**Exercise 1: Basic Component with Inline CSS**

Create a simple functional component that displays a greeting message with inline styling.

**Task**

* Create a **Greeting** component that takes a **name** prop and displays a message.
* Apply inline styling to change the text color and font size.

**Solution:**

Here's how you can implement the **Greeting** component using an external CSS file for styling.

**File Structure:**

src/

├── Greeting.js

├── Greeting.css

├── index.js

**Greeting.js:**

import React from "react";

import "./Greeting.css"; // Import the external CSS file

const Greeting = ({ name }) => {

return (

<div className="greeting">

Hello, {name}! Welcome to our React app.

</div>

);

};

export default Greeting;

**Greeting.css:**

.greeting {

color: blue;

font-size: 24px;

font-weight: bold;

margin: 10px;

}

**index.js:**

import React from "react";

import ReactDOM from "react-dom";

import Greeting from "./Greeting";

const App = () => {

return (

<div>

<Greeting name="John" />

</div>

);

};

**Output:**

When you run this code, the greeting message (Hello, John! Welcome to our React app.) will appear with the styles applied from the Greeting.css file.

**Exercise 2: Component with External CSS File**

Create a component with a separate CSS file for styling.

**Task**

* Create a **Card** component that displays a title and a description.
* Use an external CSS file to style the card with a border, padding, and margin.

**Solution:**

Here's how to create a **Card** component with an external CSS file for styling:

**File Structure:**

src/

├── Card.js

├── Card.css

├── App.js

**Card.js:**

import React from "react";

import "./Card.css"; // Import the external CSS file

const Card = ({ title, description }) => {

return (

<div className="card">

<h2 className="card-title">{title}</h2>

<p className="card-description">{description}</p>

</div>

);

};

export default Card;

**Card.css:**

.card {

border: 1px solid #ccc;

border-radius: 8px;

padding: 16px;

margin: 16px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

background-color: #f9f9f9;

}

.card-title {

font-size: 20px;

font-weight: bold;

color: #333;

margin-bottom: 8px;

}

.card-description {

font-size: 16px;

color: #555;

}

**App.js:**

import React from "react";

import ReactDOM from "react-dom";

import Card from "./Card";

const App = () => {

return (

<div>

<Card

title="Card Title"

description="This is a description of the card component. It is styled using an external CSS file."

/>

</div>

);

};

**Output:**

When you run this code, the **Card** component will be displayed with:

* A light border.
* Rounded corners.
* Padding and margin for spacing.
* A slight shadow for better visual appeal.

This makes use of a clean external CSS file for separation of concerns.

**Exercise 3: Component with Dynamic CSS Classes**

**Create a component that applies different CSS classes based on props.**

**Task**

* Create a **Button** component that takes a **type** prop (**primary** or **secondary**).
* Apply different CSS classes based on the **type** prop to change the button's appearance.

**Solution:**

Here's how to create a **Button** component that applies different CSS classes dynamically based on the type prop:

**File Structure:**

src/

├── Button.js

├── Button.css

├── App.js

**Button.js:**

import React from "react";

import "./Button.css"; // Import the external CSS file

const Button = ({ type, label }) => {

// Dynamic class assignment based on the `type` prop

const buttonClass = type === "primary" ? "button-primary" : "button-secondary";

return <button className={`button ${buttonClass}`}>{label}</button>;

};

export default Button;

**Button.css:**

/\* Common styles for all buttons \*/

.button {

padding: 10px 20px;

font-size: 16px;

border-radius: 5px;

border: none;

cursor: pointer;

transition: background-color 0.3s ease;

}

/\* Styles for primary button \*/

.button-primary {

background-color: #007bff;

color: white;

}

.button-primary:hover {

background-color: #0056b3;

}

/\* Styles for secondary button \*/

.button-secondary {

background-color: #6c757d;

color: white;

}

.button-secondary:hover {

background-color: #495057;

}

**App.js:**

import React from "react";

import ReactDOM from "react-dom";

import Button from "./Button";

const App = () => {

return (

<div style={{ display: "flex", gap: "10px", padding: "20px" }}>

<Button type="primary" label="Primary Button" />

<Button type="secondary" label="Secondary Button" />

</div>

);

};

**Explanation:**

1. **Dynamic Class**:
   * The type prop determines which CSS class (button-primary or button-secondary) is applied in addition to the common button class.
2. **Styling**:
   * The Button.css file contains separate styles for the primary and secondary button types, including hover effects.
3. **Flexibility**:
   * You can easily add more button types (e.g., success, danger) by extending the CSS and modifying the logic in Button.js.

**Output:**

* **Primary Button**: Blue background with white text.
* **Secondary Button**: Gray background with white text.
* Both buttons include hover effects to darken the background color.

**Exercise 4: Component with Conditional Styling**

Create a component that conditionally applies inline styles based on a prop.

**Task**

* Create a **Status** component that displays a status message.
* Use a **status** prop to change the text color based on the status (**success** or **error**).

**Solution:**

Here’s how you can create a **Status** component with conditional inline styling based on the status prop:

**File Structure:**

src/

├── Status.js

├── App.js

**Status.js:**

import React from "react";

const Status = ({ status, message }) => {

// Conditional inline styles based on the status prop

const statusStyle = {

color: status === "success" ? "green" : status === "error" ? "red" : "black",

fontSize: "18px",

fontWeight: "bold",

margin: "10px",

};

return <div style={statusStyle}>{message}</div>;

};

export default Status;

**App.js:**

import React from "react";

import ReactDOM from "react-dom";

import Status from "./Status";

const App = () => {

return (

<div>

<Status status="success" message="Operation completed successfully!" />

<Status status="error" message="An error occurred. Please try again." />

<Status message="This is a neutral status message." />

</div>

);

};

**Explanation:**

1. **Dynamic Styling**:
   * The status prop determines the color of the text:
     + success: Green.
     + error: Red.
     + Default: Black (if no status is provided).
2. **Inline Styling**:
   * The styles are applied directly via the style attribute.
3. **Flexibility**:
   * The message prop allows you to customize the status message.

**Output:**

1. A green message: **"Operation completed successfully!"**
2. A red message: **"An error occurred. Please try again."**
3. A black message: **"This is a neutral status message."**

This approach provides a simple way to apply conditional inline styles in React based on props.

**Exercise 5: Component with CSS Modules**

Create a component that uses CSS modules for scoped styling.

**Task**

* Create a **Profile** component that displays a name and an image.
* Use CSS modules to style the component with scoped class names to avoid style conflicts.

**Solution:**

**File Structure:**

├── App.js

├── Profile.js

├── Profile.module.css

├── index.js

**Profile.js:**

import React from "react";

import styles from "./Profile.module.css"; // Import the CSS module

const Profile = ({ name, image }) => {

return (

<div className={styles.profile}>

<img src={image} alt={name} className={styles.profileImage} />

<h2 className={styles.profileName}>{name}</h2>

</div>

);

};

export default Profile;

**Profile.module.css:**

.profile {

border: 2px solid #ddd;

border-radius: 10px;

padding: 20px;

width: 250px;

text-align: center;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

background-color: #f9f9f9;

margin: 10px;

}

.profileImage {

border-radius: 50%;

width: 100px;

height: 100px;

object-fit: cover;

margin-bottom: 10px;

}

.profileName {

font-size: 20px;

color: #333;

font-weight: bold;

}

**App.js:**

import React from "react";

import Profile from "./Profile";

const App = () => {

return (

<div style={{ display: "flex", justifyContent: "center", padding: "20px" }}>

<Profile

name="John Doe"

image="https://via.placeholder.com/100"

/>

<Profile

name="Jane Smith"

image="https://via.placeholder.com/100"

/>

</div>

);

};

export default App;

**index.js:**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

ReactDOM.render(<App />, document.getElementById("root"));

**Explanation:**

1. **CSS Modules**:
   * Scoped styles ensure that class names are unique, preventing style conflicts.
   * The imported styles object contains dynamically generated class names based on the file and class.
2. **Profile Component**:
   * Displays a profile picture and name.
   * Styled with CSS Modules for isolated styling.
3. **Integration in App.js**:
   * The Profile component is included in App.js, allowing for multiple instances with different data.

**Output:**

* Two **Profile** components displayed side by side:
  + Each with a circular image, bold name, and clean card styling.
* The styling is scoped to the Profile component, preventing conflicts with global styles.