Luminis Amsterdam 1st Meetup React

Speakers @SanderMeinema en Maarten Wilschut

Date 12 July 2018

My world is:

We can only see a short distance ahead, but we can see plenty there that needs to be done. - Alan Turing

All our worlds › www.luminis.eu

Kahoot Survey

For all of us to get to know each other.

Outline

1. Introduction to React
2. Hands-on

Introduction to React

The power of React is that the view will be re-rendered based on changes in your data.

➡ By Facebook (they maintain over 50.000 components themselves)
➡ Library, not a framework
➡ Abstraction of DOM: virtual DOM
➡ Declarative views
➡ Composed of components
  ★ Simple / Stateless
  ★ Stateful
  ★ Connected (to a an app-wide store)

Component Tree

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>React App</title>
  </head>
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
  </body>
</html>
```

index.js

```javascript
import React from 'react';
import ReactDOM from 'react-dom';
import App from './app';

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

Component Tree

app.js

```
export default class App extends React.Component {
  render() {
    return (
      <Router>
        <div className="app">
          <header className="app-header">
            <h1 className="app-title">Luminis DevCon</h1>
            <Link to="/">Home</Link>
            <Link to="/schedule">Schedule</Link>
            <Link to="/speakers">Speakers</Link>
          </header>
          <main className="app-main">
            <Route exact path="/" component={Home}/>
            <Route path="/schedule" component={Schedule}/>
            <Route path="/speakers" component={Speakers}/>
          </main>
        </div>
      </Router>
    );
  }
}
```

Simple / Stateless React Component

```
import React from ‘react'

class HelloWorld extends React.Component {
    render() {
        return (
            <div>
                Hello World!
            </div>
        );
    }
}

render(<HelloWorld />, mountNode);
```

http://blog.isquaredsoftware.com/presentations/2017-02-react-redux-intro/#/14

Stateful React Component

```
class Counter extends React.Component {
    state = {counter : 0}

    onClick = () => {
        this.setState({counter : this.state.counter + 1});
    }

    render() {
        const {counter} = this.state;

        return (
            <div>
                Button was clicked:
                <div>{counter} times</div>

                <button onClick={this.onClick}>Click Me</button>
            </div>
        );
    }
}
render(<Counter />, mountNode);
```

http://blog.isquaredsoftware.com/presentations/2017-02-react-redux-intro/#/23

Connected React Component

luminis
Conversing worlds

```
const mapStateToProps = (state: { root: {schedule: ScheduleModel} }) => {
    // Extract isLoading and presentations from Redux state
    const {isLoading, presentations} = state.root.schedule;

    return {isLoading, presentations};
};

const mapDispatchToProps = (dispatch: Function) => {
    // Add fetchSchedule to props
    return {
        fetchSchedule: () => dispatch(fetchSchedule())
    };
};

export class Schedule extends React.Component<any, any> {
    …
}

export default connect(mapStateToProps, mapDispatchToProps)(Schedule);
```

React Context API

Instead of passing props down the component tree, you can use the Context API for data that is considered global.

```
const ThemeContext =
React.createContext('light');

class ThemeProvider extends React.Component {
  state = {theme: 'light'};

  render() {
    return (
      <ThemeContext.Provider
value={this.state.theme}>
        {this.props.children}
      </ThemeContext.Provider>
    );
  }
}
```

```
class ThemedButton extends React.Component {
  render() {
    return (
      <ThemeContext.Consumer>
        {theme => <Button theme={theme} />}
      </ThemeContext.Consumer>
    );
  }
}
```

Event Handlers

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { items: [], text: '' };
    this.handleChange = this.handleChange.bind(this);
  }

  render() {
    return (
      <div>
        <h3>TODO</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <label htmlFor="new-todo">
            What needs to be done?
          </label>
          <input
            id="new-todo"
            onChange={this.handleChange}
            value={this.state.text}
          />
          <button>
            Add #{this.state.items.length + 1}
          </button>
        </form>
      </div>
    );
  }

  handleChange(e) {
    this.setState({ text: e.target.value });
  }

  handleSubmit = e => {
    e.preventDefault();
    if (!this.state.text.length) {
      return;
    }
    const newItem = {
      text: this.state.text,
      id: Date.now()
    };
    this.setState(prevState => ({
      items: prevState.items.concat(newItem),
      text: ''
    }));
  }
}

class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(item => (
          <li key={item.id}>{item.text}</li>
        ))}
      </ul>
    );
  }
}

ReactDOM.render(<TodoApp />, mountNode);
```

React Component Lifecycle (API 1/5)

Mounting
Updating
Unmounting
New props
setState()
forceUpdate()
constructor
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount
"Render Phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
"Pre-Commit Phase"
Can read the DOM.
"Commit Phase"
Can work with DOM, run side effects, schedule updates.

http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

React Component Lifecycle (API 2/5)

Mounting

These methods are called when an instance of a component is being created and inserted into the DOM:

- constructor(props)
- **static** getDerivedStateFromProps(nextProps, prevState) > stateChange
- componentWillMount() / UNSAFE_componentWillMount()
- render()
- componentDidMount()

Unmounting

This method is called when a component is being removed from the DOM:

- componentWillUnmount()

React Component Lifecycle (API 3/5)

Updating

An update can be caused by changes to props or state. These methods are called when a component is being re-rendered:

- componentWillReceiveProps() / UNSAFE_componentWillReceiveProps()
- **static** getDerivedStateFromProps(nextProps, prevState) > stateChange
- shouldComponentUpdate()
- componentWillUpdate() / UNSAFE_componentWillUpdate()
- render()
- getSnapshotBeforeUpdate() > snapshotValue
- componentDidUpdate(prevProps, prevState, snapshot)

React Component Lifecycle (API 4/5)

Error Handling

This method is called when there is an error during rendering, in a lifecycle method, or in the constructor of any child component.

- componentDidCatch(error, info)

getDerivedStateFromProps

Conversing worlds

```
static getDerivedStateFromProps(nextProps, prevState) {
  const stateMutation = {};

  // Add a trigger to focus on the last added speaker
  if (nextProps.speakers.lastAddedSpeaker && nextProps.speakers.lastAddedSpeaker !== prevState.lastAddedSpeaker) {
    stateMutation.focusOnLastAddedSpeaker = true;
    stateMutation.lastAddedSpeaker = nextProps.speakers.lastAddedSpeaker;
  }

  return stateMutation;
}
```

React Component (API 5/5)

Other APIs

Each component also provides some other APIs:

- setState()
- forceUpdate() - use, for example, for external data changes

Class Properties

- defaultProps - if the prop is not defined
- displayName - for debug messages

Instance Properties

- props - passed by the parent caller (read-only)
- state - use *as* immutable

Questions?

```
render() {
    return (
        <AnsweringQuestionsComponent>
            { this.props.questionsFromPublic.map(
                (question, index) =>
                    (<Answer key={`question-${index}`} question={question} />))
            }
        </AnsweringQuestionsComponent>
    );
```

Hands-on

https://github.com/mrtnw/kennissessie-react

Sources and more information

- https://reactjs.org/
- https://spring.io/guides/tutorials/react-and-spring-data-rest/
- https://medium.com/@franleplant/react-higher-order-components-in-depth-cf9032ee6c3e
- http://blog.isquaredsoftware.com/2017/02/presentation-react-redux-intro/