React Forms

❮ Previous
Next ❯

Just like in HTML, React uses forms to allow users to interact with the web page.

Adding Forms in React

You add a form with React like any other element:

Example:

Get your own React.js Server

Add a form that allows users to enter their name:

```
function MyForm() {
  return (
    <form>
      <label>Enter your name:
        <input type="text" />
      </label>
    </form>
  )
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyForm />);
```

Run Example »

This will work as normal, the form will submit and the page will refresh.

But this is generally not what we want to happen in React.

We want to prevent this default behavior and let React control the form.

Handling Forms

Handling forms is about how you handle the data when it changes value or gets submitted.

In HTML, form data is usually handled by the DOM.

In React, form data is usually handled by the components.

When the data is handled by the components, all the data is stored in the component state.

You can control changes by adding event handlers in the `onChange` attribute.

We can use the `useState` Hook to keep track of each inputs value and provide a "single source of truth" for the entire application.

See the React Hooks section for more information on Hooks.

Example:

Use the `useState` Hook to manage the input:

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';
```

```
function MyForm() {
  const [name, setName] = useState("");

  return (
    <form>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
    </form>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyForm />);
```

Run Example »

Submitting Forms

You can control the submit action by adding an event handler in the `onSubmit` attribute for the `<form>`:

Example:

Add a submit button and an event handler in the `onSubmit` attribute:

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';

function MyForm() {
  const [name, setName] = useState("");

  const handleSubmit = (event) => {
    event.preventDefault();
    alert(`The name you entered was: ${name}`)
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
      <input type="submit" />
    </form>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyForm />);
```

Run Example »

Multiple Input Fields

You can control the values of more than one input field by adding a `name` attribute to each element.

We will initialize our state with an empty object.

To access the fields in the event handler use the `event.target.name` and `event.target.value` syntax.

To update the state, use square brackets [bracket notation] around the property name.

Example:

Write a form with two input fields:

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';

function MyForm() {
  const [inputs, setInputs] = useState({});

  const handleChange = (event) => {
    const name = event.target.name;
    const value = event.target.value;
    setInputs(values => ({...values, [name]: value}))
  }

  const handleSubmit = (event) => {
    event.preventDefault();
    alert(inputs);
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
      <input
```

```
          type="text"
          name="username"
          value={inputs.username || ""}
          onChange={handleChange}
        />
        </label>
        <label>Enter your age:
          <input
            type="number"
            name="age"
            value={inputs.age || ""}
            onChange={handleChange}
          />
          </label>
          <input type="submit" />
      </form>
    )
  }

  const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<MyForm />);
```

Run Example »

Note: We use the same event handler function for both input fields, we could write one event handler for each, but this gives us much cleaner code and is the preferred way in React.

Textarea

The textarea element in React is slightly different from ordinary HTML.

In HTML the value of a textarea was the text between the start tag `<textarea>` and the end tag `</textarea>`.

```
  <textarea>
    Content of the textarea.


</textarea>
```

In React the value of a textarea is placed in a value attribute. We'll use the `useState` Hook to manage the value of the textarea:

Example:

A simple textarea with some content:

```
  import { useState } from 'react';
  import ReactDOM from 'react-dom/client';

  function MyForm() {
    const [textarea, setTextarea] = useState(
      "The content of a textarea goes in the value attribute"
    );

    const handleChange = (event) => {
      setTextarea(event.target.value)
    }

    return (
      <form>
        <textarea value={textarea} onChange={handleChange} />
      </form>
    )
  }

  const root = ReactDOM.createRoot(document.getElementById('root'));


root.render(<MyForm />);
```

Run Example »

Select

A drop down list, or a select box, in React is also a bit different from HTML.

In HTML, the selected value in the drop down list was defined with the `selected` attribute:

HTML:

```
<select>
  <option value="Ford">Ford</option>
  <option value="Volvo" selected>Volvo</option>
  <option value="Fiat">Fiat</option>

</select>
```

In React, the selected value is defined with a `value` attribute on the `select` tag:

Example:

A simple select box, where the selected value "Volvo" is initialized in the constructor:

```
function MyForm() {
  const [myCar, setMyCar] = useState("Volvo");

  const handleChange = (event) => {
```

```
    setMyCar(event.target.value)
  }

  return (
    <form>
      <select value={myCar} onChange={handleChange}>
        <option value="Ford">Ford</option>
        <option value="Volvo">Volvo</option>
        <option value="Fiat">Fiat</option>
      </select>
    </form>
  )
}
```

Run Example »

By making these slight changes to `<textarea>` and `<select>`, React is able to handle all input elements in the same way.