Events and React State

Download Demo Code <../react-events-state-demo.zip>

Goals

- Attach event handlers to components in React
- Define React state
- Initialize and change state with ***useState***
- Write event handlers to change component state

Events in React

DOM vs. React

- React events are *camelCase*, rather than *lowercase*.
- With JSX you pass a function as event handler, rather than a string.

Example

```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

Is slightly different in React:

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

Event Attributes

Any event you can listen for in JS, you can listen for in React.

Examples:

- Mouse events: `onClick`, `onMouseOver`, etc
- Form events: `onSubmit`, etc
- Keyboard events: `onKeyDown`, `onKeyUp`, `onKeyPress`
- Full list <https://reactjs.org/docs/events.html#supported-events>

An example in a component

demo/click-me/src/GoodClick.js

```
import React from "react";
```

```
function handleClick() {
  console.log("GoodClick clicked!");
}

function GoodClick() {
  return (
    <button onClick={handleClick}>
      GoodClick
    </button>
  );
}

export default GoodClick;
```

Functions vs. Invocations

Remember: event listeners expect to receive *functions* as values.

Don't invoke your function when you pass it!

❌

demo/click-me/src/BrokenClick.js

```
function handleClick() {
  console.log("BrokenClick clicked!");
}

function BrokenClick() {
  return (
    <button onClick={handleClick()}>
      BrokenClick
    </button>
  );
}
```

✅

demo/click-me/src/GoodClick.js

```
function handleClick() {
  console.log("GoodClick clicked!");
}

function GoodClick() {
  return (
    <button onClick={handleClick}>
      GoodClick
    </button>
  );
}
```

React State

Core React Concepts

- component
  - building block of React
  - combines logic and presentation
- prop
  - data passed to a component *(or found via defaults)*
  - immutable; component cannot change its own props
- state
  - data specific to a component
  - can change!

What common things belong in state

- Hiding or showing some data (toggling)
- Fetching data from an API (starts empty and changes to be populated)
- Themes, colors or styles that change based on an event
- When working with some information, ask yourself - will this ever change?
  - If so, it should be somewhere in state!

State

In React, state is created using ***useState***

useState returns an array with two values

- What the piece of state is
- A function to change it

```
const [mood, setMood] = useState("happy");
```

We are using array destructuring to extract the values from the result of ***useState***.

Initial State

To set initial state, do so in the component:

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

function Person() {
  const [mood, setMood] = useState("happy");

  return(
    <div> Your mood is {mood} </div>
  );
}
```

- We import ***useState*** from React
- ***useState*** takes one argument - whatever you'd like the initial state to be
- You must call ***useState*** in the component
- You cannot call ***useState*** in loops or conditionals
- Try to do state initialization early in your function component

Naming conventions

- The name of the hook is called ***useState***.
- We can call the return values from useState whatever we want.
- However, it's conventional to go with "x" and "setX".

Changing State

- We'll do this using our ***setMood*** function!
- Whatever we pass to this function will be the new value of ***mood***

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

function Person() {
  const [mood, setMood] = useState("happy");

  return (
    <div>
      <div> Your mood is {mood} </div>
      <button onClick={() => setMood('excited')}> Change! </button>
    </div>
  );
}
```

We wrap the ***setMood*** call in an arrow function so that ***onClick*** receives a function.

Click Rando

Let's see another example!

demo/click-me/src/random.js

```
/** get a random integer between 0 and max. *
function getRandom(max) {
  return Math.floor(Math.random() * max);
}

export { getRandom };
```

demo/click-me/src/ClickRando.js

```
import React, { useState } from "react";
import { getRandom } from "./random";

/** A random number that changes. */
function ClickRando(props) {
  const max = props.maxNum;
  const [num, setNum] = useState(getRandom(max

  return (
    <i onClick={() => setNum(getRandom(max))}>
      Click Rando: {num}
    </i>
  );
}

export default ClickRando;
```

Multiple Pieces of State

You can call ***useState*** multiple times if a component needs multiple pieces of state.

demo/click-me/src/Complex.js

```
import React, { useState } from "react";
import { getRandom } from "./random";

/** An example of a component with state/props/children. */

function Complex(props) {
```

```
  const [pushed, setPushed] = useState(false);
  const [num, setNum] = useState(getRandom(props.maxNum));

  function handleClick() {
    setPushed(true);
    setNum(getRandom(props.maxNum));
  }

  return (
    <button className="btn" onClick={handleClick}>
      <b>{pushed ? `Number: ${num}` : props.text}</b>
    </button>
  );
}

export default Complex;
```

State vs Props

A common question: what belongs in state and what belongs in props?

If the data will ever change, it needs to be in state!

Example: Let's build a game!

- If we want to build a game with a board, we might want a component called ***GameBoard***.
- ***GameBoard*** will have a score - props or state?
- ***GameBoard*** will have a certain numRows - props or state?
- ***GameBoard*** will have a certain numColumns - props or state?
- ***GameBoard*** will display text if the game is over - props or state?

Coming Up

- More on state
- More on events
- Passing functions that change state
- Testing!