REACT

CORE CONCEPTS

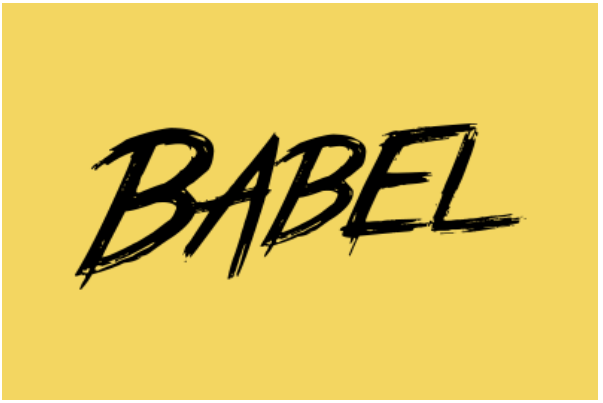
PART1

NOVEMBER 15, 2016

AGENDA

1. Intro to React library
2. React “Hello world”
3. What is React Element
4. JSX syntax
5. Intro to Components (props and state)

Prerequisites

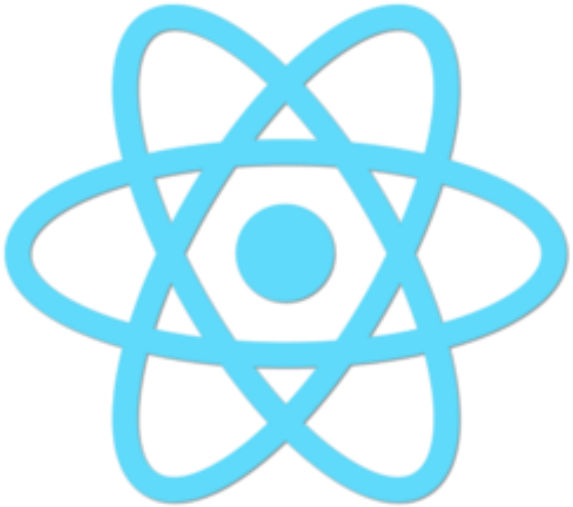


Certainly refresh ES5

- Function.prototype.bind()
- Array.prototype.map()/filter()/forEach()

Intro to React

What is React?

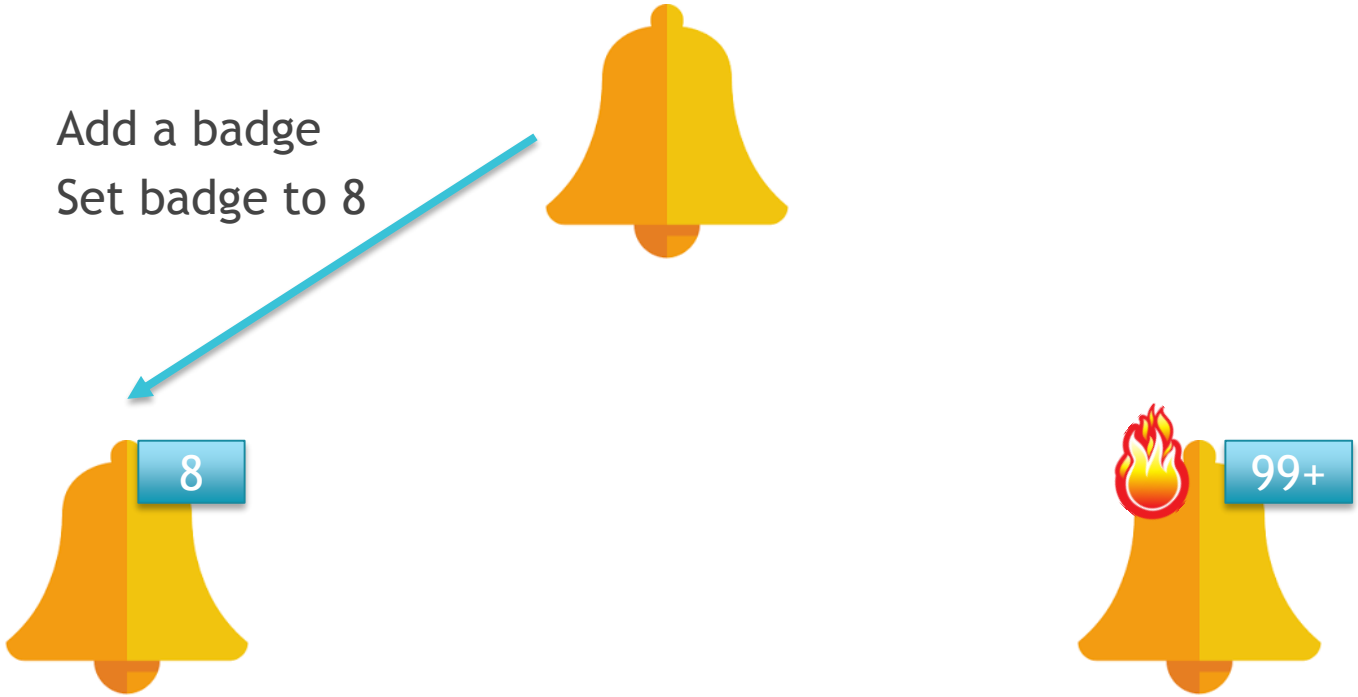


A JAVASCRIPT **LIBRARY** FOR BUILDING USER INTERFACES

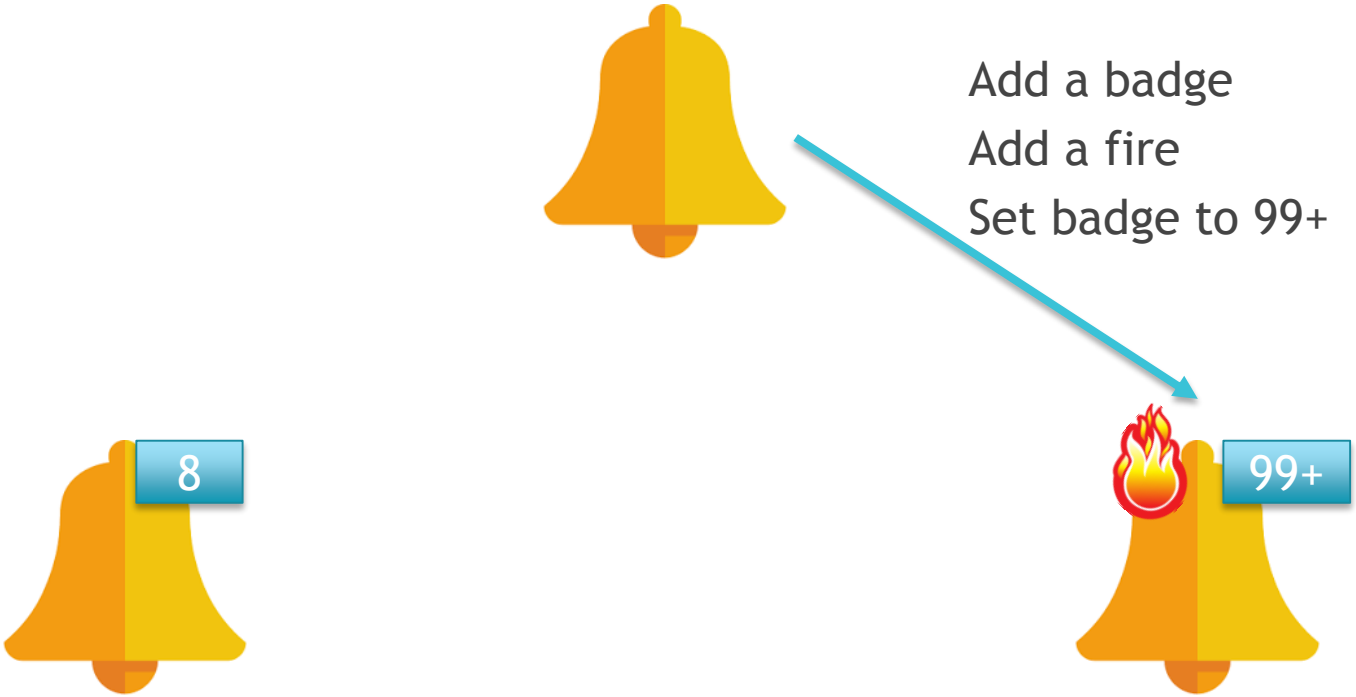
THE “**V**” IN **MVC**

Notification Widget

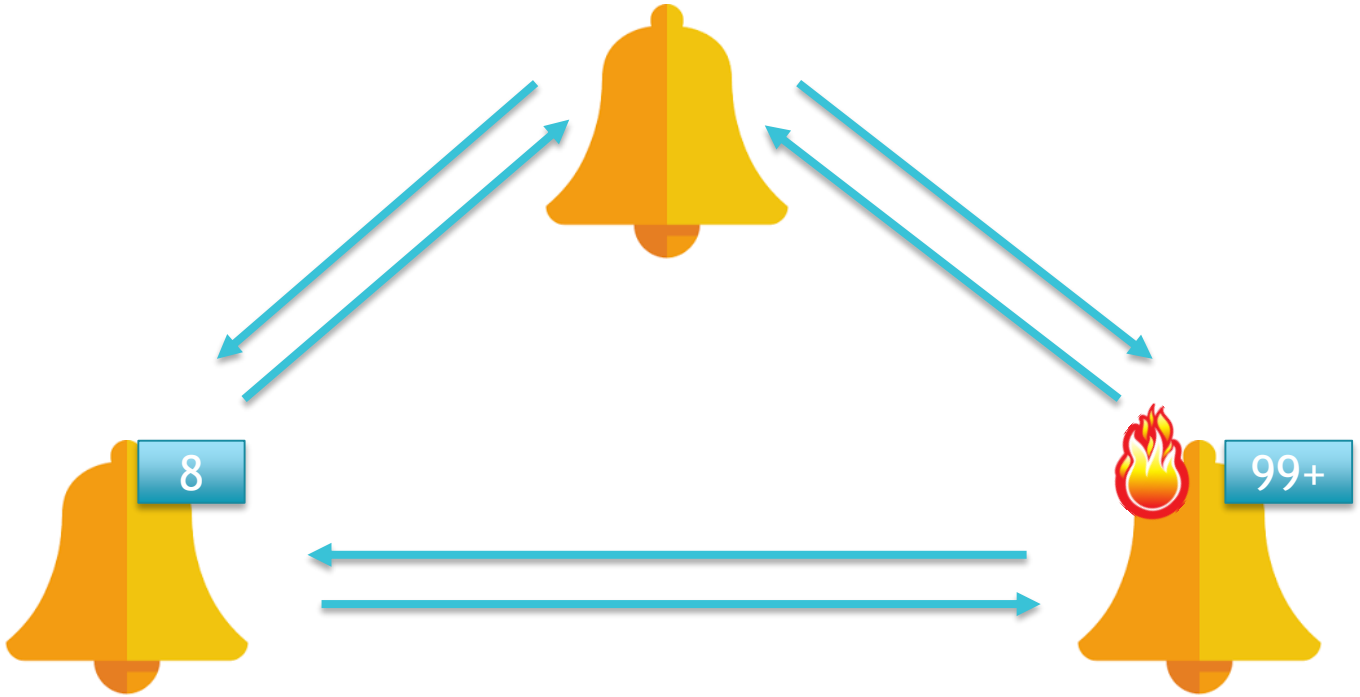
Notification Widget



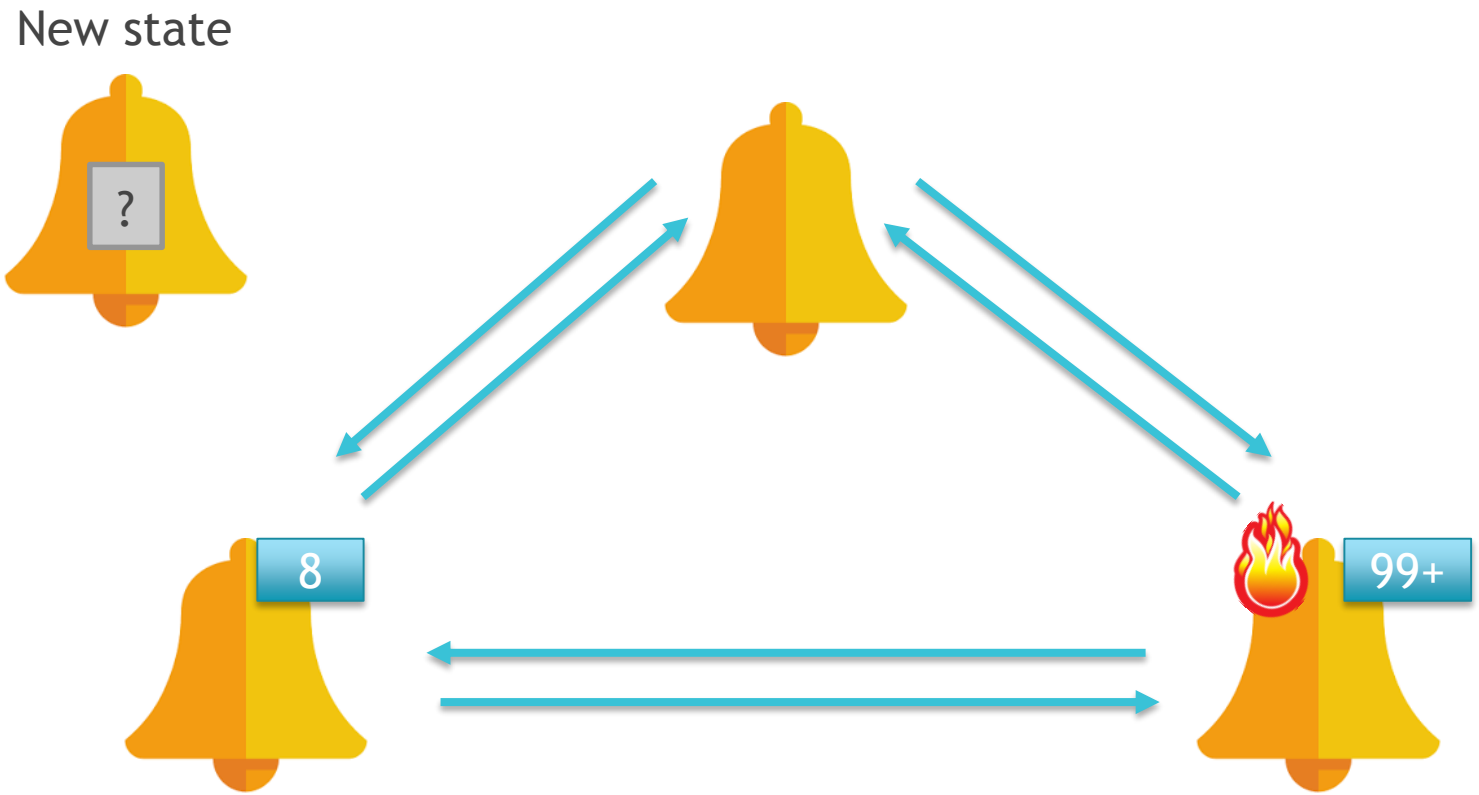
Notification Widget



Notification Widget

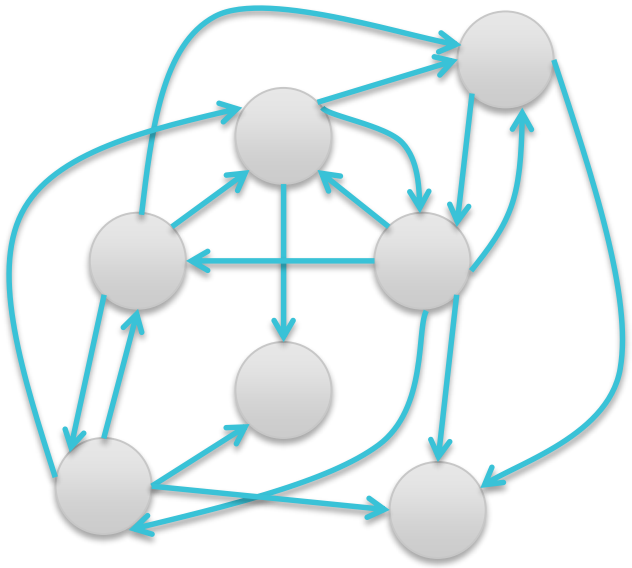


Notification Widget



State transition complexity

$$O(N*(N-1)) \rightarrow O(N^2-N)$$



States	State transitions
3	6
5	20
10	90
100	9900
500	249500

State transition complexity

O(N*(N-1)) ➔ O(N^2-N)

Mutation is hard

State transition complexity

O(N*(N-1)) ➔ O(N^2-N)

Mutation is hard

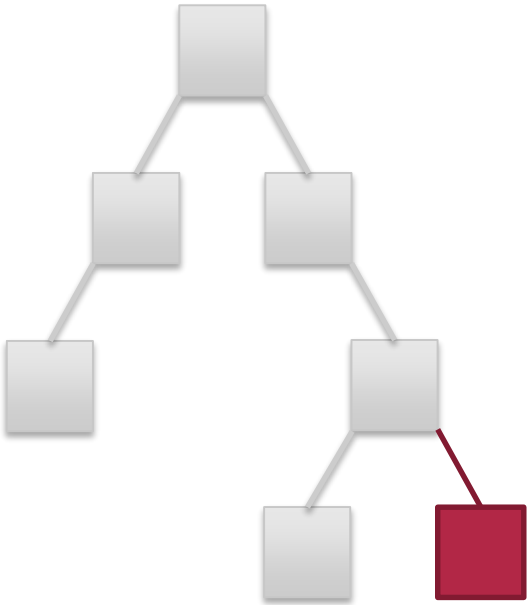
Let’s not do mutation!

React's Design

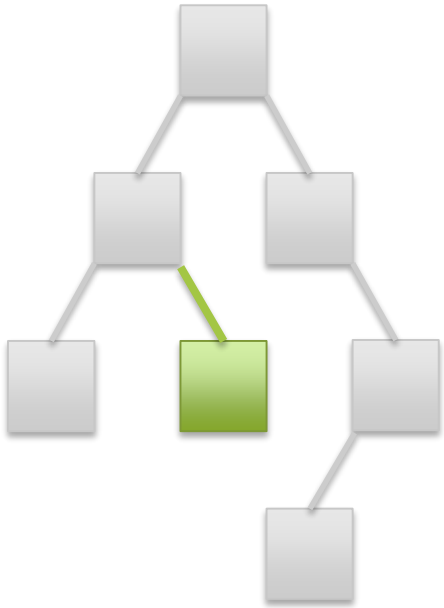
Rebuild the whole DOM, for every change

Sounds expensive

Virtual DOM



Virtual DOM t=1



Virtual DOM t=2

Mutations in real DOM

Benefits

1. Quite simple
2. Highly scalable
3. Good performance

Companies that use React

and others…

https://github.com/facebook/react/wiki/Sites-Using-React

React “Hello world”

React Installation

npm install --save **react react-dom**

```
import React from "react";
import ReactDOM from "react-dom"
```

I don’t want to use npm:

```
<script src= "https://unpkg.com/react@15/dist/react.js"></script>
<script src= "https://unpkg.com/react-dom@15/dist/react-dom.js"></script>
```

React Native Renderer
npm install --save react-native-renderer

Hello world

```
import React from "react";
import ReactDOM from "react-dom";


var element1 = React.createElement(
        "h1",
        {id:"id_001", className:"main_title"},
        "Hello World!“
);

ReactDOM.render(element1, document.getElementById("container"));
```

```
<!DOCTYPE html>
···<html> == $0
  ▶<head>…</head>
  ▼<body>
    ▼<div id="root">
      ▼<div id="container">
          <h1 data-reactroot id="id_001" class="my_title">Hello World!</h1>
        </div>
      </div>
```

Render into DOM

```
Import React from "react"
Import ReactDOM from "react-dom"


var element1 = React.createElement(
        "h1",
        {id:"id_001", className:"main_title"},
        "Hello World!“
);

ReactDOM.render(element1, document.getElementById("container"));
```

```
/*
* @param {ReactElement} nextElement element to render.
* @param {DOMElement} container DOM element to render into.
* @param {function} callback function triggered on completion.
*/

ReactDOM.render(nextElement, container, [callback]);
```

React Element

React Element

```
Import React from "react"
Import ReactDOM from "react-dom"

var element1 = React.createElement(
      "h1",
      {id:"id_001", className:"main_title"},
      "Hello World!“
);

ReactDOM.render(element1, document.getElementById("container"));
```

```
/*
* @param {string || function || class} type element type.
* @param {Object || null} config element properties.
* @param {string || Array} children element children.
*/

React.createElement(type, config, children);
```

Structure of React Element

```
var ReactElement = function(...) {
//...
      var element = {
//...
            type: type,  // "h1"
            props: {},    // {id:"id_001", className:"main_title", children:"Hello World!"}
            key: key,
            ref: ref,
//...
      };
//...
//ReactElement is immutable!
      Object.freeze(element.props);
      Object.freeze(element);

      return element;
};
```

Example with children

```
var elem1 = React.createElement(
        "h1",
        {id:"id_001", className:"my_title"},
        "Hello World!“
);

var elem2 = React.createElement(
        "div",
        {id:"id_002", className:"my_desc"},
        "Some description...“
);

var root = React.createElement("div",null,elem1,elem2);

ReactDOM.render(root, document.getElementById("container"));
```

```
<!DOCTYPE html>
<html> == $0
  <head>…</head>
  <body>
    <div id="root">
      <div id="container">
        <div data-reactroot>
          <h1 id="id_001" class="my_title">Hello World!</h1>
          <div id="id_002" class="my_desc">Some description...</div>
        </div>
      </div>
    </div>
```

Structure of React tree (the simplest VDOM)

```
var root={
      type: type,
      key: key,
      ref: ref,
      props:{
            children:[
                  elem1 = {…},
                  elem2 = {
                        type: type,
                        key: key,
                        ref: ref,
                        props:{
                              //..
                              children:[…]
                              //..
                        }
                  }
            ]
      }
}
```

Updating the Rendered Element

```
function tick(){

        const root = React.createElement(
                "div",
                null,
                React.createElement("h1",null,"Hello, World!"),
                React.createElement(“h2",null, ("It is " + new Date()))
        );

        ReactDOM.render(root, document.getElementById(“root"));
}

setInterval(tick,1000);
```

Hello, world!

It is 12:26:46 PM.

Console Sources Network Timeline

```
▼<div id="root">
  ▼<div data-reactroot>
      <h1>Hello, world!</h1>
    ▼<h2>
        <!-- react-text: 4 -->
        "It is "
        <!-- /react-text -->
        <!-- react-text: 5 -->
        "12:26:46 PM"
        <!-- /react-text -->
        <!-- react-text: 6 -->
        "."
        <!-- /react-text -->
      </h2>
    </div>
</div>
```

HTML vs React.createElement()

ReactElement tree

```
React.createElement("div", null,
React.createElement( "h1", null, "Title"),
React.createElement("div", {className: "child1" },
React.createElement("div", { className: "child1_1" },
"Sometext"), React.createElement("div", {
className: "child1_2" }, "Sometext")),
React.createElement("div", {className:"child2"},
React.createElement("ul", null,
React.createElement("li", null, "1"),
React.createElement("li", null, "2"),
React.createElement("li", null, "3"))));
```

HTML

```
<div>
  <h1>Title</h1>
  <div class ="child1">
    <div class = "child1_1">Some text</div>
    <div class = "child1_2">Some text</div>
  </div>
  <div class = "child2">
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
    </ul>
  </div>
</div>
```

JSX syntax

Why do we need JSX?

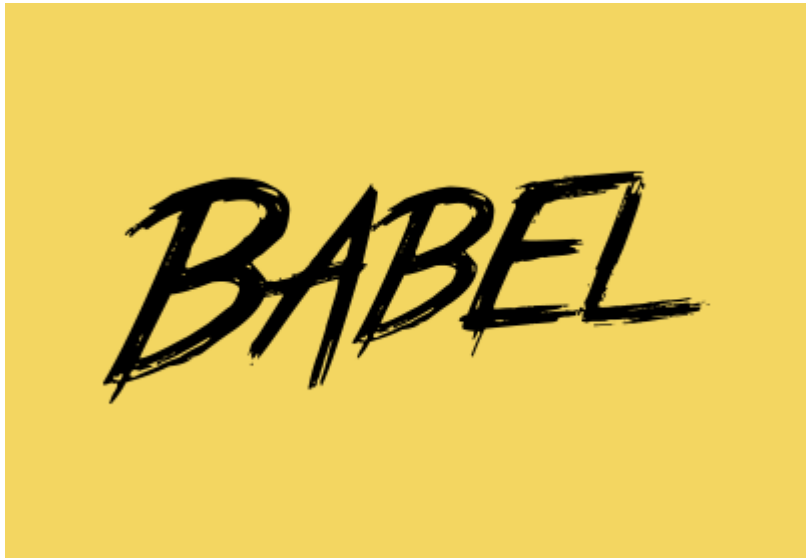
JSX

```
var elem = <div>
  <h1>Title</h1>
  <div className ="child1">
    <div className = "child1_1">Some text</div>
    <div className = "child1_2">Some text</div>
  </div>
  <div className = "child2">
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
    </ul>
  </div>
</div>
```

HTML

```
<div>
  <h1>Title</h1>
  <div class ="child1">
    <div class = "child1_1">Some text</div>
    <div class = "child1_2">Some text</div>
  </div>
  <div class = "child2">
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
    </ul>
  </div>
</div>
```

JSX → JS



React preset

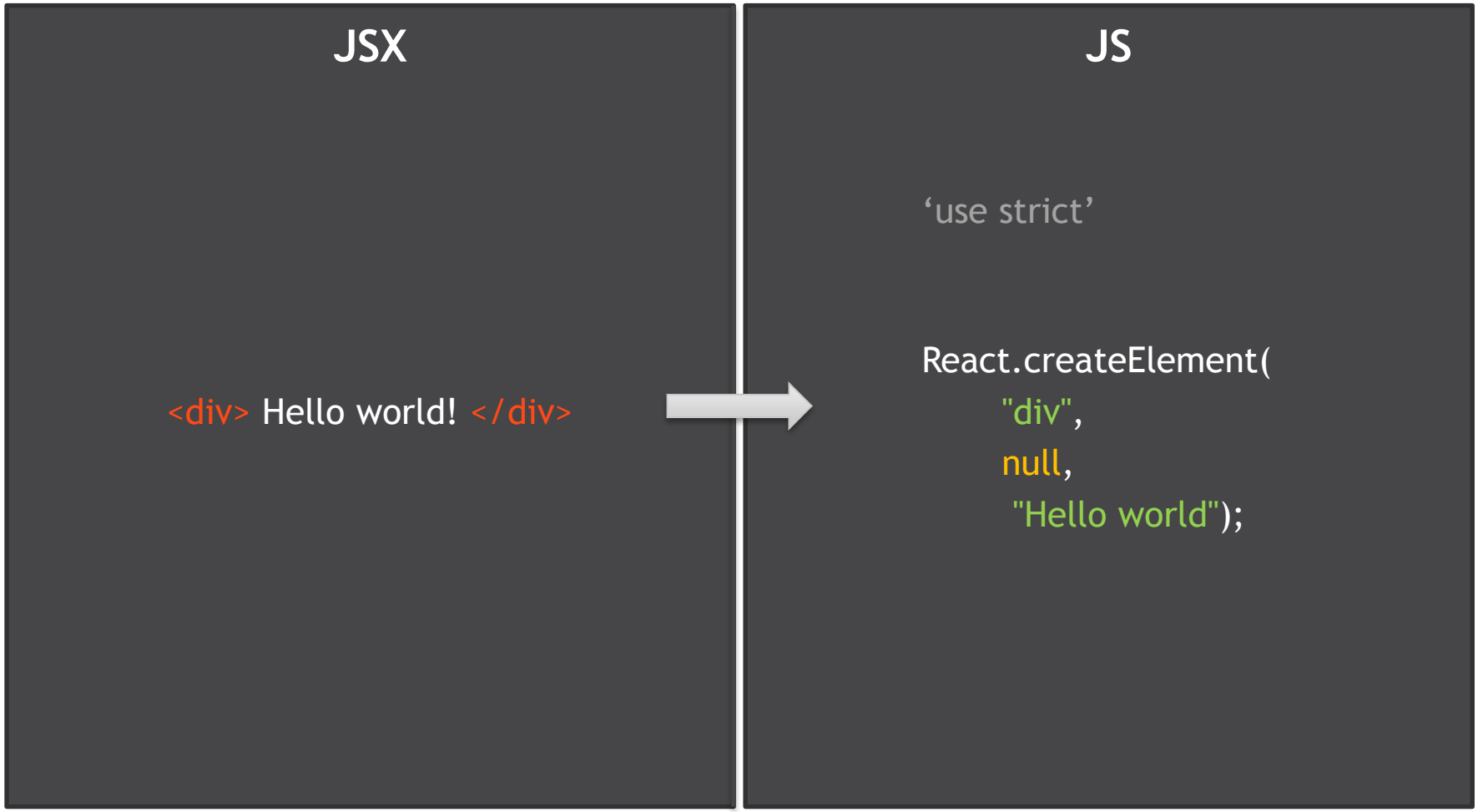
```
$ npm install -save -dev babel-preset-react
{
      “presets”: [“react”]
}
```

Simple way for dev purposes

```
<script src="https://unpkg.com/babel-core@5.8.38/browser.min.js"></script>

<script type="text/babel" src=“.jsx"></script>
```

JSX Basics


JSX
 Hello world!
JS
'use strict'
React.createElement(
"div",
null,
"Hello world");

JSX Tricks

JSX

```
<div>
</div>
<div>
</div>
```

Console


```
<div></span>
```

JSX is an Expression

```
function getGreeting(user){
      if (user) {
            return <h1>Hello, Friend</h1>
      }
      return <h1>Hello, Stranger</h1>
}
```

Javascript Expression in JSX

```
function getGreeting(user){
      if (user) {
            return <h1>Hello, {formatName(user)}</h1>
      }
      return <h1>Hello, Stranger</h1>
}
```

Javascript Expression in JSX

JSX

```
const planet = "Earth";

<div>
    Hello {planet}!
</div>
```

JS

```
const planet = "Earth";
React.createElement(
"div",
null,
"Hello",
planet,
 "!");
```

Javascript Statement in JSX

JSX

```
<div>
    {if (true) return "Hello"}
</div>
```

Console

```
▶ Uncaught SyntaxError: Unexpected token (2:2)
<div>
    {if (true) return “Hello”}
</div>
```

Javascript Conditional Operator in JSX

JSX

```
<div>
    {true? "Hello" :null}
</div>
```

JS

```
React.createElement(
    "div",
    null,
    true ? "Hello" : null
);
```

String literals and default value

JSX

```
<div className={"my_best_class"}></div>
<div className= "my_best_class" ></div>

<div className></div>
```

JS

```
React.createElement(
"div",
{ className: "myClass" });

React.createElement(
"div",
 { className: true });
```

Booleans, Null, and Undefined Are Ignored

```
<div/>

<div></div>

<div>{false}</div>

<div>{null}</div>

<div>{true}</div>


<div>
    {showHeader && <div>Header</div> }
    <div>Content</div>
</div>
```

Rendering a list of JSX expressions

```
var data = ["item1", "item2", "item3", "item4", "item5"];

var list =
        <ul>
            {data.map((elem) => <li>{elem}</li>)}
        </ul>

ReactDOM.render(list, document.getElementById("container"));
```

```
···<!DOCTYPE html> == $0
  <html>
  ▶<head>…</head>
  ▼<body>
    ▼<div id="root">
      ▼<div id="container">
        ▼<ul data-reactroot>
            <li>item1</li>
            <li>item2</li>
            <li>item3</li>
            <li>item4</li>
            <li>item5</li>
          </ul>
        </div>
      </div>
```

Console

```
▶Warning: Each child in an array or iterator should have a unique "key"    react.js:20478
prop. Check the top-level render call using <ul>. See https://fb.me/react-warning-keys for
more information.
    in li
```

JSX tricks (common mistake)

JSX

```
<div>
    {
    comments.forEach((comment)=>{
        return
        <div>
            {comment}
        </div>
    })
}
</div>
```

JS

```
React.createElement(
    "div",
    null,
    Comments.forEach(
        function(comment){
            return;
            React.createElement(
                "div",
                null,
                comment
            );
        }
    )
);
```

JSX tips

JSX

```
<div>
      {
      comments.forEach((comment)=>{
            return (
                  <div>
                        {comment}
                  </div>
            );
      })
}
</div>
```

JS

```
React.createElement(
      "div",
      null,
      Comments.forEach(
            function(comment){
                  return React.createElement(
                        "div",
                        null,
                        comment
                  );
            })
);
```

JSX and styles

JSX

```
var divStyle = {
        color: "white",
        background : "red",
        WebkitTransition: "all",
        msTransition: "all"
};

ReactDOM.render(
<div style={divStyle}>Hello world!</div>,
document.getElementById("container"));
```

HTML

```
<div id="container">
        <div style = "transition: all;
                      color: white;
                      background-color: red;">
                Hello world!
        </div>
</div>
```

All Supported HTML Attributes

accept acceptCharset accessKey action allowFullScreen allowTransparency alt async autoComplete autoFocus autoPlay capture cellPadding cellSpacing challenge charSet checked cite classID className colSpan cols content contentEditable contextMenu controls coords crossOrigin data dateTime default defer dir disabled download draggable encType form formAction formEncType formMethod formNoValidate formTarget frameBorder headers height hidden high href hrefLang htmlFor httpEquiv icon id inputMode integrity is keyParams keyType kind label lang list loop low manifest marginHeight marginWidth max maxLength media mediaGroup method min minLength multiple muted name noValidate nonce open optimum pattern placeholder poster preload profile radioGroup readOnly rel required reversed role rowSpan rows sandbox scope scoped scrolling seamless selected shape size sizes span spellCheck src srcDoc srcLang srcSet start step style summary tabIndex target title type useMap value width wmode wrap

Handling events

JSX

```
<button onClick = {activeLasers}>
      Active Lasers
</button>
```

HTML

```
<button onclick = “activeLasers()”>
      Active Lasers
</button>
```

Components

Simple functional component

```
function Welcome(props) {
      return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Andrey"/>;

ReactDOM.render(element, document.getElementById("container"));
```

Simple functional component

```
function Welcome(props) {
      return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name = "Andrey"/>;

ReactDOM.render(element, document.getElementById("container"));
```

Simple functional component

```
function Welcome(props) {
      return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Andrey"/>;

ReactDOM.render(element, document.getElementById("container"));
```

Rendering a Component

1. We call ReactDOM.render() with <Welcome name="Andrey"> element;
2. React calls the Welcome component with {name:'Andrey'} as the props;
3. Our Welcome component returns a <h1>Hello, Andrey</h1> element as the result.
4. React DOM efficiently updates the DOM to match <h1>Hello, Andrey</h1>

Composing Components

```
function Welcome(props) {
      return <h1> Hello, {props.name} </h1>;
}

function App(){
      return(
      <div>
            <Welcome name ="Andrey" />
            <Welcome name ="Ivan" />
            <Welcome name = "Olga" />
      </div>
      );
}

ReactDOM.render(<App/>,
document.getElementById("container"));
```

DOM

```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
  ▼<div id="root">
    ▼<div id="container">
      ▼<div data-reactroot>
        ▶<h1>…</h1>
        ▶<h1>…</h1>
        ▶<h1>…</h1>
        </div>
      </div>
    </div>
```

Composing Components

```
var data = ["Andrey", "Ivan", "Olga"];

function Welcome(props) {
      return <h1> Hello, {props.name} </h1>;
}

function App(){
      return(
            <div>
                  {data.map(elem=><Welcome name={elem}/>)}
            </div>
      );
}

ReactDOM.render(<App/>, document.getElementById("container"));
```

Class Component

```
class App extends React.Component {
        constructor(props){
                super(props);
                this.state = {data: ["Andrey", "Ivan", "Olga"]}
        }
        render(){
                return (
                <div>
                        {this.state.data.map(elem=><Welcome name={elem}/>)}
                </div>
                );
        }
}
ReactDOM.render(<App/>, document.getElementById("container"));
```

How to update Class Component

```
class App extends React.Component {
        constructor(props){
                super(props);
                this.state = {data: ["Andrey", "Ivan", "Olga"]}
        }
        handleClick(){
                const names = this.state.data;
                names.push('New Name');
                this.setState({data: names});
        }
        render(){
                return (
                <div>
                        {this.state.data.map(elem=><Welcome name={elem}/>)}
                        <button onClick={this.handleClick.bind(this)}>Add Name</button>
                </div>
                );
        }
}
ReactDOM.render(<App/>, document.getElementById("container"));
```

Updating of React App

ES6 and ES5 in Components

ES5

```
var Welcome = React.createClass({
        getInitialState: function() {
                return {date: new Date()};
        },

        handleClick: function(){
                this.setState ({date: new Date()});
        },

        render: function(){
        return (
        <div>
                <h1>Helloo, {this.props.name}</h1>
                <h2>It is {this.state.date}</h2>
                <button
onClick={this.handleClick}>Refresh</button>
        </div>
                );
        }
});
```

ES6

```
class Welcome extends React.Component {
        constructor(props){
                super(props);
                this.state = {date: new Date()}
        }
        handleClick(){
                this.setState({date:new Date()});
        }
        render(){
        return (
        <div>
                <h1>Hello, {this.props.name}</h1>;
                <h2>It is {this.state.date}</h2>
                <button
onClick={this.handleClick.bind(this)}>Refresh</button>
        </div>
                );
        }
}
// or you can use arrow function
//... onClick={()=>this.handleClick()} ...
```

THANK YOU!

QUESTIONS?