React - from zero to hero

Start building React apps from day one.

DayOne

The Plan

What is React

The React Way

Env setup

JSX

Components

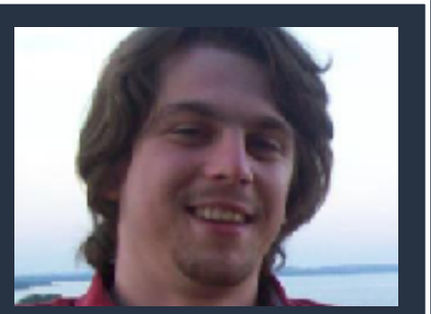
Code, code, code …

Before we begin

1. Install git
2. Install NodeJS
3. git clone https://github.com/DayOnePI/react-zero-to-hero
4. npm install

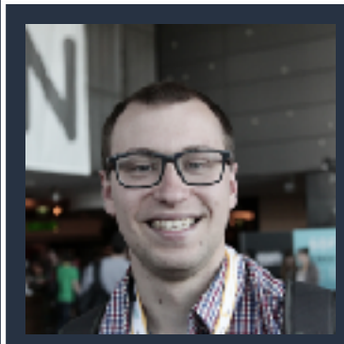
The Team

Marek Piechut



Playing with react for more than 3 years. Not very keen on ES6 classes, but love stateless functional components.

@marekpiechut
marek@dayone.pl

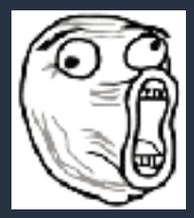


Bartek Witczak

React ninja from version 0.13. Functions everywhere. Big proponent of Immutable JS.

@bartekwitczak
bartek@dayone.pl

What is React

- Javascript library for building user interfaces
- Not a full SPA framework, only UI
- but you can use it to build other stuff
  - ReactNative - (almost) native mobile apps
  - React VR - … yes VR 

The React Way

- Declarative
  - update ≈ full page reload
- Component based
- Virtual DOM
- One way data binding
  - yes, working with forms is hell
- Immutability

JSX

- Simple HTML-like tag syntax
- Put JS code in {}
- Most of HTML is already there

JSX

```
<div className='col-md-4 grey-panel pb-3'>
   <hr/>
   <div className='text-center'>
      <button onClick={this.showNewPopup}
         className='btn btn-danger'>Add new</button>
   </div>
</div>
```

Exercise 1

Add some nifty HTML to page using JSX notation

```
$ git checkout ex1
$ npm install
$ npm start
$ vim src/banner.js
ESC :q ENTER
```

Components

- Components are a core concept in React
- Everything that goes to page has to go through a component
- Component has a lifecycle (more about it later)
- … and you can use it for much more than rendering UI
- Simplest form: stateless functional component

Functional components

```
import React from 'react'

const Welcome = ({system}) => (
    <span>Welcome to the {system}</span>
)
```

- `import React from ‘react’` - just put it everywhere
- It’s just a function, really. No magic here.
- Will render Welcome to the App when passed “App” as system
- Can be used like HTML tag: `<Welcome system=‘App’/>`
- Must start with a capital letter!

Exercise 2

Create custom component showing bike details

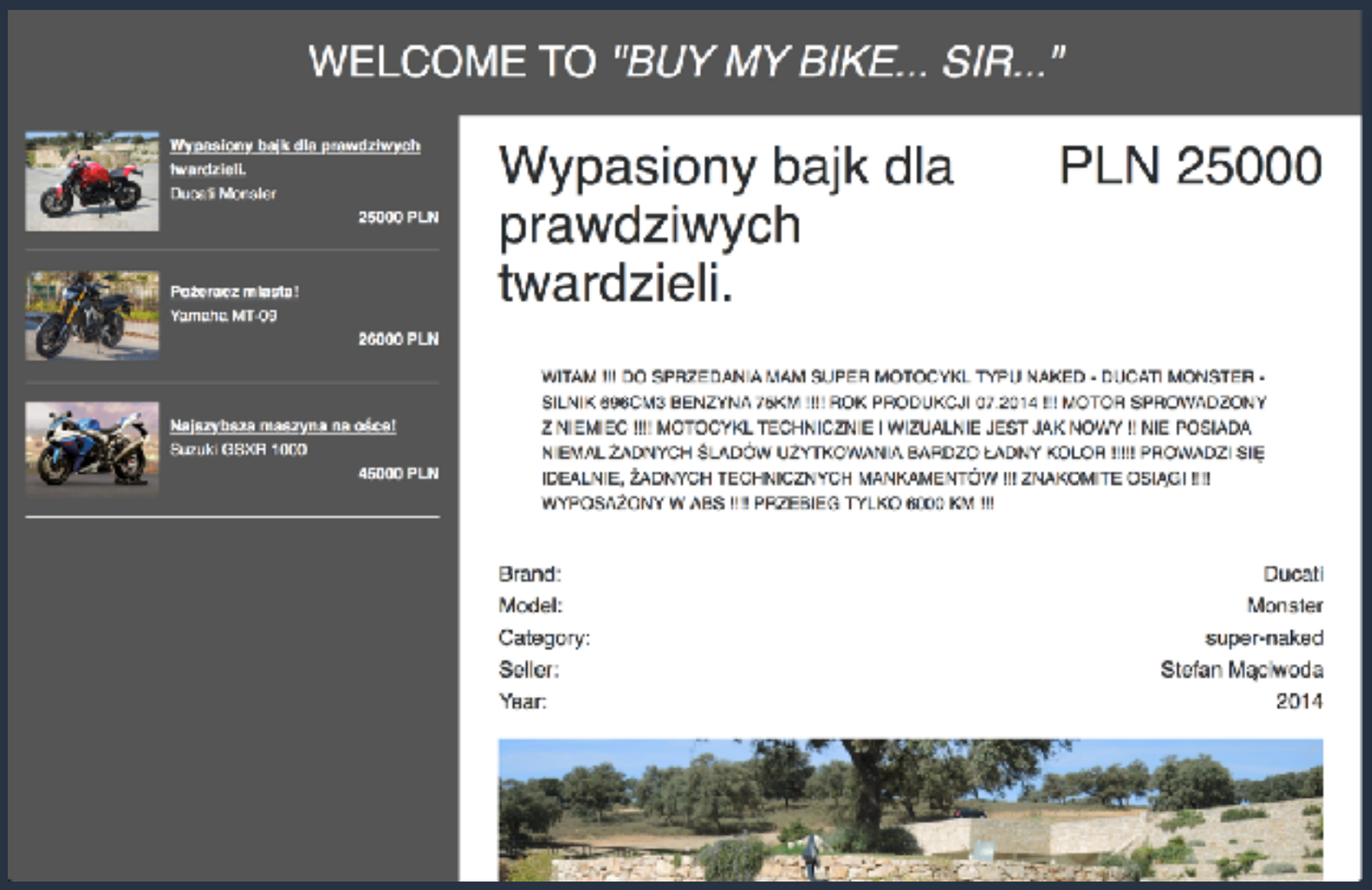
```
$ git checkout ex2
$ npm start
$ vim src/bike-details.js
ESC :q ENTER
```

Passing data down

- Props
  - Sent as tag params
  - Can be anything: string, object, array…
  - Immutable
  - Tag children also go here
- Context
  - hush, not today… powerful but easy to misuse

Exercise 3

Create bikes list




```
$ git checkout ex3
$ npm start
$ vim src/bikes-list.js
ESC :q ENTER
```

Using components

- You can pass components as props
- or as children, just like in HTML

```
<BikeDetails bike={<Bike brand='Ducati' model='Monster'/>} /> //props.bike

//OR

<BikeDetails>
    <Bike brand='Ducati' model='Monster'/> //props.children
</BikeDetails>
```

- Which one looks better?

Stateful components

- If you need more complex behavior
- If you do network requests
- If you’re creating a form
- If your component is expensive to render
- If you're integrating a stateful library
- Stateless components are preferred though

```
export default React.createClass({
    displayName: 'AddNew',

    getInitialState() {
        return {}
    },

    handleChange(e) {
        const {name, value} = e.target
        this.setState({[name]: value})
    },

    render() {
        const {desc} = this.state
        return (
            <form>
                <h1>Add new</h1>
                <div className="form-group">
                    <label>Description</label>
                    <input name="desc"
                           className="form-control"
                           onChange={this.handleChange}
                           value={desc}
                    />
                </div>
            </form>
        )
    }
})
```

Exercise 4

Make bikes selectable
- clicking on list changes bike details

```
$ git checkout ex4
$ npm start
$ vim src/main.js
ESC :q ENTER
```

Forms

- Unmanaged forms
    - As usual in HTML - user does what he wants and you need to get data somehow
- Managed forms
    - React synchronizes form data with state
    - You have callback on input change and update state
- Which one to use?
    - Use managed
    - When in doubt… use managed
    - When you're not sure… use managed
    - If really no other choice, use unmanaged

Exercise 5

Form to add a new bike

```
$ git checkout ex5
$ npm start
$ vim src/add-new.js
ESC :q ENTER
```

Lifecycle

1. componentWillMount()
2. render()
3. componentDidMount()
4. componentWillReceiveProps()
5. shouldComponentUpdate()
6. componentWillUpdate()
7. render()
8. componentDidUpdate()
9. componentWillUnmount()

HOC

- Functions that take component and return wrapping component
- Cross-cutting concerns
  - Logging
  - Access rights
  - Loading indicators
  - State management (redux)

HOC

```
const withLoading = (Component) => ({loading, ...props}) => {
    if(loading) {
        return <Loading />
    } else {
        return <Component {...props} />
    }
}
```

Exercise 6

Log lifecycle changes of a component using HOC

```
$ git checkout ex6
$ npm start
$ vim src/add-new.js
ESC :q ENTER
```

Thank You

Eny ketions?

DayOne