Libraries in React

A brief introduction to React APIs and their uses.

React is only the view layer for your application. There is some internal state management offered by React, but apart from this, it is only a component library which renders HTML for your browser. Everything else can be added from APIs (e.g. browser API, DOM API), JavaScript functionalities or external libraries. It's not always simple to choose the right library for complementing your React application, but once you have a good overview of the different options, you can pick the one which fits best to your tech stack.

For instance, fetching data in React can be done with the native fetch API like so. *Lines 2 and 3* define constant variables that return a JSON based on a query from a website (API). *Lines 15-17* in the `componentDidMount()` method (this will be discussed in future lessons) fetch the data and set the title within the state to a title returned in the JSON file from the API given here.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

But it is up to you to use another library to fetch data in React. Axios is one popular choice for React applications:

```
import React, { Component } from 'react';
import axios from 'axios';

class App extends Component {
  constructor(props){
    super(props);
    this.state = {
```

```
      data: null,
    };
  }


  componentDidMount() {
    axios.get('https://api.mydomain.com')
      .then(data => this.setState({ data }));
  }

  render() {
    // JSX
  }
}

export default App;
```

So once you know about your problem which need to be solved, React's extensive and innovative ecosystem should give you plenty of options solving it. There again it's not about React, but knowing about all the different JavaScript libraries which can be used to complement your application.