Framework Training
React
London July 2018
Exercise J-lifecycle

- This exercise reviews a working example to understand how the React Component **Life Cycle** works.

Installation

- Install and run the starter version of the project.

```
npm install
npm start
```

Review the existing project

- **Index.js** file contains a Demo component instance.
- Its **render** method contains an instance of a Life component.

```
    <Life json={file}/>
<p onClick={this.changeData}>Change data</p>
```

- The **prop** named json contains a filename “spain–2017.json”
- Clicking on the adjacent button changes this filename.

The LIFE component life-cycle

- The first method to run is the **constructor**.
- It sets the component state.

```
this.state = { data: [], letters:[] }
```

- The **render** method is called for the first time.
- Once the component has been rendered on stage, the **componentDidMount** method is called.
- It calls getJson() to load data using Fetch.

```
this.getJson("componentDidMount");
```

- When Fetch returns data, it updates state:

```
this.getJson("componentDidMount");
```

- When the state changes, the **render** method is called.
- The **componentDidUpdate** method is also called when the component updates.

Change data

- If the user clicks the **Change Data** button, it flips the order of this array of filenames in the state of the Demo component

```
json:[ "spain-2017.json","spain-2018.json" ]
```

- When the state changes its render method is called again.

```
<Life json={file}/>
```

- Because the value of the prop has changed, the **componentDidUpdate** method in the Life component is called.
- It compares old versus new props. If they have changed it calls getJson to load new JSON data.

```
if(prevProps.json !== this.props.json) {
    this.getJson("componentDidUpdate");
}
```

- **getJson** loads the data and changes component state, which triggers another call to **render**.
- Review the Life Cycle using the browser console and React DevTools.