**React.js**

Links: <https://reactjs.org/> and <https://github.com/facebook/react>

***Introduction .***

Hello everyone. My name is Irina. In this presentation, we’ll look at a cool Java Script library called React. We’ll get acquainted with its basics, find out what are the pros and cons. Here goes!

 React is a JavaScript library for building user interfaces.

 Renders your UI and responds to events

 It also uses the concept called Virtual DOM, creates an in-memory data structure cache, enumerates the resulting differences, and then updates the browser’s displayed DOM efficiently.

 One of the unique features of React.js is not only it can perform on the client side, but it can also be rendered on the server side, and they can work together interoperable

* UI

**Why was React developed?**

* Complexity of two-way data binding.
* Bad UX from using "cascading updates" of DOM tree
* A lot of data on a page changing over time
* Complexity of Facebook's UI architecture

**react is maintained by**

GitHub stats:

**WEBSITES USING REACT**

Websites worldwide made with react

**Why should You use React?**

* To read and understand views
* Concept of components is the *future* of web development
* If your page uses a lot of fast updating data or real time data - React is the way to go
* Once you and your team is over the React's learning curve, developing your app will become a lot faster

**React: the good**

**Easy to understand what a component will render**

* Code → predictable code
* You don't really need to study JS in the view file in order to understand what the file does

**Easy to mix HTML and JS**

* You do it already with template libraries (e.g. Handlebars, Mustache, Underscore etc.)

**No complex two-way data flow**

* Uses simple one-way reactive data flow
* Easier to understand than two-way binding
* Uses less code

**React is fast!**

* Real DOM is slow
* JavaScript is fast
* Using virtual DOM objects enables fast batch updates to real DOM, with great productivity gains over frequent cascading updates of DOM tree

**React: the "bad"**t that React is w.

**You DON'T GET any of the following:**

* Event system (other than vanilla DOM events)
* Any AJAX capabilities whatsoever
* Any form of a data layer
* Promises
* Any application framework

**Building JSX requires some extra work**

* But most of the work is already done for you by react-tools

**No support for older browsers**

* React won't work with IE8
* There some polyfills / shims that help

**Try React:**

**Online Playgrounds:**

[CodePen](https://codepen.io/pen?&editable=true&editors=0010), [CodeSandbox](https://codesandbox.io/s/new)

**Use Webpack or Create React App:**

[Webpack 4](https://www.valentinog.com/blog/babel/), [Create React App](https://github.com/facebook/create-react-app)

**Fundamentals**

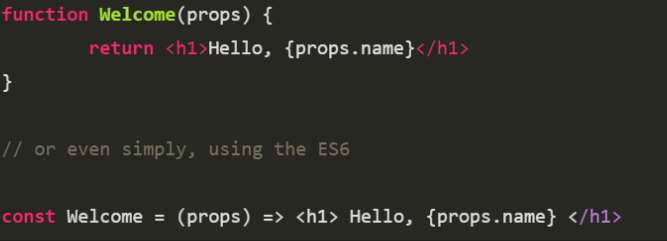
Most important terms in React

**Component**

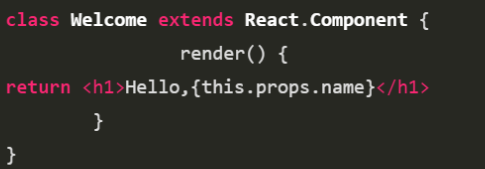
Components are self-contained reusable building blocks of web application. React components are basically just idempotent functions (same input produces same output). They describe your UI at any point in time, just like a server-rendered app.

**Component**

The simplest way to define a component is to write a JavaScript function:



You can also use an ES6 class to define a component:



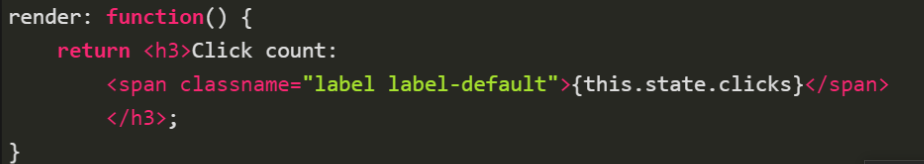
**Props**

* Passed down to component from parent component and represents data for the component
* accessed via **this.props**



**State**

* Represents internal state of the component
* Accessed via **this.state**
* When a component's state data changes, the rendered markup will be updated by re-invoking **render() method**



**JSX**

* Arguably, one of the coolest things in React
* Easier to read and understand large DOM trees
* Translates to plain JavaScript using react-tools
* XML-like syntax for generating component's HTML



**Virtual DOM**

* The virtual DOM is used for efficient re-rendering of the DOM
* React aims to re-render the virtual tree only when the state changes
* Uses 2 virtual trees (new and previous) to find differences and batch update real DOM
* Observes data changes (setState) and does dirty-checking to know when to re-render component
* Whenever possible, does not update entire component in real DOM - only computes a patch operation that updates part of the DOM

**Lifecycle react application**

[**More information**](http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/)

**Thanks for attention**