Local Component State

This section will guide you through the basics of React. It covers state and interactions in components as we move past static components. We will also cover the different ways to declare a component, and how to keep components composable and reusable.

Local component state, also known as the internal component state, allows you to save, modify, and delete properties stored in your component. The ES6 class component then uses a constructor to initialize local component state. The constructor is called only once when the component initializes:

Let's introduce a class constructor.

```
class App extends Component {

  constructor(props) {
    super(props);
  }
  ...

}
```

The App component is a subclass of `Component`, so the `extends Component` is in the App component declaration.

It is mandatory to call `super(props);`. It sets `this.props` in your constructor in case you want to access them there. They would be `undefined` when accessing `this.props` in your constructor otherwise. In this case, the initial state of the component should be the sample list of items:

```
const list = [
  {
    title: 'React',
    url: 'https://reactjs.org/',
    author: 'Jordan Walke',
    num_comments: 3,
    points: 4,
    objectID: 0,
  },
  ...
];
```

```
class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      list: list,
    };
  }

  ...

}
```

The state is bound to the class using the `this` object, so you can access the local state of the whole component. For instance, it can be used in the `render()` method. Previously, you have mapped a static list of items in your `render()` method that was defined outside of your component. Now you are about to use the list from your local state in your component.

```
import React, { Component } from 'react';
require('./App.css');

const list = [
  {
    title: 'React',
    url: 'https://reactjs.org/',
    author: 'Jordan Walke',
    num_comments: 3,
    points: 4,
    objectID: 0,
  },
  {
    title: 'Redux',
    url: 'https://redux.js.org/',
    author: 'Dan Abramov, Andrew Clark',
    num_comments: 2,
    points: 5,
    objectID: 1,
  },
];

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      list: list,
    };

  }

  render() {
    return (
```

```
      <div className="App">
        { this.state.list.map(item =>
          <div key={item.objectID}>

            <span>
              <a href={item.url}>{item.title}</a>
            </span>
            <span>{item.author}</span>
            <span>{item.num_comments}</span>
            <span>{item.points}</span>
          </div>
        )}
      </div>
    );
  }
}

export default App;
```

The list is part of the component now, in the local component state. You could add, change, or remove items from your list. Every time you change your component state, the `render()` method of your component will run again. That's how you can change your local component state and see the component re-render the correct data from the local state.

Be careful not to mutate the state directly. Instead, you should use a method called `setState()` to modify your states. We will cover these concepts in more depth next chapter.

Exercises:

- Experiment with the local state
  - Define more initial state in the constructor
  - Use and access the state in your `render()` method

Further Readings:

- Read about the ES6 class constructor