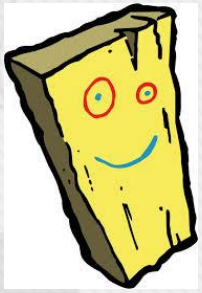
What to do with types in JavaScript

about:TypeScript

Who's talkin to you

- Vedran Maršić AKA Fosna
  - dev at



axilis

  - organizing programming workshops for kids
    - Logo, Scratch, HTML, CSS, JavaScript

JavaScript is the assembly language of the web!

Author *unknown*, mentined at:

The Birth & Death of JavaScript
A talk by Gary Bernhardt from PyCon 2014
bit.ly/RtLaCe

JavaScript alternatives

- CoffeeScript
- Dart
- GWT
- Script#
- TypeScript
- …

Folks rail against static typing but they don't complain about **JSLint**. **TypeScript** offers **optional** type annotations - it's hardly a *perversion* of JavaScript.

Why does TypeScript have to be the answer to anything?
by Scott Hanselman
http://bit.ly/U1dHQR

TypeScript overview

Super set of JavaScript

Optional static typing

Static type checking at compile time

- smarter rename, refactor
- go to definition, declaration
- call hierarchy

Code sugar for classical OOP

Produces idiomatic JavaScript

by Anders Hejlsburg

Original author of Turbo Pascal

Chief architect of Delphi

Lead architect of C#

Anounced in Oct 2012

v 1.0 in Apr 2014

Open Source @ typescript.codeplex.com/

TypeScript trans(com)piler writen in TypeScript

Editor support:

Visual Studio 2012+,

SublimeText, Vi, Emacs, your favorite browser

http://www.typescriptlang.org/Playground

TypeScript language features

classes, modules, function expressions

type annotations, type inference

interfaces, generics

enums

Nicked with author approval. Thank you **Vlad**!
https://speakerdeck.com/vladiliescu/typescript-vs-coffeescript


TYPESCRIPT


COFFEESCRIPT

Code sugar for classical OOP

CLASSES, INHERITANCE

JavaScript

```javascript
function Dinosaur(name, species) {
    this.name = name;
    this.species = species;
}

Dinosaur.prototype.sayHi = function () {
    console.log("Hi, I'm " + this.name + " and I'm a " + this.species + "!");
};
```

```javascript
function Triceratops(name) {
    Dinosaur.call(this, name, 'Triceratops');
}
Triceratops.prototype = new Dinosaur();
Triceratops.prototype.constructor = Triceratops;
Triceratops.prototype.sayHi = function () {
    Dinosaur.prototype.sayHi.call(this);
    console.log('I rule!');
};
```

```javascript
var cera = new Triceratops('Cera');
cera.sayHi();
```

TypeScript

```
class Dinosaur {
  constructor(public name: string, public species: string) {
  }

  public sayHi() {
    console.log("Hi, I'm " + this.name + " and I'm a " + this.species + "!");
  }
}
```

```
class Triceratops extends Dinosaur {
  constructor(name: string) {
    super(name, 'Triceratops');
  }

  sayHi() {
    super.sayHi();
    console.log('I rule!');
  }
}
```

```
var cera = new Triceratops('Cera');
cera.sayHi();
```

CoffeeScript

```
class Dinosaur
  constructor: (@name, @species) ->

  sayHi: ->
    console.log("Hi, I'm #{this.name} and I'm a #{this.species}!");
```

```
class Triceratops extends Dinosaur
  constructor: (name) ->
    super(name, 'Triceratops');

  sayHi: ->
    super();    # <--
    console.log('I rule!');
```

```
cera = new Triceratops('Cera');
cera.sayHi();
```

JavaScript

```
var __extends = 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 Dinosaur = (function () {
    function Dinosaur(name, species) {
        this.name = name;
        this.species = species;
    }
    Dinosaur.prototype.sayHi = function () {
        console.log('Hi, I\'m ' + this.name + ' and I\'m a ' +
this.species + '!');
    };
    return Dinosaur;
})();

var Triceratops = (function (_super) {
    __extends(Triceratops, _super);
    function Triceratops(name) {
        _super.call(this, name, 'Triceratops');
    }
    Triceratops.prototype.sayHi = function () {
        _super.prototype.sayHi.call(this);
        console.log('I rule!');
    };
    return Triceratops;
})(Dinosaur);

var cera = new Triceratops('Cera');
cera.sayHi();
```

```
var Dinosaur, Triceratops, cera, __hasProp = {}.hasOwnProperty,
    __extends = function(child, parent) { for (var key in parent) { if
(__hasProp.call(parent, key)) child[key] = parent[key]; } function ctor() {
this.constructor = child; } ctor.prototype = parent.prototype;
child.prototype = new ctor(); child.__super__ = parent.prototype; return
child; };

Dinosaur = (function() {
  function Dinosaur(name, species) {
    this.name = name;
    this.species = species;
  }
  Dinosaur.prototype.sayHi = function() {
    return console.log("Hi, I\'m " + this.name + " and I\'m a " +
this.species + "!");
  };
  return Dinosaur;
})();

Triceratops = (function(_super) {
  __extends(Triceratops, _super);

  function Triceratops(name) {
    Triceratops.__super__.constructor.call(this, name, 'Triceratops');
  }
  Triceratops.prototype.sayHi = function() {
    Triceratops.__super__.sayHi.call(this);
    return console.log('I rule!');
  };

  return Triceratops;
})(Dinosaur);

cera = new Triceratops('Cera');
cera.sayHi();
```

Pesky global scope management, actually.

MODULES

TypeScript

```
module Dinosaurs {

    var petrie = { name: 'Petrie', species: 'Pteranodon' };
    export var spike = { name: 'Spike', species: 'Stegosaurus' };

}
```

```
var Dinosaurs;
(function (Dinosaurs) {

    var petrie = { name: 'Petrie', species: 'Pteranodon' };

    Dinosaurs.spike = { name: 'Spike', species: 'Stegosaurus' };

})(Dinosaurs || (Dinosaurs = {}));
```

JavaScript

CoffeeScript

```
petrie = { name: 'Petrie', species: 'Pteranodon' }
window.spike = { name: 'Spike', species: 'Stegosaurus' };
```

```
(function() {
  var petrie;

  petrie = {name: 'Petrie', species: 'Pteranodon'};

  window.spike = {name: 'Spike', species: 'Stegosaurus'};

}).call(this);
```

JavaScript

Going external

SimpleModule.ts

```
import m = require('mod');
export var t = m.something + 1;
```

AMD / RequireJS SimpleModule.js:

```
define(["require", "exports", 'mod'], function(require, exports, m) {
    exports.t = m.something + 1;
});
```

CommonJS / Node SimpleModule.js:

```
var m = require('mod');
exports.t = m.something + 1;
```

Fix this. Like lambda expressions.

FUNCTION EXPRESSIONS

```javascript
var mouseMoveTracker = {
    count: 0,
    startTracking: function () {
        window.onmousemove = function (e) {
            this.count++;
            console.log(this.count);
        };
    }
};

mouseMoveTracker.startTracking();
```

JavaScript

CoffeeScript

```
mouseMoveTracker =
  count: 0,
  startTracking: ->
    window.onmousemove = (e) =>
      this.count++;
      console.log(this.count);

mouseMoveTracker.startTracking();
```

TypeScript

```
var mouseMoveTracker = {
  count: 0,
  startTracking: function () {
    window.onmousemove = (e) => {
      this.count++;
      console.log(this.count);
    };
  }
};

mouseMoveTracker.startTracking();
```

```javascript
var mouseMoveTracker = {
    count: 0,
    startTracking: function () {
        var _this = this;
        window.onmousemove = function (e) {
            _this.count++;
            console.log(_this.count);
        };
    }
};

mouseMoveTracker.startTracking();
```

JavaScript

With type annotations.

INTERFACES

TypeScript

```
function greetDino(dino: { name: string; sayHi(): void; }) : void
{
  console.log("Look, it's " + dino.name + "!");
  dino.sayHi();
}
```

```
greetDino({
  name: 'Spike',
  sayHi: function () { }
});

greetDino(new Triceratops('Cera'));
```

TypeScript

```
interface IDinosaur {
  name: string;
  sayHi();
}
```

```
function greetDino(dino: IDinosaur) : void
{
  console.log("Look, it's " + dino.name + "!");
  dino.sayHi();
}
```

```
greetDino({
  name: 'Spike',
  sayHi: function () { }
});

//Triceratops doesn't actually implement IDinosaur but respects its contract
greetDino(new Triceratops('Cera'));
```

http://bit.ly/1oKVaox

What impressed me the most is what TypeScript **doesn't** do. It **doesn't output** type checking into your JavaScript code.

Thoughts on TypeScript
by Nicholas C. Zakas
http://bit.ly/SkGe2y

Type definition files

Like header files in C, C++

```
interface JQuery {
    text(content: string);
}
interface JQueryStatic {
    get(url: string, callback: (data: string) => any);
    (query: string): JQuery;
}
declare var $:JQueryStatic
```

* JQuery declaration for demo. Real declarations are verbose.

Definitely Typed
https://github.com/borisyankov/DefinitelyTyped

Of course, all valid Javascript is valid Typescript so you can just change the file extension and be compiling it but if you want **all the benefits** you will want to take advantage of some **type annotations** and that's the **manual part**.

Typescript - a real world story of adoption in TFS
by Brian Harry
http://bit.ly/SkGe2y

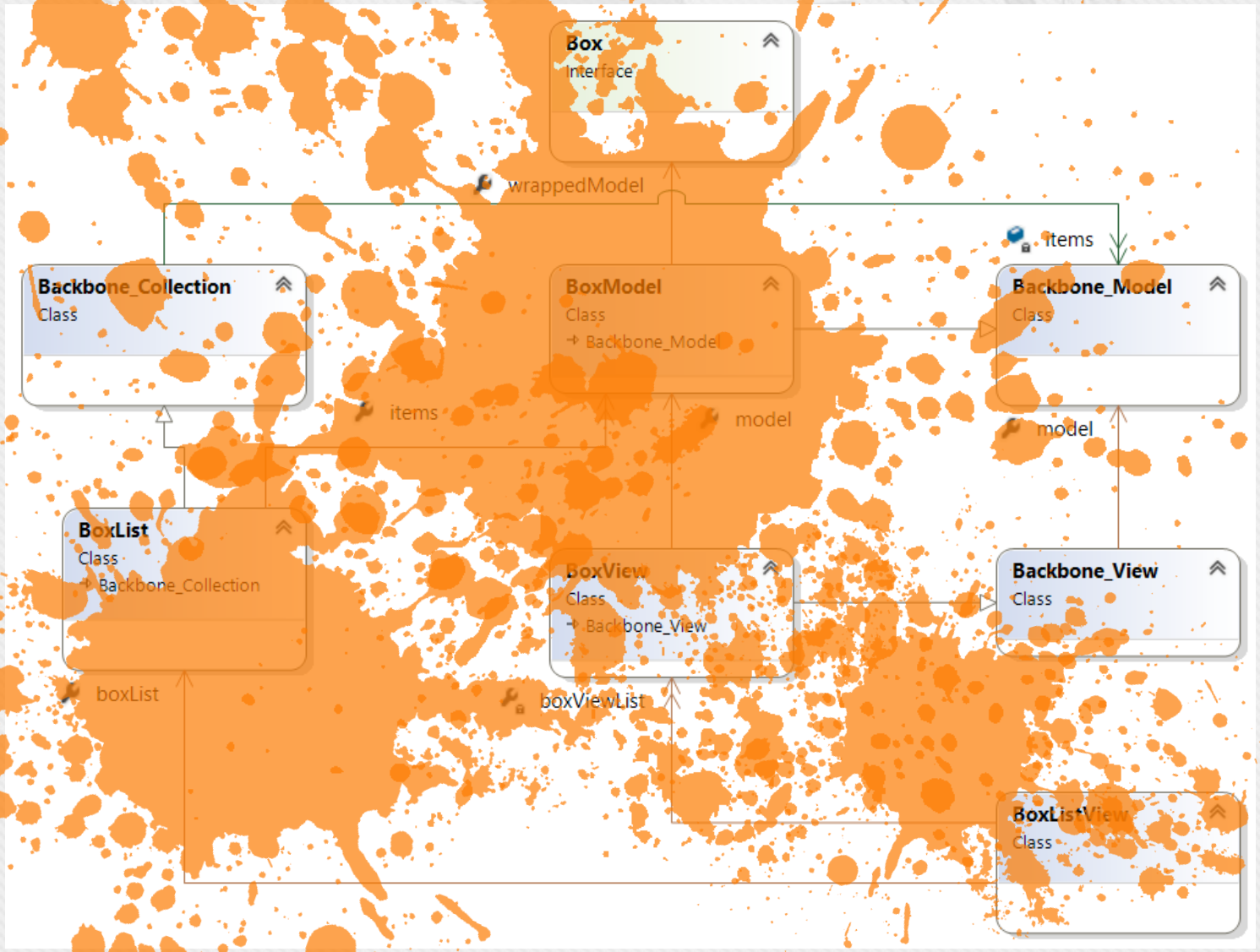
CoffeScript has more code sugar to be fair.
Check out if interested:
CoffeScript the Good Parts
by Bruno Dević
http://slides.com/brunodevic/coffeescript-the-good-parts

COFFESCRIPT CLOSURE

DEMO

https://github.com/Fosna/TypeBoxes

Object model



Interesting quotes

FINAL THOUGHTS

Microsoft's TypeScript may be the best of the many JavaScript front ends. It seems to generate the **most attractive code**.

post on Google+

by Douglas Crockford

bit.ly/SnJzw5

And I think it **should take pressure off** of the ECMAScript Standard for new features like **type declarations** and classes. Anders has shown that these can be provided nicely by a **preprocessor**, so there is no need to change the underlying language.

same post on Google+
by Douglas Crockford
bit.ly/SnJzw5

I think that JavaScript's loose typing is one of its best features and that type checking is way overrated. **TypeScript adds sweetness, but at a price**. It is not a price I am willing to pay.

same post on Google+

by Douglas Crockford

bit.ly/SnJzw5

TypeScript feature that suprised me the most, the quality of the JavaScript generated by the TypeScript transcompiler and the transcompiler itself **as a tool to learn** JavaScript best practices.

Learn TypeScript to Improve Your JavaScript

by Jaime González García

http://bit.ly/1ojyhVJ

References

Spread throughout the presentation.

PHOTO CREDITS

http://www.flickr.com/photos/jimmonk/8392597425/

http://www.flickr.com/photos/jimmonk/8392597145/