
React

INDEX

React
or
React JS

Introduction

- React is a free and open-source JavaScript library for building user interfaces based on UI component.
- React and React JS both are same.

Link: [React Official site]

Q. What are the challenges in modern web development?
Ans. Web started in early 1990's by Tim Berners Lee. Current Generation, 90% users are using web with smart devices, so traditional web applications have some following issues with modern devices.

1. Unified UX
    a. Application must have same experience across all devices.
    b. Mobile users must get access to everything.
2. Fluid UX
    a. User stays on one page and can get access to everything on to the page.
3. Loosely coupled and extendable
    a. You can build the features remotely.
    b. Push the new features onto existing app without having catastrophic failures.
4. Simplified Deployment
    a. Single step installing and open the application

Q. What is the solution for above problem?
Ans. Better to build the applications as

- SPA [Single page applications]
    - E.g., Twitter, Facebook, Instagram, Netflix, etc.
- PWA [Progressive Web applications]
    - E.g., Online editor websites.

Q. How to build SPA & PWA and can we user HTML, CSS, JavaScript, jQuery?
Ans. Yes, we can use HTML, CSS, JavaScript, jQuery to build SPA.

Is sues with Javascript and jQuery

- Large DOM manipulation (jQuery)
- Lot of coding
- Heavy
- More memory
- Legacy

Modern Technologies to build SPA, PWA: Angular, React, Vue, Knockout JS, Backbone, Ember, etc.

Q. What is difference between Angular and React?

Ans.

- Angular is a Framework. React is a Library.
- If any project is looking for a complete Framework in front and also better UI then go with Angular.
- If you want to build a heavy application with good UI along with flow of application to control client-side server then go for Angular.
- Already the Backend application has built using any other Framework only better UI you need then go with React.

Different Library of JavaScript

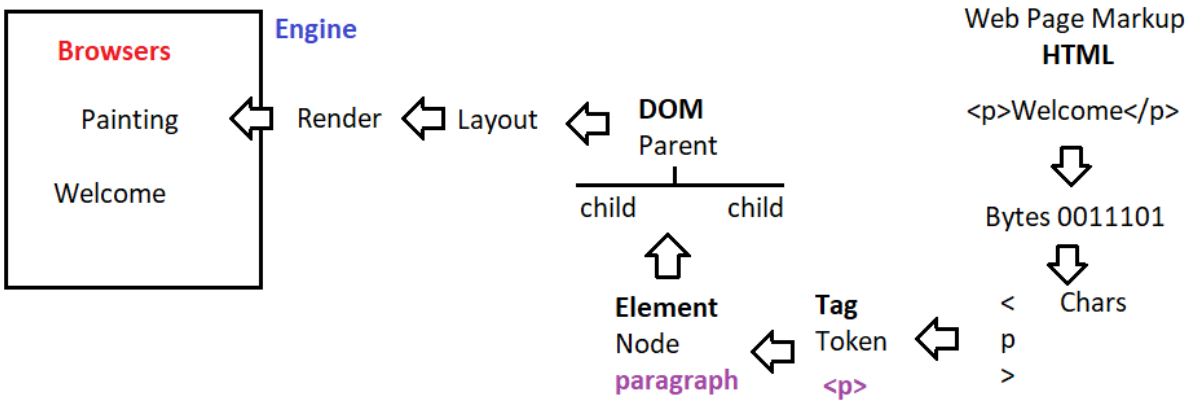
- jQuery, React, Rx JS, Redux

Background Preparation

Q. What is DOM?

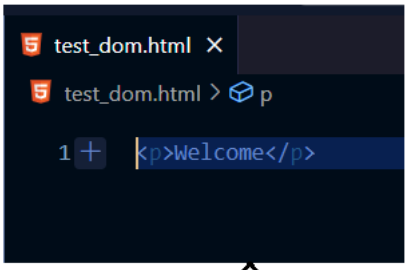
Ans.

- DOM stands Document Object Model.
- DOM is a hierarchy of presenting elements in browser.
- HTML represents the DOM and JavaScript, jQuery manipulates the DOM. Nothing but HTML is a static DOM and JavaScript will make is as dynamic DOM.



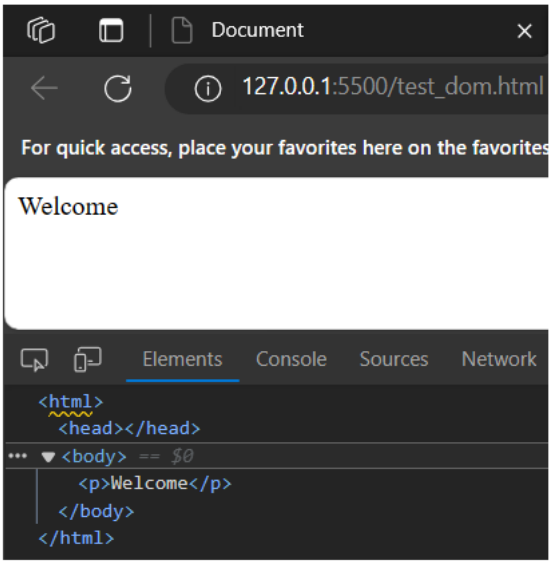
Let visualize the DOM.

1. Create a file called test_dom.html
2. Just simply add tag “<p>welcome<p>”
3. Now open that HTML file in browser.
4. Inspect the page.

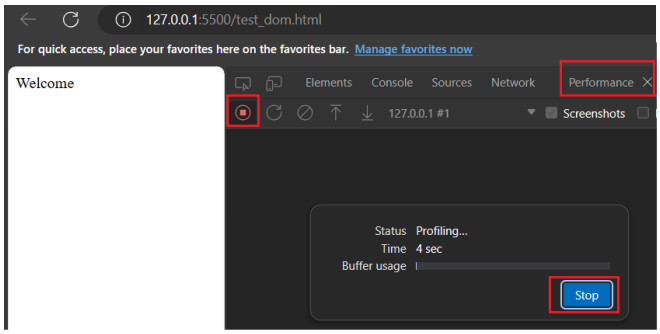


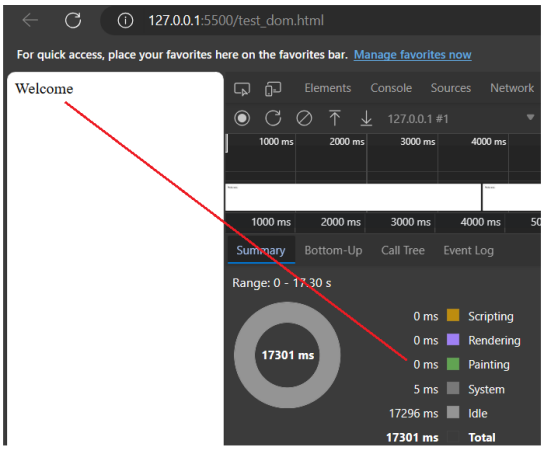
Here we are not following the hierarchy.

This hierarchy order of presentation is called DOM

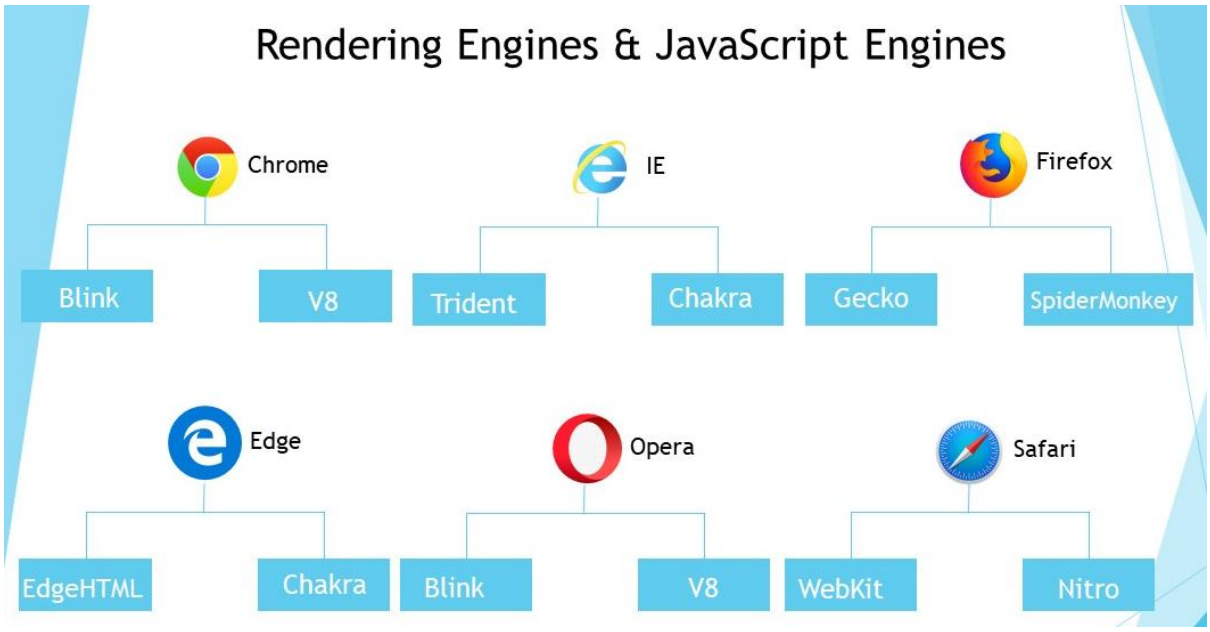


Everything visible in browser is the result of Painting





Browser with their Engines



DOM and CSSOM

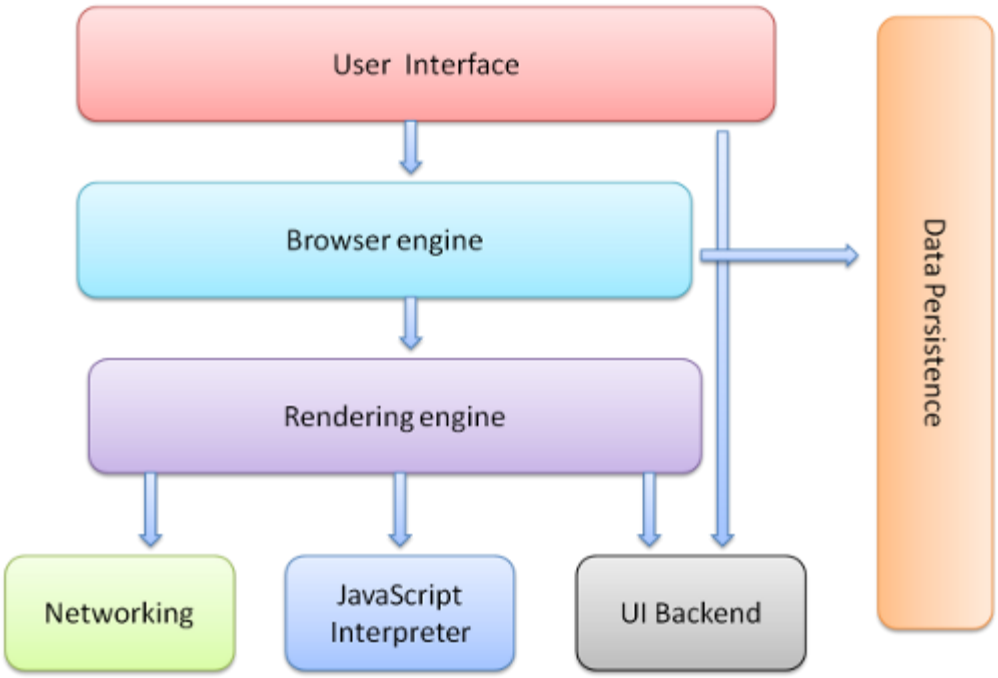
test_dom.html > ...

```
1   <style>
2       p {
3           color: red;
4       }
5   </style>
6   <p>Welcome</p>
```

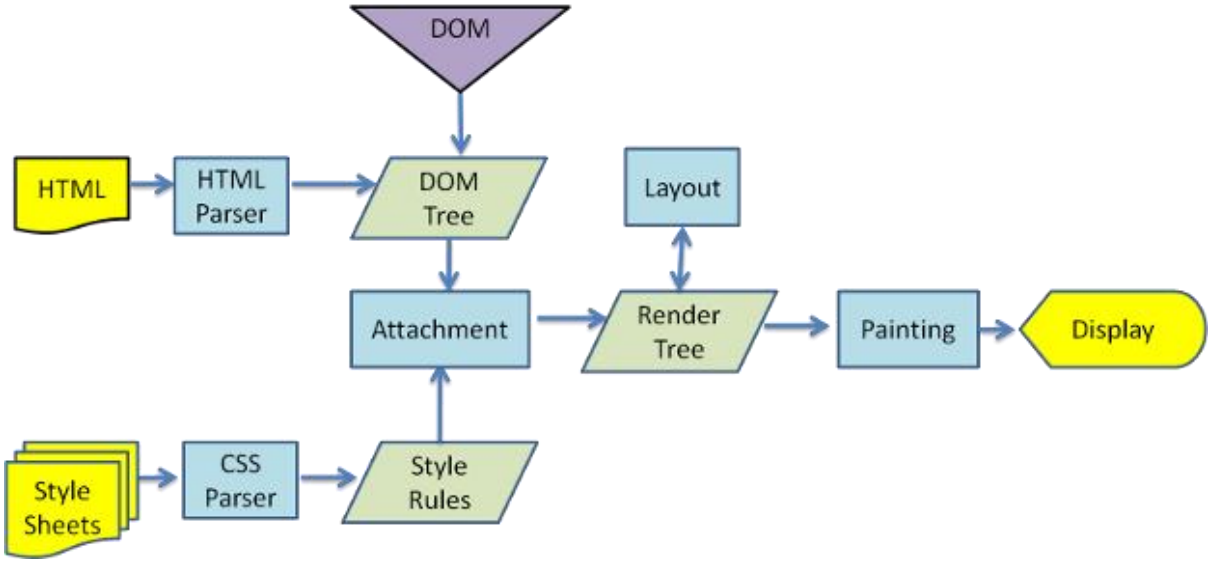
CSSOM

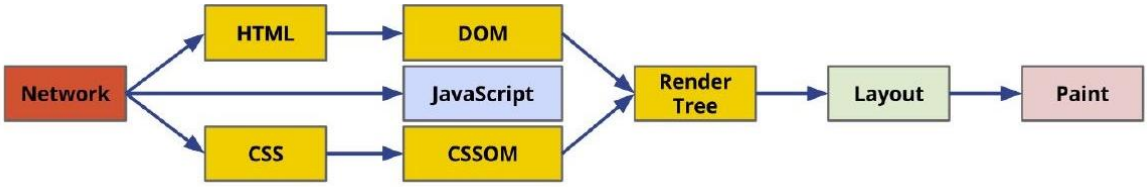
DOM

Browser Architecture

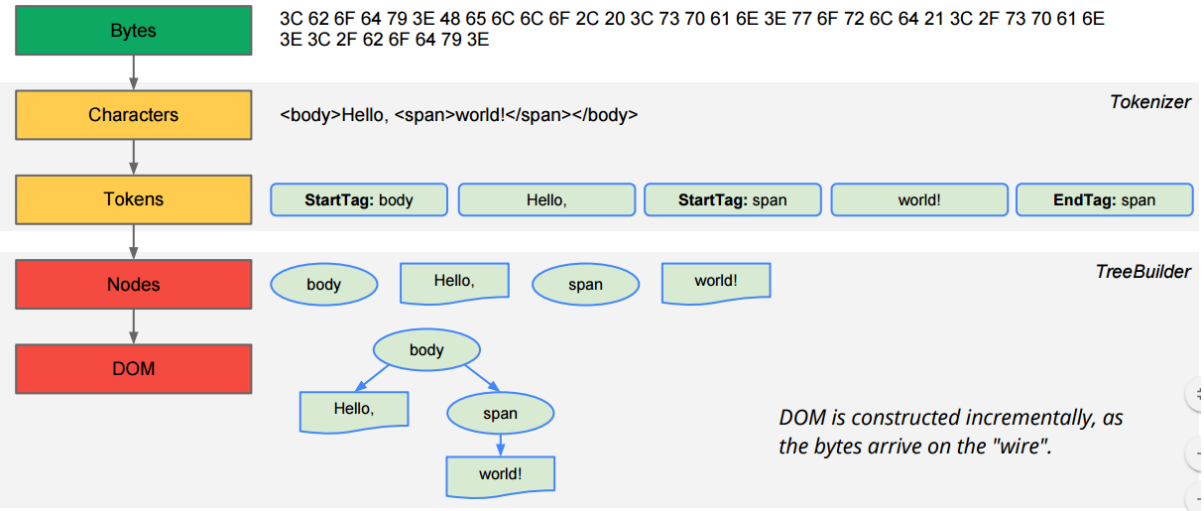


Critical rendering path

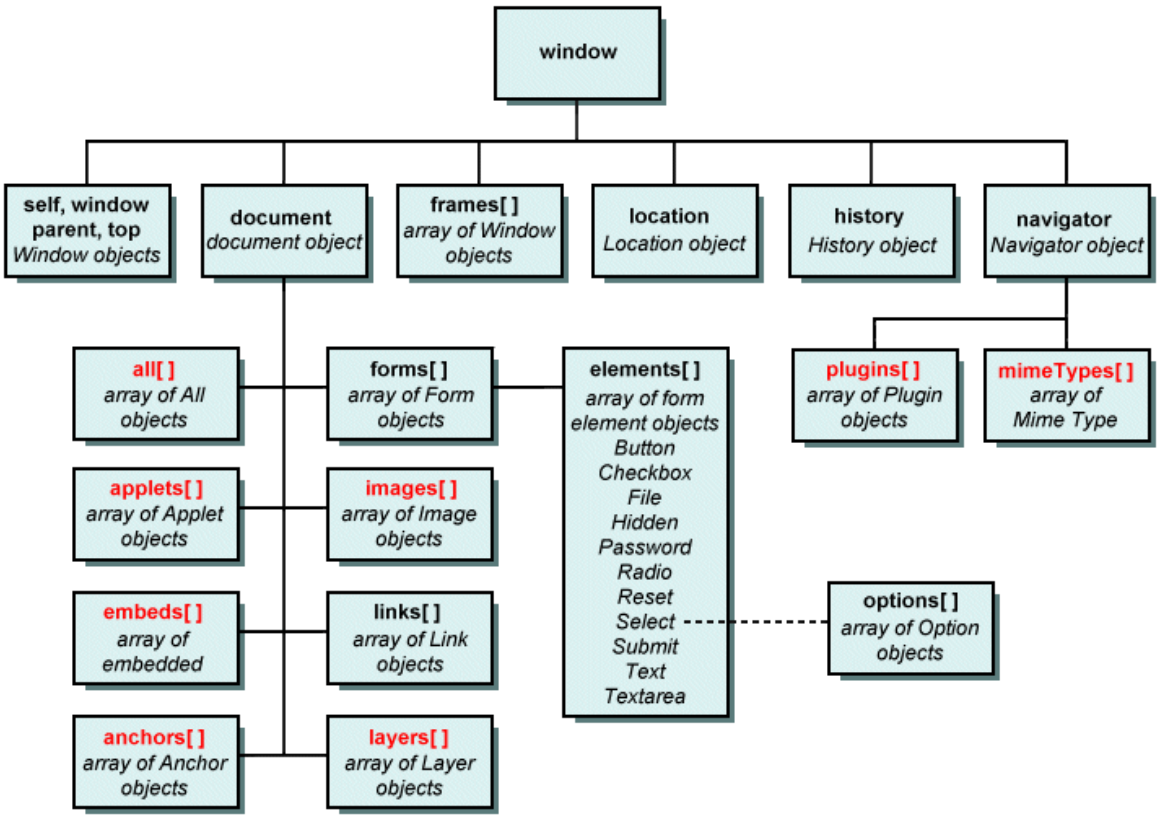




How HTML Parse Working



DOM object Hierarchy



Q. What is Shadow DOM?

Ans. It is a hierarchy configured for every component in HTML.

- Shadow DOM or Component is comprising of following things
  - Markup - HTML
  - Style - CSS
  - Logic or function - JavaScript

Let visualize the Shadow DOM.

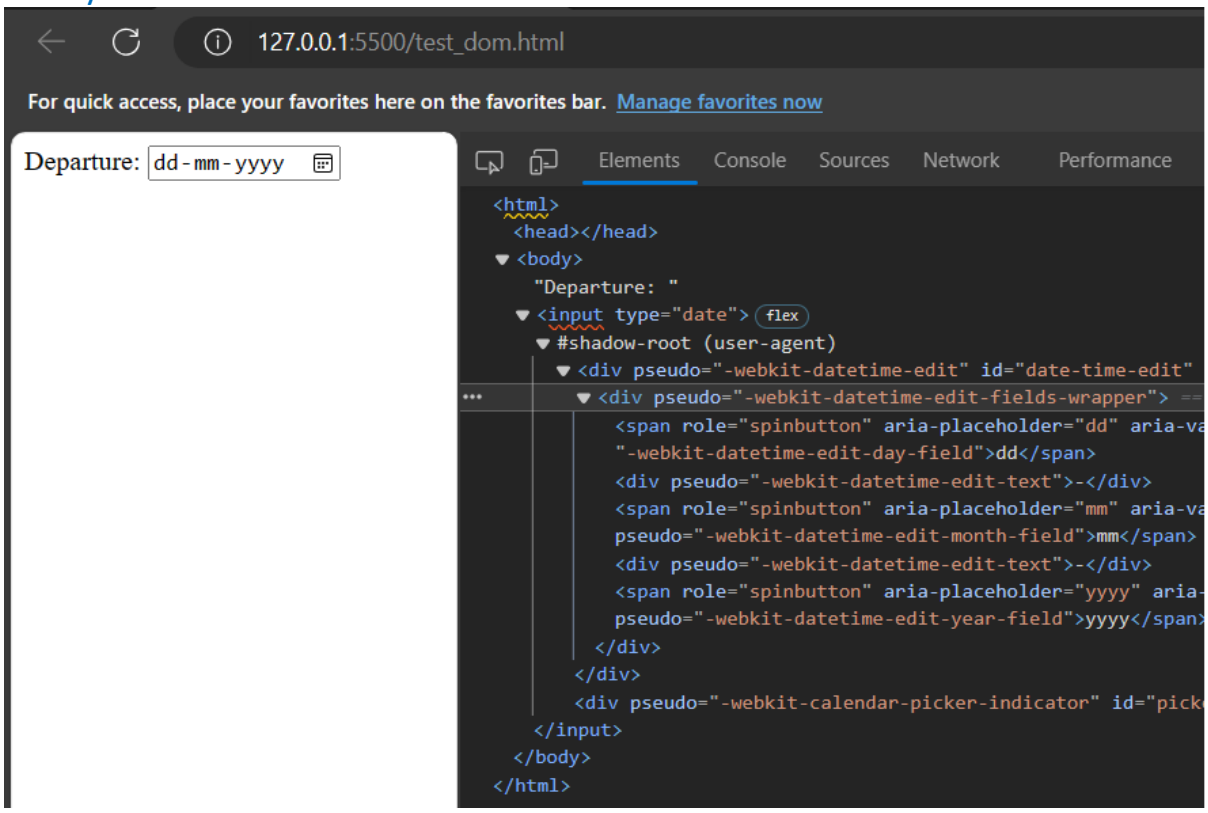
1. Create a file called test_dom.html
2. Just simply add the below code

```
Departure: <input type="date">
```

3. Now open that HTML file in browser and inspect the page,

Note: Directly Shadow DOM is not visible, first enable the below setting Right click on the web page and inspect then go to setting > Preferences > Elements section then check “Show user agent shadow DOM” option.

Now you can see the Shadow DOM



Key features of React

1. Virtual DOM
    - It is a copy of actual DOM in memory.
    - Update are first appended into Virtual DOM and then into actual DOM.
    - Faster in rendering output.
2. Component Based
    - Reusability, Maintainability, Testability, Loosely Coupled
3. Modular
    - Modular means application specific library.
    - Modular means the library required for that situation will be loaded.
    - Implements lazy loading.
    - Make the application light weight.
    - Make the application faster.
4. AOT technique
    - JavaScript is JIT technique (Just in Time)
    - React and Angular using AOT technique (Ahead of time).
    - Compile the Script code in application itself not in Browser.

Q. What are the issues with React?

Ans. React have the following issues

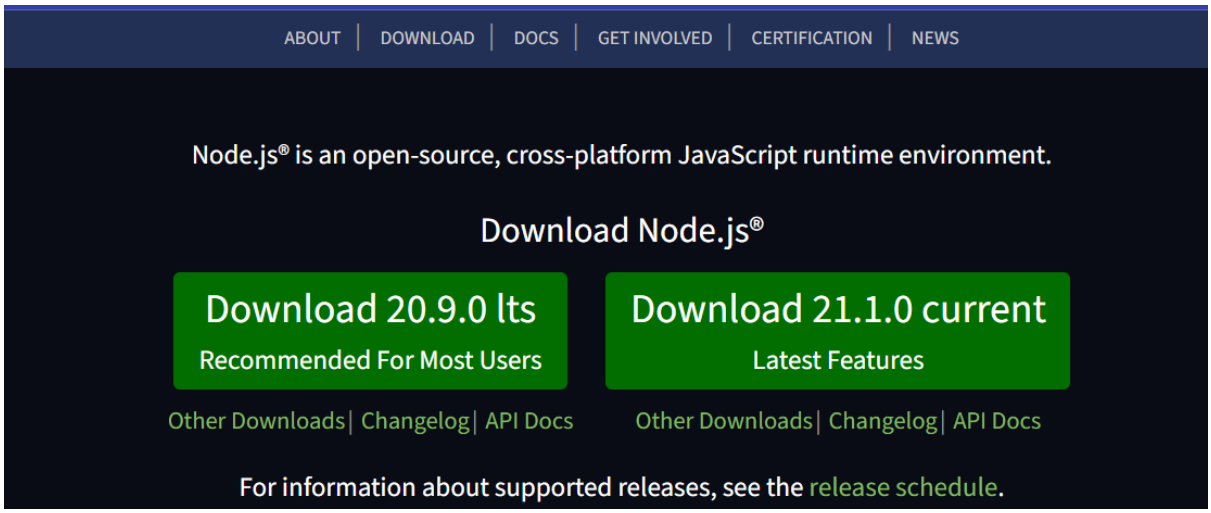
- React is not designed for our requirements hence lot of GAP's.
- To fill the GAP's, we have to depend on lot of 3rd parties.
  Like, [Telerik]
- SEO (Search engine optimization) issues.
- Pace of development.
- Poor documentation.

Setup Environment for Web Development

Step 1: Download and install "Node JS" on your PC. [Node JS]

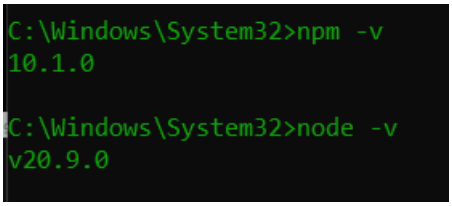
- We are downloading Node JS for a package manger called NPM.
- NPM stands for Node Package Manager. It is a package manager for JavaScript.
- There are various package manages
    - NPM
    - Bower
    - Yarn
    - RubyGems

Download the “Recommend for Most Users” Node JS and install it.



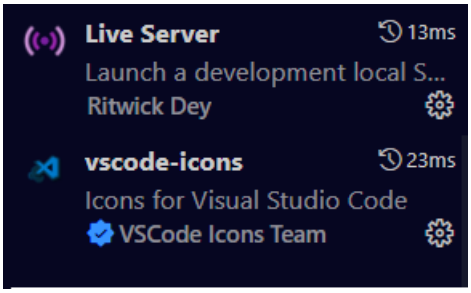
After install check the version of Node JS and NPM. For that go to command prompt and run the below commands.

```
> node -v
> npm -v
```



Step 2: Download and install visual studio code. [Link]

Step 3: Download the following extension in Visual studio code.



Create a New Project

Step 1: Create a new folder on your PC for project.
E:\react-webapp

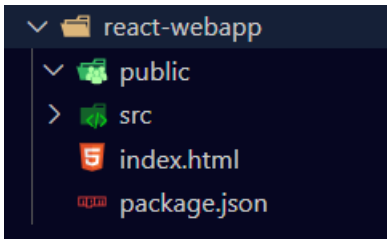
Step 2: Open the folder in your VS-Code editor. Or open CMD in that folder user "**code .**" in CMD.

Step 3: Open terminal from "Terminal" menu or use shortcut (CTRL+SHIFT +`).

Step 4: Run the following command, then it will add a file to your folder i.e., **package.json**.

```
> npm init -y
```

Step 5: Create the below structure folder and files



public: It comprises of static resources like HTML, images, other documents.
src: It comprises of dynamic resources likes JS, CSS, Sass, TS files.
index.html: Main file

Step 6: Now run using Live server.

Note: **package.json** contains meta data i.e. information about your application like which version, what dependency all things are there.

JavaScript Versions:

- ES5 [ECMA Script]
- ES6 [Maximum]
- ES7
- ESNext

Note:

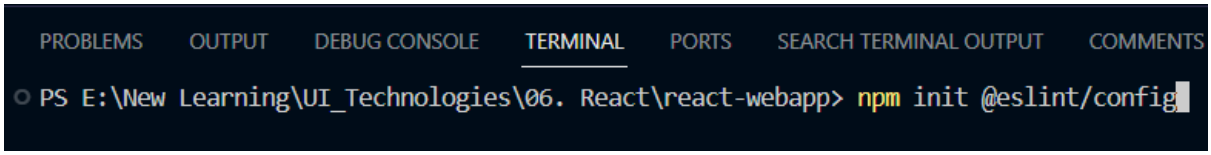
- ✓ For JavaScript language the analysis tool is ESLint. [Link]
- ✓ For JavaScript you can add "ESLint" extension in Visual studio code.

- ✓ If any project contains JavaScript the rules for can define by using a configuration file called as **eslintrc.json**.
- ✓ To add ESLint in your project i.e., nothing but adding the **eslintrc.json** configuration in the project use the following command in terminal.

```
> npm init @eslint/config
```

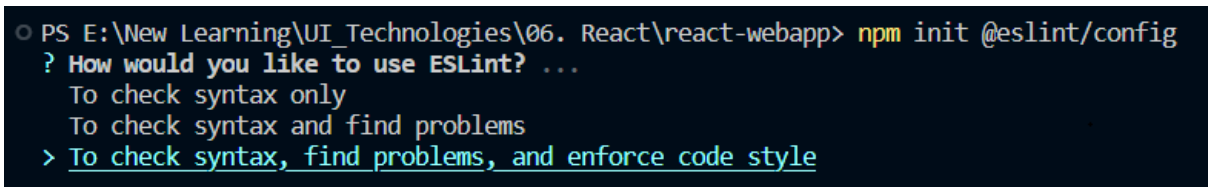
Add ESLint in your Project

Step 1: Open Terminal on the project and run the below command

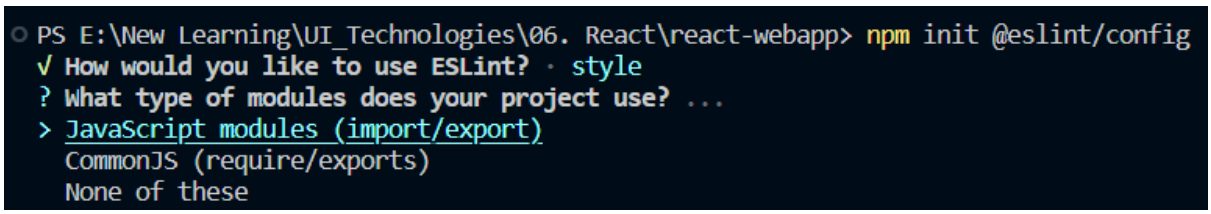


Step 2: Now it will ask the following questions show choose the option according to your need.

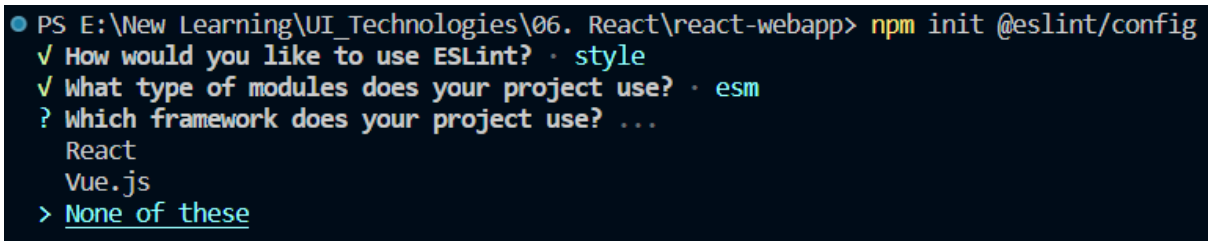
Here choose the **To check syntax, find problems, and enforce code style** option.



Here choose the **JavaScript modules (import/export)** option.



Here we are not reacting react application so as of now choose **None of these**



Here chooses the **No** option, because we are not using TypeScript as of now.

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
? Does your project use TypeScript? » No / Yes
```

Here chooses the **Browser** option.

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
? Where does your code run? ...  (Press <space> to select, <a> to toggle all, <i> to invert selection)
√ Browser
√ Node
```

Here choose the **Use a popular style guide** option.

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
? How would you like to define a style for your project? ...
> Use a popular style guide
  Answer questions about your style
```

Here choose the **Google** one then next once choose **JSON** option.

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
? Which style guide do you want to follow? ...
  Airbnb: https://github.com/airbnb/javascript
  Standard: https://github.com/standard/standard
> Google: https://github.com/google/eslint-config-google
  XO: https://github.com/xojs/eslint-config-xo
```

```
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
√ Which style guide do you want to follow? · google
? What format do you want your config file to be in? ...
  JavaScript
  YAML
> JSON
```

Here chooses **Yes** option

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
√ Which style guide do you want to follow? · google
√ What format do you want your config file to be in? · JSON
Checking peerDependencies of eslint-config-google@latest
The config that you've selected requires the following dependencies:

eslint-config-google@latest eslint@>=5.16.0
? Would you like to install them now? » No / Yes
```

Here choose **npm** option.

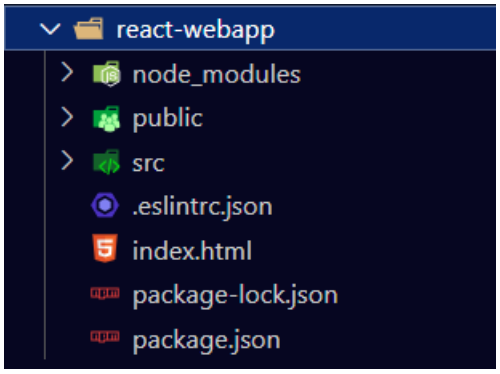
```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
√ Which style guide do you want to follow? · google
√ What format do you want your config file to be in? · JSON
Checking peerDependencies of eslint-config-google@latest
The config that you've selected requires the following dependencies:

eslint-config-google@latest eslint@>=5.16.0
√ Would you like to install them now? · No / Yes
? Which package manager do you want to use? ...
> npm
  yarn
  pnpm
```

Now the download will start, it will take some time after that you can see the Successfully created message.

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
√ Which style guide do you want to follow? · google
√ What format do you want your config file to be in? · JSON
Checking peerDependencies of eslint-config-google@latest
The config that you've selected requires the following dependencies:

eslint-config-google@latest eslint@>=5.16.0
√ Would you like to install them now? · No / Yes
√ Which package manager do you want to use? · npm
Installing eslint-config-google@latest, eslint@>=5.16.0
[#########.........] \ idealTree:eslint-plugin-react: timing idealTree:node_modules/eslint-plugin-react Completed in 130ms
```

```
PS E:\New Learning\UI_Technologies\06. React\react-webapp> npm init @eslint/config
√ How would you like to use ESLint? · style
√ What type of modules does your project use? · esm
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ How would you like to define a style for your project? · guide
√ Which style guide do you want to follow? · google
√ What format do you want your config file to be in? · JSON
Checking peerDependencies of eslint-config-google@latest
The config that you've selected requires the following dependencies:

eslint-config-google@latest eslint@>=5.16.0
√ Would you like to install them now? · No / Yes
√ Which package manager do you want to use? · npm
Installing eslint-config-google@latest, eslint@>=5.16.0

added 193 packages, and audited 194 packages in 3s

90 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Successfully created .eslintrc.json file in E:\New Learning\UI_Technologies\06. React\react-webapp
```

Step 3: Now you can see some extract folder and files are added to your project.



ES6 [ECMA Script 2016]

Basics of JavaScript

Variables

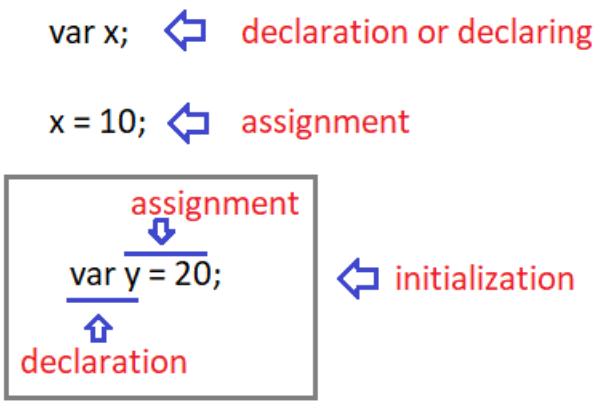
- Storage locations in memory where we can store a value and use it as a part of any expression.
- JavaScript variables are declared by using
  - var
  - let
  - const

Syntax:

```
var variableName;
```

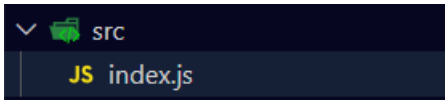
var

- It defines as a function scope variable.
- You can declare and access from any another block.
- var allows declaring or declaration, initialization and assignment.



- var allows shadowing.
- var allows hoisting.

Create a JS file in src folder as like below.



index.js

```
function f1() {
   var x;  //declaration
   x = 10; //assignment
   if (x == 10) {
      var y = 20; //initialization
      var y = 30; //shadowing
   }
   z = 40; //hoisting
   console.log("X - " + x + ", Y - " + y + ", z - " + z);
   var z;
}

f1();
```

Then open terminal and run the below command to see the output.

```
....\src> node index.js
```

Note: Best tutorial For JavaScript [Link].

let

- It is used to define block scope variable.
- Block scope variable is accessible to its inner blocks.
- let allows declaring, assignment, initialization.
- let will not allow shadowing and hoisting.

index.js

```
function f1() {
   let x;  //declaration
   x = 10; //assignment
   if (x == 10) {
      let y = 20; //initialization
   }
   console.log("X - " + x + ", Y - " + y);
}

f1();
```

If you run the you will get error on the terminal.

const

- It also a block scope variable.
- It will not allow assignment and declaration.
- It will allow only initialization.
- It will not allow shadowing and hoisting.
- const is not readOnly.

Data Types

- Data types define the data structure.
- DS defines type and its range.
- JavaScript datatypes are 2 types,
    - Primitives
    - Non-Primitives

Primitive types – These are immutable, they can't change the structure.

- **number**
    - signed -1
    - unsigned 1
    - float 34.33
    - double 356.446

- decimal 2350
- exponent $2e^3$ [2000]
- binary 0b1010 [10]
- hex [0, f]
- octa [0o756]

- **string**
  - single quote 'text'
  - double quote "text"
  - back tick `text` [embedded expression => ${}]

  Ex: var msg2 = `Hello! ${username} you will be ${age+1} next year. `;

- **boolean**
  - true
  - false

- **null** - If value is not supplied dynamically into any reference at runtime, then it returns null.

- **undefined** - If value is not supplied dynamically into any reference at compile time, then it returns undefined.

Non-Primitive types – These are mutable, they can change the structure.

- **Array**
  - Reduce overhead.
  - Reduce complexity.
  - JS array can handle various types of values.
  - JS array size can change dynamically.
  - It supports "de-structuring".

Syntax:

```
var products = [];
var products = new Array ();
```

Ex:

```
var values = ["Raja", 10, true, ["hyd", "delhi"], function()
{console.log("hello!")}];
var [name, is, stock, cities, hello] = values; //de-structuring
console.log(cities);
hello();
```

- Array methods
  - Reading - toString(), join(), slice(), map(), for .. in, for .. of
  - Adding - push(), unshift(), splice()
  - Removing - pop(), shift(), splice()
  - Sorting - sort(), reverse()
  - Searching - find(), filter()

- **Object**
  - It is used to keep all related data and logic together.
  - It is key and value collection; key must be only String type.
  - JSON refers to data only.
  - An object can have data and function also.

    Syntax:

```
var tv = {
        “name”: “Samsung TV”,
        “price”: 45000.44,
        “cities”: [“Delhi”, “Hyd”],
        “rating”: {rate: 4.2, count: 5609},
        “qty”: 2
        “total”: function() {
                return this.qty * this.price;
        }
}
```

  - Issues with Object,
    - It is key and value collection; key must be only string type.

```
{
        “key”: any
}
```

    - It is slow in access.
    - You need explicit iterators to read the keys and values like for ... in, for ... of.
    - It required explicit operators to manipulate like delete, in.
    - No size for object.
    - To solve these issue use Map.

- **Map**
  - It is a key and value collection.
  - Key can be any type.

- It provides implicit iterators.
- It is fast.
- It provides implicit methods to manipulate.
- It is not for structured data.

Syntax:

```
var ref = new Map();
```

- Methods
  - set(), get(), keys(), values(), entries(), delete(), has(), clear().

Note: The major issue with structured data in JavaScript is unique identification.

- **Symbol**
  - It is a primitive type.
  - It is immutable type.
  - It is hidden from iterator.
  - It is implicitly used for iterators.

Syntax:

```
const ref = Symbol("description");
{
    [ref]: value
}
```

Date Types

- Date is defined by using Date()

Syntax:

```
var mfd = new Date("YY-MM-DD Hr:Min:Sec.MilliSec");
```

- Methods
  - getHours(), getMinutes(), getSeconds(), getMilliSeconds(), getDate(), getDay(), getMonth(), getFullYear(), toLocaleDateString(), toLocaleTimeString().

Regular Expression

- It defines a pattern to validate the input value.
- Patterns are built by meta characters and quantifiers.

- JavaScript regular expression is defined using “/ /”.
- To Learn Better Regular Expression [Link].

Operators

- Arithmetic Operators [+, -, *, /, %, ++, --, **]
- Comparison Operators [==, ===]
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Special Operators [typeof, instanceof, in, new, void, delete, etc.]

Statements

- Selection statements
  - If – else, switch, case, default
- Looping statements
  - for, while, do while
- Iteration statements
  - for … in, for … of
- Jump statements
  - break, continue, return
- Exception Handling statements
  - try, catch, throw, finally

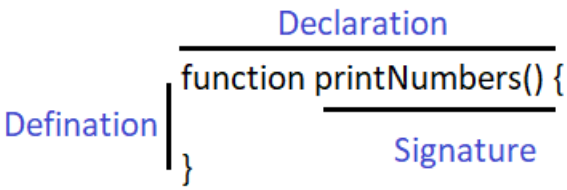
Functions

- A function is used for refactoring the code.
- Refactoring is the process of extracting a set of lines and storing under a reference of file or function.
- So that we can re-use the code.

Syntax:

```
function name() {
        statements;
}
name();
```

- Every function comprises of
  - Declaration – function printNumbers()
  - Signature – printNumbers()
  - Definition – {}



Function Parameters

- Parameters are used to modify function.
- Parameters defined in function declaration are known as “Formal parameters”.
- Formal parameter is just a memory reference name.
- We can store any type of value in parameter.
    - Primitive
    - Non primitive
    - Function
- Every parameter is mandatory.
- Every parameter has order dependency.
- We can define multiple parameters but have to use in same order.
- We can’t ignore any parameter.
- JavaScript will not support function overload.
- JavaScript function parameter can hold a function.
- Functions are defined as arguments to handle “Call Back”.
- Call Back is the process of execution function according to state and situation.
- ECMA standards define maximum 1024 parameters are allowed.
- ES5 > allows to use “Rest Parameter”.

- **Rest Parameter**
    - A single Rest parameter can allow multiple arguments.
    - It is defined by using “…paramName”.
    - Every function can have only one rest parameter.
    - Rest parameter must be the last parameter in formal list.

Syntax:

```
function detasils(…productDetails) {
        ……….
}
```

- **Spread Technique**
    - It is an iterator implicitly to read values from an array and pass as

arguments into a function.

- JavaScript ES6 introduced spread technique.
- Rest is about parameters and spread is about arguments.

Syntax:

```
function sum(x, y, z) {
        return x + y + z;
}
const numbers = [1, 2, 3];
console.log(sum(...numbers));
```

Function Return

- Function can be void type.
- Void is a return type. That means it not returns nothing.
- Function can be discarded with void, then the function will return something.
- You can discard void by using return.
- If method return type is void then the memory is deleted, if we discard the void then memory, is parent.
- A function can return any types of data.

```
function sum(a, b) {
    var c = a + b;
}
```

```
function sum(a, b) {
    return a + b;  //terminated from her
}

sum(10+30)  //Here is work as function and
            Storing the value also
            in the function memory.
```

Q. Why we need a function with return?

Ans. By default, every function is void type; after performing the functionality it cleans up the memory and deletes, but if you want any function to keep it memory and use the memory for handling the data of that function then better to use function return.

Q. Can a void function have return keyword?

Ans. Yes, we can. While testing some of functionality is not completed so we don't want to compiler should reach there at that time we have use return.

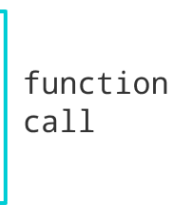
Q. Can function have multiple return?
Ans. Yes, but it will return only one at a time depending on situation.

Function Recursion

- Recursion is a technique calling a function within the context.
- Recursion is a process of calling itself. A function that calls itself is called a recursive function.
- It is used to create batch operations.

Syntax:

```
function recurse() {
    // function code
    recurse();
}

recurse();
```



- A recursive function must have a condition to stop calling itself. Otherwise, the function is called indefinitely.
- Once the condition is met, the function stops calling itself. This is called a base condition.

Note:

- ✓ Multiple operation we group together and executed at a time is called as bath operation.
- ✓ Generally, batch file extension is .bat.
- ✓ This .bat file is not related to the function batch operation.
- ✓ All command we are using those are nothing but .bat file internally.

Arrow Function

- Arrow function expression is a compact alternative to a traditional function expression.
- It is a short hand technique of writing a function.
- Function without name is known as Anonymous function.

Syntax:

- () => expression
- param => expression
- (param) => expression
- (param1, paramN) => expression
- () => {

Statements
}
- param => {
  statements
  }
- (param1, paramN) => {Statements}

- Ex:
```
const materials = ['Hydrogen', 'Helium', 'Lithium', 'Beryllium'];
console.log(materials.map((material) => material.length));
```

Note: Different types of problem we might face while developing frontend.
- ✓ CORS – Cross Origin Resource Sharing
- ✓ XSS – Cross Site Scripting attacks
- ✓ Cross Page Posting
- ✓ Request Forgery
- ✓ Injection attacks

JavaScript OOP

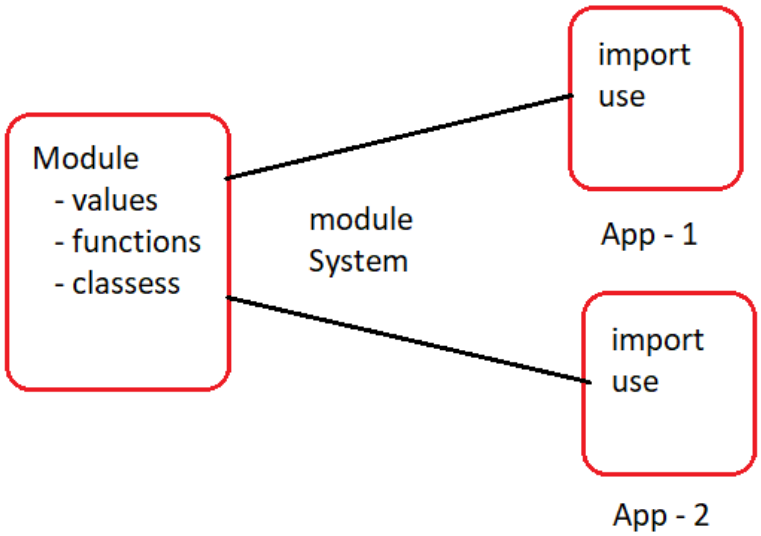
- Different programming systems are
  - POPS (Process Oriented Programming System)
    - C, Pascal, Cobol
  - OBPS (Object Based Programming System)
    - JavaScript, Visual Basic
  - OOPS (Object Oriented Programming System)
    - C++, Java, C#, TypeScript
- JavaScript is not an OOP language.
- It supports only few features or characteristics of OOP.

Features of OOPS:
- Code reusability
- Code extensibility
- Code security
- Loosely coupled
- Dynamic memory

Issue with OOPS:
- Will not support low level features
- Can't directly interact with hardware
- Slow and Heavy

JavaScript Modules

- Module is a set of functions, values and classes.
- Module is required to build a library for project.
- Library enables reusability.



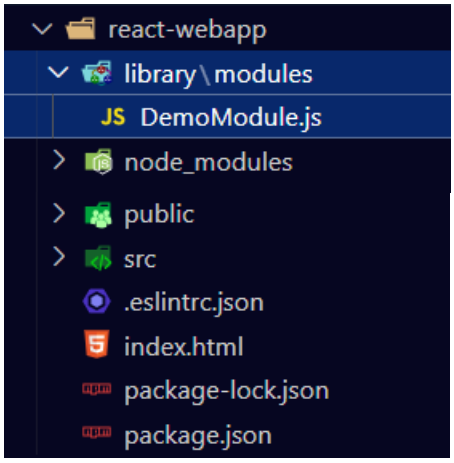
- To use modules in any project you need to setup module system.
- There are various module systems
  - UMD [Universal Module Distribution]
  - AMD [Asynchronous Module Distribution]
  - Common JS
  - Required JS
- JavaScript ES6 > uses module system called “Common JS”.

Creating a Module

- A module is JavaScript file
- i.e.
  - “product.js” -> Module name: product
  - “employee.js” -> Module name: employee
- Every module keeps its members as private, so we can’t access them outside module.
- If we want to access any member outside module then you have to mark it with “export”.

```
export const username = “John”;
export function hello () {

}
```

Directory Structure of react-webapp:



- Create a library folder inside that create a modules folder in that folder create DemoModule.js

DemoModule.js

```
const userName = "John";

function Hello() {
  return `Hello ! ${userName}`;
}

export function WelcomeMsg() {
  return `${hello()} Welcome to my app!`;
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script type="module">
    import { WelcomeMsg } from "./library/modules/DemoModule.js";
    document.querySelector("h1").innerHTML = WelcomeMsg();
  </script>
</head>
<body>
  <h1></h1>
```

```
</body>
</html>
```

- Node module system is different and JavaScript module system is different.
- JavaScript default module system is Common JS and browser also have same module system
- So, if you want to run by using node command that will not work so use HTML file to execute in browser then it will work.
- Every module can have one default export.
- Default export is used for eager loading.

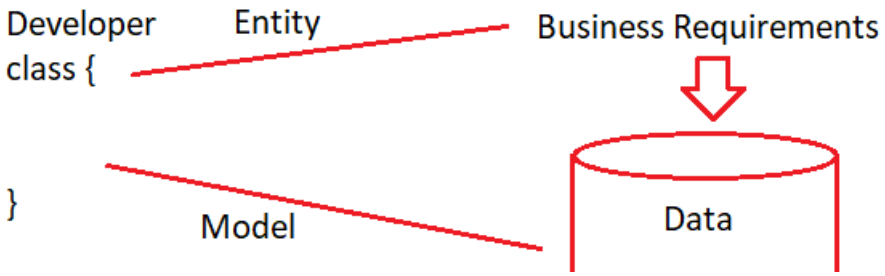
Syntax:

```
export default function name () {

}
import {name} from “Module.js” //invalid
import name from “Module.js” //valid
```

- Every module can have only one default function. We can use default for variables.
- Default can be along with others
  import defaultName, {name} from “Module.js” //valid
- Default must be declared first.
  Import {name}, defaultName from “Module.js” //invalid

JavaScript Classes

- Class is a program templates.
- Template comprises of sample data and logic which you can implement and customize according to requirements.
- Class is used as an Entity, Model, Blue-print.



- If a class is designed to communicate with business requirements, then

it's called as entity.

- If a class is designed to communicate with data requirements, then it's called as model.
- Class can have the following members,
  - Property
  - Accessor
  - Constructor
  - Method

Q. Can we declare a variable and function in class?

Ans. No, we can't declare variable and function in class.

Property

- Property is used to configure data and to store data.
- Property is mutable.
- It can change the structure according to state and situation.

Syntax:

```
class Name {
    property = value;
}
```

- Property can handle any type of value
  - Primitive
  - Non primitive
- Property can change its structure by using "accessors".

Accessor

- Accessors are used to control a property.
- Accessors will give a fine-grained control over property.
- Accessors are 2 types
  - Getter
  - Setter
- Getter is used to read and return value.
  get aliasName() { }
- Setter is used to write a value into property
  set aliasName(newValue) {}

index.html

```
<script>
```

```
    var userName = prompt("Enter Name");
    var role = prompt("Enter Role");
    var productName = prompt("Enter Product Name");

    class Product {
        _productName;

        get ProductName() {
            return this._productName;
        }

        set ProductName (newName) {
            if (role == "admin") {
                this._productName = newName;
            }
            else {
                document.write(`Hello! ${userName}, your role ${role} not authorized
to set Product Name`);
            }
        }
    }

    let obj = new Product();
    obj.ProductName = productName;
    if (obj.ProductName) {
        document.write(`Product Name - ${obj.ProductName}`);
    }
</script>
```

Method

- A method specifies the actions to perform.
- Functionality in class is defined by using method.

Syntax:

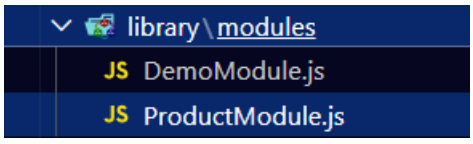
```
function name () {

}
class Product {
    name () {
```

```
        }
    }
```

- All method feature is same as function. Like parameters, rest parameter, return, recursion, arrow techniques.
- Method is mutable.

index.html

```
<script>
  class Product {
    _productName = "Samsung TV";
    _price = 45000.44;
    _qty = 2;
    _rating = {
      vendorRating: {rate: 4.3, count: 40},
      customerRating: {rate: 3.5, count: 5700}
    };

    get CustomerRating() {
      return this._rating.customerRating.rate;
    }

    total() {
      return this._qty * this._price;
    }

    print() {
      document.write(`Name - ${this._productName}<br>
                  Price - ${this._price}<br>
                  Qty - ${this._qty}<br>
                  Total Price - ${this.total()}`);
    }
  }
  let tv = new Product();
  tv.print();
</script>
```



- Add a ProductModule.js under library\modules.

ProductModule.js

```
export class Product {
   _productName = "Samsung TV";
   _price = 45000.44;
   _qty = 2;
   _rating = {
      vendorRating: {rate: 4.3, count: 40},
      customerRating: {rate: 3.5, count: 5700}
   };

   get CustomerRating() {
      return this._rating.customerRating.rate;
   }

   total() {
      return this._qty * this._price;
   }
}

export default class Category {
   _categoryName = "electronics";
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <script type="module">
      import Category, {Product} from "./library/modules/ProductModule.js";
      let product = new Product();
      let category = new Category();
      document.querySelector("p").innerHTML = `Name -
${product._productName}<br>Total - ${product._qty}
                              <br>Category - ${category._categoryName}`;
   </script>
```

```
</head>
<body>
  <p></p>
</body>
</html>
```

Constructor

- Constructor is a special type of sub-route which is responsible for instantiation or creating object.
- Constructor is a software design pattern.
- Every class have a constructor.
- You can write explicit constructor to define actions to perform at the time of creating object.
- In javascript the constructors are anonymous, they don't a name.
- Here constructor can't overload

Syntax:

```
class ClassName {
        constructor () {

        }
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    class Database {
      constructor() {
        alert('Connected to database');
      }

      insert(){
        document.write('Record inserted');
      }
      update() {
```

```
            document.write('Record updated');
        }
        delete() {
            document.write('Record deleted');
        }
    }

    function insertClick () {
        let db = new Database();
        db.insert();
    }

    function updateClick() {
        let db = new Database();
        db.update();
    }

    function deleteClick() {
        let db = new Database();
        db.delete();
    }
  </script>
</head>
<body>
  <button onclick="insertClick()">Insert Record</button>
  <button onclick="updateClick()">Update Record</button>
  <button onclick="deleteClick()">Delete Record</button>
</body>
</html>
```

Inheritance

- It enables reusability and extensibility.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
```

```
<script>
    class HDFC_Version1 {
        _personal = "Personal Banking";
        _nri = "NRI Banking";

        print() {
            document.write(`${this._personal}<br>${this._nri}`);
        }
    }

    class HDFC_Version2 extends HDFC_Version1 {
        _loans = "Personal, Car, Bike Loans";
        print() {
            super.print();
            document.write(`<br>${this._loans}`);
        }
    }

    class HDFC_Version3 extends HDFC_Version2 {
        _agri = "Govt. Schemes";
        print() {
            super.print();
            document.write(`<br>${this._agri}`);
        }
    }
    function installClick() {
        var version = document.getElementById("versions").value;
        var obj;
        switch (version) {
            case "version1":
                obj = new HDFC_Version1();
                obj.print();
                break;
            case "version2":
                obj = new HDFC_Version2();
                obj.print();
                break;
            case "version3":
                obj = new HDFC_Version3();
                obj.print();
```

```
            break;
        }
      }
    </script>
</head>
<body>
  <fieldset>
   <legend>Install Bank App</legend>
   <select name="app" id="versions">
      <option>Select Version</option>
      <option value="version1">HDFC Version 1</option>
      <option value="version2">HDFC Version 2</option>
      <option value="version3">HDFC Version 3</option>
   </select>
   <button onclick="installClick()">Install</button>
  </fieldset>
</body>
</html>
```

- Rules:
  - The derived class constructor must call super class constructor.
  - Newly extended class is called as derived class.
  - Existing class is called as super class.
  - Super class constructor can be called by using super() keyword.
  - This super() must be the first line in derived class constructor.
  - If super class constructor is a parameterized constructor then while calling super() we have to pass the parameters also.

index.html

```
<script>
   class Super {
      constructor() {
         document.write('Super class constructor');
      }
   }

   class Derived extends Super {
      constructor() {
         super();
         document.write('Derived class constructor');
```

```
    }
  }

  let obj = new Derived();
</script>
```

Polymorphism

- Poly mean many and morphos means forms
- Configure a component to work different situations and with different behaviour is referred as polymorphism.

index.html

```
<script>
  class Employee {
    _firstName;
    _lastName;
    _designation;

    print() {
      document.write(`${this._firstName} - ${this._lastName} -
${this._designation}`);
    }
  }

  class Developer extends Employee {
    _firstName = "Raj";
    _lastName = "Kumar";
    _designation = "Software Engineer";
    _role = "Developer Role : Build, Debug, Test, Deploy";
    print() {
      super.print();
      document.write(this._role);
    }
  }

  class Admin extends Employee {
    _firstName = "Kiran";
    _lastName = "Kumar";
    _designation = "Admin";
    _role = "Admin Role : Authorization, Authentication";
```

```
    print(){
      super.print();
      document.write(this._role);
    }
  }

  class Manager extends Employee {
    _firstName = "Sudhanshu";
    _lastName = "Sharma";
    _designation = "Manager";
    _role = "Manager Role: Project Management, Approvals";
    print(){
      super.print();
      document.write(this._role);
    }
  }

  let employees = new Array(new Developer(), new Admin(), new Manager());
  let designation = prompt("Enter Designation");
  for (let employee of employees) {
    if(employee._designation==designation) {
      employee.print();
    }
  }
</script>
```

React

- It is a JavaScript library for building UI.
- React or React JS are same.
- React is used in building UI for SPA [Single page application] and PWA [Progressive Web Application] like mobile.
- PWA are built by using React and React Native [iOS, Android].

Using React in Existing Web application

- We can implement react in any page by using “CDN” links.
- We can implement react in any project by downloading library.
- React in any page requires following libraries
  - React Core Library
  - React DOM Library

- Babel Library [as a compiler]

- React core library enables react in page.
- React DOM library handles virtual DOM.
- Babel library is compiler for JavaScript in react.

Using CDN in any Page

- Link for React and React DOM [Link]
- Bable [Link]

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>

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

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

  <script type="text/babel">
    //ReactDOM.render("Welcome to react",
document.getElementById("container")); //OLD – Up to React 17
    const root =
ReactDOM.createRoot(document.getElementById("container"));
    root.render("Welcome to React");
  </script>
</head>
<body>
  <h1>React Web Application Home</h1>
  <div id="container"></div>
</body>
</html>
```

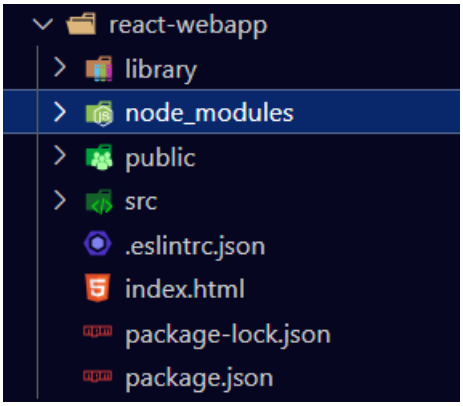
Using Offline Library

- Download and install React and Babel library for your Project.

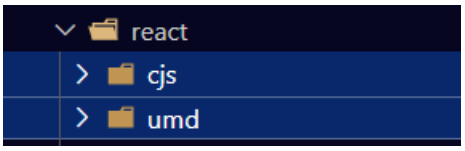
> npm install react --save
> npm install react-dom --save
> npm install @babel/standalone --save
or
> npm install react react-dom @babel/standalone --save

Note:

- ✓ We can install them in any order there is no issue but linking time we should follow the order.
- ✓ We can install them in one attempt by giving space in between as like above.
- ✓ Any module or package those are installed by using package manager will be store in **"node_modules"** folder



- ✓ If you expand **node_modules** go to the **react** folder and expand that you can see "**cjs**" and "**umd**" those are module system. So according to the module system we have add the files.



Create a home.html file in public folder

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <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("Welcome to React HOME");
  </script>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

Note:

- ✓ "render()" of virtual DOM can render a plain text content [RC Data type] or a Component.
- ✓ Component can be a class or a function that renders Layout

Component Rule

- It can be a function or class.
- It must return markup.
- Component name first letter should be capital.

Syntax:

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

Using Arrow function:

```
const Login = ()  => (
        <markup></markup>;
);
```

- You can render any component into Virtual DOM.
  root.render(<Login/>);

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <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">
    //Login Component
    function Login() {
      return (
        <div>
          <h2>Login Page</h2>
          <form action="/login" method="post">
            <label for="username">Username: </label><br />
            <input id="username" type="text" name="username"/><br/>
            <label for="password">Password: </label><br />
            <input id="password" type="password"
name="password"/><br/>
            <button type="submit">Login</button>
          </form>
        </div>
      );
    }
    const root = ReactDOM.createRoot(document.getElementById("root"));
    root.render(<Login/>);
  </script>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

Using Arrow function

home.html

```
  <script type="text/babel">
```

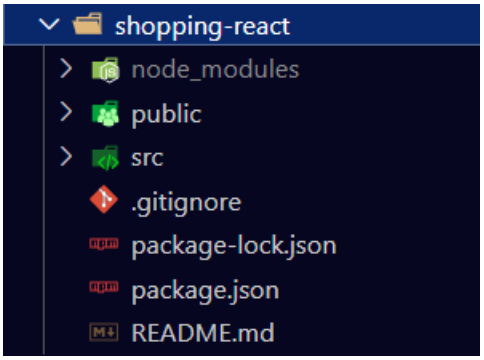
```
    //Login Component
    const Login = () => (
        <div>
          <h2>Login Page</h2>
          <form action="/login" method="post">
            <label for="username">Username: </label><br />
            <input id="username" type="text" name="username"/><br/>
            <label for="password">Password: </label><br />
            <input id="password" type="password"
name="password"/><br/>
            <button type="submit">Login</button>
          </form>
        </div>
    );
    const root = ReactDOM.createRoot(document.getElementById("root"));
    root.render(<Login/>);
  </script>
```

Create a new React application

- Open any location on PC with command prompt and use the below command.
  > npx create-react-app <application_name>

Note: When we create a react application by default it using Javascript, to use Typescript as language while creating the application we need to give just a command.

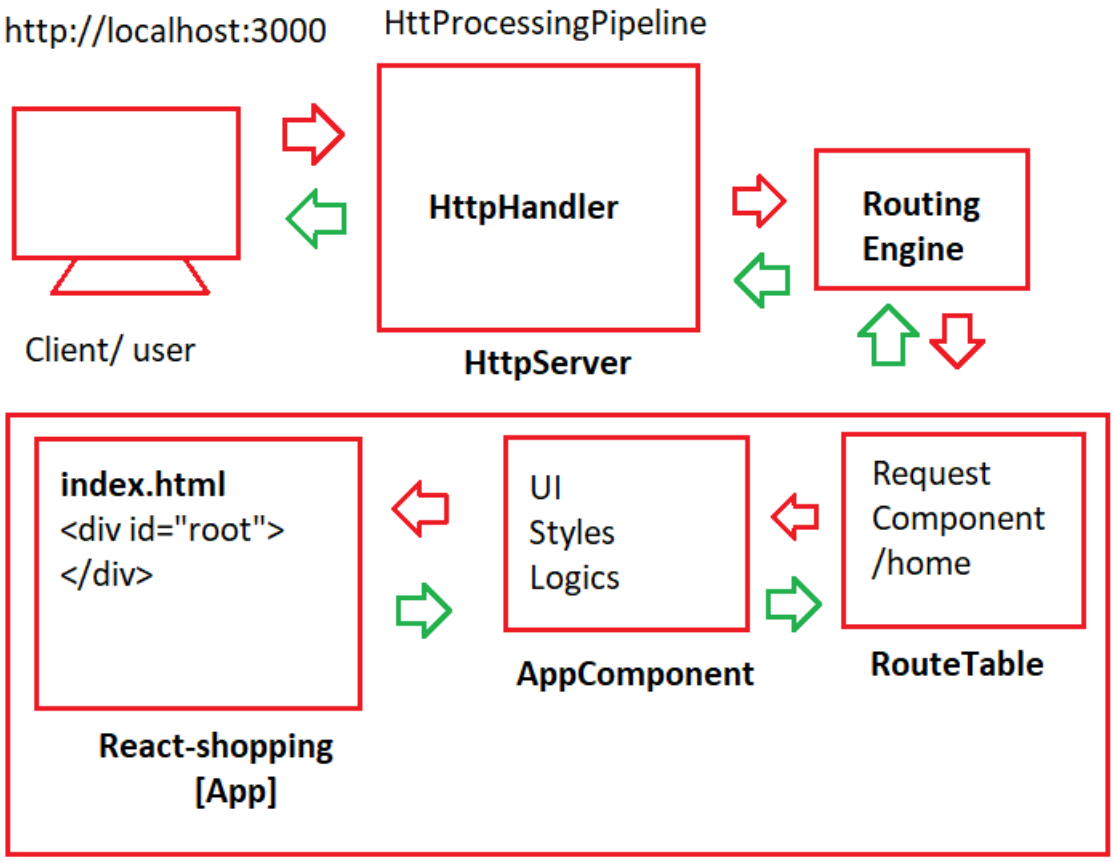
- Open the project folder in VS code.



- Application File system comprises of following files and folder.

File/ Folder	Description
node_modules	It comprises all library files
public	It comprises of static resources [html, image, text, …]
src	It comprises of dynamic resources [js, ts, css, scss, …]
.gitignore	It configures the resources, which are intended not to include into GIT.
package.json	It comprises of project meta data.
package-lock.json	It is used by installer to install the packages locally.
README.md	It is helping document.

- Open terminal and use following command
  > npm start
- Then you can see the default react screen on browser that running with default port no 3000.



Flow of the application

1. Client makes a request from browser http://localhost:3000

2. The request is processed on “Web server – HttpServer”.
3. Request is processed by “HttpHandler” in HttpProcessing pipeline.
4. HttpHandler uses a Routing Engine.
5. Routing Engine verifies the request by using “Routing Table”.
6. Route Table comprises of information about the resources provide by application.
7. If requested is not available in rout table it returns 404: Not Found.
8. If requested path is available then it returns 200: OK.
9. AppComponent is rendered into Root of index.html.
10. index.html is rendered into Browser.

Q. Which component is render by Default?
Ans. AppComponent

Q. Where is the source of AppComponent?
Ans. It is in “src/App.js”

Q. How index.html is able to access the scripts without linking the script files?
Ans. By using “WebPack”.

Q. What is WebPack?
Ans. WebPack is bundler [Link], it is used to bundle all resources of your application. [Scripts, styles, images, etc.]

- It can configure which script to render in HTML page.
- Without linking script file, it can render the scripts.
- It uses “Mapping” and “Model Binding”.

Q. What is source for index.html? from where index.html is accessing script?
Ans. src/index.js

Q. Can we run react application in browser where JavaScript is disabled?
Ans. No

Q. How to know javascript enable or not?
Ans. <noscript>You need to enable JavaScript to run this app.</noscript>. You can check in public/index.html file.

React Components

- Component is a template that comprises of
  - Presentation

- Styles
- Logic

- Presentation is defined using HTML.
- Style is defined by using CSS.
- Logic is defined by using JavaScript or TypeScript.
- Components are building block for React application.
- React components are designed by using
  - Function
  - Class

Function	Class
Light Weight	Heavy
Less Memory	More Memory
Fast in rendering	Slow in rendering
Hard to extend	Easy to extend

Function Component

- Component function can be parameter less or parametrized.
- Every function component must return presentation.

Syntax:

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

- Component uses JSX [Javascript Extension Library].
- JSX will not allow multiple lines without a block.

```
<h2>Head-1</h2>          //invalid
<p>Line-1</p>

<div>
  <h2>Head-1</h2>        //valid
  <p>Line-1</p>
</div>
```

- Complete presentation must return as one fragment.

```
<div> </div>
<> </>
```

<React.Fragment> </React.Frament>

- JSX will not allow void elements. Every element must have end tag.
  <img> //invalid
  <img></img> //valid
  <img/> //valid

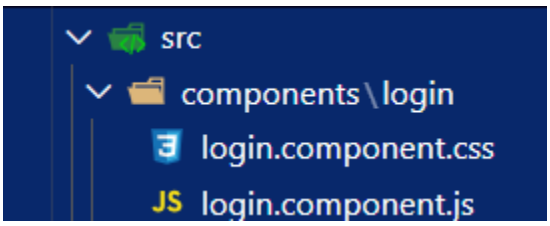
- You can't bind any content to attribute, only properties are allowed.
   //invalid
   //valid
  <img src= "path"> //valid

Designing a Component with Bootstrap styles

1. Install bootstrap and bootstrap-icons library for project.
   > npm install bootstrap --save
   > npm install bootstrap-icons –save

2. Link bootstrap css and icons css to index.js
   import '../node_modules/bootstrap/dist/css/bootstrap.css'
   import '../node_modules/bootstrap-icons/font/bootstrap-icons.css';

3. Add a new folder into "src" by name "components" in that folder add a subfolder "login" on that folder add files as link below



4. Add the following code on the respective files

login.component.js

```
import "./login.component.css";
export function LoginComponent(){
  return (
    <div id="login-form-container" className="container-fluid">
      <form>
        <h2> <span className="bi bi-person-fill"></span> User Name</h2>
        <div className="mb-2">
          <label className="form-label">Email:</label><br />
```

```
            <input type="text" name="email" className="form-
control"/><br/>
          </div>
          <div className="mb-2">
            <label className="form-label">Password:</label><br />
            <input type="password" name="password" className="form-
control"/><br/>
          </div>
          <div className="mb-2">
            <button className="btn btn-primary w-100">Login</button>
          </div>
        </form>
      </div>
    )
}
```

login.component.css

```
form {
  width: 300px;
  border: 1px solid gray;
  border-radius: 10px;
  padding: 20px;
}

#login-form-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 500px;
}
```

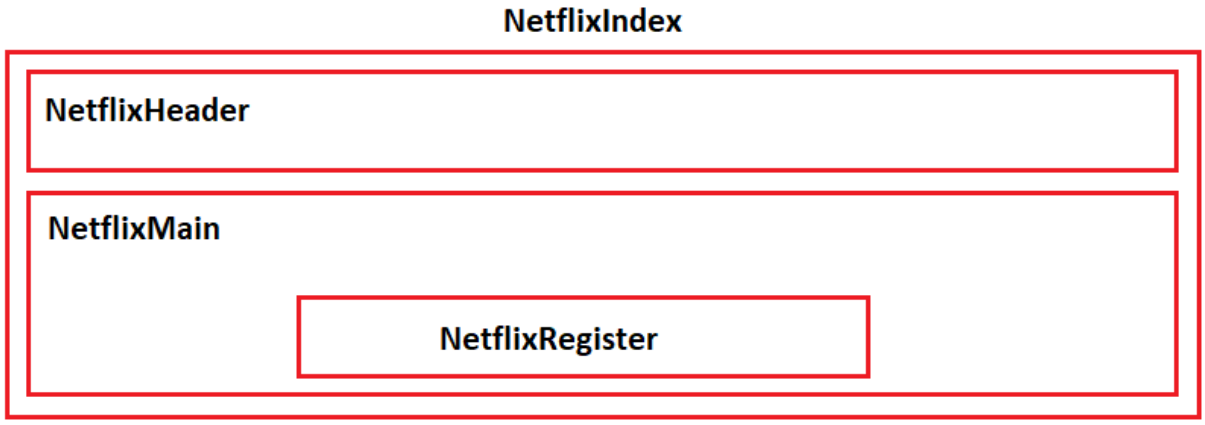
5. Go to index.js in src folder and add the following codes

index.js

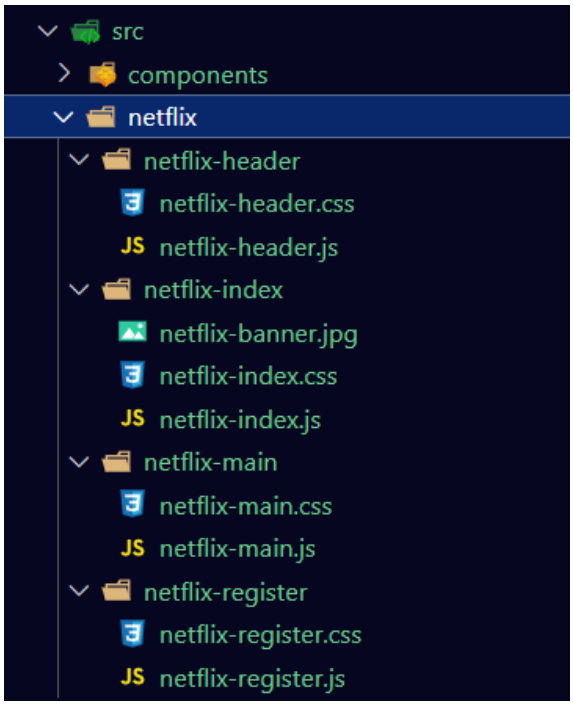
```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import '../node_modules/bootstrap/dist/css/bootstrap.css'
import '../node_modules/bootstrap-icons/font/bootstrap-icons.css';
import { LoginComponent } from './components/login/login.component';
```

```
import reportWebVitals from './reportWebVitals';

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

Netflix component Hierarchy



Create the following “netflix” folder and their subfolder with files



netflix-header.js

```
import "./netflix-header.css";
```

```
export function NetflixHeader() {
 return (
  <div className="header">
   <div className="brand-title">Netflix</div>

   <div className="lang-reg">
    <div className="input-group">
     <span className="bi bi-globe input-group-text"></span>
     <select className="form-select">
      <option selected value="1">
       English
      </option>
      <option value="2">हिन्दी</option>
     </select>
     <div className="ms-4">
      <button className="btn btn-danger">Signin</button>
     </div>
    </div>
   </div>
  </div>
 );
}
```

netflix-header.css

```
.brand-title {
 font-size: 50px;
 font-family: Arial, Helvetica, sans-serif;
 color: #e50914;
 text-transform: uppercase;
 font-weight: bold;
}

.header {
 display: flex;
 justify-content: space-between;
 padding: 30px;
}
```

Netflix Banner Image [Link]

netflix-index.js

```
import "./netflix-index.css";
import { NetflixHeader } from "../netflix-header/netflix-header";
import { NetflixMain } from "../netflix-main/netflix-main";

export function NetflixIndex() {
  return (
    <div id="banner" className="netflix-index">
      <div id="shade">
        <NetflixHeader />
        <main>
          <div className="main">
            <NetflixMain />
          </div>
        </main>
      </div>
    </div>
  );
}
```

netflix-index.css

```
#banner {
  height: 100vh;
  background-image: url("./banner.jpg");
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
}

#shade {
  height: 100vh;
  background-color: rgba(0, 0, 0, 0.6);
}

.main {
  height: 400px;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

netflix-main.js

```
import { NetflixRegister } from "../netflix-register/netflix-register";
import "./netflix-main.css";

export function NetflixMain() {
 return (
  <div className="text-white text-center">
   <h1 className="main-heading">Unlimited movies, TV shows and
more</h1>
   <p className="sub-heading">Starts at ₹149. Cancel anytime.</p>
   <NetflixRegister />
  </div>
 );
}
```

netflix-main.css

```
.main-heading {
 font-size: 50px;
 font-weight: bolder;
 margin-bottom: 20px;
}

.sub-heading {
 font-size: 20px;
 font-weight: bold;
}
```

netflix-register.js

```
export function NetflixRegister() {
 return (
  <div className="mt-5">
   <p className="register-p fs-5">
    Ready to watch? Enter your email to create or restart your membership.
   </p>

   <form className="register-form input-group input-group-lg">
    <input
     type="email"
     className="register-email form-control"
     placeholder="Email address"
    />
```

```
      <button className="register-button btn btn-danger">
        Get Start
        <span className="bi bi-chevron-right"></span>
      </button>
    </form>
  </div>
 );
}
```

index.js

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

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <NetflixIndex />
 </React.StrictMode>
);
```

Note: This much only, those files are not present here leave as default.

Data Binding

- Data binding is the process of accessing data from source and binding to UI.
- Identifying the changes in UI and updating back into source.
- Data binding is handled in JavaScript and jQuery by using
  - DOM methods
  - DOM events.
- React simplified the data binding.
- Data binding is of 2 types
  - One way Binding
  - Two-way Binding
- React supports only One way Binding.

One way Binding

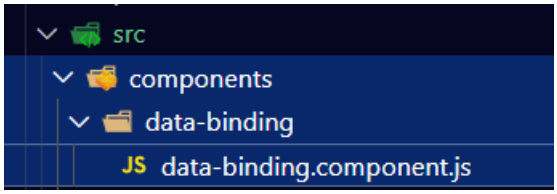
- One way binding is the process of accessing from source and binding to UI. It is unidirectional.
- React uses data binding expression “{}”.
- React dan bind any data without using DOM methods.

Syntax:

```
var username = “John”;
<div> Hello! {username} </div>
```

Ex:

Create the below Folder and File.



data-binding.component.js

```
export function DataBindingComponent() {

  var product = {
    name: "Samsung TV",
    price: 9999.99,
    stock:  true
  };

  return (
    <div className="container-fluid">
      <h2 className="product">Product Details</h2>
      <dl>
        <dt>Name</dt>
        <dd>{product.name}</dd>
        <dt>Price</dt>
        <dd>{product.price}</dd>
        <dt>Stock</dt>
        <dd>{product.stock ? 'In stock' : 'Out of stock'}</dd>
      </dl>
    </div>
  )
}
```

index.js

```
import { DataBindingComponent } from './components/data-binding/data-binding.component';

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

Note: Every iterating item which is generated dynamically must have a unique key in React.

Syntax:

collection.map(item=> <markup key={item}> {item} </markup>);

Array

data-binding.component.js

```
export function DataBindingComponent() {
  var categories = ["Electronics", "Footwear"];

  return (
    <div className="container-fluid">
      <h1>Data Binding - using Array</h1>
      <ol>
        {
          categories.map(category => <li key={category}>{category}</li>)
        }
      </ol>
      <select>
        {
          categories.map(category =>
            <option key={category} value={category}>{category}</option>)
        }
      </select>
      <ul className="list-unstyled">
        {
          categories.map(category =>
            <li key={category} ><input type="checkbox"/>{category}</li>
          )
```

```
          }
        </ul>
        <div>
          {
            categories.map(category=>
                <div key={category} >
                  <button>{category}</button>
                </div>
              )
          }
        </div>
      </div>
  )
}
```

Array of Objects

data-binding.component.js

```
export function DataBindingComponent() {

  var products = [
    {name: "Tv", price: 4500.44, stock: true},
    {name: "Mobile", price: 34500.44, stock: false},
    {name: "Nike Casuals", price: 5200.44, stock: true}
  ];

  return (
    <div className="container-fluid">
      <h1>Product table</h1>
      <table className="table table-hover">
        <thead>
          <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Stock?</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          {
            products.map(product=>
```

```
                    <tr key={product.name}>
                      <td>{product.name}</td>
                      <td>{product.price}</td>
                      <td>{product.stock ? 'Yes' : 'No'}</td>
                      <td>
                        <button className="btn btn-info">
                          <span className="bi bi-eye-fill"></span>
                        </button>
                        <button className="btn btn-warning">
                          <span className="bi bi-pen-fill"></span>
                        </button>
                        <button className="btn btn-danger">
                          <span className="bi bi-trash-fill"></span>
                        </button>
                      </td>
                    </tr>
                  )
                }
              </tbody>
            </table>
          </div>
    )
}
```

Nested Data

data-binding.component.js

```
export function DataBindingComponent() {

    var menus = [
        {category: "Electronics", products: ["TV", "Mobile"]},
        {category: "Footwear", products: ["Nike Casuals", "Lee Boot"]},
    ];

    return (
        <div className="container-fluid">
        <h2 className="data-array">Select category</h2>
            {
                menus.map(item =>
                    <details key={item.category}>
                        <summary>{item.category}</summary>
```

```
            <ul>
              {
                item.products.map(product =>
                  <li key={product}>{product}</li>
                  )
              }
            </ul>
          </details>
      )
    }

    <h1>Shopping  Menu</h1>
    <ol>
      {
        menus.map(menu=>
          <li key={menu.category}>{menu.category}
            <ul>
              {
                menu.products.map(product=>
                  <li key={product}>{product}</li>
                )
              }
            </ul>
          </li>
        )
      }
    </ol>

    <h1>Select Product</h1>
    <select>
      {
        menus.map(item=>
          <optgroup label={item.category} key={item.category}>
            {
              item.products.map(prod =>
                <option value={prod} key={prod}>{prod}</option>
              )
            }
          </optgroup>
        )
```

```
          }
        </select>

      </div>
    )
}
```

Card Design

data-binding.component.js

```
export function DataBindingComponent() {

  var Products = [
    {Name: "Camera", picture:"camera.jpeg", Price:34000, Stock:true},
    {Name: "Headphone", picture:"headphone.jpeg", Price:5000, Stock:false},
    {Name: "Watch", picture:"watch.jpg", Price:2800, Stock:true},
    {Name: "Phone", picture:"phone.webp", Price:97000, Stock:true}
  ];

  return (
    <div className="container-fluid">
      <h2 className="text-center text-bg-info">Product cart</h2>

      <div className="d-flex flex-wrap">
        {
          Products.map(product=>
            <div className="card m-2 p-2 w-25" key={product.Name}>
              <img src={product.picture} className="card-img-top"
alt={product.Name} />

              <div className="card-body">
                <h5 className="card-title">{product.Name}</h5>
                <p className="card-text">
                  Price : {product.Price}<br/>
                  In stock ? {product.Stock?"Yes":"No"}
                </p>
                <button class="btn btn-primary">Buy now</button>
              </div>

            </div>
          )
```

```
        }
      </div>
    </div>
  )
}
```

State

- Variables are immutable, so don't use variables for storing data in react.
- Variables we are using internally for immutable operations.
- Always recommended to use "State".
- Every web application uses protocol "http | https".
- Http is a state less protocol.
- Http use the mechanism "go-get-forget".
    - Go – Establish connection with server.
    - Get – Get Response from Server
    - Forget – Clean up response details.
- State less nature is good for server as is manages memory well.
- State less nature is a drawback for client if he needs continuous operations.
- Web application use various state management techniques
    - Query String
    - Cookie
    - Session
    - Application, etc.
- React implicitly provides state to manage data.
- React version up to 17 state is available only for class components.
- React 18 + version provide state for function components.
- React 18 uses "useState()" hook [method] for function components.

Note:

- ✓ Which techniques we are using in react that is not enough to manage the states in react. So, for that reason react peoples are using an extra third-party library that is call Redux.
- ✓ It is always recommended to maintain your data in React state.

- State in function component is defined by using a hook "useState".
- useState is Generic type.
- State in React can handle any type of Primitive and Non-Primitive data.
- It is open for any type and also restricts specific data type based on the value type initialized.

Syntax:

import {userState} from “react”;

- userState hook requires a getter and setter [accessor].
- Getter can access value stored in state.
- Setter can set value into state.

Syntax:

cons {getter, setter} = useState(initialValue);
cons [getter, setter] = useState(initialValue);

Note: Any React 18 hook can’t be used in class components.

Q. Why state is configured with const?
Ans. You can’t use state without initialization. State must be ready before you start using component. Component is initialized with state.

Example to interact with state for one way binding
data-binding.component.js

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

export function DataBindingComponent() {

  const [userName] = useState("John");

  return (
    <div className="container-fluid">
     <h2>Data Binding</h2>
     Use Name : <input type="text" value={userName}/>
     <p>
      Hello! {userName}
     </p>

    </div>
  )
}
```

Using Array
data-binding.component.js

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

```
export function DataBindingComponent() {

  const [categories] = useState(["electronics", "footweat"]);
  return (
    <div className="container-fluid">
     <ol>
      {
         categories.map(category=>
           <li key={category}>{category}</li>
         )
      }
     </ol>

    </div>
  )
}
```

Two-way Binding

- React doesn't supports two-way binding.
- You have to implements explicitly.
- Two-way binding required events in React.
- It is the process of identifying the changes in UI and updating the changes in to data source.

data-binding.component.js

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

export function DataBindingComponent() {

  const [userName, setUserName] = useState("john");

  function handleUseName(e) {
    setUserName(e.target.value)
  }

  return (
    <div className="container-fluid">
      <h2>Register User</h2>
      <dl>
```

```
            <dt><label htmlFor="username">Name: </label></dt>
            <dd><input type="text" id="username"
onKeyUp={handleUseName}/></dd>
          </dl>
          <p>Hello ! {userName}</p>
        </div>
    )
}
```

data-binding.component.js

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

export function DataBindingComponent() {
    const [name, setName] = useState("");
    const [price, setPrice] = useState(0);
    const [city, setCity] = useState("");
    const [stock, setStock] = useState(false);

    function nameChange(e){
        setName(e.target.value);
    }

    function priceChange(e){
        setPrice(Number(e.target.value));
    }

    function cityChange(e){
        setCity(e.target.value);
    }

    function stockChange(e) {
        setStock(e.target.checked);
    }

    return (
        <div className="container-fluid">
          <div className="row">
            <div className="col-3">
              <h1>Register</h1>
              <dl>
```

```
                <dt>Name</dt>
                <dd><input type="text" onChange={nameChange} /></dd>
                <dt> Price</dt>
                <dd><input type="number" onChange={priceChange}/></dd>
                <dt>City</dt>
                <dd>
                  <select onChange={cityChange}>
                    <option>Delhi</option>
                    <option>Hyd</option>
                  </select>
                </dd>
                <dt>Stock</dt>
                <dd className="form-switch">
                  <input type="checkbox" onChange={stockChange}
className="form-check-input"/> Available
                </dd>
              </dl>
            </div>
            <div className="col-9">
              <h1>Details</h1>
              <table>
                <dl>
                  <dt>Name</dt>
                  <dd>{name}</dd>
                  <dt>Price</dt>
                  <dd>{price}</dd>
                  <dt>City</dt>
                  <dd>{city}</dd>
                  <dt>Stock</dt>
                  <dd>{stock==true?"Available":"out of stock"}</dd>
                </dl>
              </table>
            </div>
          </div>
        </div>
    )
}
```

Data From API

- API is “Application programming interface”.

- It is a concept of distributed computing.
- Distributed computing allows 2 different applications running on 2 different machines to share information.
- Two different applications running in two different processes of same machine can share information.
- API main role is to make date reachable to any device and any platform.
- Worlds first API is “Ebay”.
- API have 3 specifications
    - SOAP
    - REST
    - JSON
- SOAP [Service Oriented Architecture Protocol]. So here
    - Consumer send request as XML
    - Provider send response as XML
- REST [Representational State Transfer]
    - Consumer send request as query request
    - Provider send response as XML
- JSON [JavaScript Object Notation]
    - Consumer send request as JSON
    - Provider send response as JSON
- Various distributed computing technologies
    - CORBA
    - DCOM
    - RMI
    - EJB
    - Web Services
    - Remoting

Consuming API in React:

- React can use various JavaScript promises and jQuery library or any 3rd party for consuming API.
- React implicitly don’t have any library for API.
- we can consume the data from API using the following ways.
    - JavaScript Promise - fetch()
    - jQuery Ajax - $getJSON(), $.ajax()
    - 3rd Party
        - Axios - axios()
        - WhatWgFetch

- Storing of API data is done by using “state”: useState().

- We can fetch date from any API and store in a reference of state on various events.
- If we want to fetch the date from API at the time of initialization of component then we have to use a hook call “useEffect()”
- useEffect is a hook that defines action to perform at the time of
    - Mounting a component
    - Unmounting a component

Using JavaScript Fetch Promise

Syntax:

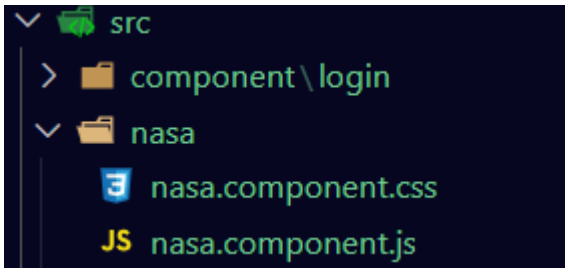
fetch(“URL”).then(response in binary).
then(response in JSON).catch()

Note: JavaScript, jQuery and other third party is using a browser object to communicate with API i.e. XmlHttpRequest.

Note:

- ✓ NASA API as free [Link].
- ✓ Endpoint URL to use in application [Link].

Create the below Folder and File.



index.js

```
import { NasaComponent } from "./nasa/nasa.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <NasaComponent />
  </React.StrictMode>
);
```

nasa.component.js

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

```
export function NasaComponent() {
 const [nasaData, setNasaData] = useState([]);

 useEffect(() => {
  loadPhotos();
 }, []);

 function loadPhotos() {
  fetch(
   "https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY"
  )
   .then((response) => response.json())
   .then((data) => {
    setNasaData(data.photos);
   });
 }

 return (
  <div className="container-container-fluid mt-3">
   <h2>
    Mars Rover Photos <button>Load Data</button>{" "}
   </h2>
   <table className="table table-hover">
    <thead>
     <tr>
      <th>Photo Id</th>
      <th>Preview</th>
      <th>Camera</th>
      <th>Rover</th>
     </tr>
    </thead>
    <tbody>
     {nasaData.map((item) => (
      <tr key={item.id}>
       <td>{item.id}</td>
       <td>
        <img alt="img" src={item.img_src} width="100" height="100" />
       </td>
```

```
          <td>{item.camera.full_name}</td>
          <td>{item.rover.name}</td>
         </tr>
        ))}
      </tbody>
    </table>
  </div>
 );
}
```

Presenting in the form of Card

nasa.component.js

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

export function NasaComponent() {
 const [nasaData, setNasaData] = useState([]);

 function loadPhotos() {
  fetch(
   "https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY"
  ).then((response) => response.json())
   .then((data) => {
    setNasaData(data.photos);
   });
 }

 useEffect(() => {
  loadPhotos();
 }, []);

 return (
  <div className="container-container-fluid mt-3">
   <h2>Mars Rover Photos</h2>

   <div className="d-flex flex-wrap">
    {nasaData.map((item) => (
     <div key={item.id} className="card m-2 p-2">
      <img
```

```
          src={item.img_src}
          className="card-img-top"
          alt={item.camera.name}
         />
         <div className="card-body">
          <dl>
           <dt>Camera</dt>
           <dd>{item.camera.full_name}</dd>
           <dt>Rover</dt>
           <dd>{item.rover.name}</dd>
          </dl>
         </div>
        </div>
      ))}
    </div>
  </div>
 );
}
```

nasa.component.css

```
img.card-img-top {
 width: 268px;
 height: 200px;
}
```

Issues with Fetch:

- It is a Javascript promise which return data in binary.
- You have to explicitly convert into JSON.
- It uses sync technique [blocking].
- Not good in error handing (catch).
- CORS issues [Cross Origin Resource Sharing].
- Security issues [XSS – Cross Site Scripting Attacks].

Using jQuery Ajax

- To overcome form all the above problem we have to use jQuery Ajax.
- jQuery Ajax methods
  - $.getJSON()
  - $.ajax()
  - $.ajaxStart()
  - $.ajaxStop()

- $.ajaxComplete()
- $.ajaxSuccess()
- $.ajaxError()

Features

- Returns data in JSON
- It can handle any type of Data
- No conversions required
- Better in error handling
- Async technique but explicitly
- Handle CORS
- Handle Security issue XSS

Syntax:

```
$.ajax({
        method: "get/post/put/delete",
        url: "",
        success: function() {},
        Error: function() {}
})
```

- Success function of jQuery returns a response object that contains various response details like
  - status – 404, 200
  - statusText – Not found, OK
  - data – data in file or URL
  - header, etc. – request details [method name, host, ….]

Steps to use jQuery Ajax:

Step 1: Download and install jQuery for that we have to use below command.

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

Step 2: Import jQuery

```
Import $ from "jquery";
```

nasa.component.js

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

```
export function NasaComponent() {
 const [nasaData, setNasaData] = useState([]);

 function loadPhotos() {
  $.ajax({
     method: "get",
     url: "https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY",
     success: (data) => {
        setNasaData(data.photos);
     }
  });
 }

 useEffect(() => {
  loadPhotos();
 }, []);

 return (
  <div className="container-container-fluid mt-3">
   <h2>Mars Rover Photos</h2>
   <div className="d-flex flex-wrap">
    {nasaData.map((item) => (
     <div key={item.id} className="card m-2 p-2">
      <img
       src={item.img_src}
       className="card-img-top"
       alt={item.camera.name}
      />
      <div className="card-body">
       <dl>
        <dt>Camera</dt>
        <dd>{item.camera.full_name}</dd>
        <dt>Rover</dt>
        <dd>{item.rover.name}</dd>
       </dl>
      </div>
     </div>
    ))}
   </div>
```

```
  </div>
 );
}
```

Using 3rd Party

- There are so many third parties are there so one of the popular 3rd parties is **Axios.**
- It is light weight and it supports both sync and async techniques.
- It can handle multiple requests simultaneously at the same time.
- Good is error handing and more secured.

Syntax:

a. Single request

```
axios({
        method: “”,
        url: “”,
        data: “”
})
```

b. Multiple requests

```
axios({
        [
                {method: “”, url: “”, data: “”},
                {method: “”, url: “”, data: “”}
        ]
})
```

Note: Learn and read any third-party library go to [npmjs.com].

Steps to use Axios:

Step 1: Download and install Axios for that we have to use below command.

```
> npm install axios -- save
```

Step 2: Import axios

```
Import axios from “axios”;
```

nasa.component.js

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

```
export function NasaComponent() {
 const [nasaData, setNasaData] = useState([]);

 function loadPhotos() {
  axios
   .get(
    "https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEMO_KEY"
   )
   .then((response) => {
    setNasaData(response.data.photos);
   })
   .catch((ex) => {
    console.log(ex);
   });
 }

 useEffect(() => {
  loadPhotos();
 }, []);

 return (
  <div className="container-container-fluid mt-3">
   <h2>Mars Rover Photos</h2>
   <div className="d-flex flex-wrap">
    {nasaData.map((item) => (
     <div key={item.id} className="card m-2 p-2">
      <img
       src={item.img_src}
       className="card-img-top"
       alt={item.camera.name}
      />
      <div className="card-body">
       <dl>
        <dt>Camera</dt>
        <dd>{item.camera.full_name}</dd>
        <dt>Rover</dt>
        <dd>{item.rover.name}</dd>
       </dl>
```

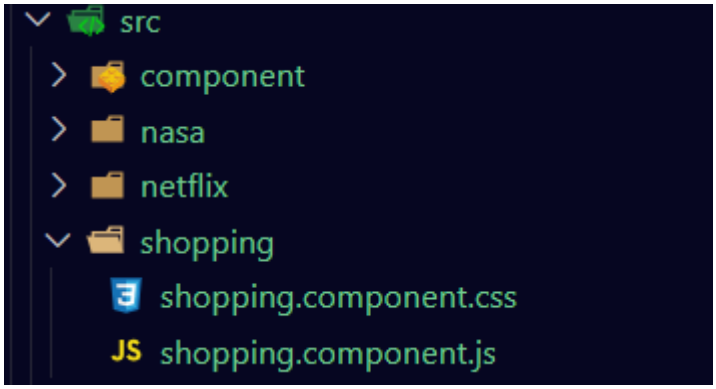
```
        </div>
      </div>
    ))}
  </div>
 </div>
 );
}
```

Shopping ERP with Fake store API

API Link: [Fake Store API]

GET	/products	[] 20 products {}
GET	/products/1	{} specific product
GET	/products/categories	[] categories
GET	/products/category/electronics	[] products of specific category

Create the below Folder and File.



shopping.component.js

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

import "./shopping.component.css";

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

 function loadCategories() {
  axios
   .get("https://fakestoreapi.com/products/categories")
   .then((response) => {
```

```
      response.data.unshift("ALL");
      setCategories(response.data);
    });
  }

  function loadProducts(url) {
    axios.get(url).then((response) => {
      setProducts(response.data);
    });
  }

  function handleCategoryChanged(event) {
    if (event.target.value === "ALL") {
      loadProducts("https://fakestoreapi.com/products");
    } else {
      loadProducts(
        `https://fakestoreapi.com/products/category/${event.target.value}`
      );
    }
  }

  useEffect(() => {
    loadCategories();
    loadProducts("https://fakestoreapi.com/products");
  }, []);

  return (
    <div className="container-fluid">
      <header className="bg-dark text-center text-white p-2">
        <h2>
          <span className="bi bi-cart"> Shopper.</span>
        </h2>
      </header>
      <section className="mt-3 row">
        <nav className="col-2">
          <div>
            <label className="form-label">Select Category</label>
            <div>
              <select onChange={handleCategoryChanged} className="form-
select">
```

```
          {categories.map((category) => (
            <option key={category} value={category}>
              {category.toUpperCase()}
            </option>
          ))}
        </select>
      </div>
    </div>
    <div className="mt-3">
      <label className="form-label">Choose Category</label>
      <div>
        <ul className="list-unstyled">
          {categories.map((category) => (
            <li key={category}>
              <input
                className=""
                onChange={handleCategoryChanged}
                name="category"
                type="radio"
                value={category}
              />
              {category.toUpperCase()}
            </li>
          ))}
        </ul>
      </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
          alt="demo"
          className="card-img-top"
          src={product.image}
          height="200"
        />
        <div className="card-header">
          <p>{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"></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 Cart
            </button>
          </div>
        </div>
      ))}
    </main>
  </section>
 </div>
);

}
```

shopping.component.css

```
main {
 height: 500px;
 overflow: auto;
}

.card-header {
 height: 160px;
}

.card {
 width: 280px;
}
```

index.js

```
import { ShoppingComponent } from "./shopping/shopping.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <ShoppingComponent />
 </React.StrictMode>
);
```

React Style and Class Binding

- Style binding is the process of configuring inline styles for any element dynamically.

  <h1 style= “border: 1px solid red”>
- JSX will not support CSS style attributes directly in any element.
- Styles can bind with HTML elements as JSON object.
- Style binding allows to change the style of any element dynamically.

Syntax:

<h1 style={{propertyName: ‘value’, propertyName: ‘value’}}>

Ex: <h1 style={{color: ‘red’, textAlign: ‘center’}}>

Create the below Folder and File.

styledemo.component.js

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

export function StyleDemoComponent() {
 const [styleObject, setStyleObject] = useState({
  border: "2px solid black",
  boxShadow: "2px 2px 3px black",
  borderRadius: "5px",
```

```
  });

  const [userError, setUserError] = useState("");

  function handleUserName(e) {
    var enteredValue = e.target.value;
    if (enteredValue.charCodeAt(0) >= 65 && enteredValue.charCodeAt(0) <=
90) {
      setStyleObject({
        border: "2px solid green",
        boxShadow: "2px 2px 3px green",
        borderRadius: "5px",
      });
      setUserError("");
    } else {
      setStyleObject({
        border: "2px solid red",
        boxShadow: "2px 2px 3px red",
        borderRadius: "5px",
      });
      setUserError("User name must start with uppercase latter");
    }
  }

  return (
    <div className="container-fluid">
      <h2>Register User</h2>
      <dl>
        <dt>User Name</dt>
        <dd>
          <input
            className=""
            type="text"
            style={styleObject}
            onInput={handleUserName}
          />
        </dd>
        <dd className="text-danger">{userError}</dd>
      </dl>
    </div>
```

```
  );
}
```

index.js

```
import { StyleDemoComponent } from "./component/styled-
demo/styledmo.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <StyleDemoComponent />
  </React.StrictMode>
);
```

Another Example:

styledemo.component.js

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

export function StyleDemoComponent() {
  const [styleObject, setStyleObject] = useState({
    backgroundColor: "",
    textAlign: "",
  });

  function backgroundChange(e) {
    setStyleObject({
      backgroundColor: e.target.value,
      textAlign: styleObject.textAlign,
    });
  }

  function alignChange(e) {
    setStyleObject({
      backgroundColor: styleObject.backgroundColor,
      textAlign: e.target.value,
    });
  }

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

```
    <h2>Choose style</h2>
    <dl>
     <dt>Background Color</dt>
     <dd>
      <select className="color-list" onChange={backgroundChange}>
       <option value="yellow">Yellow</option>
       <option value="green">Green</option>
       <option value="red">Red</option>
      </select>
     </dd>
     <dt>Text Align</dt>
     <dd>
      <select className="align-list" onChange={alignChange}>
       <option value="left">Left</option>
       <option value="center">Center</option>
       <option value="right">Right</option>
      </select>
     </dd>
    </dl>
    <h1 style={styleObject}>Sample Text</h1>
   </div>
  );
}
```

Note: If we have to apply multiple effects then inline style is not great we have to use Class Binding.

Class Binding

- It allows to bind any CSS class to HTML element.

- Class Binding allows to change the class dynamically.

styledemo.component.js

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

import "./styledemo.component.css";

export function StyleDemoComponent() {
  const [theme, setTheme] = useState("");
```

```
 function handleThemeChange(e) {
  if (e.target.checked) {
   setTheme("dark-theme");
  } else {
   setTheme("");
  }
 }
 return (
  <div
   className="container-fluid d-flex justify-content-center align-items-
center"
   style={{ height: "400px" }}
  >
   <form className={theme}>
    <div className="form-switch">
     <input
      className="form-check-input"
      onChange={handleThemeChange}
      type="checkbox"
     />
    </div>

    <h3 className="">
     <span className="bi bi-person-fill"></span> User Login
    </h3>
    <dl>
     <dt>User Name</dt>
     <dd>
      <input className="form-control" type="text" />
     </dd>
     <dt>Password</dt>
     <dd>
      <input className="form-control" type="password" />
     </dd>
    </dl>
    <button className="btn btn-primary w-100">Login</button>
   </form>
  </div>
 );
}
```

styledemo.component.css

```
form {
  width: 300px;
  border: 2px solid gray;
  border-radius: 20px;
  padding: 20px;
}

.dark-theme {
  background-color: rgb(59, 56, 56);
  color: white;
}
```

Event Binding

- Event is a message sent by sender to its subscriber in order to notify the change.
- Event follows a software design pattern called “Observer”.
- Event uses a delegate mechanism [Function Pointer].

Syntax:

function insertClick() { => Subscriber

}

<button onclick = “insertClick()”> => Sender

- Sender sends a notification a subscriber performs the functionality.
- “Observer” is a communication pattern.
- JavaScript events are browser events.
- React is not using DOM. It is using virtual DOM.
- React uses a library called “Synthetic Events”.
- Synthetic events are mapped t Browser event.

Ex: onClick (Synthetic Event) onclick (Browser Event)

- Synthetic events are categories into various group based on their role and responsibility.
  - Mouse Events
    - onMouseOver
    - onMouseOut

- onMouseDown
- onMouseUp
- onMouseMove

- Keyboard Events
  - onKeyUp
  - onKeyDown
  - onKeyPress

- Button Events
  - onClick
  - onDblClick
  - onContextMenu

- Form events
  - onSubmit
  - onReset

- Clipboard Events
  - onCut
  - onCopy
  - onPaste

- Element State Events
  - onChange
  - onFocus
  - onBlur
  - onSelect
  - onSelectStart

- Touch Events
  - onTouchStart
  - onTouchEnd
  - onTouchMove

- Timer Events
  - setTimeout()
  - clearTimeout()
  - setInterval()
  - clearInteval()

Syntax:

```
function handleInsertClick() {

}
<button onClick={handleInsertClick}> Inset </button>

onClick={handleInsertClick}      => Event Handler
onClick                          => Event
```

- Events handler in function component will not have arguments.

Note: JavaScript event handler allows the following arguments.

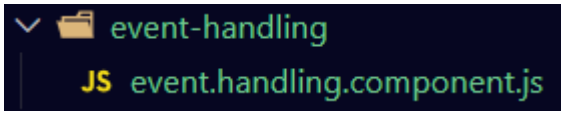
- Default arguments
  - this – sends information about current object.
  - Event – sends information about current event.
- Custom arguments
  - “”, true, [], {}

- React Synthetic event will not allow arguments in functional component.
- React Synthetic event can provide a reference which can access both object and event details.

Syntax:

```
function handleInsertClick(syntheticEvent) {
        syntheticEvent.EventArgs
        syntheticEvent.target.ObjectProperties
}

<button onClick={handleInsertClick}> Insert </button>
```

Create the below Folder and File.



event.handling.component.js

```
export function EventHandlingComponent() {
 function insertClick(syntheticEvent) {
  console.log("Button clicked");
  document.write(`Button Id: ${syntheticEvent.target.id} <br>
    Button Class: ${syntheticEvent.target.className}<br>
```

```
      X Position: ${syntheticEvent.clientX}<br>
      CTRL key: ${syntheticEvent.ctrlKey}`);
  }

  return (
    <div className="container-fluid">
      <h2>Events</h2>
      <button className="btn btn-primary" onClick={insertClick}>
        Insert
      </button>
    </div>
  );
}
```

index.js

```
import { EventHandlingComponent } from "./component/event-
handling/event.handling.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <EventHandlingComponent />
  </React.StrictMode>
);
```

Another Example

event.handling.component.js

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

export function EventHandlingComponent() {
  const [products] = useState([
    { id: 1, name: "TV" },
    { id: 2, name: "Mobile" },
  ]);

  function addClick(e) {
    var result = products.find((product) => product.id == e.target.id);
    document.write(result.name + "<br>" + result.id);
  }

  return (
```

```
  <div className="container-fluid">
    <h2 className="">Products</h2>
    <div className="">
      {products.map((product) => (
        <div key={product.id}>
          <p>{product.name}</p>
          <button id={product.id} onClick={addClick}>
            <span className="bi bi-cart4"></span> Add to Cart
          </button>
        </div>
      ))}
    </div>
  </div>
 );
}
```

Products in Tabular format having Edit and delete option:

event.handling.component.js

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

export function EventHandlingComponent() {
 const [products] = useState([
  { id: 1, name: "TV" },
  { id: 2, name: "Mobile" },
 ]);

 return (
  <div className="container-fluid">
    <h2 className="">Products</h2>
    <table className="table table-hover">
      <thead>
        <tr>
          <th>Product ID</th>
          <th>Product Name</th>
        </tr>
      </thead>
      <tbody>
       {products.map((product) => (
         <tr key={product.id}>
           <td>{product.id}</td>
```

```
          <td>{product.name}</td>
          <td>
            <button className="btn btn-danger">
              <span className="bi bi-trash-fill"></span>
            </button>
            <button className="ms-2 btn btn-warning">
              <span className="bi bi-pencil-fill"></span>
            </button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
 </div>
 );
}
```

Mouse Events

- Mouse events define what action to perform when mouse pointer is over the element, out the element, down the element and mouse is moving all interaction we can mange by the below events
  - onMouseOver
  - onMouseOut
  - onMouseDown
  - onMouseUp
  - onMouseMove

- Most commonly user mouse event arguments are
  - clientX – To get the mouse X position
  - clientY – To get the mouse Y position
  - shiftKey
  - ctrlKey
  - altKey
  - keyCode
  - charCode

event.handling.component.js

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

export function EventHandlingComponent() {
```

```
 const [slide, setSlide] = useState("slide-1.jpg");

 function handleOver() {
  setSlide("slide-2.jpg");
 }

 function handleOut() {
  setSlide("slide-1.jpg");
 }

 return (
  <div className="container-fluid">
   <img
    className="w-100"
    onMouseOver={handleOver}
    onMouseOut={handleOut}
    src={slide}
    alt="sales"
   />
   <p>Move mouse over Image</p>
  </div>
 );
}
```

index.js

```
import { EventHandlingComponent } from "./component/event-
handling/event.handling.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <EventHandlingComponent />
 </React.StrictMode>
);
```

Another Example – onMouseOver & onMouseOut

event.handling.component.js

```
export function EventHandlingComponent() {

 function StartMarquee(e) {
```

```
    e.target.start();
  }

  function StopMarquee(e) {
    e.target.stop();
  }

  return (
    <div className="container-fluid">
      <h2 className="">Mouse Event</h2>
      <marquee onMouseOver={StopMarquee} onMouseOut={StartMarquee}
scrollamount="10">
        <img className="me-2" width="100" height="100" src="asp.net.png" />
        <img className="me-2" width="100" height="100" src="data-
structure.jpg" />
        <img width="100" height="100" src="digital-marketing.png" />
      </marquee>
    </div>
  );
}
```

Another Example – onMouseMove

event.handling.component.js

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

export function EventHandlingComponent() {
  const [styleObject, setStyleObject] = useState({
    position: "",
    top: "",
    left: "",
  });

  function getPosition(e) {
    setStyleObject({
      position: "fixed",
      top: e.clientY + "px",
      left: e.clientX + "px",
    });
  }
```

```
 return (
  <div className="container-fluid" onMouseMove={getPosition}>
   <div style={{ height: "1000px" }}>
    <p>Move Pointer to Test</p>
   </div>
   <img style={styleObject} src="flag.gif" width="50" height="50" />
  </div>
 );
}
```

Image Source: flag.gif

Another Example – onMouseOver

event.handling.component.js

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

export function EventHandlingComponent() {

 const [shirt, setShirt] = useState("shirt.jpg")

 function changeColor(e) {
  switch (e.target.id) {
   case "red":
    setShirt("red-shirt.jpg")
    break;
   case "green":
    setShirt("green-shirt.jpg")
    break;
   case "blue":
    setShirt("blue-shirt.jpg")
    break;
  }
 }

 return (
  <div className="container-fluid">
   <h2>T-Shirt</h2>
   <img alt="t-shirt" src={shirt} width="200" height="200" />
   <div className="mt-3">
```

```
      <span id="red" onMouseOver={changeColor}
style={{backgroundColor:'red', color:'white', cursor:'grab'}} className="me-2
p-2">Red</span>
      <span id="green" onMouseOver={changeColor}
style={{backgroundColor:'green', color:'white', cursor:'grab'}} className="me-
2 p-2">Green</span>
      <span id="blue" onMouseOver={changeColor}
style={{backgroundColor:'blue', color:'white', cursor:'grab'}} className="me-2
p-2">Blue</span>
    </div>
  </div>
 );
}
```

Image Source: shirt.jpg

Keyboard Events

- Key events are indicating what action to perform when an user is keying in the characters.
- We have the following events,
    - onKeyUp
    - onKeyDown
    - onKeyPress

- Most commonly user mouse event arguments are
    - keyCode
    - charCode
    - which
    - shiftKey
    - ctrlKey
    - altKey

Example – onKeyUp & onKeyPress

event.handling.component.js

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

export function EventHandlingComponent() {
 const [users] = useState([
  { user_id: "John" },
  { user_id: "John12" },
```

```
  { user_id: "David" },
]);

const [msg, setMsg] = useState("");
const [errorClass, setErrorClass] = useState("");
const [toggle, setToggle] = useState({ display: "none" });

function verifyUserID(e) {
  console.log("VALUE -  " + e.target.value);
  for (var user of users) {
    if (user.user_id === e.target.value) {
      console.log("MSG  ");
      setMsg("User Name taken - Try again");
      setErrorClass("text-danger");
      break;
    } else {
      setMsg("User Name available");
      setErrorClass("text-success");
    }
  }
}

function verifyPassword(e) {
  if (e.which >= 65 && e.which <= 90) {
    setToggle({ display: "block" });
  }
  else {
    setToggle({ display: "none" });
  }
}

return (
  <div className="container-fluid">
    <h2>Key Events</h2>
    <dl>
      <dt>User ID</dt>
      <dd>
        <input type="text" onKeyUp={verifyUserID} />
      </dd>
      <dd className={errorClass}>{msg}</dd>
```

```
      <dt>Password</dt>
      <dd>
       <input type="password" onKeyPress={verifyPassword} />
      </dd>
      <dd style={toggle} className="text-warning">
       <span className="bi bi-exclamation-triangle"></span> Warning: Caps
ON
      </dd>
     </dl>
    </div>
  );
}
```

Element State Events

- This event has triggered when the state is changed for a particular element for that we can use the below events,
  - onChange
  - onFocus
  - onBlur
  - onSelect
  - onSelectStart

event.handling.component.js

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

export function EventHandlingComponent() {
  const [userName, setUserName] = useState("");
  const [userError, setUserError] = useState("");
  const [cityName, setCityName] = useState("");
  const [cityError, setCityError] = useState("");

  function handleUserName(e) {
    setUserName(e.target.value);
  }

  function submitClick() {
    var validatedFlag = true;
    if (userName === "") {
      validatedFlag = false;
      setUserError("User Name is required");
```

```
  }

  if (cityName === "notcity" || cityName === "") {
    validatedFlag = false;
    setCityError("Please Select Your City");
  }

  if (validatedFlag) {
    document.write("Hello! " + userName + " Your City is " + cityName);
  }
}

function handleUserBlur() {
  if (userName === "") {
    setUserError("User Name is required");
  } else {
    setUserError("");
  }
}

function handleUserKeyUp() {
  if (userName === "") {
    setUserError("User Name is required");
  } else {
    setUserError("");
  }
}

function cityChange(e) {
  if (e.target.value === "notcity") {
    setCityError("Please Select Your City");
  } else {
    setCityName(e.target.value);
    setCityError("");
  }
}

function userFocus() {
  setUserError("User Name is Mandatory");
}
```

```
return (
 <div className="container-fluid">
  <h2>Register User</h2>
  <dl>
   <dt>User Name</dt>
   <dd>
    <input
     type="text"
     onFocus={userFocus}
     onKeyUp={handleUserKeyUp}
     onChange={handleUserName}
     onBlur={handleUserBlur}
    />
   </dd>
   <dd className="text-danger">{userError}</dd>

   <dt>Your City</dt>
   <dd>
    <select onChange={cityChange}>
     <option value="notcity" disabled selected>
      Select City
     </option>
     <option value="Delhi">Delhi</option>
     <option className="Hyderabad">Hyderabad</option>
     <option className="Chennai">Chennai</option>
    </select>
   </dd>
   <dd className="text-danger">{cityError}</dd>
  </dl>
  <button onClick={submitClick} className="btn btn-primary">
   Submit
  </button>
 </div>
);
}
```

Clipboard Events

- Clipboard events handle events that occur when the clipboard is modified.

- onCut
- onCopy
- onPaste

event.handling.component.js

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

export function EventHandlingComponent() {

 const [msg, setMsg] = useState("");

 function handleCut() {
  setMsg("Removed and placed on Clipboard");
 }

 function handleCopy() {
  setMsg("Copied to Clipboard");
 }

 function handlePaste() {
  setMsg("Inserted from Clipboard");
 }

 return (
  <div className="container-fluid">
   <dl>
    <dt>Mobile</dt>
    <dd>
     <input
      type="text"
      onCut={handleCut}
      onCopy={handleCopy}
      onPaste={handlePaste}
     />
    </dd>
    <dd>{msg}</dd>
   </dl>
  </div>
 );
}
```

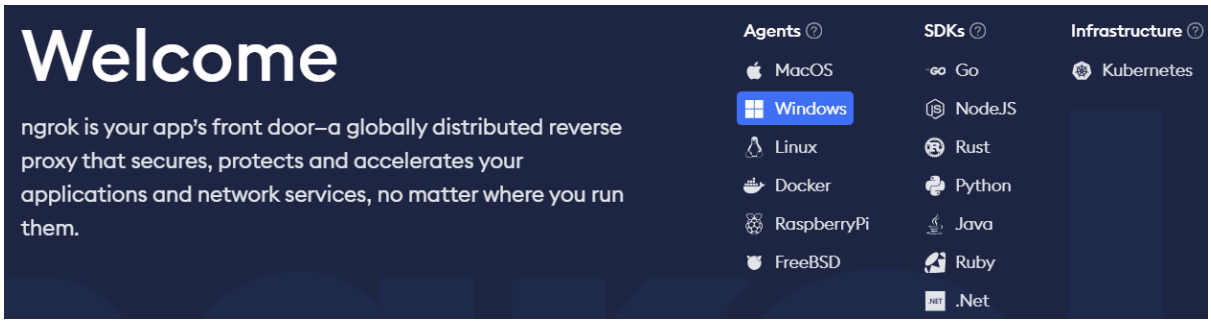
Touch Events

- These events are work only for touch screen devices like smart phone, tab, touch screen laptop and TVs.
- We can't control without touch screen device.
- For that we have the below events
    - onTouchStart
    - onTouchEnd
    - onTouchMove

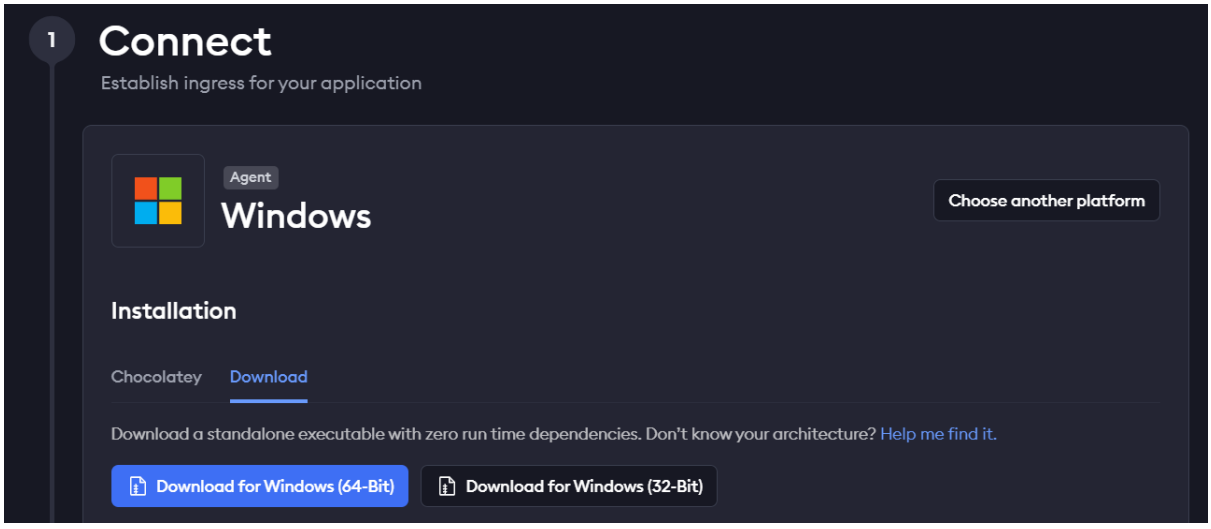
To get your local code access in you Touch screen device or in others device

Step 1: Login or Signup with ngrok.

Step 2: Choose your operation System, In my case i.e. **Windows**

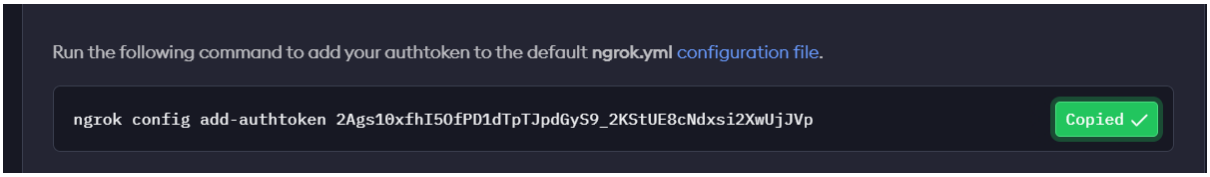


Step 3: Click on **Download for Windows (64-Bit)**

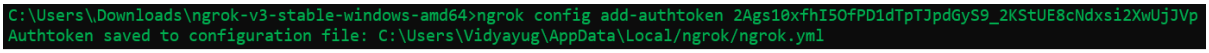


Step 4: Extract that file then double click on **ngrok** exe file parent inside the that then we will get a CMD.

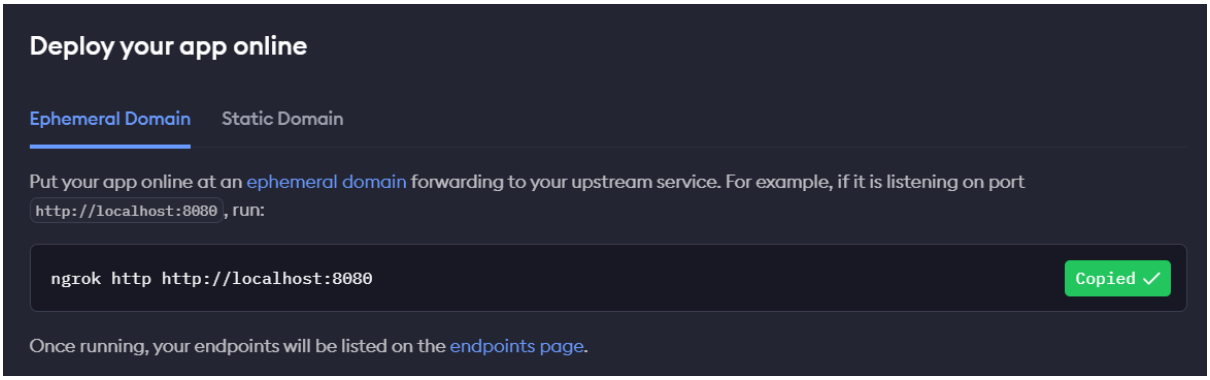
Step 5: Then again go to browser and scroll down then you will a configuration command copy that one.



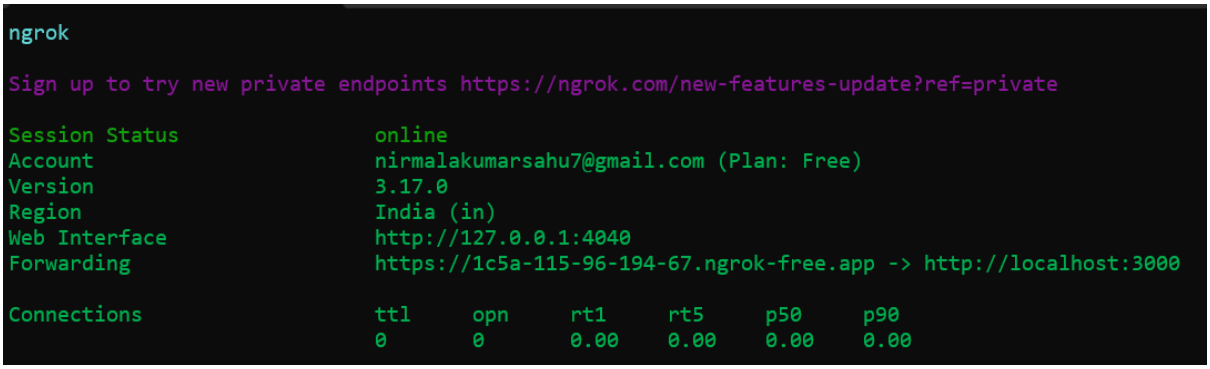
Step 6: Now paste the command in the command prompt and hit enter.



Step 7: Now again go to the browser and check for the Ephemeral Domain command and copy that.



Step 8: Then paste the command in command prompt and change the port number with our React application port number then hit enter. Now you can see a forwarding address has come by using that URL we can access the application in our phone or in other devices. So, that we can test the Touch events.



event.handling.component.js

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

export function EventHandlingComponent() {
 const [styleObject, setStyleObject] = useState({
  position: "",
  left: "",
  top: "",
 });

 function getPosition(e) {
  setStyleObject({
   position: "fixed",
   left: e.touches[0].clientX + "px",
   top: e.touches[0].clientY + "px",
  });
 }

 return (
  <div className="container-fluid">
   <p>Drag and Drop with your Touch</p>
   <img onTouchMove={getPosition} src="shirt.jpg" width="100" height="100"
/>
  </div>
 );
}
```

Example – onTouchStart

event.handling.component.js

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

export function EventHandlingComponent() {
 const [msg, setMsg] = useState("");

 function selectedCourse(e) {
  switch (e.target.id) {
   case "asp":
    setMsg("ASP.Net is used for building server side web applications");
    break;
   case "ds":
```

```
      setMsg("Data Structures is used for storing and organizing data");
      break;
    case "dm":
      setMsg("Digital Marketing is used for promoting products online");
      break;
  }
 }

 return (
  <div className="container-fluid">
   <div className="mt-3">
    <img id="asp" onTouchStart={selectedCourse} src="asp.net.png"
     width="100" height="100" />
    <img id="ds" onTouchStart={selectedCourse} src="data-structure.jpg"
     width="100" height="100" />
    <img id="dm" onTouchStart={selectedCourse} src="digital-marketing.png"
     width="100" height="100" />
   </div>
   <p>{msg}</p>
  </div>
 );
}
```

Example – onTouchStart & onTouchEnd

event.handling.component.js

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

export function EventHandlingComponent() {
 const [styleObject, setStyleObject] = useState({
  with: "",
  height: "",
 });

 function zoomImage(e) {
  setStyleObject({
   width: "400px",
   height: "400px",
  });
 }
```

```
 function zoomOut(e) {
   setStyleObject({
     width: "100px",
     height: "100px",
   });
 }

 return (
   <div className="container-fluid">
     <div onTouchEnd={zoomOut}
       className="d-flex justify-content-center align-items-center"
       style={styleObject}
     >
       <img onTouchStart={zoomImage} src="asp.png" height="100"
width="100" />
     </div>
   </div>
 );
}
```

Form Events

- These events are executed when the form actions happed, to handle those actions we have following events
  - onSubmit
  - onRest

event.handling.component.js

```
export function EventHandlingComponent() {
 function submitClick(e) {
   e.preventDefault();
   alert("Form Submitted");

 }

 return (
   <div className="container-fluid">
     <form onSubmit={submitClick}>
       <dl>
         <dt>User Name</dt>
         <dd>
```

```
        <input type="text" name="username" />
       </dd>
      </dl>
      <button>Submit</button>
    </form>
   </div>
 );
}
```

Note: You can prevent default functionality for any element by using event argument “e.preventDefault()”.

Timer Event

- Timer events are the events that help to execute a piece of code at a specific time interval. For that we can use the following events
  - setTimeOut()
    - It defines the delay time for ant task in memory.
    - Task the removed from memory and set into process.
  - clearTimeOut()
    - It removes any task from memory before it is released into process.
  - setInterval()
    - It loads the task into memory and releases into process without deleting from memory.
    - The task can execute at regular time intervals.
  - clearInterval()
    - It is used to remove the task from memory.

event.handling.component.js

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

export function EventHandlingComponent() {
 const [msg, setMsg] = useState("");

 function message1() {
  setMsg("Hello!");
 }

 function message2() {
  setMsg("How are you?");
```

```
 }

 function message3() {
  setMsg("Welcome to React");
 }

 var m1, m2, m3;
 function handleClick() {
  m1 = setTimeout(message1, 2000);
  m2 = setTimeout(message2, 4000);
  m3 = setTimeout(message3, 6000);
 }

 function cancelClick() {
  clearTimeout(m2);
 }

 return (
  <div className="container-fluid">
   <div className="mt-2">
    <button className="btn btn-primary" onClick={handleClick}>
     Show Message
    </button>
    <button className="ms-2 btn btn-danger" onClick={cancelClick}>
     Cancel Message-2
    </button>
   </div>
   <h2 className="text-center">{msg}</h2>
  </div>
 );
}
```

Example – setInterval & clearInterval

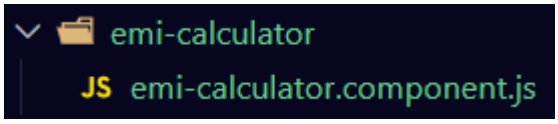
event.handling.component.js

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

export function EventHandlingComponent() {
 const [time, setTime] = useState("");

 function loadTime() {
```

```
  var now = new Date();
  setTime(now.toLocaleTimeString());
 }

 var timer;

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

 function handleStopClick() {
  clearInterval(timer);
 }

 return (
  <div className="container-fluid">
   <div className="text-center">
    <h2 className="text-center mt-4">{time}</h2>
    <button className="me-2" onClick={handleStopClick}>
     <span className="bi bi-stop-fill"></span>
    </button>
    <button>
     <span className="bi bi-play-fill"></span>
    </button>
   </div>
  </div>
 );
}
```

Example – EMI Calculator

Create the below Folder and File.



index.js

```
import { EmiCalculatorComponent } from "./component/emi-calculator/emi-calculator.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
```

```
  <React.StrictMode>
    <EmiCalculatorComponent />
  </React.StrictMode>
);
```

emi-calculator.component.js

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

export function EmiCalculatorComponent() {
  const [amount, setAmount] = useState(50000);
  const [years, setYears] = useState(5);
  const [rate, setRate] = useState(21.75);
  const [emi, setEmi] = useState(0);
  const [emiCalStyle, setEmiCalStyle] = useState({
    display: "none",
  });

  function amountChange(e) {
    setAmount(e.target.value);
  }

  function yearsChange(e) {
    setYears(e.target.value);
  }

  function rateChange(e) {
    setRate(e.target.value);
  }

  function calculateEMI() {
    setEmiCalStyle({
      display: "block",
    });
    var principal = amount;
    var r = rate / (12 * 100);
    var n = years * 12;
    var emi = (principal * r * Math.pow(1 + r, n)) / (Math.pow(1 + r, n) - 1);

    setEmi(Math.round(emi));
  }
```

```
 return (
  <div className="container-fluid">
   <h2 className="">Personal Loan EMI Calculator</h2>
   <div className="row m-3 border border-dark border-2 rounded p-4">
    <div className="col">
     Amount you Need <input type="text" className="rounded"
      size="6" readOnly value={`₹ ${amount}`} />
    </div>
    <div className="col">
     For <input type="text" className="rounded" size="1" readOnly
      value={years} /> Years
    </div>
    <div className="col">
     Interest rate <input type="text" className="rounded" size="2"
      readOnly value={rate} /> %
    </div>
   </div>
   <div className="row m-3 border border-dark border-2 rounded p-4">
    <div className="col">
     &#8377;50,000 <input type="range" min="50000" max="1000000"
      onChange={amountChange} /> &#8377;10,00,000
    </div>
    <div className="col">
     1 Year <input type="range" min="1" max="5" onChange={yearsChange}
/> 5 Year
    </div>
    <div className="col">
     10.75% <input type="range" min="10.75" max="21.75"
onChange={rateChange} /> 21.75%
    </div>
   </div>
   <div className="row m-3">
    <div className="col text-end">
     <button className="btn btn-primary" onClick={calculateEMI}>
      Calculate
     </button>
    </div>
   </div>
   <div className="row mt-3">
```

```
      <h3 className="text-center" style={emiCalStyle}>
        Your monthly installment is &#8377;{emi} for Amount
&#8377;{amount}{" "}
        with Rate {rate}%.
      </h3>
    </div>
  </div>
 );
}
```

Component Properties

- Components a building-blocks for application.
- Component provides reusable and customized template.
- Every component is reusable but can't be customized.
- To allow customization of component, we have to use "Properties".
- Every component can have properties.
- Properties are used to modify the component.

  Ex: Function
  - A function can be parameter less.
  - A parameter less function will do the same task every time.
  - A parametrized function can change according to state and situation.

- Component parameters are called as "Properties".
  <h1 align="center"> align is a property of h1 component

  Syntax:

```
export function Name (props) {
        return (
                <div>{props.ref}</div>
        );
}

<Name ref="value"/>
```

Create the below Folder and File.

- First modify the exiting login component.

login.component.js

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

export function LoginComponent(props) {

 return (
  <div id="login-container" className="container-fluid">
   <form>
    <h2 className="text-center">
     <span className="bi bi-person-fill"></span> {props.title}
    </h2>
    <div className="mb-2">
     <label className="form-label">{props.loginId}</label>
     <div>
      <input type={props.loginIdType} className="form-control" required />
     </div>
    </div>
    <div className="mb-2">
     <label className="form-label">Password</label>
     <div>
      <input type="password" className="form-control" required />
     </div>
    </div>
    <div className="mb-2">
     <button className="btn btn-primary w-100">Login</button>
    </div>
   </form>
  </div>
 );
}
```

properties.component.js

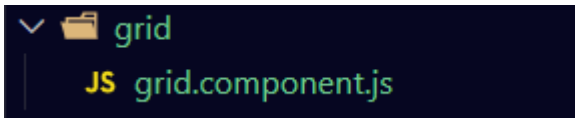
```
import { LoginComponent } from "../login/login.component";

export function PropertiesComponent() {
 return (
  <div className="container-fluid">
   <LoginComponent title="User Login" loginId="User Name"
loginIdType="email" />
```

```
  </div>
 );
}
```

index.js

```
import { PropertiesComponent } from "./component/properties-
demo/properties.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <PropertiesComponent />
 </React.StrictMode>
);
```

Create the below Folder and File.



grid.component.js

```
export function GridComponent(props) {
 return (
  <table style={{ backgroundColor: props.theme }}
   className="table table-hover caption-top">
   <caption>{props.title}</caption>
   <thead>
    <tr>
     {props.fields.map((field) => (
      <th key={field}>{field}</th>
     ))}
    </tr>
   </thead>
   <tbody>
    {props.data.map((item) => (
     <tr key={item.id}>
      {props.fields.map((field) => (
       <td key={field}>{item[field]}</td>
      ))}
     </tr>
    ))}
```

```
    </tbody>
   </table>
  );
}
```

properties.component.js

```
import { useState } from "react";
import { GridComponent } from "../grid/grid.component";

export function PropertiesComponent() {
  const [employeeFields] = useState(["First Name", "Last Name",
"Designation"]);
  const [employeeData] = useState([
   {
    id: 101,
    "First Name": "John",
    "Last Name": "Doe",
    Designation: "Software Engineer",
   },
   {
    id: 102,
    "First Name": "Jane",
    "Last Name": "Doe",
    Designation: "UI Engineer",
   },
   {
    id: 103,
    "First Name": "John",
    "Last Name": "Doe",
    Designation: "Software Engineer",
   },
  ]);

  const [productFields] = useState(["Name", "Price", "Stock", "Rating"]);
  const [productData] = useState([
   {
    id: 1,
    Name: "Product 1",
    Price: 100,
    Stock: 10,
```

```
    Rating: 4.5,
  },
  {
    id: 2,
    Name: "Product 2",
    Price: 200,
    Stock: 20,
    Rating: 4.8,
  },
  {
    id: 3,
    Name: "Product 3",
    Price: 200,
    Stock: 20,
    Rating: 4.8,
  },
 ]);

 return (
  <div className="container-fluid">
    <GridComponent title="Employee Grid" fields={employeeFields}
      data={employeeData} theme="red" />
    <hr style={{ color: "red" }} />
    <GridComponent title="Products Grid" fields={productFields}
      data={productData} theme="green" />
  </div>
 );
}
```

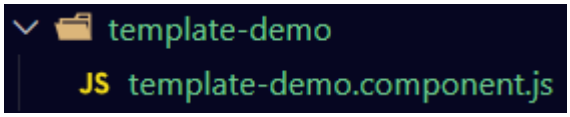
Conditional Rendering

- A component template can change according to state and situation.

Ex:

```
function Name() {
        if(condition)
                return value;
        else
                return value;
}
```

Create the below Folder and File.



template-demo.component.js

```
export function TemplateDemoComponent(props) {
  if (props.layout === "horizontal") {
    return (
      <div>
        <nav className="">
          <span className="me-4">Home</span>
          <span className="me-4">About</span>
          <span className="me-4">Contact</span>
        </nav>
      </div>
    );
  } else if (props.layout === "vertical") {
    return (
      <div>
        <nav className="">
          <div className="mb-3">Home</div>
          <div className="mb-3">About</div>
          <div className="mb-3">Contact</div>
          <div className="mb-3">Blog</div>
        </nav>
      </div>
    );
  }
}
```

properties.component.js

```
import { useState } from "react";
import { TemplateDemoComponent } from "../template-demo/template-
demo.component";

export function PropertiesComponent() {
  const [orientation, setOrientation] = useState("");

  function layoutChange(e) {
    if (e.target.value === "horizontal") {
```

```
    setOrientation("horizontal");
  } else if (e.target.value === "vertical") {
    setOrientation("vertical");
  }
 }

 return (
  <div className="container-fluid">
   <h2>Conditional Rendering</h2>
   <div className="mb-2">
    <select className="" onChange={layoutChange}>
     <option value="-1" selected disabled>
      Select layout
     </option>
     <option value="horizontal">Horizontal</option>
     <option value="vertical">Vertical</option>
    </select>
   </div>
   <TemplateDemoComponent layout={orientation} />
  </div>
 );
}
```

Footer icon changes with layout change

template-demo.component.js

```
export function TemplateDemoComponent(props) {
 if (props.layout === "horizontal") {
  return (
   <div>
    <nav className="">
     <span className="me-4">Facebook</span>
     <span className="me-4">Twitter</span>
     <span className="me-4">Instagram</span>
    </nav>
   </div>
  );
 } else if (props.layout === "vertical") {
  return (
   <div>
    <nav className="">
```

```
      <div className="mb-3 bib bi-facebook"></div>
      <div className="mb-3 bi bi-twitter"></div>
      <div className="mb-3 bi bi-instagram"></div>
      <div className="mb-3 bi bi-linkedin"></div>
    </nav>
   </div>
  );
 }
}
```

Class Components

- We can use Javascript class for designing a component.

  Syntax:

  ```
  export class Name {
  }
  ```

- Every component class gets the behaviour of a component by extending "React.Component" base class.

  Syntax:

  ```
  export class Name extends React.Component {
  }
  ```

Note: Technically methods are void type, functions will have return.

- Class component returns a fragment by using "render()".
- "render()" is of type "React.Node".
- React.Node creates and add element into Virtual DOM.
- "render()" must have return statement.

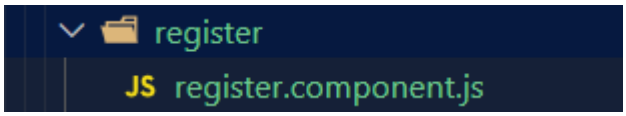
  Syntax:

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

Data Binding in Class component

- Class can store data in property.
- You can access the property within class by using “this” keyword.

Create the below Folder and File.



register.component.js

```
import React from "react";

export class RegisterComponent extends React.Component {
 Title = "User Register";
 render() {
  return (
   <div className="container-fluid">
    <h1>{this.Title}</h1>
   </div>
  );
 }
}
```

index.js

```
import { RegisterComponent } from
"./component/register/register.component";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <RegisterComponent />
 </React.StrictMode>
);
```

- Component class properties are immutable.
- We have to handle data by using state.

Q. Can we use “useState()” hook in class component?
Ans. No, Hooks are only for functions.

Q. Why useState can’t be used in class component?

Ans. useState is a function, class can't have function as class member.

Q. How to handle state in class component?
Ans. "React.Component" base class defined implicit state.

- Every class component in React is defined with built in state. Hence class components are also called as "Stateful components".

register.component.js

```
import React from "react";

export class RegisterComponent extends React.Component {
 constructor() {
  super();
  this.state = {
   title: "User Register",
  };
 }

 render() {
  return (
   <div className="container-fluid">
    <h1>{this.state.title}</h1>
   </div>
  );
 }
}
```

- State must be configured in constructor() of component class.
- State is object type

Syntax:

```
constructor() {
        super();
        this.state = {
                key : value – any type
        }
}

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

register.component.js

```
import React from "react";

export class RegisterComponent extends React.Component {
 constructor() {
  super();
  this.state = {
   title: "Product Details",
   product: {
    name: "TV",
    price: 1000,
    stock: true,
   },
   categories: ["All", "Electronics", "Fashion"],
  };
 }

 render() {
  return (
   <div className="container-fluid">
    <h2>{this.state.title}</h2>
    <dl>
     <dt>Name</dt>
     <dd>{this.state.product.name}</dd>
     <dt>Price</dt>
     <dd>{this.state.product.price}</dd>
     <dt>Stock</dt>
     <dd>
      {this.state.product.stock === true ? "Available" : "Out of stock"}
     </dd>
    </dl>
    <h3>Select Category</h3>
    <select>
     {this.state.categories.map((category) => (
      <option key={category} value={category}>
       {category}
      </option>
     ))}
    </select>
   </div>
```

```
  );
 }
}
```

- If we want to store any data into state then we have to defined by using method “setState()”.

Note: Class and style binding in class is same as function component.

Event Binding in Class component

- All events are same as Synthetic events.
- Event handlers are same.
- Event args are same.
- Class event uses a class method, not function.

Syntax:

```
handleClick() {

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

- Event handler in class component must bind to class in order to use state.
- There are 2 ways

```
constructor() {
  this.handleChange = this.handleChange.bind(this);
}
```

OR

```
<select onChange={this.handleChange.bind(this)}> </select>
```

register.component.js

```
import React from "react";

export class RegisterComponent extends React.Component {
 constructor() {
  super();
  this.state = {
   name: "",
   price: "",
   city: "",
```

```
  };
  this.handleNameChange = this.handleNameChange.bind(this);
  this.handlePriceChange = this.handlePriceChange.bind(this);
  this.handleCityChange = this.handleCityChange.bind(this);
  this.handleRegisterClick = this.handleRegisterClick.bind(this);
}

handleNameChange(event) {
  this.setState({ name: event.target.value });
}

handlePriceChange(event) {
  this.setState({ price: event.target.value });
}

handleCityChange(event) {
  this.setState({ city: event.target.value });
}

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

render() {
  return (
    <div className="container-fluid">
      <div className="row">
        <div className="col-3">
          <dl>
            <dt>Name</dt>
            <dd>
              <input type="text" onChange={this.handleNameChange}
                name="name" className="form-control" />
            </dd>
            <dt>Price</dt>
            <dd>
              <input type="number" onChange={this.handlePriceChange}
                name="price" className="form-control" />
            </dd>
            <dt>City</dt>
```

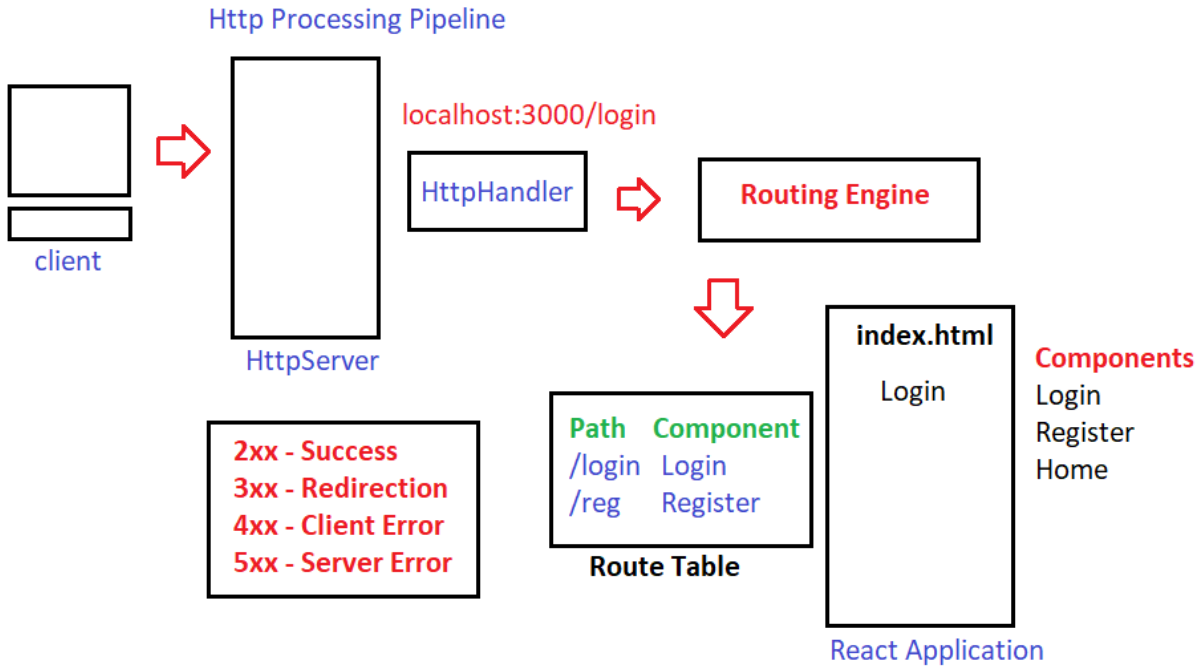
```
          <dd>
           <select name="city"  onChange={this.handleCityChange}
            className="form-select">
            <option value="delhi">Delhi</option>
            <option value="hyderabad">Hyderabad</option>
            <option value="chennai">Chennai</option>
           </select>
          </dd>
         </dl>
         <button className="btn btn-primary w-100"
          onClick={this.handleRegisterClick}> Register </button>
        </div>
        <div className="col-9">
         <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>
    </div>
   );
  }
}
```

Q. If we can't use hooks in class component then how to handle useEffects?
Ans. By using component life cycle hooks.

Component Lifecycle Hooks

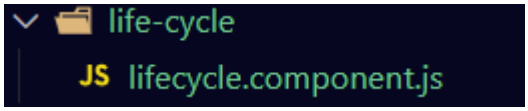
- Every class component has 3 phases
  - Mount
  - Update
  - Unmount
- **Mount:**
  - Component is created
  - Library for component is loaded
  - State is configured

- Values can be initialized into state
- It is managed by a method called “ComponentDidMount()”



- **Update**
  - Style Binding
  - Class Binding
  - Data Binding
  - Event Binding
  - It is managed by a method called “ComponentWillUpdate()”
- **Unmount**
  - All events are unsubscribed
  - Memory allocated for the component is destroyed
  - Unmount phase will fire up when another component is requested

Create the below Folder and File.



lifecycle.component.js

```
import React from "react";

class LoginComponent extends React.Component {
 componentDidMount() {
```

```
  alert("Login Component Requested");
 }

 componentWillUnmount() {
  alert("Login Component Unmounted");
 }

 render() {
  return (
   <div>
       <h2>Login</h2>
   </div>
  );
 }
}

class RegisterComponent extends React.Component {
 componentDidMount() {
  alert("Register Component Requested");
 }

 componentWillUnmount() {
  alert("Register Component Unmounted");
 }

 render() {
  return (
   <div><h2>Register</h2></div>
  );
 }
}

export class LifeCycleComponent extends React.Component {
 constructor() {
  super();
  this.state = {
   component: "",
  };
  this.handleLoginClick = this.handleLoginClick.bind(this);
  this.handleRegisterClick = this.handleRegisterClick.bind(this);
```

```
}

handleLoginClick() {
 this.setState({ component: <LoginComponent /> });
}

handleRegisterClick() {
 this.setState({ component: <RegisterComponent /> });
}

render() {
 return (
  <div>
   <h1>Life Cycle Hooks</h1>
   <div>
    <button onClick={this.handleLoginClick}>Login</button>
    <button onClick={this.handleRegisterClick}>Register</button>
   </div> <hr />
   <div>{this.state.component}</div>
  </div>
 );
}
}
```

Q. How mount and unmount is managed in function component?
Ans. useEffect()

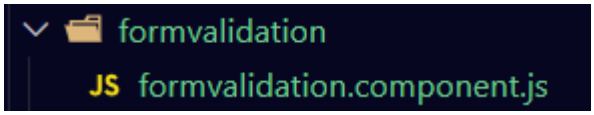
Syntax:

```
useEffect(() => {
        … actions on mount…
        return {
                actions on unmount…
        }
}, [])
```

Form and Validations

- React uses lot of functions and events to handle forms and validation.
- Validation is the process of verifying user input.
- Validation is required to ensure that contractionary and unauthorized data is not get stored into database.

Create the below Folder and File.



formvalidation.component.js

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

export function FormValidationComponent() {
 const [userDetails, setUserDetails] = useState({
  name: "",
  age: 0,
  mobile: "",
  city: "",
 });

 const [nameError, setNameError] = useState("");
 const [ageError, setAgeError] = useState("");
 const [mobileError, setMobileError] = useState("");
 const [cityError, setCityError] = useState("");

 function NameChange(e) {
  setUserDetails((prev) => ({ ...prev, name: e.target.value }));

  if (userDetails.name !== "") {
   setNameError("");
  }
 }

 function AgeChange(e) {
  setUserDetails((prev) => ({ ...prev, age: e.target.valueAsNumber }));
  if (userDetails.age !== "") {
   setAgeError("");
  }

  if (isNaN(userDetails.age)) {
   setAgeError("Age should be a number");
  } else {
   setAgeError("");
  }
 }
```

```
function MobileChange(e) {
  setUserDetails((prev) => ({ ...prev, mobile: e.target.value }));
  if (userDetails.mobile === "") {
    setMobileError("Mobile Number is required");
  } else if (!userDetails.mobile.match(/\+91\d{10}/)) {
    setMobileError("Invalid Mobile : +91 and 10 digits");
  } else {
    setMobileError("");
  }
}

function CityChange(e) {
  setUserDetails((prev) => ({ ...prev, city: e.target.value }));
  if (userDetails.city !== "") {
    setCityError("");
  }
}

function FormSubmit(e) {
  e.preventDefault();
  if (userDetails.name === "") {
    setNameError("User Name is required");
  }

  if (userDetails.age === "" || userDetails.age === 0) {
    setAgeError("Age is required");
  } else if (isNaN(userDetails.age)) {
    setAgeError("Age must be a number");
  }

  if (userDetails.mobile === "") {
    setMobileError("Mobile Number is required");
  } else if (!userDetails.mobile.match(/\+91\d{10}/)) {
    setMobileError("Invalid Mobile : +91 and 10 digits");
  }

  if (userDetails.city === "") {
    setCityError("City is required");
  }
```

```
  alert(JSON.stringify(userDetails));
}

return (
  <div className="container-fluid">
    <div className="row">
      <form className="col-3" onSubmit={FormSubmit}>
        <h2 className=""> Register User</h2>
        <dl>
          <dt><label>Name:</label></dt>
          <dd>
            <input type="text" id="name" name="name"
              className="form-control" onChange={NameChange} />
          </dd>
          <dd className="text-danger">{nameError}</dd>
          <dt><label>Age:</label></dt>
          <dd>
            <input type="text" id="age" name="age"
              className="form-control" onChange={AgeChange} />
          </dd>
          <dd className="text-danger">{ageError}</dd>
          <dt> <label>Mobile:</label> </dt>
          <dd>
            <input type="text" id="mobile" name="mobile"
              className="form-control" onChange={MobileChange} />
          </dd>
          <dd className="text-danger">{mobileError}</dd>
          <dt><label>City:</label></dt>
          <dd>
            <select id="city" name="city" className="form-select"
              onChange={CityChange} >
              <option value="Pune">Pune</option>
              <option value="Mumbai">Mumbai</option>
              <option value="Nagpur">Nagpur</option>
            </select>
          </dd>
          <dd className="text-danger">{cityError}</dd>
        </dl>
        <button className="btn btn-primary form-control">Submit</button>
```

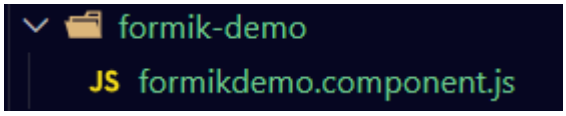
```
    </form>
   </div>
  </div>
 );
}
```

- Developers are not satisfied with the above complex approach so they are depending on 3rd party forms like,
  - Telerik and Kendo Forms
  - Formik
  - DevExpress Forms
    etc.

Formik Forms

- To install formik use the below command
  > npm install formik --save
- To get more forms click on [React JS], [Telerik], [Formik]
- In order to use it we need to import formik base class
  import {useFormik} from "fomik";
- Create a reference for formik
  const formik = useFormik({
    .. formik properties
  })
- Formik properties
  - **initialValues:** configures the form data
  - **handleBlur:** configure actions on element blur
  - **handleChange:** configure actions on element value change
  - **handleSubmit:** defines actions to perform when form submitted
  - **validate:** It uses validation function that returns validation errors
  - **validationSchema:** It uses pre-defined validation services

Create the below Folder and File.



formikdemo.component.js

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

export function FormikDemoComponent() {
 const formik = useFormik({
```

```
  initialValues: {
    name: "",
    age: 0,
    mobile: "",
    city: "",
  },
  onSubmit: (values) => {
    alert(JSON.stringify(values));
  },
  validate: function (values) {},
});

return (
  <div className="container-fluid mt-3">
    <div className="row">
      <form className="col-3" onSubmit={formik.handleSubmit}>
        <h2 className=""> Register User</h2>
        <dl>
          <dt>
            <label>Name:</label>
          </dt>
          <dd>
            <input type="text" id="name" name="name"
              className="form-control" onChange={formik.handleChange} />
          </dd>
          <dt>
            <label>Age:</label>
          </dt>
          <dd>
            <input type="text" id="age" name="age"
              className="form-control" onChange={formik.handleChange} />
          </dd>
          <dt>
            <label>Mobile:</label>
          </dt>
          <dd>
            <input type="text" id="mobile" name="mobile"
              className="form-control" onChange={formik.handleChange} />
          </dd>
          <dt>
```

```
          <label>City:</label>
         </dt>
         <dd>
          <select id="city" name="city" className="form-select"
           onChange={formik.handleChange} >
           <option value="Pune">Pune</option>
           <option value="Mumbai">Mumbai</option>
           <option value="Nagpur">Nagpur</option>
          </select>
         </dd>
        </dl>
        <button className="btn btn-primary form-control">Submit</button>
       </form>
      </div>
     </div>
  );
}
```

- Formik can use validation function or validation schema for verifying values.
- Function is configured and designed by developer.
- Schema is configured by using anther 3rd party called “Yup”.

Validation Function:

```
function VerifyUserDetails(formData) {
  var errors = {};
  logic for verifying form data
  return errors;
}

const formik = useFormik({
  initialValues : {},
  validate : VerifyUserDetails
});

<div>{formik.errors.FieldName}</div>
```

formikdemo.component.js

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

```
export function FormikDemoComponent() {

 function VerifyUserDetails(formData) {
  var errors = {};
  if (!formData.name) {
   errors.name = "Name is required";
  } else if (formData.name.length < 4) {
   errors.name = "Name should be at least 4 characters long";
  } else if (formData.name.length > 10) {
   errors.name = "Name should not be more than 10 characters long";
  }

  if (formData.age === "" || formData.age === 0) {
   errors.age = "Age is required";
  } else if (isNaN(formData.age)) {
   errors.age = "Age should be a number";
  }

  if (!formData.mobile.match(/^((\+91-?)|0)?[0-9]{10}$/)) {
   errors.mobile = "Invalid mobile number";
  }

  if (formData.city === -1 || formData.city === "") {
   errors.city = "City is required";
  }

  return errors;
 }
 const formik = useFormik({
  initialValues: {
   name: "",
   age: 0,
   mobile: "",
   city: "",
  },
  validate: VerifyUserDetails,
  onSubmit: (values) => {
   alert(JSON.stringify(values));
  },
 });
```

```
return (
 <div className="container-fluid mt-3">
  <div className="row">
   <form className="col-3" onSubmit={formik.handleSubmit}>
    <h2 className=""> Register User</h2>
    <dl>
     <dt><label>Name:</label></dt>
     <dd>
      <input type="text" id="name" name="name"
       className="form-control" onChange={formik.handleChange}/>
     </dd>
     <dd className="text-danger">{formik.errors.name}</dd>
     <dt><label>Age:</label></dt>
     <dd>
      <input type="text" id="age" name="age" className="form-control"
       onChange={formik.handleChange} />
     </dd>
     <dd className="text-danger">{formik.errors.age}</dd>
     <dt><label>Mobile:</label></dt>
     <dd>
      <input type="text" id="mobile" name="mobile"
       className="form-control" onChange={formik.handleChange} />
     </dd>
     <dd className="text-danger">{formik.errors.mobile}</dd>
     <dt><label>City:</label></dt>
     <dd>
      <select id="city" name="city" className="form-select"
       onChange={formik.handleChange}>
       <option value="-1" selected disabled> Select your city </option>
       <option value="Pune">Pune</option>
       <option value="Mumbai">Mumbai</option>
       <option value="Nagpur">Nagpur</option>
      </select>
     </dd>
     <dd className="text-danger">{formik.errors.city}</dd>
    </dl>
    <button className="btn btn-primary form-control">Submit</button>
   </form>
  </div>
```

```
  </div>
 );
}
```

Validation Schema using Yup

- Yup is a library for formik to define validation schema.
- It provides built in validation functions for verifying data type and value.
- To install Yup
  > npm install yup --save
- To import
  import required, min, max, pattern from “yup”;
  or
  import * as yup from “yup”;

```
const formik = useFormik({
  initialValue: {},
  validationSchema: yup.object({
        FieldName: yup.required(“Name Required”).min(4).max(10)
                .pattern().string()
  })
});
```

formikdemo.component.js

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

export function FormikDemoComponent() {
 const formik = useFormik({
  initialValues: {
   name: "",
   age: 0,
   mobile: "",
   city: "",
  },
  validationSchema: yup.object({
   name: yup
    .string()
    .required("Name is required")
    .min(4, "Name too short min 4 chars")
    .max(10, "Name too long max 10 chars"),
```

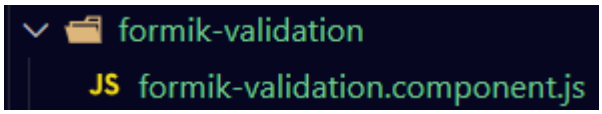
```
    age: yup.number().required("Age is required"),
    mobile: yup
      .string()
      .required("Mobile number is required")
      .matches(/^((\+91-?)|0)?[0-9]{10}$/, "Invalid Mobile Number"),
    city: yup.string().required(),
  }),
  onSubmit: (values) => {
    alert(JSON.stringify(values));
  },
});

return (
  <div className="container-fluid mt-3">
    <div className="row">
      <form className="col-3" onSubmit={formik.handleSubmit}>
        <h2 className=""> Register User</h2>
        <dl>
          <dt><label>Name:</label></dt>
          <dd>
            <input type="text" id="name" name="name"
              className="form-control" onChange={formik.handleChange} />
          </dd>
          <dd className="text-danger">{formik.errors.name}</dd>
          <dt><label>Age:</label></dt>
          <dd>
            <input type="text" id="age" name="age" className="form-control"
              onChange={formik.handleChange} />
          </dd>
          <dd className="text-danger">{formik.errors.age}</dd>
          <dt> <label>Mobile:</label> </dt>
          <dd>
            <input type="text" id="mobile" name="mobile"
              className="form-control" onChange={formik.handleChange} />
          </dd>
          <dd className="text-danger">{formik.errors.mobile}</dd>
          <dt> <label>City:</label> </dt>
          <dd>
            <select id="city" name="city" className="form-select"
              onChange={formik.handleChange} >
```

```
          <option value="-1" selected disabled>
            Select your city
          </option>
          <option value="Pune">Pune</option>
          <option value="Mumbai">Mumbai</option>
          <option value="Nagpur">Nagpur</option>
        </select>
      </dd>
      <dd className="text-danger">{formik.errors.city}</dd>
    </dl>
    <button className="btn btn-primary form-control">Submit</button>
  </form>
 </div>
</div>
);
}
```

Formik validation with HTML elements

- Configure formik with “useFormik” hook.
- Configure validation schema with “yup”.
- Bind the formik values with HTML elements by using {…formik.getFieldProps(“FieldName”)}
- Error message is winded by using {formik.errors.FieldName}

Create the below Folder and File.



formik-validation.component.js

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

export function FormikValidation() {
 const formik = useFormik({
  initialValues: {
   name: "",
   age: 0,
   mobile: "",
  },
```

```
  validationSchema: yup.object({
    name: yup
      .string()
      .required("Name is required")
      .min(4, "Name too short min 4 chars")
      .max(10, "Name too long max 10 chars"),
    age: yup.number().required("Age is required"),
    mobile: yup
      .string()
      .required("Mobile number is required")
      .matches(/^((\+91-?)|0)?[0-9]{10}$/, "Invalid Mobile Number"),
  }),
  onSubmit: (values) => {
    alert(JSON.stringify(values));
  },
});

return (
  <div className="container-fluid">
    <h2>Register User</h2>
    <form onSubmit={formik.handleSubmit}>
      <dl>
        <dt>Name:</dt>
        <dd>
          <input type="text" name="name" {...formik.getFieldProps("name")} />
        </dd>
        <dd className="text-danger">{formik.errors.name}</dd>
        <dt>Age:</dt>
        <dd>
          <input type="text" name="age" {...formik.getFieldProps("age")} />
        </dd>
        <dd className="text-danger">{formik.errors.age}</dd>
        <dt>Mobile:</dt>
        <dd>
          <input type="text" name="mobile" {...formik.getFieldProps("mobile")}/>
        </dd>
        <dd className="text-danger">{formik.errors.mobile}</dd>
      </dl>
      <button>Register</button>
    </form>
```

```
  </div>
 );
}
```

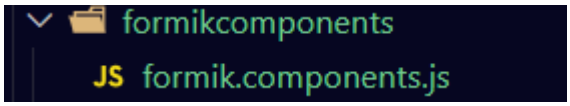
Formik validation with Formik components

- Formik provides pre-defined components for form and validation.
  - <Formik>
  - <Form>
  - <Field>
  - <ErrorMessage>
- Component provides template design and functionality.
- It is clean in separation and error handling.

Syntax:

```
<Formik initialValues={} onSubmit={} validationSchema={}>
        <Form>
                <Field name="FieldName" type="text"></Field>
                <ErrorMessage name="FieldName" ></ErrorMessage>
                <Field as="select">
                        <option></option>
                </Field>
        </Form>
</Formik>
```

Create the below Folder and File.



formik components.js

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

export function FormikComponents() {

 return (
  <div className="container-fluid">
   <h2>Register User</h2>
   <Formik
    initialValues={{
     name: "",
```

```
        age: 0,
        mobile: "",
      }}
      validationSchema={yup.object({
        name: yup
          .string()
          .required("Name is required")
          .min(4, "Name too short min 4 chars")
          .max(10, "Name too long max 10 chars"),
        age: yup
          .number()
          .required("Age is required"),
        mobile: yup
          .string()
          .required("Mobile number is required")
          .matches(/^((\+91-?)|0)?[0-9]{10}$/, "Invalid Mobile Number"),
        city: yup.string().required(),
      })}
      onSubmit={(values) => {
        alert(JSON.stringify(values));
      }}
    >
      <Form>
        <dl>
          <dt>Name:</dt>
          <dd> <Field type="text" name="name" /></dd>
          <dd className="text-danger"><ErrorMessage name="name"/></dd>
          <dt>Age:</dt>
          <dd><Field type="number" name="age" /></dd>
          <dd className="text-danger"><ErrorMessage name="age"/></dd>
          <dt>Mobile:</dt>
          <dd><Field type="text" name="mobile" /></dd>
          <dd className="text-danger"><ErrorMessage name="mobile"/></dd>
        </dl>
        <button className="btn btn-primary">Register</button>
      </Form>
    </Formik>
  </div>
 );
}
```

Formik form properties

- Form properties are used to know the status of form.
- You can access form properties from <Form> component.

Syntax:

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

props.isValid: true/false [all fields are valid]
props.dirty: true/false [modified]

formik components.js

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

export function FormikComponents() {
 return (
  <div className="container-fluid">
   <h2>Register User</h2>
   <Formik
    initialValues={{
     name: "",
     age: 0,
     mobile: "",
    }}
    validationSchema={yup.object({
     name: yup
      .string()
      .required("Name is required")
      .min(4, "Name too short min 4 chars")
      .max(10, "Name too long max 10 chars"),
     age: yup.number().required("Age is required"),
     mobile: yup
      .string()
      .required("Mobile number is required")
      .matches(/^((\+91-?)|0)?[0-9]{10}$/, "Invalid Mobile Number"),
     city: yup.string().required(),
```

```
      })}
      onSubmit={(values) => {
        alert(JSON.stringify(values));
      }}
    >
      {(props) => (
        <Form className={props.isValid ? "bg-success" : "bg-danger"}>
          <dl>
            <dt>Name:</dt>
            <dd><Field type="text" name="name" /></dd>
            <dd className="text-danger"><ErrorMessage name="name" /></dd>
            <dt>Age:</dt>
            <dd><Field type="number" name="age" /></dd>
            <dd className="text-danger"><ErrorMessage name="age" /></dd>
            <dt>Mobile:</dt>
            <dd><Field type="text" name="mobile" /></dd>
            <dd className="text-danger"><ErrorMessage name="mobile"/></dd>
          </dl>
          <button disabled={!props.isValid} className="btn btn-primary">
            Register
          </button>
          <button disabled={!props.dirty} className="btn btn-info">
            Save </button>
        </Form>
      )}
    </Formik>
  </div>
 );
}
```

Routing in React

- Every web application comprises of various techniques.
  - Data Binding
  - Model Binding
  - Even Binding
  - Style Binding
  - Class Binding
  - Caching
  - Routing
  - State Management and etc.

- Routing is a technique used in web applications to create or configure user friendly and SEO friendly URL's.
- A URL without routing
  https://www.amazon.in/electronics.php?category=mobile&brand=realme
- A URL with Routing
  https://www.amazon.in/electronics/mobiles/realme
- Routing is configured both sever side and client side.
- Routing implicitly enables navigation for SPA.
- User can stay on one page and access everything on to the page.
- Routing implicitly uses Ajax.
- Without reloading the complete page new details are added to page.
- React provides routing library which is not as a part of react core library

React version 18	React Routing V6
React Version up to 17	React Routing V5

Note: React 18 is backward compatible, React Router official [Link].

Step to use Route

1. Install react-router-dom library into project
   > npm install react-router-dom --save -D [latest stable – Developer]
   > npm install react-router-dom@latest [latest not stable]
   > npm install react-router-dom@v5 [version 5]

2. React router is not understandable to browser we need **BrowserRouter** in react, which translates the navigation mechanism of virtual DOM into actual DOM
   Import {BrowserRouter} from "react-router-dom";

3. Complete routing configuration must be within **<BrowserRouter>**
   <BrowserRouter>
   ……. Routing configuration ……
   </BrowserRouter>

4. RouteTable is created for application by using **<Routes>**, It is a collection of routes.
   <Routes>
   …. Your routes ….
   </Routes>

5. Every route is defined by using **<Route>** component.

6. Each route defines details about path and content to render.
   <Route path="" element=""></Route>

   Syntax:

```
<BrowserRouter>
        <Routes>
                <Route path element></Route>
                <Route path element></Route>
        </Routes>
</BrowserRouter>
```

7. Routes are requested in browser from URL
   http://localhost:3000/home

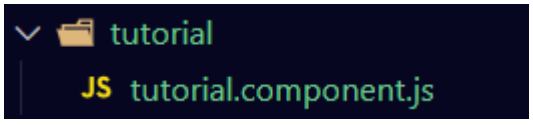
8. We can use <Link> component which can set path into URL
   <Link to="home"></Link>

Q. What is the difference between src and href.
Ans. src is a getter that means it gets the path and href is a setter that means it sets the path.

Note:

- ✓ If a path is set to existing path, then it is called a relative path. It is noting but add the current path to the existing path.
  <Link to="home></Link>
  http://localhost:3000/electronics
  http://localhost:3000/electronics/home
- ✓ If a path is replacing existing path, then it is called as Absolute path. It is nothing but sets a new path.
  <Link to="/home></Link>
  http://localhost:3000/electronics
  http://localhost:3000/home

Ex: Basic Route requested from URL
Create the below Folder and File.



tutorial.component.js

```
import { BrowserRouter, Route, Routes } from "react-router-dom";

export function TutorialComponent() {
 return (
  <div className="container-fluid">
   <h2>Tutorial - Web Technologies</h2>
   <BrowserRouter>
    <Routes>
     <Route path="html" element={
       <div>
        <h3>HTML</h3>
        <p>It is a markup language.</p>
       </div>
      }></Route>
     <Route path="css" element={
       <div>
        <h3>CSS</h3>
        <p>It is a styling language.</p>
       </div>
      }></Route>
    </Routes>
   </BrowserRouter>
  </div>
 );
}
```

Note: if client requests any component which is not in the route table, then you can render alternative using wild card routes.

<Route path="/"><Route>	What to render when user is not requesting any specific path.
<Route path="*"><Route>	What to render when the requested path is not found.

Ex: Route requested from Link

tutorial.component.js

```
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";

export function TutorialComponent() {
```

```
return (
  <div className="container-fluid">
    <h2>Tutorial - Web Technologies</h2>
    <BrowserRouter>
      <section className="row">
        <nav className="col-3">
          <div className="btn-group-vertical">
            <Link to="html">HTML</Link>
            <Link to="css">CSS</Link>
          </div>
        </nav>
        <main className="col-9">
          <Routes>
            <Route path="/"
              element={<h3>Online Tutorial for HTML and CSS</h3>}/>
            <Route path="html" element={
              <div>
                <h3>HTML</h3>
                <p>It is a markup language.</p>
              </div>
            }></Route>
            <Route path="css" element={
              <div>
                <h3>CSS</h3>
                <p>It is a styling language.</p>
              </div>
            }></Route>
            <Route path="*" element={<code>404 : Page not found</code>} >
            </Route>
          </Routes>
        </main>
      </section>
    </BrowserRouter>
  </div>
);
}
```

Note: Every route path can render element, which can be markup or a component that have markup.

tutorial.component.js

```
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";
import { LoginComponent } from "../login/login.component";
import { RegisterComponent } from "../register/register.component";

export function TutorialComponent() {
 return (
  <div className="container-fluid">
   <h2>Tutorial - Web Technologies</h2>
   <BrowserRouter>
    <section className="row">
     <nav className="col-3">
      <div className="btn-group-vertical">
       <Link to="login" className="btn btn-primary mb-2">Login</Link>
       <Link to="register" className="btn btn-primary">Register</Link>
      </div>
     </nav>
     <main className="col-9">
      <Routes>
       <Route path="/" element={<h2>Shopping Home Page</h2>} />
       <Route path="login" element={<LoginComponent />} />
       <Route path="register" element={<RegisterComponent />} />
      </Routes>
     </main>
    </section>
   </BrowserRouter>
  </div>
 );
}
```

Shopper Application

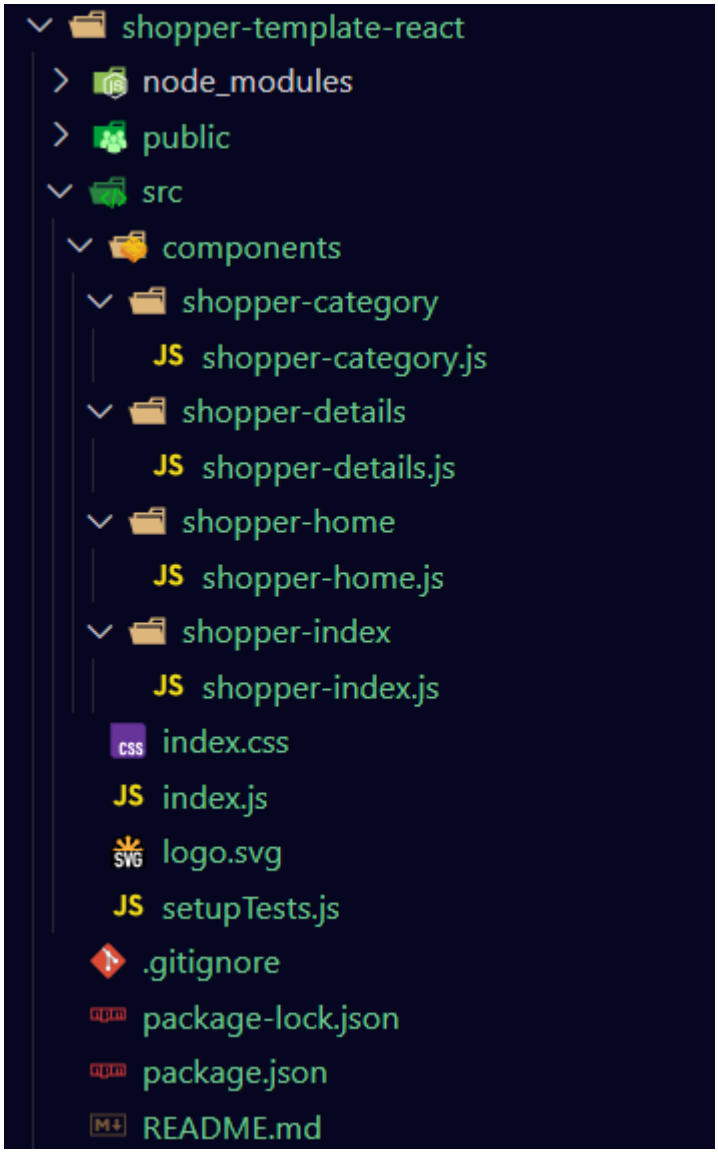
API used: https://fakestoreapi.com/products/

Create a new application and install bellow dependencies

- npx create-react-app shopper-template-react
- npm install bootstrap --save
- npm install bootstrap-icons --save
- npm install jquery -- save
- npm install axios -- save
- npm install formik -- save

- npm install yup -- save
- npm install react-router-dom --save

Create the below Folder and File.



shopper-category.js

```
import axios from "axios";
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import { Link } from "react-router-dom";

export function ShopperCategory() {
  const params = useParams();
  const [products, setProducts] = useState([]);
```

```
 useEffect(() => {
  axios({
   method: "GET",
   url: `https://fakestoreapi.com/products/category/${params.catname}`,
  }).then((response) => {
   setProducts(response.data);
  });
 }, [params.catname]);

 return (
  <div className="container-fluid">
   <h2>Shopper Category {params.catname}</h2>
   <div className="d-flex flex-wrap">
    {products.map((product) => (
     <div className="card m-2 p-2" style={{ width: "200px" }}>
      <img className="card-img-top" src={product.image}
       style={{ height: "150px" }}/>
      <div className="card-header" style={{ height: "150px" }}>
       <p>{product.title}</p>
      </div>
      <div className="card-footer">
       <Link to={"/details/" + product.id} className="btn btn-primary w-100"
       > Details </Link>
      </div>
     </div>
    ))}
   </div>
  </div>
 );
}
```

shopper-details.js

```
import axios from "axios";
import { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";

export function ShopperDetails() {
 const params = useParams();
 const [product, setProduct] = useState({
  id: 0,
```

```
  title: "",
  category: "",
  price: 0,
  rating: {
    rate: 0,
    count: 0,
  },
});

useEffect(() => {
  axios({
    method: "GET",
    url: `https://fakestoreapi.com/products/${params.id}`,
  }).then((response) => {
    setProduct(response.data);
  });
}, []);

return (
  <div className="container-fluid">
    <h2>Details</h2>
    <div className="row">
      <div className="col-3">
        <img src={product.image} width="200" height="200" />
      </div>
      <div className="col-9">
        <dl>
          <dt>Title</dt>
          <dd>{product.title}</dd>
          <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>
          <Link to={"/category/" + product.category}>
            Back to {product.category} </Link>
```

```
        </div>
      </div>
    </div>
  </div>
 );
}
```

shopper-home.js

```
export function ShopperHome() {
 return (
  <div className="container-fluid">
   <div className="d-flex justify-content-between">
    <div>
     <img alt="electronics" src="/images/electronics.jpg"
      style={{ with: "200px", height: "200px" }} />
    </div>
    <div>
     <img alt="mensclothing" src="/images/mensclothing.jpg"
      style={{ with: "200px", height: "200px" }} />
    </div>
    <div>
     <img alt="womensclothing" src="/images/womensclothing.jpg"
      style={{ with: "200px", height: "200px" }} />
    </div>
    <div>
     <img alt="jewelery" src="/images/jewelery.jpg"
      style={{ with: "200px", height: "200px" }} />
    </div>
   </div>
  </div>
 );
}
```

shopper-index.js

```
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";
import { ShopperHome } from "../shopper-home/shopper-home";
import { ShopperCategory } from "../shopper-category/shopper-category";
import { ShopperDetails } from "../shopper-details/shopper-details";

export function ShopperIndex() {
```

```
return (
 <div className="container-fluid">
  <BrowserRouter>
   <header className="d-flex p-2 justify-content-between">
    <h2 className="">Shopper.</h2>
    <nav className="d-flex">
     <div className="me-3">
      <Link className="btn" to="home">Home</Link>
     </div>
     <div className="me-3">
      <Link className="btn" to="category/men's clothing">
       Men's Fashion
      </Link>
     </div>
     <div className="me-3">
      <Link className="btn" to="category/women's clothing">
       Women's Fashion
      </Link>
     </div>
     <div className="me-3">
      <Link className="btn" to="category/jewelery">Jewellery</Link>
     </div>
     <div className="me-3">
      <Link className="btn" to="category/electronics">Electronics</Link>
     </div>
    </nav>
    <div>
     <span className="bi bi-search me-3"></span>
     <span className="bi bi-person me-3"></span>
     <span className="bi bi-heart me-3"></span>
     <span className="bi bi-cart4 me-3"></span>
    </div>
   </header>
   <div className="text-center text-white bg-dark mt-3 p-2">
     Happy Holiday Deals on Everything 
   </div>

   <div className="mt-3">
    <Routes>
     <Route path="/" element={<ShopperHome />}></Route>
```

```
        <Route path="/home" element={<ShopperHome />}></Route>
        <Route path="/category/:catname" element={<ShopperCategory />}>
        </Route>
        <Route path="details/:id" element={<ShopperDetails />}></Route>
       </Routes>
      </div>
    </BrowserRouter>
   </div>
  );
}
```

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import "../node_modules/bootstrap/dist/css/bootstrap.css";
import "../node_modules/bootstrap-icons/font/bootstrap-icons.css";
import { ShopperIndex } from "./components/shopper-index/shopper-index";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <ShopperIndex />
 </React.StrictMode>
);
```

MERN Stack

- M stands for Mongo DB
- E stands for Express Middleware
- R stands for React Front End
- N stands Node JS Server Side

Database:

1. Download and install MongoDB server on your PC
    a. Server
    b. Client [MongoDB Compass]

    [Download Link]

2. Start Server
    Run > service.msc > mongodb > start

3. Open MongoDB compass – GUI tool for managing data

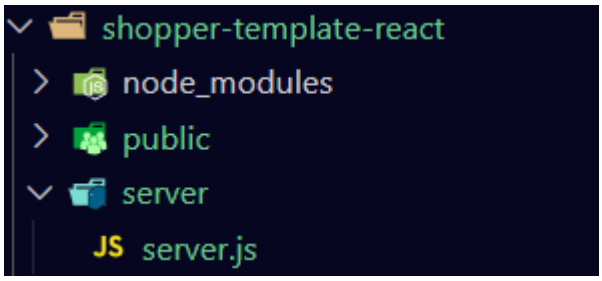
4. Connect with MongoDB server
   Mongodb://127.0.0.1:27017

5. Create a new database
   Database Name: shopper
   Collection Name: users

6. Open users collection and add a new document
   {
    "userId": "101",
    "userName": "John",
    "password": "john@11",
    "email": "john@gmail.com",
    "age": 22,
    "mobile": "+91 9876543210"
   }

Server-Side API

1. In your, react project add a new folder server and a new server side Javascript file server.js



2. Add below code on that file

server.js

```
//import technique for Node JS
var mongoClient = require("mongodb").MongoClient;
var express = require("express");
var cors = require("cors");

var connectionString = "mongodb://127.0.0.1:27017";
```

```
var app = express();
//For POST APIS
app.use(cors());
app.use(
 express.urlencoded({
  extended: true,
 })
);
app.use(express.json());

// to get all the users
app.get("/users", (request, response) => {
 mongoClient.connect(connectionString).then((clientObject) => {
  var db = clientObject.db("shopper");
  var collection = db
   .collection("users")
   .find({})
   .toArray()
   .then((documents) => {
    response.send(documents);
    response.end();
   });
 });
});

// to add user
app.post("/registeruser", (request, response) => {
 var user = {
  userId: request.body.userId,
  userName: request.body.userName,
  password: request.body.password,
  email: request.body.email,
  age: parseInt(request.body.age),
  mobile: request.body.mobile,
 };

 mongoClient.connect(connectionString).then((clientObject) => {
  var database = clientObject.db("shopper");
  database
   .collection("users")
```

```
    .insertOne(user)
    .then((result) => {
      console.log("Record Inserted");
      //response.end();
      response.redirect("/users");
    });
  });
});

app.listen(5000);
console.log("Server started : http://127.0.0.1:5000");
```

3. Install following libraries for sever side scripting
   > npm install mongodb --save
   > npm install express --save
   > npm install cors --save

4. Start API
   …/server> node server.js
   http://127.0.0.1:5000

5. Now we can test using postman
   API Requests

GET	/users	[] of users
POST	/registeruser	submits user details into database

Note: We can test get method in general browser, but we can't test post method in browser for that we need separate web debuggers.

- Fiddler
- Postman
- Swagger

Create the below Folder and File.

shopper-register.js

```
import { Formik, Form, Field, ErrorMessage } from "formik";
import axios from "axios";
import * as yup from "yup";
import { Link, useNavigate } from "react-router-dom";

export function ShopperRegister() {
  const navigate = useNavigate();

  return (
    <div className="container-fluid">
      <h3>Register User</h3>
      <Formik
        initialValues={{
          userId: "",
          userName: "",
          password: "",
          email: "",
          age: "",
          mobile: "",
        }}
        validationSchema={yup.object({
          userId: yup.string().required("User ID is required"),
          userName: yup.string().required("User Name is required"),
          password: yup.string().required("Password is required")
            .matches(/(?=.*[A-Z])\w{4,15}/,
              "Password 4 to 15 characters with at least one upper case letter"
            ),
          email: yup.string().email("Invalid email").required("Email is required"),
          age: yup.number().required("Age is required"),
          mobile: yup.string().required("Mobile is required")
            .matches(/\+91\d{10}/, "Invalid mobile +91 and 10 digits"),
        })}
        onSubmit={(values) => {
          console.log("coming to here = ", values);
          axios({
            method: "POST",
            url: "http://localhost:5000/registeruser",
            data: values,
          }).then(() => {
```

```
        alert("Register SuccessFully..");
        navigate("/login");
      });
    }}
  >
    {
      <Form>
        <dl>
          <dt>User Id</dt>
          <dd>
            <Field type="text" name="userId" />
          </dd>
          <dd className="text-danger">
            <ErrorMessage name="userId" />
          </dd>
          <dt>User Name</dt>
          <dd>
            <Field type="text" name="userName" />
          </dd>
          <dd className="text-danger">
            <ErrorMessage name="userName" />
          </dd>
          <dt>Password</dt>
          <dd>
            <Field type="password" name="password" />
          </dd>
          <dd className="text-danger">
            <ErrorMessage name="password" />
          </dd>
          <dt>Email</dt>
          <dd>
            <Field type="email" name="email" />
          </dd>
          <dd className="text-danger">
            <ErrorMessage name="email" />
          </dd>
          <dt>Age</dt>
          <dd>
            <Field type="number" name="age" />
          </dd>
```

```
          <dd className="text-danger">
            <ErrorMessage name="age" />
          </dd>
          <dt>Mobile</dt>
          <dd>
            <Field type="text" name="mobile" />
          </dd>
          <dd className="text-danger">
            <ErrorMessage name="text" />
          </dd>
        </dl>
        <button className="btn btn-primary">Register</button>
        <div>
          <Link to="/login">Existing user? Login</Link>
        </div>
      </Form>
    }
  </Formik>
 </div>
);
}
```

shopper-login.js

```
import { Formik, Form, Field } from "formik";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";

export function ShopperLogin() {
 const navigate = useNavigate();

 return (
  <div>
   <h2>User Login</h2>
   <Formik
    initialValues={{
     userId: "",
     password: "",
    }}
    onSubmit={(values) => {
     axios({
```

```
        method: "GET",
        url: "http://localhost:5000/users",
       }).then((response) => {
        for (var user of response.data) {
         if (user.userId === values.userId && ) {
          navigate("/home");
          break;
         } else {
          navigate("/invalid");
         }
        }
       });
      }}
     >
      <Form>
       <dl>
        <dt>User Id:</dt>
        <dd>
         <Field type="text" name="userId" />
        </dd>
        <dt>Password:</dt>
        <dd>
         <Field type="password" name="password" />
        </dd>
       </dl>
       <button className="btn btn-success">Login</button>
       <div>
        <Link to="/register">New user? Register</Link>
       </div>
      </Form>
     </Formik>
    </div>
  );
}
```

shopper-invalid.js

```
import { Link } from "react-router-dom";

export function ShopperInvalid() {
  return (
```

```
  <div className="text-denget">
    <h3>Invalid User Name/ Password</h3>
    <div><Link to="/login">Try Again</Link></div>
  </div>
 );
}
```

shopper-index.js

```
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";
import { ShopperCategory } from "../shopper-category/shopper-category";
import { ShopperDetails } from "../shopper-details/shopper-details";
import { ShopperHome } from "../shopper-home/shopper-home";
import { ShopperRegister } from "../shopper-register/shopper-register";
import { ShopperLogin } from "../shopper-login/shopper-login";
import { ShopperInvalid } from "../shopper-invalid/shopper-invalid";

export function ShopperIndex() {
 return (
  <div className="container-fluid">
   <BrowserRouter>
    <header className="d-flex p-2 justify-content-between">
     <h2 className="">Shopper.</h2>
     <nav className="d-flex">
      <div className="me-3">
       <Link className="btn" to="home">Home</Link>
      </div>
      <div className="me-3">
       <Link className="btn" to="register">Register</Link>
      </div>
      <div className="me-3">
       <Link className="btn" to="login">Login</Link>
      </div>
      <div className="me-3">
       <Link className="btn" to="category/men's clothing">
        Men's Fashion
       </Link>
      </div>
      <div className="me-3">
       <Link className="btn" to="category/women's clothing">
        Women's Fashion
```

```
          </Link>
        </div>
        <div className="me-3">
          <Link className="btn" to="category/jewelery">Jewellery</Link>
        </div>
        <div className="me-3">
          <Link className="btn" to="category/electronics">Electronics</Link>
        </div>
      </nav>
      <div>
        <span className="bi bi-search me-3"></span>
        <span className="bi bi-person me-3"></span>
        <span className="bi bi-heart me-3"></span>
        <span className="bi bi-cart4 me-3"></span>
      </div>
    </header>
    <div className="text-center text-white bg-dark mt-3 p-2">
      Happy Holiday Deals on Everything
    </div>
    <div className="mt-3">
      <Routes>
        <Route path="/" element={<ShopperHome />} />
        <Route path="/home" element={<ShopperHome />} />
        <Route path="/category/:catname" element={<ShopperCategory />} />
        <Route path="details/:id" element={<ShopperDetails />} />
        <Route path="register" element={<ShopperRegister />} />
        <Route path="login" element={<ShopperLogin />} />
        <Route path="invalid" element={<ShopperInvalid />} />
      </Routes>
    </div>
  </BrowserRouter>
 </div>
);
}
```

Cookies in React Application

- Cookie in web applications is a simple text document, which contains client related information.
- Cookie is one of the state management techniques used by web application.

- Cookies are used to store client details on client device so that they can be used across requests.
- Cookies are 2 types
    - InMemory cookie
    - Persistent cookie
- InMemory cookie is store in browser and it is removed when browser closed. It is temporary cookie.
- Persistent cookies are permanent cookies stored in client hard drive.
- In React we are following the below operations for cookies,
    - Create cookies
    - Store data into cookie
    - Access data from cookies
    - Delete cookie
- React can use various 3rd party cookie libraries to manage cookies.
- Install cookie library
  CMD> npm install react-cookie --save

- Cookie library is provided as a service
    a. Service is pre-defined business logic
    b. You can inject and use the service in your application.
    c. Service uses “Singleton” pattern.
    d. Service depends on
        i. Provider
        ii. Injector
    e. Provider is responsible for locating your service object inn memory.
    f. Injector is responsible for injecting your service into any component.
    g. Providers are defined at application level.
       index.js
       ```
       <React.Fragment>
           <CookiesProvider>
                   <IndexComponent/>
           </CookiesProvider>
       </React.Fragment>
       ```
    h. Injector is “useCookies()” hook.

- Cookies are created by using “useCookies()”
  const [get, set, delete] = useCookie();
  setCookie(“name”, “value”, {expiry});

get[cookieName]
delete(cookieName)

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import "../node_modules/bootstrap/dist/css/bootstrap.css";
import "../node_modules/bootstrap-icons/font/bootstrap-icons.css";
import { ShopperIndex } from "./components/shopper-index/shopper-index";
import { CookiesProvider } from "react-cookie";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <CookiesProvider>
      <ShopperIndex />
    </CookiesProvider>
  </React.StrictMode>
);
```

shopper-login.js

```
import { Formik, Form, Field } from "formik";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";
import { useCookies } from "react-cookie";

export function ShopperLogin() {
  const navigate = useNavigate();
  const [cookies, setCookie, removeCookie] = useCookies();

  return (
    <div>
      <h2>User Login</h2>
      <Formik
        initialValues={{
          userId: "",
          password: "",
        }}
        onSubmit={(values) => {
```

```
        axios({
          method: "GET",
          url: "http://localhost:5000/users",
        }).then((response) => {
          for (var user of response.data) {
            if (user.userId === values.userId &&
              user.password === values.password ) {
              setCookie("userId", values.userId);
              navigate("/home");
              break;
            } else {
              navigate("/invalid");
            }
          }
        });
      }}
    >
      <Form>
        <dl>
          <dt>User Id:</dt>
          <dd>
            <Field type="text" name="userId" />
          </dd>
          <dt>Password:</dt>
          <dd>
            <Field type="password" name="password" />
          </dd>
        </dl>
        <button className="btn btn-success">Login</button>
        <div>
          <Link to="/register">New user? Register</Link>
        </div>
      </Form>
    </Formik>
  </div>
 );
}
```

shopper-home.js

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

```
import { useCookies } from "react-cookie";
import { useNavigate } from "react-router-dom";

export function ShopperHome() {
 const [cookies, setCookie, removeCookie] = useCookies();
 const navigate = useNavigate();
 useEffect(() => {
  if (cookies["userId"] === undefined) {
   navigate("/login");
  }
 });

 function SignoutClick() {
  removeCookie("userId");
  navigate("/login");
 }

 return (
  <div className="container-fluid d-flex justify-content-between">
   <div>
    <div className="d-flex justify-content-between">
     <div>
      <img alt="electronics" src="/images/electronics.jpg"
       style={{ with: "200px", height: "200px" }} />
     </div>
     <div>
      <img alt="mensclothing" src="/images/mensclothing.jpg"
       style={{ with: "200px", height: "200px" }} />
     </div>
     <div>
      <img alt="womensclothing" src="/images/womensclothing.jpg"
       style={{ with: "200px", height: "200px" }} />
     </div>
     <div>
      <img alt="jewelery" src="/images/jewelery.jpg"
       style={{ with: "200px", height: "200px" }} />
     </div>
    </div>
   </div>
   <div>
```

```
      <h4>Hello! - {cookies["userId"]}</h4>
    </div>
    <button onClick={SignoutClick()} className="btn btn-link">
      Signout
    </button>
  </div>
 );
}
```

shopper-category.js

```
import axios from "axios";
import { useEffect, useState } from "react";
import { useNavigate, useParams } from "react-router-dom";
import { Link } from "react-router-dom";
import { useCookies } from "react-cookie";

export function ShopperCategory() {
 const params = useParams();
 const [products, setProducts] = useState([]);
 const [cookies, setCookie, removeCookie] = useCookies();
 const navigate = useNavigate();
 useEffect(() => {
  if (cookies["userId"] === undefined) {
   navigate("/login");
  }
  axios({
   method: "GET",
   url: `https://fakestoreapi.com/products/category/${params.catname}`,
  }).then((response) => {
   setProducts(response.data);
  });
 }, [params.catname]);

 return (
  <div className="container-fluid">
   <h2>
    Shopper Category {params.catname} - {cookies["userId"]}
   </h2>
   <div className="d-flex flex-wrap">
    {products.map((product) => (
```

```
      <div className="card m-2 p-2" style={{ width: "200px" }}>
        <img className="card-img-top" src={product.image}
          style={{ height: "150px" }} />
        <div className="card-header" style={{ height: "150px" }}>
          <p>{product.title}</p>
        </div>
        <div className="card-footer">
          <Link to={"/details/"+product.id} className="btn btn-primary w-100">
            Details
          </Link>
        </div>
      </div>
    ))}
   </div>
  </div>
 );
}
```

CRUD Operation

- C – Create
- R – Read
- U – Update
- D – Delete

Steps:

1. Create a new database table in MongoDB
   Database name: shopper
   Collection name: products

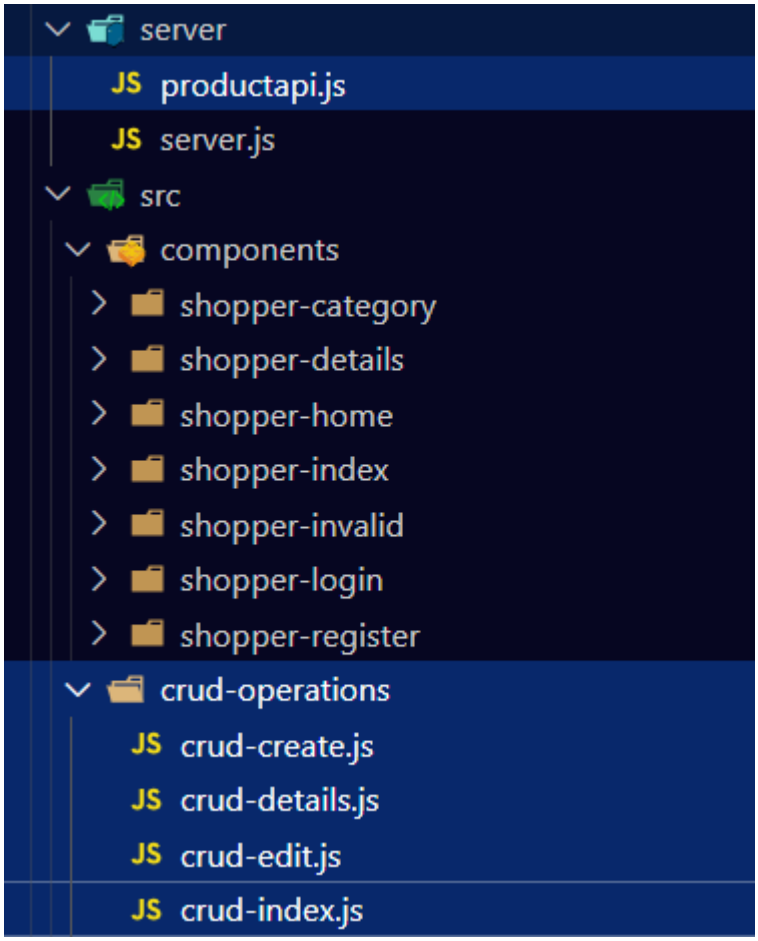
2. Add data into products collection with information
   ```
   {
    "productId": 1,
    "name": "Samsung TV",
    "price": 45600.44,
    "stock": true
   }
   ```

3. Create an API in Node JS for handling all CRUD operations
   GET /product/all Fetch all products
   GET /product/1 Fetch specific product details

POST	/product/add	Insert a new product into DB
PUT	/product/update	Update the product details
DELETE	/product/delete	Deletes any specific product

4. Then develop the frontend

Create the below Folder and File.



productapi.js

```
var mongoClient = require("mongodb").MongoClient;
var express = require("express");
var cors = require("cors");

var connectionString = "mongodb://127.0.0.1:27017";

var app = express();
app.use(cors());
app.use(
  express.urlencoded({
    extended: true,
```

```
 })
);
app.use(express.json());

//To fetch all the products
app.get("/product/all", (request, response) => {
 mongoClient.connect(connectionString).then((clientObject) => {
  var database = clientObject.db("shopper");
  database.collection("products").find({}).toArray()
   .then((documents) => {
    response.send(documents);
    response.end();
   });
 });
});

//To get specific product details
app.get("/product/:id", (request, response) => {
 mongoClient.connect(connectionString).then((clientObject) => {
  var database = clientObject.db("shopper");
  console.log(" request.params.id  - " + request.params.id);
  database.collection("products")
   .findOne({ productId: parseInt(request.params.id) })
   .then((documents) => {
    response.send(documents);
    response.end();
   });
 });
});

//To add product
app.post("/product/add", (request, response) => {
 mongoClient.connect(connectionString).then((clientObject) => {
  var database = clientObject.db("shopper");
  var product = {
   productId: request.body.productId,
   name: request.body.name,
   price: parseFloat(request.body.price),
   stock: request.body.stock == "true" ? true : false,
  };
```

```
    database.collection("products").insertOne(product)
      .then((result) => {
        response.redirect("/product/all");
        response.end();
      });
  });
});

//To update product
app.put("/product/update/:id", (request, response) => {
  mongoClient.connect(connectionString).then((clientObject) => {
    var database = clientObject.db("shopper");
    var findProduct = { productId: parseInt(request.params.id) };
    var updateProduct = {
      name: request.body.name,
      price: parseFloat(request.body.price),
      stock: request.body.stock == "true" ? true : false,
    };
    database.collection("products")
      .updateOne(findProduct, { $set: updateProduct })
      .then((result) => {
        console.log("Product Updated");
        response.end();
      });
  });
});

//To delete the product
app.delete("/product/delete/:id", (request, response) => {
  mongoClient.connect(connectionString).then((clientObject) => {
    var database = clientObject.db("shopper");
    var findProduct = { productId: parseInt(request.params.id) };
    database.collection("products").deleteOne(findProduct)
      .then((result) => {
        console.log("Product Deleted");
        response.end();
      });
  });
});
```

```
app.listen(8989);
console.log("Server started : http://127.0.0.1:8989");
```

crud-index.js

```
import axios from "axios";
import { useState, useEffect } from "react";
import { Link, useNavigate } from "react-router-dom";

export function CrudIndex() {
  const [products, setProducts] = useState([]);
  const navigate = useNavigate();

  useEffect(() => {
    axios.get("http://localhost:8989/product/all").then((response) => {
      setProducts(response.data);
    });
  }, []);

  function DeleteClick(e) {
    var flag = window.confirm("Are you sure \n Want to Delete?");
    if (flag) {
      axios.delete(
        `http://localhost:8989/product/delete/${e.currentTarget.value}`
      );
      alert("Product Deleted");
      navigate("/products");
    }
  }

  return (
    <div className="container-fluid">
      <h2>Products Grid</h2>
      <div className="mb-3">
        <Link to="/new-product" className="btn btn-primary">
          Add New Product
        </Link>
      </div>
      <table className="table table-hover">
        <thead>
          <tr>
```

```
          <th scope="col">Product Name</th>
          <th scope="col">Product Price</th>
          <th scope="col">Product Stock</th>
          <th scope="col">Action</th>
        </tr>
      </thead>
      <tbody>
       {products.map((product) => (
        <tr key={product.productId}>
          <td>{product.name}</td>
          <td>{product.price}</td>
          <td>{product.stock === true ? "Available" : "Out of stock"}</td>
          <td>
           <Link
            className="btn btn-info" to={"/crud-details/" + product.productId}>
            <span className="bi bi-eye"></span>
           </Link>
           <Link className="btn btn-warning ms-2"
             to={"/crud-edit/" + product.productId} >
            <span className="bi bi-pen"></span>
           </Link>
           <button value={product.productId} className="btn btn-danger ms-2"
            onClick={DeleteClick} >
            <span className="bi bi-trash"></span>
           </button>
          </td>
        </tr>
       ))}
      </tbody>
    </table>
  </div>
 );
}
```

crud-create.js

```
import { Formik, Form, Field, ErrorMessage } from "formik";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";
import { useEffect, useState } from "react";
```

```
export function CrudCreate() {
 const navigate = useNavigate();
 const [products, setProducts] = useState([]);
 const [idError, setIdError] = useState("");

 useEffect(() => {
  axios.get("http://localhost:8989/product/all").then((response) => {
   setProducts(response.data);
  });
 }, []);

 function VerifyId(e) {
  var id = parseInt(e.target.value);
  for (var product of products) {
   if (product.productId === id) {
    setIdError("ID already exists");
    break;
   } else {
    setIdError("");
   }
  }
 }
 return (
  <div className="container-fluid">
   <h2>Add New Product</h2>
   <Formik
    initialValues={{
     productId: 0,
     name: "",
     price: 0,
     stock: false,
    }}
    onSubmit={(values) => {
     axios({
      method: "post",
      url: "http://127.0.0.1:8989/product/add",
      data: values,
     }).then(() => {
      alert("Product Added");
      navigate("/products");
```

```
      });
    }}
  >
    {
     <Form>
      <dl>
       <dt>Product ID:</dt>
       <dd>
        <Field type="number" onKeyUp={VerifyId} name="productId" />
       </dd>
       <dd className="text-danger">{idError}</dd>
       <dt>Product Name:</dt>
       <dd> <Field type="text" name="name" /> </dd>
       <dt>Product Price:</dt>
       <dd> <Field type="number" name="price" /> </dd>
       <dt>Product Stock:</dt>
       <dd> <Field type="checkbox" name="stock" /> Available </dd>
      </dl>
      <button className="btn btn-primary">Add Product</button>
      <Link className="ms-2" to="/products"> View Products </Link>
     </Form>
    }
   </Formik>
  </div>
 );
}
```

crud-details.js

```
import axios from "axios";
import { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";

export function CrudDetails() {
 const params = useParams();
 const [product, setProduct] = useState({});

 useEffect(() => {
  axios.get(`http://localhost:8989/product/${params.id}`).then((response) => {
    setProduct(response.data);
  });
```

```
  }, []);

  return (
    <div className="container-fluid">
      <h2> Product Details</h2>
      <dl>
        <dt>Product ID:</dt>
        <dd>{product.productId}</dd>
        <dt>Product Name:</dt>
        <dd>{product.name}</dd>
        <dt>Product Price:</dt>
        <dd>{product.price}</dd>
        <dt>Product Stock:</dt>
        <dd>{product.stock === true ? "Available" : "Out of Stock"}</dd>
      </dl>
      <Link to="/products">Back to Products</Link>
    </div>
  );
}
```

crud-edit.js

```
import axios from "axios";
import { Field, Form, Formik } from "formik";
import { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";

export function CrudEdit() {
  const params = useParams();
  const [product, setProduct] = useState({});

  useEffect(() => {
    axios.get(`http://localhost:8989/product/${params.id}`).then((response) => {
      console.log(response.data);
      setProduct(response.data);
    });
  }, []);

  return (
    <div className="container-fluid">
      <h2>Edit Product</h2>
```

```
    <Formik
      initialValues={{
        productId: product.productId,
        name: product.name,
        price: product.price,
        stock: product.stock,
      }}
    >
      {
        <Form>
          <dl>
            <dt>Name</dt>
            <dd>
              <Field type="text" name="name" value={product.name}></Field>
            </dd>
            <dt>Price</dt>
            <dd>
              <Field type="number" name="price" value={product.price}></Field>
            </dd>
            <dt>Stock</dt>
            <dd>
              <Field type="checkbox" name="stock" checked={product.stock} >
               </Field> Available
            </dd>
          </dl>
          <button className="btn btn-success">Edit</button>
          <div> <Link to="/products">Back to Products</Link></div>
        </Form>
      }
    </Formik>
  </div>
 );
}
```

shopper-index.js

```
import { BrowserRouter, Link, Route, Routes } from "react-router-dom";
import { ShopperCategory } from "../shopper-category/shopper-category";
import { ShopperDetails } from "../shopper-details/shopper-details";
import { ShopperHome } from "../shopper-home/shopper-home";
import { ShopperRegister } from "../shopper-register/shopper-register";
```

```
import { ShopperLogin } from "../shopper-login/shopper-login";
import { ShopperInvalid } from "../shopper-invalid/shopper-invalid";
import { CrudIndex } from "../../crud-operations/crud-index";
import { CrudCreate } from "../../crud-operations/crud-create";
import { CrudDetails } from "../../crud-operations/crud-details";
import { CrudEdit } from "../../crud-operations/crud-edit";

export function ShopperIndex() {
  return (
    <div className="container-fluid">
      <BrowserRouter>
        <header className="d-flex p-2 justify-content-between">
          <h2 className="">Shopper.</h2>
          <nav className="d-flex">
            <div className="me-3">
              <Link className="btn" to="home"> Home</Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="products">Products</Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="register">Register</Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="login"> Login</Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="category/men's clothing">
                Men's Fashion
              </Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="category/women's clothing">
                Women's Fashion
              </Link>
            </div>
            <div className="me-3">
              <Link className="btn" to="category/jewelery"> Jewellery </Link>
            </div>
            <div className="me-3">
```

```
          <Link className="btn" to="category/electronics"> Electronics </Link>
        </div>
      </nav>
      <div>
        <span className="bi bi-search me-3"></span>
        <span className="bi bi-person me-3"></span>
        <span className="bi bi-heart me-3"></span>
        <span className="bi bi-cart4 me-3"></span>
      </div>
    </header>
    <div className="text-center text-white bg-dark mt-3 p-2">
      Happy Holiday Deals on Everything
    </div>
    <div className="mt-3">
     <Routes>
       <Route path="/" element={<ShopperHome />} />
       <Route path="/home" element={<ShopperHome />} />
       <Route path="/category/:catname" element={<ShopperCategory />} />
       <Route path="details/:id" element={<ShopperDetails />} />
       <Route path="register" element={<ShopperRegister />} />
       <Route path="login" element={<ShopperLogin />} />
       <Route path="invalid" element={<ShopperInvalid />} />
       <Route path="/products" element={<CrudIndex />} />
       <Route path="new-product" element={<CrudCreate />} />
       <Route path="crud-details/:id" element={<CrudDetails />} />
       <Route path="crud-edit/:id" element={<CrudEdit />} />
     </Routes>
    </div>
   </BrowserRouter>
  </div>
 );
}
```

React MUI

- It is Material UI library for React.
- It provides pre-defined components for React, developed by react.
- Official Website [Link].
- React 17+ versions material UI library changed.
- React 17+ version introduce “@emotion” library for components, which is not available for versions up to 17.

- The library for MUI, we will need add new google fonts and icons into old project [up tio 17].

Setup project for Material UI

CMD> npm install @mui/material -- save
CMD> npm install @emotion/react -- save
CMD> npm install @emotion/styled --save

Import components in your project

1. Import the library required for component
   a. Lazy loading
   b. Eager loading

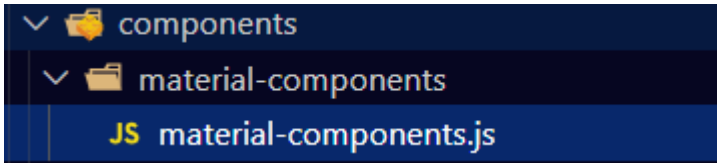
   import Button from “@mui/material/button”; - loading only button
   import {Button} from “@mui/material”; - loading complete library

2. Inject the component into your component, [MUI Components]

   <Button></Button>

3. Set values for properties in component.
   <Button propertyName=“value”>

Create the below Folder and File.



material-component.js

```
import { Button, TextField } from "@mui/material";

export function MaterialComponents() {
 function handleTextChange(e) {
  alert(e.target.value);
 }

 return (
  <div className="container-fluid">
   <h2>Bootstrap Button</h2>
```

```
    <button className="btn btn-primary">Register</button>
    <h2>React MUI Button</h2>
    <TextField label="Your Email" onChange={handleTextChange}
      variant="standard"></TextField>
    <br /> <br />
    <Button variant="contained" color="success"> Register </Button>
  </div>
 );
}
```

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import "../node_modules/bootstrap/dist/css/bootstrap.css";
import "../node_modules/bootstrap-icons/font/bootstrap-icons.css";
import { CookiesProvider } from "react-cookie";
import { MaterialComponents } from "./components/material-
components/material-components";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
 <React.StrictMode>
  <CookiesProvider>
   <MaterialComponents />
  </CookiesProvider>
 </React.StrictMode>
);
```

React Hooks

- Hook is function that returns any values or expression.
- It is designed for handling reusability, maintainability, extensibility etc.
- React introduced hooks from version 17x.
- You can't use hook in class component.
- Hooks are only for "Function components".
- React provides several pre-defined hooks and also allows to create custom hooks.
- React pre-defined hooks,
    - useState()
    - useEffect()

- useCookies()
- useParams()
- useNavigate(), etc.

- Every hook allocates memory and initializes value into memory.
- Hook is not assigning value; it is initialization of value.

useState():

- It configures a state for component, where we can store a value and use it from any location in the component.
- It defines mutable data structure.

Syntax:

```
const [getter, setter] = useState(any)
any – primitive, non-primitive
const [getter] = useState(any);   //readonly
```

useEffect():

- It defines actions to perform on component mount and unmount.
- useEffect will load only once per request.
- The dependencies define when to mount the component again.
- It is a array of dependencies.
- It is uses DI mechanism [Dependency injection].

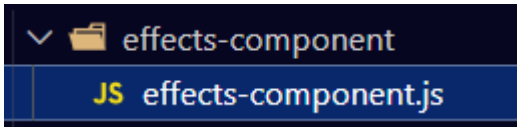
Syntax:

```
useEffect(()=> {
        //action on mount
        return() => {
                //actions on unmount
        }
}, [dependency]);
```

Q. What is difference between break and return;

Ans: return will terminate program and destroy the memory allocation for the program where as break only terminate the program.

Create the below Folder and File.



effect-component.js

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

export function Login() {
  useEffect(() => {
    alert("Login component requested and will mount");
    return () => {
      alert("Login component unmounted");
    };
  }, []);

  return (
    <div> <h2>User Login</h2> </div>
  );
}

export function Register() {
  useEffect(() => {
    alert("Register component requested and will mount");
    return () => {
      alert("Register component unmounted");
    };
  }, []);

  return (
    <div> <h2>User Register</h2> </div>
  );
}

export function EffectsComponent() {
  const [component, setComponent] = useState("");

  function LoginClick() {
    setComponent(<Login />);
  }

  function RegisterClick() {
    setComponent(<Register />);
  }
```

```
 return (
  <div className="container-fluid mt-4">
   <button onClick={LoginClick}>Login</button>
   <button onClick={RegisterClick}>Register</button>
   <hr />
   {component}
  </div>
 );
}
```

Another Example

effect-component.js

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

export function ShoppingCart(props) {
 return (
  <div className="container m-4 p-4">
   <span className="bi bi-cart4"></span> {props.count}
  </div>
 );
}

export function EffectsComponent() {
 const [count, setCount] = useState(0);

 function AddToCart() {
  alert("Item added");
  var currentCount = count + 1;
  setCount(currentCount);
 }

 return (
  <div className="container-fluid mt-4">
   <div className="position-absolute top-0 end-0">
    <ShoppingCart count={count} />
   </div>
   <div> <button onClick={AddToCart}>Add to cart</button></div>
  </div>
 );
}
```

useEffect without dependency

effect-component.js

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

export function ShoppingCart() {
 const [count, setCount] = useState(0);

 useEffect(() => {
  var currentCount = count + 1;
  setCount(currentCount);
 });

 return (
  <div className="container m-4 p-4">
   <span className="bi bi-cart4"></span> {count}
  </div>
 );
}

export function EffectsComponent() {
 const [component, setComponent] = useState("");

 function AddToCart() {
  alert("Cart Requested");
  setComponent(<ShoppingCart />);
 }

 return (
  <div className="container-fluid mt-4">
   <div>
    <button onClick={AddToCart}>Add to cart</button>
   </div>
   <div>{component}</div>
  </div>
 );
}
```

- If dependency is not defined then useEffect will mount the component at regular intervals until it is stopped.

useContext():

- Context refers to context memory of component.
- Context memory is the memory allocated for a component and is accessible to other components that run within the context of current component.
- Context memory is created for parent and it is accessible all its child component at any level of hierarchy.
- That means a parent component can transport data to its child component.
- It is used DI mechanism
  - Provider
  - Injector
- Provider is responsible for locating the value in memory.
- Injector is responsible for injecting value into any component.

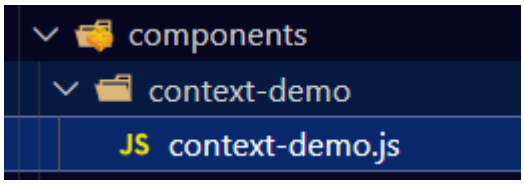
Step:

1. Create a new context memory
   let contextName = React.createContext(null);

2. Configure a provider, which looks for the value in context memory and provides to the components that run within the context.
   <contextName.Provider value= “{}”>
   <child-component/>
   </contextName.provider>

3. The components in context can access the value by using “useContext()”
   Child-component
   const ref = useContext(contextName);
   ref.value;

Create the below Folder and File.



amazon.component.js

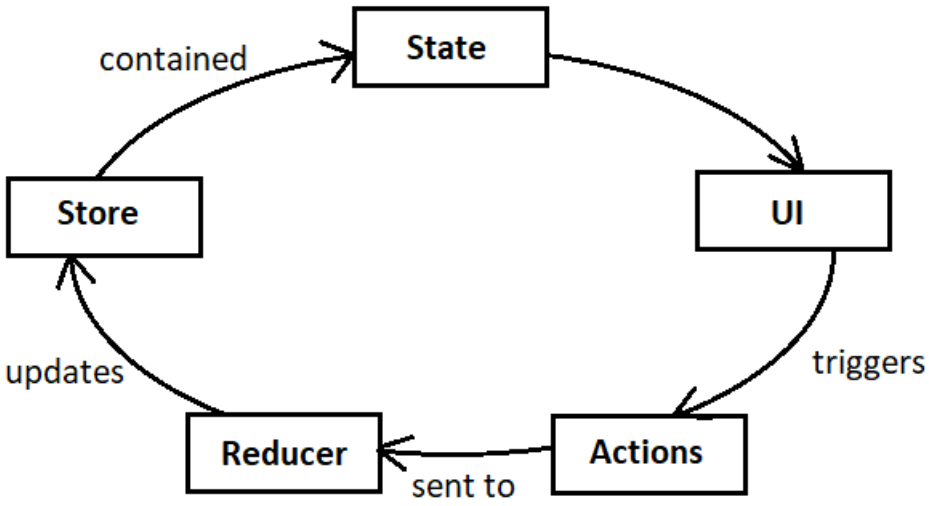
```
import React, { useContext, useState } from "react";

let UserDetailsContext = React.createContext(null);
```

```
export function MensClothing() {
 const context = useContext(UserDetailsContext);

 return (
  <div className="bg-warning p-4">
   <h4>Mens Clothing - {context.userName}</h4>
  </div>
 );
}

export function HomeComponent() {
 const context = useContext(UserDetailsContext);

 return (
  <div className="bg-light text-dark p-3">
   <h3>Home - {context.userName}</h3>
   <MensClothing />
  </div>
 );
}

export function ContextDemo() {
 const [userName, setUserName] = useState("");

 function handleUserNameChange(e) {
  setUserName(e.target.value);
 }

 return (
  <div
   className="container-fluid bg-dark text-white p-3"
   style={{ height: "300px" }}>
   <h2>Main Component</h2>
   <dl>
    <dt>User Name</dt>
    <dd> <input type="text" onChange={handleUserNameChange} /> </dd>
   </dl>
   <UserDetailsContext.Provider value={{ userName: userName }}>
    <HomeComponent />
```

```
    </UserDetailsContext.Provider>
  </div>
 );
}
```

useReducer():

- It is used to configured state for application.
- Application state can store values and male those values available across multiple requests.
- useReducer internally uses various components,
    - Store
    - Action
    - Dispatcher



Step:

1. UI triggers the action
2. Action sent to Reducer
3. Reducer will identify the action type and updates into Store.
4. Data is updated into Store.
5. Data from store is provided to state of any component.
6. Component can display the data in View [UI].

- Reducer is a component and function that handles state, action.

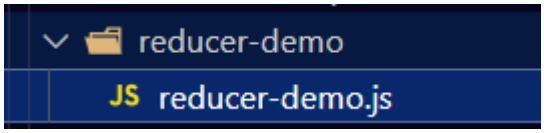
```
function reducer(state, action) {
  switch(action.type) {
        case “join”:
              return state++;
```

```
                case “exit”:
                        return state --;
    }
  }
```

- useReducer hook is used by any component to access the reducer.

```
const [state, dispatch] = useReducer(reducer, initialState);
{state.value}
onClick() {
  dispatch({type: “join”});
}
```

Create the below Folder and File.



reducer-demo.js

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

let initialState = { count: 0 };

function reducer(state, action) {
 switch (action.type) {
  case "JOIN":
   return { count: state.count + 1 };
  case "EXIT":
   return { count: state.count - 1 };
 }
}

export function ReducerDemo() {
 const [state, dispatch] = useReducer(reducer, initialState);

 function JoinClick() {
  dispatch({ type: "JOIN" });
 }

 function ExitClick() {
  dispatch({ type: "EXIT" });
```

```
  }

  return (
    <div className="container-fluid">
      <h2>Live <span className="bi bi-youtube"></span> Video Streaming</h2>
      <h4> {state.count} Watching</h4>
      <button onClick={JoinClick} className="btn btn-primary me-2">
        Join
      </button>
      <button onClick={ExitClick} className="btn btn-danger">
        Exist
      </button>
    </div>
  );
}
```

Hooks Link: [Link]

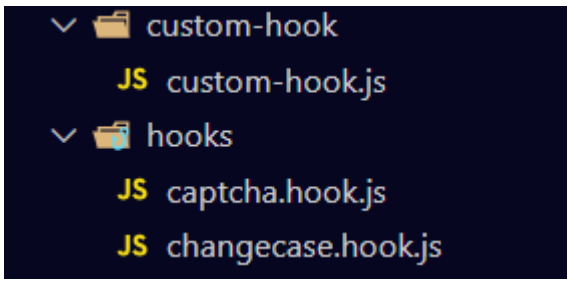
Note: Install the below extension base on browser

- [Chrome Redux DevTools Extension Link]
- [Edge Redux DevTools Extension Link]
- [Chrome React Developer Tools Extension Link]
- [Edge React Developer Tools Extension Link]
- [Chrome Reactime Extension Link]
- [Edge Reactime Extension Link]

Custom Hooks

- We can create our own hook to handle specific operations.
- Every hook is a function that returns a value.
- Hooks can't be uses in a function or any iterations.
- Hooks are used at higher level.

Create the below Folder and File.



changecase.hook.js

```
export function useChangeCase(str) {
  var firstChar = str.charAt(0);
  var restChar = str.substring(1);
  return firstChar.toUpperCase() + restChar.toLowerCase();
}
```

captcha.hook.js

```
export function useCaptcha() {
  var a = Math.random() * 10;
  var b = Math.random() * 10;
  var c = Math.random() * 10;
  var d = Math.random() * 10;
  var e = Math.random() * 10;
  var f = Math.random() * 10;

  var code = `${Math.round(a)} ${Math.round(b)} ${Math.round(c)}
                ${Math.round(d)} ${Math.round(e)} ${Math.round(f)}`;
  return code;
}
```

custom-hook.js

```
import { useCaptcha } from "../hooks/captcha.hook";
import { useChangeCase } from "../hooks/changecase.hook";

export function CustomHook() {
  const msg = useChangeCase("welCoMe to REACT");
  const captcha = useCaptcha();

  return (
    <div className="container-fluid">
      <h1>{msg}</h1>
      <h3>{captcha}</h3>
    </div>
  );
}
```

Issue with JavaScript:

- JavaScript is not strongly typed.
- JavaScript is not implicitly strictly typed. “use strict”.

- JavaScript is not an OOP language.
- Extensibility issues.
- Code Security issues.

TypeScript

- It is Strongly typed.
- It is strictly typed.
- It is an OOP language, supports all features of OOP.
- Developer uses TypeScript, which is trans compiled into JavaScript.
- It is a language introduced by “Anders Hejiberg” of “Microsoft” [C#].
- TypeScript is individually used for developing large scale applications.

Setup and install TypeScript:

1. We need to install typescript on our machine then we can use in any project for that open command prompt and run the below command.
   CMD> npm install -g typescript

2. To check is it installed or not
   CMD> tsc -v

TypeScript Program:

- Program have extension “.ts”.
- We have to trans compile typescript into javascript
  CMD> tsc program.ts
  program.js

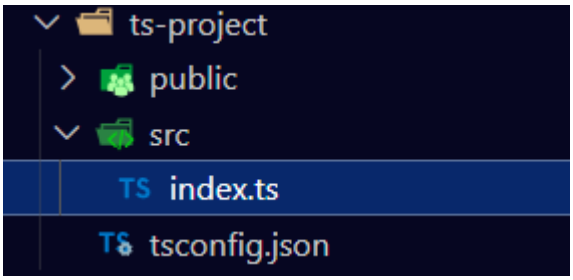
- We can run javascript program using node compiler
  CMD> node program.js
  - Only when the program contains console methods.
  - If it’s having DOM methods then use with HRML

TypeScript Project Setup:

- Create a new folder for project
  E:\ts-project

- Open in VS Code
- Open VS Code terminal and run the following command, it will create tsconfig.json file.
  CMD> tsc –init

- “tsconfig.json” defines rules for using typescript in project.
- Add new folders



TypeScript Language Basics

1. Variables

Var, let, const

2. Data types

- Typescript is strongly typed.
- Data type can be primitive or not primitive type.

Primitive types

- number, string, boolean, null, undefined

  Syntax:

  let variableName:datatype;

index.ts

```
let username: string = "45sdfsdf";
let age: number = 22;
let subscribed: boolean = true;

console.log(`Name: ${username}\nAge: ${age}\nSubscribed: ${subscribed}`);
```

To run this, go the **src** folder or VS code terminal,

E:\ts-project\src> tsc index.ts

It will generate the index.js file

E: \ts-project\src> node index.js

- If data type is not defined then it is referred as “any” type.
- TypeScript supports type inference. It sets the data type according to the value initialized.

  let x = 10; number

  let x; any

- TypeScript supports union type.
  let x: string | number;

index.ts

```
let username: string | null = prompt("Enter Name");
let x: number | string | boolean;
```

Non-Primitive Types:

- **Array**
  - TypeScript array can restrict similar types of values.
    let sales: number[] = [];
    let categories: string[] = [];
    let values: any[] = []; //all types allowed
    let values: any[] = new Array(); // all types not allowed type depends on first value
  - TypeScript array methods all are same as JavaScript.
  - TypeScript support union for Arrays also.

index.ts

```
let values: any[] = [10, 20, true, "A"];
let newValues: any[] = new Array(10, "A", true); // will not works
let workingNewValues: any[] = new Array(); // This will works
workingNewValues[0] = 10;
workingNewValues[1] = "A";
```

- **Object**
  - The type for object is defined with a set of properties and their data type.
    Syntax:
    let obj: {property: datatype, property: datatype};

index.ts

```
let product: { name: String; price: number; stock: boolean } = {
  name: "TV",
  price: 45000.44,
  stock: true,
};
```

- **Array of Object**
  let obj: {property: datatype, property: datatype}[];

index.ts

```
let products: { name: String; price: number; stock: boolean }[] = [
 {
  name: "TV",
  price: 45000.44,
  stock: true,
 },
 {
  name: "Mobile",
  price: 25000.44,
  stock: true,
 },
];
```

- **Map Type**

  let obj: any = new Map();

- **Date Type**

  let mfd: Date = new Date();

Note:

- ✓ TypeScript operators, statements, functions is same like JavaScript.
- ✓ Function can have return type defined, if it is not returning anything then its void.

  Syntax:

  function Name(): returnType | void {
  }

- ✓ Function can have parameters, but type must be defined

index.ts

```
function print(a: number, b: number): number {
 return a + b;
}
```

- ✓ Function can have optional parameters, but they can't preside with required parameter. That means all optional parameter must at last. We can't have a required parameter after optional parameter.

index.ts

```
function Details( id: number, name: string, rating?: { rate: number; count: number;}): void {

}

Details(1, "TV", { rate: 4.3, count: 3000 });
Details(1, "Mobile");
```

TypeScript OOP

1. **Contracts:**
   - A contract defines the rules for designing a component.
   - Contracts are known as “Interface”.
   - Interface defines rules for component.
   - It is defied by using “interface” keyword.

     Syntax:

     interface Product {

     }

   - Interface can contain only rules not definition.
   - Interface can have optional rules, which are defined by using “?”.

index.ts

```
interface Product {
  name: string;
  price: number;
  stock: boolean;
  rating?: {
    rate: number;
    count: number;
  };
  cities: string[];
}

let product: Product = {
  name: "TV",
  price: 435677.99,
  stock: true,
```

```
  rating: {
    rate: 4.5,
    count: 10,
  },
  cities: ["HYD", "PUNE"],
};
```

- Interface can have readOnly rules, once the values is initialized it can't be reassigned.

index.ts

```
interface Product {
  name: string;
  readonly price: number;
}

let product: Product = {
  name: "TV",
  price: 43567.99,
};

product.name = "Samsung TV";
product.price = 70000.00; //Not working
console.log(`Name: ${product.name}\n Price: ${product.price}`);
```

- Interface can have rules for functions and method.
- Interface can be extended, multiple or multilevel.

Syntax:

```
interface Name extends IOne, ITwo, IThree {

}
```

index.ts

```
interface Product {
  name: string;
  price: number;
  qty: number;
  total(): number;
  print?(): void;
```

```
}

interface Category extends Product {
  categoryName: string;
}

let product: Category = {
  categoryName: "Electronics",
  name: "TV",
  price: 43567.99,
  qty: 2,
  total() {
    return this.price * this.qty;
  },

  print() {
    console.log(
      `Product: ${this.name}\nPrice: ${this.price}\nQuantity: ${
        this.qty
      }\nTotal: ${this.total()}
      \nCategory: ${this.categoryName}`
    );
  },
};

product.print();
```

index.ts

```
interface HDFC_Version1 {
  personal: String;
  NRI: string;
}

interface HDFC_Version2 {
  loan: String;
}

interface Bank extends HDFC_Version1, HDFC_Version2 {
  bankName: string;
}
```

```
let bankDetails: Bank = {
  bankName: "HDFC BANK",
  personal: "Personal Banking",
  NRI: "NRI Banking",
  loan: "Housing loan",
};
console.log(JSON.stringify(bankDetails));
```

2. Class

- TypeScript class can have static and non-static members.
- Static refers to continuous memory.
- Non-static refers to direct memory.

Static:

- Memory allocated for first object will continue for next object.
- Static members are defined by using “static” keyword.
- Static members are accessed with in class or outside class by using class name.

Non-static:

- Memory is allocated for every object individually.
- They are disconnected.
- Members are accessed with in class using “this” ad outside class using instance of class.

index.ts

```
class Demo {
  static s = 0;
  n = 0;
  constructor() {
    Demo.s = Demo.s + 1;
    this.n = this.n + 1;
  }

  print() {
    console.log(`s=${Demo.s} n=${this.n}`);
  }
}
```

```
let obj1 = new Demo();
obj1.print();

let obj2 = new Demo();
obj2.print();

let obj3 = new Demo();
obj3.print();
```

- Class can have access modifiers for members
  - Public: It is accessible for ant location.
  - Private: It is accessible within class.
  - Protected: It is accessible within the class, and within derived class only by using derived class reference.

index.ts

```
class Super {
  public name: string = "TV";
  private price: number = 3500.0;
  protected stock: boolean = true;
}

class Dervied extends Super {
  public print(obj:Dervied) {
    obj.name;
    obj.stock;
  }
}

let obj = new Dervied();
```

- Class members
  - Property : type
  - Method : type
  - Constructor: param type
  - Accessor

index.ts

```
class Demo {
  public total(qty: number, price: number): number {
```

```
  return qty * price;
 }

 public print(): void {}
}
```

Note: Inheritance, Polymorphism as like JavaScript only.

3. Abstraction

- In OOP we use templates.
- Template comprises of features which are need to be implements.
- Template comes with some features already implemented and few need to be implemented.
- Templates are used in “Rollouts” and in implementation of modules in various domains [Finance, DW, AI, etc.]
- Templates are designed by using “Abstract classes”.
- Abstract class comprises of method which are already implemented and few methods which need to be implements.
- The incomplete methods are marked as “abstract”.
- If a class is having at least one abstract member, then the class is marked as “abstract”.
- Abstraction is the process of hiding the data structure and providing only the implementation.

index.ts

```
interface ProductContract {
 name: string;
 price: number;
 qty: number;
 total(): number;
 print(): void;
}

abstract class ProductTemplate implements ProductContract {
 public name: string = "";
 public price: number = 0;
 public qty: number = 0;
 public abstract total(): number;
 public abstract print(): void;
```

```
}

class ProductComponent extends ProductTemplate {
  name = "sumsung TV";
  price = 45000.44;
  qty = 2;
  public total(): number {
    return this.price * this.qty;
  }
  public print(): void {
    console.log(`Product Name: ${this.name}\n Product Price: ${this.price}\n
       Product Quantity: ${this.qty}`);
  }
}
```

4. Generics

- It is used to design type safe members.
- We can configure a type safe member with strongly typed nature.
- Generics can be defined for,
  - Parameters
  - Methods
  - Class
  - Property

index.ts

```
function print<T>(a: T) {
  return a;
}

print<number>(10);
print<string>("A");
```

index.ts

```
interface MongoDB {
  url: string;
}

interface Mysql {
  host: string;
  user: string;
```

```
  password: string;
  dbname: string;
}

interface Oracle {
  user: string;
  password: string;
  dbname: string;
}

class Database {
  public connection<T>(connectionString: T) {
    for (var property in connectionString) {
      console.log(`${property} : ${connectionString[property]}`);
    }
  }
}

console.log("------Connection with Oracle------");
let ora = new Database();
ora.connection<Oracle>({user: "scott", password: "tiger", dbname: "orcl",});

console.log("------Connection with Mysql------");
ora.connection<Mysql>({host: "localhost", user: "root", password: "root",
  dbname: "user",});
```

index.ts

```
interface Product {
  name: string;
  price: number;
}

interface Student {
  name: string;
  age: number;
}

class Service<T> {
  constructor(data: T) {
    for (var property in data) {
```

```
    console.log(`${property} : ${data[property]}`);
   }
  }

  public details: T | null = null;
}

let tv = new Service<Product>({ name: "Samsung TV", price: 34000.33 });
tv.details = { name: "", price: 0 };

let john = new Service<Student>({ name: "John", age: 25 });
john.details = { name: "", age: 0 };
```

5. Enum

- Enumeration is a set of constants.
- Every constant must have initialization.
- Enum can have set of constants, which can be
  - Number
  - String
  - Expression
- Enum will not allow any another type.

Q: Can we define boolean as Enum?
Ans: No, boolean is not constant.

- Enum Number constants have auto implementation for value.
- Enum number constants use ++ operator for next value.

index.ts

```
enum StatusCode {
  UNAUTHORIZED, //0
  OK = 200,
  NOT_FOUND = 404,
  DISCONNECTED, //405
}

console.log(`${StatusCode.UNAUTHORIZED}\n ${StatusCode.OK}\n
   ${StatusCode.NOT_FOUND}\n ${StatusCode.DISCONNECTED}`);
```

- Auto implementation is not supported for string type.

- Enum can have a combination of various type.

index.ts

```
enum StatusCode {
  UNAUTHORIZED = "You are not Authorized",
  OK = 200,
  NOT_FOUND = "Page not found",
  DISCONNECTED, //invalid
}

console.log(`${StatusCode.UNAUTHORIZED}\n
   ${StatusCode.OK}\n
   ${StatusCode.NOT_FOUND}\n
   ${StatusCode.DISCONNECTED}`);
```

- Enum can have an expression also.
- All conditional operators are not allowed because they are retuning boolean and boolean are not allowed.

index.ts

```
enum StatusCode {
  A = 10,
  B = 20,
  C = A + B,
}

console.log(`addition - ${StatusCode.C}`);
```

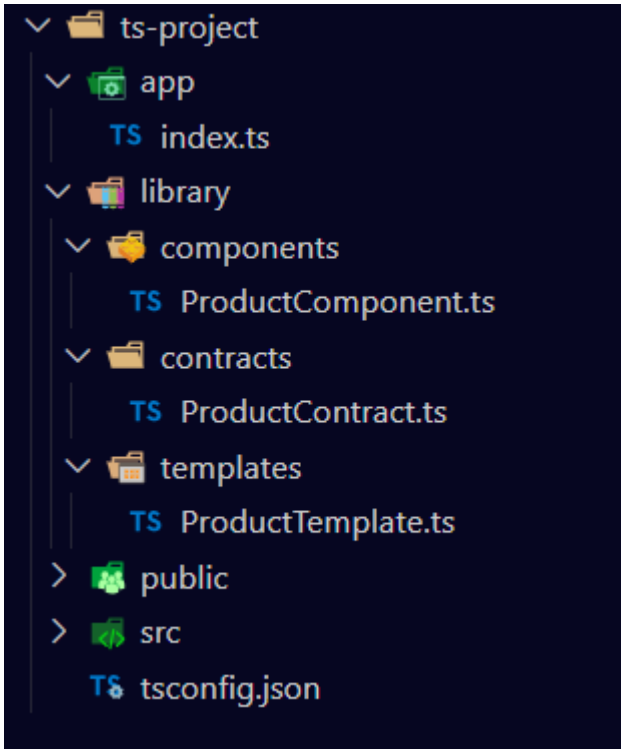
Q. What is enum reverse mapping?
Ans: It is a technique used to access key with refence of value.

index.ts

```
enum StatusCode {
  NOT_FOUND = 404,
}

console.log(`Status Code: ${StatusCode.NOT_FOUND}\n
   Status Text: ${StatusCode[404]}`);
```

Note: TypeScript modules are same likes as JavaScript.

Create the below Folder and File.



ProductContract.ts

```ts
export interface ProductContract {
  name: string;
  price: number;
  qty: number;
  total(): number;
  print(): void;
}
```

ProductTemplate.ts

```ts
import { ProductContract } from "../contracts/ProductContract";

export abstract class ProductTemplate implements ProductContract {
  public name: string = "";
  public price: number = 0;
  public qty: number = 0;
  public abstract total(): number;
  public abstract print(): void;
}
```

ProductComponent.ts

```ts
import { ProductTemplate } from "../templates/ProductTemplate";
```

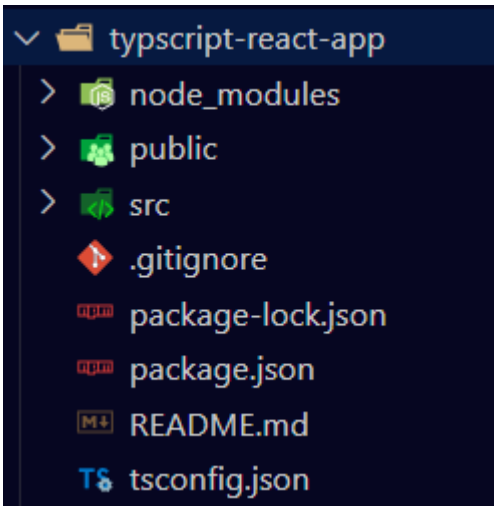
```
export class ProductComponent extends ProductTemplate {
  name = "Samsung TV";
  price = 56000.45;
  qty = 3;
  public total(): number {
    return this.price * this.qty;
  }
  public print(): void {
    console.log(`Product Name: ${this.name}\n
          Price: ${this.price}\n
          Quantity: ${this.qty}\n
          Total: ${this.total()}
          `);
  }
}
```

index.ts

```
import { ProductComponent } from
"../library/components/ProductComponent";

let tv = new ProductComponent();
tv.print();
```

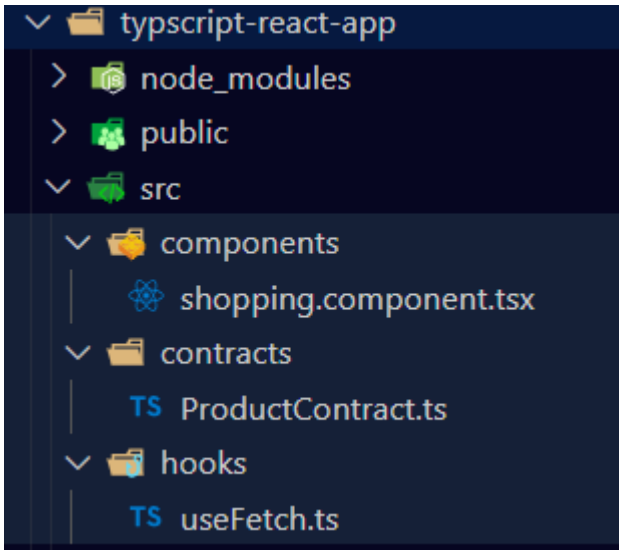
Creating a new React application with TypeScript

- Create a project using below command
  CMD> npx create-react-app appName --tempalate typescript



- File system is same with a new configuration file “tsconfig.json”.
- For component we have to use .tsx extension.
- For modules, contracts, template we have to use .ts extension.

Create the below Folder and File.



ProductContract.ts

```
export interface ProductContract {
  id: number;
  title: string;
  image: string;
  description: string;
  category: string;
  rating: {rate: number; count: number; };
}
```

useFetch.ts

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

export function useFetch(url: string) {
  const [apiData, setApiData] = useState<any[]>([]);

  useEffect(() => {
    fetch(url)
      .then((response) => response.json())
      .then((data) => {
        setApiData(data);
      });
```

```
  }, [url]);

  return apiData;
}
```

shopping.component.tsx

```
import { ProductContract } from "../contracts/ProductContract";
import { useFetch } from "../hooks/useFetch";

export function ShoppingComponent() {
  const categories: string[] = useFetch(
    "http://fakestoreapi.com/products/categories"
  );
  const products: ProductContract[] = useFetch(
    "http://fakestoreapi.com/products"
  );

  return (
    <div style={{ margin: "20px" }}>
      <h2>Shopping Online</h2>
      <div>
        <select name="categories">
          {categories.map((category) => (
            <option key={category} value={category}> {category} </option>
          ))}
        </select>
      </div>
      <ol>
        {products.map((product) => ( <li key={product.id}>{product.title}</li> ))}
      </ol>
    </div>
  );
}
```

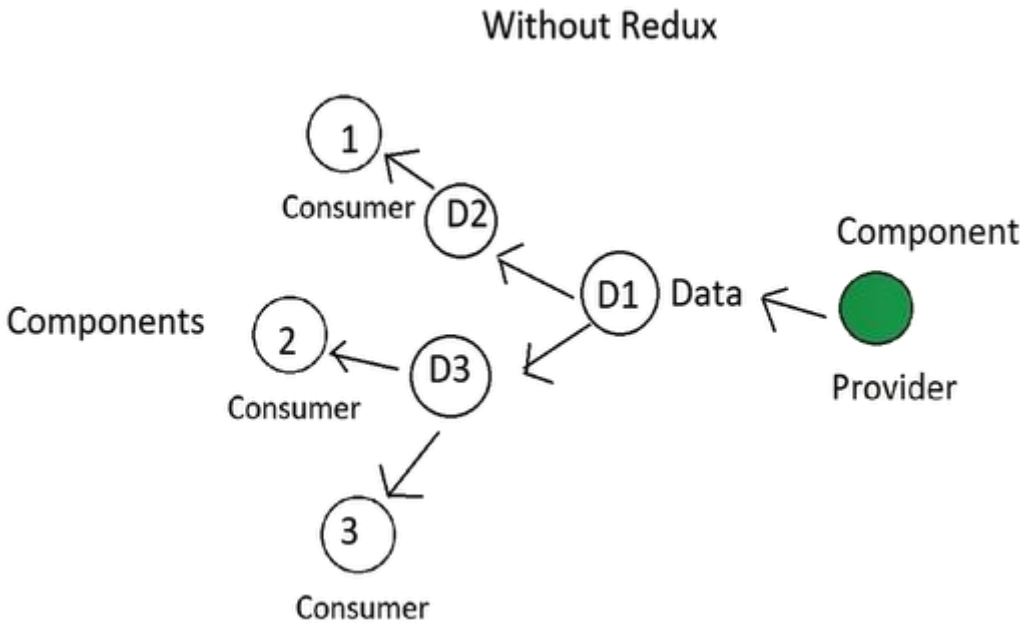
index.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import reportWebVitals from "./reportWebVitals";
import { ShoppingComponent } from "./components/shopping.component";
```

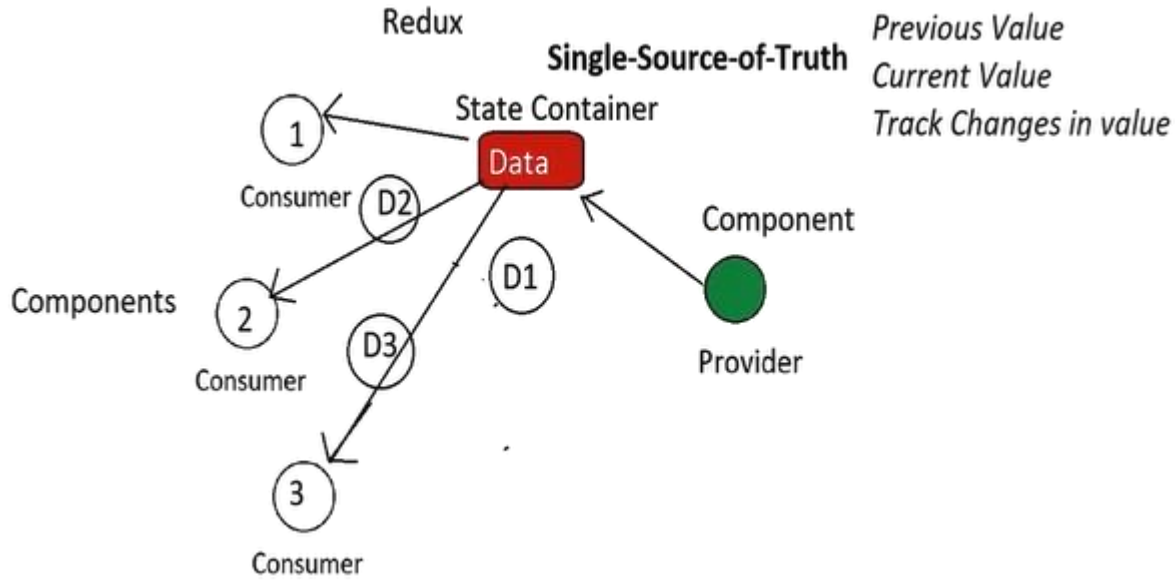
```
const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);

root.render(
  <React.StrictMode>
    <ShoppingComponent />
  </React.StrictMode>
);
```

Redux with React

- Redux is predictable state container for JavaScript apps.
- It is same as “useReducer” in React 18.
- Key terms are,
  - Initial State
  - Reducer
  - Dispatcher
  - State
  - Action
  - Provider



[With out Redux]



[With Redux]

Implementing Redux

1. Install Redux for existing react project
   CMD> npm install @reduxjs/toolkit react-redux --save

2. Create a Redux Slicer
   - Slicer can keep details like initial state, actions, name.
   - Add a new file called “slicer.tsx”

slicer.tsx

```
import { createSlicer } from “redux”;

const initialState = {
    cartItems: [],
    cartItemsCount: 0
};

const cartSlicer = creatSlicer({
    name: “cart”,
    initialState,
    reducers: {
            addToCart: (state: any, action) => {
                    state.cartItems.push(action.payload);
                    state.cartItemsCount = state.cartIterms.length;
            },
```

```
    }
});

export {addToCart} = cartSlicer.actions;
```

3. Configure store by creating a store
    - It is used to access data from slicer and handle the actions from slicer.
    - Add a new file called “store.tsx”.

    store.tsx
    ```
    import {createStore} from “redux”;
    import {cartSlicer} from “…/slicer”

    const store = createStore({
        reducer: {
                store: cartSlicer;
        }
    })
    ```

4. Configure store at application level using provider
    - Go to file “index.tsx”

    Syntax:
    ```
    <Provider store={store}>
        <app/>
    </Provider>
    ```

5. Configure actions and send the data into store
    - Actions are triggered by dispatcher.

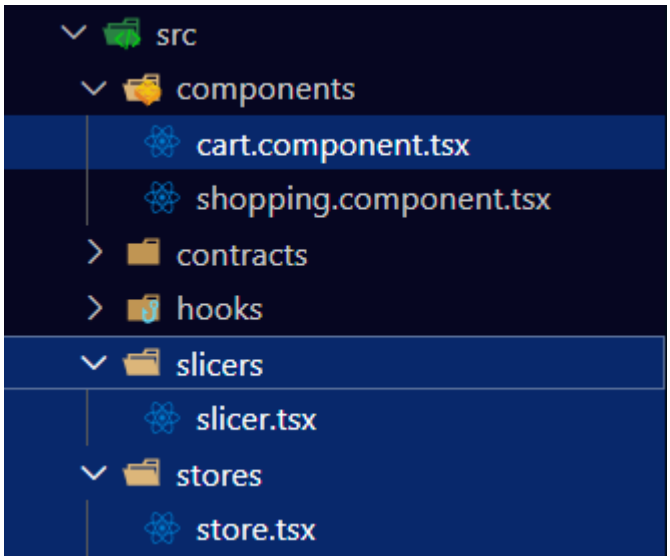
    component.tsx
    ```
    import {useDispatcher} from “redux”;

    const dispatcher = userDispatcher();

    function addToCartClicked() {
        dispatcher(addToCart(product));
    }
    ```

6. To access state from store
   import store from “../stores/store”;

   const count = store.getState().store.cartCount;
   const items:any[] = store.getState.store.cartItems;

   {count}

Create the below Folder and File.



slicer.tsx

```
import { createSlice } from "@reduxjs/toolkit";

const initialState = {
  cartItems: [],
  cartItemsCount: 0
};

const cartSlice = createSlice({
  name: 'cart',
  initialState,
  reducers: {
    addToCart: (state: any, action) => {
      state.cartItems.push(action.payload);
      state.cartItemsCount = state.cartItems.length;
    },
  }
});
```

```
export const {addToCart} = cartSlice.actions;
export default cartSlice.reducer;
```

store.tsx

```
import { configureStore } from "@reduxjs/toolkit";
import cartSlice from "../slicers/slicer";

export default configureStore({
  reducer: {
    store: cartSlice,
  },
});
```

index.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import reportWebVitals from "./reportWebVitals";
import { ShoppingComponent } from "./components/shopping.component";
import store from "./stores/store";
import { Provider } from "react-redux";

const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <ShoppingComponent />
    </Provider>
  </React.StrictMode>
);
```

shopping.component.tsx

```
import { useDispatch } from "react-redux";
import { ProductContract } from "../contracts/ProductContract";
import { useFetch } from "../hooks/useFetch";
import { addToCart } from "../slicers/slicer";
import { CartComponent } from "./cart.component";
```

```
export function ShoppingComponent() {
 const categories: string[] = useFetch(
  "http://fakestoreapi.com/products/categories"
 );

 const products: ProductContract[] = useFetch(
  "http://fakestoreapi.com/products"
 );

 const dispatch = useDispatch();
 function addToCartClick(product: any) {
  dispatch(addToCart(product));
 }

 return (
  <div style={{ margin: "20px" }}>
   <h2> Shopping Online - <CartComponent /> </h2>
   <div>
    <select name="categories">
     {categories.map((category) => (
      <option key={category} value={category}>{category}</option>
     ))}
    </select>
   </div>

   <div style={{ display: "flex", flexDirection: "column" }}>
    {products.map((product) => (
     <div key={product.id}>
      <img src={product.image} alt={product.title}
       style={{ width: "100px", height: "100px" }} />
      <div>
       <button onClick={() => addToCartClick(product)}> Add to cart</button>
      </div>
     </div>
    ))}
   </div>
  </div>
 );
}
```

cart.component.tsx

```
import { useEffect } from "react";
import store from "../stores/store";

export function CartComponent() {
 const count = store.getState().store.cartItemsCount;
 useEffect(() => {}, [count]);

 return (
  <div>
   <button>Your cart [{count}]</button>
  </div>
 );
}
```

FAQ: How to transport data from one component to another?

Ans.

- Props [parent and child], [direct pass params]
- useContext [parent and child at any level with in the context]
  [child elements within the context provider scope]
- useReducer [between requests of same component]
- useCookies [across components]
- Redux store [across components]
- useParam [route parameter across components]

React Native

- It is for making “Android and iOS” apps richer.
- There are various frameworks used for making mobile applications more interactive.
  - Apache Cordova
  - Ionic
  - Native Script
  - Flutter
  - React Native
- React uses HTML component which are not suitable or not native for mobile environment.
- React Native provides component suitable for mobile platforms.
- React Native components [Link].
- React Native uses “expo” library.

FAQ: What is difference between mobile and distributed apps?
Ans. Netflix, Amazon, Whatsapp, Skyp – distributed apps.
Calculator, calendar, phone box, gallery – mobile apps.

Create React Native Application

1. Run the below command to create a project
   CMD> npx create-expo-app react-mobile-app

2. To start the project
   CMD> npx expo statrt

Web Pack

- Web pack is a bundler.
- Bundling is the process of bundle all resources file and configure a mapping for application.
- Official site Link.

1. Create a new folder for react-application
   E:\react-webpack-app

2. Open in VS code and run the following command in terminal
   CMS> npm inti -y

   This will add “package.json” [metadata]

3. Add the following folders into project
   - public [static resources – images, text, html, ...]
   - src [dynamic resources – css, js, ts, sass, …]

4. Add following files into project
   - public/index.html
   - src/index.js

5. Install the following libraries
   CMD> npm install webpack webpack-cli --save-dev
   CMD> npm install html-webpack-plugin --save-dev
   CMD> npm install --save-dev webpack-dev-server
   CMD> npm install @babel/core @babel/preset-env @babel/preset-react babel-loader --save-dev
   CMD> npm install react react-dom --save

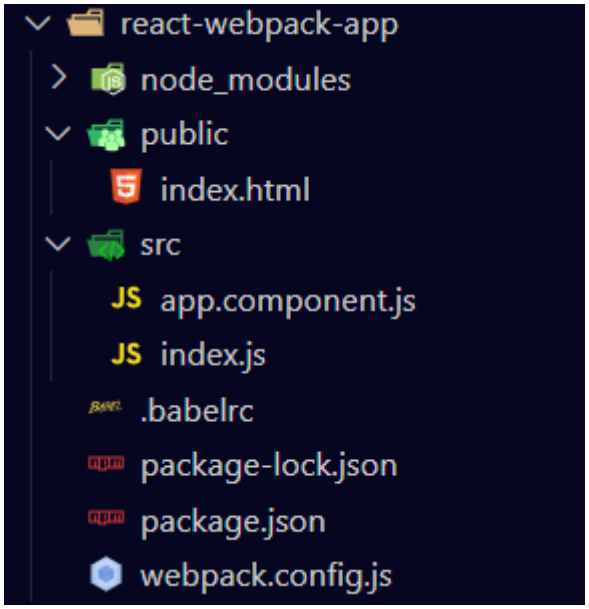
6. Create webpack configure file name webpack.config.js [Link].
   a. We can create manually and we can keep the config code.
   b. We can use command to create automatically by answering few questions [Link].

7. Create babel configuration file babelrc [Link].

8. Go to package.json and add below settings.
   "start": "webpack serve --mode development",
   "build": "webpack --mode production"

9. To run CMD> npm start
   To build CMD> npm run build

Create the below Folder and File.



index.js

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>React Web Pack</title>
</head>

<body>
```

```
  <noscript>Please Enable JavaScript</noscript>
  <div id="root"></div>
</body>

</html>
```

webpack.config.js

```
const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
 entry: "./src/index.js",
 output: {
  filename: "main.js",
  path: path.resolve(__dirname, "build"),
 },
 plugins: [
  new HtmlWebpackPlugin({
   template: path.join(__dirname, "public", "index.html"),
  }),
 ],
 devServer: {
  static: {
   directory: path.join(__dirname, "build"),
  },
  port: 8090,
 },
 module: {
  rules: [
   {
    test: /\.(js)$/,
    use: ["babel-loader"],
    exclude: /node_modules/,
   },
  ],
 },
 resolve: {
  extensions: ["*", ".js"],
 },
};
```

package.json

```
{
  "name": "react-webpack-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "webpack serve --mode development",
    "build": "webpack --mode production"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "@babel/core": "^7.26.10",
    "@babel/preset-env": "^7.26.9",
    "@babel/preset-react": "^7.26.3",
    "babel-loader": "^10.0.0",
    "html-webpack-plugin": "^5.6.3",
    "webpack": "^5.98.0",
    "webpack-cli": "^6.0.1",
    "webpack-dev-server": "^5.2.1"
  },
  "dependencies": {
    "react": "^19.1.0",
    "react-dom": "^19.1.0"
  }
}
```

.babelrc

```
{
  "presets": [
    "@babel/preset-env",
    [
      "@babel/preset-react",
      {
        "runtime": "automatic"
      }
    ]
```

```
 ]
}
```

app.component.js

```
export function AppComponent() {
 return (
  <div>
   <h2>Welcome to React</h2>
   <p>Application created with Web pack</p>
  </div>
 );
}
```

index.js

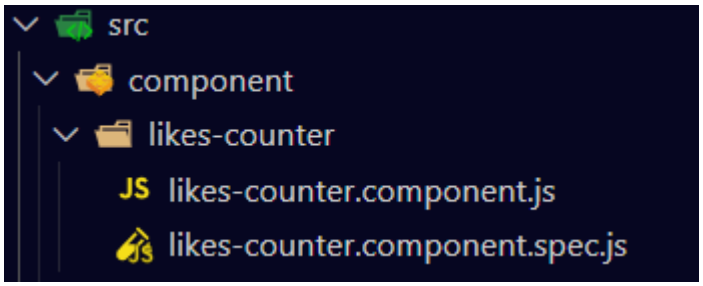
```
import React from "react";
import ReactDOM from "react-dom/client";
import { AppComponent } from "./app.component";

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

Testing React Application

- Project comprises of AS-IS and TO-BE
- AS-IS refers to what our developer have designed
- TO-BE refers to what client requirement is,
- AS-IS == TO-BE then test has passed.
- AS-IS !== TO-BE then test has failed.
- There are various testing frameworks
  - Jasmine – Karma
  - JEST
  - Protractor etc.
- NPX create-react-app is configured with a testing framework “JEST”.
- Test testing libraries
  "@testing-library/jest-dom"
  "@testing-library/react"
  "@testing-library/user-event"
- Testing every component have 3 phases
  - Arrange : Configure and import the component to test.
  - Act : Define the functionality to test.
  - Assert : Verifies the results and reports pass or fail.

- In JEST framework the above phase is managed by methods
  - it() : Configures
  - test() : defines functionality to test
  - expect() : assert
- JEST provides the following methods for configuration and testing
  - Screen : get the content on component UI
  - render : rendering the component which you have to test.
  - fireEvent : testing events
  - date-testid
  - getTestId
- JEST official document Link.

Create the below Folder and File.



likes-counter.component.js

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

export function LikesCounter() {
 const [counter, setCounter] = useState(0);

 function handleLike() {
  setCounter(counter + 1);
 }

 function handleDislike() {
  setCounter(counter - 1);
 }

 return (
  <div>
   <span data-testid="counter">{counter}</span>
   <div>
    <button data-testid="like" onClick={handleLike}>Like</button>
    <button data-testid="dislike" onClick={handleDislike}>Dislike</button>
```

```
    </div>
   </div>
  );
}
```

likes-counter.component.spec.js

```
import { screen, render, fireEvent } from "@testing-library/react";
import { LikesCounter } from "./likes-counter.component";

test("Likes Counter Test", () => {
  render(<LikesCounter />);
  const counter = screen.getByTestId("counter");
  const like = screen.getByTestId("like");

  expect(counter).toBeInTheDocument();
  fireEvent.click(like);
  expect(counter.textContent).toBe("1");
});

test("Check for Link", () => {
  render(<LikesCounter />);
  const link = screen.getByRole("link");
  expect(link).toBeInTheDocument();
});
```

Build and Deploy

- To Build the application
  CMD> npm run build

- Install firebase cloud tools
  CMD> npm install -g firebase/tools

- Open Google firebase console [Link].

- Create a project on firebase console react-shopper-template

- Login to firebase
  CMD> firebase login

- Now build the project

CMD> npm run build

- Now run the below command for Firebase initialization
  CMD> firebase init

- Answer the question as highlighted below.
  - Are you ready to proceed? **Yes**
  - Which Firebase features do you want to set up for this directory? Press Space to select features, then enter to confirm your choices. **Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys**
  - Please select an option: **Use an existing project**
  - Select a default Firebase project for this directory**: react-shopper-template-f3606**
  - What do you want to use as your public directory? **Build**
  - Configure as a single-page app (rewrite all urls to /index.html)? **No**
  - Set up automatic builds and deploys with GitHub? **No**
  - File build/index.html already exists. Overwrite? **No**

- Now deploy the project to firebase
  CMD> firebase deploy

- Now you can see deployment completed and like is there to access the application globally.

```
=== Deploying to 'react-shopper-template-f3606'...

i  deploying hosting
i  hosting[react-shopper-template-f3606]: beginning deploy...
i  hosting[react-shopper-template-f3606]: found 17 files in build
+  hosting[react-shopper-template-f3606]: file upload complete
i  hosting[react-shopper-template-f3606]: finalizing version...
+  hosting[react-shopper-template-f3606]: version finalized
i  hosting[react-shopper-template-f3606]: releasing new version...
+  hosting[react-shopper-template-f3606]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/react-shopper-template-f3606/overview
Hosting URL: https://react-shopper-template-f3606.web.app
```

Note: After any change we have build and deploy no need to do any other steps.

All React Commands Cheat sheet

To Create New Application

CMD> npx create-react-app <application_name>

To Start the Application

CMD> npm start

To Install Bootstrap

CMD> npm install bootstrap --save
CMD> npm install bootstrap-icons --save

To Link bootstrap and icons css

```
import '../node_modules/bootstrap/dist/css/bootstrap.css'
import '../node_modules/bootstrap-icons/font/bootstrap-icons.css';
```

To Install jQuery

CMD> npm install jquery -- save

To import jQuery

```
import $ from “jquery”;
```

To Install Axios

CMD> npm install axios -- save

To import axios

```
import axios from “axios”;
```

To Install Formik

CMD> npm install formik -- save

To import formik for normal validation

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

To import formik components for formik validation

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

To Install Yup

CMD> npm install yup -- save

To import yup

```
import * as yup from “yup”;
```

To Install React Router DOM

CMD > npm install react-router-dom -- save

OR

CMD> npm install react-router-dom --save -D [latest stable Dev]
CMD> npm install react-router-dom@latest [latest not stable]
CMD> npm install react-router-dom@v5 [version 5]

To import react router dom

```
import { BrowserRouter, Link, Route, Routes } from
                                        "react-router-dom";
```

To install MongoDB

CMD> npm install mongodb --save

To install Express and cors

CMD> npm install express --save
CMD> npm install cors --save

To install Cookies

CMD> npm install react-cookie --save

To install Material UI

CMD> npm install @mui/material -- save
CMD> npm install @emotion/react -- save
CMD> npm install @emotion/styled -- save

To install TypeScript

CMD> npm install -g typescript
CMD> tsc -v

To create an application using typescript

CMD> npx create-react-app appName --tempalate typescript

To install Redux for existing react project

CMD> npm install @reduxjs/toolkit react-redux --save

To create new application using typescript and redux

CMD> npx create-react-app <app-name> --template redux-typescript

To create React Native application

CMD> npx create-expo-app <app-name>

To install firebase cloud tools

CMD> npm install -g firebase/tools

To Login to firebase from your project

CMD> firebase login

To Firebase initialization

CMD> firebase init

To deploy the project to firebase

CMD> firebase deploy

---------- The END ----------