React II

Martin Ledvinka

martin.ledvinka@fel.cvut.cz

Winter Term 2016

Contents

Event Handling Notes

Event Handling - binding `this`?

ES5 and older

- `this` is automatically bound to be the enclosing object.

```
const Component = React.createClass({
  getInitialState: function() {
    return {
      value: ''
    };
  },

  _onChange: function(e) {
    this.setState({value: e.target.value});
  },
  render: function() {
    return <div>
      Enter some text:
      <input value={this.state.value} onChange={this._onChange}/>
      </div>;
  }
});
```

this is the Component instance.

Figure: See live example at `http://codepen.io/ledsoft/pen/KNBogV`

Event Handling - binding `this`?

ES6

- `this` has to be explicitly bound, otherwise it would be `undefined`.

```
class Component extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: ''
    };
  }

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

  render() {
    return <div>
      Enter some text:
      <input value={this.state.value} onChange={this._onChange.bind(this)}/>
      </div>;
  }
}
```

Have to explicitly bind this for _onChange.

Figure: See live example at `http://codepen.io/ledsoft/pen/PbBRWv`

Event Handling - binding `this`?

ES7

- `this` is automatically bound to the enclosing object.

```
class Component extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: ''
    };
  }

  _onChange = (e) => {
    this.setState({value: e.target.value});
  };

  render() {
    return <div>
      Enter some text:
      <input value={this.state.value} onChange={this._onChange}/>
      </div>;
  }
}
```

this is the Component instance.

Figure: See live example at `http://codepen.io/ledsoft/pen/rWrdzw`

(Re)Flux Recap

Flux

- Architectural pattern rather than framework,
- One way flow simplifies tracking application state and its changes,
- Separate business logic from UI components,
- works well with the *one way dataflow* philosophy of React.

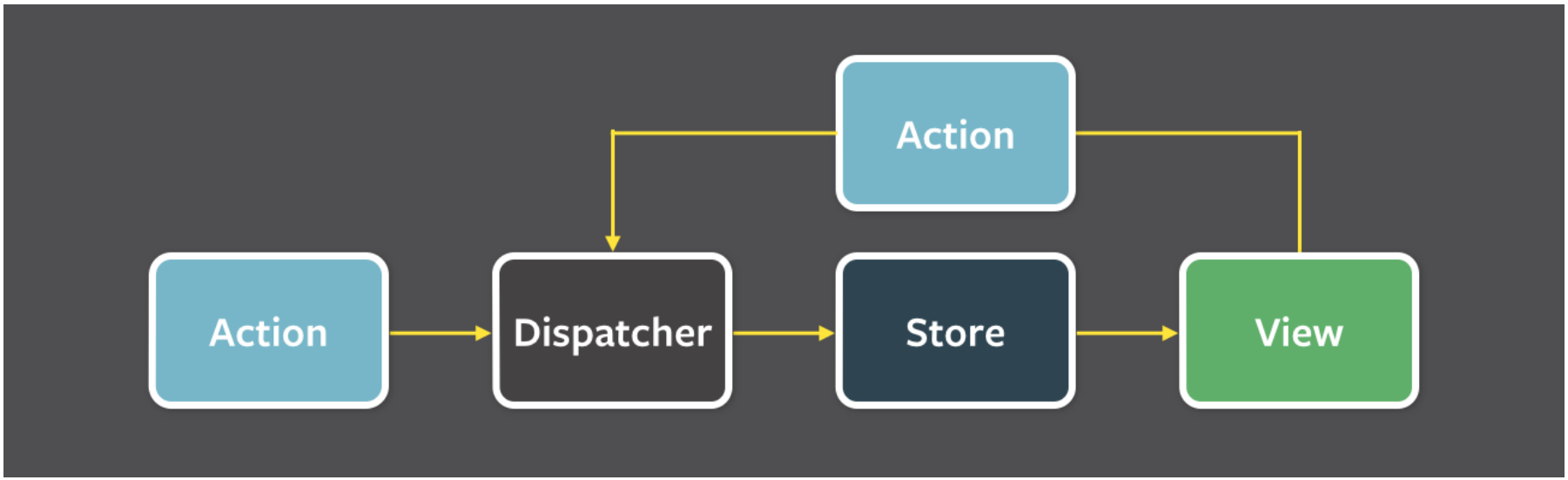

Figure: Flux architecture. Source: `https://facebook.github.io/flux/img/flux-simple-f8-diagram-with-client-action-1300w.png`

Flux Example

```
export default class TeachersController extends React.Component {
    constructor(props) {
        super(props);
        this.state = { teachers: [] };
    }

    componentDidMount() {
        Actions.loadTeachers();
        this.unsubscribe = TeacherStore.listen(this._onTeachersLoaded);
    }

    componentWillUnmount() {
        this.unsubscribe();
    }

    _onTeachersLoaded = (data) => {
        this.setState({teachers: data});
    };

    render() {
        return <Teachers teachers={this.state.teachers}/>;
    }
}
```

```
const TeacherStore = Reflux.createStore({
    listenables: [Actions],

    onLoadTeachers: function() {
        request.get(URL).accept('json').end((err, resp) => {
            if (err) {
                console.log('Error when loading teachers. Status: ' + err.status);
            } else {
                this.trigger(resp.body);
            }
        });
    }
});
```

Tasks

Tasks

Working with ear-rt, the latest version from git. Continue building up the user profile editing.

1. Upload the updated user profile data to the server,
   - Add action,
   - Implement store functionality,
   - Wire it together.
2. Reload the user after successful update.
3. Password update should open only on demand,
4. Password update should require original password to be entered (will be checked on server on submit).

Resources

- `https://facebook.github.io/react/docs/handling-events.html`,
- `https://react-bootstrap.github.io/components.html`,
- `https://github.com/reflux/refluxjs`.