

What is React.js?

Definition: React.js is a JavaScript library developed by Facebook for building fast, dynamic, and interactive user interfaces (especially SPAs).

Key Points: - Component-based architecture - Virtual DOM for fast rendering - One-way data binding

Example:

```
const element = <h1>Hello, React!</h1>;
```

Features of React.js

1. **Component-Based:** Build reusable UI parts
2. **Virtual DOM:** Faster rendering
3. **One-Way Data Binding:** Predictable data flow
4. **JSX:** HTML inside JavaScript
5. **Hooks:** Add state & logic to function components
6. **Declarative UI:** Describe what the UI should look like

Example:

```
function App() {  
  return <h2>My React App</h2>;  
}
```

JSX (JavaScript XML)

Definition: JSX allows writing HTML-like syntax inside JavaScript.

Example 1:

```
function Hello() {  
  return <h1>Welcome to React!</h1>;  
}
```

Example 2:

```
const element = <p>Hello, {name}!</p>;
```

Key Points: - Must have one parent element - Embed JavaScript with `{}` - Converted to `React.createElement()` under the hood

React Components

Definition: Components are independent, reusable pieces of UI.

1. Function Component (Modern)

```
function Welcome() {  
  return <h2>Hello, User!</h2>;  
}
```

2. Class Component (Older)

```
class Welcome extends React.Component {  
  render() {  
    return <h2>Hello, User!</h2>;  
  }  
}
```

Usage:

```
<Welcome />
```

Props (Properties)

Props are read-only values passed from parent to child.

Example 1:

```
function Welcome(props) {  
  return <h3>Hello, {props.name}</h3>;  
}  
<Welcome name="Ayesha" />
```

Example 2:

```
function UserCard({ user }) {  
  return <p>User: {user}</p>;  
}
```

State

State stores dynamic data in a component.

Example 1 (useState):

```
import { useState } from 'react';  
function Counter() {  
  const [count, setCount] = useState(0);  
  return (  
    <div>  
      <p>Count: {count}</p>  
      <button onClick={() => setCount(count + 1)}>Increase</button>  
    </div>  
  );  
}
```

Example 2:

```
function Toggle() {  
  const [isOn, setIsOn] = useState(false);  
  return <button onClick={() => setIsOn(!isOn)}>{isOn ? 'ON' : 'OFF'}</button>;  
}
```

React Hooks

Hook	Description
useState	Manage component state
useEffect	Side effects (API calls, timers)
useContext	Access global data
useRef	Access DOM elements
useReducer	Complex state logic

Example (useEffect):

```
useEffect(() => {  
  console.log('Component Mounted');  
}, []);
```

Example (useContext):

```
const UserContext = React.createContext();  
function App() {  
  return <UserContext.Provider value="Ayesha"><Child /></UserContext.Provider>;  
}  
function Child() {  
  const user = React.useContext(UserContext);  
  return <h2>Hello, {user}</h2>;  
}
```

API Integration

Example 1 (fetch):

```
useEffect(() => {  
  fetch('https://jsonplaceholder.typicode.com/users')  
    .then(res => res.json())  
    .then(data => setUsers(data));  
}, []);
```

Example 2 (Axios):

```
import axios from 'axios';
useEffect(() => {
  axios.get('/api/users').then(res => setUsers(res.data));
}, []);
```

Context API

Used for global state management.

Example:

```
const ThemeContext = React.createContext('light');
function App() {
  return <ThemeContext.Provider value="dark"><Toolbar /></ThemeContext.Provider>;
}
function Toolbar() {
  const theme = React.useContext(ThemeContext);
  return <h1>Theme: {theme}</h1>;
}
```

Redux (State Management)

Key Concepts: Store, Action, Reducer

Example (Reducer):

```
const initialState = { count: 0 };
function reducer(state = initialState, action) {
  switch(action.type) {
    case 'INCREMENT': return { count: state.count + 1 };
    default: return state;
  }
}
```

Virtual DOM

React keeps a virtual copy of the DOM to **efficiently update only changed parts**, improving performance.

Lifecycle Methods (Class Components)

Method	Description
componentDidMount	After component mounts
componentDidUpdate	After state/props update
componentWillUnmount	Before component is removed

React Router

Example:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Element vs Component

Element	Component
Describes what to render	Defines how something renders
Created using JSX	Function/Class
Example: <h1>Hello</h1>	Example: function Hello()

Top React Interview Questions

1. What is React.js?
2. What are components?
3. Difference between class and function components?
4. What is JSX?
5. What is Virtual DOM?
6. Props vs State?
7. React Hooks?
8. Explain useState and useEffect.
9. Context API?
10. Redux basics?
11. Data flow in React?
12. Keys in React?
13. React Router?
14. Performance optimization in React?

Prepared for study, interview, and revision purposes.