Framework Training
React
London July 2018
Exercise I-prop-types

- We pass arguments into React components using **props**.
- We can **validate** these props to ensure that component instances are used correctly.

Installation

- Install and run the starter version of the project.

```
npm install
npm start
```

Review the existing project

- The project displays two instances of a Nutrition component.

```
    <Nutrition name="broccoli" calories={28}
    protein={87} type="cruciferous"/>
    <Nutrition name="cucumber" calories={42}
    protein={56} type="marrow" />
```

- **PropTypes** deal with one common source of bugs: using components with the wrong/missing props.
- The **propTypes** property defines the correct type for each prop.
- The **defaultProps** property defines default values for these props.
- *These properties can be defined using two styles of syntax. This exercise uses static properties inside the class.*

PropTypes

- We import the **propTypes** React library.

```
import PropTypes from 'prop-types';
```

- We define propTypes as a **static** property within the class.

```
static propTypes = {}
```

- We can define the name prop as a string.

```
static propTypes = {
name : PropTypes.string
```

}

- If we pass a number-expression to the component instance this will trigger a run-time error.

```
<Nutrition name={45} .... />
Invalid prop `name` of type `number`
```

- Passing a correct string value will fix this.

```
<Nutrition name="broccoli" .. />
```

- We can make name a **compulsory** prop by adding **isRequired**.

```
name : PropTypes.string.isRequired
```

- Add a propType for the calories field.

```
calories : PropTypes.number
```

PropType functions

- We can perform **custom validation** by defining a function within the propTypes object.
- This function checks that the protein prop is in the range 0–100.

```
protein : (props, name) => {
    let n = Number( props[name]);
    return (n >=0 && n <=100) ? null : new Error("Protein range
0-100")
},
```

Range of allow numbers

- We can define an array of acceptable prop values for the type prop.

```
type: PropTypes.oneOf(
['leafy', 'cruciferous','root','marrow','allium'])
```

Default props

- We can define **default values** for component instances.

- Component instances can be created with certain props omitted.

```
    static defaultProps = {
        calories: 50,
        protein: 10
    }
```

- We can then create a second instance which uses the default values.

```
        <Nutrition name="cucumber" type="marrow" />
```