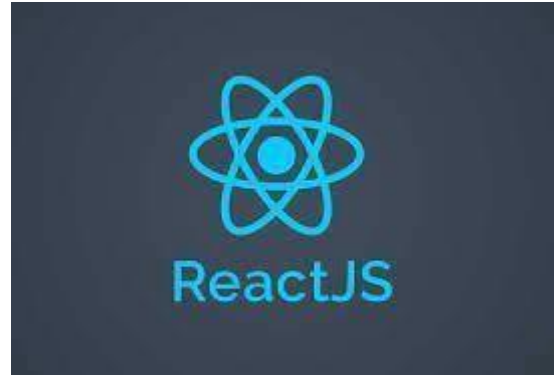


ReactJS

- ✓ ReactJS is **JavaScript library**
- ✓ ReactJS Released by **Facebook**
- ✓ Current version of ReactJS is **18.X**
- ✓ ReactJS used to develop **web applications**
- ✓ We will develop ReactJS Applications in **two ways**
 - 1) **JSX**
 - 2) **TSX**

**Features of ReactJS****1) Components**

- ✓ Each Partition of webpage called as Component
- ✓ We can create more than one component
- ✓ Components are reusable
- ✓ We can provide communication between components
- ✓ We can create components in two ways
 - 1) **Class Components**
 - 2) **Functional Components**

Example

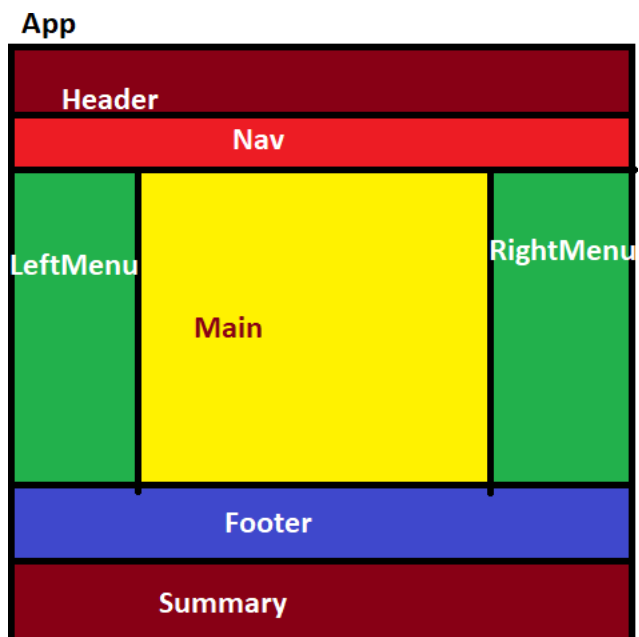
In this diagram we have **following Components**

ParentComponent

- 1) AppComponent

ChildComponent

- 1) HeaderComponent
- 2) NavComponent
- 3) LeftMenuComponent
- 4) MainComponent
- 5) RightMenuComponent
- 6) FooterComponent
- 7) SummaryComponent



Differences Between class level components and functional level components

Functional Components	Class Components
1) We can't create object to functional components	We can create object to class components
2) These components are also called as temporary components	These components are existed at the end of process (Container Components)
3) These components are called as stateless components	These components are called as stateful components
4) Supports Hooks	Supports life cycle methods
5) We can't reuse stateless components	We can reuse stateful components
6) These components are easy to understand	These components are difficult to understand
7) We will write html in return method	We will write html in render () hook

2) JSX

- ✓ JSX Stands for **JavaScript XML**
- ✓ JSX allows us to write **HTML** directly within the **JavaScript code**
- ✓ JSX used to implement **React Applications**
- ✓ JSX is an extension of the **JavaScript language**
- ✓ **Babel** will convert **JSX Expressions/Syntax** to Actual **JavaScript Code**

What are the Differences between JSX and TSX?

JSX	TSX
JSX Stands for JavaScript XML	TSX Stands for TypeScript XML
JSX Wont follows OOPS	TSX follows OOPS
JSX is not strict type var x=100;	TSX is strict Type var x: number=100;
In JSX we will write HTML into JavaScript	In TSX we will write HTML to TypeScript

3) State

- ✓ state is predefined object
- ✓ state used to store the component data
- ✓ state is mutable
- ✓ whenever state changes automatically component re-renders

**state in functional components**

- ✓ `useState ()` is the hook used to define state in functional components
- ✓ `useState ()` hook will return array

Examples:**number**

```
const [num, setNum] =useState (100);  
<h1>{num}</h1>   Output: 100  
setNum (200);  
<h1>{num}</h1>   Output: 200
```

string

```
const [str, setStr] =useState(`ReactJS`);  
<p>{str}</p>      //ReactJS  
setStr (`ReactJS18.X`);  
<p>{str}</p>      //ReactJS18.X
```

boolean

```
const [flag, setFlag] =useState(true);  
<h1>{flag}</h1>   //true  
const [flag1,setFlag1] =useState(false);  
<h1>{flag1} </h1> //false
```

conditional rendering

```
const [x, setX] =useState(`Java`);  
const [y, setY] =useState(`ReactJS`);  
const [z,setZ]= useState(true);  
{  
  z? <h1>{x}</h1>:<h1>{y}</h1>  
}  
//Java
```

lists

```
const [arr1, setArr1] =useState ([10,20,30,40,50]);  
{  
  arr1.map ((element, index) => {  
    return (<h1 key={index}>{element}</h1>)  
  })  
}
```

map () method used to iterate **list items** in **reactjs**
key property used to **tract** list items

json object

```
const [obj, setObj] =useState({key1: `Hello_1`,key2:`Hello_2`,key3:`Hello_3`});
```

```
<h1>{obj.key1} .... {obj.key2} ...{obj.key3} </h1>
//Hello_1.... Hello_2.... Hello_3
```

Array of objects

```
const [products, setProducts] =useState ([{p_id:111,
p_name:'p_one',p_cost:10000},
{ p_id:222,p_name:'p_two',p_cost:20000},
{ p_id:333,p_name:'p_three',p_cost:30000},
{p_id:444, p_name:'p_four', p_cost:40000},
{p_id:555, p_name:'p_five', p_cost:50000}]]);
```

```
<table>
  <tr>
    <th>p_id</th>
    <th>p_name</th>
    <th>p_cost</th>
  </tr>
  {
    products.map ((element, index) => {
      return (<tr key={index}>
        <td>{element. p_id} </td>
        <td> {element. p_name} </td>
        <td> {element. p_cost} </td>
      </tr>)
    })
  }
</table>
```

State in class level components

- ✓ state is predefined object used to define state in class level components

Ex.

```
this. state= {
  num:100
}
```

```
<h1>{this.state.num} </h1>   Output:100
```

- ✓ setState () is the predefined method used to change the state in class level components

```
this. setState (() => {
  num: 200
})
```

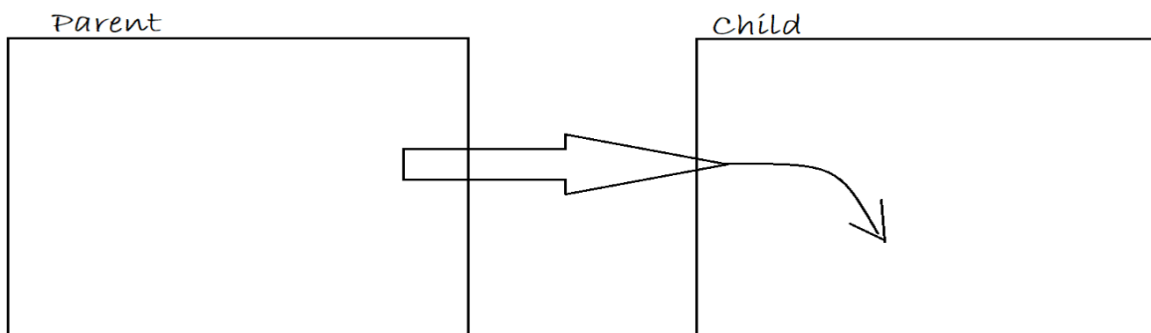
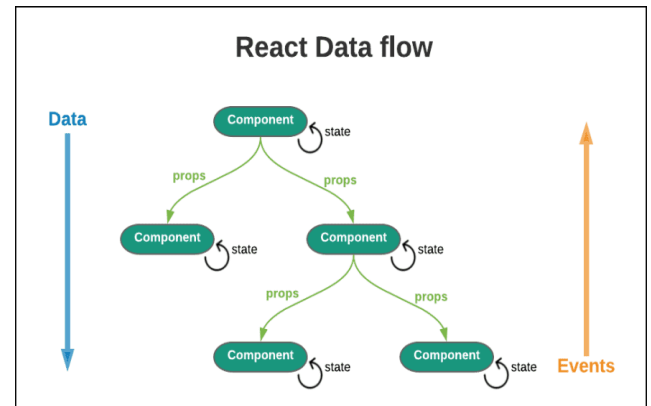
```
<h1>{this.state.num} </h1>   Output:200
```

CareerIT

REACTJS

4) props

- ✓ props are the predefined object in reactjs
- ✓ props are used to provide communication between components
- ✓ child component receives the data from parent component with the help of props
- ✓ props are immutable



Parent

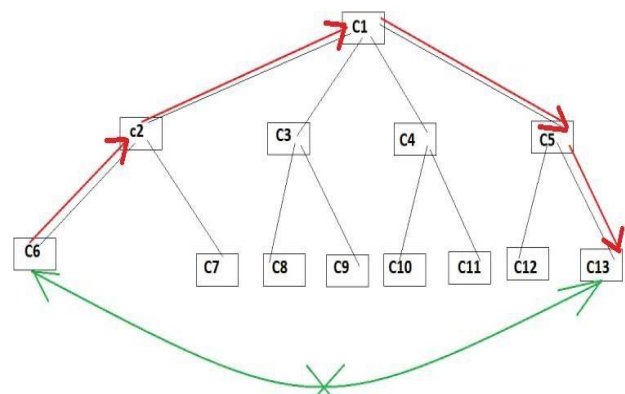
```
<Child key1=" Careerit">
</Child>
```

Child

```
<h1>{props.key1} </h1>
Output: Careerit
```

Props Drilling

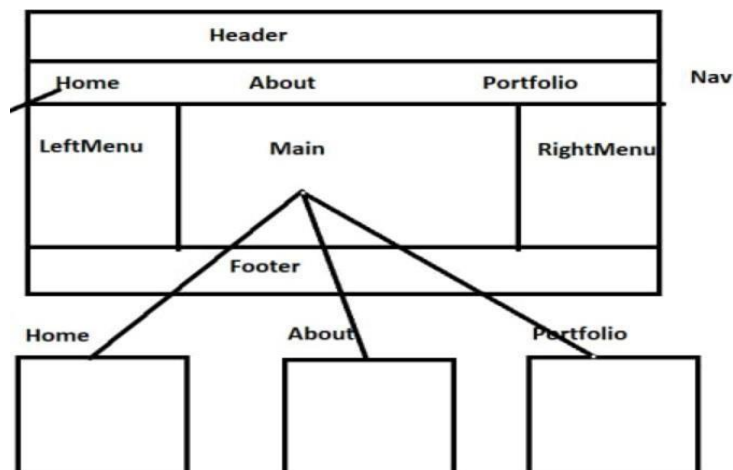
- ✓ sharing data from **source component** to **destination component** through several **interdependent components** called as **props drilling**



- ✓ In above diagram following **c6 component** is **source component**
- ✓ In above diagram following **c13 component** is **destination component**
- ✓ C2, C3, C5 and C13 components are **interdependent components**
- ✓ props drilling never **recommended** in application development
- ✓ **state management** is used to overcome props drilling
- ✓ we can implement state management in two ways
 - 1) Context API
 - 2) Redux

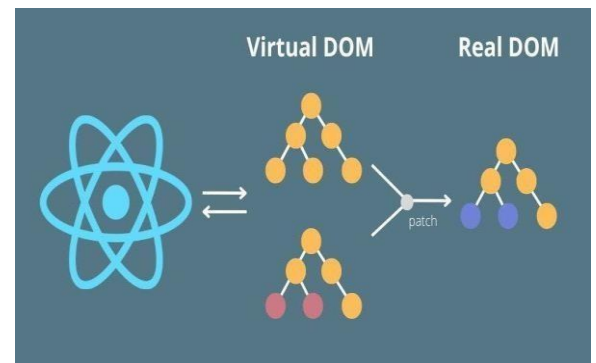
5) Single Page Applications

- ✓ Dynamically rewrites the **component content** from **server** without **refreshing** called as **Single Page Application**
- ✓ **Navigation of one component to another component** without **refreshing** in single page application called as **Routing**
(or)
- ✓ **Binding** an URL to component called as **Routing**
- ✓ **Route** is used to implement **Routings** in Single Page Application
- ✓ **Routes** is used to encapsulate all **child Routes**
- ✓ **BrowserRouter** is used for handling the **dynamic URL**.
- ✓ **HashRouter** is used for handling the **static request**
- ✓ **useParams ()** is the hook used to handle **Routing Parameters**
- ✓ **useNavigate ()** is the hook used to **navigate** from one component to another component in single page applications
- ✓ a path consisting of two asterisks (******) called as **Wildcard route in react**
- ✓ if no route matches automatically **Wildcard route will execute**
- ✓ default route endpoint equals to **"/"**
- ✓ **Outlet** allows child components to **render**
- ✓ **Outlet** behaves like **place holder**
- ✓ **React-router-dom@6** package required to implement single page applications
- ✓ We will download above package in two ways
 - 1) yarn
 - 2) npm
- ✓ below command used to download with the help of yarn
 - yarn add react-router-dom@6
- ✓ npm stands for **node packaging manager**
- ✓ below command used to download with the help of **npm**
 - npm install react-router-dom@6



6) virtual DOM

- ✓ A virtual DOM is a lightweight JavaScript representation of the Document Object Model (DOM)
- ✓ Updating the virtual DOM is comparatively faster than updating the actual DOM
- ✓ In Virtual DOM, only Changed Element will Reload Instead of All Elements
- ✓ Because of Virtual DOM Application Performance Increases



index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Virtual DOM</title>
  </head>
  <body>
    <div id="app1"></div>
    <br><br>
    <script src="index.js"></script>
  </body>
</html>
```

index.js

```
setInterval (() => {
  const element1=`
    <div>
      <div>
        Hello World!!!
      </div>
      <div>
        <input type="text" />
      </div>
      <div>
        ${new Date (). toLocaleTimeString ()}
      </div>
    </div>
  `;
  document.getElementById("app1").innerHTML=element1;
},1000);
```

Hello World !!!

10:05:16 AM



Note1: all divisions are refreshing

Note2: Application performance Degrad

ReactJS

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Virtual DOM</title>
  </head>
  <body>
    <div id="app2"></div>
    <script src="index.js"></script>

    <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>

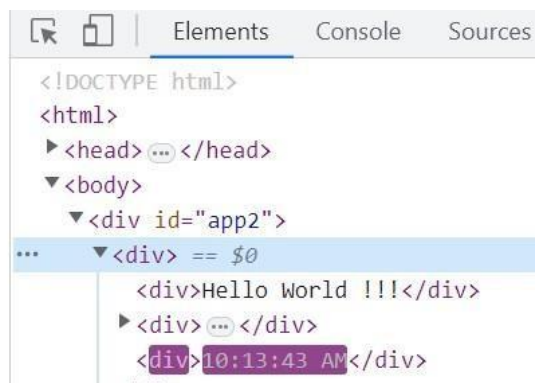
  </body>
</html>
```

Index.js

```
setInterval(() => {
  const element2=React.createElement('div', null,
    React.createElement('div',null,'Hello World !!!'),
    React.createElement('div',null,React.createElement('input',{type:'text'
  })),
    React.createElement('div',null,new Date().toLocaleTimeString()));
  ReactDOM.render(element2,document.getElementById("app2"));
},1000);
```

Hello World !!!

10:13:43 AM



Note: only one div will refresh instead of all divs

7) React Element

- ✓ Elements are the **smallest building blocks** of React apps
- ✓ An element describes **what we want to see** on the screen
- ✓ React elements are **plain objects**
- ✓ **React DOM** takes care of **updating the DOM** to match the **React elements**.

Example

```
const root = ReactDOM.createRoot(  
  document.getElementById('root')  
);  
const element = <h1>Hello, world</h1>;  
root.render(element);
```

8) Expressions/Interpolation/Data Binding

- ✓ {} called as Expressions/Interpolation/Data Binding
- ✓ Whatever we written inside expression will be evaluate

Ex.

```
const [str, setStr] =useState("Careerit");  
<h1>{str}</h1>           //Output: Careerit
```

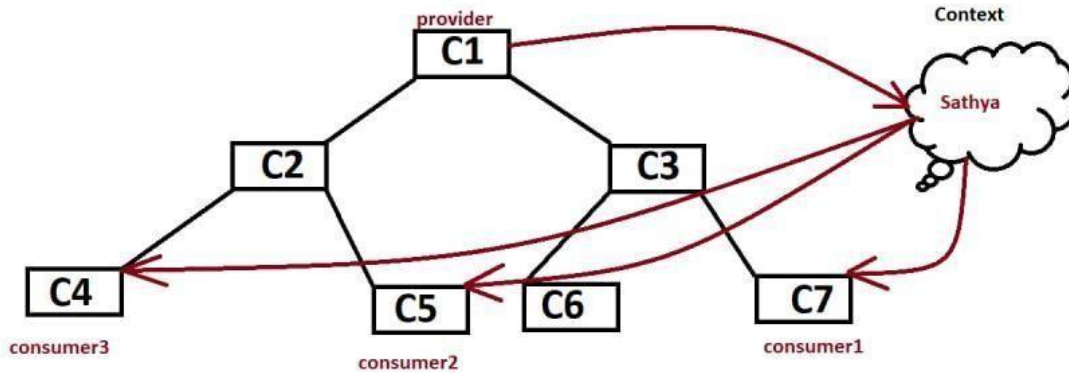
```
<h1> {10+10} </h1>    //Output:20
```

9) State Management

- ✓ State Management is the Technique
- ✓ It is used to provide **communication** between components and **sharing** data between components
- ✓ We can implement State management in two ways
 - 1) Context API
 - 2) Redux

Context API

- ✓ Context API used to implement State Management in React Applications
- ✓ Context API overcomes the props drilling
- ✓ By using Context API, we can easily transfer data between components
- ✓ Provider will store data to context
- ✓ Consumer will consume data from context
- ✓ In context API we will use Following Hooks
 - 1) createContext ()
 - 2) useContext ()
- ✓ createContext () used to store the data to context
- ✓ useContext () used to read the data from context



Advantages of Context API

- ✓ Context API is inbuilt tool in React library
- ✓ Bundle Size Never increases
- ✓ Requires Minimum Setup for Context API Integration
- ✓ Context API Suitable for Static Data
- ✓ Context API Suitable for Small Scale Web Applications

Disadvantages of Context API

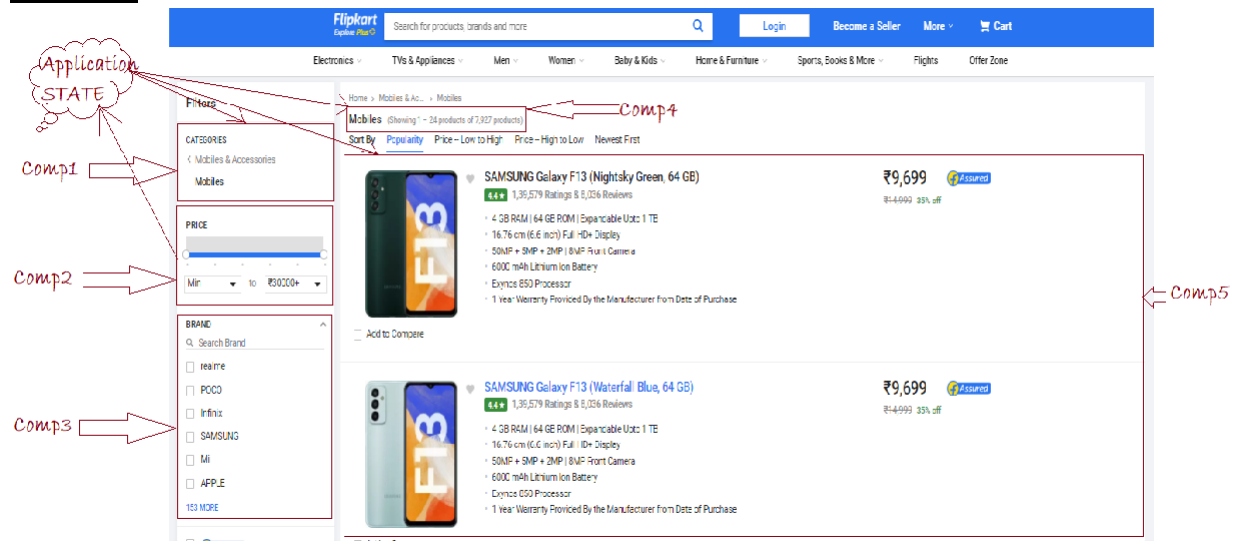
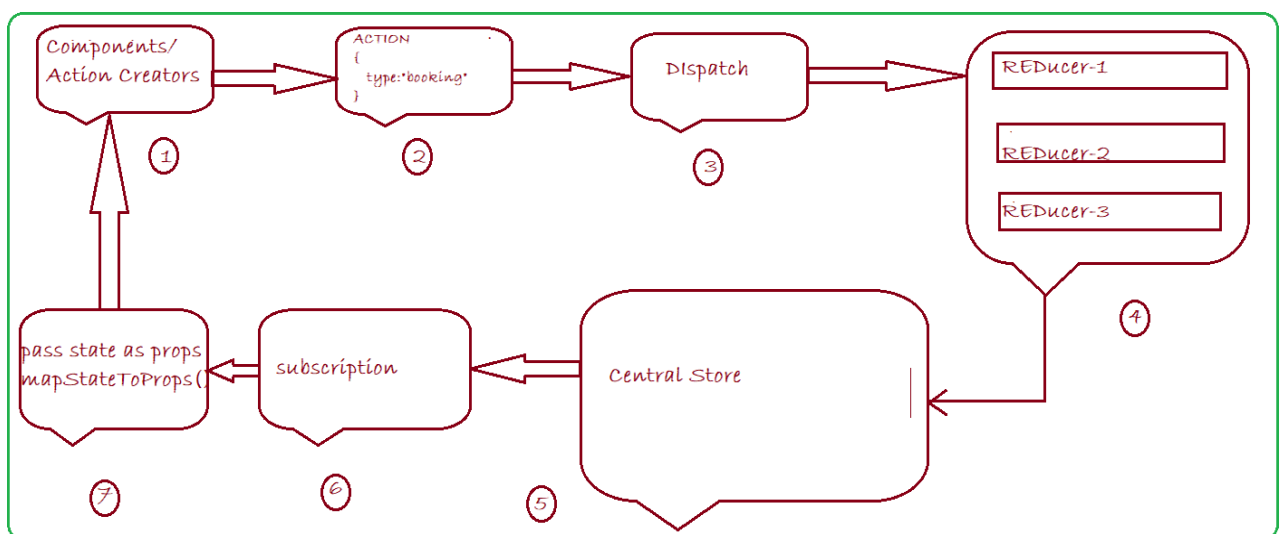
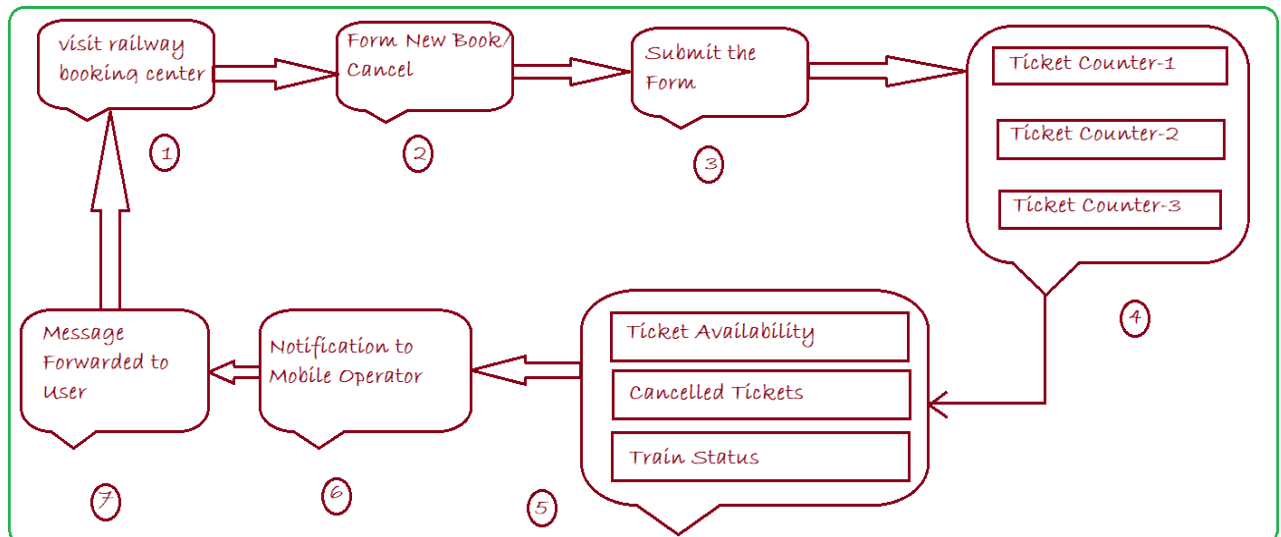
- ✓ Debugging is Difficult
- ✓ Not Suitable for Medium and Large-Scale Web Applications
- ✓ Not Suitable for Dynamic Data (Frequently updating)

Redux

Requirement

- 1) Cart in Ecommerce Application

Problem

Solution**Example**✓ **Railway Booking System**

Introduction

- ✓ Redux is a predictable state container for JavaScript apps.
- ✓ Redux used to implement state management in react applications
- ✓ Redux is 3rd party library
- ✓ Redux we can integrate with Angular, React, VueJS and VenillaJS

Principles of Redux

1. Single Store (Object Tree)
2. State is read-only (Only Change State through Actions)
3. Changes are made with pure functions

Advantages of Redux

- ✓ Suitable for Medium and Large Scaled Web Applications
- ✓ Predictable
- ✓ Centralized
- ✓ Debuggable
- ✓ Flexible
- ✓ Suitable for Dynamic Data (Frequent Updates) Ex. Cart

Disadvantages of Redux

- ✓ Bundle Size Increases
- ✓ Steep learning curve
- ✓ Boilerplate code

Analogy of Redux

- ✓ Action
- ✓ Action Creators
- ✓ Reducers
- ✓ Store
- ✓ Subscribe
- ✓ Dispatch

Action

- ✓ source of information to store called as Action
 - ✓ Action is plain JavaScript object/JSON Object
- Ex.

```
let actionObj = {  
    type: "ADD",  
    payload: 10  
};
```

Action Creators

- ✓ Action Creators are function will return Actions
- ✓ We can achieve actions reusability through Action Creators

Reducers

- ✓ We will write Business Logic in Reducers
- ✓ We can create more than one reducer
- ✓ Application Readability and Modularity increases with multiple reducers
- ✓ Reducer takes two parameters as input i.e., **action** and **previous state** and return **new state**

Store

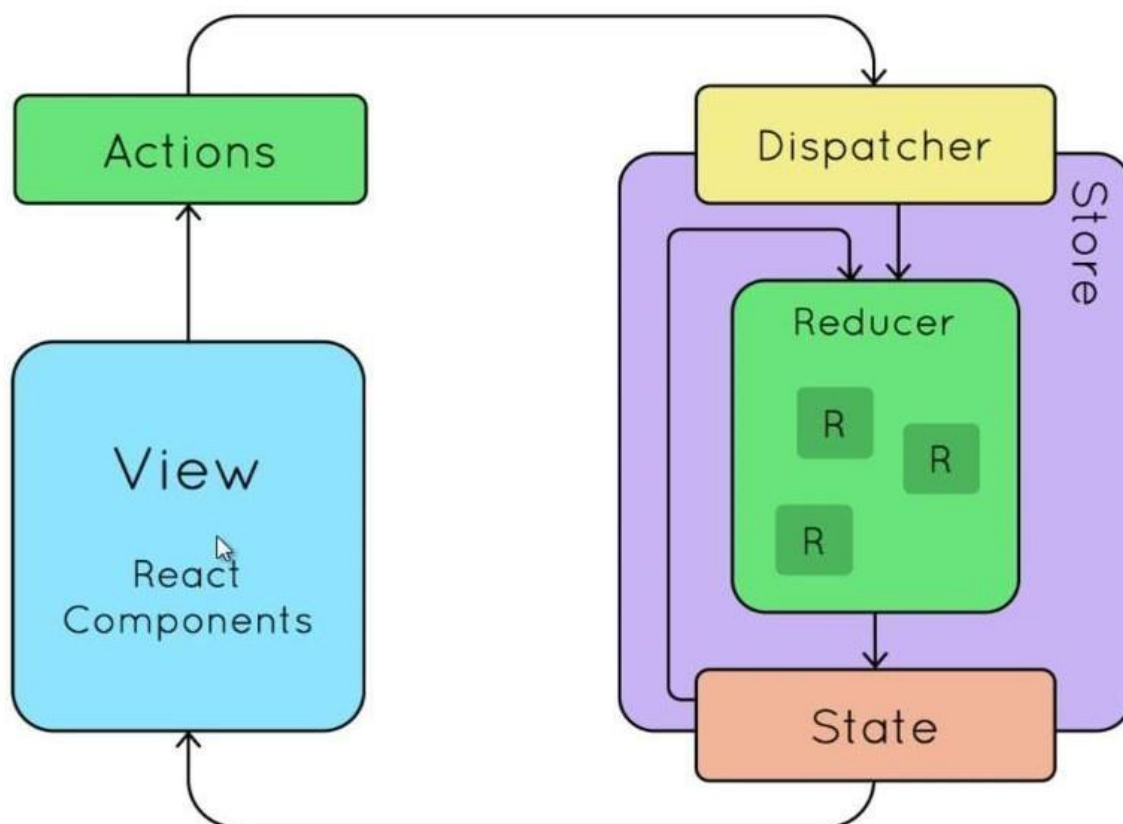
- ✓ Store is the main building block in Redux Architecture
- ✓ Store Accommodates both Reducer and Application State
- ✓ We can have only one Store (Application State)

Subscribe

- ✓ The process of Receiving new state from store called as Subscribe
- ✓ Component receives “**new state**” as props with the help `mapStateToProps ()` method

dispatch

- ✓ Process of sending actions to store called as dispatch
- ✓ `mapDispatchToProps ()` method used to perform dispatch operation

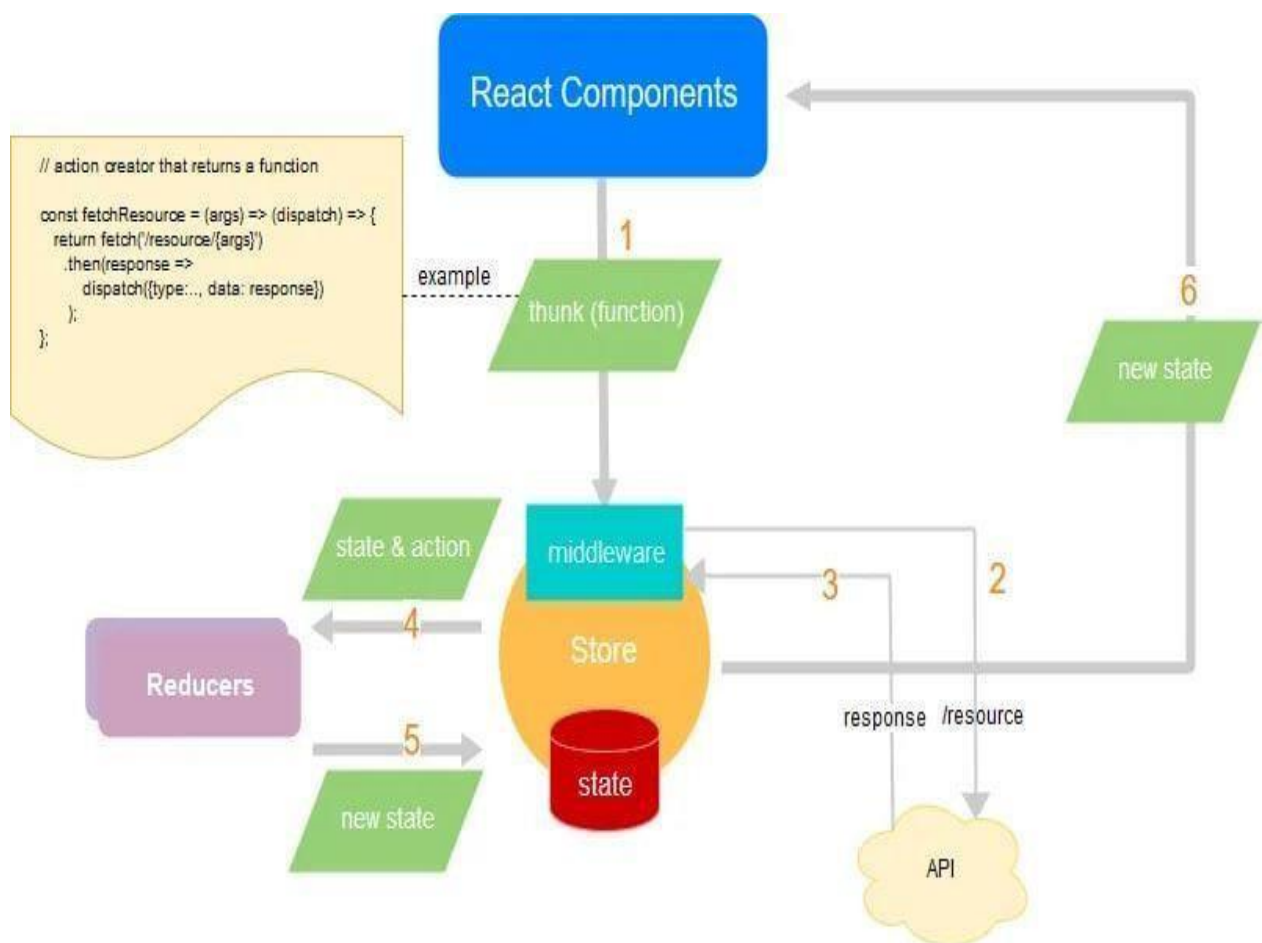
Redux Architecture

Middleware's

- ✓ Middleware's intercepts the Actions Before Reaching Reducers/Store
- ✓ Middleware's can change/cancel the Actions before Reaching Reducers/Store
- ✓ Redux Supports two Middleware's
 - 1) Thunk
 - 2) Saga

Thunk Middleware

- ✓ Thunk is Middleware
- ✓ Thunk Middleware used to delay calculations and evaluation of any operations in Redux Architecture
- ✓ Action Creator will return a function instead of object
- ✓ Returned function will receive two methods from store
 - 1) dispatch
 - 2) getState
- ✓ dispatch method used to make synchronous operation after successful completion of asynchronous operations
- ✓ getState () method used to access the state from store



Saga Middleware

- ✓ Saga is a Middleware
- ✓ Saga Middleware allows store to interact with external resources asynchronously

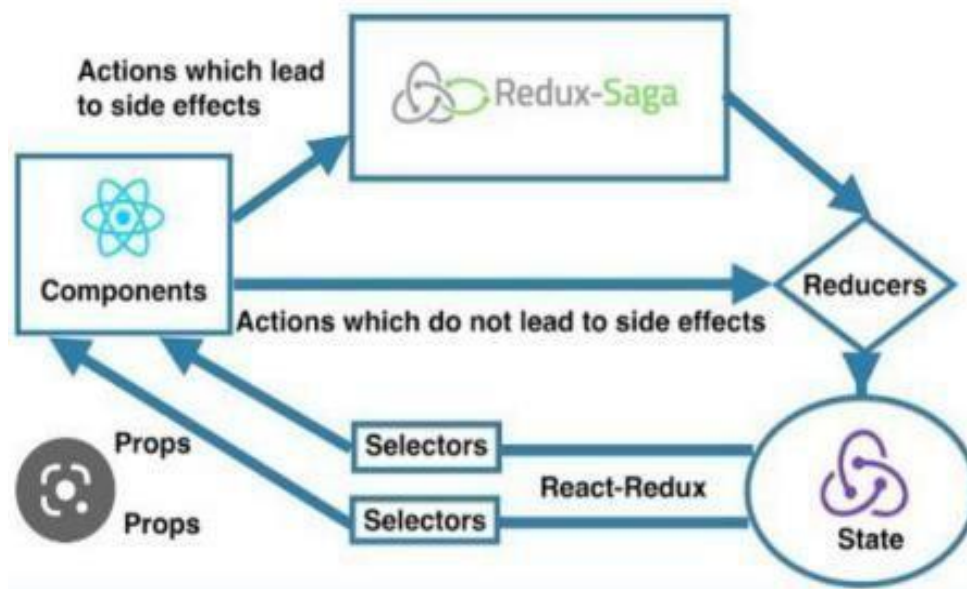
Ex.

Making Http Requests

Accessing Browser local Storage

Execution of I/O Operations

- ✓ Above Examples called as Side Effects
- ✓ In Saga Middleware we will use Generator functions
- ✓ Generator functions are introduced in ES6 version

**PRACTICE PAPER-1**

1) What is React?

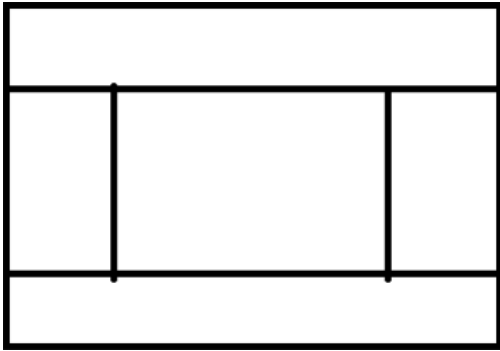
a) Library b) Framework c) both a&b d) None

2) What is Component? write minimum 5 Points

Ans:

3) Identify components in below diagram

4) Identify components in below diagram



- 5) in how many ways can we create components? _____
- 6) are functional components are called as stateless components? _____
- 7) are class components are called as stateful components? _____
- 8) How to enhance functional components _____
- 9) how to manage life cycle of class components _____
- 10) JSX Stands for _____
- 11) TSX Stands for _____
- 12) write the differences between JSX and TSX?

13) write the differences between functional components and class components in reactjs?

- 14) in JSX, we will write HTML to JavaScript? _____
15) in TSX, we will write TypeScript to HTML? _____
16) how to tract elements in lists? _____
17) are React Applications Component Based? _____
18) are Components Reusable? _____
19) can we provide Communication Between Components? _____

PRACTICE PAPER-2

- 1) What is State? Write few points

Ans:

- 2) Explain useState () hook in React

Ans:

- 3) const [__, setX] = useState("Hello");
4) const [_____ , setNum] = useState (100);
5) const [bool, _____] = useState(true);
6) store value **100 to x** variable
store value **200 to y** variable
Find addition of x and y and store to **z variable**

Ans:

- 7) is state mutable?

a) yes b) no c) may be d) all

- 8) store value **Careerit to x** variable

store value **Technologies to y** variable

concat both x and y and display result with interpolation

Ans:

- 9) write the Syntax for conditional rendering

Ans:

- 10) const [x, setX] =useState(`Java`);

```
const [y, setY] = useState(`ReactJS`);  
const [z, setZ] = useState(true);  
{  
  z? <h1>{x}</h1>:<h1>{y}</h1>  
}
```

Ans:

11) how to iterate lists in React?

a) map () b) for () c) forEach () d) for...of()

12) iterate following list

```
const [arr1, setArr1] = useState ([10,20,30,40,50]);
```

Ans:

13) how to track elements in list _____

14) iterate following list

```
const [arr2, setArr2] = useState ([“React”, “ Angular”, “ VueJS”, “ MongoDB”, “  
NodeJS”]);
```

Ans:

15) Read Data from following JSON Object

```
const [obj, setObj] =useState ({frontend:‘React’, backend:‘Boot’, database:‘MongoDB’});
```

Ans:

16) Calculate the TA, DA, HRA and PF on Salary

TA --- 7% DA --- 9% HRA --- 12% PF --- 15%

```
const [salary, setSalary] =useState (30000);
```

Ans:

17) Identify Expression/Interpolation in React

a) {} b) [] c) both a & b d) None

18) how to change state in **Class Level Components** _____

19) display following data in the form of a table

```
const [products, setProducts] =useState ([
  {p_id:111, p_name: 'p_one', p_cost:10000},
  {p_id:222, p_name:'p_two', p_cost:20000},
  {p_id:333,p_name:'p_three',p_cost:30000},
  {p_id:444,p_name:'p_four',p_cost:40000},
  {p_id:555,p_name:'p_five',p_cost:50000}]);
```

20) const [x, setX] =useState (100);

<h1>{x}</h1> Ans _____

setX (200);

<h1>{x}</h1> Ans _____

PRACTICE PAPER-3

- 1) How Child Component Receives data from Parent Component _____
- 2) are Props Immutable? _____
- 3) write the Differences Between State and Props?

- 4) pass following data from Parent Component to Child Component
- a) welcome to reactjs
 - b) 1000
 - c) true
 - d) [100,200,300,400,500]
 - e) {key1:'Hello', key2:'Welcome', key3:'ReactJS'}
 - f) [{p_name:'laptop', p_cost:50000, p_image:'laptop.png'},
 {p_name:'watch',p_cost:20000, p_image:'watch.png'},
 {p_name:'mobile',p_cost:10000,p_image:'mobile.png'}
]

PRACTICE PAPER-4

- 1) React follows which DOM _____
- 2) In react only changed element will reload/refresh? _____
- 3) What is Single Page Application?

Ans:

- 4) what is Routing?

Ans

- 5) how to implement Routing in single page applications?
a) Route b) Routes c) both a&b d) None
- 6) how to encapsulate child Routings in single page applications?
a) Route b) Routes c) both a&b d) None
- 7) how to handle Dynamic Routing in single page applications _____
- 8) how to handle Static Routing in single page applications _____
- 9) how to read routing parameters in single page applications?
a) useNavigate () b) useParams ()
c) both a & b d) None
- 10) wildcard routing starts with _____
- 11) default routing starts with _____
- 12) how to hold child components in single page applications?
a) <Outlet></Outlet> b) <Route></Route>
c) <Routes></Routes> d) None
- 13) Draw the Diagram Representing Single Page Applications?

- 14) which package required to implement single page applications?

PRACTICE PAPER-5

- 1) How to implement State Management in ReactJS?
a) Context API b) Redux c) both a&b d) None
- 2) is context API inbuilt API (yes/no) _____
- 3) is Redux inbuilt API (yes/no) _____
- 4) Identify 3rd party library?
a) Context API b) Redux c) Both a&b d) None
- 5) Which Hook used to create Context in Context API _____
- 6) Which Hook used to consume Context in Context API _____
- 7) What is provider in Context API?
Ans:

- 8) What is consumer in Context API?
Ans:

- 9) Is Context API suitable for Static Data _____
- 10) which State Management Technique Recommended for Small Scale Web Application _____
- 11) Draw the Context API State Management Diagrammatic Representation

- 1) Redux library suitable for _____
 - a) small scale web applications
 - b) medium scale web applications
 - c) large scale web applications
 - d) both b & c
- 2) What is the name of the global object which is used to manage the application state in Redux?
 - a) Store
 - b) File
 - c) Directory
 - d) None of these
- 3) The states and actions are held together in Redux using?
 - a) View
 - b) Subscribe
 - c) Reducer
 - d) None of these
- 4) The following can be used to retrieve updated state in Redux and render it again
 - a) Store
 - b) Reducer
 - c) Action
 - d) View
- 5) The number of arguments which the createStore function can have been: _____
- 6) Which of the following makes stores available in Redux?
 - a) Views
 - b) Containers
 - c) Providers
 - d) Actions
- 7) The type of data flow followed in Redux is? _____
- 8) What is used to notify the view by executing their callback functions?
 - a) Store
 - b) Reducer
 - c) Action
 - d) State
- 9) In order to retrieve the current state of our Redux store, we can use the following function
 - a) content
 - b) action
 - c) dispatch
 - d) getState
- 10) Which of the following is a core principle of Redux?
 - a) Single source of truth
 - b) The state is read only
 - c) Changes are made with pure functions
 - d) All of the above
- 11) In order to dispatch an action to change a state in our application, we can use the following method:
 - a) getState
 - b) setState
 - c) subscribe
 - d) dispatch
- 12) can we create more than one reducer (yes/no) _____
- 13) can we create more than one store (yes / no) _____
- 14) why middleware's in redux?

15) identify the middleware's in redux?
a) thunk b) saga c) both a&b d) None

16) Explain Actions in Redux?
Ans:

17) Explain Action Creators in Redux?

Ans:

18) Explain Reducer in Redux?

Ans:

19) Explain Store in Redux?

Ans:

20) write the principles of redux?

21) what is dispatch in Redux?

22) what is Subscribe in Redux?

23) what are constants in Redux?

24) draw the Architectural Diagram of Redux?

25) write Few points related to thunk middleware?

26) draw the Architectural diagram of thunk middleware?

27) how to interact with external resources asynchronously?

a) thunk b) saga c) both a&b d) None

28) write few points related to saga middleware?

29) draw the architectural diagram of saga middleware

30) explain redux toolkit?

31) explain redux devtool?

33) write the libraries required to build redux application?

34) which library used to connect redux to react _____

35) write the Differences between context API and Redux?

36) what is Flux? And draw the Architectural Diagram of Flux

37) write the Differences between Flux and Redux?

38) write the Differences between thunk and saga?

39) is redux can integrate with other frameworks (yes/no) _____

40) write the command to download following libraries?

redux-thunk

redux-saga

react-redux

Ans:

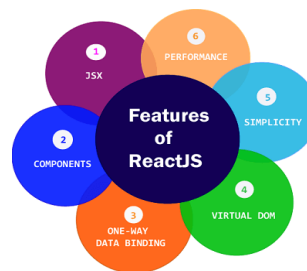
FAQ'S

1) What is React?

- ✓ ReactJS is **JavaScript library**
- ✓ ReactJS used to develop **web applications**
- ✓ ReactJS Released by **Facebook**
- ✓ We will develop ReactJS Applications in **two** ways
 - 1) JSX
 - 2) TSX
- ✓ Current version is **ReactJS 18.X**

2) What are the features of ReactJS?

- ✓ JSX
- ✓ Components
- ✓ Virtual DOM
- ✓ One way data binding
- ✓ Simplicity
- ✓ Performance



3) What the limitations of ReactJS?

- ✓ ReactJS is just a library not like Framework
- ✓ JSX takes time to understand
- ✓ Syntax are not user friendly

4) Explain JSX?

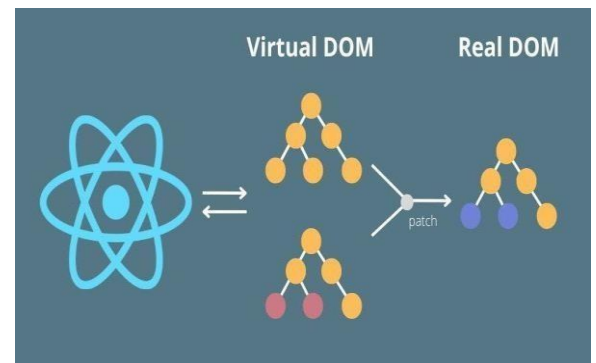
- ✓ JSX Stands for **JavaScript XML**
- ✓ JSX allows us to write **HTML** directly within the **JavaScript code**
- ✓ JSX used to implement **React Applications**
- ✓ JSX is an extension of the **JavaScript language**
- ✓ **Babel** will convert **JSX Expressions/Syntax** to Actual **JavaScript Code**
- ✓ **Performance** Increases with **JSX**

5) What is Babel?

- ✓ **Babel** is the **JavaScript Compiler**
- ✓ **Babel** will convert modern JavaScript (Latest Code) into a version **compatible with all browsers**
(Latest Code to Old Version Code)
- ✓ **Browsers** can't understand **JSX** will understand only **JavaScript**
- ✓ **Babel** will convert **JSX Syntax/Expression** to **JavaScript**

6) Explain Virtual DOM?

- ✓ A virtual DOM is a lightweight JavaScript
- ✓ Updating the virtual DOM is comparatively faster than updating the actual DOM
- ✓ In Virtual DOM, only Changed Element will Reload Instead of All Elements
- ✓ Because of Virtual DOM Application Performance Increases



7) Differences Between Angular and ReactJS?

Angular	React
Released by Google	Released by Facebook
We will use TypeScript	We will use JSX
Open-Source JavaScript framework	Open-Source JavaScript library
Two-way data binding	One way data binding
Regular DOM	Virtual DOM
Supports MVC	Supports Flux
Performance is Slow	Performance is High because of Virtual DOM
Supports unit and integration testing	Supports only unit testing

8) Differences Between Real DOM and Virtual DOM?

Real DOM	Virtual DOM
In Real DOM Updates are slower	In Virtual DOM updates are faster
Real DOM updates the HTML Directly	Virtual DOM can't update HTML Directly
DOM Manipulations are Expensive	DOM Manipulations are Easy
Memory wastage in Real DOM	There is no Memory wastage in Virtual DOM

9) Explain render ()?

- ✓ render () hook used to write the presentation logic (HTML Code) in class level components
- ✓ render () hook is mandatory hook
- ✓ when ever state changes automatically render () hook will execute

10) Differences Between State and Props?

State	Props
State Contains Component Data	Child Component Receives Data from Parent Component
State is Mutable	Props are Immutable

11) Explain setState () in ReactJS?

- ✓ **setState ()** is the predefined method used to **change/update** the state of Component (**Class Level Components / Stateful Components**)

12) Differences Between Class Level Components and Functional Components?

Functional Components	Class Components
1) We can't create object to functional components	We can create object to class components
2) These components are also called as temporary components	These components are existed at the end of process (Container Components)
3) These components are called as stateless components	These components are called as stateful components
4) Supports Hooks	Supports life cycle methods
5) We can't reuse stateless components	We can reuse stateful components
6) These components are easy to understand	These components are difficult to understand
7) We will write html in return method	We will write html in render () hook

13) Explain lists in ReactJS?

- ✓ Lists are used to display data in ordered format
- ✓ **map ()** method used to iterate lists in ReactJS

14) what are keys in ReactJS lists?

- ✓ key is a unique identifier
- ✓ key is used to identify which items changed/updates/deleted from list
- ✓ in react only changed element will reload instead of all elements in lists
- ✓ Application performance will increase

13) What is Redux?

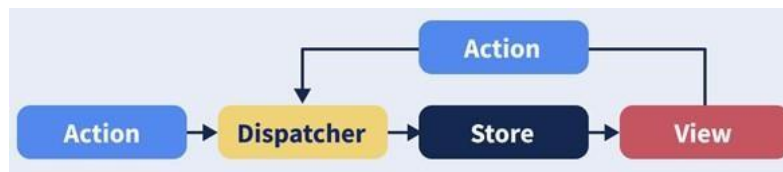
- ✓ Redux is 3rd party library
- ✓ Redux overcomes props drilling
- ✓ Redux used to maintain state of react applications
- ✓ Redux reduces burden on server
- ✓ Redux we can integrate to Angular, VueJS, Vanila's, ...
- ✓ Redux library size is around 2kb

14) Explain Flux?

- ✓ Flux is an application paradigm (Pattern)
- ✓ Flux behaves live MVC (Model-View-Controller)
- ✓ Flux Direction is Unidirectional
- ✓ Flux also used to maintain the state

Components of Flux

1. View
2. Action
3. Dispatcher
4. Store

**15) Differences between Redux and Flux?**

	Redux	Flux
number of stores	One	More
Architecture	build user interfaces	build web applications (Client server architecture)
business logic	resides on reducer	resides on store

16) Write the core principles of Redux?

- ✓ only one Application State
- ✓ State is read-only (changes are done through actions)
- ✓ Changes are made with pure functions

17) Advantages of Redux?

- ✓ Redux is used to overcome to props drilling
- ✓ Redux applications are suitable for medium and large-scale web applications
- ✓ Redux suitable for Dynamic Data (Frequently Updating)

- ✓ Debugging Easy
- ✓ Maintains only one Application State
- ✓ Performance is high
- ✓ Flexible
- ✓ Centralized
- ✓ Predictable

18) Explain Redux Toolkit?

- ✓ Redux Toolkit provides Boilerplate snippets
- ✓ Redux Toolkit saving developers "development time"
- ✓ Building Redux applications with Redux Toolkit is Easy

19) Explain Redux DevTools?

- ✓ with the help of Redux DevTools, application debugging is Easy
- ✓ with the help of Redux DevTools, we can inspect state
 - 1) previous state
 - 2) new state
- ✓ with the help of Redux DevTools we can inspect Actions

**20) what are the differences between
mapStateToProps () and mapDispatchToProps ()?**

mapStateToProps ()

- ✓ mapStateToProps () used to perform subscription in Redux architecture

mapDispatchToProps ()

- ✓ mapDispatchToProps () used to perform dispatch operation in Redux architecture

21) Explain Action in Redux?

- ✓ source of information to store called as Action
- ✓ Action is plain JavaScript object/JSON Object

Ex.

```
let actionObj = {  
  type: "ADD",  
  payload: 10  
};
```

10) Explain Constants in Redux Architecture?

- ✓ we can overcome grammatical mistakes while developing Redux Applications
- ✓ we can maintain all constants in single file
- ✓ we can import these constants to actions, reducers,

11) what are reducers?

- ✓ in reducers we will write "business logic"
- ✓ we will create "more" than one reducer

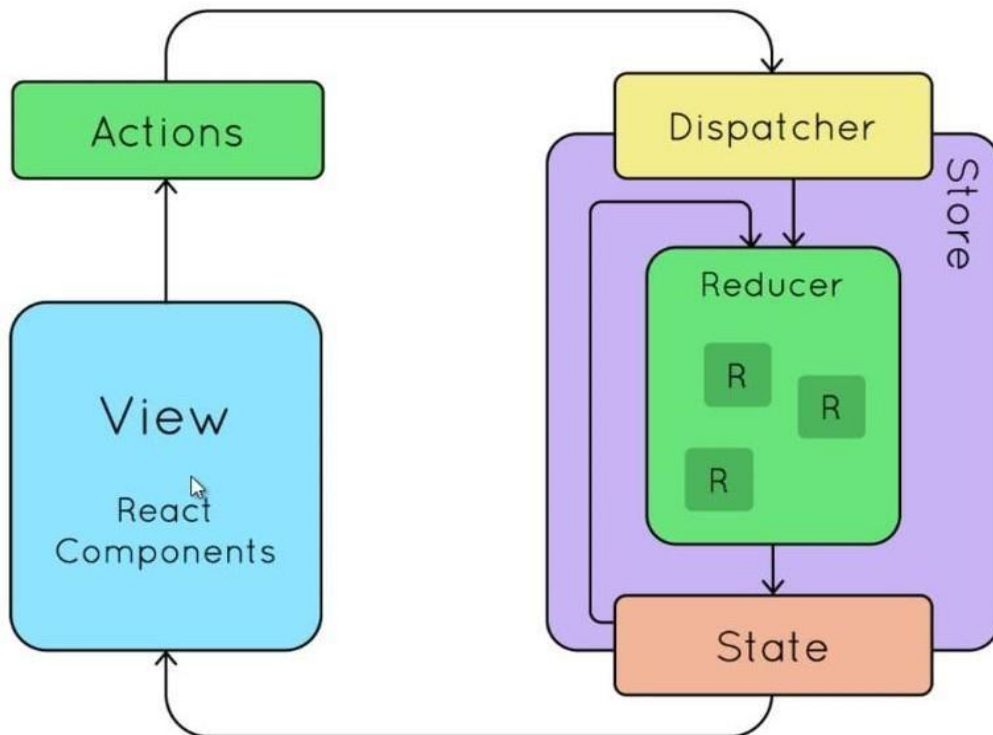
- ✓ application "readability" and "modularity" increases because of "Multiple reducers"
- ✓ reducers take the "action" and "previousState" as input and return "newState" as output

Ex.

```
function reducer (state, action) {
  -----
  -----
  return newState;
}
```

12) Explain Redux Life Cycle?

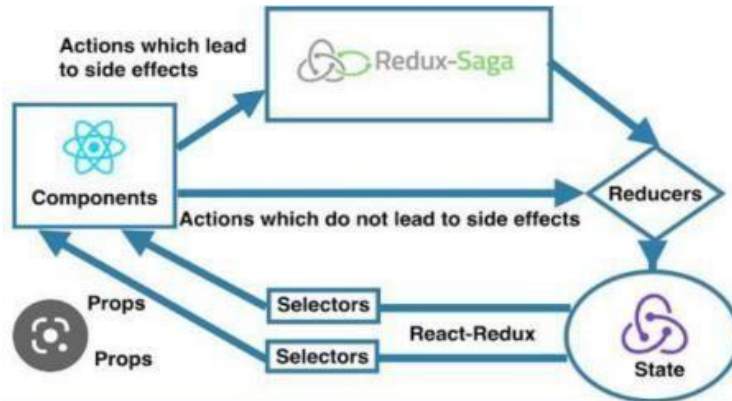
- ✓ Components Sending Actions to Store (dispatch)
- ✓ Reducer Receives the Action and Previous State from Store
- ✓ Reducer Generates newState
- ✓ Reducer will send newState to Component through Store



13) Explain Redux Saga?

- ✓ Redux Saga is the Middleware
 - ✓ Redux Saga used to connect to external resources Asynchronously
- Ex.
- Making http calls
 - Reading browser storages
 - Perform I/O operations
- ✓ Above Operations called as Side Effects

- ✓ Through **Redux Saga with Actions**, we can perform started, paused and aborted operations from **Redux Applications**



14) Explain Store in Redux?

- ✓ We can have only one **Store** in **Redux Application**
- ✓ Store contains **Application State (only one Application State)**
- ✓ Store Accommodates Reducer
- ✓ Store provides the following methods
 1. dispatch
 2. getState
 3. subscribe
 4. replaceReducer

15) Explain InitialState in Redux?

Ex.

```
function reducer (state=10, action) {

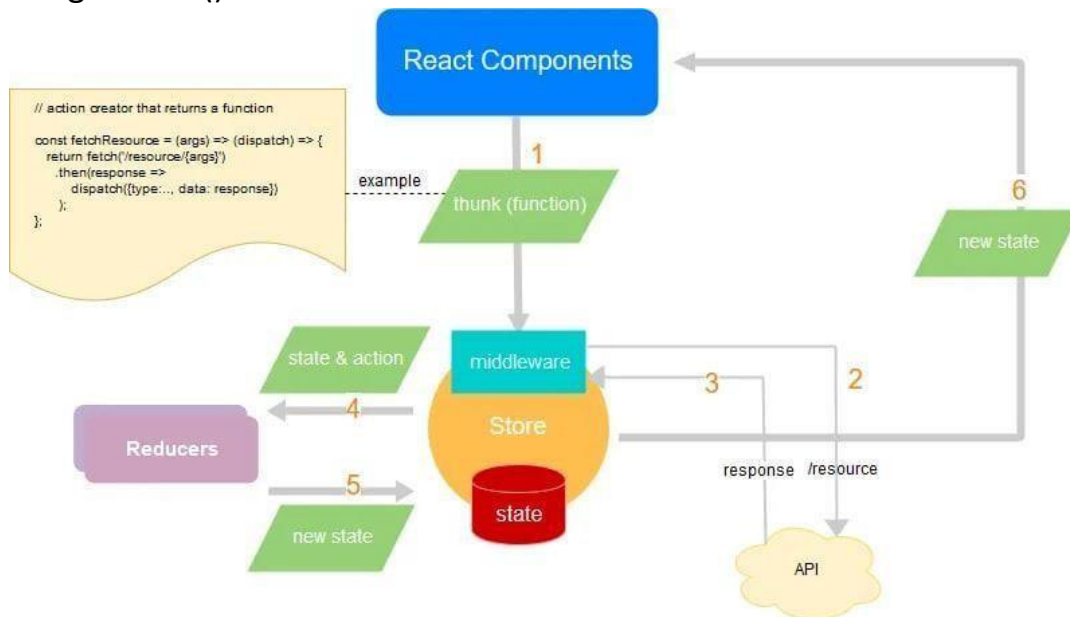
  return newState;
}
```

In above reducer function, state initialized with default value as **"10"**
Above value called as **initialState (10)**

16) Explain Redux Thunk?

- ✓ Thunk is Middleware
- ✓ Thunk Middleware used to delay calculations and evaluation of any operations in Redux Architecture
- ✓ Action Creator will return a function instead of object
- ✓ Returned function will receive two methods from store
 - 3) dispatch
 - 4) getState

- ✓ dispatch method used to make synchronous operation after successful completion of asynchronous operations
- ✓ getState () method used to access the state from store



17) Explain Workflow Features of Redux?

1) Reset

- ✓ we can reset application state

2) Revert

- ✓ we can perform **undo** operation on application state changes

3) Sweep

- ✓ we can remove **unexpected** fire of **disabled action**

4) Commit

- ✓ we can initialize state with **default value**

18) Explain subscribe in Redux

- ✓ receiving new state from store called as **subscribe**

19) Explain dispatch in Redux

- ✓ sending actions to store called as **dispatch**

REACTJS PRACTICE BOOKLET

- 1) What is React?
a) Library b) Framework c) both a&b d) None
- 2) Why React _____
- 3) Who released React?
a) Facebook b) Google c) Microsoft d) None
- 4) What is current version of React _____
- 5) What is JSX?
- 6) npm stands for _____
- 7) npx stands for _____
- 8) how to download libraries?
a) npm b) yarn c) both a&b d) None
- 9) write the installation steps for React?

Step 1.

Step 2.

Step 3.
- 10) Write steps to create React Application?

Step 1.

Step 2.

Step 3.
- 11) What is Default Port Number of React?
a) 3000 b) 4200 c) 8080 d) 27017
- 12) Why node_modules in React Project?
- 13) Which component is Default Component in React? _____
- 14) Which template is main template in React? _____
- 15) Where to deploy components in React Applications? _____
- 16) Write the Execution flow of React Application
- 17) What is component? Write minimum 5 points?

1)

2)

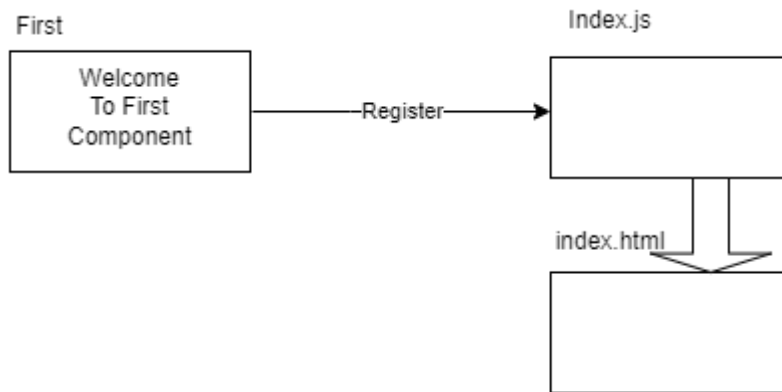
3)

4)

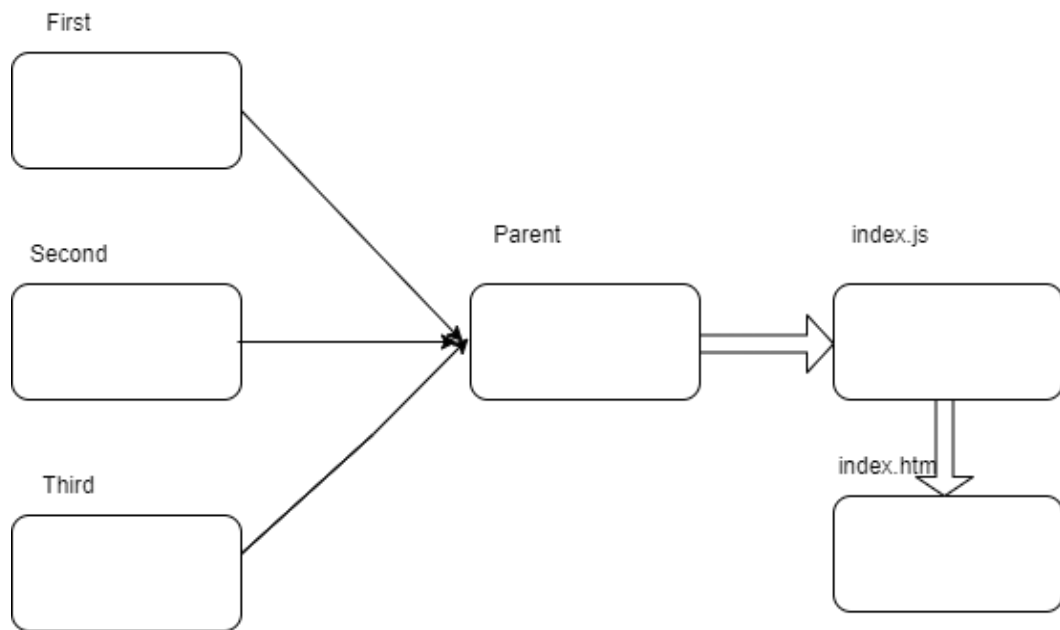
5)
- 18) Draw sample diagram to represent components in React Application?
- 19) Types of components in React

1)2)
- 20) How to enhance Functional Components _____
- 21) Hooks introduced in _____

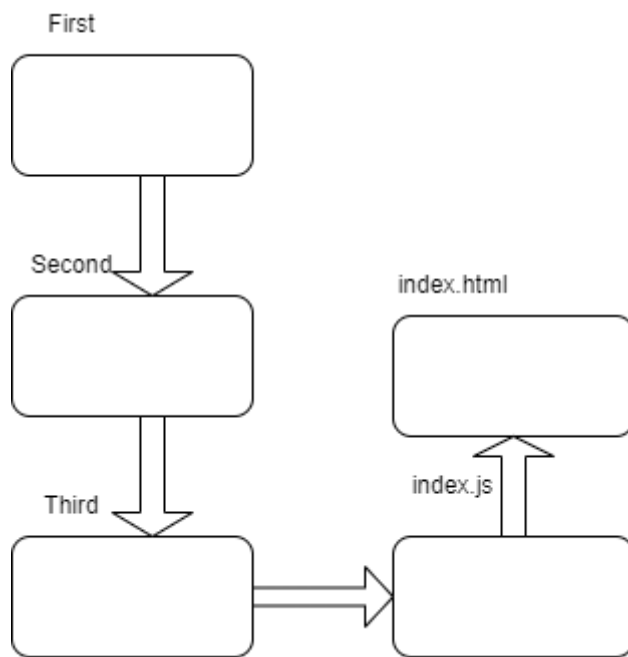
22) Write the code to create Following Component?



23) Write the code for below diagram



24) Write the code for below diagram?



25) What is state in component?

26) Write the syntax for state in component?

27) How to iterate list in react application?

- a) for loop b) while loop c) map function d) None of the Above

28) display the following list in react application?

["Angular"," React"," Node"," VueJS"," MongoDB"," Mern Stack"]

SNO	Value
0	Angular
1	React
2	Node
3	VueJS
4	MongoDB
5	Mern Stack

29) display the following Array in the form of a table?

```
[
  {"p_id":111,"p_name":"p_one","p_cost":10000},
  {"p_id":222,"p_name":"p_two","p_cost":20000},
  {"p_id":333,"p_name":"p_three","p_cost":30000},
  {"p_id":444,"p_name":"p_four","p_cost":40000},
  {"p_id":555,"p_name":"p_five","p_cost":50000}
]
```

Result:

p_id	p_name	p_cost
111	p_one	10000
222	p_two	20000
333	p_three	30000
444	p_four	40000
555	P_five	50000

REACTJS PRACTICE BOOKLET

30) how to do view encapsulation?

- a) `<></>` b) `<React.Fragment></React.Fragment>` c) both a&b d) None

31) Write the snippet for download react-router-dom@6 package?

npm :

yarn :

32) Show or hide the following message in React Application with conditional Rendering?

First

Show / Hide

Welcome to Conditional Rendering

33) Implement below example with Conditional Rendering?

Components are: 1) Dashboard

2) Success

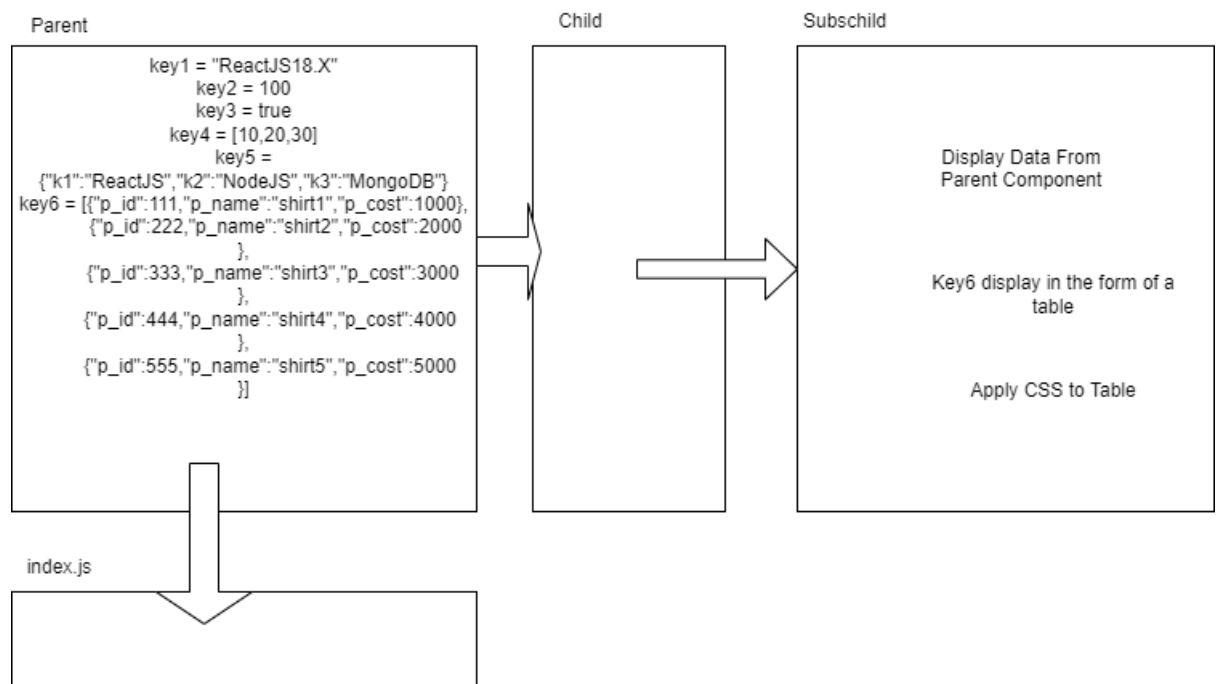
3) Failure

Write Conditional Rendering in Dashboard Component, display "Success Component" based on true and "Failure Component" based on false

34) How to provide communication between components?

- a) props b) state c) both a&b d) None

35) implement the below application with props and props drilling



36) implement basic application with props.children
pass "ReactJS" from Parent to Child through props.children

37) implement above example with State
Note: keys data should read from state

38) implement the example to share data from Child Component to Parent Component
Ex.

Share the following data

Key1: "Hello"

Key2: 100

Key3: true

39) write the basic example for click event with 0 parameters

40) write the basic example for double click with 0 parameters

41) write the basic example for click event with two parameters (i.e. Careerit & Careerit@123)

42) write the basic example for double click event with two parameters (i.e. Careerit & Careerit@123)

43) implement the Registration Example

Registration		Details	
User Name	<input type="text" value="Careerit"/>	User Name	Careerit
Mobile	<input type="text" value="12345"/>	Mobile	12345
Email	<input type="text" value="hr@Careerit.com"/>	Email	hr@Careerit.com
Address	<input type="text" value="Hyderabad"/>	Address	Hyderabad
<input type="button" value="Register"/>			

44) implement the Login Example

User Name

admin

Password

.....

Login

Login Success

45) how add CSS to Components and write Examples also?

46) consume below GET Requests and apply Conditional Rendering with Font Awesome (Spinner)

URL1:

<https://www.7timer.info/bin/astro.php?lon=113.2&lat=23.1&ac=0&unit=metric&output=json&tzshift=0>

URL2:

<https://www.w3schools.com/angular/customers.php>

URL3:

<https://reqres.in/api/unknown>

URL4:

<https://restcountries.com/v3.1/all>

47) implement below GET Requests by id

URL1:

<https://reqres.in/api/users?page=2>

URL2:

<https://reqres.in/api/unknown/2>

48) implement below POST Requests

URL1: <https://reqres.in/api/register>

Request data

```
{
  "email": "eve.holt@reqres.in",
  "password": "pistol"
}
```

Response data

```
{
  "id": 4,
  "token": "QpwL5tke4Pnpja7X4"
}
```

URL2: <https://reqres.in/api/login>

Request Data

```
{
  "email": "eve.holt@reqres.in",
  "password": "cityslicka"
}
```

Response data

```
{
  "token": "QpwL5tke4Pnpja7X4"
}
```

URL3: <https://reqres.in/api/users>

```
{
  "name": "morpheus",
  "job": "leader"
}
```

```
{
  "name": "morpheus",
  "job": "leader",
  "id": "545",
  "createdAt": "2023-03-07T06:09:04.470Z"
}
```


49) implement the below PUT Requests

URL1: <https://reqres.in/api/users/2>

Request

`/api/users/2`

```
{
  "name": "morpheus",
  "job": "zion resident"
}
```

Response

`200`

```
{
  "name": "morpheus",
  "job": "zion resident",
  "updatedAt": "2023-03-07T05:17:22.231Z"
}
```

50) implement below DELETE Request

URL : <https://regres.in/api/users/2>

Request

`/api/users/2`

Response

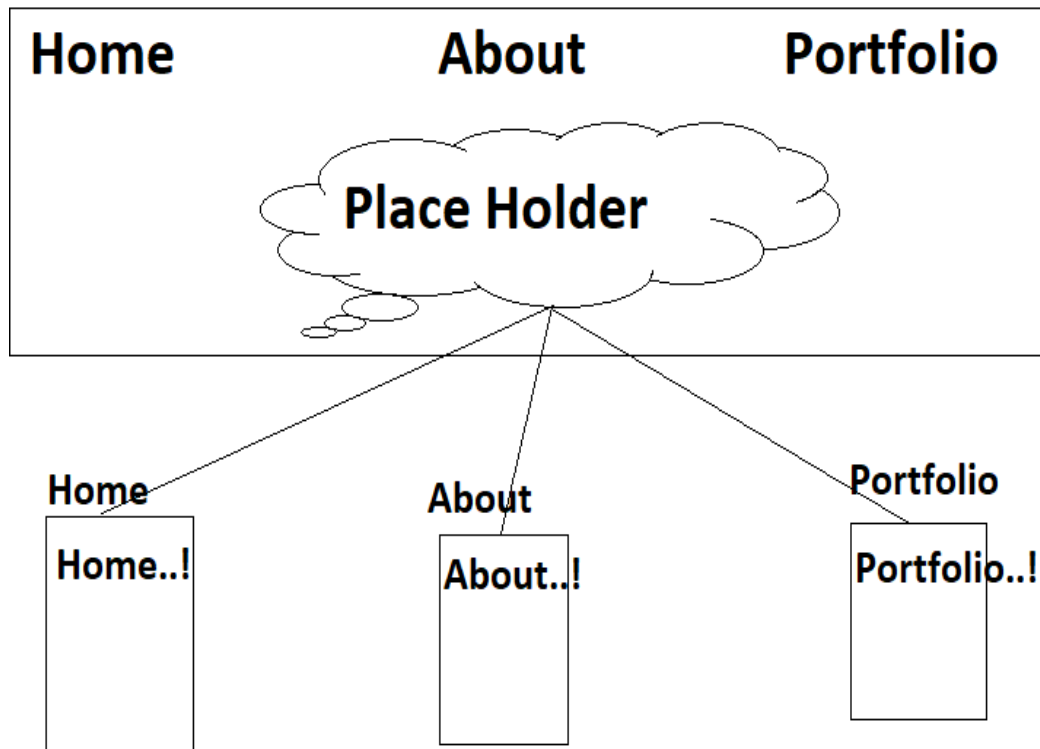
`204`

51) what are Single Page Applications? Write minimum 10 points and Draw the Diagram

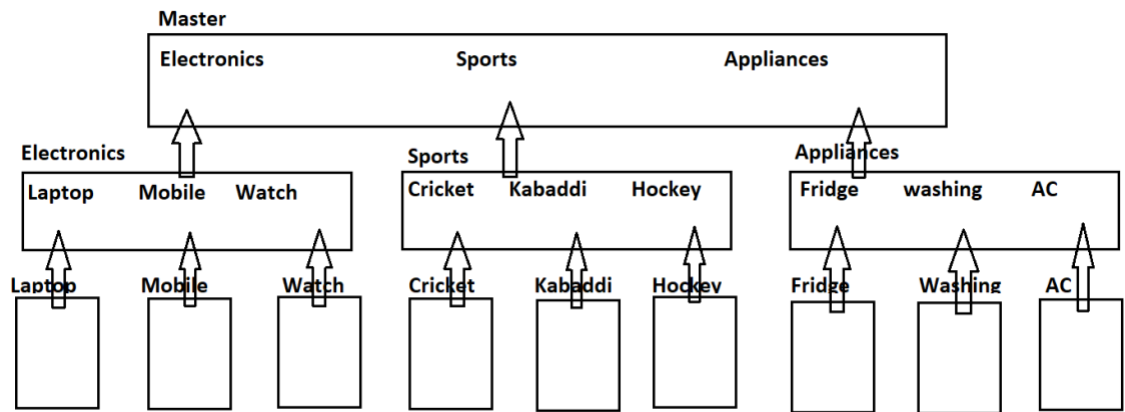
1)
2)
3)
4)
5)
6)
7)
8)
9)
10)

52) implement the below application with Single Page Applications

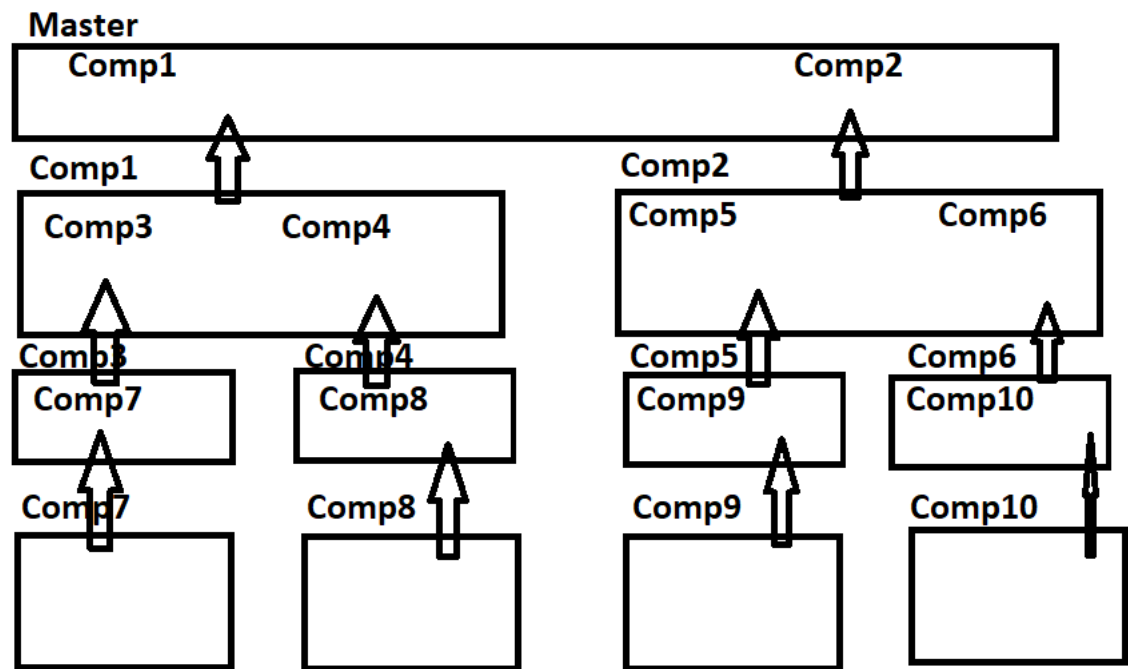
Main



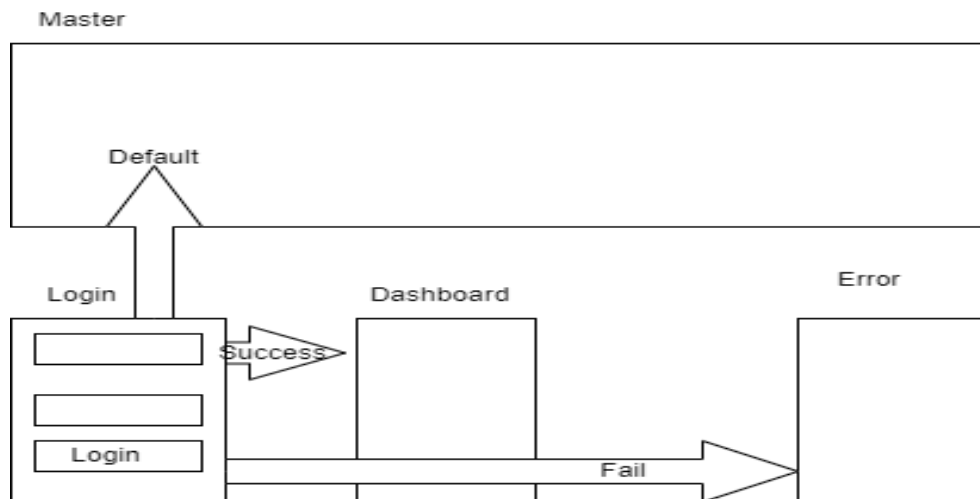
53) Implement the Below Application with Single Page Application



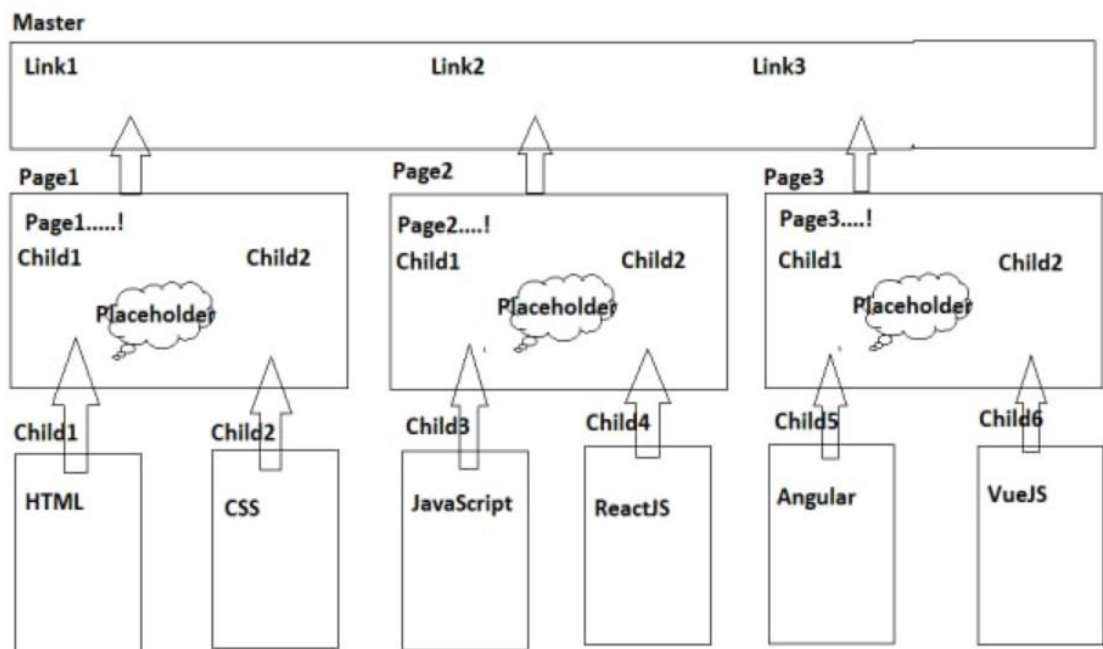
54) Implement the Below Application with Single Page Applications



55) Implement the below application with Single Page Applications



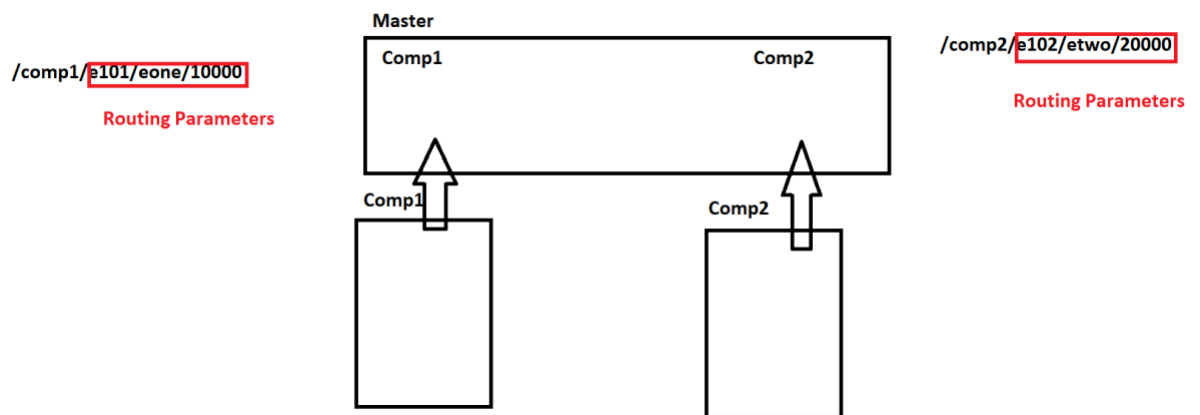
56) Implement the Below Application with Single Page Applications



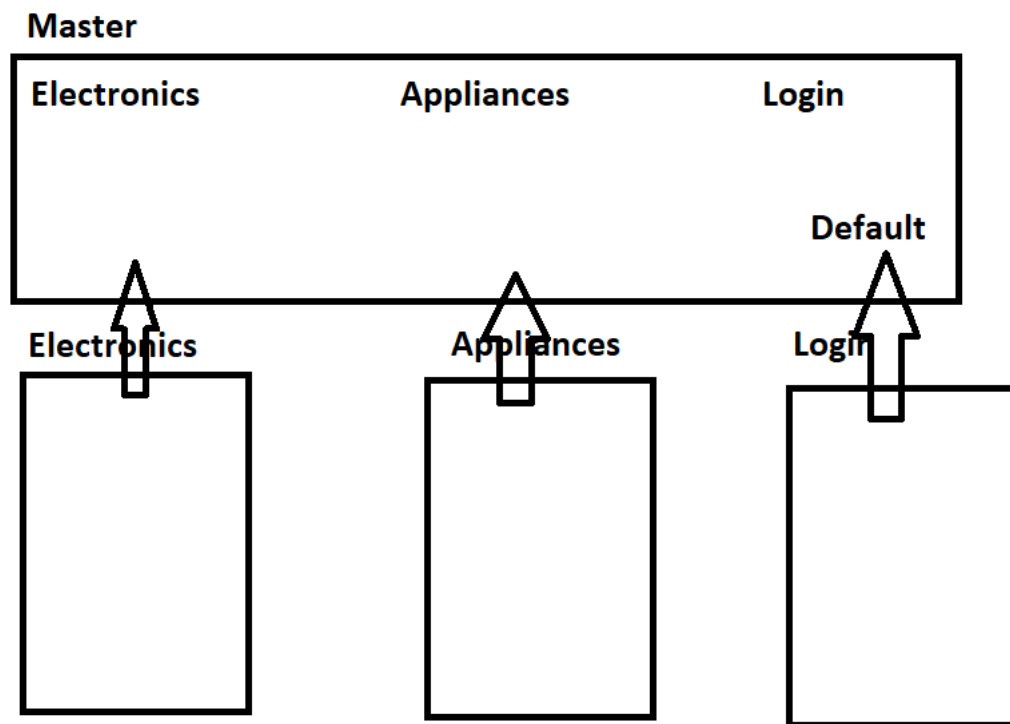
57) What is useNavigate () hook in Single Page Applications?

58) What is useParams () hook in single page applications?

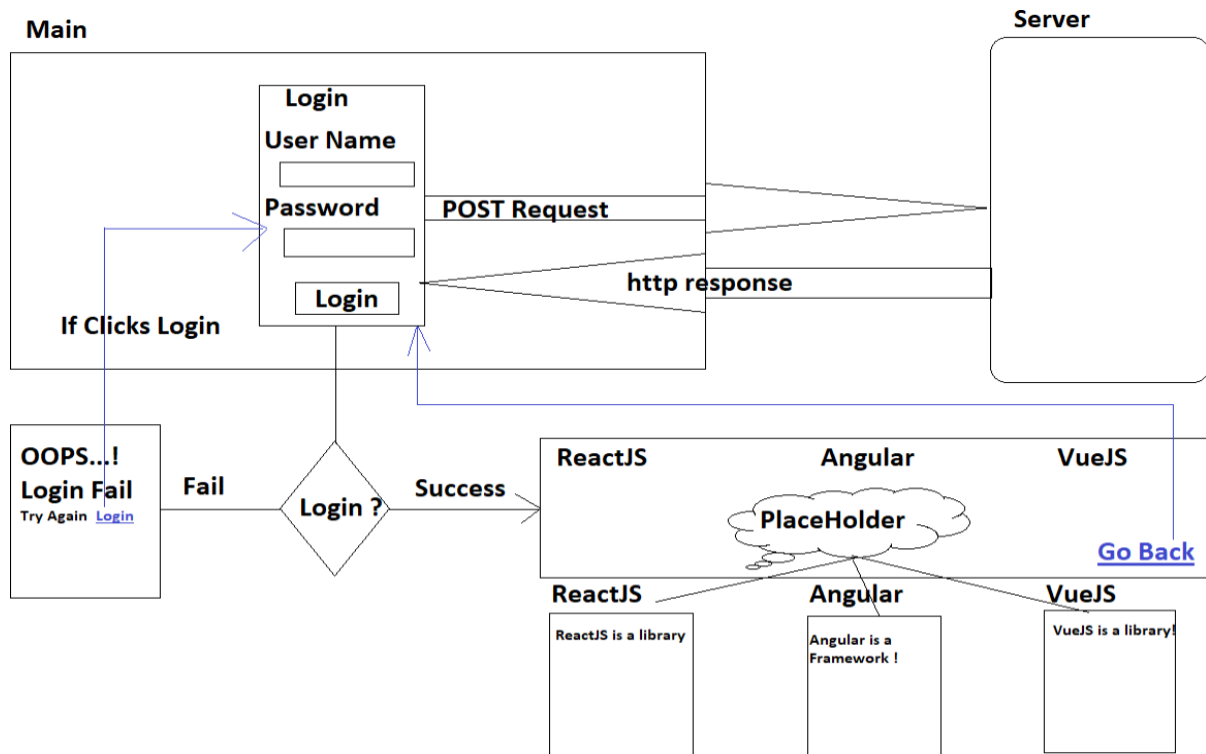
59) Pass Routing Parameters in Below Application?



60) Make Login Component as Default Component in Below Diagram



61) Implement the below project with single page applications?



62) What is <Outlet></Outlet> Component in ReactJS

63) why state management in React?

64) how to implement state management in react?

1)

2)

65) what is redux?

66) draw the architectural diagram for redux

67) what is dispatch?

68) what is Action?

69) what is reducer?

70) can we create more than one reducer (yes/no) _____

71) what is store?

72) what is state in redux architecture?

73) what is subscribe in redux architecture?

74) what is thunk middleware and draw the architectural diagram

75) what is saga middleware in redux?

76) write the hooks in state management?

1)	2)
----	----

77) how to subscribe in redux architecture _____

78) how to dispatch in redux architecture _____

79) how to combine reducers in redux architecture _____

80) consume the below rest api by using redux

<https://www.w3schools.com/angular/customers.php> (GET Request)

81)

82) consume the below rest api by using redux

<http://benfranklinserver-env.eba-5adimdsp.us-west-2.elasticbeanstalk.com/login> (post request)

body --- {"uname": "Careerit", "upwd": "Careerit@123"}

83) What is regular expression

84) Write the Syntax for Regular Expression _____

85) What is ^ operator in Regular Expression _____

86) What is \$ operator in Regular Expression _____

87) What is \d operator in Regular Expression _____

88) What is \w operator in Regular Expression _____

89) What is {} operator in Regular Expression _____

90) Write the Regular Expression for empty string validation _____

91) Write the regular expression for

- a) Should start with Capital Letter
- b) Should contains alpha numeric vales
- c) Minimum 6 Characters
- d) Maximum 8 Characters

92) Write the Validations for Password

- a) Should contain one digit
- b) Should contain one capital letter
- c) Should contain one lower case letter

- d) Should contain one special Character
- e) Must contain minimum 8 characters

93) Write the Validation for email

94) Write the Validations for Mobile Number

95) What is lodash package?

96) How to download lodash package by using yarn tool

97) Write the validation for below application (text validation)

USER NAME VALIDATION

User Name

Login Reset

98) Write the validation for below application (password validation)

PASSWORD VALIDATION

Password

Login Reset

99) Write the validation for below application (email validation)

EMAIL VALIDATION

Email

Login Reset

100) Write the validation for below application (mobile number validation)

MOBILE NUMBER VALIDATION

Mobile

Login Reset

101) Write the validations for Login Form

LOGIN FORM

User Name

Password

102) Write the Validations for Registration Form

REGISTRATION FORM

First Name (enter first name)

Last Name (enter last name)

Password (enter valid password)

Conf Password Repeat Same Password (enter valid password)

Email (enter valid email)

Age (enter age > 20)

Gender ☐ Male ☐ Female (select gender)

Skills ☐ C Lang ☐ CPP (choose skills)

☐ Java Lang ☐ Dot Net

Country Select Country (select country)

Select Date dd-mm-yyyy (select date)

Pref Time --:-- (choose pref time)

Choose Color (choose color)

Address (enter address)

Upload Resume No file chosen (**pdf files only**)

103) Write the life cycle stages of Class Level Components

a)	b)	c)	d)
----	----	----	----

104) Write the snippet (code) for constructor with state in class level components

105) Write the few life cycle methods in class level components

106) Write the few points related to render () in class level components

a)
b)
c)
d)
e)

107) Write the few points related to componentDidMount () in class level component

a)
b)
c)
d)
e)

108) What is componentWillReceiveProps () in class level components

--

109) Write the few points related to shouldComponentUpdate () in class level component

110) In initial stage which one will execute in class level components

- a) Constructor b) state c) props d) All of the Above

111) In mounting Stage which hooks will execute in class level components?

--

112) In updation stage if props are changed which life cycle methods will execute?

--

113) In updation stage if state changed which life cycle methods will execute?

--

114) In unmounting stage which life cycle method will execute

115) Draw the Diagram for life cycle methods in Class Level Component

