Framework Training
React
London July 2018
Exercise K-keys

- **Keys** are used in React when iterating over arrays of data using map.
- This exercise examines potential problems with using keys.

Installation

- Install and run the starter version of the project.

```
npm install
npm start
```

Review the project.

- This project creates a TO DO list.
- The list is held in component **state** as an array of objects.

```
[ { desc:"Large apples"}, { desc:"Small pears"} ]
```

- When the user clicks the **add-item** button, a new object is pushed into the **end** of the array.
- This works but this is a **run-time error** in the browser console.

```
Each child in an array or iterator should have a unique "key"
prop.
```

- React uses **keys** to identify changes in groups of related items.
- Here we map over the list. The error occurs because we need to give each list item (LI) a **unique key**.

```
<ul>{ props.list.map((item,n) =>
    <li><label>{item.desc}</label>
    <input type="text"/></li> )}
</ul>
```

- Map passes in two arguments: the item and an index. We can use this index as the key.

```
    <li key={n}>
```

- This approach clears the error message, but using this index value is **problematic**.

Problems with the item index as a key

- To demonstrate the problem, we will change the code to insert new items at the **start** of the list in method addItem.

```
list.unshift( this.createItem());
```

- Test this new version.
- Add some items, then edit the first input field.
- When you then add further items, the first input field **falls out of sync** with its associated label.
- We are using index keys that are numbered from 0 to N.
- When we insert a new item at the front of the list, we replace element with a key of 0 with another element with a key of 0.
- At this point React fails to reorder the elements correctly.

Unique keys

- We can avoid bugs with keys by using **unique** non-repeating IDs for each item.
- Add a method which creates unique keys based on the item desc.
- This method uses a regular expression to create a four letter code from the description.
- So “Organic large English plums” becomes “olep”

```
    createId = desc =>
desc.match(/\b(\w)/g).join("").toLowerCase()
```

- This will only create 256 different keys so their is a chance of duplicates.
- We can add a random number to the key to make duplicates very rare.

```
        createId = desc =>
desc.match(/\b(\w)/g).join("").toLowerCase() + "-" +
Math.floor(Math.random()*1000000)
```

- This will create keys like **“lsfp–564512”**
- Update **createItem** to use this method.

```
    createItem = () => {
        let desc = this.props.getFruit();
        let id = this.createId(desc);
        return { desc:desc, id:id }
    }
```

- Use this unique key in the **render** method.

```
key={item.id}
```

- Test and confirm that this solves the bug.
- Review the code in React DevTools.