# TypeScript для PHP разработчика

## Как писать на JavaScript большие приложения и не сойти с ума

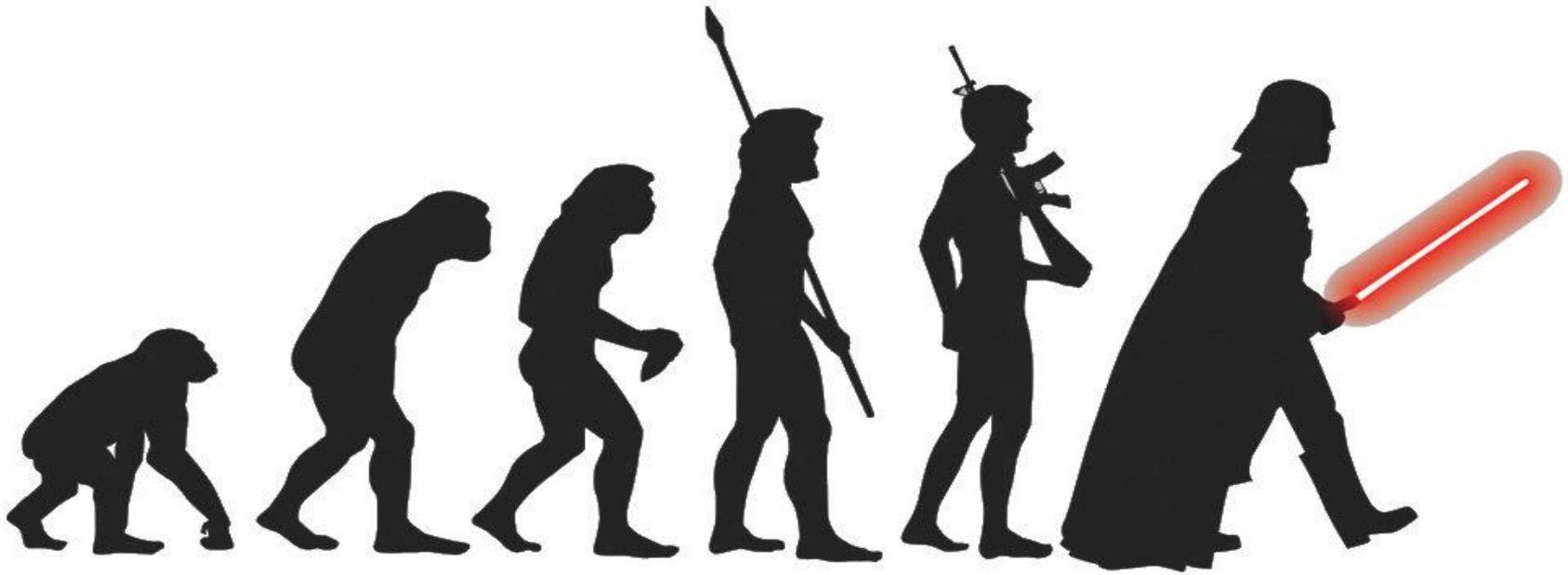
Александр Майоров

tutu.ru

DevConf

http://www.devconf.ru

# Эволюция JavaScript



# JavaScript compilers

Список всевозможных трансляторов и диалектов:goo.gl/EAabNP (340+)

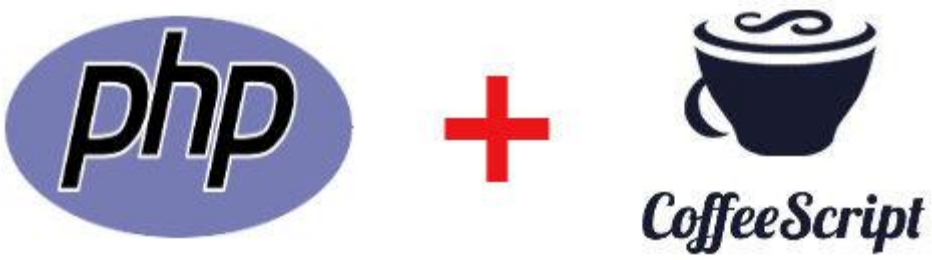

- CoffeeScript family: 12
- CoffeeScript friends (philosophically related): 25
- Static typing: 16 (TypeScript, Dart, Typecast.js, AtScript, ...)
- X-language compilers: **PHP**, Java, Ruby, Python, Erlang, Perl, Lua, Scala, C#, F#, Lisp, Scheme, OCaml, Haskell, Smalltalk, C/C++, Go, ...


CoffeeScript

:)


ruby
CoffeeScript

# 0_o



=)


php
+
TypeScript

# Дело не в кофе, а в сахаре

- CoffeeScript – сахар с фишками ES+ в стиле Ruby
- TypeScript – ES6+ с фичами и кунг фу c#


TypeScript
Светлое будущее ES6+ уже вчера

# Почему выбрали TypeScript?

- Статическая типизация
- Class, public, private, protected, static...
- Interface, enum, tuple, const, ...
- TypeScript - это и есть ES6+
- Режимы транспилирования: ES3, ES5, ES6
- Поддержка модулей: Commonjs, AMD (RequireJS)


# Мифы о TypeScript

# Мифы о TypeScript

- Закрытый исходный код
- Корректно работает только с Windows
- Разрабатывать можно только в VisualStudio
- Это другой язык, как и CoffeeScript
- Генерит лишний код
- Это же микрософт =(

# ~~Закрытый исходный код~~

- Исходники по адресу: https://github.com/Microsoft/TypeScript/

~~Корректно работает только с Windows~~

```
01. npm install -g typescript
02. tsc my_application.ts
```

# ~~Разрабатывать можно только в Visual Studio~~

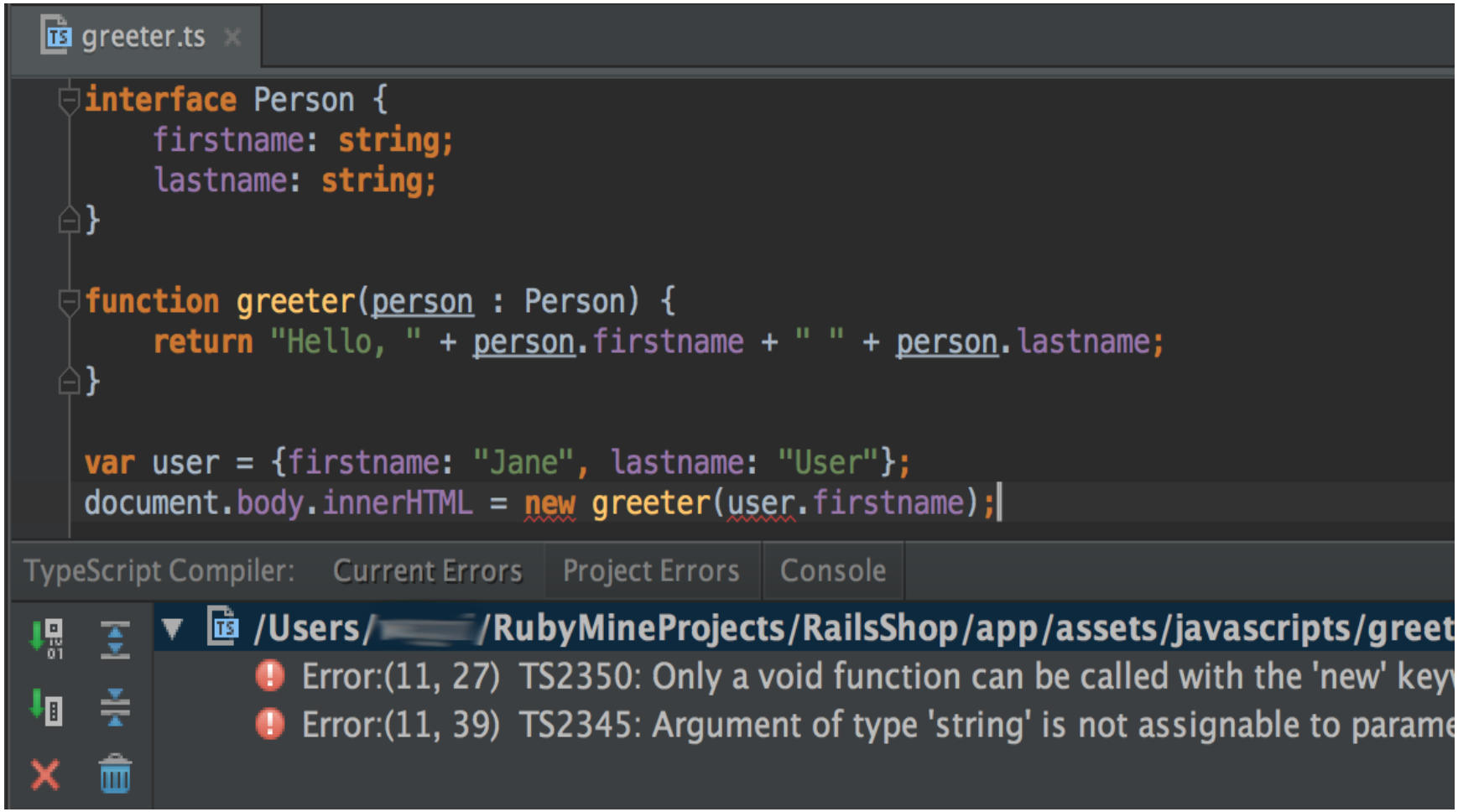
- ★ PhpStorm, WebStorm, RubyMine, ...
- ☆ Sublime Text (2,3)
- Atom (atom.io)
- TypeEcs for Eclips (typecsdev.com)

# Способы работы с TypeScript

- Консольный компилятор tsc
- Плагины для grunt, gulp, webpack, ...
- FileWatchers in IDE (PhpStorm, Sublime, Atom …)
- **Builtin TypeScript Compiler in PhpStorm**, WebStorm, …

# TypeScript Compiler

File → Settings → Language & Frameworks → TypeScript → Enable TypeScript Compiler



## ~~Это другой язык, как и CoffeeScript~~

```
let say =  (def:string):string  => `TypeScript is ${def}!`;

say('pretty amazing')
```

## ~~Генерит лишний код~~

```
var say =  (def:string):string  => `TypeScript is ${def}!`;

say('pretty amazing');
```

Результат:

```
var say = function (def) { return ("TypeScript is " + def + "!"); };

say('pretty amazing');
```

# Тот же результат на CoffeeScript

```
(function() {

  var say;

  say = (function(_this) {

    return function(def) {

      return "TypeScript is " + def + "!";

    };

    })(this);

  say('pretty amazing');

}).call(this);
```

# TypeScript vs CoffeeScript

```
var gvstr;
var _this = this; // Babel делает так же
var say = function (def) {

    return _this.gvstr = "T..." + def + "!"
};

say('pretty amazing');
```

```
(function() {
    var say;
    say = (function(_this) {
        return function(def) {
            return "T..." + def + "!";
            };
    })(this);
    say('pretty amazing');
}).call(this);
```

## ~~Генерит лишний код~~

```
enum Color { Red, Green, Blue }

var color :Color = Color.Green;
```

## ~~Генерит лишний код~~

```
var Color;

(function (Color) {

    Color[Color["Red"]  = 0] = "Red";

    Color[Color["Green"]= 1] = "Green";

    Color[Color["Blue"] = 2] = "Blue";

})(Color || (Color = {}));


var color = 1;
```

## ~~Генерит лишний код~~

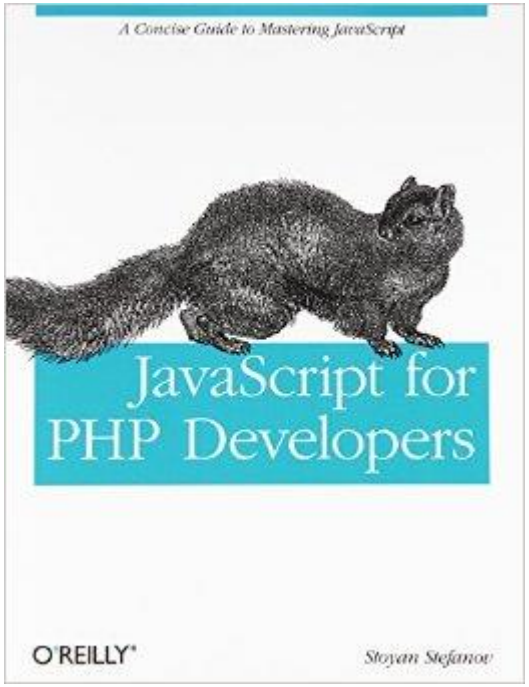
```
const enum Color { Red, Green, Blue }

var color :Color = Color.Green;
```

Результат:

```
var c = 1;
```

# ~~Java~~TypeScript для PHP разработчика

TypeScript для PHP разработчика такой же родной язык как CoffeeScript для Ruby девелопера



```php
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static $someFlag = 1;
	protected $_seatsCount = 0;

	function __construct($i) {
		$this->setSeatsCount($i);
		var_dump('Call Train');
	}
	public function getSeatsCount() {
		return $this->_seatsCount;
	}
	public function setSeatsCount($i) {
		return $this->_seatsCount = $i;
	}
}

class Sapsan extends Train {
	function __construct() {
		parent::__construct(59);
		var_dump('Call Sapsan');
	}
}

$Sapsan1 = new Sapsan();
var_dump($Sapsan1->getSeatsCount());
var_dump(Sapsan::$someFlag);
```

```php
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static $someFlag = 1;
	protected $_seatsCount = 0;

	function __construct($i) {
		$this->setSeatsCount($i);
		var_dump('Call Train');
	}
	public function getSeatsCount() {
		return $this->_seatsCount;
	}
	public function setSeatsCount($i) {
		return $this->_seatsCount = $i;
	}
}

class Sapsan extends Train {
	function __construct() {
		parent::__construct(59);
		var_dump('Call Sapsan');
	}
}

$Sapsan1 = new Sapsan();
var_dump($Sapsan1->getSeatsCount());
var_dump(Sapsan::$someFlag);
```

```javascript
var __extends = (this && this.__extends) || function (d, b) {
    for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p];
    function __() { this.constructor = d; }
    __.prototype = b.prototype;
    d.prototype = new __();
};
var var_dump = console.log.bind(this);
var TRAIN_TYPE_A = 1;
var TRAIN_TYPE_B = 2;
var Train = (function () {
    function Train(i) {
        this._seatsCount = 0;
        this.setSeatsCount(i);
        var_dump('Call Train');
    }
    Train.prototype.getSeatsCount = function () {
        return this._seatsCount;
    };
    Train.prototype.setSeatsCount = function (i) {
        return this._seatsCount = i;
    };
    Train.someFlag = 1;
    return Train;
})();
var Sapsan = (function (_super) {
    __extends(Sapsan, _super);
    function Sapsan() {
        _super.call(this, 59);
        var_dump('Call Sapsan');
    }
    return Sapsan;
})(Train);
```

# Не не, я готов писать на чем-то другом

# Но что если JavaScript был бы такой же родной как и ...

# Ищем 10 отличий

```php
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static $someFlag = TRAIN_TYPE_B;
	protected $_seatsCount = 0;

	function __construct($i) {
		$this->setSeatsCount($i);
		var_dump('Call Train');
	}
	public function getSeatsCount() {
		return $this->_seatsCount;
	}
	public function setSeatsCount($i) {
		return $this->_seatsCount = $i;
	}
}

class Sapsan extends Train {
	function __construct() {
		parent::__construct(59);
		var_dump('Call Sapsan');
	}
}

$Sapsan1 = new Sapsan();
var_dump($Sapsan1->getSeatsCount());
var_dump(Sapsan::$someFlag);
```

```php
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static $someFlag = TRAIN_TYPE_B;
	protected $_seatsCount = 0;

	constructor($i) {
		this.setSeatsCount($i);
		var_dump('Call Train');
	}
	public getSeatsCount() {
		return this.$_seatsCount;
	}
	public setSeatsCount($i) {
		return this.$_seatsCount = $i;
	}
}

class Sapsan extends Train {
	constructor() {
		super(59);
		var_dump('Call Sapsan');
	}
}

var $Sapsan1 = new Sapsan();
var_dump($Sapsan1.getSeatsCount());
var_dump($Sapsan.someFlag);
```

# После рефакторинга

```php
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static $someFlag = TRAIN_TYPE_B;
	protected $_seatsCount = 0;

	function __construct($i) {
		$this->setSeatsCount($i);
		var_dump('Call Train');
	}
	public function getSeatsCount() {
		return $this->_seatsCount;
	}
	public function setSeatsCount($i) {
		return $this->_seatsCount = $i;
	}
}

class Sapsan extends Train {
	function __construct() {
		parent::__construct(59);
		var_dump('Call Sapsan');
	}
}

$Sapsan1 = new Sapsan();
var_dump($Sapsan1->getSeatsCount());
var_dump(Sapsan::$someFlag);
```

```typescript
const TRAIN_TYPE_A = 1;
const TRAIN_TYPE_B = 2;

class Train {
	public static someFlag: number = TRAIN_TYPE_B;
	protected _seatsCount: number = 0;

	constructor(i: number) {
		this.setSeatsCount(i: number);
		var_dump('Call Train');
	}
	public getSeatsCount(): number {
		return this._seatsCount;
	}
	public setSeatsCount(i: number): number {
		return this._seatsCount = i;
	}
}

class Sapsan extends Train {
	constructor() {
		super(59);
		var_dump('Call Sapsan');
	}
}

var Sapsan1:Sapsan = new Sapsan();
var_dump(Sapsan1.getSeatsCount());
var_dump(Sapsan.someFlag);
```

# Type hinting

```php
// PHP 7

function add(int $a, int $b): int {
    return $a + $b;
}
```

```typescript
type int = number;

function add(a: int, b: int): int {
    return a + b;
}
```

# Интерфейсы

```php
interface iTemplate {
    function setVariable($key, $val);
    function getHtml($template);
}

class Template implements iTemplate {
    private $vars = [];

    public function setVariable($key, $val) {
        $this->vars[$key] = $val;
    }
    public function getHtml($template) {
        $tpl = '';
        // some do with template
        return $tpl;
    }
}
```

```javascript
interface iTemplate {
    setVariable(key, val);
    getHtml(template);
}

class Template implements iTemplate {
    private vars = {};

    public setVariable(key, val) {
        this.vars[key] = val;
    }
    public getHtml(template) {
        var tpl;
        // some do with template
        return tpl;
    }
}
```

# Интерфейсы

```php
interface iTemplate {
    function setVariable($key, $val);
    function getHtml($template);
}

class Template implements iTemplate {
    private $vars = [];

    public function setVariable($key, $val) {
        $this->vars[$key] = $val;
    }
    public function getHtml($template) {
        $tpl = '';
        // some do with template
        return $tpl;
    }
}
```

```typescript
interface iTemplate {
    setVariable(key:string, val:any): void;
    getHtml(template:string): string;
}

class Template implements iTemplate {
    private vars:Object = {};

    public setVariable(key:string, val:any): void {
        this.vars[key] = val;
    }
    public getHtml(template:string): string {
        var tpl;
        // some do with template
        return tpl;
    }
}
```

# Если вы разработчик на }{ak

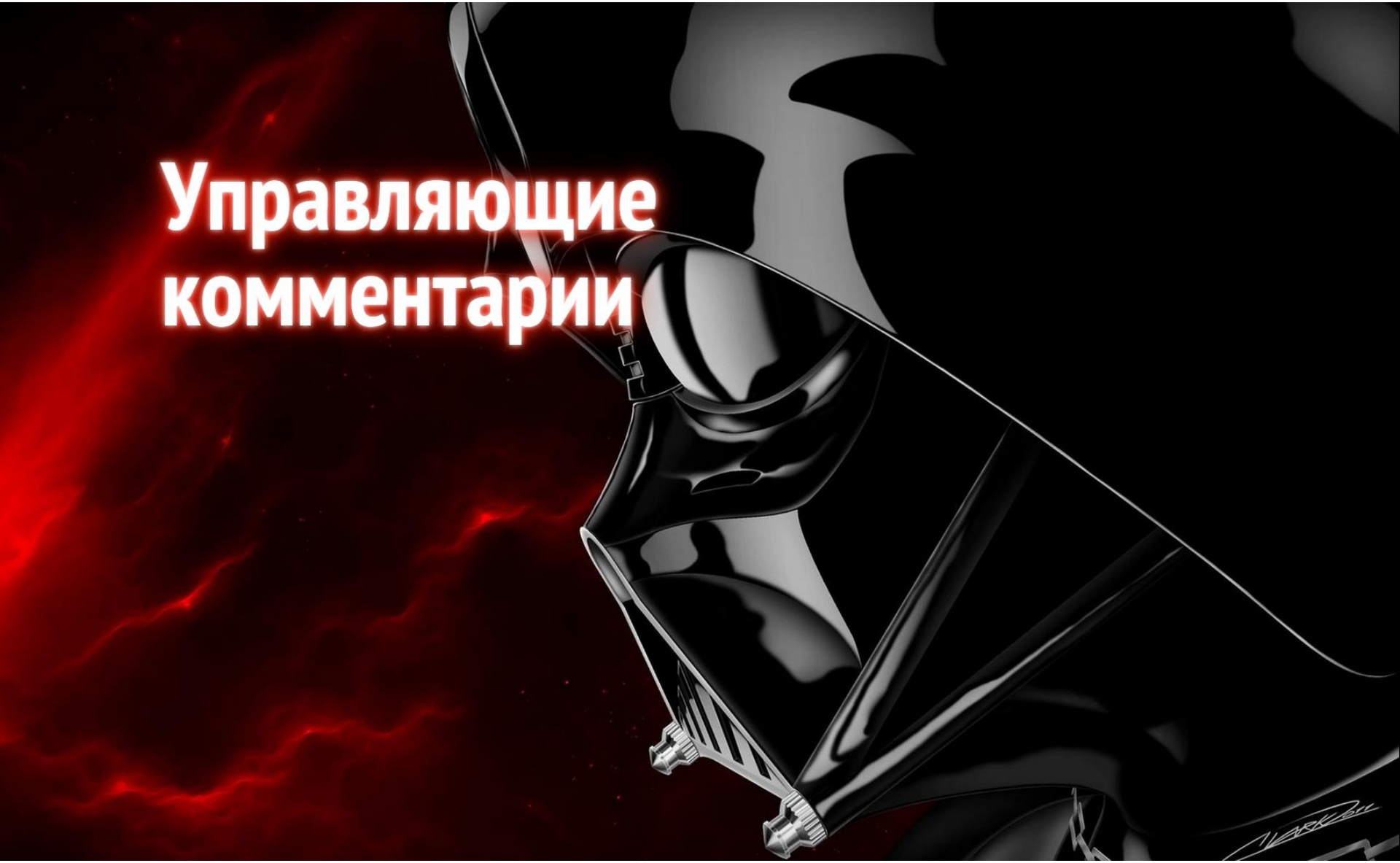
```hh
<?hh
$stringStore = null;

class Store<T> {
    public function __construct(private T $data) {}
    public function get(): T    { return $this->data; }
    public function set(T $x): void { $this->data = $x;}
}

function createStringStore(string $data): Store<string> {
    $store = new Store($data);
    // Some do with $store
    return $store;
}

$stringStore = createStringStore('Hello world');
```

```ts
// TypeScript
var stringStore:Store<string> = null;

class Store<T> {
    public constructor(private data:T) {}
    public get(): T    { return this.data; }
    public set(x:T): void { this.data = x;   }
}

function createStringStore(data:string): Store<string> {
    var store:Store<string> = new Store(data);
    // Some do with $store
    return store;
}

stringStore = createStringStore('Hello world');
```

# TypeScript поможет вам сделать мир лучше

# Что еще стоит знать про TypeScript?

# Управляющие комментарии



# Управляющие комментарии

```
/// <reference path="file-name.d.ts" />
```

# Управляющие комментарии

## /// <amd-module name="myName" />

```
/// <amd-module name="lib/md5" />
function md5() { ... }
export = md5;
```

```
define("lib/md5", ["require", "exports"], function (require, exports) {
    function md5() { ... }
    return md5;
});
```

# Управляющие комментарии

## /// <amd-dependency path="dir/file" />

```
/// <amd-dependency path="lib/global" />
```

```
define(["require", "exports", "lib/global"], function (require, exports) { ... });
```

# Управляющие комментарии

## ///<amd-dependency path="file" name="var"/>

```
/// <amd-dependency path="legacy/base/view" name="View" />
View.someDo();
```

```
define(["require", "exports", "legacy/base/view"],
    function (require, exports, View) {
        View.someDo();
    }
);
```

# Управляющие комментарии

## <amd-dependency path="..." name="..." />

```
/// <reference path="legacy/base/view.d.ts" />

/// <amd-dependency path="legacy/base/view" name="View" />
```

Эквивалентно записи:

```
/// <reference path="legacy/base/view.d.ts" />

import View = require("legacy/base/view");
```

# Управляющие комментарии

```
/// <reference no-default-lib="true" />
```

0_o

# Подключение существующего ~~legacy~~ кода

# Пример подключения *.js в *.ts

- newApp.ts
- lib/crypt.js

lib/crypt.js

```
define(function () {
    function md5() { /* ... */ }
    function uid() { /* ... */ }
    return {
        md5: md5,
        uid: uid
    }
});
```

# newApp.ts

```
class AppController {
    private _uid: string;
    constructor() { ... }
    setUid(): string {
        return this._uid = crypt.uid(); // crypt в legacy
    }
    init(): void { ... }
    run(): void { ... }
    static exec(ctrl: AppController): void { ... }
}


export = AppController;
```

# newApp.ts

```
/// <amd-dependency path="lib/crypt" />
class AppController {
    setUid(): string {}
}
```

```
define(["require", "exports", "lib/crypt"], function (require, exports) {
var AppController = (function () {
        function AppController() {}
        AppController.prototype.setUid = function () { this._uid = crypt.uid() };
        return AppController;
    })();
    return AppController;
});
```

# newApp.ts

```
/// <amd-dependency path="lib/crypt" />
declare var require:any;
var crypt = require("lib/crypt");
class AppController { ... }
```

```
define(["require", "exports", "lib/crypt"], function (require, exports) {
var crypt = require("lib/crypt");
var AppController = (function () {
        function AppController() {}
        AppController.prototype.setUid = function () { this._uid = crypt.uid() };
        return AppController;
    })();
    return AppController;
});
```

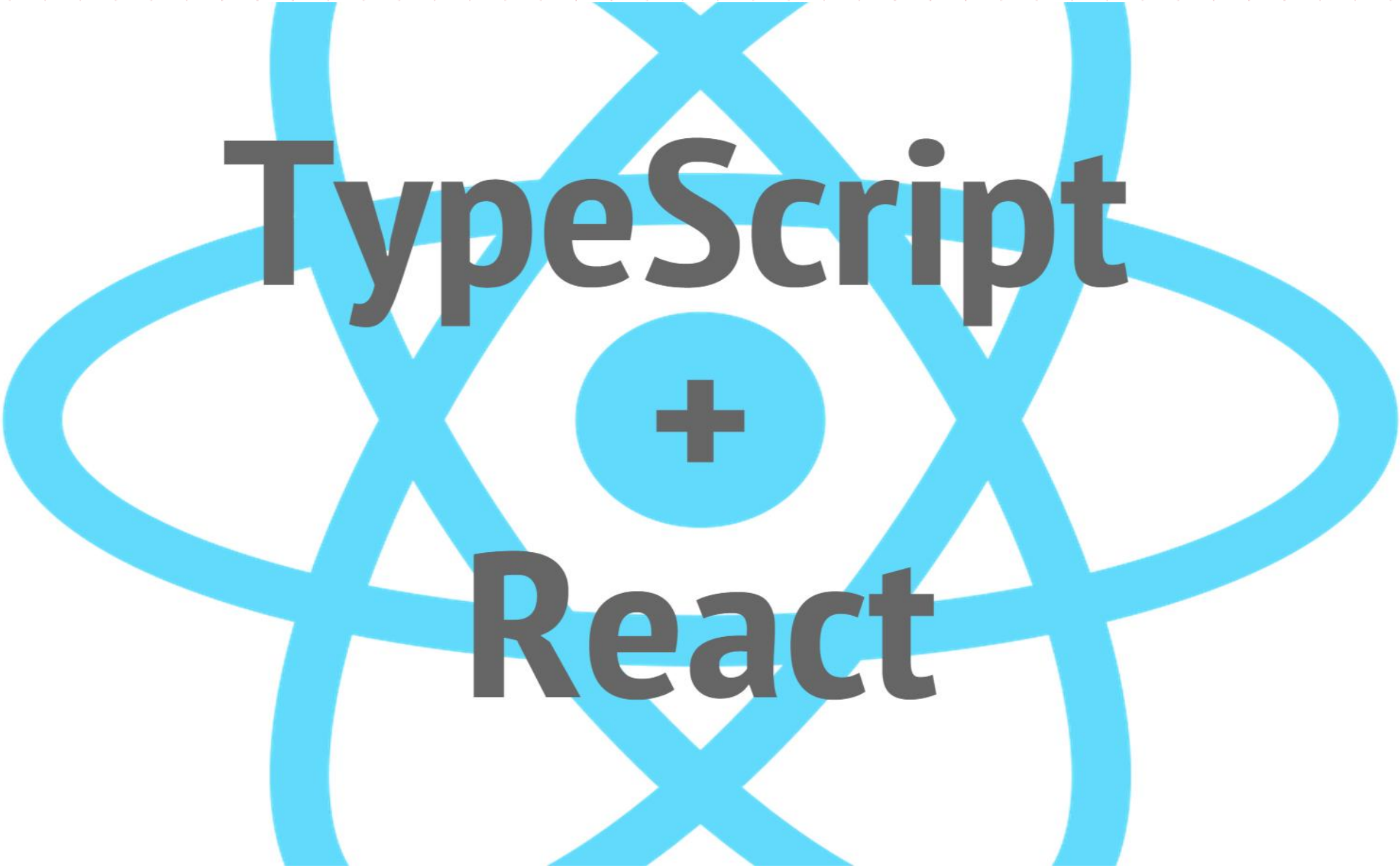
crypt.d.ts

```
declare module "lib/crypt" {

    export function md5() :string;

    export function uid() :string;

}
```

# newApp.ts

```
/// <require path="lib/crypt.d.ts" />
import crypt = require("lib/crypt");


class AppController { ... }
```

```
define(["require", "exports", "lib/crypt"], function (require, exports, crypt) {
    var AppController = (function () {
        function AppController() {}
        AppController.prototype.setUid = function () { this._uid = crypt.uid() };
        return AppController;
    })();
    return AppController;
});
```



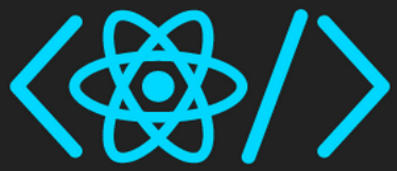
# TypeScript + React

# TypeScript ничего не знает про JSX

```
render: function() {
    return <div className="train">
        <div className="title">{this.state.title}</div>
        </div>
}
```

# TypeScript умеет жить с React

```
render: function() {
    return React.jsx(
        "<div className="train">
            <div className="title">{this.state.title}</div>
        </div>"
    )
}
```



# React Templates

# React Templates

- npm install react-templates -g

- npm install grunt-react-templates --save-dev
- WebStorm plugin: http://plugins.jetbrains.com/plugin/7648

Документация: wix.github.io/react-templates

# Пример работы TS с RT(JSX)

Структура компонента

.../page/train/seats/carMap/carMap/

- carMapItem
  - carMapItem.rt
  - carMapItem.rt.js
  - carMapItem.js / carMapItem.ts
- modelFactory.ts

## carMapItem.rt

```
<rt-require dependency="bemp/blocks/popup/component" as="Popup"/>
<rt-require dependency="bemp/blocks/train/seats/carMap/legend/component" as="Legend"/>
<div class="car_wrapper">
    <Ereg isAvailable={this.props...} schemeHeight={this...} className="...">
    </Ereg>{this.props.children}<div class="..." rt-if="...">
        <Popup
            innerHtml={this.props.errorMessage}
            active="{!this.state.popupHiddenByClick}"
            onClose={this.onPopupClosed}></Popup>
    </div>
    <Legend rt-if="this.props.showLegend" className="..."/>
</div>
```

## carMapItem.rt.js

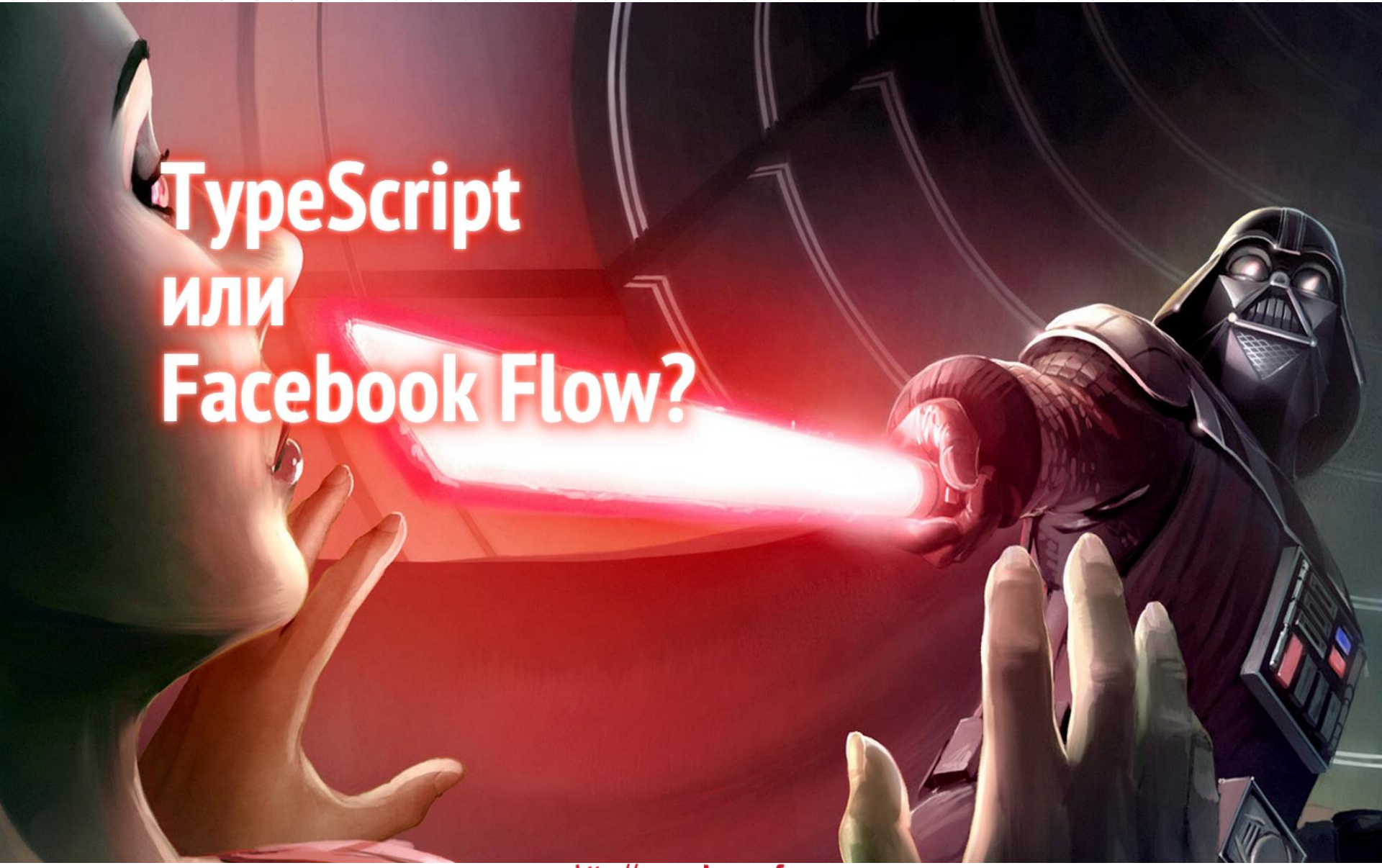
```
define([
    'react/addons', 'lodash', 'bemp/blocks/popup/component', ...
], function (React, _, Popup, ...) {
    'use strict';
    return function () {
        return React.createElement('div', {...}, React.createElement(Ereg, {
            'isAvailable': this.props.isEregAvailable, ...
        }),
        this.props.errorMessage
            ? React.createElement('div', {...}, React.createElement(Popup, {
                    'innerHtml': this.props.errorMessage,
                    'onClose': this.onPopupClosed
                }))
            : null,
            ...
    };
```

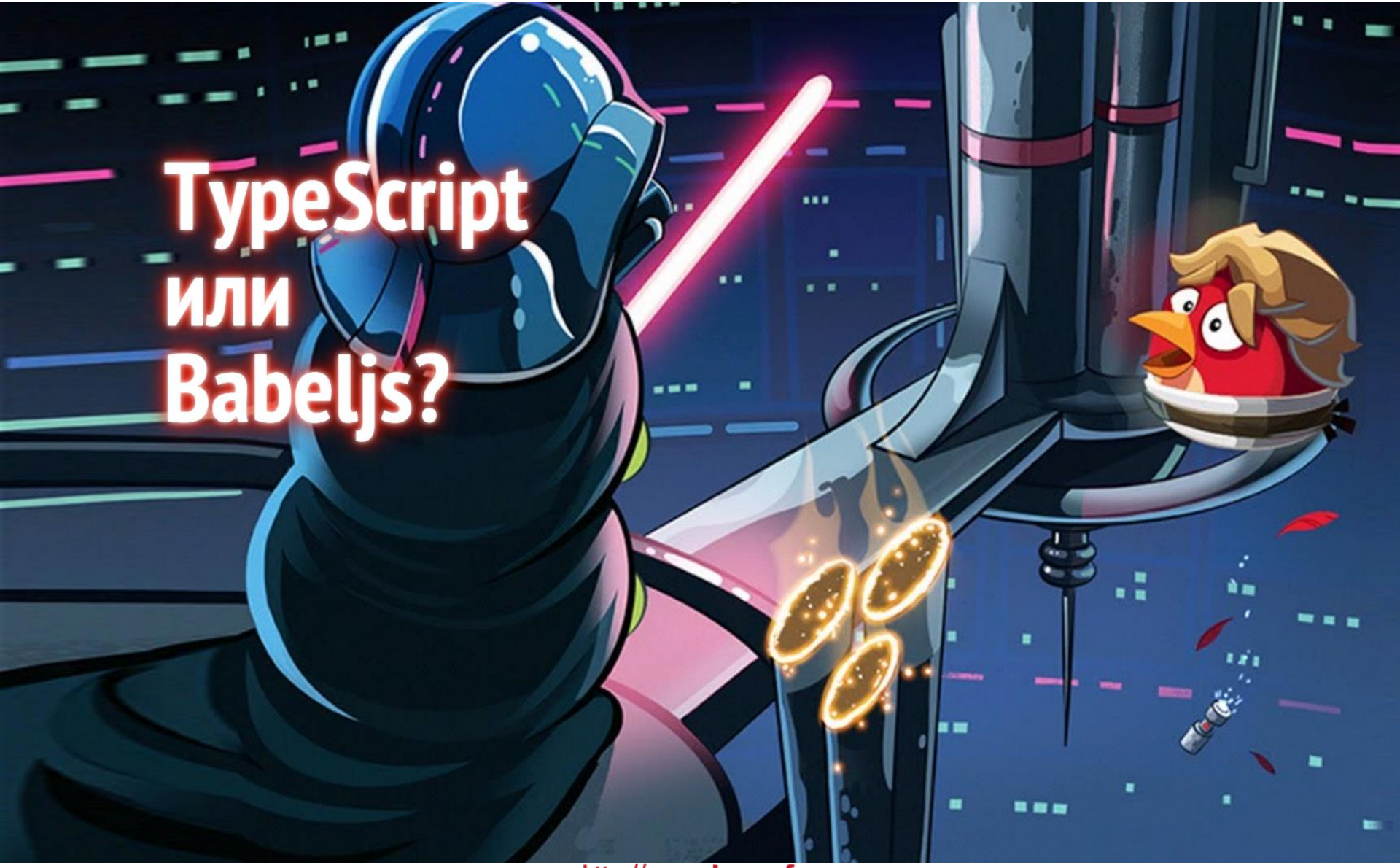
# carMapItem.js

```
define(['react', './carMapItem.rt', ...], function (React, template, ...) {
    return React.createClass({
        displayName: 'carMapItem',
        getDefaultProps: function () {
            return { ... }
        },
        checkErrorMessage: function (props) { ... },
        getInitialState: function () {
            return { ... }
        },
        onPopupClosed: function () {
            this.setState({ ... })
        },
        render: template
    });
});
```

# carMapItem.ts

```
import React = require("react");
import template = require("./carMapItem.rt");

class carMapItem extends React.Component {
        public displayName:string = 'carMapItem',
        static defaultProps = {
            ...
            onErrorShow: () => void 0
        }
        getInitialState() { return { ... } }
        onPopupClosed() { this.setState({ ... }) }
        render() { return template }
}

export = carMapItem;
```

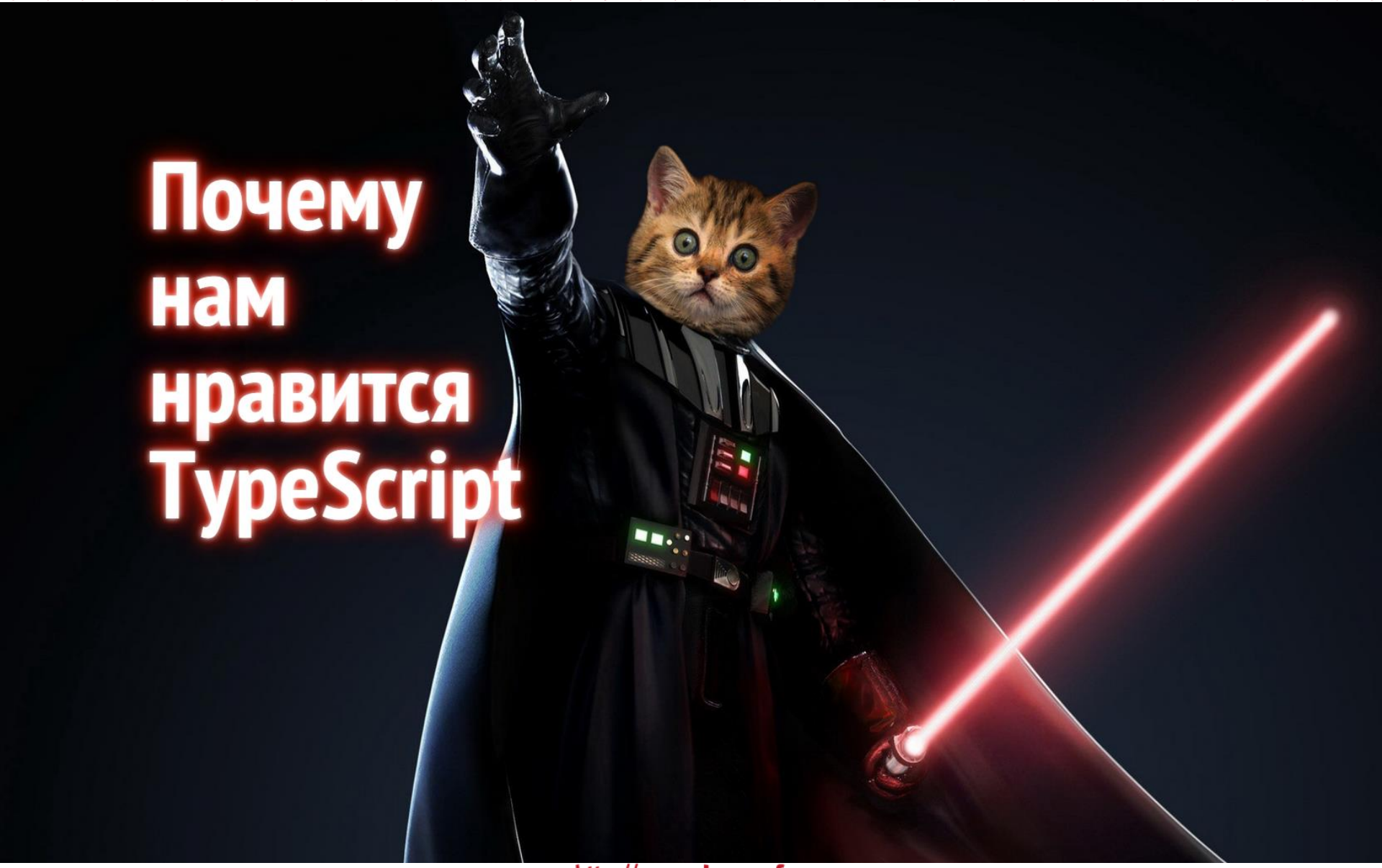

TypeScript
или
Facebook Flow?


TypeScript
или
Babeljs?


TypeScript
Microsoft
=
BABEL
+
flow

# Выбираем Flow если ...

- ... хочется JSX
- ... хочется Babel JS, но не хочется MS TS


Почему
нам
нравится
TypeScript

# Почему стоит писать на TypeScript

- Пишем на ES6+
- Типизация защищает
- Нормальные инструменты для реализации ООП
- **PHP разработчики понимают код JS разработчика**
- Активное развитие и поддержка от MS
- Хорошая поддержка в IDE
- Активное развитие инструментов:
  - Definition manager http://definitelytyped.org/tsd/
  - Doc generator http://typedoc.io/
  - Различные плагины для IDE, grunt, gulp, ...

# Вопросы?

# Александр Майоров

Tutu.ru

- mayorov@tutu.ru
- majorov.su

- Презентация: devconf2015.majorov.su