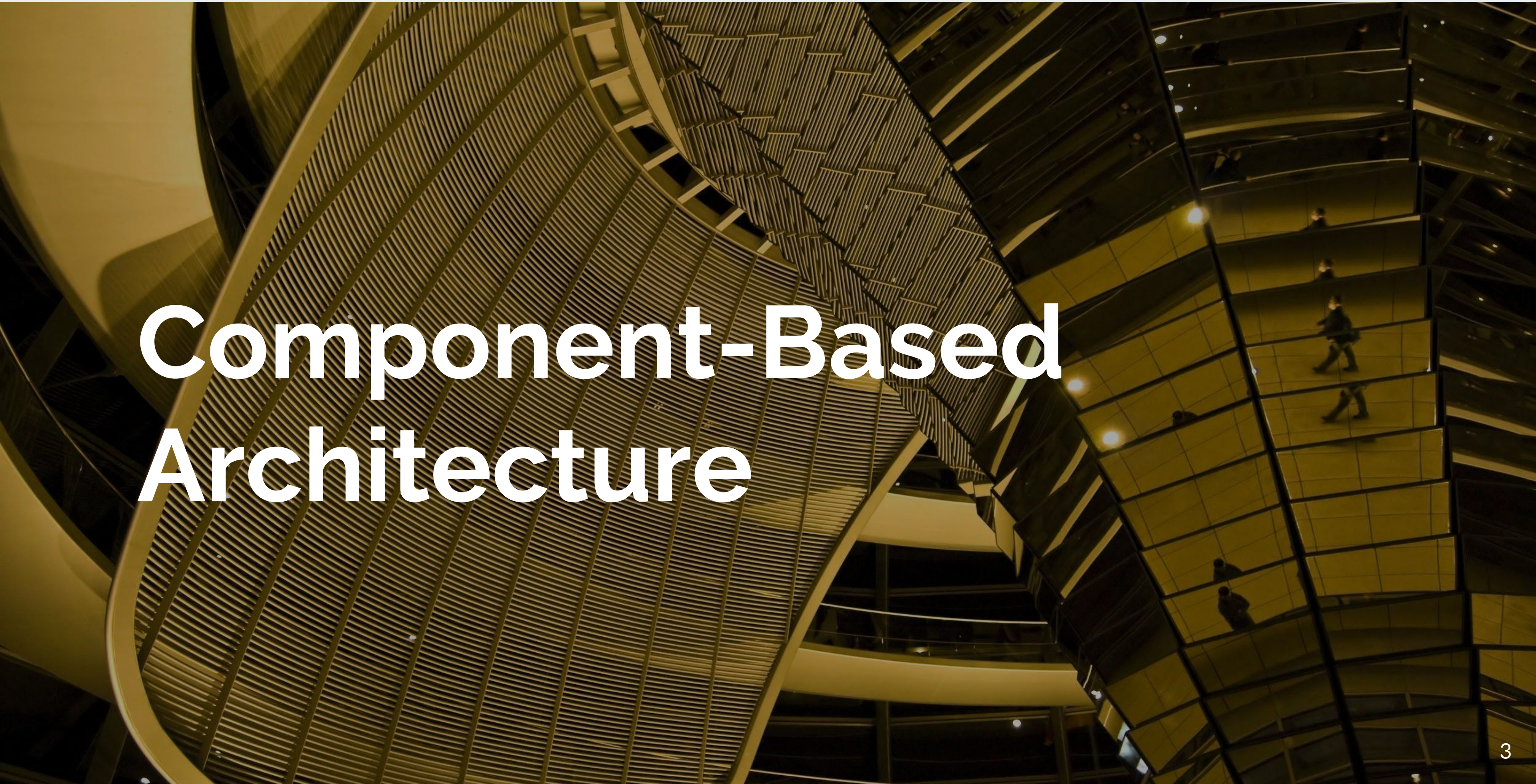
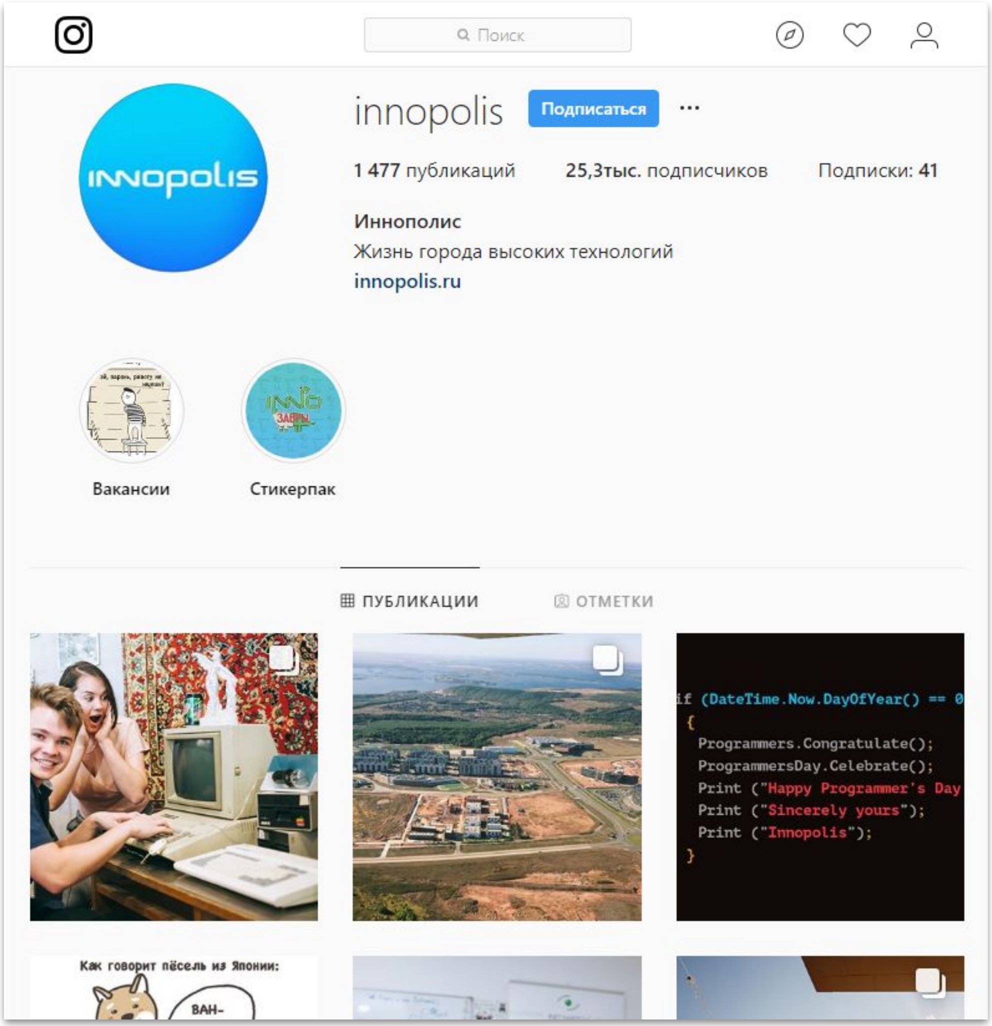
JavaScript for Enterprise Development

Week 3-1

CONTENTS



Component-Based Architecture



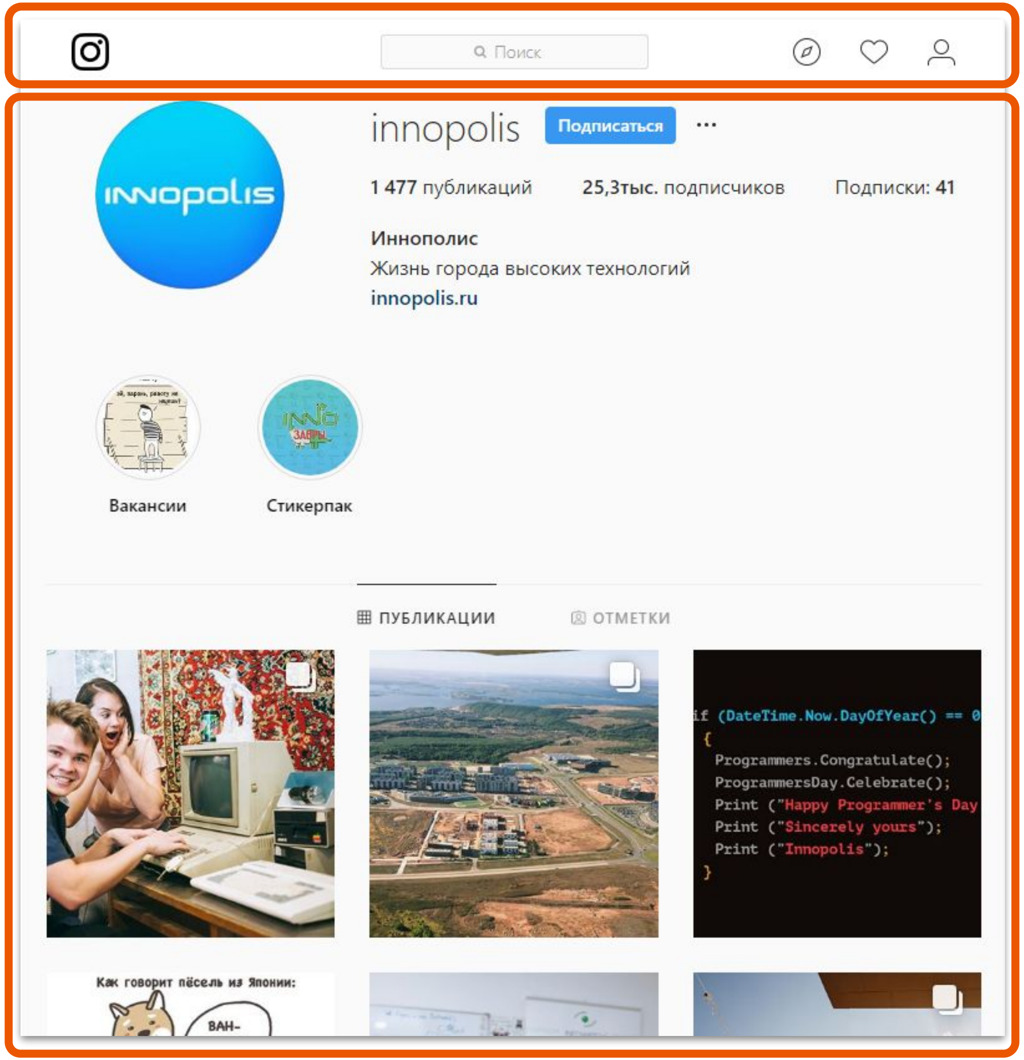
1 477 публикаций 25,3тыс. подписчиков Подписки: 41

Иннополис

Жизнь города высоких технологий

innopolis.ru

```
if (DateTime.Now.DayOfYear() == 0
 {
  Programmers.Congratulate();
  ProgrammersDay.Celebrate();
  Print ("Happy Programmer's Day
  Print ("Sincerely yours");
  Print ("Innopolis");
 }
```



innopolis Подписаться ···

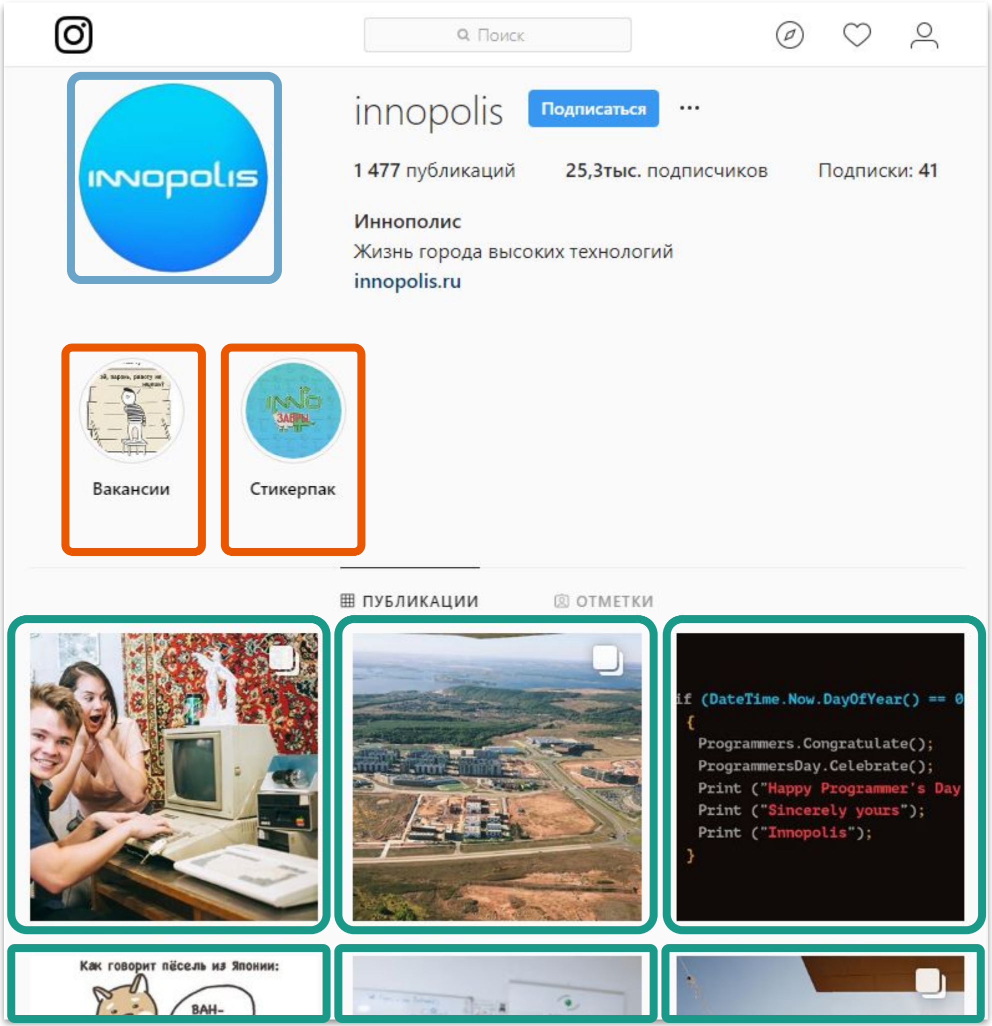
1 477 публикаций 25,3тыс. подписчиков Подписки: 41

Иннополис
Жизнь города высоких технологий
innopolis.ru

Вакансии Стикерпак

ПУБЛИКАЦИИ ОТМЕТКИ

```
if (DateTime.Now.DayOfYear() == 0
{
   Programmers.Congratulate();
   ProgrammersDay.Celebrate();
   Print ("Happy Programmer's Day
   Print ("Sincerely yours");
   Print ("Innopolis");
}
```



Поиск

innopolis Подписаться ···

1 477 публикаций 25,3тыс. подписчиков Подписки: 41

Иннополис
Жизнь города высоких технологий
innopolis.ru

```
if (DateTime.Now.DayOfYear() == 0
{
  Programmers.Congratulate();
  ProgrammersDay.Celebrate();
  Print ("Happy Programmer's Day
  Print ("Sincerely yours");
  Print ("Innopolis");
}
```

Component

Self-sustaining, independent element of user interface

- Comprises of UI and some logic
- Could be reusable
- Could contain other components

Component-based web applications

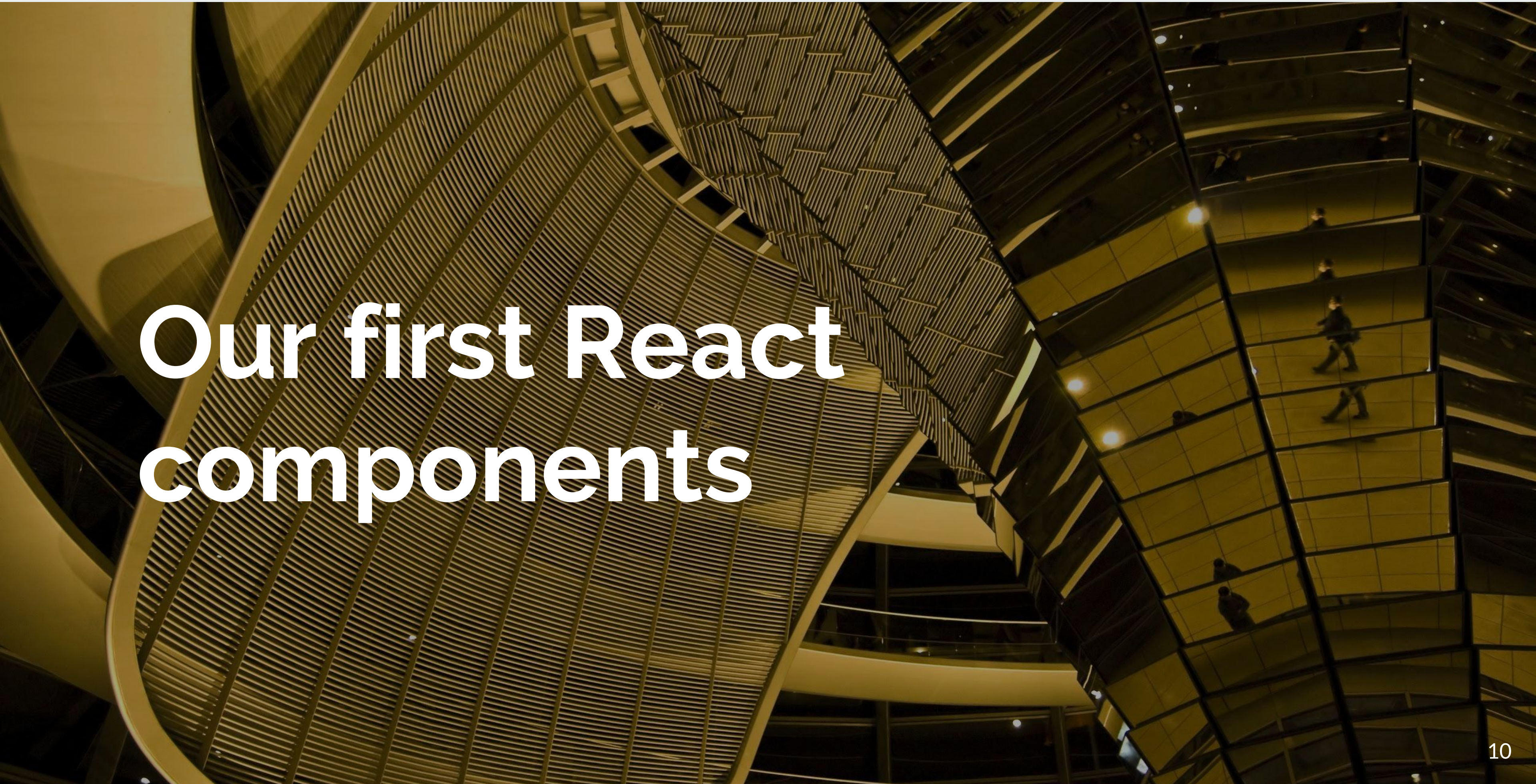
Vue.js

React

Angular

React: our further test subject

React

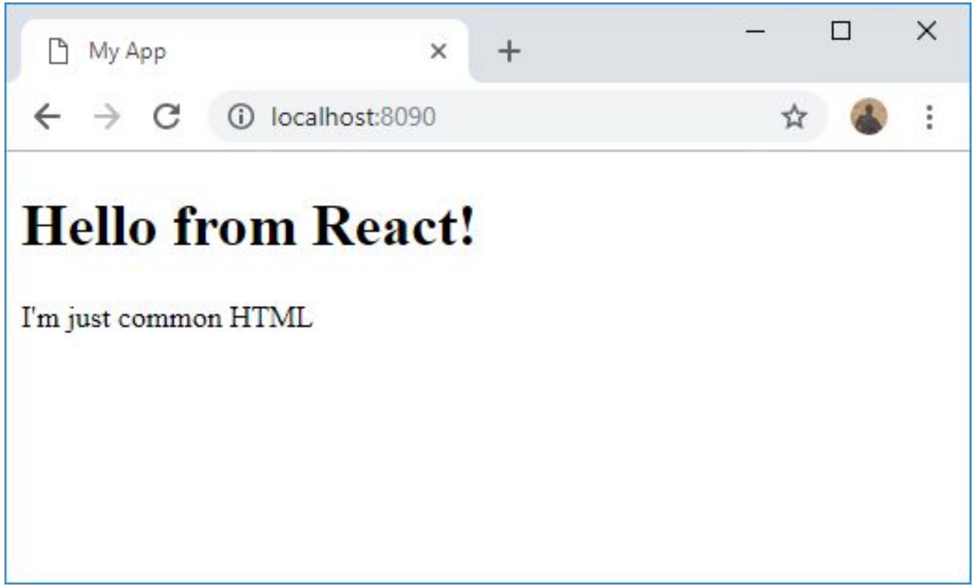


Our first React components

Simple React component

```
const MyFirstComponent = function () {
 return (
   <div>
     <h1>Hello from React!</h1>
     <span>I`m just common HTML</span>
   </div>
 )
};
```

Simple React component



Hello from React!

I'm just common HTML

JSX to JS

```
import React from "react";

const MyFirstComponent = function () {
 return React.createElement('div', {},
   React.createElement('h1', {}, 'Hello from React!'),
   React.createElement('span', {}, 'I`m just common HTML')
 );
};
```

Nested components

```
const HeaderComponent = function () {
 return <h1>Hello from React!</h1>
}

const MyFirstComponent = function () {
 return (
   <div>
     <HeaderComponent />
     <span>I'm just common HTML</span>
   </div>
 )
};
```

Props

```
const MyFirstComponent = function (props) {
 return (
   <div>
     <h1>{props.title}</h1>
     <span>I'm just common HTML</span>
   </div>
 )
};

ReactDOM.render(
 <MyFirstComponent title="Hello from React!"/>,
 document.getElementById('app')
);
```

So…

That’s the basics of React components.

How’d we get that in our browsers?

Development environment

What do we need?

1. Transpile modern JavaScript into browser-compatible code - **Babel**
2. Transpile JSX syntax into JavaScript - **@babel/preset-react**

Required packages

```
npm i
    babel-loader
    @babel/core
    @babel/preset-env
    @babel/preset-react
    react
    react-dom
--save-dev
```

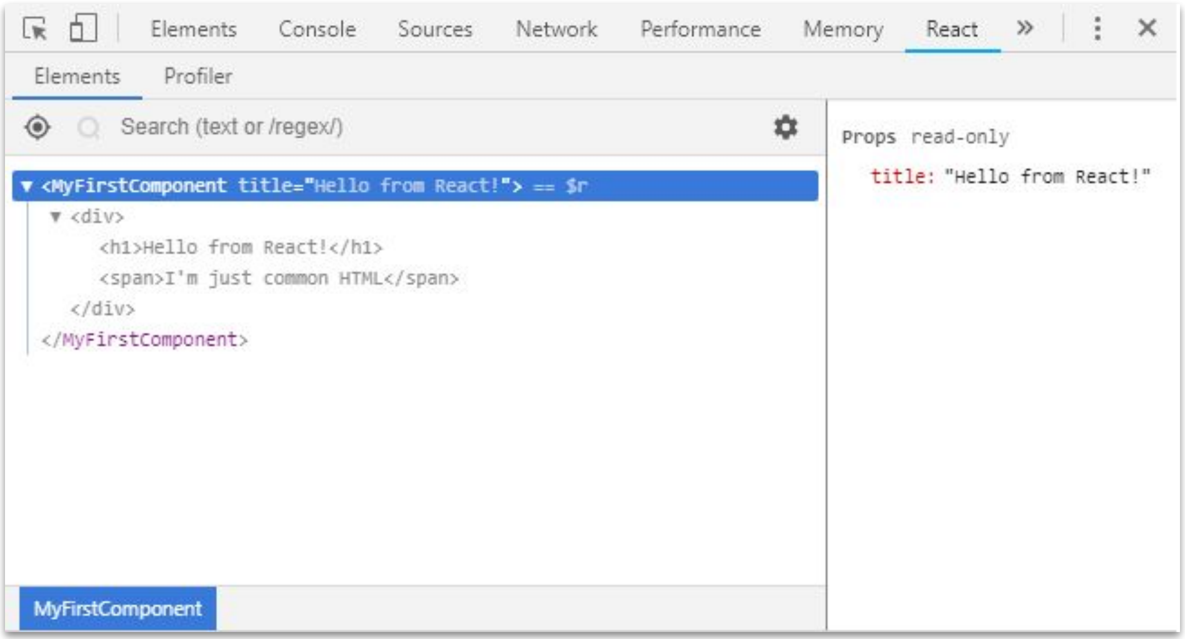
New project file: .babelrc

```
{
 "presets": [
   "@babel/preset-env",
   "@babel/preset-react"
 ]
}
```

New webpack.config.js options

```
module: {
 rules: [{
   test: /\.js$/,
   exclude: /node_modules/,
   use: {
     loader: "babel-loader"
   }
 }]
}
```

Developer tools



```
▼ <MyFirstComponent title="Hello from React!"> == $r
  ▼ <div>
      <h1>Hello from React!</h1>
      <span>I'm just common HTML</span>
    </div>
  </MyFirstComponent>
```

```
Props read-only
  title: "Hello from React!"
```

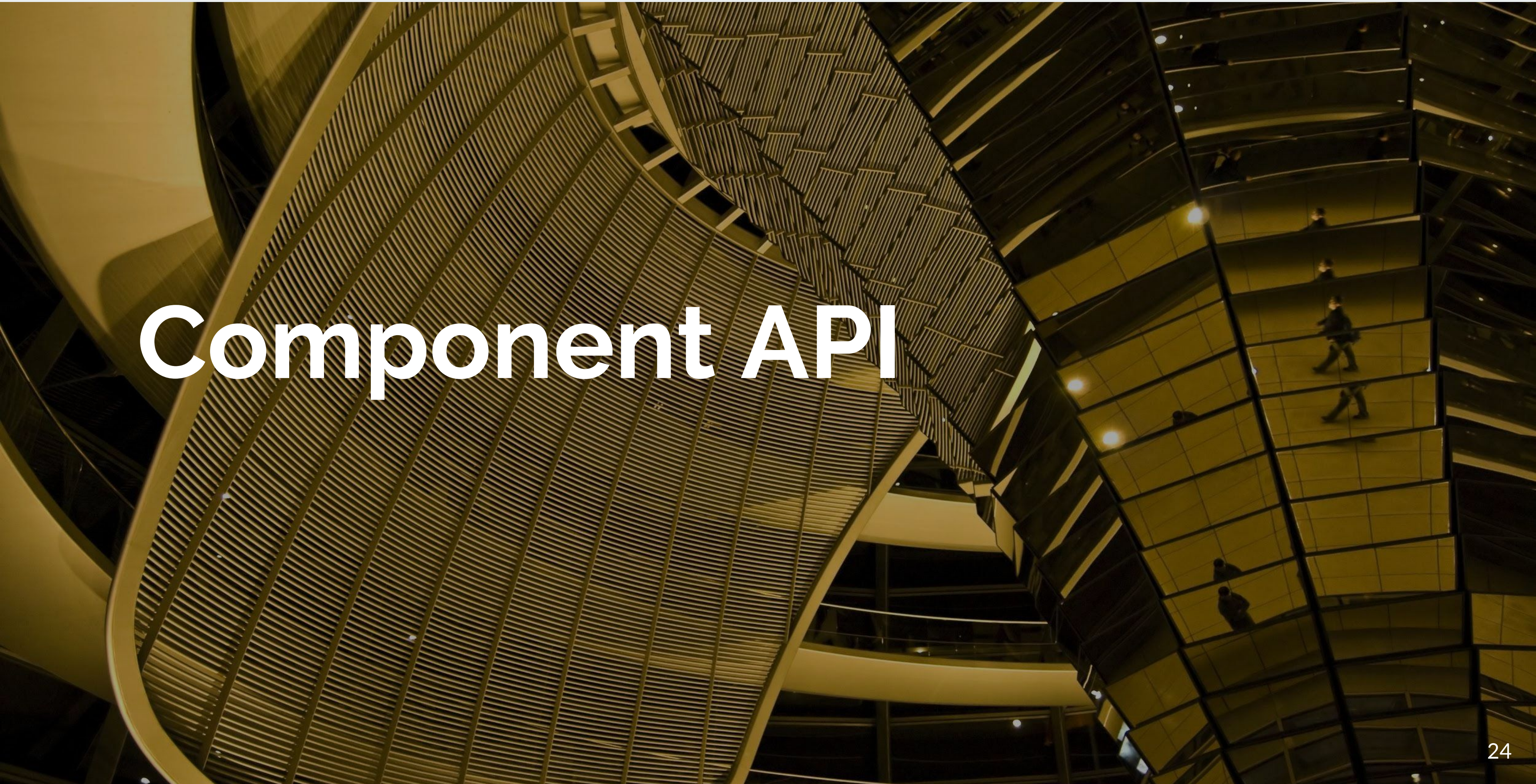
Finally ready for a serious business

Project snapshot could be deployed with

```
git clone --branch react-starter https://github.com/kos33rd/web-developer-course.git

cd web-developer-course/Week-3/project-example

npm i
```



Component API

Two types of Components

Functional

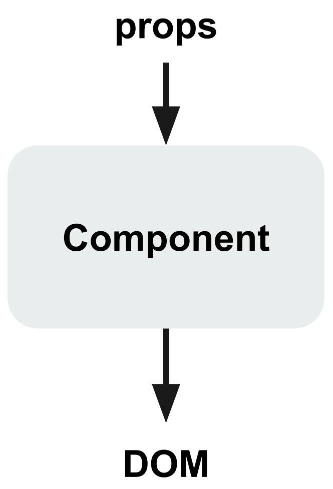
```
const Funcional = function (props) {
 return (
   <h1>{props.title}</h1>
 )
}
```

Class components

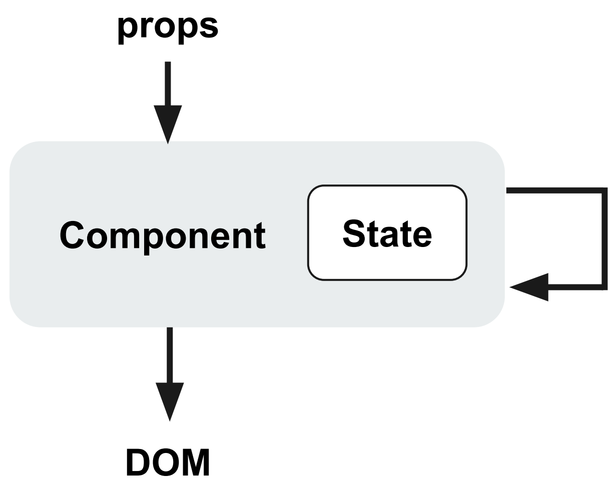
```
class ClassBased extends React.Component {
 render () {
   return (
     <h1>{this.props.title}</div>
   )
 }
}
```

Two types of Components

Functional



Class components



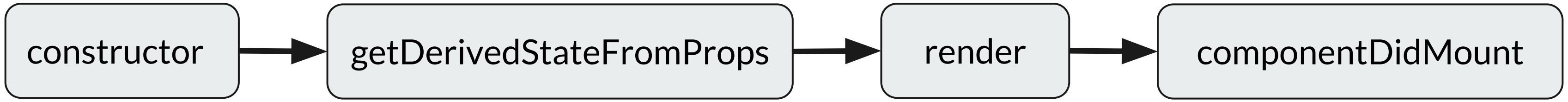
State

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

 render() {
   return (
     <button onClick={ () => this.setState({value: 'X'}) } >
       {this.state.value}
     </button>
   )
 }
}
```

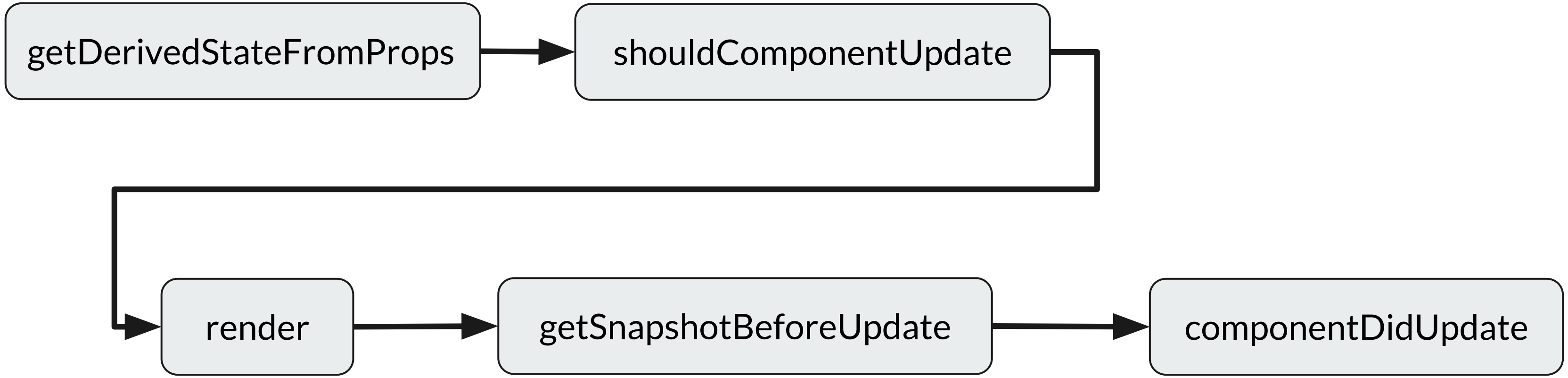
Component Lifecycle

First render (e.g. mounting)



Component Lifecycle

Props has been updated



Component Lifecycle

Component instance deletion(e.g. unmount)

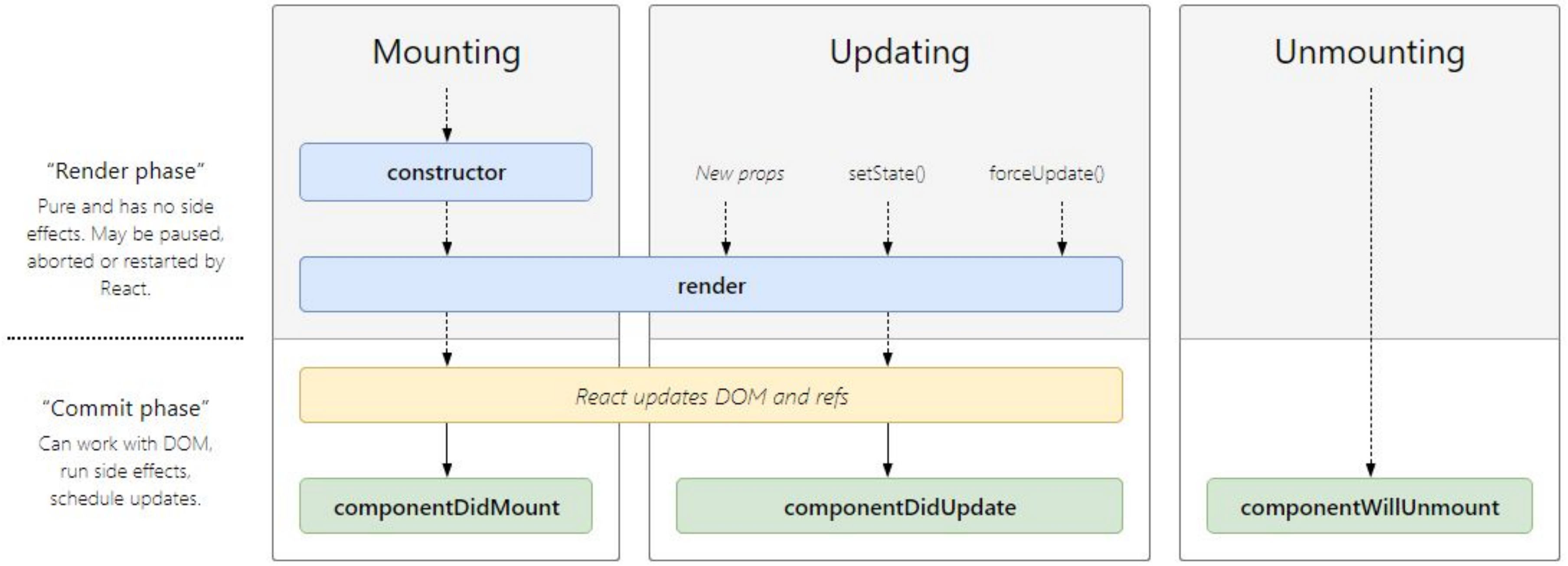
componentWillUnmount

Extra Component API

Additional component class fields and methods

- defaultProps
- displayName
- componentDidCatch

Cheat sheet


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

Assignment

Assignment for the next week

- Make a choice about component-based library or framework you will use in your project
- Set up your library / framework and bundler
- Set up routing library
- Add routes and basic layout to your application (according to your app UI)
- Make a PR

Links

React simple tutorial: https://reactjs.org/tutorial/tutorial.html

Bundler and Babel setup: https://www.valentinog.com/blog/react-webpack-babel/

Sample project scaffold:
https://github.com/kos33rd/web-developer-course/tree/react-starter/Week-3/project-example

React Lifecycle cheat sheet: http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

That's all!

Resources

- Lectures, resources and course project requirements:
  https://github.com/kos33rd/web-developer-course

- Our telegram group:
  https://t.me/JSforEntDev

- Github accounts to send PRs with complete tasks:
  @kos33rd, @AVVlasov

- News, announcements, resources and useful links:
  http://moodle.innopolis.university