REACT JS NOTES:

Introduction to React

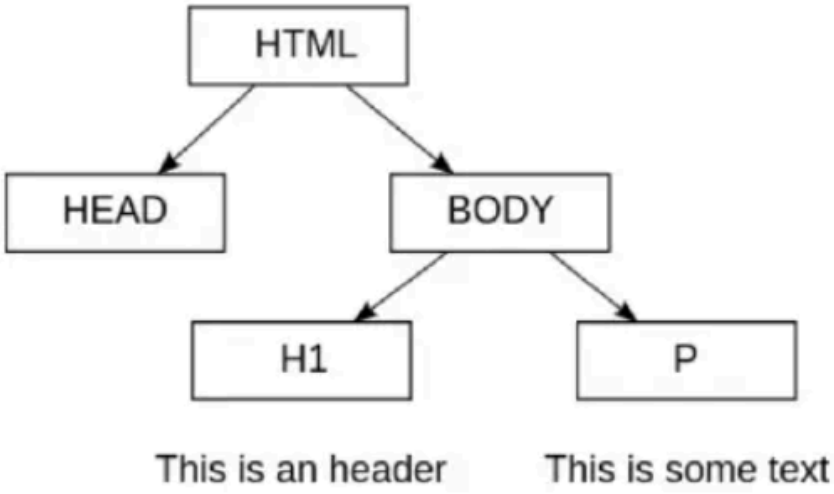
React is a JavaScript library created by Facebook for building UI. It is the most popular library for developing Single Page Applications (SPA) and is used by companies like Netflix and Instagram. React enables the creation of modern, fast, and dynamic web applications.

A library is a collection of reusable code that simplifies development by allowing developers to import specific features into their projects. For example, jQuery helps write JavaScript more efficiently without making the project dependent on it.

A framework is a complete package that includes built-in functionalities and libraries, enforcing a structured way of development. Unlike libraries, frameworks come with predefined rules, limiting flexibility. The project becomes dependent on the framework, such as Angular.

Virtual DOM

1. React uses a virtual DOM to optimize updates and improve performance. The virtual DOM is an in-memory representation of the actual DOM elements. It's a lightweight copy of the real DOM.
2. When you make changes to the state of a React component, React creates a new virtual DOM tree representing the updated state.
3. React then compares the new virtual DOM with the previous virtual DOM to determine the differences (diffing).
4. The differences are used to compute the most efficient way to update the real DOM.



Tree Structure of the Document Object Model

React.js - JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used in React to write UI components more easily. It allows mixing HTML-like code with JavaScript.

Why use JSX?

Makes code more readable and easier to write.

Babel compiles JSX into JavaScript for browser compatibility.

Helps in creating React elements efficiently.

JSX Syntax Rules:

Elements must have one parent element (use <> or <> fragments).

Use curly braces {} to embed JavaScript expressions.

Class attribute is written as className in JSX.

Self-closing tags (like) must end with /S

```
class Test extends React.Component {
  render() {
    return (
      <div>
        <p>Hello</p>
        <p>World</p>
      </div>
    );
  }
}
```

React Installation

1. Install React using Create React App

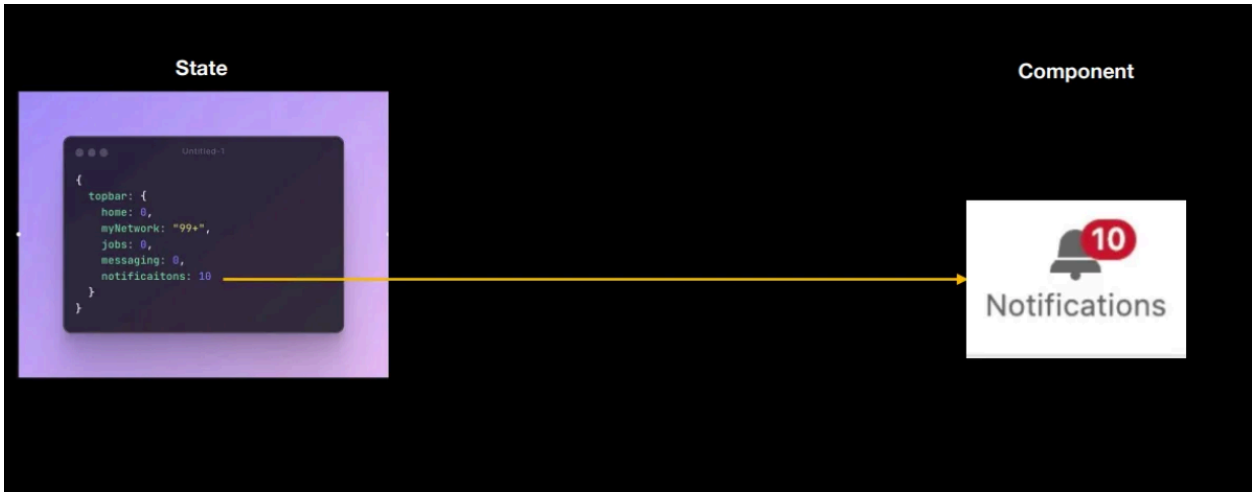
2. Navigate and start the project:

```
npx create-react-app my-app

cd my-app
```

React Components

A React component is like a LEGO brick—a reusable building block for creating UI elements. Each component has its own functionality and can be combined to build websites or applications. Components make development modular, reusable, and efficient.



Functional (Stateless) Components:

• A functional component is basically a JavaScript (or ES6) function which returns a React element. According to React official docs, the function below is a valid React component:

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

Class (Stateful) Components

• Class components are ES6 classes. They are more complex than functional components including constructors, life-cycle methods, render() function and state (data) management.

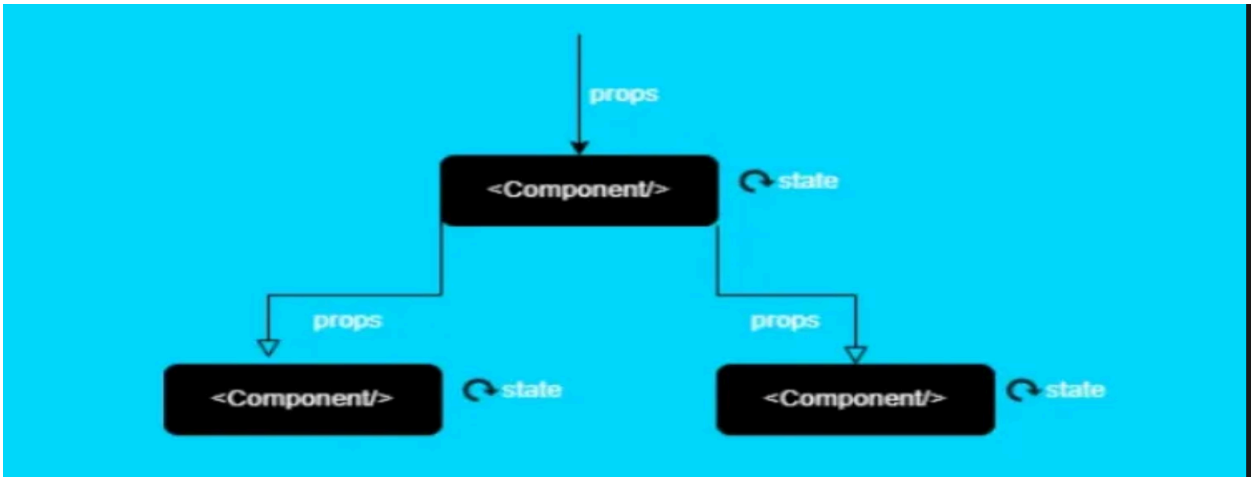
```jsx
import React, { Component } from 'react';

class ExampleComponent extends Component {
  render() {
    return (
      <div>This is an example component.</div>
    );
  }
}

export default ExampleComponent;
```

What are Props in React?

Props (Properties) in React are used to pass data between components in a uni-directional flow (from parent to child). Since React follows a component-based architecture, props enable communication between reusable components. Props are read-only, meaning child components cannot modify the data received from the parent.



```
class ParentComponent extends Component {
  render() {
    return (
      <h1>
        I'm the parent component.
        <ChildComponent />
      </h1>
    );
  }
}
```

```
const ChildComponent = () => {
  return <p>I'm the 1st child!</p>;
};
```

The problem here is that, when we call the ChildComponent multiple times:

```
class ParentComponent extends Component {
  render() {
    return (
      <h1>
        I'm the parent component.
        <ChildComponent />
        <ChildComponent />
        <ChildComponent />
      </h1>
    );
  }
}
```

It always renders the same string again and again:


```
class ParentComponent extends Component {
  render() {
    return (
      <h1>
        I'm the parent component.
        <ChildComponent text={"I'm the 1st child"} />
        <ChildComponent text={"I'm the 2nd child"} />
        <ChildComponent text={"I'm the 3rd child"} />
      </h1>
    );
  }
}
```

Recap

- Props stand for properties and is a special keyword in React
- Props are being passed to components like function arguments
- Props can only be passed to components in one-way (parent to child)
- Props data is immutable (read-only)

What is State in React?

State is a special object that holds dynamic data, allowing components to react to changes based on user actions or events. It is private to the component where it is defined but can be passed to child components via props. State is initialized in the constructor and updated using setState() instead of modifying it directly. Overusing state can impact performance, so it should be managed efficiently.

creating the state

• A class has a special method called constructor) and it is being called during object creation. We can also initialize our object properties.

• The same rule applies to state. Since state is also an object, it should be initialized inside the constructor method.

```
constructor() {
  this.state = {
    id: 1,
    name: "test"
  };
}
```

Updating the State in React

State should not be modified directly as per React's official guidelines. A component's state changes in response to user interactions (e.g., clicking a button, scrolling) or server responses. When the state updates, React automatically re-renders the UI. Always use setState() to update state instead of modifying it directly.

```
this.state.name = "testing state";  // wrong
```

Using setState( )

- Below you can see the right way of state changes in React:

```
this.setState({
  name: "testing state"
});
```

Array.map() Method

The map() method is widely used in JavaScript to iterate over an array and return a new array with modified elements. It calls a function on each element in order but does not execute for empty values. The original array remains unchanged as map() creates a new array.

Parameter	Description
function(currentValue, index, arr)	Required. A function to be run for each element in the array. Function arguments:

Argument	Description
currentValue	Required. The value of the current element
index	Optional. The array index of the current element
arr	Optional. The array object the current element belongs to

Array.filter() Method

The filter() method creates a new array containing elements that pass a test defined by a function. It does not execute for empty values and does not modify the original array.

Syntax

```
array.filter(function(currentValue, index, arr))
```

React Lifecycle Methods

React components go through three lifecycle phases:

1. Mount Phase – The component is created and added to the DOM.
Key methods: constructor(), render(), componentDidMount() (called in order).
2. Update Phase – The component re-renders when state or props change.
3. Unmount Phase – The component is removed from the DOM.

React Router

React Router is a routing library for React used to manage navigation in React apps.

Installation:

Run the following command:

```
npm install react-router-dom
```

Key Components:

1. **BrowserRouter** – Wraps the app to enable routing.
2. **Route** – Defines a **path** and the **component** to render.
3. **Link** – Provides navigation without reloading the page.

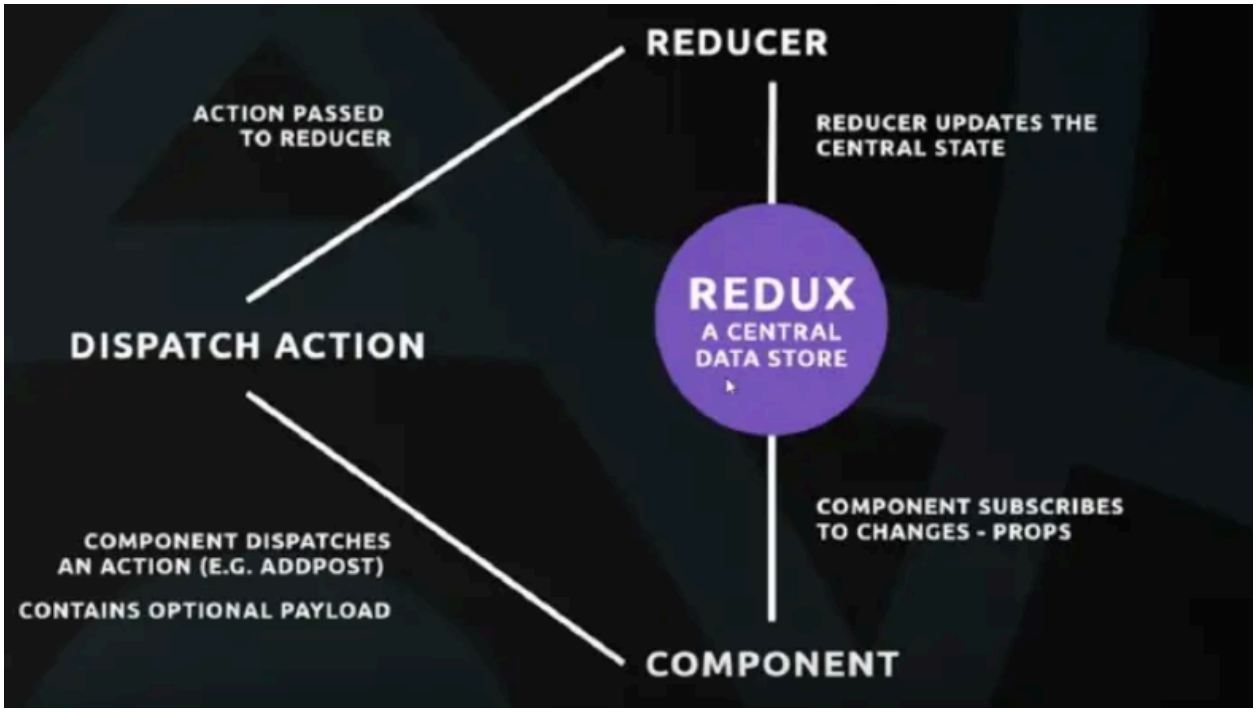
Redux in React

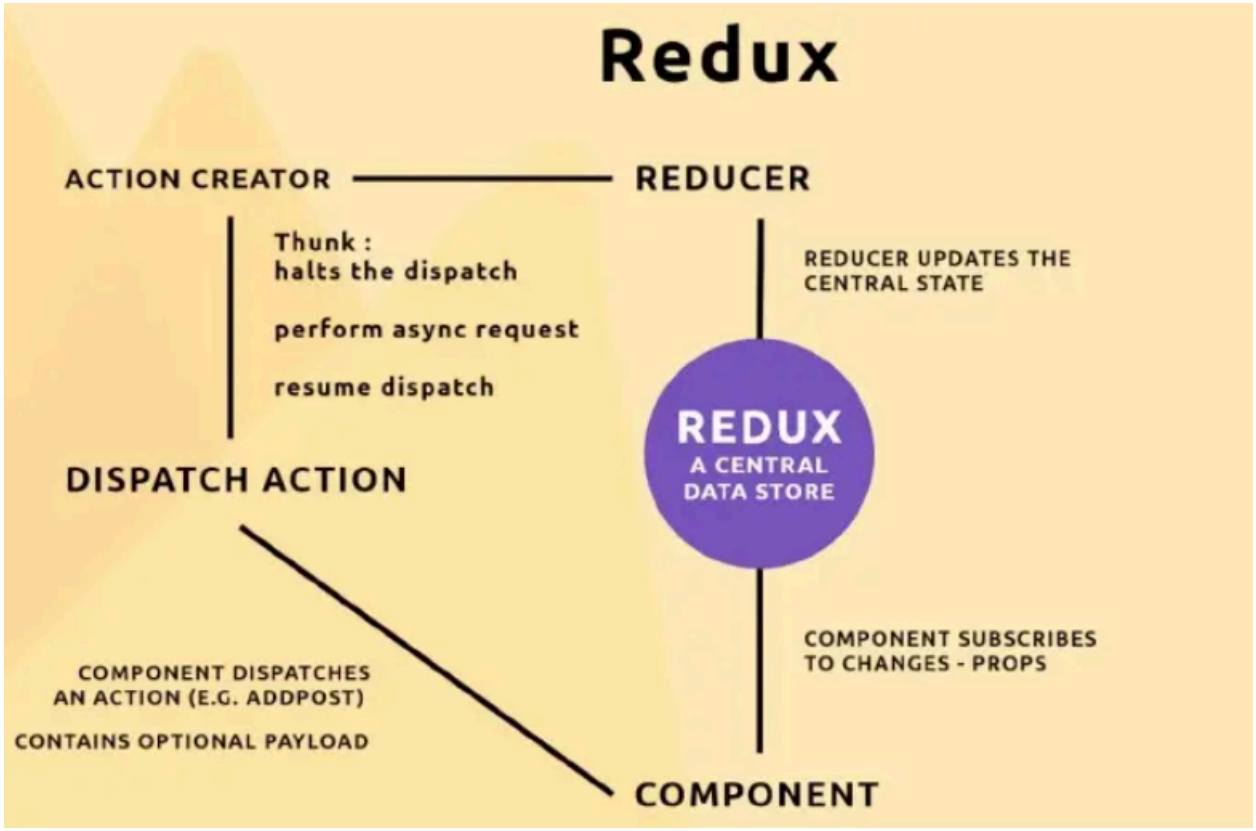
Redux is a state management library for JavaScript applications, commonly used with React to manage global state efficiently.

Why Use Redux?

1. Centralized state management.

2. Predictable **state updates


REDUCER
ACTION PASSED TO REDUCER
REDUCER UPDATES THE CENTRAL STATE
REDUX
A CENTRAL DATA STORE
DISPATCH ACTION
COMPONENT SUBSCRIBES TO CHANGES - PROPS
COMPONENT DISPATCHES AN ACTION (E.G. ADDPOST)
CONTAINS OPTIONAL PAYLOAD
COMPONENT


Redux
ACTION CREATOR
REDUCER
Thunk :
halts the dispatch
perform async request
resume dispatch
REDUCER UPDATES THE CENTRAL STATE
REDUX
A CENTRAL DATA STORE
DISPATCH ACTION
COMPONENT SUBSCRIBES TO CHANGES - PROPS
COMPONENT DISPATCHES AN ACTION (E.G. ADDPOST)
CONTAINS OPTIONAL PAYLOAD
COMPONENT