Day 3-4

React Forms & Tailwind CSS

Feedback Form with table

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

function Feedback() {
    const [formData, setFormData] = useState({
        name: '',
        age: '',
        roll: '',
        class: '',
        feedback: ''
      });
    const [tableData, setTableData] = useState([]);

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({
          ...formData, // Spread the existing formData object
          [name]: value // Update the specific property with the
new value
        });
      };

    const handleSubmit = (e) => {
        e.preventDefault();
        setTableData([...tableData, formData]);
        setFormData({
          name: '',
          age: '',
          roll: '',
          class: '',
          feedback: ''
        });
      };

      return (
        <div className="App">
          <h1>Student Feedback Form</h1>
            <div>
              <label>Name:</label>
              <input type="text"
               name="name"
               value={formData.name}
               onChange={handleChange} required />
            </div>
            <div>
              <label>Age:</label>
              <input type="number" name="age"
               value={formData.age}
               onChange={handleChange} required />
            </div>
            <div>
              <label>Roll:</label>
              <input type="text" name="roll" value={formData.roll}
onChange={handleChange} required />
            </div>
            <div>
              <label>Class:</label>
              <input type="text" name="class"
value={formData.class} onChange={handleChange} required />
            </div>
            <div>
              <label>Feedback:</label>
              <textarea name="feedback" value={formData.feedback}
onChange={handleChange} required></textarea>
            </div>
            <button type="submit" onClick={handleSubmit}>Submit</
button>

          <h2>Submitted Feedback</h2>
          <table>
            <thead>
              <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Roll</th>
                <th>Class</th>
                <th>Feedback</th>
              </tr>
            </thead>
            <tbody>
              {tableData.map((data, index) => (
                <tr key={index}>
                  <td>{data.name}</td>
                  <td>{data.age}</td>
                  <td>{data.roll}</td>
                  <td>{data.class}</td>
                  <td>{data.feedback}</td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      );
    }

export default Feedback;
```

Todo List Component

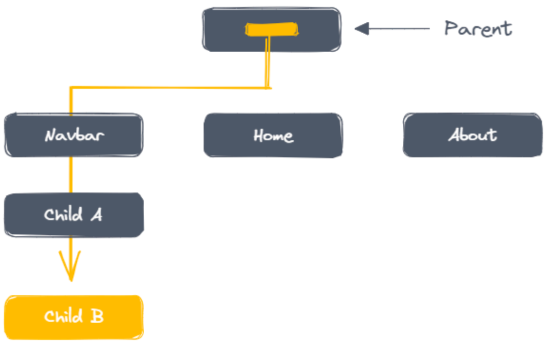
```
import React, { useState } from 'react';

function Feedback() {
   const [formData,SetFormData]=useState({
    time:'',
    note:'',
   })
   const [Table,setTable]=useState([])
   function handleChange(e){
    const {name,value}= e.target;
    SetFormData({...formData,[name]:value});
   }
   function handleSubmit(){
    setTable([...Table,formData]);
    SetFormData({
        time:'',
        note:'',
       })
   }
   function handleDelete(index) {
    const newData = Table.filter(
        (data,i) => i !== index);
    setTable(newData);
  }
   return(
    <div>
        <h3>My Todo List</h3>
            <input type="time"
            name="time"
            value={formData.time}
            onChange={handleChange}
            />
            <input type="text"
            name="note"
            value={formData.note}
            onChange={handleChange}
            />
            <button onClick={handleSubmit}>Submit</button>

        <hr></hr>
        <table>
            <thead>
                <tr>
                <th>Time</th>
                <th>Note</th>
                <th>Action</th>
                </tr>
            </thead>
            <tbody>
                {Table.map((data,index)=>(
                <tr>
                    <td>{data.time}</td>
                    <td>{data.note}</td>
                    <button onClick={()=>handleDelete(index)}
>Delete</button>
                </tr>
                ))}
            </tbody>
        </table>
    </div>
   )
}

export default Feedback;
```

Context API

In React, passing props is a fundamental concept that enables a parent component to share data with its child components as well as other components within an application.

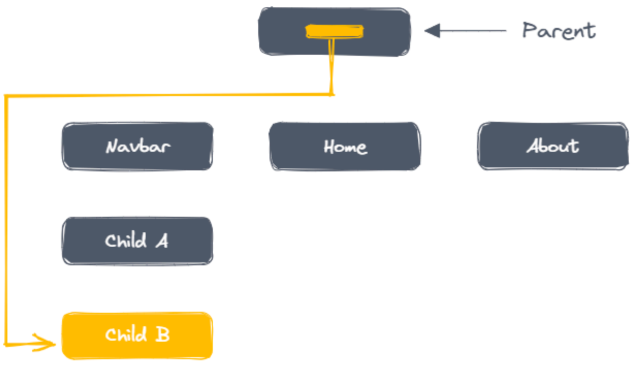
In many cases, passing props can be an effective way to share data between different parts of your application. But passing props down a chain of multiple components to reach a specific component can make your code overly cumbersome.

component tree and make it available to all other components that need it without passing props.



How the Context API Works

Context API allows data to be passed through a component tree without having to pass props manually at every level. This makes it easier to share data between components.



How to Get Started with the Context API

To start using the Context API in your applications, you'll need to follow a few simple steps:

1. Create a Context Object

First, you need to create a context object using the `createContext` function from the 'react' library. This context object will hold the data that you want to share across your application.

Create a new file named `MyContext.js` in the `src` folder and add the following code to create a context object:

```
import { createContext } from 'react';

export const MyContext = createContext("");
```

In the above code, we're importing `createContext` from React and using it to create a new context object named "MyContext". Then, we are exporting the context object so that we can use it in other parts of our application.

2. Wrap Components with a Provider

Once you've created a context object, you need to wrap the components that need access to the shared data with a Provider component. The Provider component accepts a "value" prop that holds the shared data, and any component that is a child of the Provider component can access that shared data.

It's important to note that the Provider component should be wrapped around the top-level component in an application to ensure that all child components have access to the shared data.

Here's an example that demonstrates how to wrap components with a Provider in Context API:

```
// Create a parent component that wraps child components with
a Provider

import { useState, React } from "react";
import { MyContext } from "./MyContext";
import MyComponent from "./MyComponent";

function App() {
  const [text, setText] = useState("");

  return (
    <div>
      <MyContext.Provider value={{ text, setText }}>
        <MyComponent />
      </MyContext.Provider>
    </div>
  );
}

export default App;
```
In this example, we have a parent component called App. This component has a state variable called "text", which is initially set to an empty string. We've also defined a function called `setText` that can be used to update the value of `text`.

Inside the return statement of the App component, we've wrapped the children of this component with the provider component ("MyContext.Provider"). Then we've passed an object to the value prop of the provider component that contains "text" and "setText" values.

3. Consume the Context

Now that we've created the provider component, we need to consume the context in other components. To do this, we use the "useContext" hook from React.

```
import { useContext } from 'react';
import { MyContext } from './MyContext';

function MyComponent() {
  const { text, setText } = useContext(MyContext);

  return (
    <div>
      <h1>{text}</h1>
      <button onClick={() => setText('Hello, world!')}>
        Click me
      </button>
    </div>
  );
}

export default MyComponent;
```
In this example, we've used the useContext hook to access the "text" and "setText" variables that were defined in the provider component.

Inside the return statement of "MyComponent", we've rendered a paragraph element that displays the value of `text`. We've also rendered a button that, when clicked, will call the `setText` function to update the value of `text` to "Hello, world!".

Tailwind CSS

Install Tailwind CSS

```
Install tailwindcss via npm, and then run the init command to
generate your tailwind.config.js file.

npm install -D tailwindcss
npx tailwindcss init
```

Configure your template paths

Add the paths to all of your template files in your `tailwind.config.js` file.

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Add the Tailwind directives to your CSS

Add the `@tailwind` directives for each of Tailwind's layers to your `./src/index.css` file.

@tailwind base;

@tailwind components;

@tailwind utilities;

Import /index.css in your component and use it

Tailwind Css Styling

Text :

Category	Class
Text Color	text-white, text-black, text-gray-500, text-red-500, text-yellow-500, text-green-500, text-blue-500, text-indigo-500, text-purple-500, text-pink-500
Text Size	text-xs, text-sm, text-base, text-lg, text-xl, text-2xl, text-3xl, text-4xl, text-5xl, text-6xl
Font Family	font-sans, font-serif, font-mono
Font Weight	font-thin, font-extralight, font-light, font-normal, font-medium, font-semibold, font-bold, font-extrabold, font-black
Text Alignment	text-left, text-center, text-right, text-justify
Text Decoration	underline, line-through, no-underline
Text Transform	uppercase, lowercase, capitalize, normal-case
Text Overflow	truncate, overflow-ellipsis, overflow-clip
Letter Spacing	tracking-tighter, tracking-tight, tracking-normal, tracking-wide, tracking-wider, tracking-widest
Line Height	leading-none, leading-tight, leading-snug, leading-normal, leading-relaxed, leading-loose, leading-[value]
Text Indentation	indent-0, indent-[value]
Font Style	italic, not-italic
Opacity	text-opacity-0, text-opacity-25, text-opacity-50, text-opacity-75, text-opacity-100
Whitespace Handling	whitespace-normal, whitespace-nowrap, whitespace-pre, whitespace-pre-line, whitespace-pre-wrap

Example :

```
<p class="text-xl font-bold text-blue-500 underline uppercase tracking-wide leading-relaxed">
Tailwind CSS is awesome! </p>
```

Layout:

Category	Class
Margin	m-0, m-40 (intermediate: m-1 to m-32)
Padding	p-0, p-40 (intermediate: p-1 to p-32)
Flexbox	flex, inline-flex
Flex Direction	flex-row, flex-col
Flex Wrap	flex-nowrap, flex-wrap
Justify Content	justify-start, justify-center, justify-end
Align Items	items-start, items-center, items-end
Align Self	self-auto, self-start, self-center, self-end
Grid	grid, inline-grid
Grid Columns	grid-cols-1, grid-cols-12 (intermediate: grid-cols-2 to grid-cols-11)
Grid Rows	grid-rows-1, grid-rows-6 (intermediate: grid-rows-2 to grid-rows-5)
Gap	gap-0, gap-40 (intermediate: gap-1 to gap-32)
Width	w-0, w-full (intermediate: w-1/2, w-1/3, w-1/4, etc.)
Height	h-0, h-full (intermediate: h-1/2, h-1/3, h-1/4, etc.)
Position	static, relative, absolute, fixed, sticky
Z-Index	z-0, z-50 (intermediate: z-10, z-20, z-30, z-40)
Overflow	overflow-auto, overflow-hidden, overflow-visible, overflow-scroll

Example for Layout

```html
<div class="container mx-auto p-4">
  <div class="flex flex-col items-center justify-center h-screen">
    <header class="w-full p-4 bg-blue-500">
      <h1 class="text-white">Header</h1>
    </header>
    <main class="flex-1 p-4 bg-gray-200">
      <p>Main content goes here.</p>
    </main>
    <footer class="w-full p-4 bg-blue-500">
      <p class="text-white">Footer</p>
    </footer>
  </div>
</div>
```

Example for Layout

```
<div class="container mx-auto p-4">
  <div class="flex flex-row justify-between items-start h-screen">
    <div class="flex-1 p-4 bg-red-500 m-2">
      <p class="text-white">Column 1</p>
    </div>
    <div class="flex-1 p-4 bg-green-500 m-2">
      <p class="text-white">Column 2</p>
    </div>
    <div class="flex-1 p-4 bg-blue-500 m-2">
      <p class="text-white">Column 3</p>
    </div>
  </div>
</div>
```

FlexBox Classes

Category	Class
Display	flex, inline-flex
Flex Direction	flex-row, flex-row-reverse, flex-col, flex-col-reverse
Flex Wrap	flex-nowrap, flex-wrap, flex-wrap-reverse
Align Items	items-start, items-end, items-center, items-baseline, items-stretch
Align Content	content-start, content-end, content-center, content-between, content-around, content-evenly
Align Self	self-auto, self-start, self-end, self-center, self-stretch
Justify Content	justify-start, justify-end, justify-center, justify-between, justify-around, justify-evenly
Flex	flex-1, flex-auto, flex-initial, flex-none
Flex Grow	flex-grow, flex-grow-0
Flex Shrink	flex-shrink, flex-shrink-0
Order	order-first, order-last, order-none, order-1, order-12
Gap	gap-0, gap-40 (intermediate: gap-1 to gap-32)
Width	w-0, w-full (intermediate: w-1/2, w-1/3, w-1/4, etc.)
Height	h-0, h-full (intermediate: h-1/2, h-1/3, h-1/4, etc.)
Margin	m-0, m-40 (intermediate: m-1 to m-32)
Padding	p-0, p-40 (intermediate: p-1 to p-32)

FlexBox Example

```
<div class="flex flex-row flex-wrap -mx-2">
      <div class="flex-1 md:w-1/2 px-2 mb-4">
        <div class="bg-white p-6 rounded shadow-md">
          <h2 class="text-xl font-bold mb-2">Column 1</h2>
          <p class="text-gray-700">Data 1: <span class="font-
medium">123</span></p>
        </div>
      </div>
      <div class="flex-1 md:w-1/2 px-2 mb-4">
        <div class="bg-white p-6 rounded shadow-md">
          <h2 class="text-xl font-bold mb-2">Column 2</h2>
          <p class="text-gray-700">Data 3: <span class="font-
medium">GHI</span></p>
        </div>
      </div>
      <div class="flex-1 md:w-1/2 px-2 mb-4">
        <div class="bg-white p-6 rounded shadow-md">
          <h2 class="text-xl font-bold mb-2">Column 3</h2>
          <p class="text-gray-700">Data 3: <span class="font-
medium">$300</span></p>
        </div>
      </div>
      <div class="w-full md:w-1/2 px-2 mb-4">
        <div class="bg-white p-6 rounded shadow-md">
          <h2 class="text-xl font-bold mb-2">Column 4</h2>
          <p class="text-gray-700">Data 1: <span class="font-
medium">X</span></p>
        </div>
      </div>
    </div>
```

Grid Classes

Table 1-3

Category	Class
Display	grid, inline-grid
Grid Template Columns	grid-cols-1, grid-cols-12
Grid Template Rows	grid-rows-1, grid-rows-6
Gap	gap-0, gap-40
Column Gap	col-gap-0, col-gap-40
Row Gap	row-gap-0, row-gap-40
Auto Flow	grid-flow-row, grid-flow-col, grid-flow-row-dense, grid-flow-col-dense
Auto Columns	auto-cols-auto, auto-cols-min, auto-cols-max, auto-cols-fr
Auto Rows	auto-rows-auto, auto-rows-min, auto-rows-max, auto-rows-fr
Template Areas	grid-areas-none

Example:

```
    <div class="grid grid-cols-3 gap-4">
      <div class=" bg-red-400 p-6 rounded shadow-md">
        <h2 class="text-xl justify-center font-bold mb-2">Column
1</h2>
      </div>
      <div class="bg-green-400 p-6 rounded shadow-md">
        <h2 class="text-xl font-bold mb-2">Column 2</h2>
      </div>
      <div class="bg-blue-400 p-6 rounded shadow-md">
        <h2 class="text-xl font-bold mb-2">Column 3</h2>
      </div>
      <div class="bg-yellow-400 p-6 rounded shadow-md">
        <h2 class="text-xl font-bold mb-2">Column 4</h2>
      </div>
    </div>
```

Alignment Classes

Table 1-4

Category	Classes	Description
Align Items	`items-start`, `items-end`, `items-center`, `items-baseline`, `items-stretch`	Aligns items vertically within their container based on the cross axis alignment.
Align Content	`content-start`, `content-end`, `content-center`, `content-between`, `content-around`, `content-evenly`	Aligns content horizontally within a flex container with multiple lines of content.
Align Self	`self-auto`, `self-start`, `self-end`, `self-center`, `self-stretch`	Aligns individual items within a flex container, overriding the default align-items setting.
Justify Content	`justify-start`, `justify-end`, `justify-center`, `justify-between`, `justify-around`, `justify-evenly`	Aligns items along the main axis (horizontally) within a flex container, distributing space accordingly.

Sizing Classes

Table 1-5

Category	Classes
Width	`w-{size}`, `min-w-{size}`, `max-w-{size}`
Height	`h-{size}`, `min-h-{size}`, `max-h-{size}`
Padding	`p-{size}`, `px-{size}`, `py-{size}`, `pt-{size}`, `pr-{size}`, `pb-{size}`, `pl-{size}`
Margin	`m-{size}`, `mx-{size}`, `my-{size}`, `mt-{size}`, `mr-{size}`, `mb-{size}`, `ml-{size}`
Responsive	`sm:{class}`, `md:{class}`, `lg:{class}`, `xl:{class}`, `2xl:{class}`
Aspect Ratio	`aspect-{ratio}`

Example:

```
<div className="container mx-auto">
     {/* Small Card */}
     <div className="bg-blue-200 w-64 h-32 p-4 mb-4">
       <p className="text-gray-700">Small Card</p>
     </div>

     {/* Medium Card */}
     <div className="bg-green-200 w-96 h-48 p-4 mb-4">
       <p className="text-gray-700">Medium Card</p>
     </div>

     {/* Large Card */}
     <div className="bg-yellow-200 w-full h-64 p-4 mb-4">
       <p className="text-gray-700">Large Card</p>
     </div>

     {/* Responsive Card */}
     <div className="bg-pink-200 sm:w-full md:h-96 p-4 mb-4">
       <p className="text-gray-700">Responsive Card</p>
     </div>
   </div>
```

Category	Classes	Description
Background Color	`bg-{color}`	Applies a solid background color where `{color}` can be a color name or hexadecimal value.
Background Image	`bg-{image}`, `bg-cover`, `bg-contain`	Applies a background image where `{image}` is the URL or path to the image.
Background Size	`bg-auto`, `bg-cover`, `bg-contain`	Sets the size of the background image.
Background Position	`bg-center`, `bg-left`, `bg-right`, `bg-top`, `bg-bottom`, `bg-left-top`, `bg-right-top`, `bg-left-bottom`, `bg-right-bottom`	Positions the background image relative to the container.
Background Repeat	`bg-no-repeat`, `bg-repeat`, `bg-repeat-x`, `bg-repeat-y`	Controls how the background image is repeated.
Background Attachment	`bg-fixed`, `bg-local`, `bg-scroll`	Controls whether and how the background image scrolls with the content.

Example:

```
 <div className="container mx-auto">
      {/* Background Color */}
      <div className="bg-blue-200 p-4 mb-4">
        <p className="text-gray-700">Background Color</p>
      </div>

      {/* Background Image */}
      <div className="bg-cover bg-center h-64 mb-4"
style={{ backgroundImage: "url('https://upload.wikimedia.org/
wikipedia/commons/thumb/4/41/Sunflower_from_Silesia2.jpg/1600px-
Sunflower_from_Silesia2.jpg')" }}>
        <p className="text-white p-4">Background Image</p>
      </div>

      {/* Gradient Background */}
      <div className="bg-gradient-to-r from-purple-500 via-
pink-500 to-red-500 p-4 mb-4">
        <p className="text-white">Gradient Background</p>
      </div>
    </div>
```

Border Classes

Category	Classes	Description
Border Width	`border`, `border-{size}`, `border-t-{size}`, `border-r-{size}`, `border-b-{size}`, `border-l-{size}`	Sets border width and sides.
Border Style	`border-solid`, `border-dashed`, `border-dotted`, `border-double`, `border-none`	Sets border style.
Border Color	`border-{color}`, `border-transparent`	Sets border color.
Rounded Corners	`rounded`, `rounded-{size}`, `rounded-t-{size}`, `rounded-r-{size}`, `rounded-b-{size}`, `rounded-l-{size}`	Sets rounded corner size and sides.
Border Collapse	`border-collapse`	Collapses borders on table elements.

Example for Border Classes

```
  <div class="container p-2 py-4">
<div class="mx-16 grid grid-cols-5 gap-2 p-2">
        <button class="ring-0 ring-green-600 bg-green-400
                    ring-opacity-25 w-full h-12 rounded-lg">
            Ring-0
        </button>
        <button class="ring-1 ring-green-600 bg-green-400
                    ring-opacity-25 w-full h-12 rounded-lg">
            Ring-1
        </button>
        <button class="ring-2 ring-green-600 bg-green-400
                    ring-opacity-25 w-full h-12 rounded-lg">
            Ring-2
        </button>
        <button class="ring-4 ring-green-600 bg-green-400
                    ring-opacity-25 w-full h-12 rounded-lg">
            Ring-4
        </button>
        <button class="ring-8 ring-green-600 bg-green-400
                    ring-opacity-25 w-full h-12 rounded-lg">
            Ring-8
        </button>
    </div>
    <div className="container mx-auto">
      {/* Border Example */}
      <div className="border border-gray-300 p-4 mb-4">
        <p className="text-gray-700">Border Example</p>
      </div>

      {/* Rounded Corners */}
      <div className="border border-blue-500 rounded-lg p-4 mb-4">
        <p className="text-blue-500">Rounded Border</p>
      </div>
    </div>
        </div>
```

Table Classes

Category	Classes	Description
Table Container	`container`	Centers the table within its container.
Table	`table-auto`, `table-fixed`	Sets the table layout (`auto` or `fixed`).
Table Head	`thead`, `bg-gray-50`	Styles the table head section and sets background color for head rows.
Table Row	`tr`, `even:bg-gray-100`	Styles table rows and alternates background color for even rows.
Table Cell	`th`, `td`, `border`, `border-gray-200`, `px-6`, `py-4`	Styles table header and data cells, adds borders and padding.
Text Alignment	`text-left`, `text-center`, `text-right`	Aligns text within table cells (left, center, right).
Table Borders	`border-collapse`, `divide-y`, `divide-gray-200`	Collapses borders between table cells, adds vertical dividers between rows with specified color.
Responsive Table	`overflow-x-auto`	Enables horizontal scrolling on small screens for wide tables.

Example:

```
<div className="container mx-auto justify-centre py-3 px-2">
      <table className="min-w-full divide-y divide-gray-200">
        <thead className="bg-gray-50">
          <tr>
            <th className="px-6 py-3 text-left text-xs font-small
text-gray-500 uppercase tracking-wider">Name</th>
            <th className="px-6 py-3 text-left text-xs font-small
text-gray-500 uppercase tracking-wider">Email</th>
            <th className="px-6 py-3 text-left text-xs font-small
text-gray-500 uppercase tracking-wider">Role</th>
          </tr>
        </thead>
        <tbody className="bg-white divide-y divide-gray-200">
          <tr className="even:bg-gray-100">
            <td className="px-6 py-4 whitespace-nowrap text-
xs">John Doe</td>
            <td className="px-6 py-4 whitespace-nowrap  text-
xs">john.doe@example.com</td>
```

```
            <td className="px-6 py-4 whitespace-nowrap  text-
xs">Admin</td>
          </tr>
          <tr>
            <td className="px-6 py-4 whitespace-nowrap  text-
xs">Jane Smith</td>
            <td className="px-6 py-4 whitespace-nowrap  text-
xs">jane.smith@example.com</td>
            <td className="px-6 py-4 whitespace-nowrap  text-
xs">User</td>
          </tr>
        </tbody>
      </table>
    </div>
```

Form Classes

Category	Classes	Description
Form Container	`container`	Centers the form within its container.
Form	`max-w-md`, `bg-white`, `shadow-md`, `rounded`, `px-8`, `pt-6`, `pb-8`, `mb-4`	Styles for form container and layout.
Input Fields	`w-full`, `sm:w-full`, `md:w-full`, `lg:w-full`, `bg-gray-100`, `border-gray-300`, `text-gray-700`, `rounded-md`, `shadow`, `appearance-none`, `py-2`, `px-3`, `leading-tight`, `focus:outline-none`, `focus:border-blue-500`	Styles for input fields including size, colors, borders, and focus states.
Labels	`block`, `text-sm`, `font-bold`, `text-gray-700`, `mb-2`	Styles for labels.
Buttons	`bg-blue-500`, `hover:bg-blue-600`, `text-white`, `font-bold`, `py-2`, `px-4`, `rounded`, `focus:outline-none`, `focus:shadow-outline`	Styles for buttons including background color, hover state, and focus state.
Checkboxes	`form-checkbox`, `text-blue-500`	Styles for checkboxes.
Radios	`form-radio`, `text-blue-500`	Styles for radio buttons.
Validation States	`border-red-500`, `text-red-500`	Styles for error messages and borders in validation states.
Responsive Design	`sm:`, `md:`, `lg:`	Responsive utilities for adjusting layout and styles based on screen size.

Example Form

```
   <form className="max-w-md mx-auto bg-white shadow-md rounded
px-8 pt-6 pb-8 mb-4">
        <div className="mb-4">
          <label className="block text-gray-700 text-sm font-bold
mb-2" htmlFor="username">
            Username
          </label>
          <input
            className="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none
focus:border-blue-500"
            id="username"
            type="text"
            placeholder="Enter your username"
          />
        </div>
        <div className="mb-6">
          <label className="block text-gray-700 text-sm font-bold
mb-2" htmlFor="password">
            Password
          </label>
          <input
            className="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none
focus:border-blue-500"
            id="password"
            type="password"
            placeholder="Enter your password"
          />
        </div>
        <div className="flex items-center justify-between">
          <button
            className="bg-blue-500 hover:bg-blue-600 text-white
font-bold py-2 px-4 rounded focus:outline-none focus:shadow-
outline"
            type="button"
          >
            Sign In
          </button>
        </div>
      </form>
```