Handling File Uploads in React Forms

To handle file uploads in React, you use an `<input>` element with type `"file"` and capture the selected file in state. You can then pass the file to a backend API for processing.

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



function FileUpload() {

  const [selectedFile, setSelectedFile] = useState(null);



  const handleFileChange = (event) => {

    setSelectedFile(event.target.files[0]);

  };



  const handleSubmit = (event) => {

    event.preventDefault();

    if (selectedFile) {

      const formData = new FormData();

      formData.append('file', selectedFile);



      // Make API request

      fetch('/upload', {

        method: 'POST',

        body: formData,

      })

        .then(response => response.json())

        .then(data => console.log(data))

        .catch(error => console.error('Error:', error));
```

```
   } else {
     alert('Please select a file.');
   }
 };

 return (
   <form onSubmit={handleSubmit}>
     <input type="file" onChange={handleFileChange} />
     <button type="submit">Upload</button>
   </form>
 );
}

export default FileUpload;
```

Handling Dependent Dropdowns in React Forms

Dependent dropdowns are dropdown menus where the options in one depend on the selection in another. To implement this, manage the states of both dropdowns and filter options dynamically.

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

function DependentDropdowns() {
 const countries = [
   { name: 'India', states: ['Maharashtra', 'Gujarat', 'Kerala'] },
   { name: 'USA', states: ['California', 'Texas', 'Florida'] },
 ];

 const [selectedCountry, setSelectedCountry] = useState('');
 const [states, setStates] = useState([]);
 const handleCountryChange = (event) => {
   const country = event.target.value;
   setSelectedCountry(country);
   const countryData = countries.find(c => c.name === country);
   setStates(countryData ? countryData.states : []);
 };

 return (
   <div>
     <label>
       Country:
       <select onChange={handleCountryChange}>
         <option value="">Select Country</option>
         {countries.map((country, index) => (
           <option key={index} value={country.name}>{country.name}</option>
         ))}
       </select>
     </label>

     <label>
       State:
       <select disabled={!states.length}>
         <option value="">Select State</option>
         {states.map((state, index) => (
           <option key={index} value={state}>{state}</option>
         ))}
       </select>
     </label>
   </div>
 );
```

```
}

export default DependentDropdowns;
```

Handling Multiple Checkboxes in a React Form

To handle multiple checkboxes, use state to keep track of which checkboxes are selected. You can use an array to store the values of the checked boxes.

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

function MultiCheckboxForm() {
  const options = ['Option 1', 'Option 2', 'Option 3'];
  const [selectedOptions, setSelectedOptions] = useState([]);

  const handleCheckboxChange = (event) => {
    const value = event.target.value;
    setSelectedOptions(prev =>
      prev.includes(value)
        ? prev.filter(option => option !== value)
        : [...prev, value]
    );
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log('Selected options:', selectedOptions);
  };

  return (
    <form onSubmit={handleSubmit}>
      {options.map((option, index) => (
        <label key={index}>
        <input
          type="checkbox"
          value={option}
          checked={selectedOptions.includes(option)}
          onChange={handleCheckboxChange}
        />
        {option}
      </label>
    ))}
    <button type="submit">Submit</button>
  </form>
 );
}

export default MultiCheckboxForm;
```