React JS

Fundamentals
HTTP
Routing
Redux
Utilities.

=============================Fundamentals=====================================

01.Introduction
  i. What is React
  ii. Why learn React
  iii. What are Prerequisites

02.Environmetal Set Up (Nodejs & text editor(VS Code))
03.Hello World
04.Component & Types of Component.
  i. Functional Component
  ii. Class Component.

05.JSX
06.Props
07.State
08.Event Handling
09.Component Communication.
  i)Parent to Child (By props)
  ii)Child to Parent (By Using callback+props)

10.Conditional Rendering
  i)if/else
  ii)Element variables
  iii)Ternary Conditional Operator
  iv)Short circuit Operator

11.List and Keys
12.React CSS /Styling React Components
  i)CSS stylesheets
  ii)Inline styling
  iii)CSS Modules
  iv)CSS in JS Libaries

13.React Forms
14.Lifecycle of Components

15.React Fragment
16.React Portals

17.Pure Component
18.Memo

19.React Refs
20.Higher Order Components

20.Render props
21.Context

01#.Introduction
=   ===========
- React is a declarative, efficient, and flexible JavaScript library for building user interfaces.

#i)What is React ?
- Open source JavaScript library for building user interface.
- Not a framework.
- Focus on UI
- It allows us to create reusable UI components

#ii)Why learn React ?
- Created and mainted by Facebook.
- Huge community.
- demand skillset

- It is a component based architecture.
- It allows us to create reusable UI components.
- It is declarative.(Tell to React what you want and React will build actual UI)

#React creates a VIRTUAL DOM in memory.
- Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.

- React will handle efficiently updating and rendering of the components.
- React only changes what needs to be changed!

#Integration react into any o f your application.
- We can integrate portion of page or complete page or even entire application itself.

#iii).What are Prerequisites ?
- HTML,CSS,Javascript fundamentals.
- ES6

- Javascript -'this' keyword ,filter,map,and reduce.
- ES6   -let & const,arrow functions,template literals,default parameters,object literals,rest and spread operators and destructuring assignment.

#02.Environmetal Set Up
==   ===============
- Install Nodejs from https://nodejs.org
- Text editor of your own interest (eg :VS Code)

03.Hello World
= =========
#npx
```
-npx create-react-app <project_name>
->npx create-react-app my-app
->cd my-app
->npm start
```
Where npx is a package runner tool that comes with npm 5.2+ or higher

#npm
->npm install create-react-app -g
->create-react-app <project_name>

#yarn
->yarn create react-app my-app

Note :
If you've previously installed create-react-app globally via npm install -g create-react-app,
we recommend you uninstall the package using npm uninstall -g create-react-app
or yarn global remove create-react-app
to ensure that npx always uses the latest version.

#Folder Structure:

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
    └── setupTests.js
```

package.json -->contains dependencies and scripts .
index.js -->start point of appliction.

#04.Component.
== =========

```
----------------------
|       Header        |
|---------------------|------------->Root(App) Component
|        |         |S     |
|SNav |MCon|Nav|
|---------------------|
|      Footer          |
----------------------
```

Here Header,SNav,MCon,Footer are components,these are wrapped by Root(App) component.

#Component :
- Components are like functions that return HTML elements.

  (or)
- Components are independent and reusable bits of code.

Component Types:
----
i)Functional Component
ii)Class Component

i)Functional Component
- These components are stateless components and
- It does support react life-cycle methods.
- These components can be used for presentation purpose.

```
                         ____________________
  __Properties____  |JavaScript Function   |________HTML(JSX)
    (Props)          |____________________|
```

eg:
```
   function Welcome(props) {
      return <h1>Hello, {props.name}</h1>;
  }
```

ii)Class Component
- These components are statefull components.
- It can support react life-cycle methods by extending react components.
- These components can be used when you want to create methods , state for an component.

```
                         ____________________
  __Properties____ |        ES6              |________HTML(JSX)
    (Props)          |___(State)_________|
```

eg :
```
   class Welcome extends React.Component {
       render() {
          return <h1>Hello, {this.props.name}</h1>;
       }
     }
```

- When creating a React component, the component's name must start with an UPPER case letter.
- The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML

Functional Vs Class Components
 - --------------------------------------

Functional	Class
• Simple Functions	• More feature rich
• Absence of 'this' Keyword	• 'this' keyword
• No State	• Maintain their own private data -State

• No lifecycle hooks	• Provide lifecycle hooks
• Stateless/Dumb/Presentationl	• Stateful/Smart/Container.

***** Imp*****
Note :
Hooks are new feature proposal that let you use state and React feactures without writing a class.
Hooks are introduced in React v16.7.0-aplha.

•So that Function component also stateful component by using Hooks Concept.

05.JSX
= ====
• JavaScript XML (JSX) -Extension to the javaScript language syntax.
• JSX allows us to write HTML in React.
• JSX tags have a tag name,attributes and children.
• You are not required to use JSX, but JSX makes it easier to write React applications.
• JSX Utimately transpiles to pure javascript which is understood by the browsers.
• With JSX you can write expressions inside curly braces { }.

#Internally Conversion of JSX :
• JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods
• JSX converts HTML tags into react elements at runtime.

#With JSX
```
 import React from 'react';
 import ReactDOM from 'react-dom';
 const  element = <h1>With JSX In React</h1>;
 ReactDOM.renderelement, document.getElementById('root'));
```

#Without JSX
```
import React from 'react';
import ReactDOM from 'react-dom';
const element = React.createElement('h1', {}, 'Without JSX In React!');
ReactDOM.render(element , document.getElementById('root'));
```

#JSX Difference
```
  class  -->className
  for    -->htmlFor
 camelCase property name convention :
  .onclick  -->onClick
  .tabindex-->tabIndex
```

06.Props
-- =====
•Props are arguments passed into React components via HTML attributes.
(or)
•React Props are like function arguments in JavaScript and attributes in HTML.
•To send props into a component, use the same syntax as HTML attributes:
•React Props are read-only! and props are immutable.

07.State

= ====
•React components has a built-in state object.
•The state object is where you store property values that belongs to the component.
•Whenever the state object changes, the component re-renders.
•The state object is initialized in the constructor.
•To change a value in the state object, use the this.setState() method.

#Props Vs State
The difference between Props and State is....

Props	State
•props get passed to the component	• state is managed within the component
•Function parameters	• Variables declared in the function body
•props are immutable	• state canbe change
•props -Functional Components	• useState Hook -functional component
this.props-Class Componets	this.state -class component

08.Event Handling
== ===========
•Just like HTML, React can perform actions based on user events.
•React has the same events as HTML: click, change, mouseover etc.
•React events are written in camelCase syntax:
  -- onClick instead of onclick
•React event handlers are written inside curly braces:
  -- onClick={shoot} instead of onClick="shoot()"

We can use four approch in Event binding in class component.They are
i)Bind method on elemet itself
```
<button onClick={this.clickHandler.bind(this)}>Click Me</button>
```
ii)Arrow Function on elemet itself
```
 <button onClick={()=>this.clickHandler()}>Click Me</button>
```
iii)Bind method in constructor.
```
 constructor(){
     super();
     this.changeMessage=this.changeMessage.bind(this)
  }
  <button onClick={this.changeMessage}>Click Me</button>
```
iv)Arrow Function
```
    changeMessageByArW=()=>{
        console.log(this)
        this.setState({
            message:'Thanks....!'
         })
    }
  <button onClick={this.changeMessageByArW}>Click Me</button>
```

09.Component Communication.
 == ===================
 If you want to pass any data from one component to another component,we can use props
 Props provide one-way communication from a parent to a child,

But by using callback we can pass child to parent also.

i)Parent to Child (By props)
ii)Child to Parent (By Using callback+props)

i)Parent to Child (By props)
• In this type of communication, a parent passes the data to the child by adding an extra attribute in the child comp onent declaration.

```
  eg : <ChildComponent name='Pojitha'/>   (From P-C)
```

ii)Child to Parent (By Using callback+props)
• Data from a child can be passed to the parent using a callback. This can be achieved by using the following steps.

a)Create a callback method in parent and pass it to the child using props.
b)Child can call this method using "this.props.[yourCallbackName]" form child and pass data as argument.

```
eg :     In Parent                                              (From C-P)
          setName=(name)=>{
           this.setState({name})
          }
          <ChildComponent setName={this.setName}/>

        In Child
          sendParent =()=>{
               this.props.setName('pojitha')
          }
         <button onClick={this.sendParent}>Click Me</button>
```

10.Conditional Rendering
= ================
i)if/else
ii)Element variables
iii)Ternary Conditional Operator   // true?' Yes' :'No'
iv)Short circuit Operator   // true &&' Yes'

11.List and Keys
= ==========
•Keys help React identify which items have changed, are added, or are removed. Keys should be given to the eleme nts inside the array to give the elements a stable identity

```
 Eg :(i)
        List WITHOUT key attribute           (Insertion end at the List)
           #01                                          #02
          <ul>                                         <ul>
              <li>John</li>                               <li>John</li>
              <li>Roja</li>                                 <li>Roja</li>
           </ul>                                             <li>Ramu</li>
                                                        </ul>
```

•React will iterates both list (01 and 02) at same time for comparsion,
if it find any difference (#01 items should match #02 items + (1 or more items)),
then react mutate (changes) in List.

--React comparison like these...

#01 List	#02 List	Changes
<li>John</li>	<li>John</li>	NO
<li>Roja</li>	<li>Roja</li>	NO
	<li>Ramu</li>	YES

Then finally ,react insert item into list.

(ii)

```
        List WITHOUT key attribute           (Insertion beginning at the List)
          #01                                          #02
                                                     <ul>
           <ul>                                         <li>Ramu</li>
             <li>John</li>                              <li>John</li>
             <li>Roja</li>                                <li>Roja</li>
          </ul>                                        </ul>
```

--React comparison like these...

#01 List	#02 List	Changes
<li>John</li>	<li>Ramu</li>	YES
<li>Roja</li>	<li>John</li>	YES
	<li>Roja</li>	YES

Note :(*****Imp*******)
Then react will keep as it is instead of mutation becoz list #02 is different

•Thats Why React ask key to Identify to item in List for which items have changed, are added, or are removed.

(ii)

```
        List WITH key attribute
          #01                                          #02
                                                     <ul>
           <ul>                                         <li key="3">Ramu</li>
             <li key="1">John</li>                     <li key="1">John</li>
             <li key="2">Roja</li>                      <li key="2">Roja</li>
          </ul>                                       </ul>
```

•Here React match/compare Original List items(#01) with subsequent List items(#02),
it find out key 1 and key 2 matches ,key 3 is extra.Then react insert extra item/children at top
of Original List.

Importtant points :

• A 'key' is a special string attribute you need to include when creating lists of elements
• Keys are give to elements a stable identity.
•Keys help React identity which items have changed,are added ,or removed.

Index as Key:

•When you don’t have stable IDs for rendered items, you may use the item index as a key

```
const todoItems = todos.map((todo, index) =>
  // Only do this if items have no stable IDs
   <li key={index}>  {todo.text} </li>
);
```

We don't recommend using indexes for keys if the order of items may change. becoz ...

#Index as key anti-pattern

```
 (#01)                                    (#02)                                      (#03)
 <ul>                                      <ul>                                       <ul>
      <li key="0">1</li>                    <li key="0"></li>                       <li key="0">1</li>
       <li key="1">2</li>   -------------> <li key="1"></li> --------------->     <li key="1">2</li>
       <li key="2">3</li>                    <li key="2"></li>                        <li key="2">3</li>
 <ul>                                          <li key="3"></li>                         <li key="3"></li>
                                              </ul>                                        </ul>
                                       (When You insert item in the beginning)        (After mutation)
```

•When you insert item in the beginning ,if key as Index then it add like (0,1,2,3...)[#02].
 Then react realize that by comparing list (#01 and #02) Lists ,key 3 is extra .
 so that list #03 renders in UI.

#When to use index as key ? (https://reactjs.org/docs/lists-and-keys.html)
• The items in your list do not have a unique id.
• If list is a static and will not change.
•The list will never be reordered or filtered.

Note :Infact React internally uses index as key in list ,if you not specify

For better solution :Uses id as key or npm id generated items.

12.React CSS /Styling React Components
= ==========================

There are four ways to style React component with CSS.They are:

  i)CSS stylesheets (Regular)
  ii)Inline styling
  iii)CSS Modules
  iv)CSS in JS Libaries

i)CSS stylesheets

•To use CSS stylesheets in react componnet, You can write your CSS styling in a separate file,
 just save the file with the .css file extension
•Import the stylesheet in your react component like
   import './App.css';

```
App.css.
---------
.primary{
 color:blue;
}
```

ii)Inline Styling

•To style an element with the inline style attribute, the value must be a JavaScript object:
• In JSX, JavaScript expressions are written inside curly braces

eg : <h1 style={{color: "red"}}>Hello Style!</h1>

#camelCased Property Names (in javascript)
• The inline CSS is written in a JavaScript object, properties with two names, like background-color, must be written with camel case syntax.
•Use backgroundColor instead of background-color:
eg :<h1 style={{backgroundColor: "lightblue"}}>Hello Style!</h1>

iii)CSS Modules
•Another way of adding styles to your application is to use CSS Modules.
•The CSS inside a module is available only for that component that imported it.
•You do not have to worry about name conflicts.
•Create the CSS module with the .module.css extension, example: mystyle.module.css.

```
mystyle.module.css.
----------------------
.secondary{
 color:gray;
}
```

•We can import the stylesheet in your component like
```
      import styles from './mystyle.module.css';
```

eg :
```
   <h1 className={styles.secondary}>Hello ....!</h1>
```

13.React Forms
= ==========
•In Regular Html ,form elements like input,textarea,..etc are responsible to handle the user input on their own and u pdate the respective value.
• But, in React , form elements are controlled by components are called controlled components

#controlled components
```
                  this.state={
                    name :''
                  }
          this.changeHandler=(event)=>{
             this.setState({name:event.target.value})
         }

 <input type="text" value={this.state.name} onChange={this.changeHandler}/>
```

• When ever user enter input value ,react component handle and update the respective value.

14.Lifecycle of Components
= ==================
• React component has a lifecycle methods ,it divided into four phases ,they are.

i)Mounting	:When an instance of component is being created and inserted into the DOM.
ii)Updating	:When a comp is being re-rendered as a result of changes to either its props or state
iii)Unmounting	:When a comp is being removed from the DOM
iv)Error Handling	:When there is an error during rendering ,in a lifecycle method, or in the constructor

of any child comp.

#lifecycle methods

i)Mounting	:constructor,static getDerivedStateFromProps,render, and componentDidMount
ii)Updating	:static getDerivedSateFromProps,shouldComponentUpdate,render, getSnapshotBeforeUpdate and componentDidUpdate.
iii)Unmounting	:componentWillUnmount
iv)Error Handling	:static getDerivedStateFromError and componentDidCatch

i)Mounting LifeCycle Methods :

#01 constructor •A special function that will get called whenever a new component is created.
- super(props)
- Initializing state (i.e this.state)
- Binding that event handlers
- Don not use HTTP request.

#02 static getDerivedSateFromProps(props,state)
- When the state of the component depends on changes in props over time.
- Set the state (When initial stage of component depends on props)
- Don not use HTTP request.

Note :
- The getDerivedStateFromProps() method is called right before rendering the element(s) in the DOM.

#03 render
- Only required method
- Read props and state and return JSX.
- Do not change state or interact with DOM or making ajax calls.
- If it have children,then Children components lifecycle methods are executed

#04 componentDidMount
- These method will be called only Once in whole lifecycle methods of given component.
- Invoked immediately after a component and all its children components have beeb rendered to the DOM.
- We can interact with DOM or perform any ajax calls to load data.

ii)Updating LifeCycle Methods :

#01 static getDerivedStateFromProps(props,state)
- These method is called every time ,when a component is re-renderd.
- Set the state (When initial stage of component depends on props)
- Don not use HTTP request.

#02 shouldComponentUpdate (nextProps,nextSate)
- It dictates if the component should re-rendered or not
- if false->React comp doesnot re-render ,true--->re-render
- Used for performance optimization.
- Don not use HTTP request.

#03 render
- Only required method
- Read props and state and return JSX.
- Do not change state or interact with DOM or making ajax calls.
- If it have children,then Children components lifecycle methods are executed

#04 getSnapshotBeforeUpdate(prevProps,prevState)
- •It accepts previous props and state .
- •It called right before the changes from the virtual DOM are to be reflected in the DOM.
- •Capture some information from the DOM.
- •These method will either return null or return a value.
  Returned value will be passed as third parameter to next method(componentDidUpdate).

#05 componentDidUpdate(prevProps,prevState,snapshot)
- •Called after the render is finished in the re-rendered cycles .
- •We can make ajax class (Before ajax calls compare props and decide).

iii)Unmounting LifeCycle method
  #componentWillUnmount() ---(Clean up method)
- •When a component is removed from the DOM, or unmounting as React likes to call it.
  or
- •These method is called immediately before a component is unmounted and destroyed.
  Usage : Canceling any network requests ,removing event handlers or invalidating or destroying or closing connection

iv)Error Handling LifeCycle method
#01 state getDerivedStateFromError(error)
#02 componentDidCatch(error,info)
• When there is an error either during rendering , in a lifecycle method ,or any children constructor

  #Error boundaries (Error Handling LifeCycle)
- •A class component that implements either one or both of the lifecycle methods getDerivedStateFromError or componentDidCatch becomes an error boundary
- • The static method getDerivedStateFromError is used to render a fallback UI after an error is thrown and the componentDidCatch method is used to log the error information
- • The placement of error boundaries also matter Whether entire application or one component

15.Fragment
= ========
• If you need to return multiple elements from a component. React Fragment helps in returning multiple elements.

Syntax:

```
<React.Fragment>
     <h2>Child-1</h2>
     <p> Child-2</p>
</React.Fragment>
```

Shorthand Fragment:

```
<>
     <h2>Child-1</h2>
     <p> Child-2</p>
</>
```

16.React Portals

== ========
• Portals provide a first-class way to render children into a DOM node that exists outside the DOM hierarchy of the parent component.('root)

•ReactDOM.createPortal(child, container)

•The first argument (child) is any renderable React child, such as an element, string, or fragment.
The second argument (container) is a DOM element.

#Before React v16
•Generally, when you want to return an element from a component's render method, it is mounted as a new div into t he DOM and render the children of the closest parent component.

```
render() {
// React mounts a new div into the DOM and renders the children into it
  return (
   <div>
     {this.props.children}
   </div>
  );
}
```

#portal
•But, sometimes we want to insert a child component into a different location in the DOM. It means React does not want to create a new div. We can do this by creating React portal.

```
render() {
 return ReactDOM.createPortal(
   this.props.children,
   myNode,
 );
}
```

Note :Actually we can use modals ,tooltips,overflow menus..etc

17.Pure Component (Only work for class component)
= ============

Regular Component	Pure Component
• A regular component does not implement the shouldComponentUpdate method. it always return true by default.	•A pure component on the other hand implements shouldComponentUpdate method with shallow props and state comparison

• A pure component implements shouldComponentUpdate with a shallow props and state comparison.

```
    SC of prevState with currentState    __Difference___________>                Re-render component
    SC of prevProps with currentProps                              >
```

#Shallow Comparison (SC)

Primitive Type (SC)
• a (SC) b return true if a and b have the same value and are of the same type.
Eg : string 'Apple' (SC) string 'Apple' return true

Complex Types (SC)
• a (SC) b return true if a and b reference that exact same object.
eg:
```
    i) var a={1,2,3,4};
      var b={1,2,3,4};
      var c=a;

      var ab=(a===b) ;//false
      var ac=(a===c);//true

   ii)  var a={x:1,y:2};
      var b={x:1,y:2};
      var c=a;

      var ab=(a===b) ;//false
      var ac=(a===c);//true
```

Note :
• We can create pure component by extending the PureComponent class.
• A pure component implements shouldComponentUpdate with a shallow props and state comparison.
•If there is no difference,the component not re-render-performance boost
•It is a good idea to ensure that all the children components aslo pure to avoid unexpected behaviour.
•Never mutate state.Always return a new object that reflects the state.

18.Memo
= =====
•Memo is similar to Pure Component (Only for class component) ,but memo is for functional component
•React.memo is a higher order component.

• A Memo component -- shallow comparison. of props.

SC of prevProps with currentProps __Difference___________> Re-render component

(or)
• If your component renders the same result given the same props, you can wrap it in a call to React.memo for a performance boost in some cases by memoizing the result.
• React.memo only checks for prop changes only.

19.React Refs
= ========
• Refs is the shorthand used for references in React.It is similar to keys in React.
•It is an attribute which makes it possible to store a reference to particular DOM nodes or React elements
•It provides a way to access React DOM nodes or React elements

We can create refs in two ways .they are..
 i) React.createRef()
 ii)Callback refs

i)React.createRef()
ii)Callback refs
```
      this.cbRef=null
      this.setCbRefs=(element)=>{
        this.cbRef=element
      }
```

#Passing refs to Children component from parent component

#Forwarding Ref from one component to another component

•Ref forwarding is a technique for automatically passing a ref through a component to one of its children
•We can be performed by making use of the React.forwardRef() method.
•However, it can be useful for some kinds of components, especially in reusable component libraries

20.Higher Order Components:(HOC)
= ===================
• If you want to reuse common piece of code,then we can use HOC's
• A higher-order component (HOC) is an advanced technique in React for reusing component logic

A higher-order component is a function that takes a component and returns a new component.

```
const NewComponent = higherOrderComponent(OriginalComponent);
```

• Whereas a component transforms props into UI, a higher-order component transforms a component into anothe
r component.
•HOCs are common in third-party React libraries, such as Redux's connect and Relay's createFragmentContainer. .
.etc

HOC?#To share common functionality between components

20.Render props
= ==========
• The term 'render prop' refer to a technique for sharing code between React components using a
prop whose value is a function.
•A component with a render prop takes a function that returns a React element and calls it instead of implementing it
s own render logic.

```
<DataProvider render={data => (
  <h1>Hello {data.target}</h1>
)}/>
```

•Libraries that use render props include React Router, Downshift and Formik.

21.Context
= ======
•Context provides a way to pass data through the component tree without having to pass props manually
at every level

```
   #component tree
                              App Component
                                    |              (username)
                        --------------------------------
                        |          |                |
                        A          B                C
                                   |                |
                                   D            E
                                                |
                                                F
```
• If you want to pass data (ex :username Comp C-to -F) ,Actually we can pass props manually at every level like C-E,E-F.
•Instead of passing props manually at every level we can use context

Steps
i)Create the context
ii)Provide a context value
iii)Consume the context value

i)Create the context
```
//#Step :01
const UserContext=React.createContext()

const UserProvider=UserContext.Provider

const UserConsumer=UserContext.Consumer
```

ii)Provide a context value
```
 {/* Step:02 */}
     <UserProvider value="Viru">
        <C/>
   </UserProvider>
```

iii)Consume the context value
```
 //Step:03
 <UserConsumer>
              {
                (username)=>{
                    return <h2>Hello {username}....!</h2>
                }
              }
 </UserConsumer>
```

#Consume the context value in different ways
• Componet*.contextType=UserContext
eg : E.contextType=UserContext

• static contextType=UserContext (If class accpets public fields)
#limitations of contextType
.It only works for class components
. You can subscribe single context using contextType.

=============================HTTP (axios)=====================================
Promise based HTTP client for the browser and node.js