
componentWillUnmount()
- Events are unsubscribed
- Memory allocated for methods are cleaned up
- State values are set to Initial values
- Memory allocated for component is cleaned

React Pure and Impure Components

FAQ: What are Pure and Impure Components?
Ans :

Impure:
- A component created by using "React.Component" is impure.
- It will re-render entire content even when there is no change.
- If a property or state changed then entire component will re-render.
- It is heavy on application
- It is slow in access.

Pure:
- A component created by using "React.PureComponent" is pure.
- It will not re-render entire component if there is no change.
- rendering content will happen only when change occured
  [state, property]
- It is faster in access
- It improves the performance
- Pure component identifies the changes by using
  "ChangeDetection" techniques

```
  InitialValue == FinalValue   => No Change Detected
 InitialValue !== FinalValue => Then Change Detected
```

- Pure component implicitly verifies initial and final values in a phase called

  "ComponentWillUpdate()"
  "ComponentDidUpdate()"

- They re-render content only when external state changes.

FAQ: What are pure and impure functions?
Ans: Impure function can modify the values outside function scope.

Syntax: Impure Function

```
var x = 0;

function Addition(a, b)
{
    x = a + b;
}
Addition(10, 20)
 console.log(x);
```

Syntax: Pure Function

```
        function  Addition(a,b)
        {
            return  a + b;
        }
        var x = Addition(10,20);
```

Ex:

```
import React from "react";


export class PureDemo extends React.PureComponent
{
     constructor(){
        super();
        this.state = {
            products: ["TV", "Mobile", "Watch"]
        }
     }

     handleLoadClick(){
        alert("hi");
        this.setState({
            products: ["TV", "Mobile", "Watch", "Shoes"]
        })
     }
     componentDidUpdate(){
        console.log("Component Will Render on Update");
     }

     render(){
        return(
            <div>
                <h2>Pure Component <button
onClick={this.handleLoadClick.bind(this)}>Load</button> </h2>
                <ol>
                    {
                        this.state.products.map((product)=>
                         <li key={product}>{product}</li>
                        )
                    }
                </ol>
            </div>
        )
     }
}
```

FAQ: Can we have pure and impure function components?
Ans: It can be for function component.

Conditional Rendering, React Component Properties in Class

Class Component Properties

- Function component properties are defined by using function parameters

```
    function Name(props)
    {
    }
```

- Class constructor allows to handle properties.
- Properties is an object with reference of keys and values.

Syntax:

```
    class  Template  extends  React.Component
    {
       constructor(props)
       {
        super();
       }
       render() {
          return(
              <div>{this.props.key}</div>
           )
       }
    }
```

Ex:
ClassPropertieDemo.jsx

```
import React from "react";

export class LoginTemplate extends React.Component
{
   constructor(props){
      super();
   }
    render(){
      return(
         <div className="container-fluid">
            <h2>{this.props.title}</h2>
```

```
                <dl>
                    <dt>{this.props.loginid}</dt>
                    <dd><input type={this.props.logintype} /></dd>
                    <dt>Password</dt>
                    <dd><input type="password"/></dd>
                </dl>
                <button>Login</button>
            </div>
        )
     }
}

export class ClassPropertiesDemo extends React.Component
{
    render(){
        return(
            <div>
                <h2>Class Properties</h2>
                <LoginTemplate title="Admin Login" loginid="Email" logintype="email" />
            </div>
        )
    }
}
```

Note : You can access function component in class and vice versa.

Conditional Rendering

- It is the process of changing the Virtual DOM dynamically, rendered by any component.
- You can any decision making statements to return and render specified DOM content.

Ex:
ConditionalRender.jsx

```
import React from "react";

export class ToolbarTemplate extends React.Component
{
     constructor(props){
        super();
     }
     render(){
         if(this.props.layout=="vertical"){
            return(
                <nav>
                    <div className="bi bi-facebook mb-3"></div>
                    <div className="bi bi-twitter mb-3"></div>
                    <div className="bi bi-linkedin mb-3"></div>
                    <div className="bi bi-instagram mb-3"></div>
                </nav>
            )
         } else {
            return(
                <nav className="d-flex">
                    <div className="bi bi-facebook me-3"></div>
                    <div className="bi bi-twitter me-3"></div>
                    <div className="bi bi-linkedin me-3"></div>
                    <div className="bi bi-instagram me-3"></div>
                </nav>
            )
         }
     }
}

export class ConditionalRender extends React.Component
{
     constructor(){
        super();
        this.state = {
            viewLayout: ''
        }
     }
     handleLayoutChange(e){
         this.setState({
            viewLayout: e.target.value
         })
     }
     render(){
        return(
            <div className="container-fluid">
                <h2>Select a Toolbar Layout</h2>
                <div className="w-25">
                    <select onChange={this.handleLayoutChange.bind(this)} className="form-
select">
                      <option value="-1">Select Layout</option>
                      <option value="horizontal">Horizontal</option>
                      <option value="vertical">Vertical</option>
                    </select>
                </div>
                <h3>Toolbar</h3>
                <ToolbarTemplate layout={this.state.viewLayout}/>
            </div>
        )
     }
```

```
}


Ex:
import React from "react";

export class ToolbarTemplate extends React.Component
{
    constructor(props){
       super();
    }
    render(){
        if(this.props.layout=="vertical"){
           return(
               <nav>
                  <div className="btn-group-vertical">
                       <button className="btn btn-danger mb-2"> <span className="bi bi-
facebook"></span> Facebook</button>
                       <button className="btn btn-danger mb-2"> <span className="bi bi-
twitter"></span> Twitter</button>
                       <button className="btn btn-danger mb-2"> <span className="bi bi-
linkedin"></span> Linked In</button>
                       <button className="btn btn-danger mb-2"> <span className="bi bi-
instagram"></span> Instagram</button>
                  </div>
               </nav>
           )
        } else {
           return(
               <nav className="d-flex">
                   <div className="bi bi-facebook me-3"></div>
                   <div className="bi bi-twitter me-3"></div>
                   <div className="bi bi-linkedin me-3"></div>
                   <div className="bi bi-instagram me-3"></div>
               </nav>
           )
        }
    }
}

export class ConditionalRender extends React.Component
{
    constructor(){
       super();
       this.state = {
            viewLayout: ''
       }
    }
    handleLayoutChange(e){
        this.setState({
           viewLayout: e.target.value
        })
    }
    render(){
       return(
           <div className="container-fluid">
               <h2>Select a Toolbar Layout</h2>
               <div className="w-25">
                   <select onChange={this.handleLayoutChange.bind(this)} className="form-
select">
                       <option value="-1">Select Layout</option>
                       <option value="horizontal">Horizontal</option>
                       <option value="vertical">Vertical</option>
                   </select>
               </div>
               <h3>Toolbar</h3>
               <ToolbarTemplate layout={this.state.viewLayout}/>
           </div>
       )
    }
}
```

FUNCTION COMPONENT VS CLASS COMPONENT

Class Component
- It is statefull component.
- It have implicit state configured.
- It is easy to extend.
- It provides various mutable techniques which are easy for desinging a reusable template.
- Properties and Methods
- Accessors with will fine control over the properties in class.
- It uses more memory.
- It is slow in rendering.
- It is complex in configurations.
- It uses lot of dependencies.

Function Components:
- It is stateless component.
- You have use state explicitly.
- A function provides all immutable members, however you can use state for making it mutable.
- Hard to extend
- It is faster

- It uses less memory
- It is modular.
- It provides various built-in hooks.

REACT FORMS AND VALIDATIONS

- Form provides an UI from where user can handle "CRUD" operations

C	Create
R	Read
U	Update
D	Delete

- Form in React is designed by using <form> element.
- Default events for <form> element are

```
onSubmit
onReset
```

- Default events have some default functionality to handle, you can prevent default funcitonality by using event argument

        "preventDefault()"

```
handleSubmitClick(e)
{
    e.preventDefault();
}
```

Ex:

```
export function FormDemo(){

    function handleFormSubmit(e){
      e.preventDefault();
      alert("Form Submitted");
    }
    return(
        <div className="container-fluid">
            <h2>Register User</h2>
            <form onSubmit={handleFormSubmit}>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" name="UserName"/></dd>
                <dt>Age</dt>
                <dd><input type="number" name="Age"/></dd>
                <dt>City</dt>
                <dd>
```

```
                    <select name="City">
                        <option>Delhi</option>
                        <option>Hyd</option>
                    </select>
                </dd>
                <dt>Mobile</dt>
                <dd>
                    <input type="text" name="Mobile"/>
                </dd>
            </dl>
            <button>Register</button>
            <button>Submit</button>
            <button type="button">Test</button>
            </form>
        </div>
    )
}
```

Ex:
FormDemo.jsx

```
import { useState } from "react";


export function FormDemo(){

    const [userDetails, setUserDetails] = useState({"UserName":"", "Age":0, "City":"", 
"Mobile":""});


    function handleNameChange(e){
        setUserDetails({
            UserName: e.target.value,
            Age: userDetails.Age,
            City: userDetails.City,
            Mobile: userDetails.Mobile
        })
    }

    function handleAgeChange(e){
        setUserDetails({
            UserName: userDetails.UserName,
            Age: e.target.value,
            City: userDetails.City,
            Mobile: userDetails.Mobile
        })
    }
    function handleCityChange(e){
        setUserDetails({
            UserName: userDetails.UserName,
            Age: userDetails.Age,
            City: e.target.value,
            Mobile: userDetails.Mobile
        })
    }
    function handleMobileChange(e){
        setUserDetails({
            UserName: userDetails.UserName,
            Age: userDetails.Age,
            City: userDetails.City,
            Mobile: e.target.value
        })
    }


    function handleFormSubmit(e){
      e.preventDefault();
      alert(JSON.stringify(userDetails));
    }
    return(
        <div className="container-fluid">
            <h2>Register User</h2>
            <form onSubmit={handleFormSubmit}>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" onChange={handleNameChange}
name="UserName"/></dd>
                <dt>Age</dt>
                <dd><input type="number" name="Age" onChange={handleAgeChange}/></dd>
                <dt>City</dt>
                <dd>
                    <select name="City" onChange={handleCityChange}>
                        <option>Delhi</option>
                        <option>Hyd</option>
                    </select>
                </dd>
                <dt>Mobile</dt>
                <dd>
                    <input type="text" name="Mobile" onChange={handleMobileChange}/>
                </dd>
            </dl>
            <button>Register</button>
            </form>
        </div>
    )
}
```

React Form Validations and Formik Library

Validating React Form

- Validation is the process of verifying user input.
- Validation is required to ensure that contradictionary and unauthorized data is not get stored into database.
- React implicitly have to depend on lot of JavaScript functions, which include

    a) string handling
    b) array functions
    c) math functions
    d) date functions
    e) number functions etc...
    b) regular expression

Ex:

```
import { useState } from "react";


export function FormDemo(){

    const [userDetails, setUserDetails] = useState({"UserName":"", "Age":0, "City":"",
"Mobile":""});
    const [errors, setErrors] = useState({"UserName":"", "Age":"", "City":"", "Mobile":""});


    function handleNameChange(e){
       if(e.target.value==""){
          setErrors({
            UserName: "User Name Required",
            Age: errors.Age,
            City: errors.City,
            Mobile: errors.Mobile
          })
       } else {
            setErrors({
                UserName: ""
            })
            setUserDetails({
                UserName: e.target.value,
                Age: userDetails.Age,
                City: userDetails.City,
                Mobile: userDetails.Mobile
            })
```

```
   }
}

function handleAgeChange(e){
    if(isNaN(e.target.value)){
        setErrors({
            Age: "Age must be a number"
        })
    } else {
        setErrors({
            Age: ""
        })
        setUserDetails({
            UserName: userDetails.UserName,
            Age: e.target.value,
            City: userDetails.City,
            Mobile: userDetails.Mobile
        })
    }
}

function handleCityChange(e){
   if(e.target.value=="-1") {
      setErrors({
        City: "Please Select your city"
      })
   } else {
     setErrors({
        City: ""
     })
    setUserDetails({
        UserName: userDetails.UserName,
        Age: userDetails.Age,
        City: e.target.value,
        Mobile: userDetails.Mobile
    })
   }
}
function handleMobileChange(e){
     if(e.target.value.match(/\+91\d{10}/)) {
        setUserDetails({
            UserName: userDetails.UserName,
            Age: userDetails.Age,
            City: userDetails.City,
            Mobile: e.target.value
        })
        setErrors({
              Mobile: ""
          })
        } else {
          setErrors({
              Mobile: "Invalid Mobile"
          })
        }
    }


    function handleFormSubmit(e){
      e.preventDefault();
      alert(JSON.stringify(userDetails));
    }
    return(
        <div className="container-fluid">
          <h2>Register User</h2>
          <form onSubmit={handleFormSubmit}>
          <dl>
              <dt>User Name</dt>
              <dd><input type="text" onChange={handleNameChange}
name="UserName"/></dd>
              <dd className="text-danger">{errors.UserName}</dd>
              <dt>Age</dt>
              <dd><input type="text" name="Age" onChange={handleAgeChange}/></dd>
              <dd className="text-danger">{errors.Age}</dd>
              <dt>City</dt>
              <dd>
                  <select name="City" onChange={handleCityChange}>
                      <option value="-1">Choose City</option>
                      <option value="Delhi">Delhi</option>
                      <option value="Hyd">Hyd</option>
                  </select>
              </dd>
              <dd className="text-danger">{errors.City}</dd>
              <dt>Mobile</dt>
              <dd>
                  <input type="text" name="Mobile" onChange={handleMobileChange}/>
              </dd>
              <dd className="text-danger">{errors.Mobile}</dd>
          </dl>
          <button>Register</button>
          </form>
        </div>
    )
}
```

3rd Party Form Library for React

- Formik
- Kendo [Telerik]
- DevExpress etc..

Formik Library

1. Install

```
> npm  install  formik  --save                  [in your project terminal]
```

2. Import

```
    import  { useFormik } from  "formik";
```

3. Configure a form using formik

```
    const  formik =  useFormik({
            initialValues:    It is an object that defines the values a form have to handle.,
            onSubmit: It defines the actions to perform when form is submitted,
            validate :  It refers to a function that validates the values of form.
            validationSchema: It uses pre-defined validation functions.
            onBlur    : It defines actions to perform when an element looses focus.
    })
```

4. You have to bind the formik object with form elements

```
        <input type="text"  name="Age"  onChange={formik.handleChange}
        <form  onSubmit={formik.handleSubmit}>
```

Ex: formik-demo.jsx

```
import { useFormik } from "formik";

export function FormikDemo()
{
    const formik = useFormik({
        initialValues: {
            "UserName": "",
            "Age": 0,
            "City":"",
            "Mobile":""
        },
        onSubmit: (values) => {
            alert(JSON.stringify(values));
        }
    })
    return(
        <div className="container-fluid">
           <form onSubmit={formik.handleSubmit}>
           <h2>Register - Formik Form</h2>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" onChange={formik.handleChange}
name="UserName"/></dd>
                <dt>Age</dt>
                <dd><input type="text" onChange={formik.handleChange} name="Age" /></dd>
                <dt>City</dt>
                <dd>
                    <select onChange={formik.handleChange} name="City">
                        <option>Choose City</option>
                        <option>Delhi</option>
                        <option>Hyd</option>
                    </select>
                </dd>
                <dt>Mobile</dt>
                <dd>
                    <input type="text" onChange={formik.handleChange} name="Mobile" />
                </dd>
            </dl>
            <button>Register</button>
           </form>
        </div>
    )
}
```

Ex: Formik Validation

```
import { useFormik } from "formik";

export function FormikDemo()
{

    function UserVaidation(formBody)
    {
        var errors = {};

        if(formBody.UserName==""){
            errors.UserName="User Name Required";
        } else if(formBody.UserName.length<4) {
            errors.UserName="User Name too short min 4 chars required";
        } else {
            errors.UserName="";
        }
        if(isNaN(formBody.Age)){
            errors.Age = "Age must be a number";
        } else {
            errors.Age ="";
        }

        if(formBody.City=="-1"){
            errors.City = "Please Select your City";
        } else {
            errors.City = "";
        }

        if(formBody.Mobile.match(/\+91\d{10}/)) {
            errors.Mobile = "";
        } else {
            errors.Mobile = "Invalid Mobile"
        }

        return errors;
    }

    const formik = useFormik({
        initialValues: {
            "UserName": "",
            "Age": 0,
            "City":"",
            "Mobile":""
        },
        validate: UserVaidation,
        onSubmit: (values) => {
            alert(JSON.stringify(values));
        }
    })
    return(
        <div className="container-fluid">
          <form onSubmit={formik.handleSubmit}>
          <h2>Register - Formik Form</h2>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text" onBlur={formik.handleBlur}
onChange={formik.handleChange} name="UserName"/></dd>
                <dd className="text-danger">{formik.errors.UserName}</dd>
                <dt>Age</dt>
                <dd><input type="text" onChange={formik.handleChange} name="Age" /></dd>
                <dd className="text-danger">{formik.errors.Age}</dd>
                <dt>City</dt>
                <dd>
                    <select onChange={formik.handleChange} name="City">
                        <option value="-1">Choose City</option>
                        <option value="Delhi">Delhi</option>
                        <option value="Hyd">Hyd</option>
                    </select>
                </dd>
                <dd className="text-danger">{formik.errors.City}</dd>
                <dt>Mobile</dt>
                <dd>
                    <input type="text" onChange={formik.handleChange} name="Mobile" />
                </dd>
                <dd className="text-danger">{formik.errors.Mobile}</dd>
            </dl>
            <button>Register</button>
           </form>
        </div>
    )
}
```

React Formik and Yup library

Formik Library
Formik Validation - using function

Yup Library for Validation Schema

- Validation schema refers to a validation structure, which is an object that contains pre-defined validation functions.

```
        required()
        number()
        string()
        min()
        max()
        matches()  etc..
```

1. Install yup library

```
> npm  install yup  --save
```

2. Import the function that you need from yup

```
    import  { required,  matches, min }  from  "yup";
    import  * as yup from "yup";

            yup.required()
            yup.min() etc..
```

3. Formik will use yup library for validation schemas.

Ex:
 yup-demo.jsx

```
import { useFormik } from "formik";
import * as yup from "yup";

export function YupDemo(){

    const formik = useFormik({
        initialValues : {
            "UserName":"",
            "Age": 0,
            "City":"",
            "Mobile":""
        },
        onSubmit: (values)=>{
            alert(JSON.stringify(values));
        },
        validationSchema: yup.object({
            "UserName": yup.string()
                        .required("User Name Required")
                        .min(4, "Name too short..")
                        .max(10, "Name too long.."),
            "Age": yup.number()
                     .required("Age Required"),
            "Mobile": yup.string()
                      .required("Mobile Required")
                      .matches(/\+91\d{10}/, "Invalid Mobile +91 with 10 digits")
        })
    })

    return(
        <div className="container-fluid">
        <form onSubmit={formik.handleSubmit}>
        <h2>Register - Formik Yup</h2>
         <dl>
            <dt>User Name</dt>
            <dd><input type="text"
onChange={formik.handleChange}  name="UserName"/></dd>
            <dd className="text-danger">{formik.errors.UserName}</dd>
            <dt>Age</dt>
            <dd><input type="text" onChange={formik.handleChange} name="Age" /></dd>
            <dd className="text-danger">{formik.errors.Age}</dd>
            <dt>City</dt>
            <dd>
                <select onChange={formik.handleChange}  name="City">
                    <option value="-1">Choose City</option>
                    <option value="Delhi">Delhi</option>
                    <option value="Hyd">Hyd</option>
                </select>
            </dd>
            <dd className="text-danger"></dd>
            <dt>Mobile</dt>
            <dd>
                <input type="text" onChange={formik.handleChange} name="Mobile" />
            </dd>
            <dd className="text-danger">{formik.errors.Mobile}</dd>
        </dl>
        <button>Register</button>
       </form>
    </div>
   )
}
```

- Formik can handle various properties and events of any element in component by using "spread" syntax.

```
    <input type="text"  {...formik.getFieldProps("Name")} >
```

Ex:

```
import { useFormik } from "formik";
import * as yup from "yup";

export function YupDemo(){

    const formik = useFormik({
        initialValues : {
            "UserName":"",
            "Age": 0,
            "City":"",
            "Mobile":""
        },
        onSubmit: (values)=>{
            alert(JSON.stringify(values));
        },
        validationSchema: yup.object({
            "UserName": yup.string()
                           .required("User Name Required")
                           .min(4, "Name too short..")
                           .max(10, "Name too long.."),
            "Age": yup.number()
                      .required("Age Required"),
            "Mobile": yup.string()
                         .required("Mobile Required")
                         .matches(/\+91\d{10}/, "Invalid Mobile +91 with 10 digits")
        })
    })

    return(
        <div className="container-fluid">
        <form onSubmit={formik.handleSubmit}>
        <h2>Register - Formik Yup</h2>
         <dl>
             <dt>User Name</dt>
             <dd><input type="text" {...formik.getFieldProps("UserName")}
name="UserName"/></dd>
             <dd className="text-danger">{formik.errors.UserName}</dd>
             <dt>Age</dt>
             <dd><input type="text" {...formik.getFieldProps("Age")} name="Age" /></dd>
             <dd className="text-danger">{formik.errors.Age}</dd>
             <dt>City</dt>
             <dd>
                 <select {...formik.getFieldProps("City")} name="City">
                     <option value="-1">Choose City</option>
                     <option value="Delhi">Delhi</option>
                     <option value="Hyd">Hyd</option>
                 </select>
             </dd>
             <dd className="text-danger"></dd>
             <dt>Mobile</dt>
             <dd>
                 <input type="text" {...formik.getFieldProps("Mobile")} name="Mobile" />
             </dd>
             <dd className="text-danger">{formik.errors.Mobile}</dd>
         </dl>
         <button>Register</button>
        </form>
       </div>
    )
}
```

FORMIK COMPONENTS

- Formik components are more light weight and rich in markup, styles and functionality.

```
    <Formik>
    <Form>
    <Field>
    <ErrorMessage>
```

- <Formik> is container that handles your form element.
  It comprises of initialValues, onSubmit, onBlur, validation, validationSchema.

```
      <Formik   initialValues={}  onSubmit={}   validationSchema={ }>
        {
          <Form>

          </Form>
        }
    </Formik>

        <form>        HTML form
        <Form>         Formik form
```

- <Field> is a component for rendering <input> <textarea> <select>..

```
        <Field type="text" name="UserName">
        <Field type="email">
        <Field as="select">
```

- <ErrorMessage> is a component that can track errors object and return error.

- You can capture the properties returned by form so that you can handle form state.

```
                isValid
                dirty
                errors
```

Syntax:

```
            props => <Form> </Form>
```

Ex:
formik-component.jsx

```
import { Formik, Form, Field, ErrorMessage } from "formik";
import * as yup from "yup";

export function FormikComponent()
{
    return(
        <div className="container-fluid">
            <h2>Formik Components</h2>
            <Formik
              initialValues={{
                  "UserName":"",
                  "Age": 0,
                  "City":"",
           "Mobile":""
        }}

        onSubmit={
           (values)=>{
              alert(JSON.stringify(values));
           }
        }

        validationSchema={
           yup.object({
              "UserName": yup.string().required("Name Required")
                           .min(4, "Name too short")
                           .max(10, "Name too long"),
              "Age": yup.number().required(),
              "Mobile": yup.string().required().matches(/\+91\d{10}/,"Invalid Mobile")
           })
        }
      >
         {
            props =>
            <Form>
                <dl>
                    <dt>User Name</dt>
                    <dd><Field type="text" name="UserName"/></dd>
                    <dd className="text-danger"><ErrorMessage name="UserName"/></dd>
                    <dt>Age</dt>
                    <dd><Field type="text" name="Age"/></dd>
                    <dd className="text-danger" ><ErrorMessage name="Age"/></dd>
                    <dt>City</dt>
                    <dd>
                        <Field as="select">
                            <option>Choose City</option>
                            <option>Delhi</option>
                            <option>Hyd</option>
                        </Field>
                    </dd>
                    <dt>Mobile</dt>
                    <dd><Field type="text" name="Mobile" /></dd>
                    <dd className="text-danger"><ErrorMessage  name="Mobile"/></dd>
                </dl>
                <button disabled={!props.isValid}>Register</button>
                <button disabled={!props.dirty}>Save</button>
            </Form>
         }
      </Formik>
   </div>
```

```
    )
}
```