PROJECT TRAINING WORKSHOP

React JS

RAAFI
Builds You Best

React JS

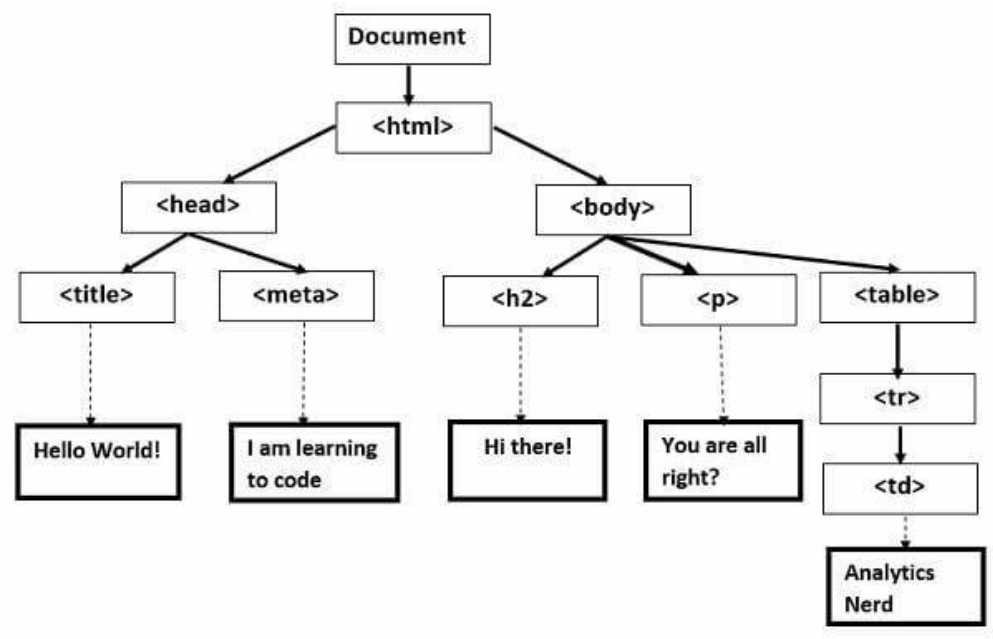
- **What is React JS?**
  - **A Library which supports. JSX =. JavaScript + XML (i.e., Design Language + Programming Language)**

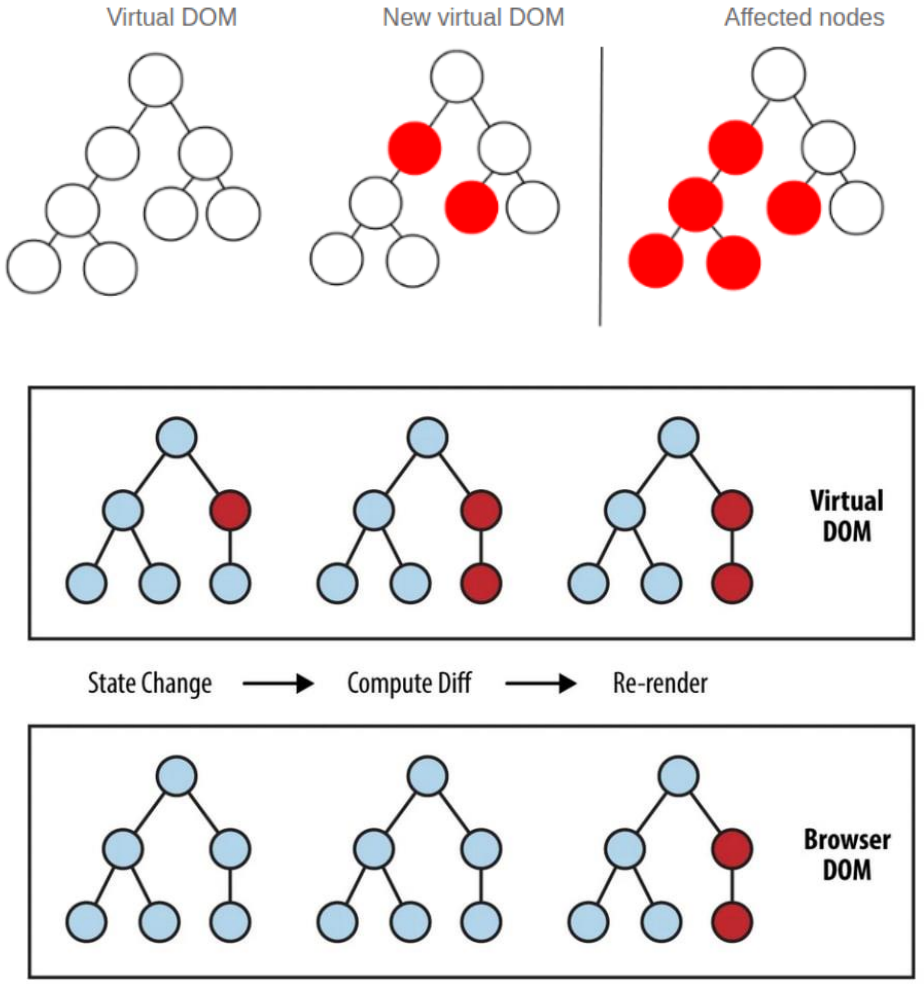
- **What are the roles of React JS?**

Virtual DOM	User Interface Building	Reusable Components & UI Elements	State Management
Lifecycle Hooks	React Native	Server-Side Rendering	Testable Components
Eco System & Community			

- **What are the types of React JS Application?**
  - **Single Page Applications**
  - **Web Applications**
  - **Mobile Applications**
  - **Progressive Web Apps**
  - **Real Time Applications**

React JS – Understanding Real & Virtual DOM





- Observable Pattern
- Diffing

RAAFI
Builds You Best

React JS – Creating React App

Pre-requisites	Node JS & NPM (Node Package Manager
Creating a React App	*npx create-react-app my-react-app*
Select the folder	*cd my-react-app*
Running the app	*npm start*
Build for Production	*npm run build*

React JS – Exploring the Folder Structure

```
my-react-app/
  ├── node_modules/
  ├── public/
  |   ├── index.html
  |   ├── favicon.ico
  |   └── manifest.json
  ├── src/
  |   ├── index.js
  |   ├── App.js
  |   ├── components/
  |   |   ├── Component1.js
  |   |   ├── Component2.js
  |   ├── containers/
  |   ├── styles/
  |   |   ├── App.css
  |   ├── assets/
  |   ├── services/
  |   ├── utils/
  ├── package.json
  ├── README.md
```

React JS – Understanding JSX

My First Component

```
import React from 'react';

function MyComponent() {
  return <div>Hello, World!</div>;
}

export default MyComponent;
```

Including in Index.js

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

Important React API

- createRoot()
- createElement()
- appendChild()
- render()

```
//without JSX
const myElement = React.createElement('div', {}, 'Hello, World!');
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);

//with JSX
const myElement = <div>Hello, World!</div>;
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

BABEL

```
1 const element = <div>Hello World</div>;
```

```
1 const element = /*#__PURE__*/React.createElement("div",
  null, "Hello World");
```

React JS – Adding Expression, Attributes and Events Handlings

```
import React from 'react';

function Greet(props)
{
  return <div>Hello, {props.name} </div>;
}

export {Greet};
```

```
import React from 'react';

function Button() {
  return (
    <button
      className="btn"
      onClick={() => alert('You Clicked Me!')}
    >
      Click me
    </button>
  );
}
export {Button};
```

```
import { Greet} from './MyComponent';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
    <Greet name="Hi VueData"/>
  </React.StrictMode>
);
```

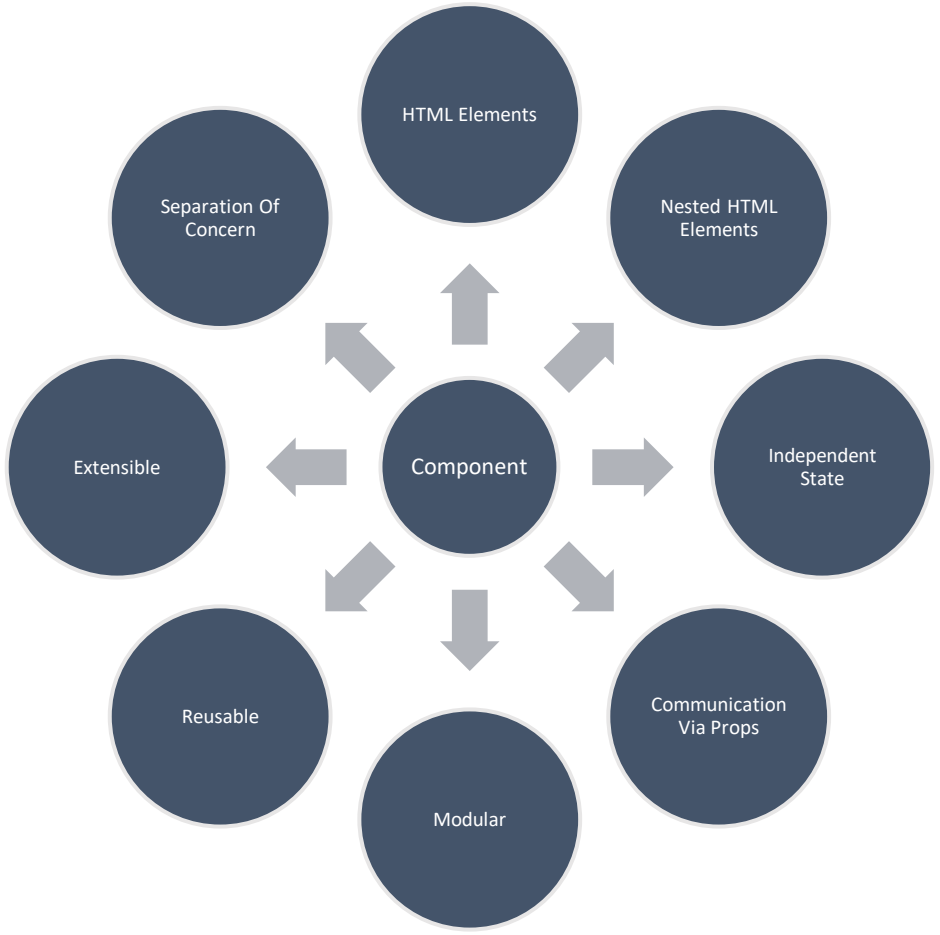
```
import { Button } from './MyComponent';

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

React JS – Nesting, Dynamic Content, Comment

```
import React from 'react';
import {Button, CancelButton} from './MyComponent';

function MyCompositeComponent() {
    return (
      <div>
        <Button fn="Hi" ln=" VueData Trainees!!" />
        <CancelButton />
    </div>
    );
  }

  export {MyCompositeComponent};
```

```
import React from 'react';
import {concat} from './utils';

function Button(props) {
  if(props.action === 'Update')
  {
    return (
      <button
        className="btn"
        onClick={() => alert(concat(props.fn, props.ln))}
      >
        {/* This button is update button */}
        {props.action}
      </button>
    );
  }
  else
  {
    return (
      <button
        className="btn"
        onClick={() => alert(concat(props.fn, props.ln))}
      >
        {/* This button is Insert button */}
        Insert
      </button>
    );
  }
}
```

React JS – Understanding Components



React JS – Types of Components

Function

```
import React from 'react';
import {Button, CancelButton} from './MyComponent';

function MyCompositeComponent() {
    return (
      <div>
        <Button fn="Hi" ln=" VueData Trainees!!" action="Update" />
        <CancelButton />
    </div>
    );
  }

  export {MyCompositeComponent};
```

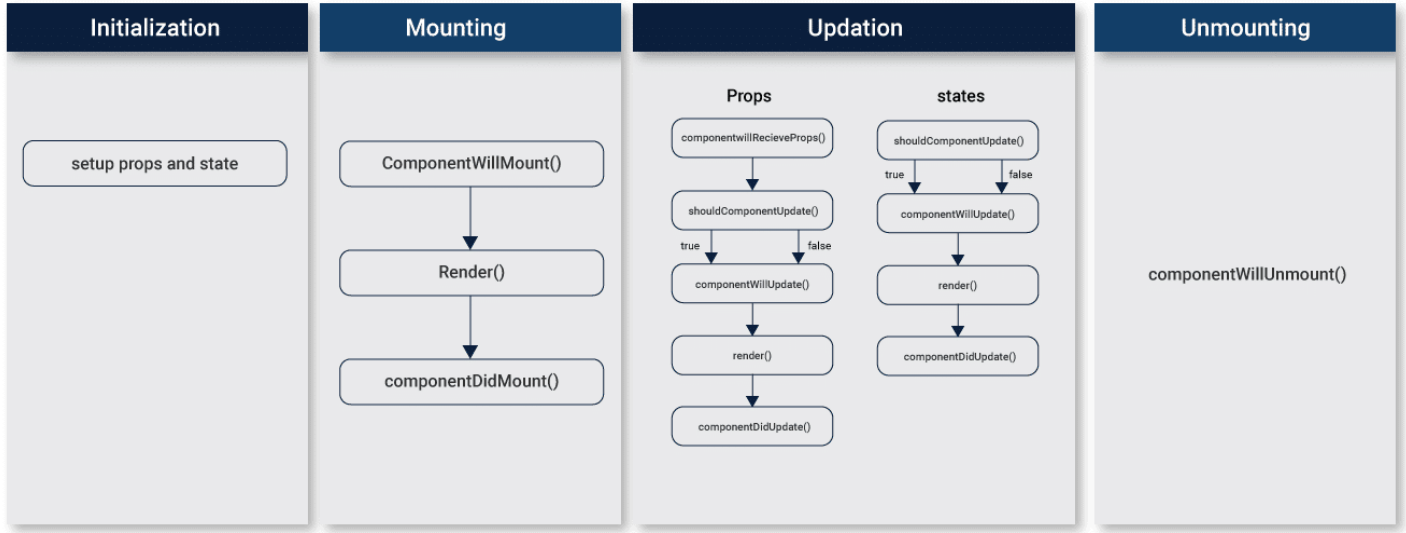
Class

```
import React, { Component } from 'react';

class Toggle extends Component {
  constructor(props) {
    super(props);
    this.state = { mode: 'ON' };
  }

  toggleme = () => {
    this.setState({ mode : this.state.mode === 'ON'? 'OFF': 'ON' });
  };

  render() {
    return (
      <div>
        <p> MODE: {this.state.mode}</p>
        <button onClick={this.toggleme}>Press Me</button>
      </div>
    );
  }
}

export default Toggle;
```

Functional Components – Simple, Stateless#

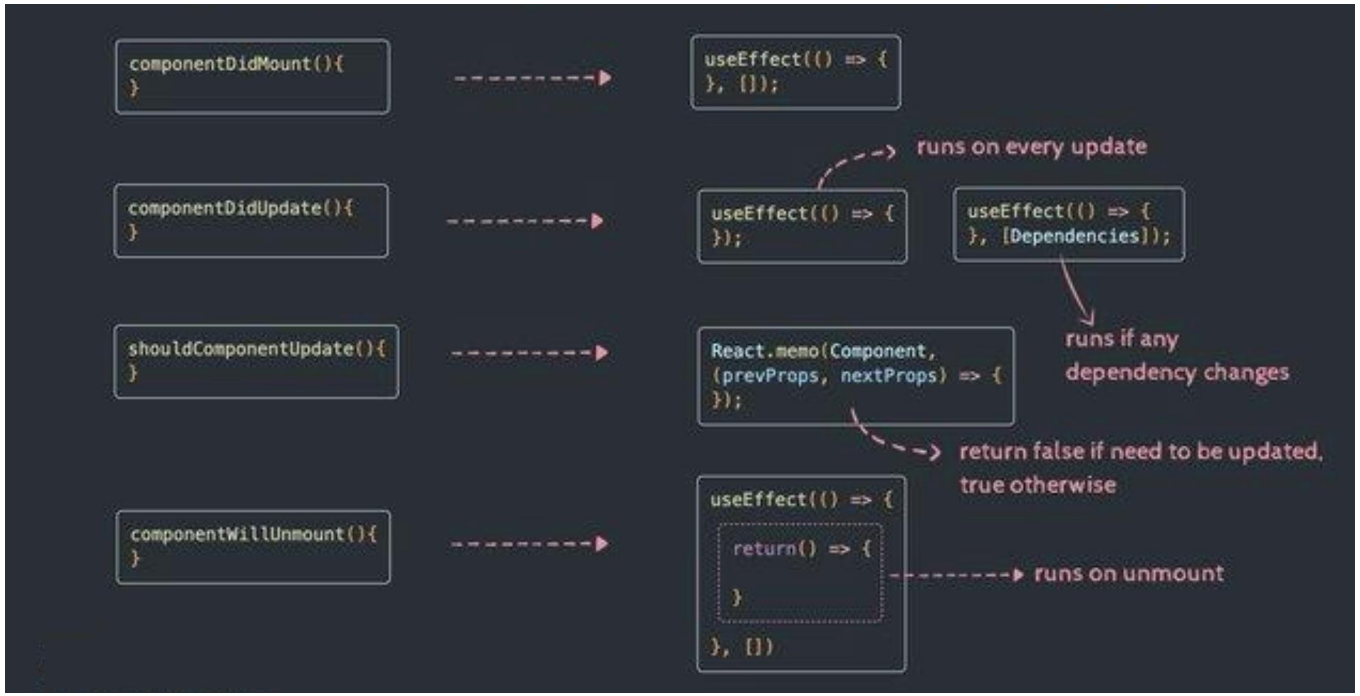
Class Components – Stateful

React JS – Class Components Lifecycle




```jsx
import React, { Component } from 'react';
class LifecycleDemo extends Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }
  componentDidMount() {
    // Called after component is mounted
    console.log('Component mounted');
  }
  componentDidUpdate(prevProps, prevState) {
    // Called after component updates
    console.log('Component updated');
  }
  componentWillUnmount() {
    // Called before component is unmounted
    console.log('Component will unmount');
  }
  incrementCount = () => {
    this.setState({ count: this.state.count + 1 });
  }
  render() {
    return (
      <div>
        <h1>Class-Based Component Lifecycle</h1>
        <p>Count: {this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
      </div>
    );
  }
}
export default LifecycleDemo;
```

React JS – Functional Component Lifecycle (Hooks)




```jsx
import React, { useState, useEffect } from 'react';

function LifecycleDemoHooks() {
  const [count, setCount] = useState(0);

  // componentDidMount
  useEffect(() => {
    console.log('Component mounted');
    return () => {
      // componentWillUnmount
      console.log('Component will unmount');
    };
  }, []);

  // componentDidUpdate
  useEffect(() => {
    console.log('Component updated');
  }, [count]);

  const incrementCount = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>React Hooks Component Lifecycle</h1>
      <p>Count: {count}</p>
      <button onClick={incrementCount}>Increment</button>
    </div>
  );
}

export default LifecycleDemoHooks;
```

React JS – Hooks

useState

useEffect

useContext

useReducer

useRef

useMemo

useCallback

customHook

React JS – Hooks – useState()

```
const [state, setState] = useState(initialState);
```

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}

export {Counter};
```

- Usage
  - Adding state to a component

React JS – Hooks – useEffect()

```
useEffect(setup, dependencies?)
```

```
import React, { useState, useEffect } from 'react';

function DataFetcher() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Simulate data fetching
    fetch('https://jsonplaceholder.typicode.com/todos/1')
      .then((response) => response.json())
      .then((json) => setData(json));
  }, []);

  return <div>{data ? <p>Data: {data.title}</p> : <p>Loading...</p>}</div>;
}
```

- Usage
  - Connecting to an external system
  - Fetching data with Effects
  - Specifying reactive dependencies

React JS – Hooks – useContext()

```
const value = useContext(SomeContext)
```

- Usage
  - Passing data deeply into the tree
  - Specifying a fallback default value

```jsx
import React, { useState, useContext } from 'react';
// Context setup
const ThemeContext = React.createContext('light');

function ThemeProvider({ children }) {
  const [theme, setTheme] = useState('light');

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}
function ThemeToggler() {
  const { theme, setTheme } = useContext(ThemeContext);
  const toggleTheme = () => {
    setTheme(theme === 'light' ? 'dark' : 'light');
  };
  return (
    <button onClick={toggleTheme}>
      Toggle Theme ({theme})
    </button>
  );
}
function App1() {
  return (
    <ThemeProvider>
      <div>
        <h1>React Hooks Example</h1>
        <ThemeToggler />
      </div>
    </ThemeProvider>
  );
}
export default App1;
```

React JS – Hooks – useReducer()

```
const [state, dispatch] = useReducer(reducer, initialArg, init?)
```

```
import React, { useReducer } from 'react';

const initialState = { count: 0 };

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      return state;
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'increment' })}>Increment</button>
      <button onClick={() => dispatch({ type: 'decrement' })}>Decrement</button>
    </div>
  );
}
```

Reducer - consolidate all the state update logic outside your component in a single function

- Usage
  - Adding a reducer to a component

React JS – Hooks – useRef()

```
const ref = useRef(initialValue)
```

```
import { useRef } from 'react';

export default function Form() {
  const inputRef = useRef(null);

  function handleClick() {
    inputRef.current.focus();
  }

  return (
    <>
      <input ref={inputRef} />
      <button onClick={handleClick}>
        Focus the input
      </button>
    </>
  );
}
```

- Usage
  - Referencing a value with a ref

React JS – Hooks – useMemo()

```
const cachedValue = useMemo(calculateValue, dependencies)
```

```jsx
import { useState } from 'react';
import { createTodos } from './utils.js';
import TodoList from './TodoList.js';

const todos = createTodos();

export default function App() {
  const [tab, setTab] = useState('all');
  const [isDark, setIsDark] = useState(false);
  return (
    <>
      <button onClick={() => setTab('all')}>
        All
      </button>
      <button onClick={() => setTab('active')}>
        Active
      </button>
      <button onClick={() => setTab('completed')}>
        Completed
      </button>
      <br />
      <label>
        <input
          type="checkbox"
          checked={isDark}
          onChange={e => setIsDark(e.target.checked)}
        />
        Dark mode
      </label>
      <hr />
      <TodoList
        todos={todos}
        tab={tab}
        theme={isDark ? 'dark' : 'light'}
      />
    </>
  );
}
```

```jsx
import { useMemo } from 'react';
import { filterTodos } from './utils.js'

export default function TodoList({ todos, theme, tab }) {
  const visibleTodos = useMemo(
    () => filterTodos(todos, tab),
    [todos, tab]
  );
  return (
    <div className={theme}>
      <ul>
        {visibleTodos.map(todo => (
          <li key={todo.id}>
            {todo.completed ?
              <s>{todo.text}</s> :
              todo.text
            }
          </li>
        ))}
      </ul>
    </div>
  );
}
```

```jsx
export function filterTodos(todos, tab) {
  let startTime = performance.now();
  while (performance.now() - startTime < 500) {
    // Do nothing for 500 ms to emulate extremely slow code
  }

  return todos.filter(todo => {
    if (tab === 'all') {
      return true;
    } else if (tab === 'active') {
      return !todo.completed;
    } else if (tab === 'completed') {
      return todo.completed;
    }
  });
}
```

- Usage
  - Skipping expensive recalculations
  - Skipping re-rendering of components
  - Memoizing a function

React JS – Hooks – useCallback()

```
const cachedFn = useCallback(fn, dependencies)
```

```
import React, { useState, useCallback } from 'react';

function ClickCounter() {
  const [count, setCount] = useState(0);

  const increment = useCallback(() => {
    setCount(count + 1);
  }, [count]);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
}


export default ClickCounter;
```

- Usage
  - Skipping re-rendering of components

RAAFI
Builds You Best

React JS – Hooks – CustomHook()

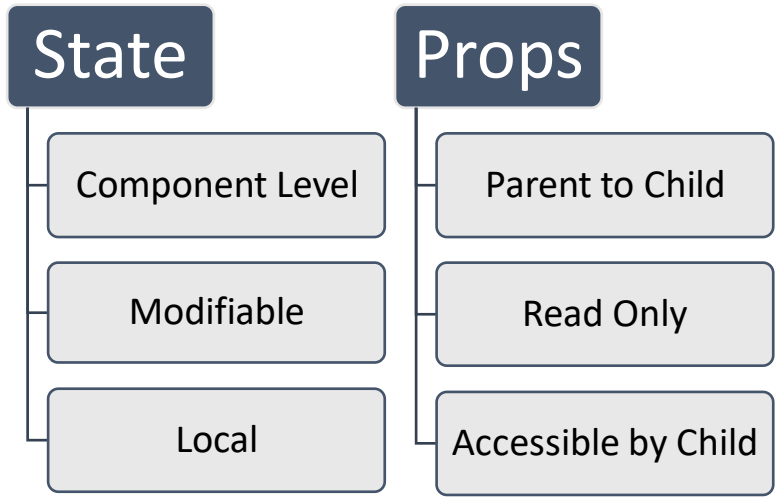
```jsx
import React from 'react';
import useCustomerData from './CustomHook';

// React component using the custom hook
function CustomerManagement() {
  const { addCustomer, removeCustomer, getAllCustomers } = useCustomerData();

  const handleAddCustomer = () => {
    const newCustomer = {
      id: Date.now(), // You can generate a unique ID here
      name: 'John Doe',
      email: 'john@example.com',
    };
    addCustomer(newCustomer);
  };

  const handleRemoveCustomer = (customerId) => {
    removeCustomer(customerId);
  };

  const customerList = getAllCustomers();

  return (
    <div>
      <h1>Customer Management</h1>
      <button onClick={handleAddCustomer}>Add Customer</button>
      <ul>
        {customerList.map((customer) => (
          <li key={customer.id}>
            {customer.name} - {customer.email}
            <button onClick={() => handleRemoveCustomer(customer.id)}>Remove</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default CustomerManagement;
```

```jsx
// Custom hook to manage customer data
function useCustomerData() {
  const [customers, setCustomers] = useState([]);

  // Function to add a new customer
  const addCustomer = (customer) => {
    setCustomers([...customers, customer]);
  };

  // Function to remove a customer by ID
  const removeCustomer = (customerId) => {
    const updatedCustomers = customers.filter(
      (customer) => customer.id !== customerId);
    setCustomers(updatedCustomers);
  };

  // Function to get all customers
  const getAllCustomers = () => {
    return customers;
  };

  return {
    addCustomer,
    removeCustomer,
    getAllCustomers,
  };
}
export default useCustomerData;
```

React JS – State Vs Props

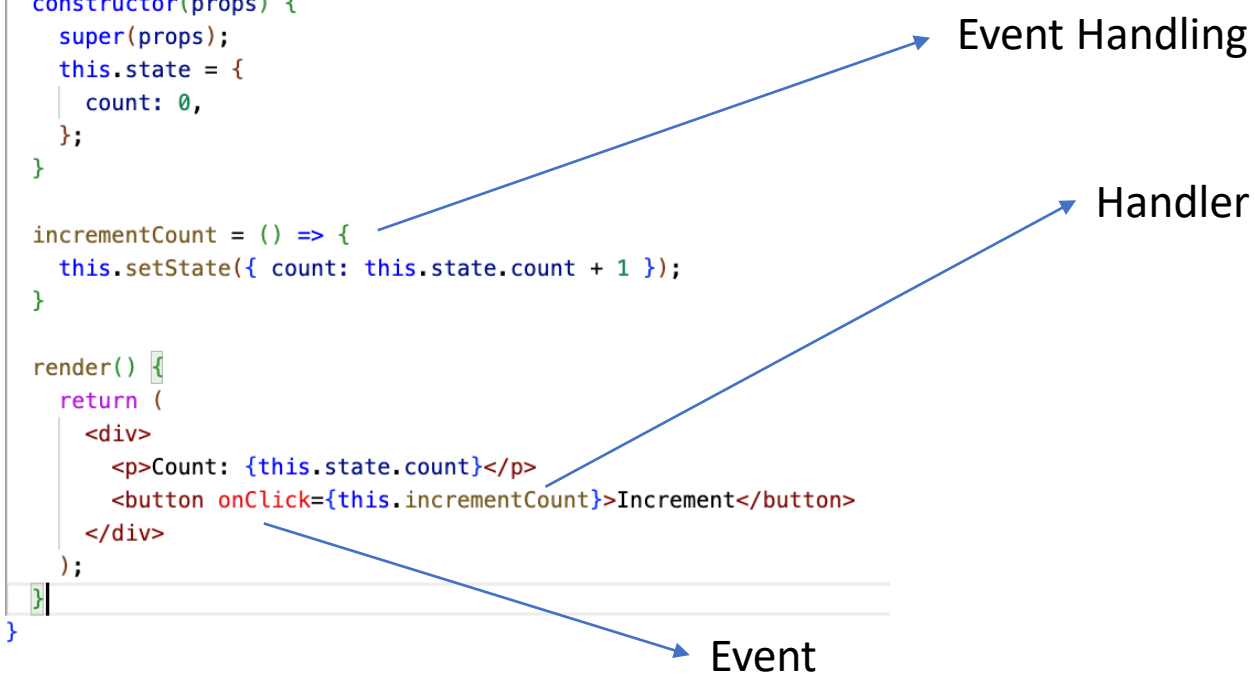
```jsx
import React, { Component } from 'react';

class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  incrementCount = () => {
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
      </div>
    );
  }
}

export default Counter;
```

```jsx
import React from 'react';

function Greet(props) {
  return <h1>Hello, {props.name}!</h1>;
}

function Parent() {
  return <Greet name="John" />;
}

export default Parent;
```



React JS – Event Handling & Form Submission

```jsx
import React, { Component } from 'react';

class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  incrementCount = () => {
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
      </div>
    );
  }
}

export default Counter;
```




```jsx
import React, { Component } from 'react';

class FormSubmission extends Component {
  constructor(props) {
    super(props);
    this.state = { inputText: '' };
  }

  handleInputChange = (event) => {
    this.setState({ inputText: event.target.value });
  }

  handleSubmit = (event) => {
    event.preventDefault();
    alert(`You submitted: ${this.state.inputText}`);
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <input
          type="text"
          value={this.state.inputText}
          onChange={this.handleInputChange}
        />
        <button type="submit">Submit</button>
      </form>
    );
  }
}

export default FormSubmission;
```

React JS – Conditional Rendering

```jsx
import React from 'react';

function App() {
  const isLoggedIn = true;

  return (
    <div>
      {isLoggedIn ? (
        <WelcomeUser />
      ) : (
        <LoginPrompt />
      )}
    </div>
  );
}

function WelcomeUser() {
  return <h1>Welcome, User!</h1>;
}

function LoginPrompt() {
  return <p>Please log in to continue.</p>;
}

export default App;
```

```jsx
import React from 'react';

function App() {
  const isMember = false;

  return (
    <div>
      {isMember ? <MemberContent /> : <GuestContent />}
    </div>
  );
}

function MemberContent() {
  return <p>Welcome, member!</p>;
}

function GuestContent() {
  return <p>Join us to access more content.</p>;
}

export default App;
```

```jsx
import React from 'react';

function App() {
  const showMessage = true;

  return (
    <div>
      {showMessage && <div>Hi VueData</div>}
    </div>
  );
}

export default App;
```

```jsx
import React from 'react';

function App() {
  const items = [1, 2, 3, 4, 5];

  return (
    <div>
      {renderList(items)}
    </div>
  );
}

function renderList(items) {
  if (items.length === 0) {
    return <p>No items available.</p>;
  } else {
    return (
      <ul>
        {items.map((item) => (
          <li key={item}>{item}</li>
        ))}
      </ul>
    );
  }
}

export default App;
```

React JS – React Router

Install : npm install react-router-dom

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Home from './Home';
import About from './About';
import Contact from './Contact';

function Landing() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </Router>
  );
}

export default Landing;
```

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />}>
    <Route path="/team" element={<Team />} />
    <Route path="/history" element={<History />} />
  </Route>
  <Route path="/contact" element={<Contact />} />
</Routes>
```

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/users/:userId" element={<UserProfile />} />
  <Route path="/products/:productId" element={<ProductDetail />} />
</Routes>
```

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  {/* Error route */}
  <Route path="*" element={<NotFound />} />
</Routes>
```

React JS – Handling API Request & Response

```jsx
import React, { Component } from 'react';

class Comments extends Component {
  constructor() {
    super();
    this.state = {
      data: [],
      loading: true,
    };
  }
  componentDidMount() {
    fetch('https://jsonplaceholder.typicode.com/comments')
      .then((response) => response.json())
      .then((data) => {
        console.log(data);
        this.setState({ data, loading: false });
      })
      .catch((error) => {
        console.error('Error fetching data:', error);
        this.setState({ loading: false });
      });
  }
  render() {
    const { data, loading } = this.state;
    return (
      <div>
        {loading ? (
          <p>Loading...</p>
        ) : (
          <ul>
            {data.map((item) => (
              <li key={item.id}>ABC{item.name}</li>
            ))}
          </ul>
        )}
      </div>
    );
  }
}
export default Comments;
```

```jsx
import React, { useState, useEffect } from 'react';

function Comments() {
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/comments')
      .then((response) => response.json())
      .then((data) => {
        setData(data);
        setLoading(false);
      })
      .catch((error) => {
        console.error('Error fetching data:', error);
        setLoading(false);
      });
  }, []);

  return (
    <div>
      {loading ? (
        <p>Loading...</p>
      ) : (
        <ul>
          {data.map((item) => (
            <li key={item.id}>{item.name}</li>
          ))}
        </ul>
      )}
    </div>
  );
}

export default Comments;
```

```jsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function Comments() {
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const apiUrl = 'https://jsonplaceholder.typicode.com/comments';
    axios
      .get(apiUrl)
      .then((response) => {
        setData(response.data);
        setLoading(false);
      })
      .catch((error) => {
        console.error('Error fetching data:', error);
        setLoading(false);
      });
  }, []);

  return (
    <div>
      {loading ? (
        <p>Loading...</p>
      ) : (
        <ul>
          {data.map((item) => (
            <li key={item.id}> {item.name}</li>
          ))}
        </ul>
      )}
    </div>
  );
}

export default Comments;
```

React JS – Error Handling

```jsx
function MyComponent() {
  const [error, setError] = useState(null);

  const handleAction = () => {
    try {
      // Code that might throw an error
    } catch (err) {
      setError('An error occurred: ' + err.message);
    }
  }

  return (
    <div>
      {error && <p>{error}</p>}
      <button onClick={handleAction}>Perform Action</button>
    </div>
  );
}

async function fetchData() {
  try {
    const response = await fetch('https://example.com/api/data');
    if (!response.ok) {
      throw new Error('Network request failed');
    }
    const data = await response.json();
    return data;
  } catch (error) {
    console.error(error);
    // Display an error message to the user
  }
}
```

```jsx
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  componentDidCatch(error, errorInfo) {
    // You can log the error or report it to an error tracking service
    console.error(error, errorInfo);
    this.setState({ hasError: true });
  }

  render() {
    if (this.state.hasError) {
      return <div>Something went wrong.</div>;
    }
    return this.props.children;
  }
}
```

```jsx
import React, { useState } from 'react';

function ErrorBoundary({ children }) {
  const [hasError, setHasError] = useState(false);
  const [error, setError] = useState(null);

  const componentDidCatch = (error, errorInfo) => {
    setHasError(true);
    setError(error);
    // You can log the error or send it to a logging service.
    console.error('Error:', error);
    console.error('Error Info:', errorInfo);
  };

  if (hasError) {
    return (
      <div>
        <h2>Something went wrong</h2>
        <p>We apologize for the inconvenience. Please try again later.</p>
        <p>Error Details: {error && error.toString()}</p>
      </div>
    );
  }

  return children;
}
```

```jsx
function App() {
  return (
    <div>
      <h1>Error Boundary Example</h1>
      <ErrorBoundary>
        <ChildComponent />
      </ErrorBoundary>
    </div>
  );
}
```

React JS – Testing Components

npm install --save-dev jest @testing-library/react @testing-library/jest-dom

```jsx
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <p data-testid="count-display">Count: {count}</p>
      <button data-testid="increment-button" onClick={increment}>Increment</button>
      <button data-testid="decrement-button" onClick={decrement}>Decrement</button>
    </div>
  );
}

export default Counter;
```

```jsx
// Counter.test.js

import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import Counter from './CounterTest';

test('renders the Counter component', () => {
  const { getByText, getByTestId } = render(<Counter />);
  const countDisplay = getByTestId('count-display');
  const incrementButton = getByTestId('increment-button');
  const decrementButton = getByTestId('decrement-button');

  // Check if the component renders initially with a count of 0
  expect(countDisplay).toHaveTextContent('Count: 0');

  // Simulate a click on the "Increment" button and check if the count increases
  fireEvent.click(incrementButton);
  expect(countDisplay).toHaveTextContent('Count: 1');

  // Simulate a click on the "Decrement" button and check if the count decreases
  fireEvent.click(decrementButton);
  expect(countDisplay).toHaveTextContent('Count: 0');
});
```

```
PASS src/Counter1.test.js
  ✓ Increment the Counter component (16ms)
  ✓ Decrement the Counter component (3ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        4.213s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

React JS – Assignment

Create React JS application (Components) for Meeting room booking application using Mongo DB