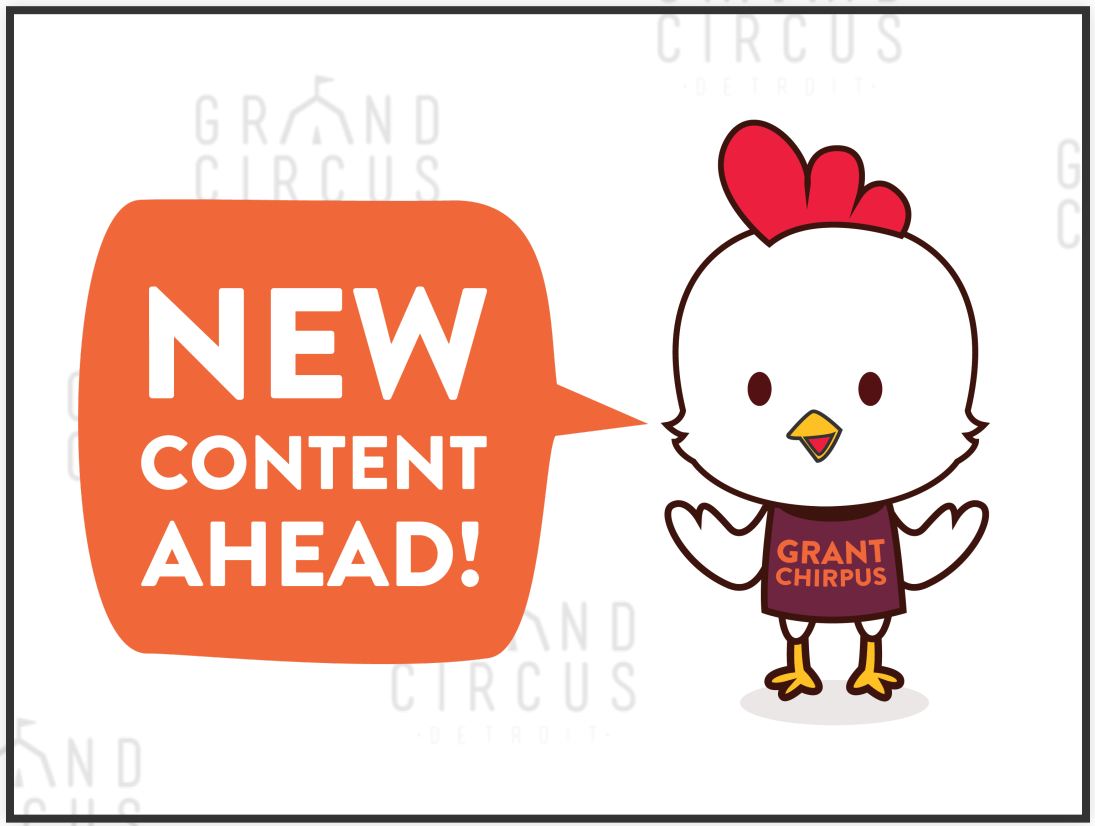
REACT WORKSHOP

SESSION 2

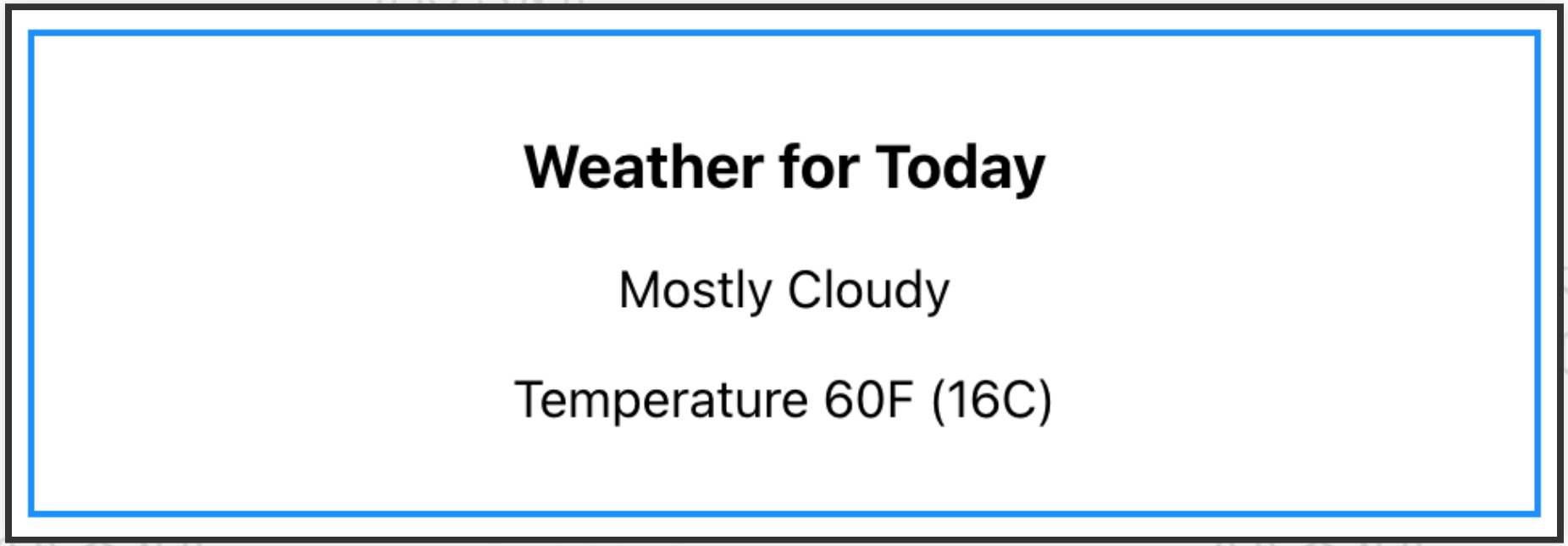
REVIEW

- Components
- JSX
- State


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

PROPS

CODE ALONG



If you get behind, don't worry. There will be time at the end to catch up.

KEY TERM: PROPS

We use props (short for "Properties") to pass data from one component to another. They are sent in attributes and received via a destructured function parameter.

```
// Send
<NameTag name="Tigger" />

// Receive
function NameTag({name}) {
```

SENDING PROPS

Props can be sent using string literals or {} expressions.

```
<Contact name="Grant" email="hello@grandcircus.co" />
<Grade label="Geography Quiz" score={3} />
<Grade label={myQuiz} score={10 - 7} passing={false} />
<Grade label="Alaska Worksheet" score={myScore} passing={isPassingScore(myScore)} />
```

SENDING PROPS

Also, a prop without a value is true.

```
<Star name="Sun" hasPlanets={true} />
```

is the same as

```
<Star name="Sun" hasPlanets />
```

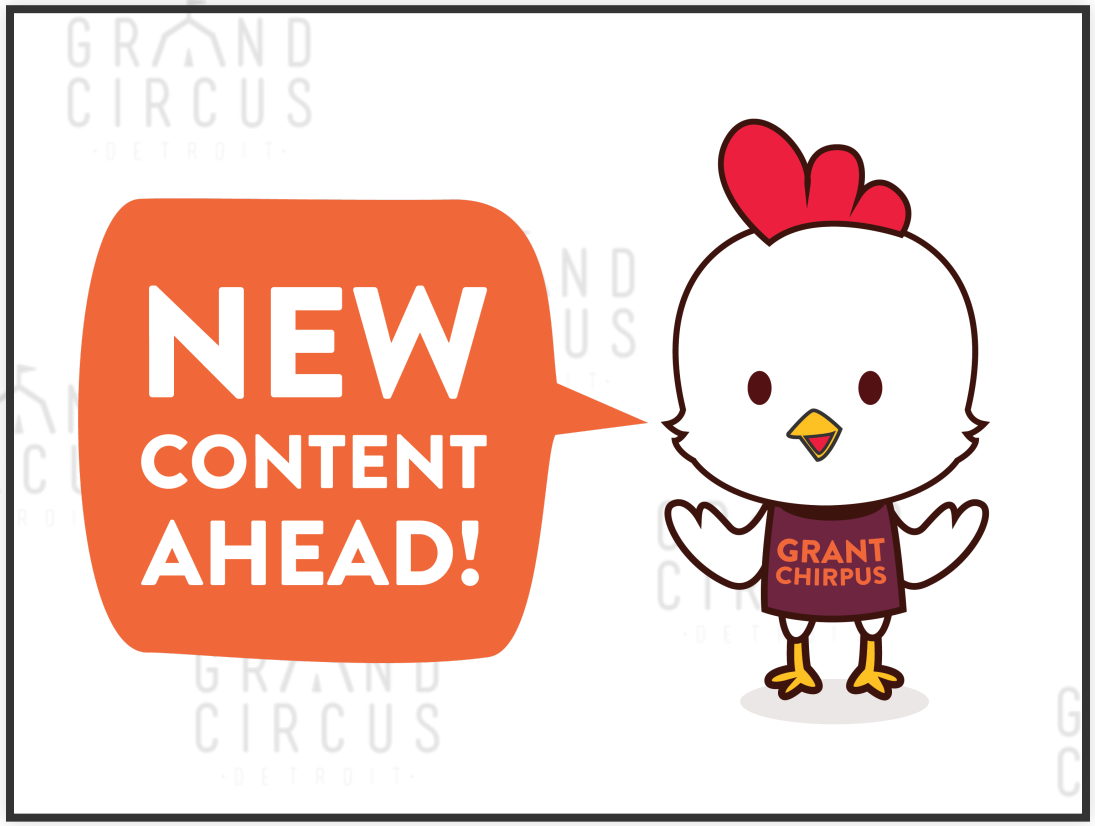
STATE VS. PROPS

State	Props
Owned and controlled by the component. Not set from outside. Can change over time.	Controlled by a parent component. Not changed by this component.

EXERCISE 5

QUOTES

USING STATE & PROPS TOGETHER


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

ARRAYS

CODE ALONG

If you get behind, don't worry. There will be time at the end to catch up.

ARRAYS WITH JSX

Use Array.map to convert the data into JSX elements or components.

```
const things = ["raindrops", "whiskers", "kettle", "mittens"];

<ul>
  {things.map(thing => <li>{thing}</li>)}
</ul>
```

USE A KEY

React prefers that you provide a key. The key should uniquely identify the item in the array.

This is not required, but it helps performance. You'll get a console warning if you omit it.

```
const things = ["raindrops", "whiskers", "kettle", "mittens"];

<ul>
  {things.map(thing => <li key={thing}>{thing}</li>)}
</ul>
```

EXAMPLE WITH OBJECTS

```
const things = [
  { name: "raindrops", description: "on roses" },
  { name: "whiskers", description: "on kittens" },
  { name: "kettle", description: "bright, copper" }
];

<ul>
  {things.map(thing =>
    <Thing key={thing.name} name={thing.name} desc={thing.description} />)}
</ul>
```

WHAT MAKES A GOOD KEY?

- Unique
- Doesn't change
- You can use the array index if the array will not have items moved or deleted.

See: Official Docs

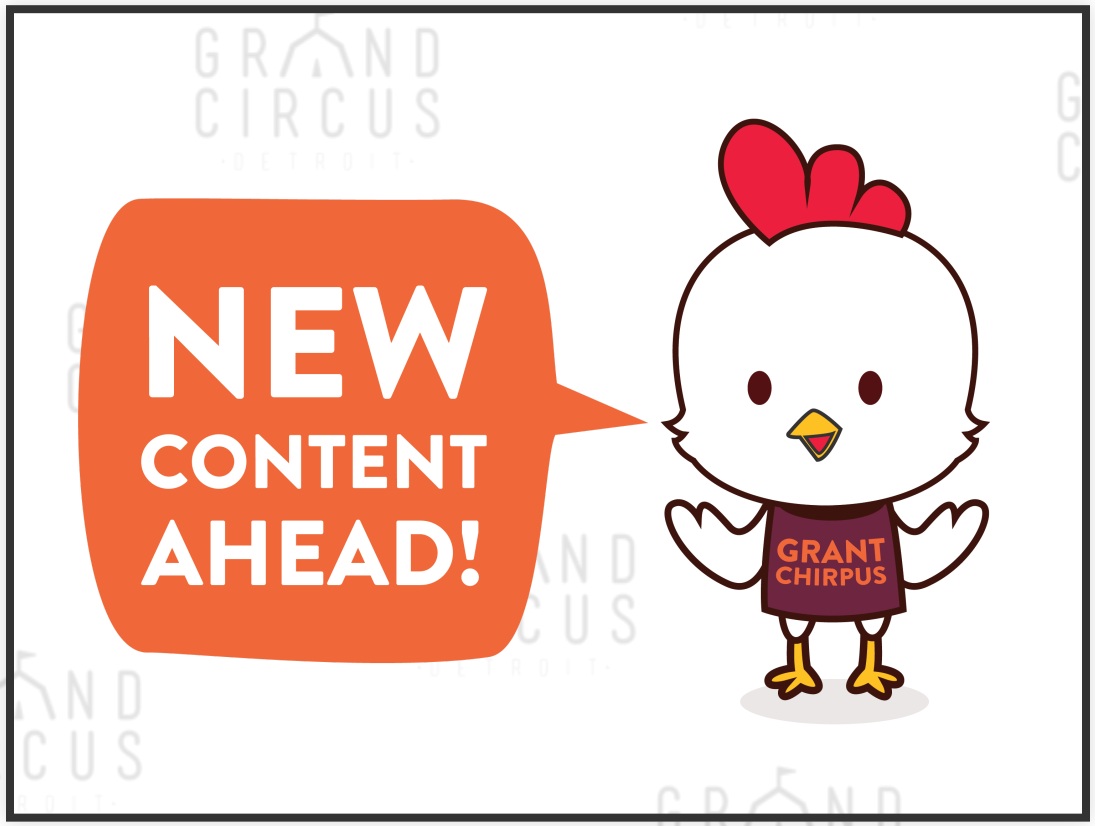
EXAMPLE USING INDEX

```
const things = [
  { name: "raindrops", description: "on roses" },
  { name: "whiskers", description: "on kittens" },
  { name: "kettle", description: "bright, copper" }
];

<ul>
  {things.map((thing, i) =>
    <Thing key={i} name={thing.name} desc={thing.description} />)}
</ul>
```

EXERCISE 6

QUOTE LIST


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

CALLING APIS

DEMO

Just watch. Don't code along.

STORE IN STATE

All variable data in React is stored in component state somewhere. This includes data loaded from APIs.

Initialize the state to an empty value until the API loads.

```
const [weatherData, setWeatherData] = useState([]);
```

USEEFFECT

Keep in mind that the component function runs often to refresh what's shown on the page.

The `useEffect` hook allows us to run code just once when the component first loads. It takes two parameters:

1. a callback function for your code.
2. an array of dependencies, which if empty indicates to only run the function once.

```
useEffect(() => {
  // API call goes here.
}, []);
```

USEEFFECT

See the React documentation to learn more about useEffect.

https://reactjs.org/docs/hooks-effect.html

API CALL

You can use any JavaScript tool to make your API call. Here's an example with `fetch`, which is built into modern browsers.

When the data comes back from the API, call the "set" function on your state.

```
fetch("https://api.weather.gov/gridpoints/DTX/65,33/forecast")
.then(res => res.json())
.then(data => {
  // when data comes back from API, set state with the part we need
  setWeatherData(data.properties.periods);
});
```

USE THE DATA

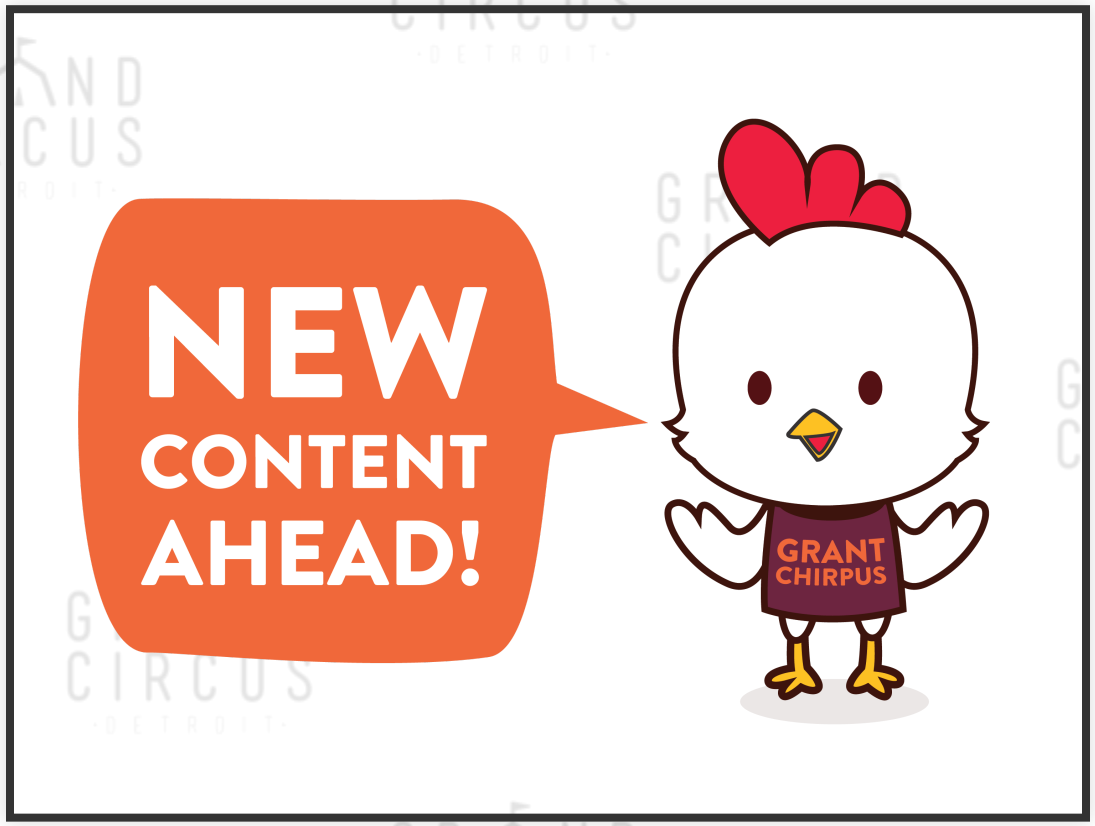
Just use the state data to display the information.

Keep in mind that for a short time while the API is loading, the data will be empty.

```
{ weatherData.length === 0 ? "Loading..." :
  weatherData.map(wx =>
    <Weather key={wx.number} time={wx.name} conditions={wx.shortForecast} />)
}
```

EXERCISE 7

QUOTE API


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

CALLBACK PROPS

DEMO

Just watch. Don't code along.

CALLBACK PROPS

Which component holds the state of the current code?

Which component has the buttons?

CALLBACK PROPS

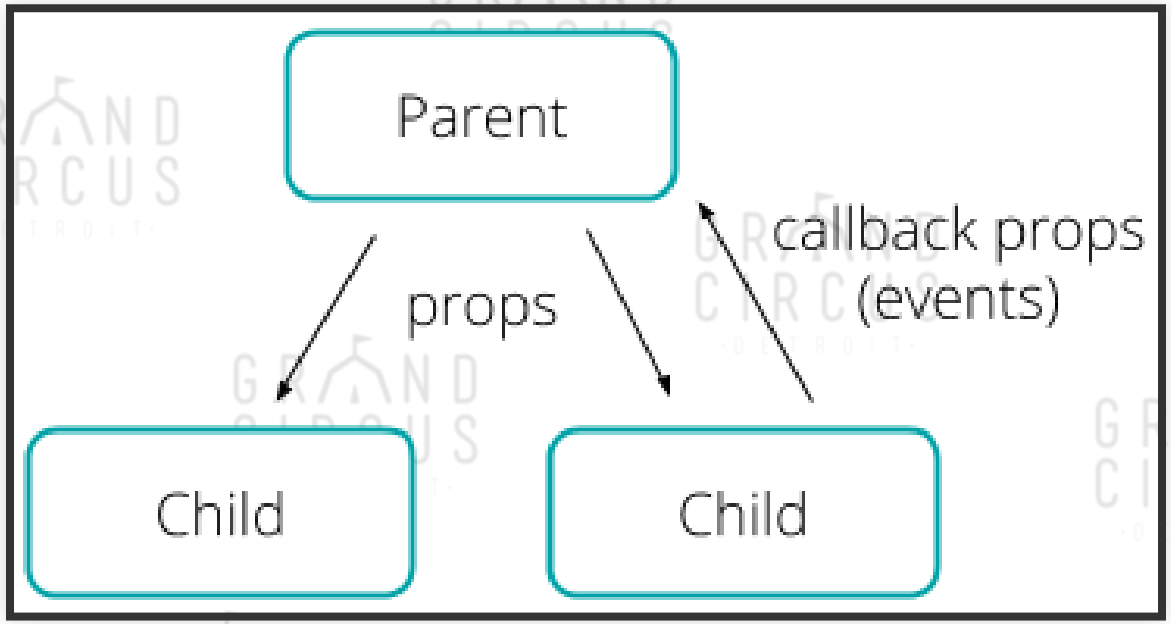
`CodeEntry` passes in a function (appendToCode)

```
<Keypad onKey={appendToCode}/>
```

`Keypad` calls that function (the onKey prop)

```
<button onClick={() => onKey(val)}>{val}</button>
```

DATA FLOW BETWEEN COMPONENTS



Props pass data down to children. *Callbacks* can pass data and events up to parents.

FUTHER TOPICS FOR STUDY

- Modifying arrays and objects in state following immutability practices
- Using forms with React
- Project: Create a CRUD app: a list where a user can add (via a form), remove, and modify items.