Framework Training
React
London July 2018
Exercise B-component

- This exercise creates a component which uses **map** to **iterate** over an array of objects.

Installation

- Install the starter version of the project.
- Open the terminal at the correct folder.

```
npm install
npm start
```

Plain JS

- We can use plain Javascript to turn an array of strings into HTML markup. Reviewing this code will help gain an understanding of how React works.
- Review the code in **help/turn-array-into-markup.js**.

Create a React component

- **Cities.js** defines a module which exports an array of objects.

```
let cities = [
{ name: "Seville", temp: 88.. },
{ name: "Trujillo", temp: 64.. }
];
```

- We will create a Spain component to display this data, and then create an instance of the component in index.js.
- In **Spain.js**, import React and its component class.

```
import React from "react";
import { Component } from "react";
```

- Import the CSS and the data.

```
import "./App.css";
import { cities } from "./cities";
```

- To create a minimal React component, it must contain one render method.
- That method must return one top level HTML element.

```
class Spain extends Component {
    render() {
        return <section>Spain</section>;
    }
}

export default Spain;
```

- A component instance can be created in index.js.
- Import the component definition.

```
import Spain from "./Spain";
```

- The instance needs to be injected into a specific element in the DOM in public/index.html:

```
    <section class="page"></section>
```

- In index.js store a reference to that element into a variable.

```
    let el = document.querySelector(".page");
```

- Use React to render an instance of the Spain component into that element.

```
    ReactDOM.render(<Spain />, el);
```

- The component instance should appear on the page.
- Create a heading in the render method and style it. Note the React JSX syntax uses className to avoid the reserved word class.

```
return (
    <section className="holiday">
        <h1>Spain</h1>
    </section>
);
```

Iterating over the cities data using map()

- We can use the Javascript **map function** to iterate over the cities array and apply a function to each element.

```
    import { cities } from "./cities";
    cities.map(city => console.log(city.name, city.region));
```

- We can use map and JSX inside the render method to create markup.

```
{cities.map(city => <p className="city">{city.name}</p>)}
```

- This works but generates a React error. React wants unique key attributes on each iterated element.
- Add a key attribute using the ID property in each city.
- Add the region as a SPAN within the paragraph.

```
        {cities.map(city => (
            <p key={city.id} className="city">
                {city.name}
                <span>{city.region}</span>
            </p>
        ))}
```

- Add a population property to each object. Display the population and temperature for each city

Review

- The Spain component is not **pure**: it relies on the external local variable cities being in scope.
- We have not defined any way of passing data into the component, so that it can be reused with different cities.

Stateless version

- The Spain component does not have any internal state, so it could be defined as a stateless ES6 arrow function.

```
let Spain = () => {

    return (
        <section className="holiday">
            <h1>Spain</h1>

            {cities.map(city => (
                <p key={city.id} className="city">
                    {city.name}
                    <span>{city.region}</span>
                </p>
            ))}
        </section>
    )
```

```
};
```