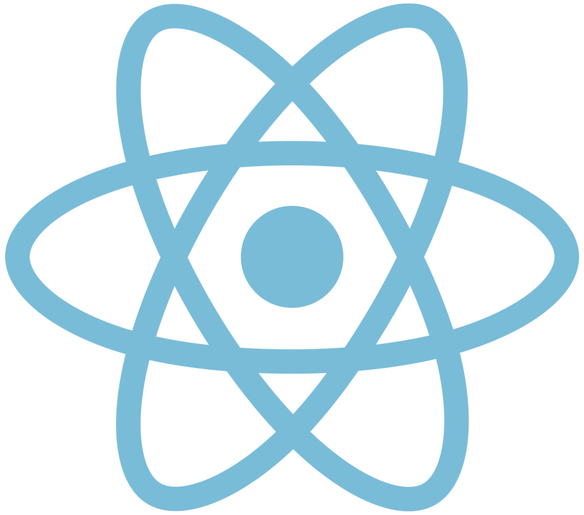

React State

Agenda

- The problem - why would we need state
- Props vs State
- Hooks
- Event handlers
- Takeaways

State vs. Props

- **Props** are data passed in from a parent component
  - Values can be rendered
  - Values are NOT intended to be changed
- **State** is defined within a component
  - Values can be rendered
  - Created with the sole purpose of being changed
  - Changes cause the component to re-render

Intro to hooks

- A “hook” is the term for a special function provided by a library (such as React)
- They provide bundled functionality for common issues
- To create component state we use the **`useState`** hook

```
import React, { useState } from ‘react’

const [count, setCount] = useState(0)
```

Defining state

- Use the `useState` function from React
  - Returns 2 item array (value & update function)
  - Often use array destructuring to define
  - Pass initial value to `useState`
- Must be defined *inside* of the component

```
const [val, setVal] = useState('initial value')
```

Updating state

- Always use the `setXxx` function from `useState`
  - Never with the assignment operator (`=`)
- Usually called from within an event handler
- Causes the component to be re-rendered
- Might need to use `preventDefault` to prevent the event from performing its default behaviour (not related to state)

Adding event handlers

- Added to elements as attributes/props
- Use camelCase `onEvent` syntax
- Always are assigned a function as their value

```
<button onClick={handleClick}>Click me</button>
<input name=”form” onChange={handleChange} />
```

Takeaways

- Props are for passing data from one component to another
- State is for data that is changing - and changes cause a re-render
- There are multiple hooks available. useState is used for defining, using, and updating state
- State is usually changed with even handlers