

React

Martin Ledvinka

`martin.ledvinka@fel.cvut.cz`

Winter Term 2017



Contents

- 1 React
 - JSX
 - React Principles
- 2 Flux
- 3 React vs Other JS Frameworks
- 4 Useful Libraries
- 5 Demo



React



React

A JavaScript library for building user interfaces.

- <https://facebook.github.io/react/>,
- Initial release in 2013,
- Created and developed by Facebook,
- Open-source,
- Used by Facebook and Instagram,
- *React Native* for developing native applications for iOS, Android and UWP in JS,
- Easy to integrate into legacy web applications.



JSX

- Provides syntactic sugar for expressing component structure declaratively in JS,
- Code in JSX is compiled into plain JavaScript.

Example

JSX code

```
<MyButton color="blue" shadowSize={2}>  
  Click Me  
</MyButton>
```

Compiles into:

```
React.createElement(  
  MyButton,  
  {color: 'blue', shadowSize: 2},  
  'Click Me'  
)
```

JSX Principles

- Component name/tag name comes first after opening `<`,
- Properties specified like HTML attributes,
- Property names are *camelCased*,
- Component names should begin with capital letter, e.g. `MyButton`,
- Child elements written between open tag and end tag,
- Tags can be without children – open tags ends with `/ >`,
- JavaScript expressions enclosed in `{}`,
- Value-less properties evaluate to true, e.g. `horizontal`.



JSX Example

```
render() {
  const report = this.props.report,
        valid = ReportValidator.isValid(report);
  if (!report) {
    return <Panel header=<h2 className='panel-title pull-left'>{this.i18n('occurrencereport.title')}</h2>
      bsStyle='primary'/>;
  }
  return <Panel header=<h3>{this.i18n('occurrencereport.title')}</h3>
    bsStyle='primary'>
    <Occurrence occurrence={report.occurrence} onChange={this.props.onChange}/>

    <div className='row'>
      <div className='col-xs-12'>
        <Input type='text' name='narrative' label={this.i18n('narrative')} rows={8}
          placeholder={this.i18n('narrative')} title={this.i18n('report.narrative-tooltip')}
          value={report.narrative} onChange={this._onChange}/>
      </div>
    </div>

    <ButtonToolbar className='float-right detail-button-toolbar'>
      <Button bsStyle='success' bsSize='small'
        title={this.i18n(valid ? 'detail.save-tooltip' : 'detail.invalid-tooltip')}
        onClick={this.props.save} disabled={!valid}>
        {this.i18n('save')}
      </Button>
      <Button bsStyle='link' bsSize='small' title={this.i18n('cancel-tooltip')}
        onClick={this.props.cancel}>{this.i18n('cancel')}</Button>
    </ButtonToolbar>
  </Panel>;
}
```

Figure : Example of a more complex JSX structure.



Technical Intermezzo

We use ES6 (latest JS standard) and ES7 experimental (future JS standard) features:

- *Fat arrow function syntax*
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions,
 - <http://codepen.io/ledsoft/pen/JbKbqD>
- ES6 classes
 - <https://developer.mozilla.org/cs/docs/Web/JavaScript/Reference/Classes>,
 - <http://codepen.io/ledsoft/pen/aBZpzO>
- ES7 *property initializers*,
 - <http://codepen.io/ledsoft/pen/YpWNNQ>
- *Babel* to compile everything into plain old JS supported by all modern browsers.



Basic Features/Principles

- Composable components,
 - React elements – React representations of standard HTML elements, e.g. `div`, `img` etc.,
 - Components – user-defined components,
- Immutable properties (props) + mutable state,
- Virtual DOM,
- One way dataflow.



User-defined Components

- 1 Function taking properties and returning stuff to render,

```
const ExternalLink = (props) => {  
  var classes = classNames('external-link', props.className);  
  return <a href={props.url} title={props.title} target='_blank' className={classes}/>;  
};
```

- 2 Class extending `React.Component` with a render method, which returns stuff to render.

```
class Teachers extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
  render() {...}  
  _renderRows() {...}  
}  
Teachers.propTypes = {  
  teachers: PropTypes.array.isRequired  
};
```



Components

- Must render a single element (can be null),
- Are essentially functions,
 - They take arguments (see below) and return result,
- Take a *properties* (`props`) object with configuration as argument/constructor parameter,
- Must behave as *pure functions* with respect to their props – i.e. `props` are **immutable**,
- Two kinds:
 - *Stateless* are just functions of their properties,
 - *Statefull* keep their state which can be modified (usually as response to events).



Props vs. State

props Component receives `props` as parameters for rendering,

state Component keeps state and operates on it, possibly passing it to its children as their `props`.

State is modified by calling the `setState` lifecycle method of the component.

Component functions do not have state.

<http://codepen.io/ledsoft/pen/YpWNNQ>



Component Lifecycle

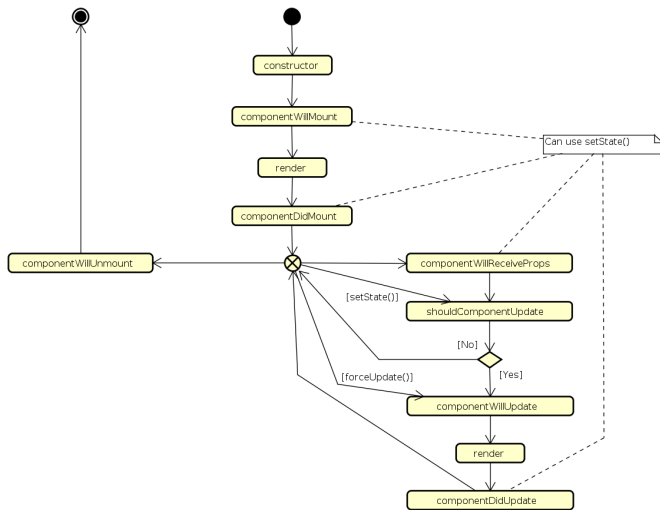


Figure : React component lifecycle methods.



Virtual DOM

- DOM manipulation is expensive,
- React uses in-memory abstract model of DOM,
- On update:
 - 1 Calls render of each component,
 - 2 Compares the result with its original in the virtual DOM,
 - 3 Calculates changes for the real DOM,
 - 4 Performs necessary updates to the real DOM,
- Heuristics turn $O(n^3)$ problem into a linear one.



Virtual DOM

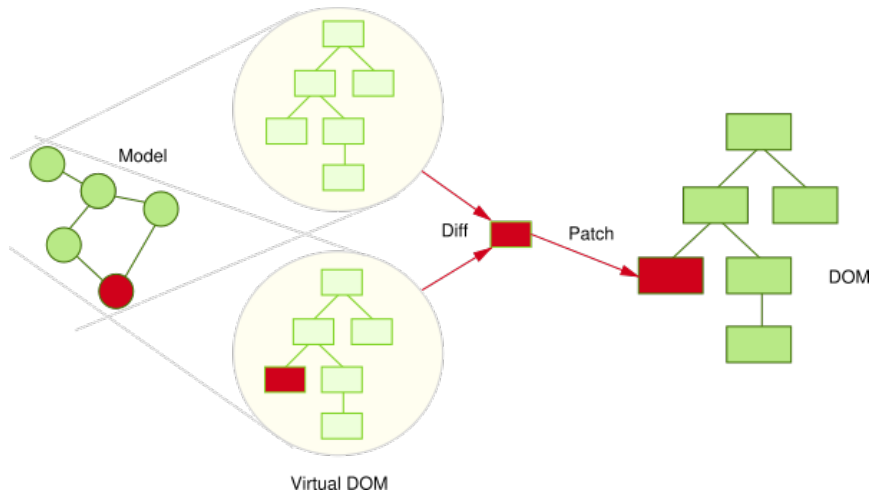


Figure : React Virtual DOM. Source:

http://teropa.info/images/onchange_vdom_change.svg



One Way Dataflow

- Data flow from ancestor components to their descendants,
- Handlers passed down to handle user's input,
- Typically:
 - A component keeps state of a portion of the view,
 - It passes the state down to its descendants as props,
 - It may also pass handlers for updating the state,
 - Descendants are stateless components displaying data according to their props,
- Makes it easier to reason about the application's state,
- <http://codepen.io/ledsoft/pen/KNNmaa>.



Flux



Why another UI pattern?

Two way binding used usually in MVC can lead to messy code. Difficult to track application state and its changes.

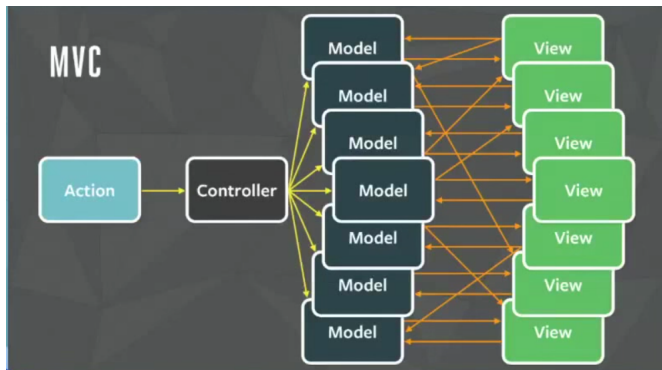


Figure : MVC problems. Source: <https://youtu.be/nYkdrAPrdcw>.



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.

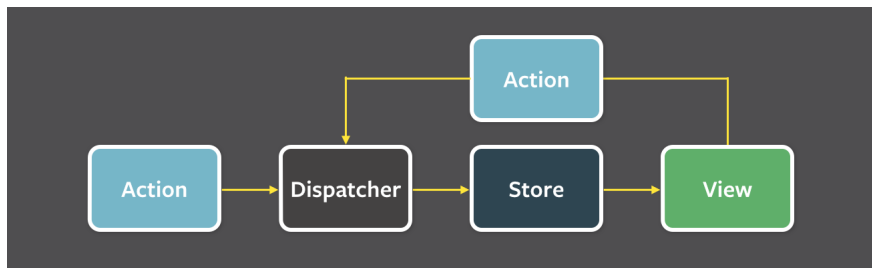


Figure : Flux architecture. Source: <https://facebook.github.io/flux/img/flux-simple-f8-diagram-with-client-action-1300w.png>



Flux Parts

Actions

- Represent events to which application logic should respond,
- May encapsulate data with them,
- Fired by:
 - UI components,
 - Communication with backend.

Dispatcher

- Single dispatcher per application,
- Dispatches actions to the stores,
- Facebook provides open source implementation, which it also uses in production.



Flux Parts II

Stores

- Represent application state and business logic,
- Stores listen to actions and perform operations,
- Can be a collection of business objects,
- Or represent a single domain model object.

Views

- Standard React components,
- Usually one component acts as a *Controller-View*,
 - Listens for store triggers,
 - Loads data from store into its own state,
 - Passes data down to sub-components,
- Controller-View can be one for the whole page, or multiple for sections of a page.

Flux Example

```

export default class TeachersController extends Reflux.Component {
  constructor(props) {
    super(props);
    this.state = {};
    this.store = TeacherStore;
  }

  componentDidMount() {
    Actions.loadTeachers();
  }

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

```

```

export default class TeacherStore extends Reflux.Store {

  constructor() {
    super();
    this.state = {
      teachers: []
    };
    this.listenables = Actions;
  }

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

```



React vs Other JS Frameworks



AngularJS

- Developed by Google, recently published v5 (aggressive release cycle since v2),
- Encourages use of MVC with two-way binding,
- HTML templates enhanced with hooks for the JS controllers,
- New components created using *directives*,
- Built-in routing, AJAX,
- <https://angularjs.org/> (v1), <https://angular.io/> (v2 and later).

```
1. <!doctype html>
2. <html ng-app>
3.   <head>
4.     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/
5.       angular.min.js"></script>
6.   </head>
7.   <body>
8.     <div>
9.       <label>Name:</label>
10.      <input type="text" ng-model="yourName" placeholder="Enter a name here">
11.      <hr>
12.      <h1>Hello {{yourName}}!</h1>
13.    </div>
14.  </body>
```



Ember

- Open source framework,
- Templates using Handlebars,
- Encourages MVC with two-way binding,
- New components created using Handlebars templates + JS,
- Built-in routing, AJAX,
- <http://emberjs.com/>.

```
1 <div>
2   <label>Name:</label>
3   {{input type="text" value=name placeholder="Enter your name"}}
4 </div>
5 <div class="text">
6   <h3>My name is {{name}} and I want to learn Ember!</h3>
7 </div>
```



BackboneJS

- Open source framework,
- Provides models with key-value bindings, collections,
- Views with declarative event handling,
- View rendering provided by third-party libraries - e.g. jQuery, React,
- Built-in routing, AJAX,
- <http://backbonejs.org/>.

```
var Todo = Backbone.Model.extend({

  defaults: function() {
    return {
      title: "empty todo...",
      order: Todos.nextOrder(),
      done: false
    };
  },

  toggle: function() {
    this.save({done: !this.get("done")});
  }

});
```



Useful Libraries



Useful Libraries

React Router

- Routing in React applications,
- Mapping views to URL and handling route transitions,
- <https://github.com/ReactTraining/react-router>.

React Bootstrap

- React components based on the Bootstrap UI library,
- Makes building good looking application easier,
- <https://react-bootstrap.github.io/>.

SuperAgent

- Lightweight AJAX API,
- Useful for communication with backend web services,
- <https://visionmedia.github.io/superagent/>.

Useful Libraries II

Reflux

- Alternative implementation of Flux,
- Simplifies stores, provides dispatcher,
- <https://github.com/reflux/refluxjs>.

Redux

- Alternative to React's Flux,
- State container for JavaScript applications,
- <https://github.com/reactjs/redux>.

React Intl

- Internationalization for React applications,
- React components + API for formatting, translations, handling plurals etc.,
- <https://github.com/yahoo/react-intl>.

Useful Libraries III

Babel

- JavaScript compiler,
- Enables use of advanced and novel JS features in a way compatible with older browsers,
- <https://babeljs.io/>.

Jasmine

- BDD-style test framework for JavaScript,
- <https://jasmine.github.io/>.

Jest

- JavaScript test framework,
- Developed and used by Facebook,
- <https://facebook.github.io/jest/>.

Demo



React Demo

`https://codepen.io/ledsoft/pen/JrbYxy`



The End

Thank You



Resources

- <http://todomvc.com/>,
- <https://babeljs.io/blog/2015/06/07/react-on-es6-plus>,
- <http://teropa.info/blog/2015/03/02/change-and-its-detection-in-javascript-frameworks.html>,
- <https://facebook.github.io/react/>,
- <https://facebook.github.io/flux/docs/overview.html>.

