DEMYSTIFYING TYPESCRIPT

5 COMMON MYTHS DEBUNKED

Created by Kamran Ayub / @kamranayub

ABOUT ME

I work at General Mills as an application developer, I maintain some open source projects, and I like to play video games

github.com/kamranayub

FACT

WRITING LARGE JAVASCRIPT APPLICATIONS IS HARD

THE MYSTICAL TYPESCRIPT

DEMYSTIFIED

MYTH #1

“TypeScript is not interoperable with JavaScript”

AKA

“I have to rewrite my JavaScript codebase”

WHAT LANGUAGE IS THIS?

```
var Point = function (x, y) {
        this.x = x;
        this.y = y;
};
Point.prototype.distance = function (other) {
        other = other || new Point(0.0, 0.0);
        return Math.sqrt(
        Math.pow(this.x - other.x, 2) +
        Math.pow(this.y - other.y, 2));
};

var p1 = new Point(5, 6);
var p2 = new Point(1, 1);

// Calculate distance between two points
console.log("Distance between", p1, p2, "is", p1.distance(p2));
```

It's JavaScript!

And it's TypeScript!

ALL VALID JAVASCRIPT IS VALID TYPESCRIPT

FACTS

- TypeScript *compiles* into plain old JavaScript
- Static typing is optional
- There is no overhead because TypeScript is not a runtime

DECLARATION FILES

Declaration files make it easy to work with existing libraries

github.com/borisyankov/DefinitelyTyped

QUICK EXAMPLE

MYTH #1

“TypeScript is not interoperable with JavaScript”

TypeScript is a *typed superset* of JavaScript that compiles down to plain JavaScript

MYTH #2

“It's a whole new language”

AKA

“I have to re-learn JavaScript”

YES AND NO

Are there new language features to learn? **Yes**

Is it a whole new language? **No**

PERSPECTIVE

- TypeScript brings ECMAScript 6 to you now
- You will need to learn the new syntax *anyway*
- There are only a few new things to learn
- TypeScript can target different versions of ECMAScript

STATIC TYPING

```
function distance(x1, y1, x2, y2) {
        return Math.sqrt(
        Math.pow(x1 - x2, 2) +
        Math.pow(y1 - y2, 2));
}

alert(distance(0, 0, 5, 5));   // OK
alert(distance("a", 0, 5, 5)); // NaN
```

CLASSES & MODULES

```
var Point = function (x, y) {
        this.x = x;
        this.y = y;
};
Point.prototype.distance = function (other) {
        other = other || new Point(0.0, 0.0);
        return Math.sqrt(
        Math.pow(this.x - other.x, 2) +
        Math.pow(this.y - other.y, 2));
};
```

AND THERE'S MORE!

Classes, modules and static typing are only a few examples

Rest (...args), default parameters, overloading, fat arrow, interfaces, AMD/CommonJS modules, etc.

typescriptlang.org/Playground

MYTH #2

“It's a whole new language”

Much of the syntax and features are borrowed straight from ECMAScript 6

MYTH #3

“You can only use TypeScript in Visual Studio”

AKA

“I'm on Linux!”

FACTS

- The TypeScript compiler and language service is open-source
- You can use TypeScript on any platform and any OS
- You can integrate TypeScript into any editor of your choice

FIRST-CLASS EXPERIENCE

Visual Studio provides the best tooling support, no question

Express edition is free on Windows

WHAT ABOUT THE REST OF US?

- Sublime has a TypeScript plugin
- You can create a compile step in Grunt or Gulp
- You can `` `npm install -g typescript` `` on any platform
- WebStorm is a great alternative IDE for every platform: jetbrains.com/webstorm/

LET'S DO IT NOW

We're going to setup TypeScript in Sublime Text

... and if we have time, Visual Studio "Monaco"

MYTH #3

“You can only use TypeScript in Visual Studio”

TypeScript is open-source and can integrate with just about any editor you use

MYTH #4

“Unit testing will eliminate any need of TypeScript”

AKA

“I don't need no stinkin' type checking”

CONSIDER THE FOLLOWING

“Well, everything looks okay, let's run this”

```
TypeError: undefined is not a function
```

*“**** ***ing ****!!!!”*

SOUND FAMILIAR?

HOW ABOUT THIS?

“We need to change one of the parts of this app”

“Ready the blood sacrifices so the JavaScript Refactor Gods will have mercy on our souls”

A READ-ONLY CODEBASE IS NOT GOOD

THE SAD TRUTH

Not everyone writes JavaScript unit tests

AND EVEN IF YOU HAVE TESTS

TypeScript eliminates an entire class of errors

A PRACTICAL EXAMPLE

bit.ly/buildts2

MYTH #4

“Unit testing will eliminate any need of TypeScript”

TypeScript can catch easily overlooked errors and reduces risk of runtime errors

MYTH #5

“TypeScript is like CoffeeScript or Dart”

AKA

“Don't we already have this?”

NOPE*

COFFEESCRIPT

```
class Point
  constructor(@x, @y) ->

  distance(other) ->
    dx = @x - other.x
    dy = @y - other.y
    Math.sqrt dx * dx + dy * dy
```

- Inspired by Ruby & Python
- Is not statically typed
- Is more of a "macro" or expansion language

DART

```
import 'dart:math';

class Point {
  num x;
  num y;

  Point(this.x, this.y);

  num distance(Point p) {
    num dx = x - p.x;
    num dy = y - p.y;
    return sqrt(dx * dx + dy * dy);
  }
}
```

- Supports static types
- Compiles into JavaScript
- Full on SDK/framework and language

TYPESCRIPT

```
class Point {
  constructor(public x: number, public y: number) {

  }

  distance(p: Point) {
    var dx = this.x - p.x;
    var dy = this.y - p.y;

    return Math.sqrt(dx * dx + dy * dy);
  }
}
```

- Good middle-ground, outputs idiomatic JavaScript
- Looks more like (and will target) ECMAScript 6
- Simpler than Dart, "it's just JavaScript"

MYTH #5

“TypeScript is like CoffeeScript or Dart”

There are pretty big differences between these languages

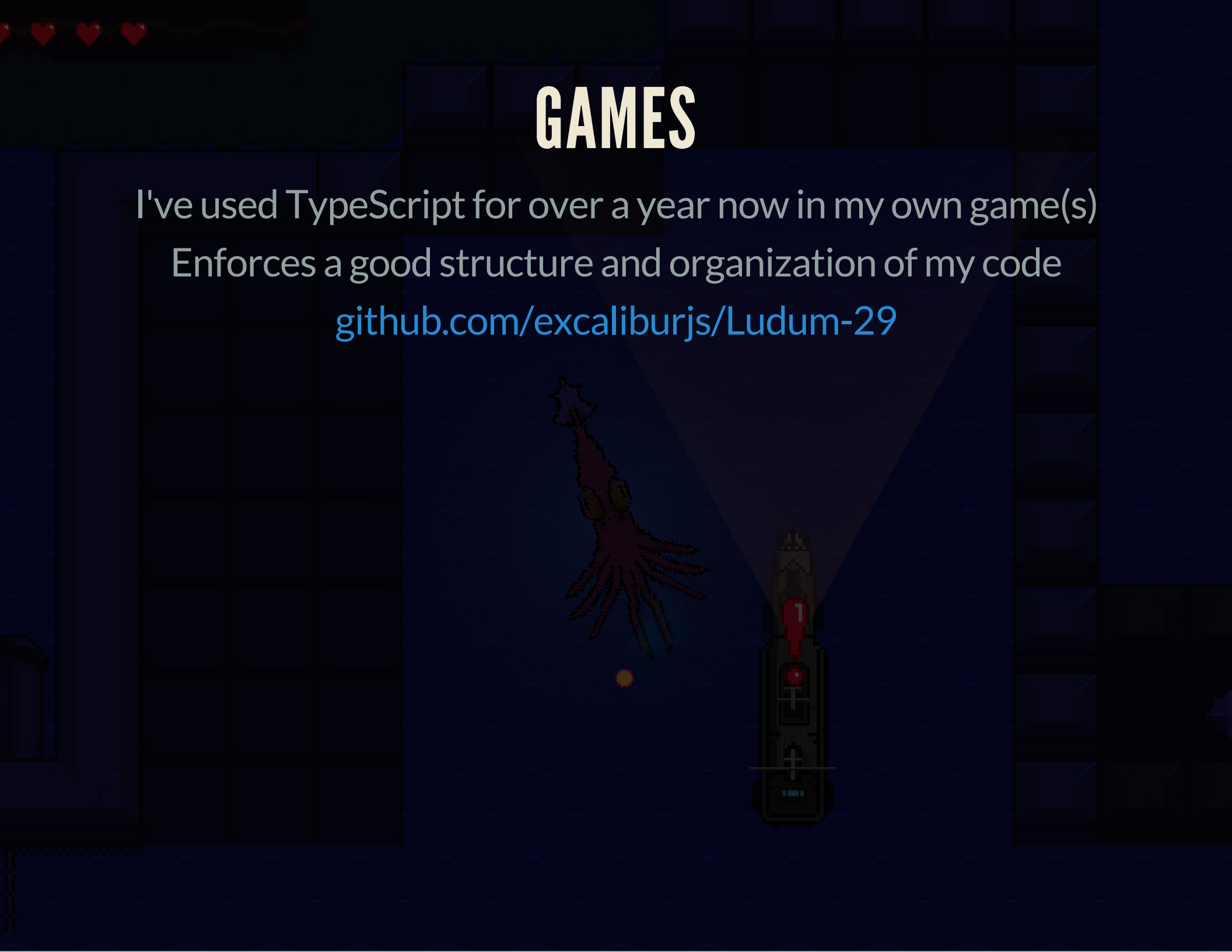
PRACTICAL USE CASES

GAMES

I've used TypeScript for over a year now in my own game(s)

Enforces a good structure and organization of my code

github.com/excaliburjs/Ludum-29



LIBRARIES

Provides type-safe API and type-rich documentation

Provides structure and organization to your framework

Great for OSS projects where others can contribute

github.com/excaliburjs/Excalibur

LARGE APPLICATIONS

Visual Studio "Monaco" was written in TypeScript

bit.ly/buildts

THANK YOU

@kamranayub - kamranicus.com