REACTJS

Wednesday, November 27, 2024 12:16 PM

What is ReactJS?

React.js is javascript library
The mainly focus on building UI as soon as possible
Single Page Application

Why react is fast

1. Virtual DOM
2. HOOK

DOM v/s Virtual DOM

React1

Library	Framework
A set of assistance modules, objects, classes, functions, pre-written code, and so on.	Includes a variety of APIs, compilers, support applications, libraries, and so on.
Can be easily substituted by another library.	Are tough to replace.
When we call a method from a library, we are in control.	Inversion of control, i.e. the framework calls us.
Since developing a library needs less code, performance and load time are improved.	The construction of a framework necessitates large amounts of code, which reduces performance and increases load time.
Libraries can be simply linked into existing programs to add specific functionality.	It is tough to incorporate a framework seamlessly into an existing project.

Feacture of ReactJS

- Single page application
- Hook Support
- Virtual DOM
- Component-based architecture
- Easy to learn
- Performance
- One-way data binding
- JSX
- Resuablity

REACT RUN USING CDN :-

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <script
      src="https://unpkg.com/react@18/umd/react.development.js"
      crossorigin
    ></script>
    <script
      src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
      crossorigin
    ></script>
    <script
      src="https://unpkg.com/@babel/standalone@7.10.3/babel.min.js"
      crossorigin
    ></script>
  </head>
  <body>
    <div id="mydiv"></div>
    <script type="text/babel">
      function Test() {
        const test = () => {
        alert("test");
      };
        return <button onClick={test}>Click me</button>;
      }
      class Hello extends React.Component {
        render() {
          return (
            <div>
              Hello React WIth Code <Test />
            </div>
          );
        }
      }
      ReactDOM.render(<Hello />, document.getElementById("mydiv"));
    </script>
  </body>
</html>
```

How to create react app :

1 Way :
bun create vite <app_name>
cd app_name
bun install
bun run dev

2 Way:
npx create-react-app my-app

3 Way:
npm init react-app my-app

```
my-app/
├── public/
│   ├── index.html
│   ├── favicon.ico
│   └── manifest.json
├── src/
│   ├── components/
│   │   └── Header.jsx
│   ├── App.css
│   ├── App.jsx
│   ├── index.css
│   ├── index.jsx
├── .gitignore
├── package.json
├── README.md
└── yarn.lock / package-lock.json
```

- **public/**: Contains static assets like index.html and images that don't change during runtime.
- **src/**: Contains all source code, including components, styles, and utilities.
- **App.jsx**: The root React component.
- **index.jsx**: The entry point where React DOM renders the app.

Aspect	package.json	package-lock.json
Purpose	Outlines project metadata, dependencies, scripts, and more.	Outlines project metadata, dependencies, scripts, and more.
Creation	Created manually by the developer or via npm init.	Automatically generated by npm when running npm install.
Content	Includes project name, version, dependencies, devDependencies, scripts, etc.	Contains detailed version information and the source for each installed package.
Dependency Versions	Lists dependencies with version ranges.	Specifies exact versions of each package, ensuring consistency.
Project Collaboration	Ensures that the correct packages are installed.	Ensures that the same version of packages is used by all developers.
Version Control	Typically checked into version control.	Also checked into version control to lock dependency versions for all users.
Update Frequency	Updated manually when adding or updating packages.	Updated automatically whenever packages are added or updated.
User Intervention	Requires user intervention for updates or changes.	Managed by npm, requiring no direct user modification.

What is component?

Component nothing but piece of code that can resue.

Types :

1. Class based components
2. Functional base components
3. Higher Order components
4. Pure components

5. Controlled components
6. Uncontrolled components

User.jsx

Functional Based Components:

```
function User() {
  return (
    <>
      <h1>Hello User</h1>
    </>
  );
}
export default User;
```

How to import :

```
import User from './components/User'
function App() {
  return (
    <>
      <h1>Hello React</h1>
      <User></User>
    </>
  )
}
export default App
```

Class based componets:

User1.jsx

```
import React,{Component} from "react";
class User1 extends Component{
    render()
    {
        return(
            <>
            <h3>Hello User1</h3>
            </>
        )
    }
}
export default User1;
```

```
How to import: same as above
```

What is JSX?

JSX (JavaScript Syntax Extension) is a syntax extension for JavaScript that allows developers to write HTML-like code in JavaScript files

Click Event :

```
function Event() {
  function clicked() {
    alert("Clicked");
  }
  return (
    <>
      <button onClick={clicked}>Click Me</button>
      {/* <button onClick={()=>{alert("click")}}>Click Me</button> */}
      {/* <button onClick={()=>{clicked()}}>Click Me</button> */}
      <h1>{name}</h1>
    </>
```

```
  );
}
export default Event;
```

What is State?

state is a built-in object that allows components to store and manage data that can change over time.

Why we use state?

```
// In vaiable case, component not rerender because we use state or props
```

Using functional based component

```
import React,{useState} from "react";
// In vaiable case, component not rerender because we use state or props
function State(){
    const [cnt,setCnt] = useState(0);
    function click()
    {
        setCnt(cnt+1);
    }
    return(
        <>
            <h2>Hello {cnt}</h2>
            <button onClick={click}>click</button>
        </>
    )
}
export default State;
```

Using class based component

```
import React from "react";
class State1 extends React.Component{
    constructor()
    {
        super();
        this.state={
            cnt:1
        }
    }
    update()
    {
        this.setState({cnt:this.state.cnt+1})
    }
    render(){
    return(
        <>
            <h1>Count :- {this.state.cnt}</h1>
            <button onClick={()=>this.update()}>Update</button>
        </>
    )
}
}
export default State1;
```

What is props?

Props nothing but properties
Props is a special keyword in React that stands for properties and is used for passing data from one component to another.

Functional:

```
import React, { useState } from "react";
function Props(props) {
  const { name, age, other } = props;
  const [name1, setName] = useState(name);
  function update() {
    setName("Ram");
  }
  return (
    <>
      <h1>My name is {name1}</h1>
      <h1>My age is {age}</h1>
      <h1>
        <ul>
          <li>{other.city}</li>
          <li>{other.country}</li>
          <li>{other.postalCode}</li>
          <li>{other.state}</li>
          <li>{other.street}</li>
        </ul>
      </h1>
      <button onClick={update}>Update Name</button>
    </>
  );
}
export default Props;
```

App.jsx :

```
// import Event from './components/Event'
// import User from './components/User'
// import User1 from './components/User1'
// import State from "./components/State"
import Props from "./components/Props";
// import State1 from "./components/State1"
function App() {
  return (
    <>
      {/* <h1>Hello React</h1> */}
      {/* <User></User> */}
      {/* <User1></User1> */}
      {/* <Event></Event> */}
      {/* <State></State> */}
      {/* <State1></State1> */}
      <Props
        name={"RRK"}
        age={22}
        other={{
          street: "123 Main Street",
          city: "Springfield",
          state: "IL",
          postalCode: "62701",
          country: "USA",
        }}
      ></Props>
    </>
  );
}
export default App;
```

Class :

```
import React from "react";
class Props1 extends React.Component {
```

```
  constructor(props) {
    super(props);
    this.state = {
      cnt: props.cnt,
    };
  }
  update() {
    this.setState({ cnt: this.state.cnt + 1 });
  }
  render() {
    return (
      <>
        <p>Count: {this.state.cnt}</p>
        <button onClick={() => this.update()}>Update</button>
      </>
    );
  }
}
export default Props1;
```

App.jsx

```
// import Event from './components/Event'
// import User from './components/User'
// import User1 from './components/User1'
// import State from "./components/State"
// import Props from "./components/Props";
// import State1 from "./components/State1"
import Props1 from "./components/Props1";
function App() {
  return (
    <>
      {/* <h1>Hello React</h1> */}
      {/* <User></User> */}
      {/* <User1></User1> */}
      {/* <Event></Event> */}
      {/* <State></State> */}
      {/* <State1></State1> */}
      {/* <Props
        name={"RRK"}
        age={22}
        other={{
          street: "123 Main Street",
          city: "Springfield",
          state: "IL",
          postalCode: "62701",
          country: "USA",
        }}
      ></Props> */}
      <Props1 cnt={0}></Props1>
    </>
  );
}
export default App;
```