Framework Training
React
London July 2018
Exercise D-props

- This exercise turns the Spain component into a more reusable Nation component.
- Data is passed into **multiple instances** of the Nation component as **React props**.

Installation

- Install and run the starter version of the project.

```
npm install
npm start
```

Review the starter version of the project

- Data for Spanish and Japanese cities is imported into the main index.js file.

```
import { es } from "./cities/spain";
import { jp } from "./cities/japan";
```

- public/index.html contains two sections where the component instances will be injected.

```
<section class="spain"></section>
<section class="japan"></section>
```

- Create two React component instances in index.js

```
let spain = document.querySelector(".spain");
let japan = document.querySelector(".japan");

ReactDOM.render( <Nation/> , spain );
ReactDOM.render( <Nation/> , japan );
```

- We can pass the data into the component instances as props.

```
ReactDOM.render( <Nation country={es}/> , spain );
ReactDOM.render( <Nation country={jp}/> , japan );
```

- In the Nation component, props are passed in via the constructor.

```
    constructor( props ) {
    super( props );
    console.log( this.props.country );
    }
```

- We can display the country name in the render method:

```
<h1>{ this.props.country.name }</h1>
```

- We can use destructuring to write more concise syntax:

```
let { name,cities } = this.props.country;
<h1>{ name }</h1>
```

- Remove the React comments to re-enable the map code:

```
{ cities.map .. }
```

- View the component instances in the React DevTools.

Stateless components

- Nation can be redefined as a stateless ES6 arrow function.

```
let Nation = ( props ) => {
    let { name,cities } = props.country;
    return ( <section> .. )
}
```