Agenda

- React
- JSX
- React Component

What is React

- It is a JS library used to develop Single Page Application
- React is used to develop client side applications
- React features
  - Has a component-driven architecture
  - Uses virtual DOM for improving the application performance
  - Eco-system: React Router, React Redux Toolkit, React Native
- since it is a library, it requires less memory compared to Angular
- it is bit faster than Angular

History

- Created by: Jordan Walke at Facebook (2011)
- First used: Facebook News Feed (2011)
- Open-sourced: May 2013 (React Conf 2015 marked major adoption)
- Current maintainers: Meta (Facebook) + community (OpenJS Foundation)

React Important Concepts

1. Component Architecture

- Reusable Components

```
// Component = JavaScript function returning JSX
function Greeting() {
    return <h2>Welcome to React!</h2>;
}

// Component Composition
function App() {
    return (
        <div>
            <h1>Hello World</h1>
            <Greeting />
            <Greeting />
        </div>
    );
}
```

2. JSX Styling

- Inline styles as objects

```
<div className="container" style={{ color: "red" }}>
    <p>Styled content</p>
</div>
```

```
const style = {
    container: {
        color: "red"
    },
    content: {
        fontSize: "14px",
        color: "blue"
    }
}

<div className="container" style={style.container}>
    <p style={style.content}>Styled content</p>
</div>
```

- class vs className

```
// Correct
<div className="book-card">...</div>

// Incorrect
<div class="book-card">...</div>
```

3. Embedding JavaScript in JSX/Data binding

- Using the variable value inside a html tag
- In react, it will be done using interpolation
- use the `{}` brackets for loading the variables value inside the html tag

```
const myvar = 100
<h3>{myvar}</h3>
```

- To render an object, split the object into properties and use interpolation to render those properties

```
const jamesBond = {
    name: "James Bond",
    age: 65,
    address: "London",
    speciality: "Adventurous Spy",
};

// Variables in curly braces
<div>Name: {jamesBond.name}</div>
<div>Address: {jamesBond['address']}</div>

// Expressions work too
<div>Status: {age > 60 ? 'Veteran' : 'Active'}</div>
```

- to render an array, use the map function to iterate over the array and use interpolation to render the properties of the object

```
const heros = [
    {
        name: "Iron Man",
        age: 48,
        address: "Stark Tower, New York",
        speciality:
            "Genius-level intellect, Powered armor suit, Advanced technology",
    },
    {
        name: "Captain America",
        age: 105,
        address: "Brooklyn, New York",
        speciality: "Super strength, Enhanced agility, Vibranium shield combat",
    },
];

const MarvelHeros = () => {
    return (
        <div>
            {heros.map((hero) => (
                <div>
                    <h3>Marvel Hero</h3>
                    <div>Name: {hero.name}</div>
                    <div>Address: {hero.address}</div>
                    <div>Age: {hero.age}</div>
                    <div>Speciality: {hero.speciality}</div>
                </div>
            ))}
        </div>
    );
};
export default MarvelHeros;
```

Vite

- a build tool used to build multiple types of project
- e.g. react, angular, svelte, preact etc.

- build
  - compile the project: minifying (reducing the file size) the js files
  - add the external referenced packages
  - create a bundle (collection of required files) for deployment
- create a vite project using npm

project management

```
# create a new project
# > npm create vite <application name>
> npm create vite app1
> yarn create vite app1
> npx vite app1

# go to the project
> cd app1

# install the dependencies
> npm install
> yarn

# start running the application
> npm run dev
> yarn dev

# build the application
> npm run build
> yarn build

# unit test the application
> yarn test
```

Project hierarchy

- node_modules (directory)
  - contains all the dependencies mentioned in package.json
- public (directory)
  - contains the public resources like images, audio, video etc.
- src (directory)
  - contains the application source code (components, assets etc)
  - assets (directory)
    - contains application assets like images
  - components (directory)
    - contains the reusable components
  - pages (directory)
    - contains the components which represent the website pages
  - services (directory)
    - contains the functions which connects the frontend to the backend

        - the function will call the REST APIs
    - App.css
        - contains the css rules for App component
    - App.jsx
        - contains the application's startup or main component
        - the application starts with App component
    - index.css
        - contains the css rules shared across the components
    - main.jsx
        - contains the application startup code
- .gitignore
    - used to mention the files which need not to be added to the git repository
- eslint.config.js
    - configuration file used by ESLint
- index.html
    - only html file in the project
    - contains the application startup code
- package.json
    - contains the application dependencies, scripts and metadata
    - dependencies
        - list of modules required to run the application
        - these modules will be bundled when the app gets built
    - devDependencies
        - list of modules required to develop the application
        - these modules will NOT be bundled when the application gets built
- Readme.md
    - contains the information about the application
- vite.config.js
    - contains the vite configuration
- yarn.lock
    - contains the packages installed with the required versions
    - created when yarn is used to install the dependencies
- package-lock.json
    - contains the packages installed with the required versions
    - created when npm is used to install the dependencies

Project startup

- yarn dev or npm run dev is fired
- starts lite web server on port 5173
- the server loads the file named index.html
- index.html loads src/main.jsx
- main.jsx
    - find an object of div having root id
    - renders the first component named App (which is defined in src/App.jsx)

React Component

- Component is reusable entity which contains user interface defined in html code
- A component can be loaded using the component name as a tag (enclosed by < and >)
- It Contains -

1. user interface: created using JSX syntax
2. state (optional): used to store some data for internal use
3. event handlers (optional): used to handle events generated by the component UI
4. props (optional): parameters (properties) of a component
5. mandatory to return only one element as a return value of a component. If you have multiple elements, then wrap them inside a parent element

- Types of components

1. Functional component

- Component created using a function
- A javascript function which returns a JSX code to create its user interface
- Before react 16, functional components were used only for stateless implementation (the component which does not require to maintain the state)
- stateless means a component which is not maintaining its state
- Since the react 16 introduced a concept called as a react hooks, it is possible now to create functional components to store the state
- Hence the class components are not need anymore and by default we use a function to create component
- it is faster than class component
- it is light weight component

```
const root = document.getElementById('root')
// functional component
function Dummy() {
    return <div>this is a functional component </div>
}
// here the <Dummy /> is function component
ReactDOM.createRoot(root).render(<Dummy />)
```