4. React 18

4.1 Introduction

- Open Source - JavaScript Library → Used for Building UI
- React integrates well with other libraries
  - Useful in building enterprise-scale applications
- Component-based Architecture
- Declarative
  - Just tell React - what we want to achieve
  - React with DOM Library will build the actual UI
- To set up a new React app

```
npx create-react-app appNameHere
```

4.2 Folder Structure

- Package JSON
  - Dependencies and Scripts

    - Scripts → To run, build, and test the application
    - ES Lint → To highlight any error
- Yarn Lock or Package Lock JSON
    - Ensures consistent installation of dependencies
- Node Modules
    - All dependencies are installed here
    - Created on Create-React-App or when *npm* install is executed
- Progressive Web Apps
    - Fav Icon ICO - Logo PNG - Manifest JSON
    - Robots TXT → Required for Search Engine and is not React specific
- Index HTML
    - Only one HTML file
    - This will be served in the browser
- Source Folder (SRC)
    - Index JS
        - Root Component → App
        - DOM Element → Element with ID Root → This is the sole element in Index HTML file
    - App JS
        - Simple Functional Component
        - Represents the view in the browser
    - App CSS
        - For Styling
    - App Test JS
        - For Unit Tests - Simple Test Case
    - Index CSS
        - Applies style to the body tag
        - Indexed in Index CSS

- Logo SVG
  - Referenced in App JS → React Logo that swivels
- Report Web Vitals
  - Performance and Analytics
- Setup Tests
  - Associated with Tests

4.3 Component-based Structure

- Root (App)
  - Header
  - Side Nav
  - Main Content
  - Footer
- **Functional Components**
  - Simple JavaScript Functions
  - Accepts Properties and Returns HTML (JSX)

4.4 Export Types

- Default Export
  - We can change the name to anything.

```
import React from 'react'
const Greet = () => {
  <div>
      Greet
  </div>
}

export default Greet;
```

- Named Export
  - We need to use the specific name

- Use angle brackets while importing

```
import React from 'react'
export const Greet = () => {
  <div>
      Greet
  </div>
}

--

import {Greet} from 'Greet.jsx'
```

4.5 JSX - JavaScript Extension

- Describes how the UI should look like
- Differences
  - Class → class Name
  - for → html For
  - Camel Case naming convention

4.6 Props (Properties)

- Parameters that can be utilized by components to dynamically render data
- Props are immutable
  - We can not modify the value → Under any circumstance
  - Props is read-only

```
# To define Props in the Root Component
# Simply pass in the values

function App() {
  return (
    <div className="App">
      <Greet myName="Manpreet" />
      <Greet myName="Gullu" />
      <Greet myName="Nikhil" />
    </div>
  );
}

# In the Component
```

```
import React from "react";

const Greet = ({ myName }) => {
  return <h1>Hello {myName}!</h1>;
};

export default Greet;
```

- To pass-in unknown content → Content in-between the tags
  - We can use the reserved keyword → children

```
# In App JS

import "./App.css";
import Greet from "./Components/Greet";

function App() {
  return (
    <div className="App">
      <Greet myName="Manpreet" />
      <Greet myName="Gullu" />
      <Greet myName="Nikhil">
        <p> Sample Children Content </p>
      </Greet>
    </div>
  );
}

export default App;

--------------------------------------------------------------------------------

# in Greet Component

import React from "react";

const Greet = (props) => {
  return (
    <div>
      <h1>I am {props.myName}</h1>
      <h2>{props.children}</h2>
    </div>
  );
};

export default Greet;
```

4.7 State

Props	State
Get passed to the component	Managed within the component
Simple function parameters	Variables in function body
Immutable	Mutable
Props	use State

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

const Message = () => {
  const [message, setMessage] = useState("Welcome Visitor!");
  return (
    <div>
      <h1>{message}</h1>
      <button onClick={() => setMessage("Thanks for Sub!")}>Sub!</button>
    </div>
  );
};

export default Message;
```

4.8 Hooks

- Special feature that allows to *Hook Into* React Features
- Examples
  - use State
  - use Reducer
  - use Effect
  - use Context
  - use Ref
  - use Memo
  - use Callback
  - use Transition
  - use Deferred Value

4.9 Event Handling

- Whenever a user interacts → Events are fired
- Do not add ( ) on click Handler
  - This will call the function when the component is loaded

```
import React from "react";

const ClickHandler = () => {
  const clickHandler = () => {
    console.log("Button Clicked");
  };

  return (
    <div>
      <button onClick={clickHandler}>Click</button>
    </div>
  );
};

export default ClickHandler;
```

4.10 Parent Child Communication

- Pass function as prop from Parent to Child
- Here greetHandler is the prop which contains a reference to the function → greet Parent
- This prop is accessed in Child JSX and set to on Click

```
# Parent JSX

import React from "react";
import Child from "./Child";

const Parent = () => {
  const greetParent = () => {
    alert("Hello Parent!");
  };
  return <Child greetHandler={greetParent} />;
};

export default Parent;

# Child JSX

import React from "react";

const Child = (props) => {
```

```
  return (
    <div>
      <button onClick={props.greetHandler}>Greet Parent</button>
    </div>
  );
};

export default Child;
```

- To pass in any parameters, we would have to make use of Arrow functions in Child Component.

```
# Child JSX
import React from "react";

const Child = (props) => {
  return (
    <div>
      <button onClick={() => props.greetHandler("Child")}>Greet Parent</button>
    </div>
  );
};

export default Child;

# Parent JSX

import React from "react";
import Child from "./Child";

const Parent = () => {
  const greetParent = (name) => {
    alert(`Hello Parent from ${name}`);
  };
  return <Child greetHandler={greetParent} />;
};

export default Parent;
```

- Not much of change but we can simply use arrow functions → pass in the required parameters and access them in the Parent component.

4.11 Conditional Rendering

- We may need to show or hide content based on specific conditions
- Conditional statement works the same way as they do in JavaScript

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

const UserGreeting = () => {
  const [isLoggedIn, loggedIn] = useState(false);
  return (
    <div>
      <h1>Hi{isLoggedIn ? " Nik" : " User"}</h1>
      <button onClick={() => loggedIn(true)}>
        Log {isLoggedIn ? " Out" : " In"}
      </button>
    </div>
  );
};

export default UserGreeting;
```

- Here, we have used use State to check whether or not the user is logged in
- Based on the log in status
  - We can use conditional rendering to show or hide specific content
- **Short Circuit Operator**
  - We can use this when we have to either show *something* or *nothing.*

```
<h1>Hi{isLoggedIn && " Nik"}</h1>
```

- Here, we will evaluate the expression
  - If it is true → ‘Nik’ will be displayed
  - Else → nothing

4.12 List Rendering

- To render an array of values
  - We can use *map* → to traverse through the list of Arrays and display them in an ordered manner.

```
import React from "react";

const NameList = () => {
  const names = ["Alpha", "Bravo", "Charlie", "Delta", "Echo"];
  return (
    <div>
      {names.map((name) => (
```

```
        <h2>{name}</h2>
      ))}
    </div>
  );
};

export default NameList;
```

- However, we need unique key for each *name* while rendering the list.
- Helps in resolving bugs that might occur due to filtering and sorting.

4.13 Styling Basics

- **CSS Stylesheets**
  - Create a Component | JSX
  - Create a Stylesheet | CSS
  - Import CSS in JSX
  - use class Name = ‘class Name Here’
  - We have to make use of Class Name keyword.

```
# JSX
import React from "react";
import "./myStyles.css";

const Stylesheet = () => {
  return <h1 className="primary">Stylesheet</h1>;
};

export default Stylesheet;

# CSS
.primary {
  color: orange;
}
```

- **Inline Styling**
  - Specify CSS properties as an Object of Key-Value Pairs
  - Key → CSS Property : Camel Case
  - Value → CSS Value : String
  - Use *style* keyword for JSX Element

```
import React from "react";

const InLine = () => {
  const heading = {
    fontSize: "72px",
    color: "orange",
  };
  return <h1 style={heading}>InLine</h1>;
};

export default InLine;
```

- CSS Modules
  - Classes are locally scoped by default
  - Only the JSX file that imported CSS file will be able to use particular style
  - We have to use *module* keyword in the name

```
# CSS - app.module.css

# JSX

import styles from "./Components/appStyles.module.css";
<h1 className={styles.success}>Success</h1>
```

- CSS in JS Libraries

4.14 Forms

- Define Form Tag
- Define a state to store the initial value and update value - use State
- Define on Change function on Input field → Capture event and use it to set value

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

const Form = () => {
  const [initialValue, setValue] = useState("");
  const handleInput = (e) => {
    setValue(e.target.value);
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log(`Username is ${initialValue}`);
  };
```

```
  return (
    <div>
      <form onSubmit={handleSubmit}>
        <label htmlFor="username">Username: </label>
        <input
          type="text"
          name="username"
          value={initialValue}
          onChange={handleInput}
        />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
};

export default Form;
```

- Text Area and Select Option

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

const Form = () => {
  const [initialValue, setValue] = useState("");
  const [initialCountry, setCountry] = useState("");
  const [initialTextArea, setTextArea] = useState("");

  const handleInput = (e) => {
    setValue(e.target.value);
  };

  const handleTextArea = (e) => {
    setTextArea(e.target.value);
  };

  const handleCountry = (e) => {
    setCountry(e.target.value);
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log(
      `Username is ${initialValue} : ${initialTextArea} : ${initialCountry}`
    );
  };
  return (
    <div>
      <form onSubmit={handleSubmit}>
        <label htmlFor="username">Username: </label>
        <input
          type="text"
          name="username"
          value={initialValue}
          onChange={handleInput}
        />
        <br />
```

```
        <label htmlFor="description">Comments: </label>
        <textarea
          name="description"
          id=""
          cols="10"
          rows="10"
          value={initialTextArea}
          onChange={handleTextArea}
        ></textarea>
        <select name="country" id="" onChange={handleCountry}>
          <option value="none" selected disabled hidden>
            Select an Option
          </option>
          <option value="India">India</option>
          <option value="Canada">Canada</option>
        </select>
        <button type="submit">Submit</button>
      </form>
    </div>
  );
};

export default Form;
```

4.15 HTTP GET and POST

GET Request

- We can make use of use Effect and use State hook
- Simply, fetch the data
  - Convert to JSON
  - Set the data in the array

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

const PostList = () => {
  const [posts, setPosts] = useState([]);
  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts")
      .then((response) => response.json())
      .then((data) => setPosts(data))
      .catch((err) => console.log(err));
  }, []);

  return (
    <div>
      <ul>
        <h2>
          {posts.map((post) => (
            <li id={post.id}>{post.title}</li>
```

```
          ))}
        </h2>
      </ul>
    </div>
  );
};

export default PostList;
```

POST Request

- Create a Form Tag and necessary input elements
- For each element → use State
- use Fetch command with POST method
- Include necessary headers

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

const PostForm = () => {
  const [uid, setUid] = useState("");
  const [initialTitle, setTitle] = useState("");
  const [initialBody, setBody] = useState("");

  const submitHandler = (e) => {
    e.preventDefault();
    fetch("https://jsonplaceholder.typicode.com/posts", {
      method: "POST",
      body: JSON.stringify({
        title: initialTitle,
        body: initialBody,
        userId: uid,
      }),
      headers: {
        "Content-type": "application/json; charset=UTF-8",
      },
    })
      .then((response) => response.json())
      .then((json) => console.log(json));
  };

  return (
    <form onSubmit={submitHandler}>
      <div>
        <input
          type="text"
          name="userid"
          value={uid}
          onChange={(e) => setUid(e.target.value)}
          placeholder="User ID"
        />
      </div>
```

```
      <div>
        <input
          type="text"
          name="title"
          value={initialTitle}
          onChange={(e) => setTitle(e.target.value)}
          placeholder="Title"
        />
      </div>
      <div>
        <input
          type="text"
          name="body"
          value={initialBody}
          onChange={(e) => setBody(e.target.value)}
          placeholder="Body"
        />
      </div>
      <button type="submit">Submit</button>
    </form>
  );
};

export default PostForm;
```

4.16 use Transition

- Generating a simple Search List using JSON

```
import React, { useState } from "react";
import NAMES from "../Components/data.json";

const Transition = () => {
  const [query, setQuery] = useState("");
  const handleChange = (event) => {
    event.preventDefault();
    setQuery(event.target.value);
  };

  const filteredNames = NAMES.filter(
    (NAME) => NAME.firstname.includes(query) || NAME.lastname.includes(query)
  );

  return (
    <div>
      <input type="text" onChange={handleChange} value={query} />
      {filteredNames.map((NAME) => (
        <h1 key={NAME.id}>
          {NAME.firstname} {NAME.lastname}
        </h1>
      ))}
    </div>
  );
```

```
};

export default Transition;
```

- With use Transition

```
import React, { useState, useTransition } from "react";
import NAMES from "../Components/data.json";

const Transition = () => {
  const [query, setQuery] = useState("");
  const [inputVal, setInputVal] = useState("");
  const [isPending, startTransition] = useTransition();
  const handleChange = (event) => {
    event.preventDefault();
    setInputVal(event.target.value);
    startTransition(() => setQuery(event.target.value));
  };

  const filteredNames = NAMES.filter((NAME) => {
    return NAME.firstname.includes(query) || NAME.lastname.includes(query);
  });

  return (
    <div>
      <input type="text" onChange={handleChange} value={inputVal} />
      {isPending && <h2>Updating Lists</h2>}
      {filteredNames.map((NAME) => (
        <div>
          <h1 key={NAME.id}>
            {NAME.firstname} {NAME.lastname}
          </h1>
        </div>
      ))}
    </div>
  );
};

export default Transition;
```