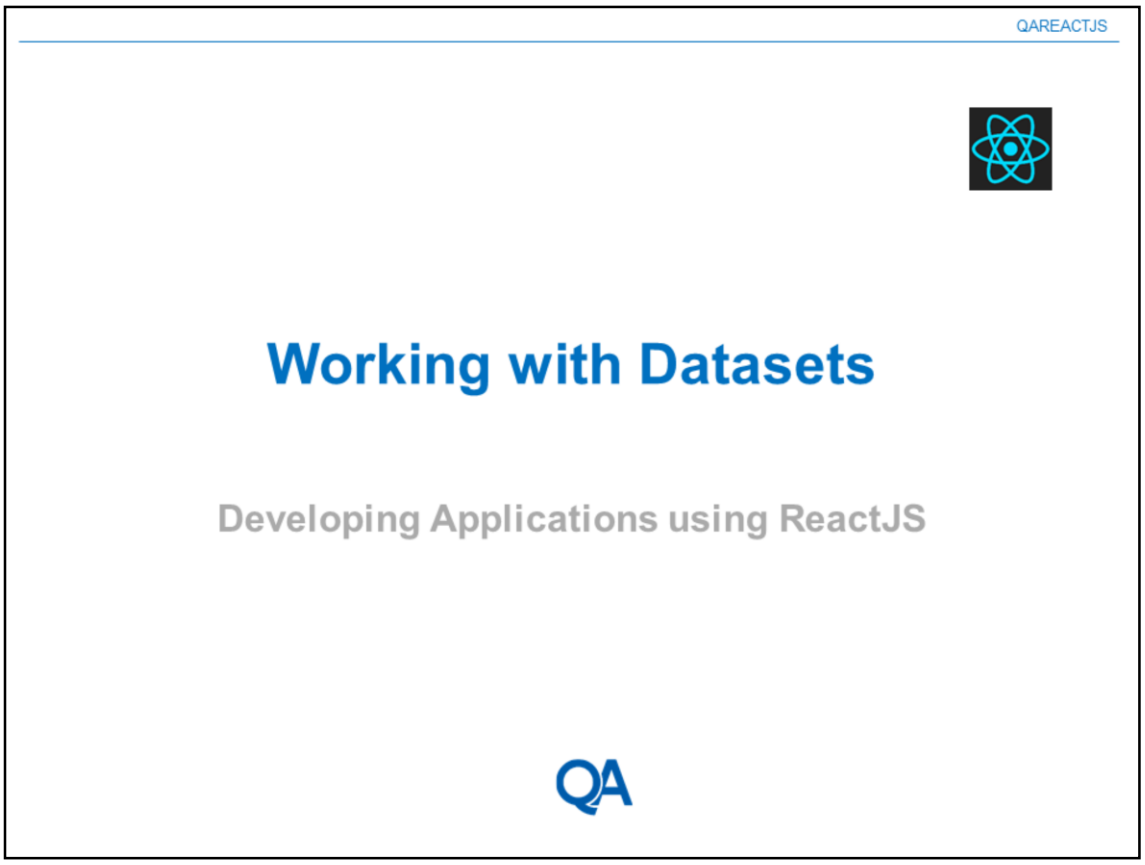

QAREACTJS
Working with Datasets
Developing Applications using ReactJS
QA

Objectives

- To be able to add to a dataset from within a web page
- To be able to read in a dataset to display on a page
- To be able to request and receive data from a RESTful service

Adding to a Dataset

- **Many applications require users to fill in a form**
- **The form data needs to be collected and then processed somewhere**
- **React can retrieve form data and then push it into an existing state**
  - This can then be outputted as JSON to wherever it is needed

Collecting Data from Forms

- **Data needs to be collected from the form before input loses focus**
  - It actually needs to be collected each time it changes
    - Otherwise it is difficult to make function elegantly
- **State in the component containing the form can be used to hold the data until it is passed back to the application**
- **Each input element needs to call a 'handleChange' function, passing in its name so that its value can be retrieved and saved**

```
…
    <input type=text name="myName" value={this.state.myName}
       onChange={this.handleChange("myName").bind(this)} />
…
```

- **An anonymous function can then be used to set the state as required using the valueName passed in**

```
handleChange(valueName) {
  return (event) =>
      { this.setState({ [valueName]: event.target.value });};
}
```

Collecting Data from Forms

- **Once the data has been collected on the form, the onSubmit event can handle passing data back to the component's parent**

```
…
      <form id="addMyName"
        onSubmit={this.handleSubmit.bind(this)} >
…
```

- **The function handleSubmit also needs to be in the form component's class – it calls the onSubmit in the parent**
  - Current state is passed in and then reset to clear the form

```
handleSubmit(event) {
        event.preventDefault();
        this.props.onSubmit(this.state);
        this.setState({
                myName: ''
        });
}
```

Collecting Data from Forms

- **In the parent component, the form's tag and an onSubmit should be used to call the submit handler in the parent**
  - The arguments that were provided in the form component are automatically passed through the the handler

```
…
     <FormComponent
    onSubmit={this.handleSubmitInParent.bind(this)} />
…
```

  - This will place the FormComponent in the app and ensure that its submission is handled
- **handleSubmitInParent() can then handle the data**
  - In this example, the new data is pushed into an existing array and then output to localStorage as JSON
  - Anything that JavaScript can do with data can now be done!

Collecting Data from Forms

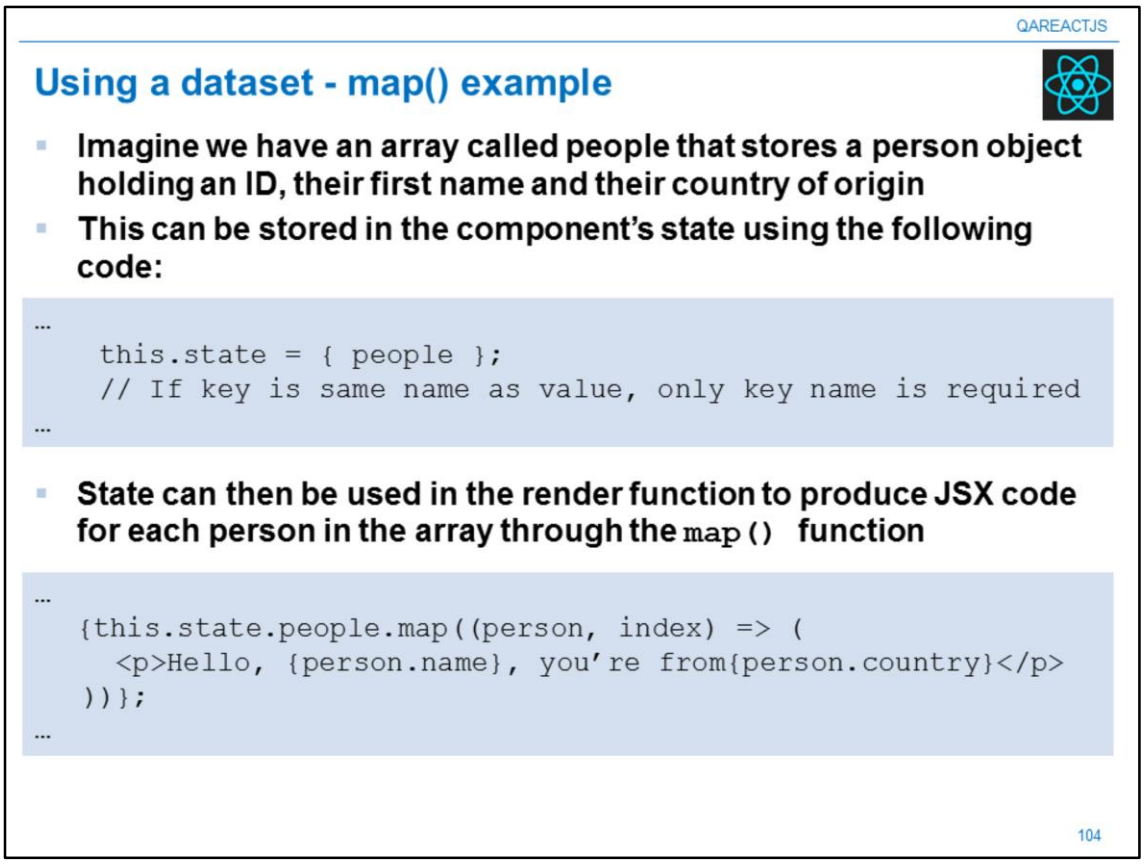
- **handleSubmitInParent() code:**

```
handleSubmitInParent(dataFromForm) {
       let array = this.state.array;
       array.push( {
              id : array.length,
              name: dataFromForm.myName
       });

       this.setState({array});
       localStorage.array = JSON.stringify(array);
}
```

Data Sources

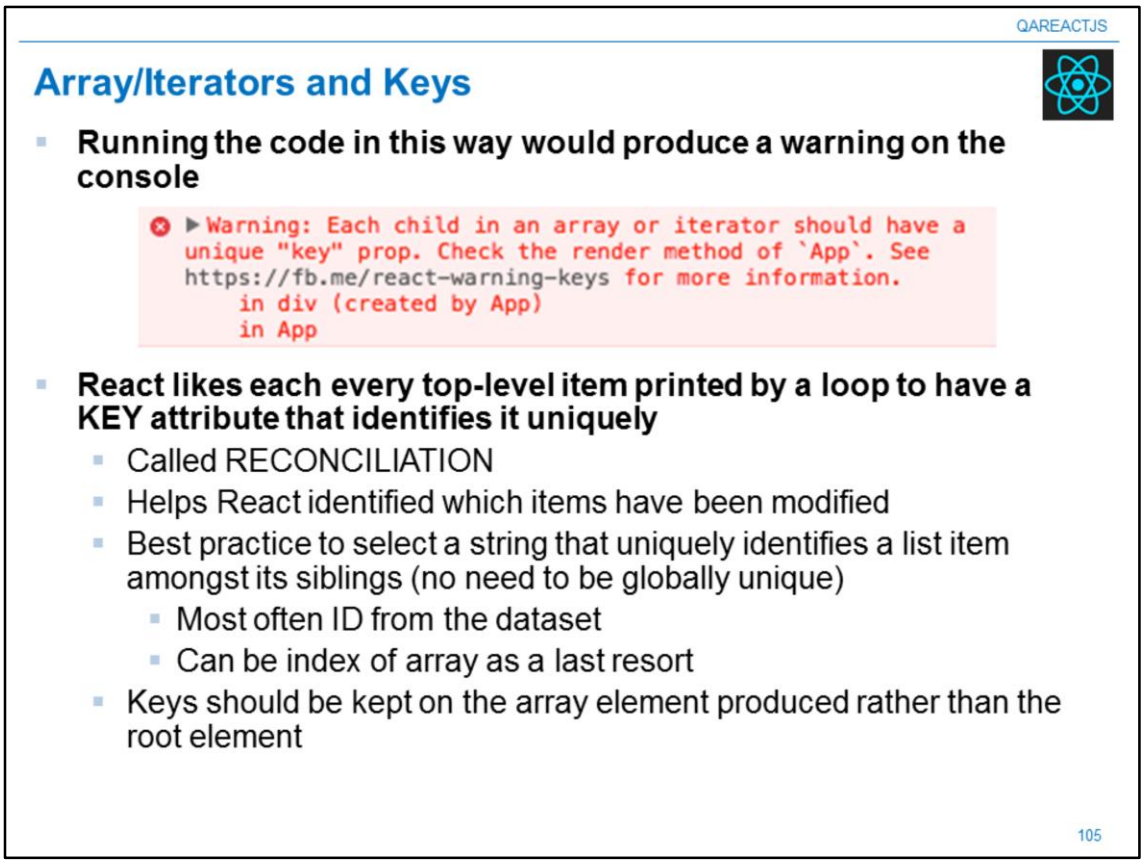
- **React is for producing front end components**
- **Need to be able to convert data from data sources into state and/or props**
  - E.g. Displaying a list of items in a table from a HTTP search response
- **Can convert data source into or create an array**
  - Array can then be stored in state
  - `render()` function can then use JavaScript map function to return state array as modified array of Components which are then displayed on the screen
- **map() works in the following way**
  1. For every item in the array, takes item itself and its index
  2. Creates a new anonymous function that receives item and index and returns a value of the modified data



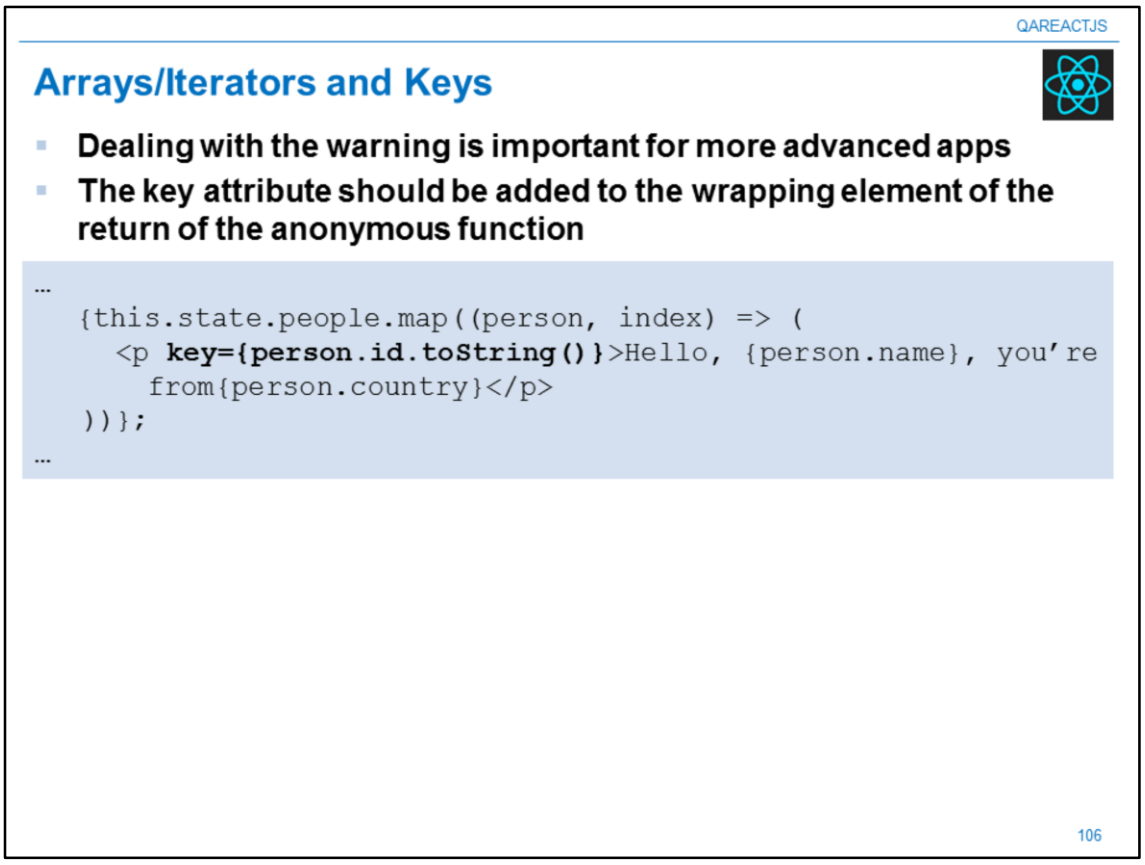
Remember: => (called fat-arrow notation) is ES6's way of declaring a return from an anonymous function.

Multiple HTML elements could be used here as long as they have a wrapping element, eg.

```
…
    {this.state.people.map((person, index) => (
        <div>
            <p>Hello, {person.name}!</p>
            <p>You're from {person.country}.</p>
            <hr />
        <div>
    ))};
…
```

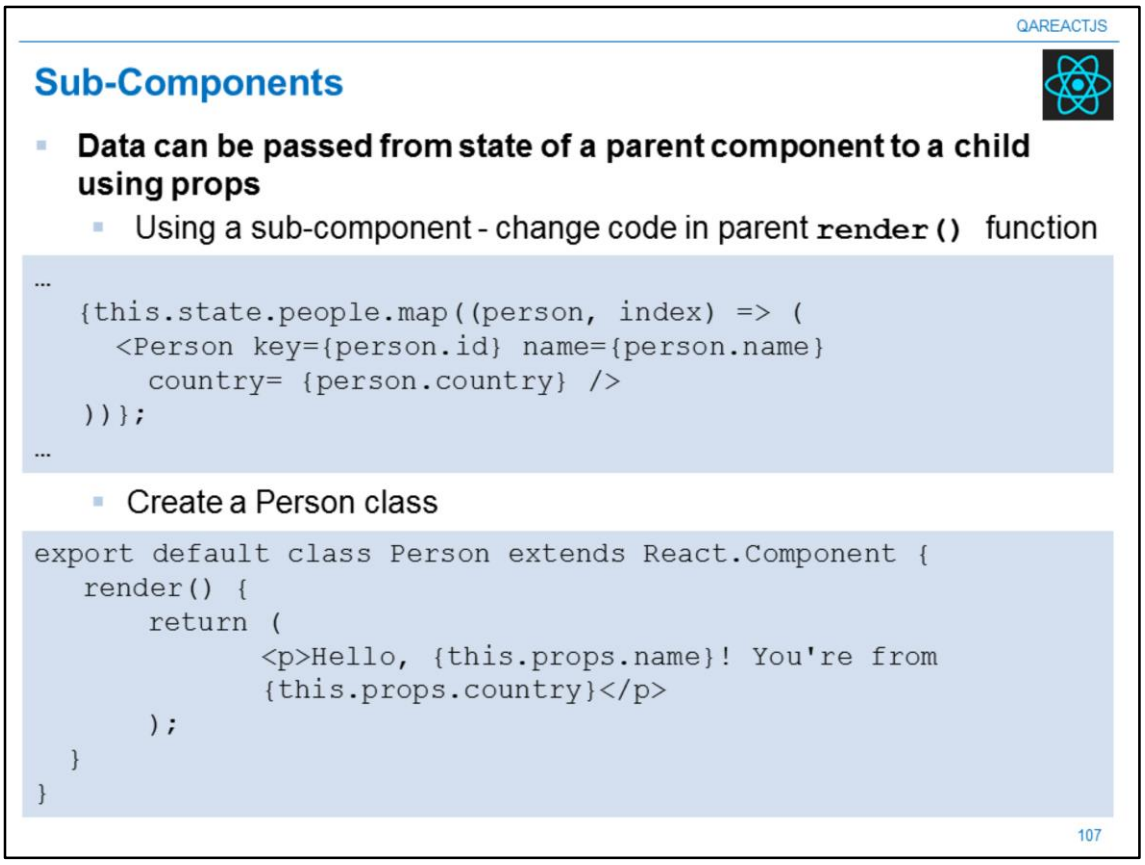


For more information on RECONCILIATION see:
https://facebook.github.io/react/docs/reconciliation.html



If multiple HTML elements used, the key should be included in the wrapping element, eg.

```
…
    {this.state.people.map((person, index) => (
        <div key={person.id.toString()}>
            <p>Hello, {person.name}!</p>
            <p>You're from {person.country}.</p>
            <hr />
        <div>
    ))};
…
```



Note that the key attribute stays within the parent as this is the wrapping component

Functional Components

- **Person could have been written as a functional component:**

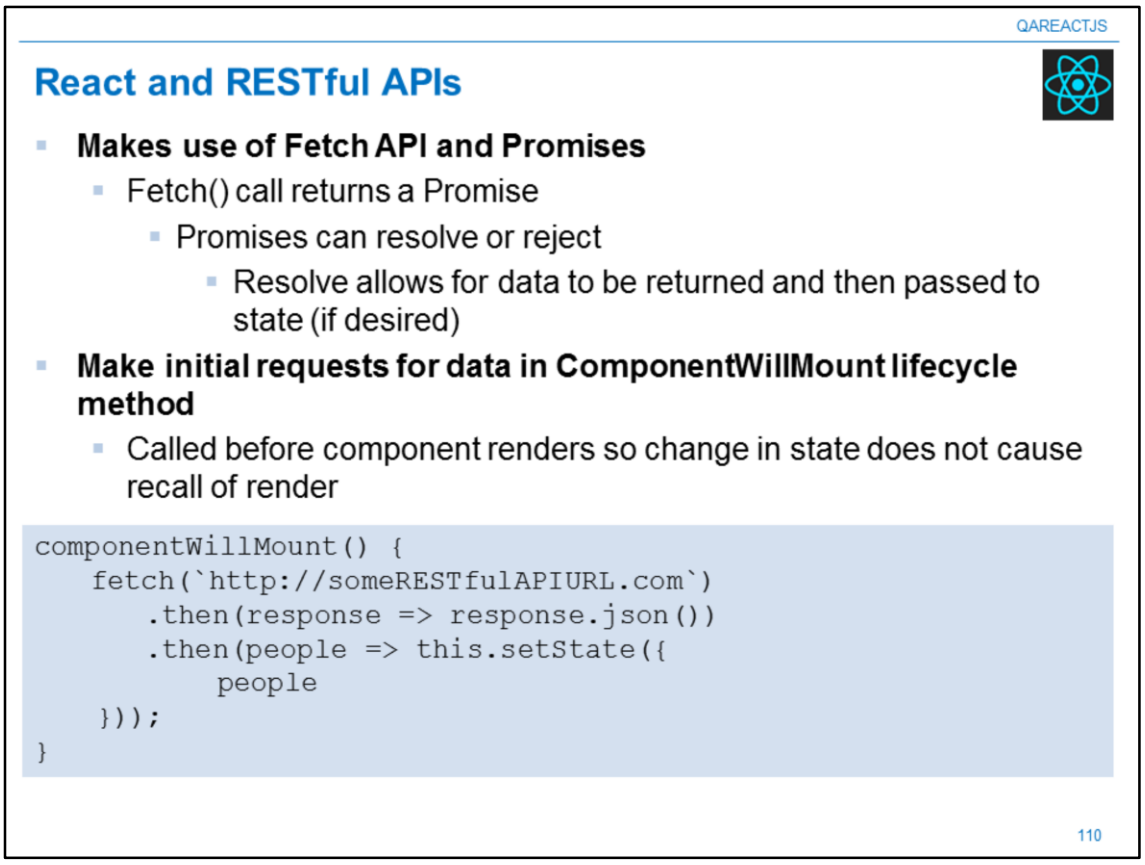
```
const Person = (props) => (
  <p>Hello, {props.name}! You're from {props.country}</p>
);

export default Person;
```

- The functional component can exist in a file by itself or as part of another
- If it is to be used in class in another file, the export statement is needed
- This would be the recommended way to achieve this

RESTful APIs

- **Default data source for most Web and mobile applications**
- **REST stands for Representational State Transfer**
  - Lightweight
  - Maintainable
  - Scalable
- **Not dependent on any protocol**
  - Most use HTTP or HTTPS as the underlying protocol
- **Can be made to provide data in JSON (desirable for JS applications) or XML**



A Promise is a placeholder for asynchronous data that will be available immediately, some time in the future or not at all. Each .then passes the result of the previous action as the argument for the next.

For more information on Using Fetch see https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

For more information on Promises see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Objectives

- To be able to add to a dataset from within a web page
- To be able to read in a dataset to display on a page
- To be able to request and receive data from a RESTful service

Exercise time!

- **EG07 – Working With Datasets**