App and Web Intelligence Lab

Shan-Hung Wu & DataLab

CS, NTHU

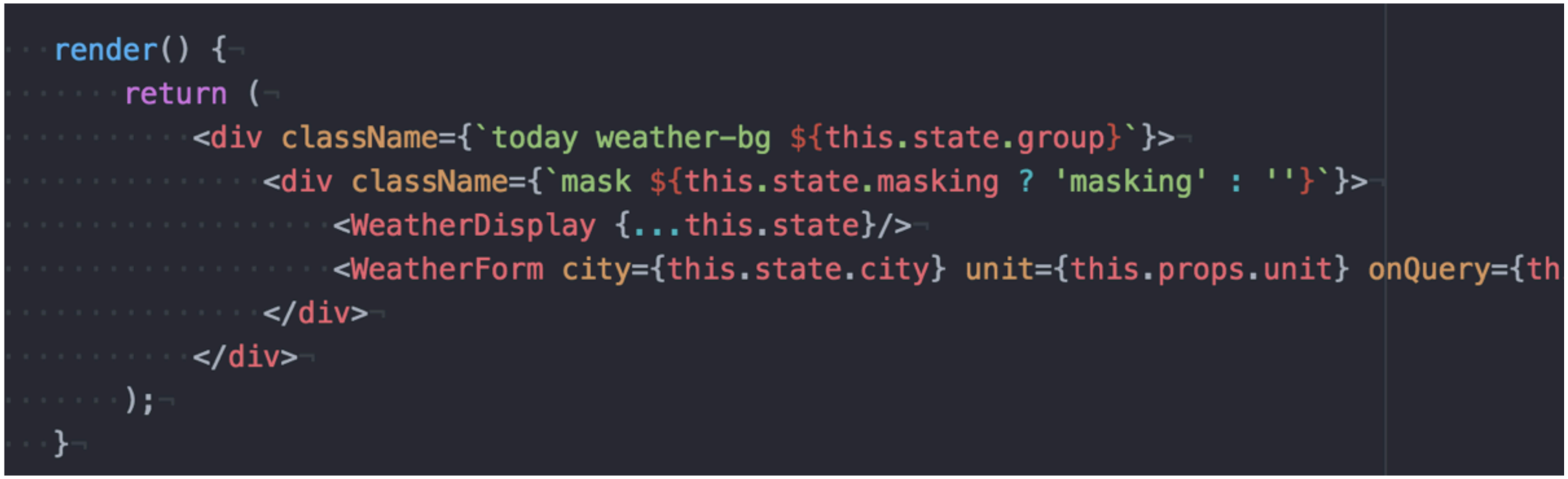
Resources

https://nthu-datalab.github.io/unitecsummercourse/

React

- Component-based
- Debug-friendly
  - Syntax errors fails at compile time
- Extends to mobile landscape (React Native)

Write HTML in JS



```
  render() {
      return (
          <div className={`today weather-bg ${this.state.group}`}>
              <div className={`mask ${this.state.masking ? 'masking' : ''}`}>
                  <WeatherDisplay {...this.state}/>
                  <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={th
              </div>
          </div>
      );
  }
```

JSX File

```
// in index.js
import React from 'react';
import ReactDOM from 'react-dom';

window.onload = function() {
  let name = 'Bob';
  ReactDOM.render(
    <h1>Hello {name}</h1>, // JSX, no quotes
    document.getElementById('root')
  );
};
```

- ***JSX*** allows writing HTML in JS
- Compiled to normal ***objects*** by Babel

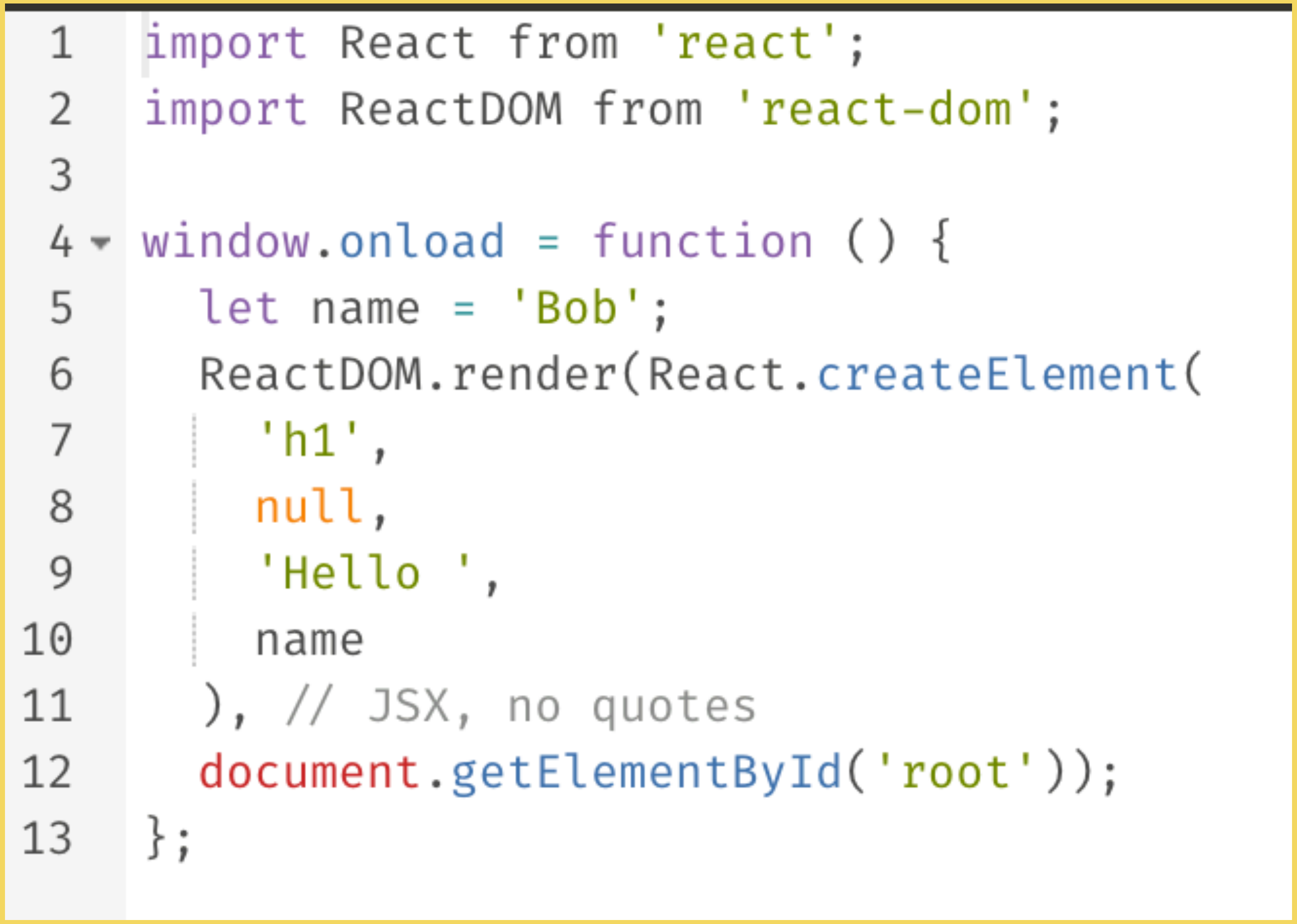
```
const el = <h1>Hello {name}</h1>;
// compiled to
const el = React.createElement('h1', ...);
```

ES6/7 and BABEL

- ES6 (2015) and 7 (2016) are not fully supported by major browsers yet
- Babel: a transpiler that transforms ES6/7 syntax into ES5
- Modular: plug-ins and presets
  - E.g., preset-es2015
  - Only syntax translation by default

ES6/7 and BABEL

- babeljs.io REPL



```
import React from 'react';
import ReactDOM from 'react-dom';

window.onload = function () {
  let name = 'Bob';
  ReactDOM.render(React.createElement(
    'h1',
    null,
    'Hello ',
    name
  ), // JSX, no quotes
  document.getElementById('root'));
};
```

Fast Re-rendering

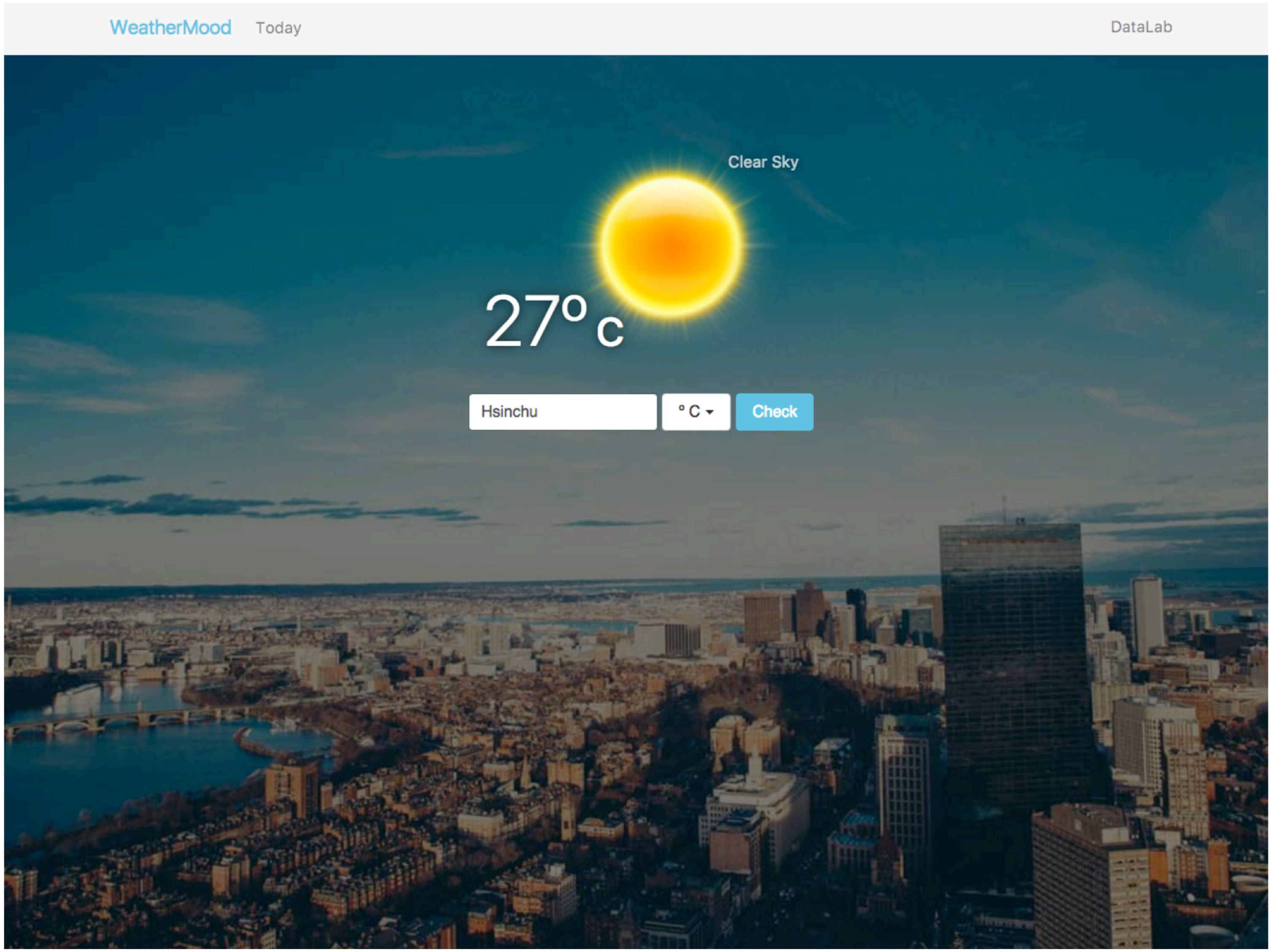
- React keeps a virtual DOM in memory
  - Updates only ***changed elements***

```
function tick() {
  const date = new Date().toLocaleTimeString();
  const el = (
    <div>
      <h1>Hello</h1>
      <h2>It's {date}.</h2>
    </div>
  );
  ReactDOM.render(el, document.getElementById('root'));
}
setInterval(tick, 1000);
```

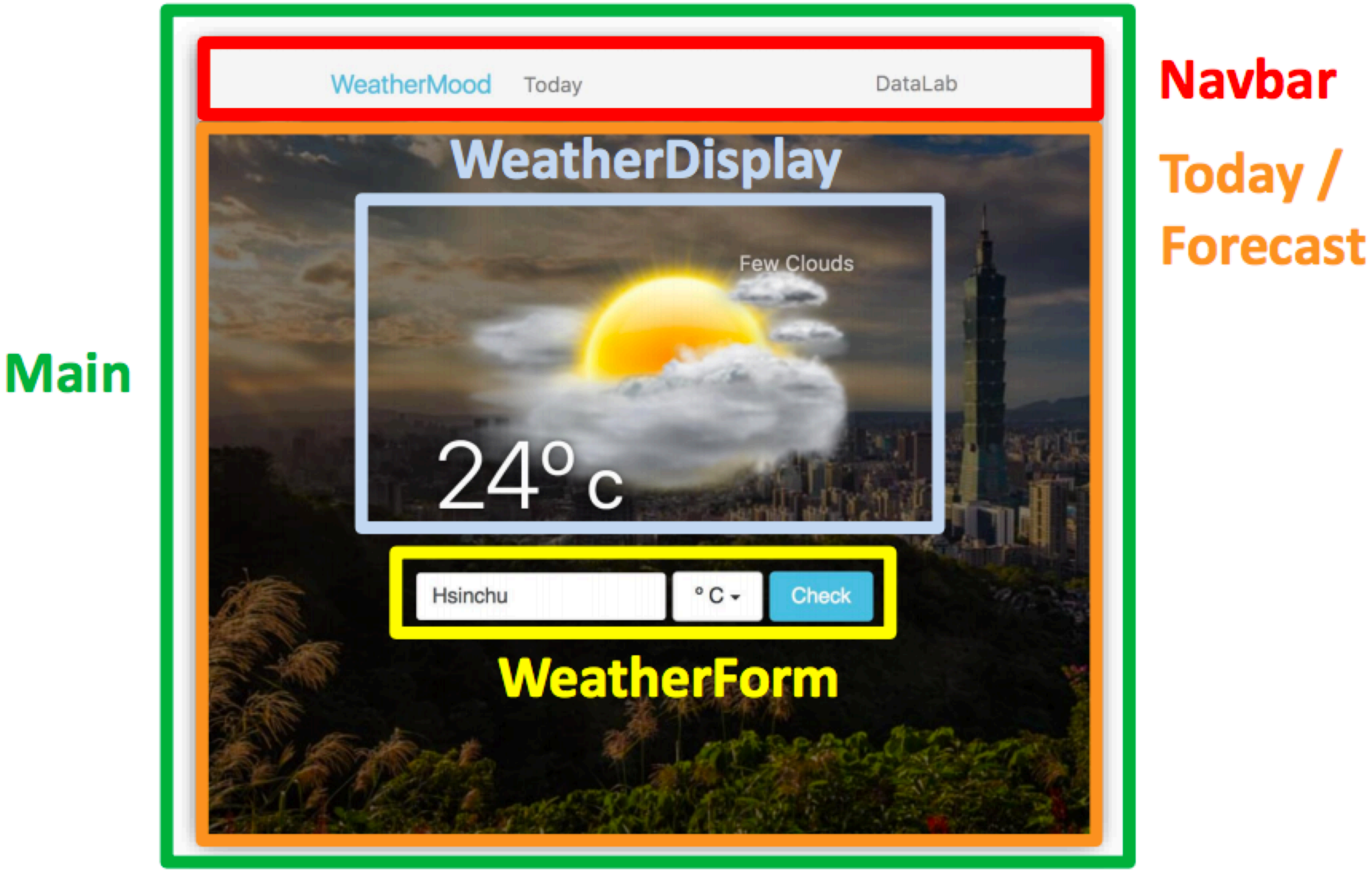
Components

- Small piece of the UI
- Interaction
- Own structure, methods
- Reusable

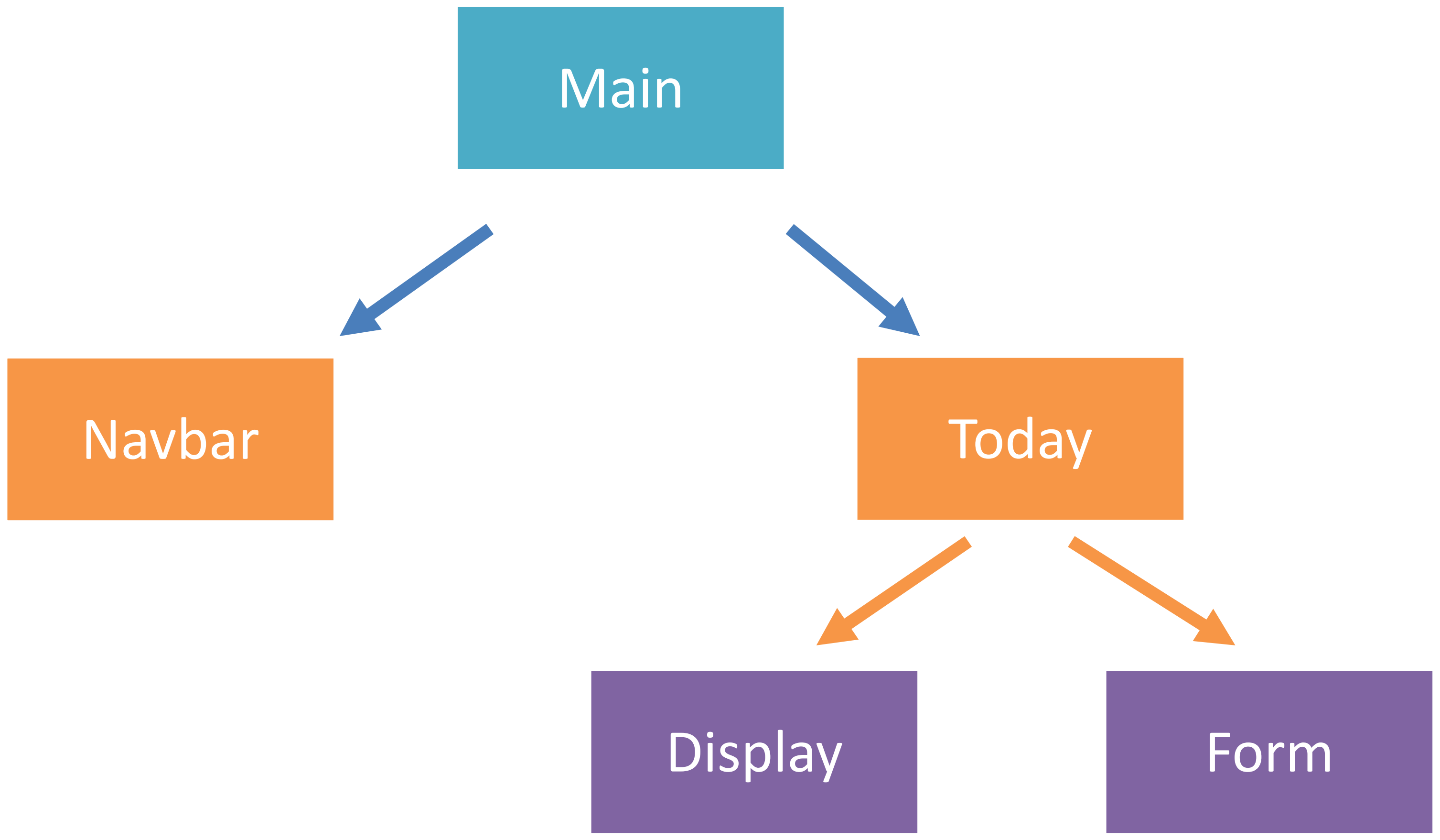
Components



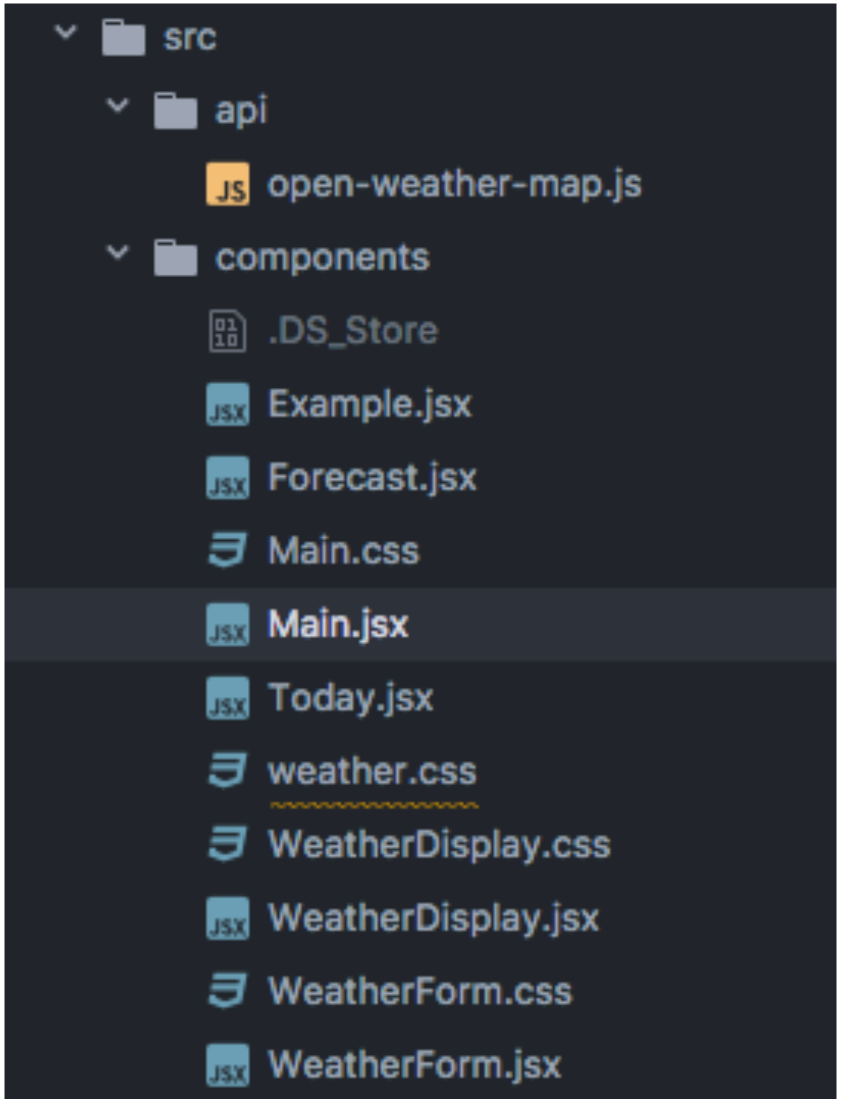
Components


Navbar
WeatherMood
Today
DataLab
WeatherDisplay
Few Clouds
24°c
Today / Forecast
Main
Hsinchu
° C
Check
WeatherForm

Components

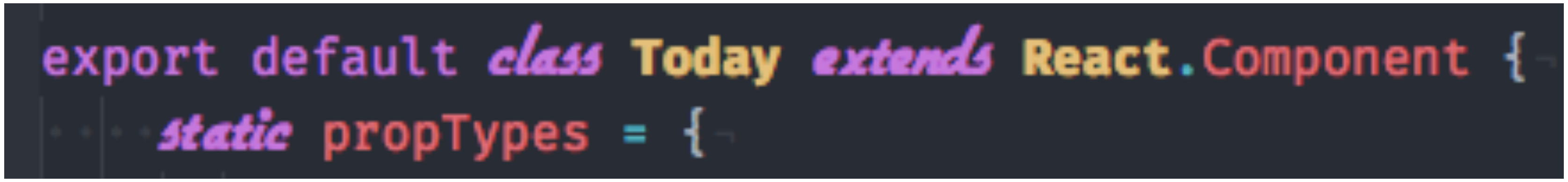


Components

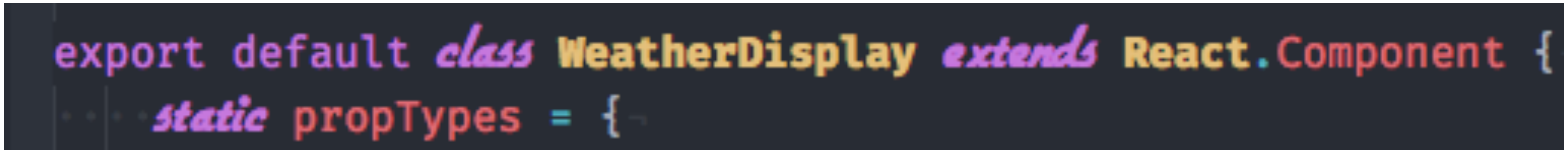


```
src
  api
    open-weather-map.js
  components
    .DS_Store
    Example.jsx
    Forecast.jsx
    Main.css
    Main.jsx
    Today.jsx
    weather.css
    WeatherDisplay.css
    WeatherDisplay.jsx
    WeatherForm.css
    WeatherForm.jsx
```

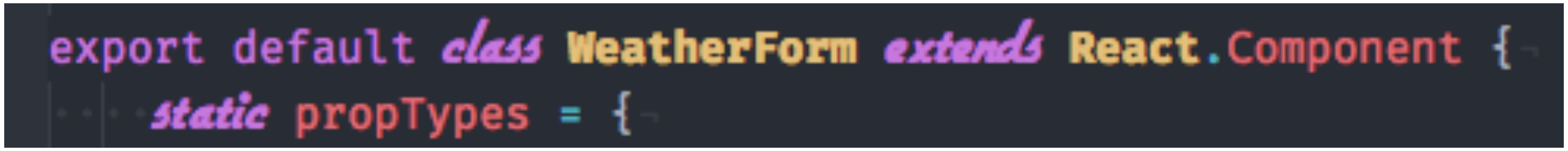
Components



```
export default class Today extends React.Component {
    static propTypes = {
```



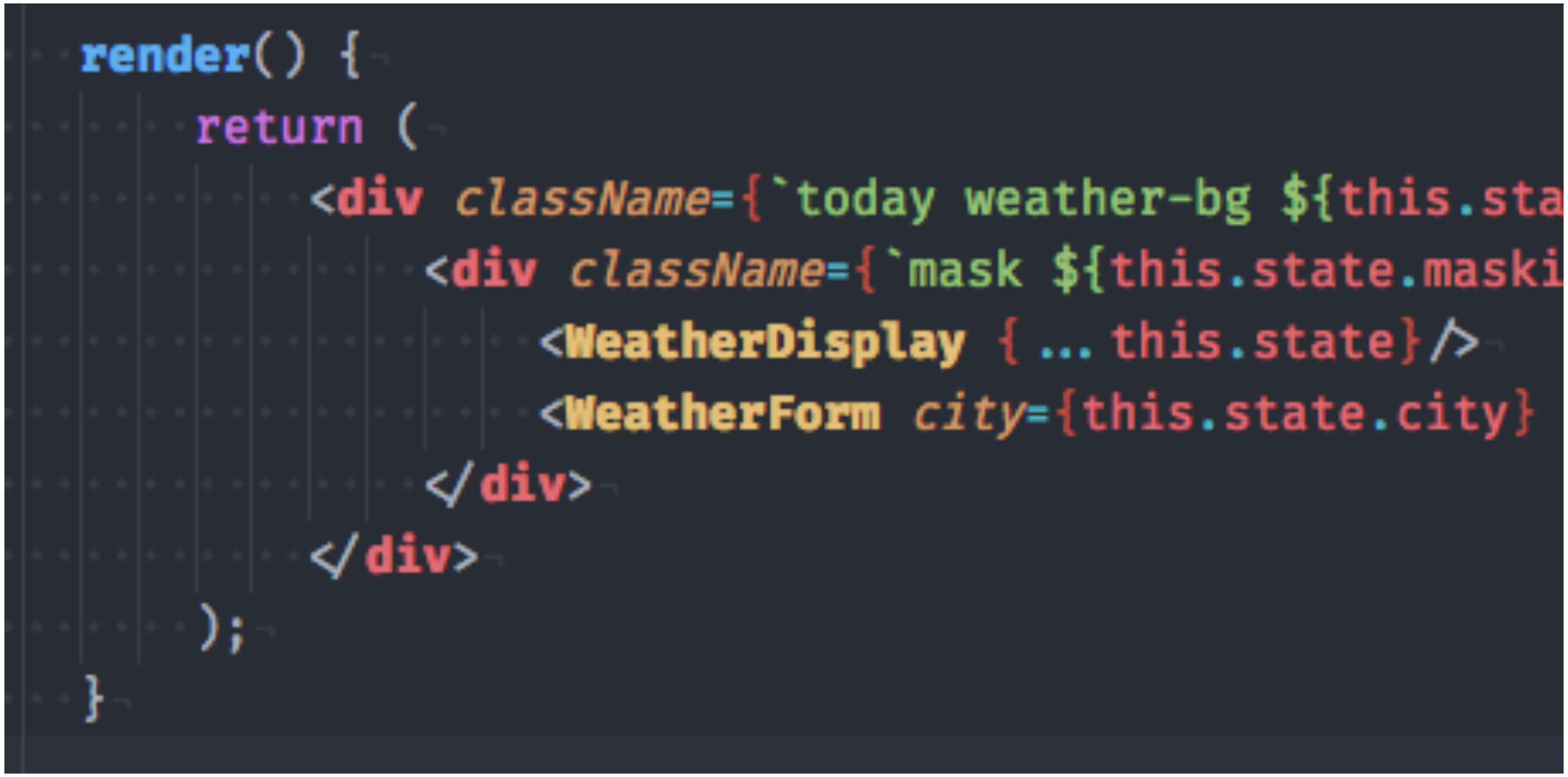
```
export default class WeatherDisplay extends React.Component {
    static propTypes = {
```



```
export default class WeatherForm extends React.Component {
    static propTypes = {
```

Components

- Inside Today.jsx:



```
render() {
    return (
        <div className={`today weather-bg ${this.sta
            <div className={`mask ${this.state.maski
                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city}
            </div>
        </div>
    );
}
```

Think about

How to communicate between components ?

by David Blackwell

Solution

- Props
- State

🤔

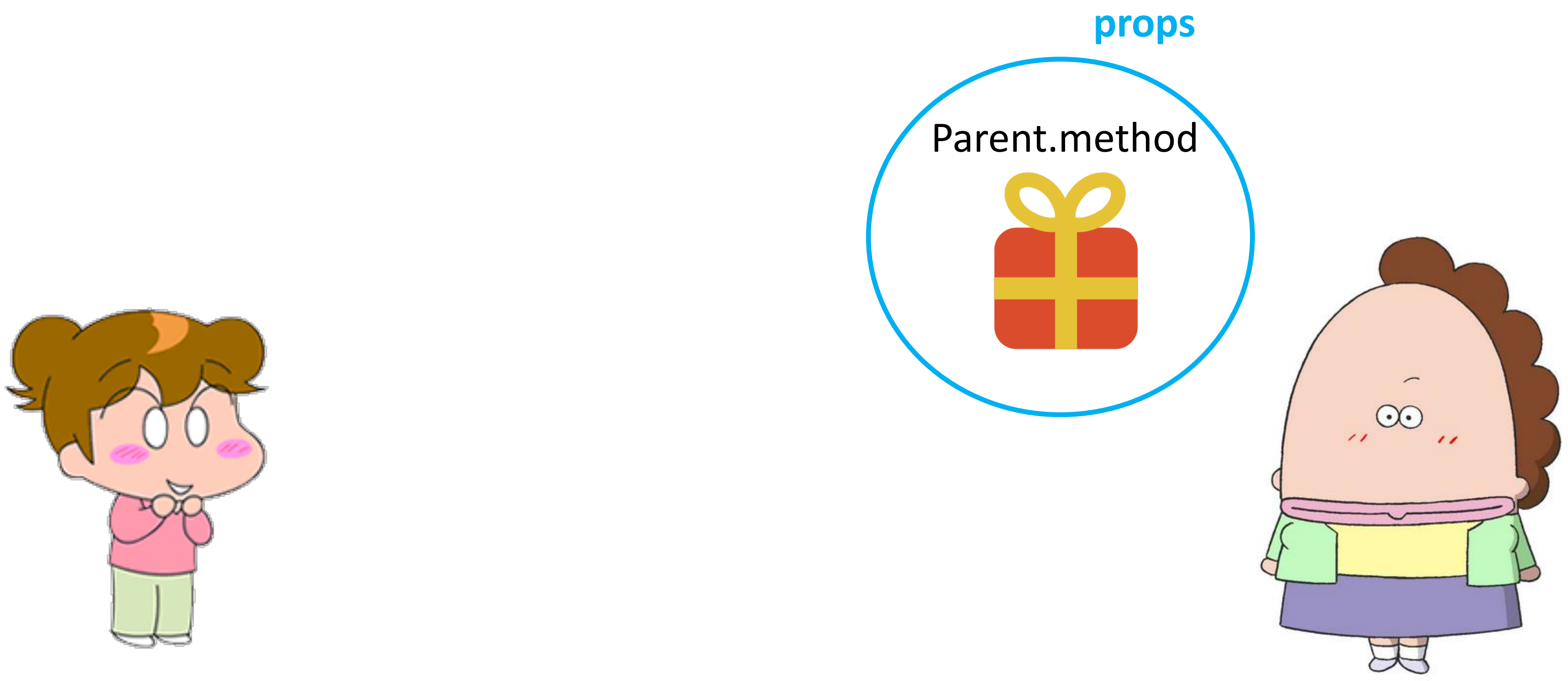
Props

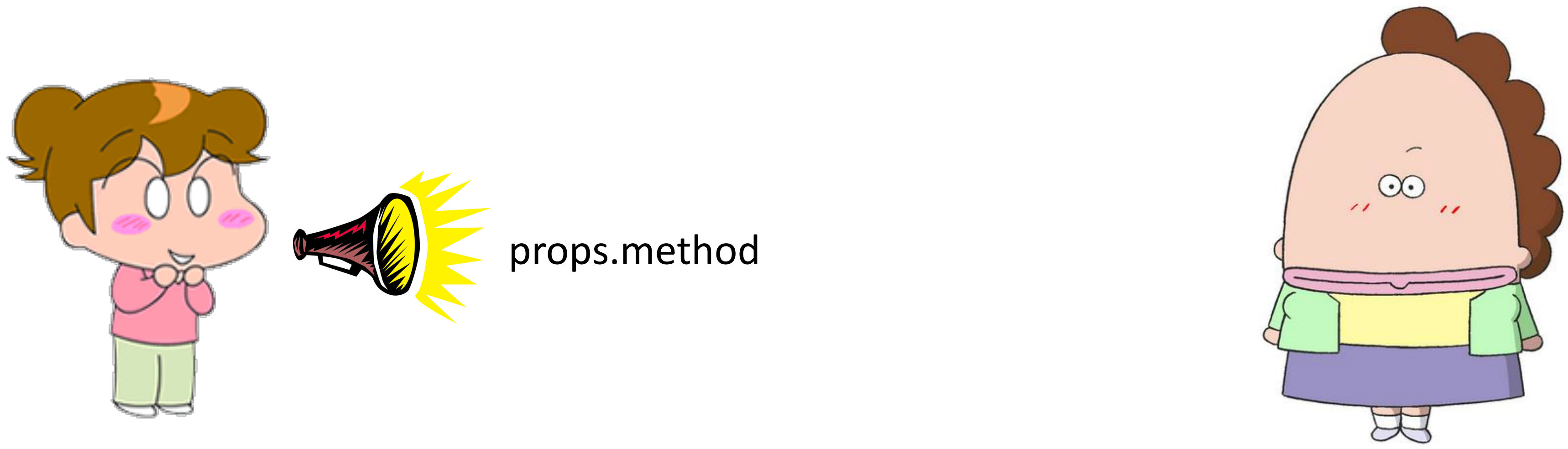
- Read only
- Parent to Child’s property

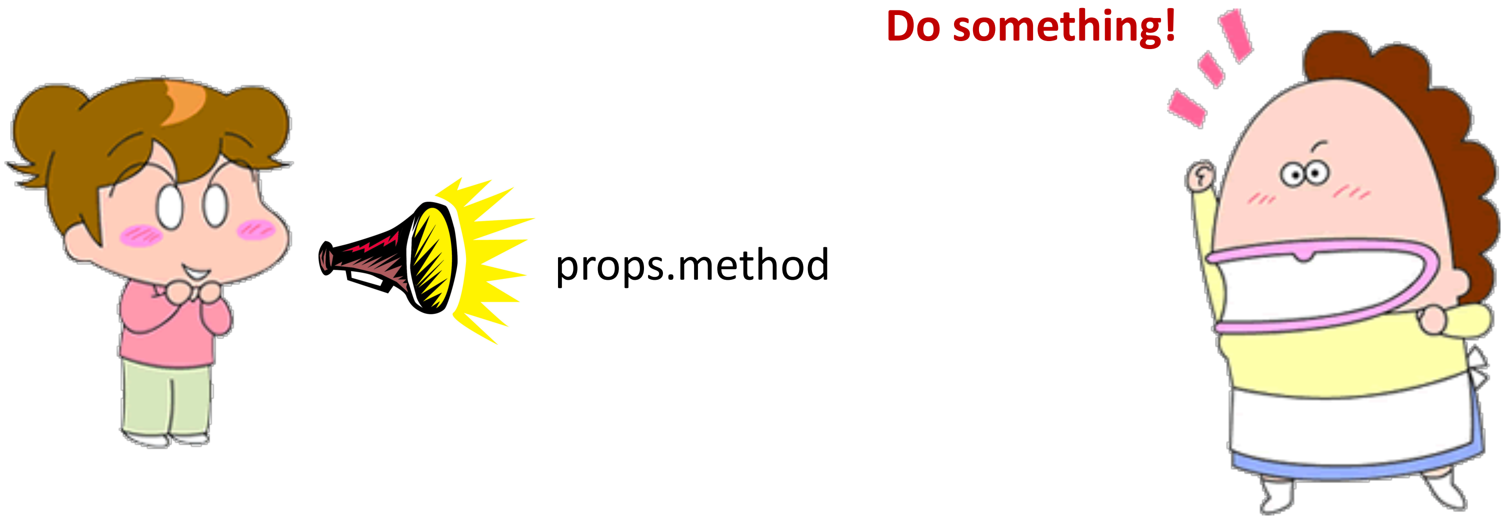
```
ReactDOM.render(
  <div>
    <Component name='Alice'>
    <Component name='Bob'>
  </div>,
  document.getElementById('root');
);
```

```
class Component extends React.Component {
  constructor(props) {
    super(props); ...
  }
  render() {
    return <h2>This is {this.props.name}<h2>;
  }
}
```

Now, we can let a child component talk to its parent


props
Parent.method

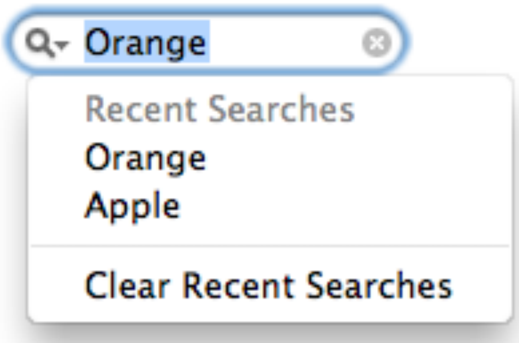

props.method


props.method
Do something!

How about a parent talking to its child?

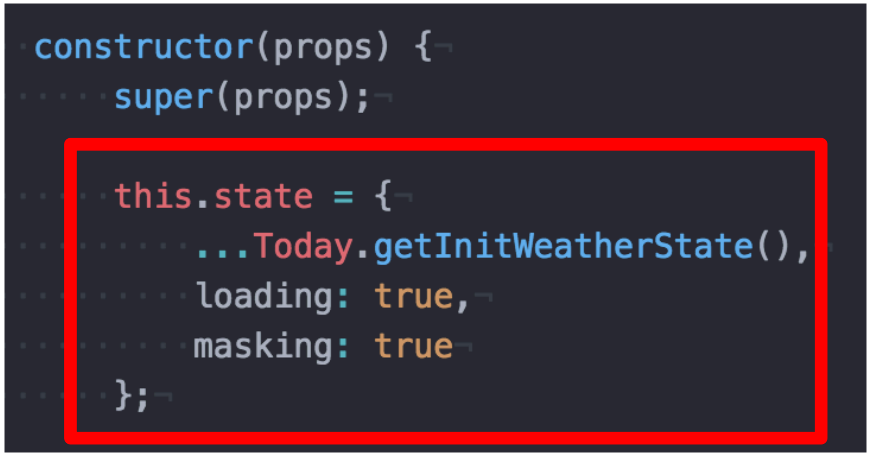
State

- A component may have its own ***states***
- Keep track of e.g., user input or program status

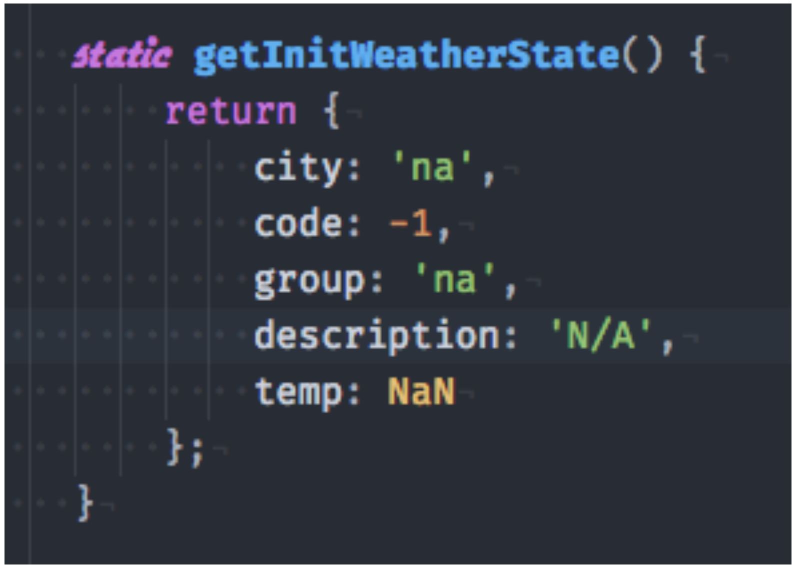


State

- Can be modified



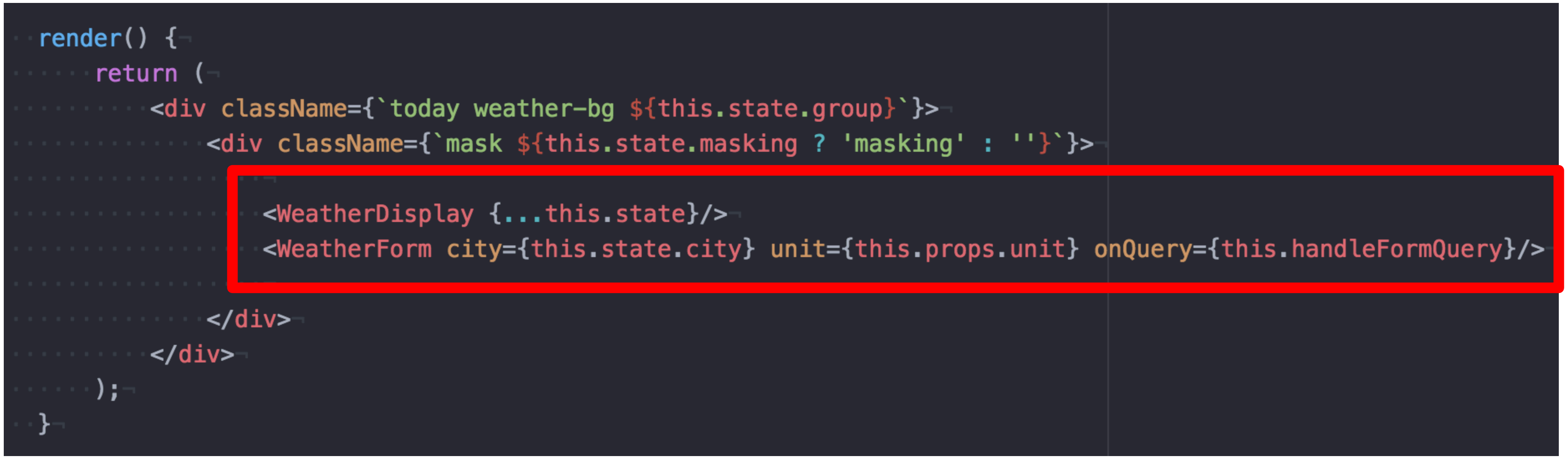
```
constructor(props) {
    super(props);

    this.state = {
        ...Today.getInitWeatherState(),
        loading: true,
        masking: true
    };
```



```
static getInitWeatherState() {
    return {
        city: 'na',
        code: -1,
        group: 'na',
        description: 'N/A',
        temp: NaN
    };
}
```

State

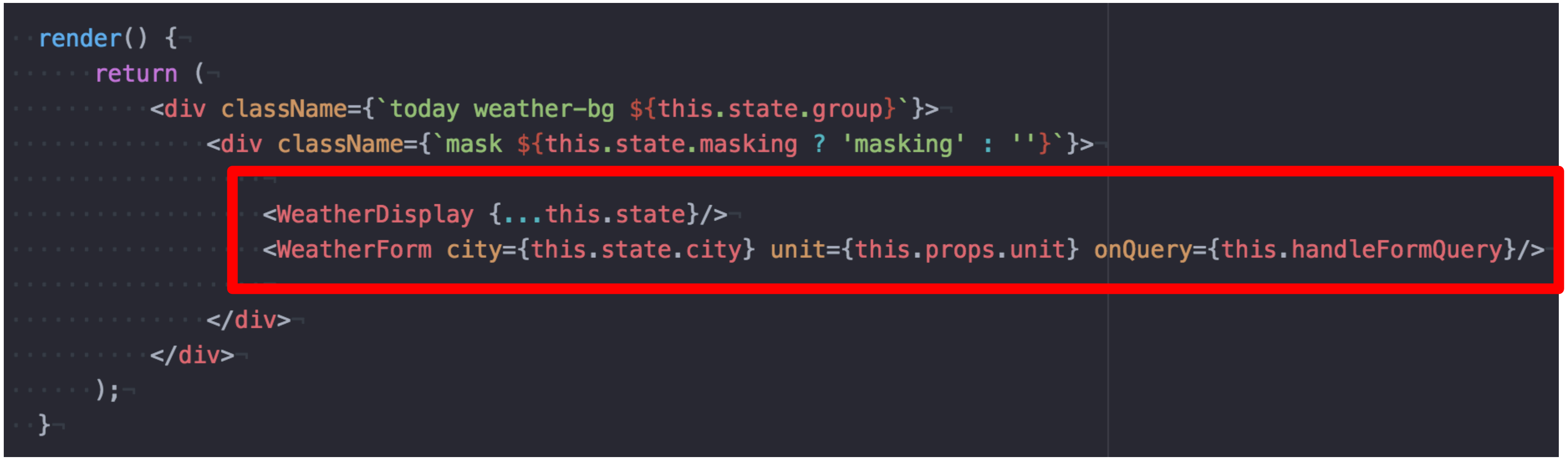
- Can be modified



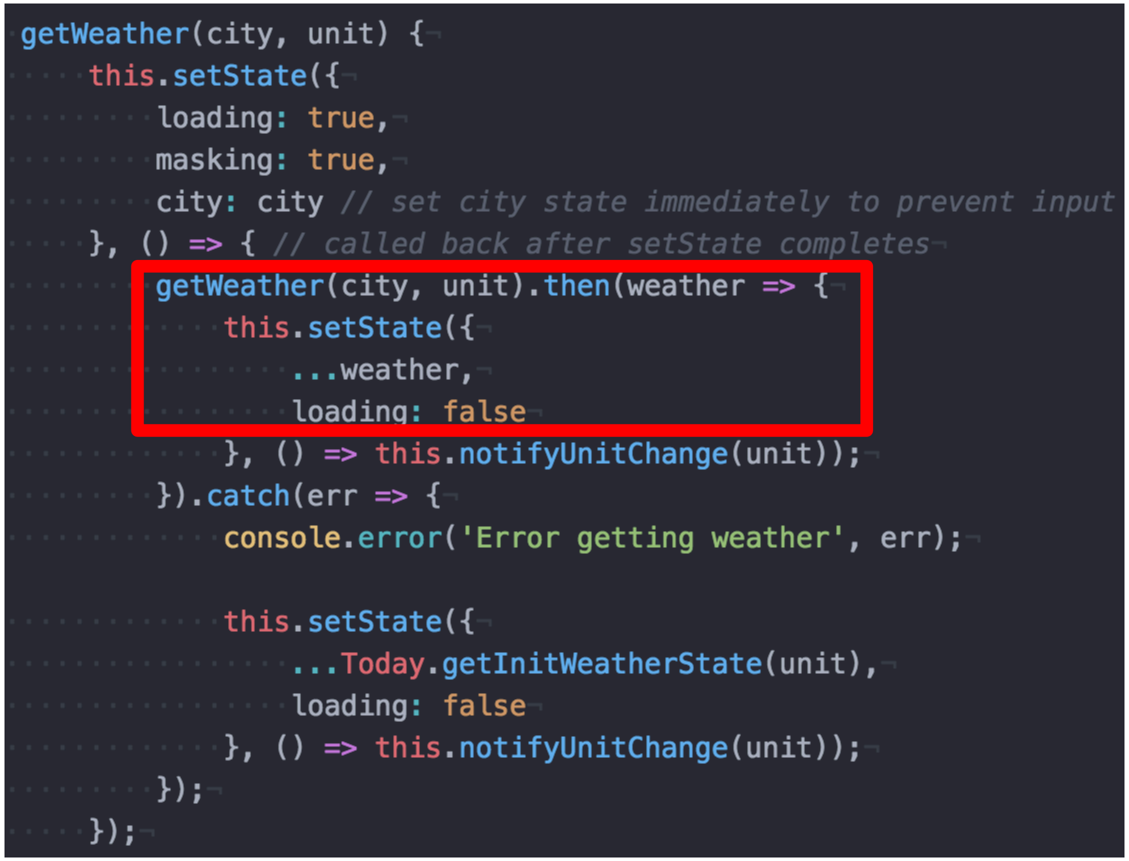
```
  render() {
      return (
          <div className={`today weather-bg ${this.state.group}`}>
              <div className={`mask ${this.state.masking ? 'masking' : ''}`}>

                  <WeatherDisplay {...this.state}/>
                  <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>

              </div>
          </div>
      );
  }
```

State

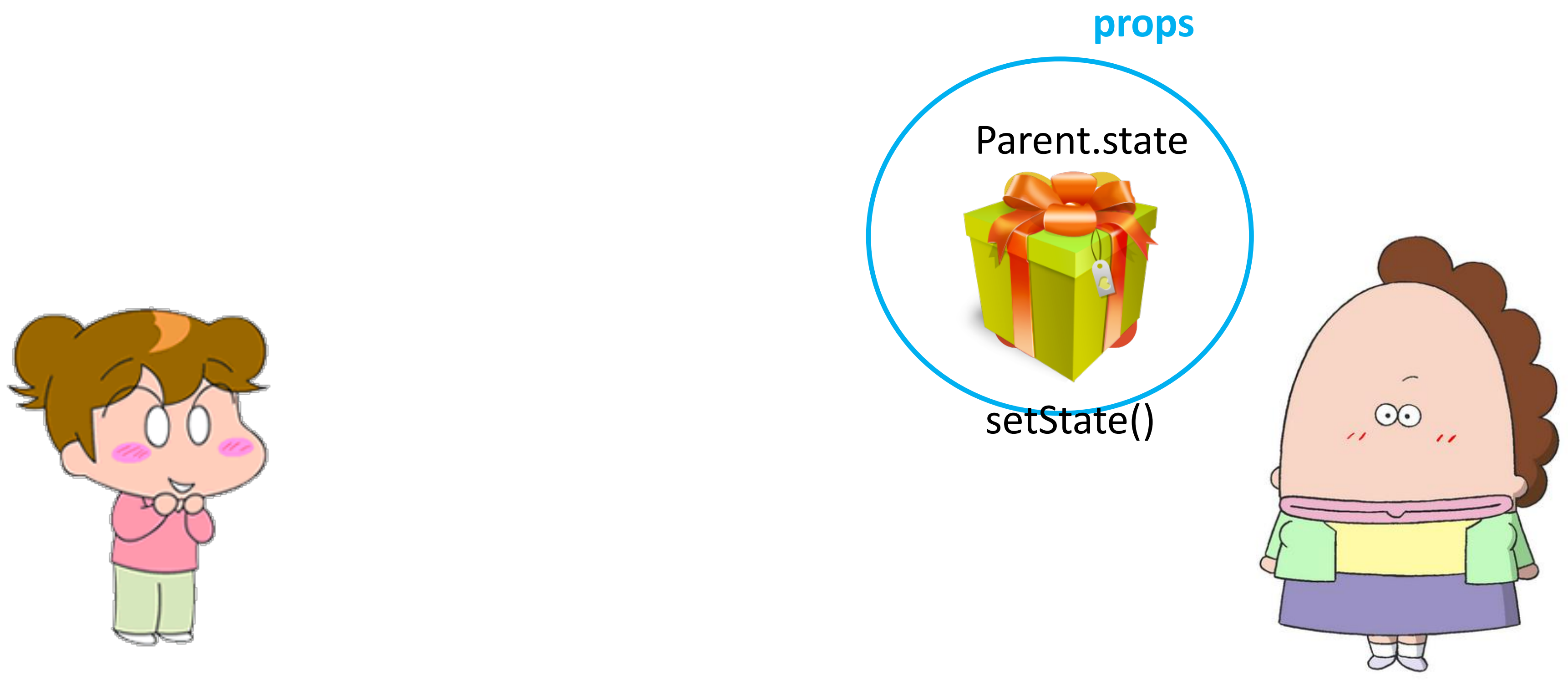
- Can be modified
- If state is used in render(), when you call setState(), it will trigger re-render automatically.



```
render() {
    return (
        <div className={`today weather-bg ${this.state.group}`}>
            <div className={`mask ${this.state.masking ? 'masking' : ''}`}>

                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>

            </div>
        </div>
    );
}
```



```
getWeather(city, unit) {
    this.setState({
        loading: true,
        masking: true,
        city: city // set city state immediately to prevent input
    }, () => { // called back after setState completes
        getWeather(city, unit).then(weather => {
            this.setState({
                ...weather,
                loading: false
            }, () => this.notifyUnitChange(unit));
        }).catch(err => {
            console.error('Error getting weather', err);

            this.setState({
                ...Today.getInitWeatherState(unit),
                loading: false
            }, () => this.notifyUnitChange(unit));
        });
    });
```



Trigger re-render!!

Data Flows

- Downward:

```
// in Main.render()
<Component count={this.state.count}>

// in Component.render()
<h2>Countdown: {this.props.count}</h2>
```

Data Flows

- Upward:

```
// in Main.render()
<Component onReset={this.handleReset}>

// in Component.render()
<button onClick={this.props.onReset}>Reset</button>
```

Component Life Cycle

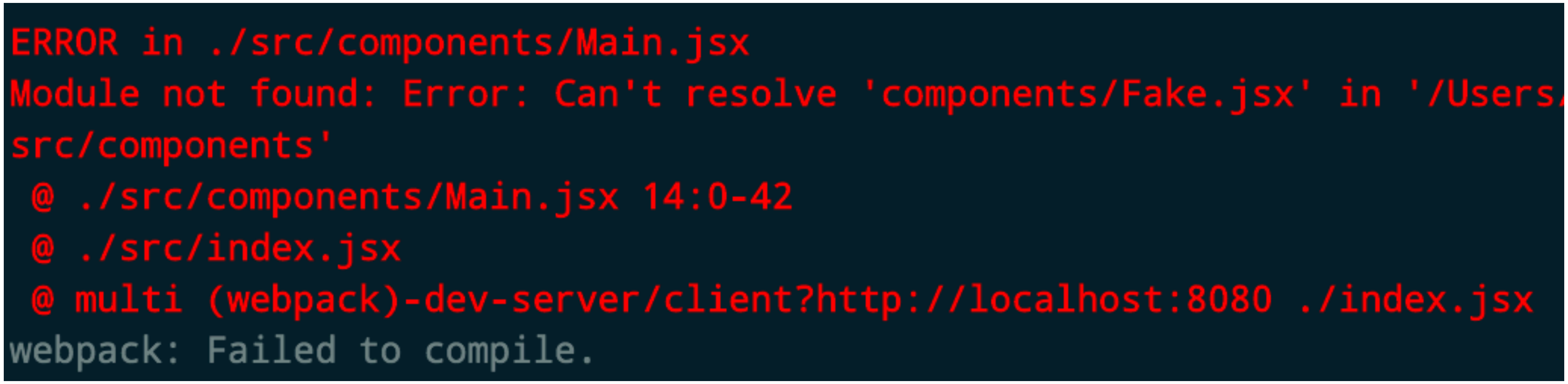
- Life Cycle

Thinking in React

Thinking in React

- Component Hierarchy
- Data Flow
- Identifying States
- Lifting states to common ancestors

Syntax errors during compiling

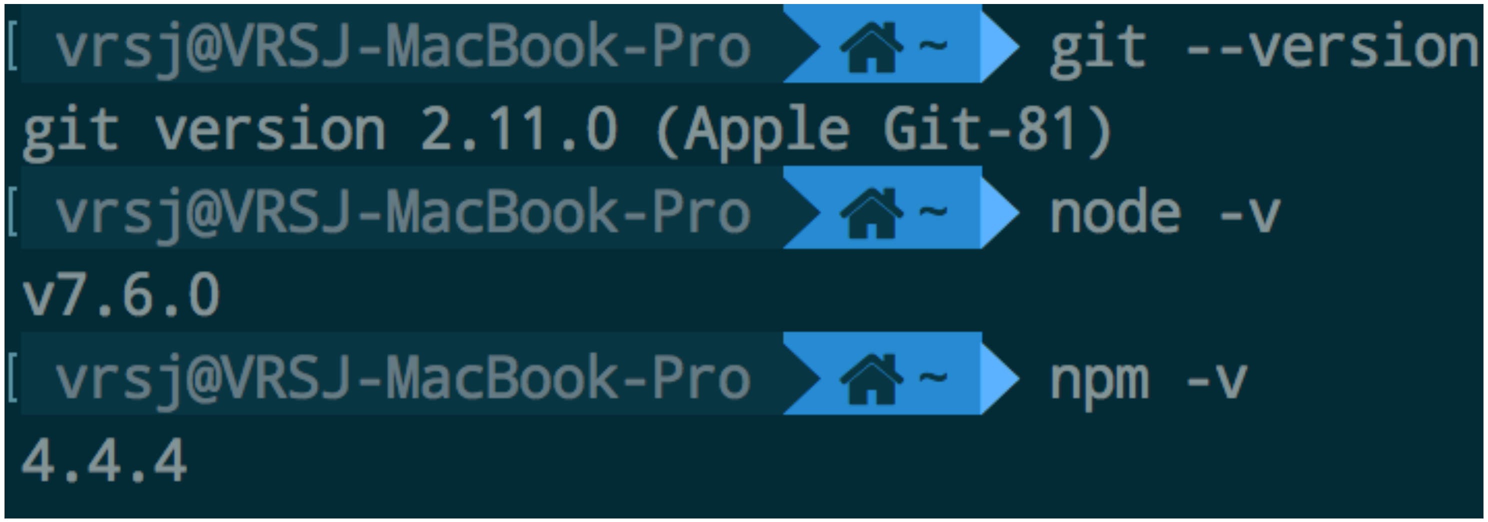


```
ERROR in ./src/components/Main.jsx
Module not found: Error: Can't resolve 'components/Fake.jsx' in '/Users/
src/components'
 @ ./src/components/Main.jsx 14:0-42
 @ ./src/index.jsx
 @ multi (webpack)-dev-server/client?http://localhost:8080 ./index.jsx
webpack: Failed to compile.
```

Hands on

Hands on

- Install git
- Install npm & nodejs



```
[ vrsj@VRSJ-MacBook-Pro  ~  git --version
git version 2.11.0 (Apple Git-81)
[ vrsj@VRSJ-MacBook-Pro  ~  node -v
v7.6.0
[ vrsj@VRSJ-MacBook-Pro  ~  npm -v
4.4.4
```

Hands on

- New app: React installation

Hands on

- Clone the example React Project that is on the website

```
$ git clone https://github.com/rolisanchez/example-react-weathermood
```

Hands on

- Run the example project:

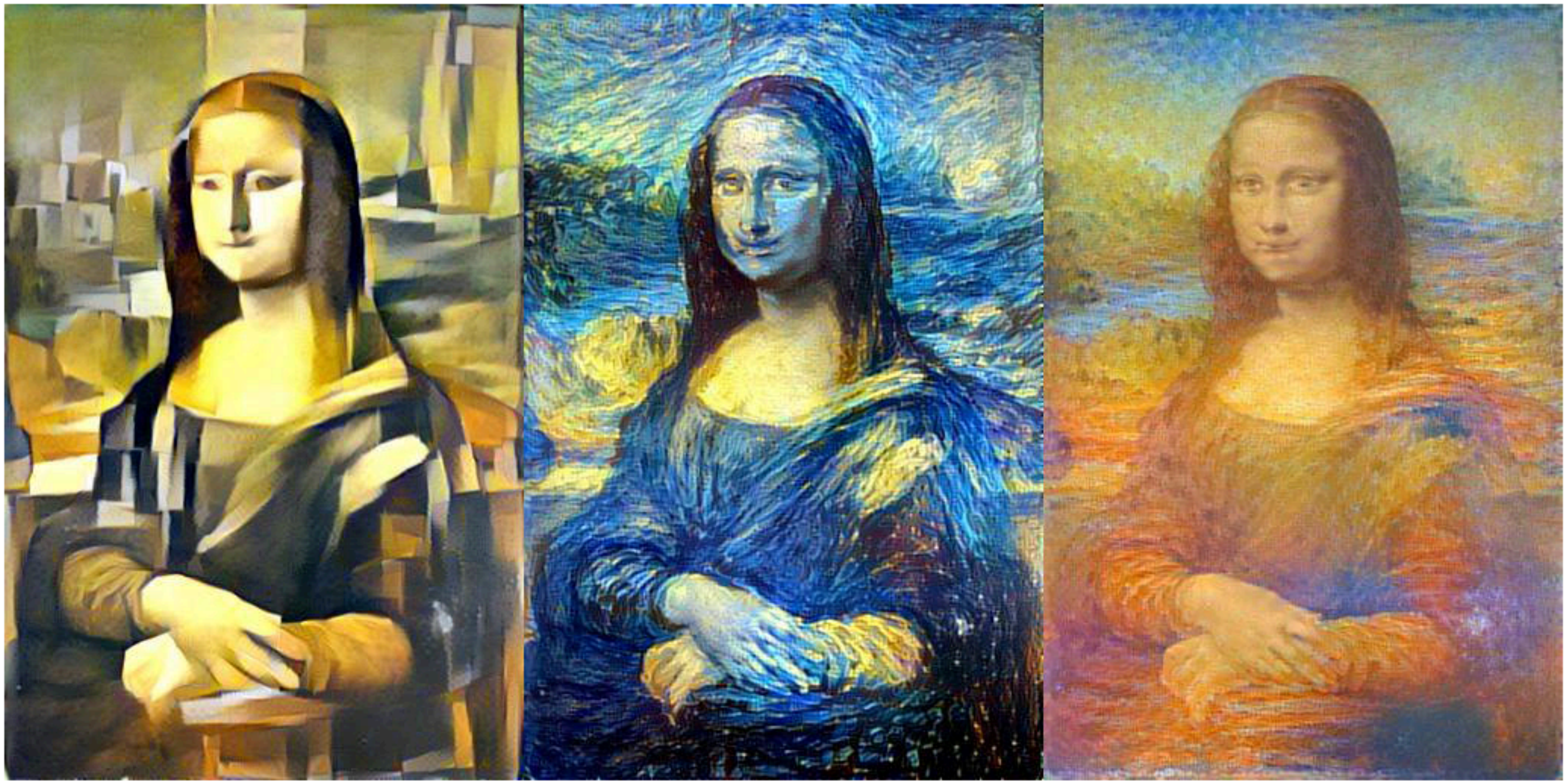
```
$ npm install
$ webpack
$ npm start
```

Hands on

- Take a look at the Style Transfer example app

Hands on

Mona Lisa restyled by Picasso, van Gogh, and Monet.



Hands on

- Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style."
- Deep Neural Networks?

Challenge

Using the React Base Project:

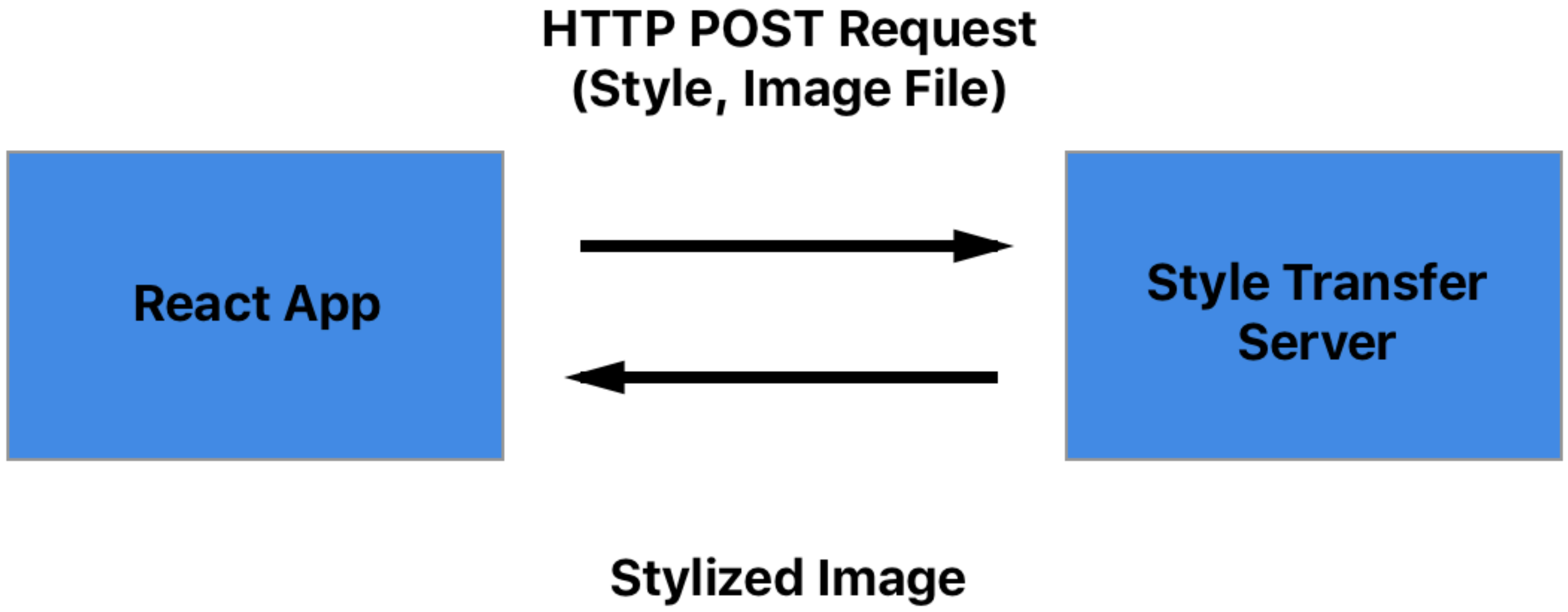
- Create a React Application that communicates with the Style Transfer Server

Challenge

Same procedure to run the project:

```
$ npm install
$ webpack
$ npm start
```

Challenge


HTTP POST Request
(Style, Image File)
React App
Style Transfer Server
Stylized Image

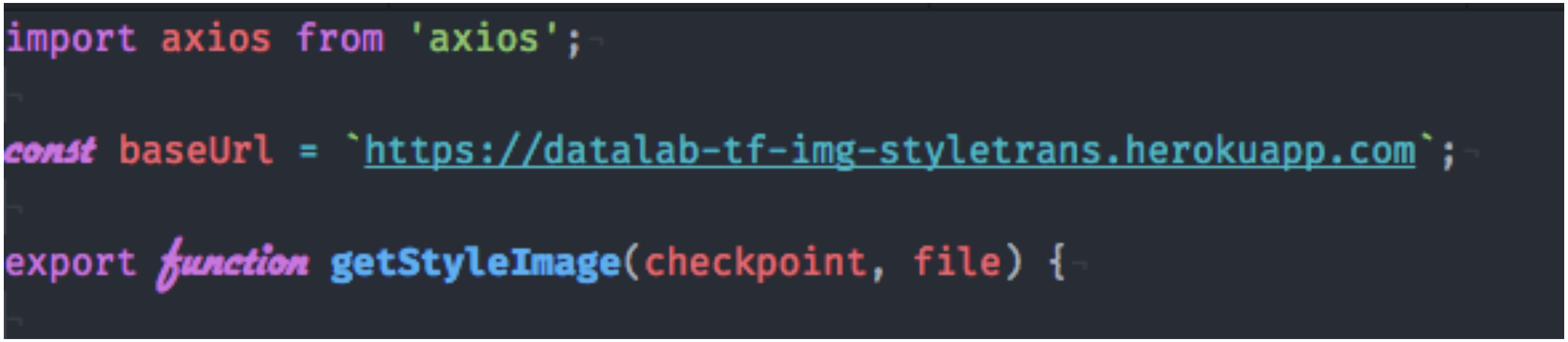
Challenge

Submitted form needs these fields:

- checkpoint: which style you want (already on the form)
- file: The image to be stylized

Challenge

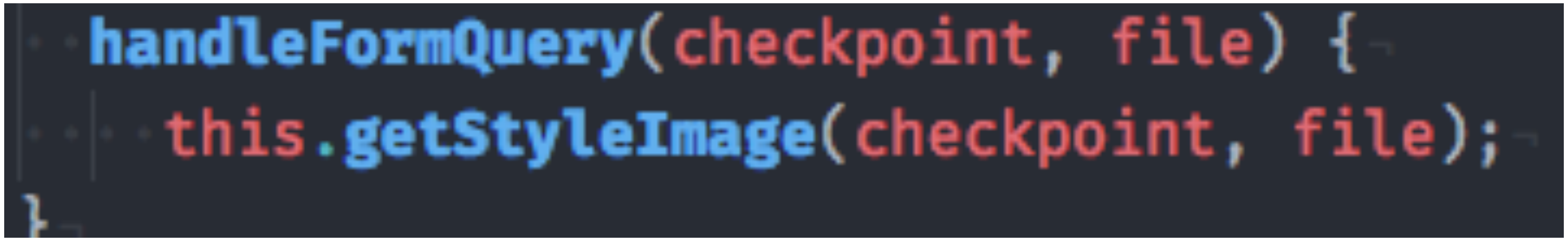
Get style image API (get-style-image.js):



```
import axios from 'axios';

const baseUrl = `https://datalab-tf-img-styletrans.herokuapp.com`;

export function getStyleImage(checkpoint, file) {

```

Challenge

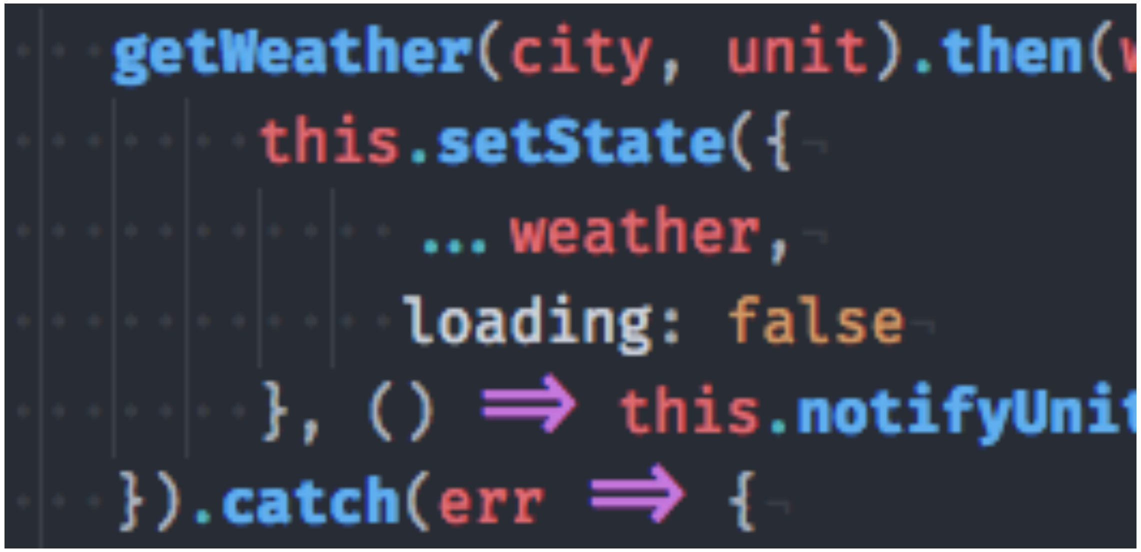
Handling the query (Style.jsx):



```
  handleFormQuery(checkpoint, file) {
    this.getStyleImage(checkpoint, file);
}
```

Challenge

Take a look at getWeather from example project:



```
  getWeather(city, unit).then(
      this.setState({
          ...weather,
          loading: false
      }, () => this.notifyUni
  }).catch(err => {
```

Challenge

Resources:

- Axios HTTP
- React Forms
- StackOverflow - ReactJS

Thank you
Gracias
謝謝