TypeScript

the better javascript?

Andreas Jägle

GDG DevFest 2015 Karlsruhe

Me

- Andreas Jägle
- Software Developer @cluetec

- Java / Spring / NoSQL / Web / JavaScript
- Passion for tooling & build processes

- mail@ajaegle.de
- @ajaegle

You?

- Software Developers?
- Java / C# / etc.?
- JavaScript?
- That thing on the right?

```
var Greeter = (function () {
  function Greeter(greeting) {
    if (!greeting) { greeting = "Hello, "; }
    this.greeting = greeting;
  }
  Greeter.prototype.greet = function (name) {
    console.log(this.greeting + name);
  };
  return Greeter;
})();

new Greeter().greet("Karlsruhe");
new Greeter("Hallo, ").greet("Karlsruhe");
```

Agenda

- Motivation
- TypeScript Overview
- TypeScript Features
- Tooling
- Demo
- Conclusion

Motivation

Motivation

```
Object.defineProperty(window, "myProp", {"value": "booyah", readOnly: true});

function distance() {
  var distance = parseInt(document.getElementById("distance"));
  if (distance > 100) {
    document.getElementById("longDistanceWarn").setAttribute("hidden", false);
  }
  return distance;
}
function updateCosts() {
  costs = (distance * 0.01) + 10;
  document.querySelectorAll(".costs").textContent = costs;
}

document.getElementById("calcButton").addEventListener(updateCosts, "click");
```

Motivation

```
Object.defineProperty(window, "myProp", {"value": "booyah", readOnly: true});

function distance() {
  var distance = parseInt(document.getElementById("distance"));
  if (distance > 100) {
    document.getElementById("longDistanceWarn").setAttribute("hidden", false);
  }
  return distance;
}
function updateCosts() {
  costs = (distance * 0.01) + 10;
  document.querySelectorAll(".costs").textContent = costs;
}

document.getElementById("calcButton").addEventListener(updateCosts, "click");
```

Motivation

```
Object.defineProperty(window, "myProp"  {"value": "booyah", readOnly: true});

function distance() {
  var distance = p                    entById("distance"));
  if (dista
    documen                           stanceWa          Attribute("hidden", false);
  }
  return dist
}
function updateCo
  costs = (dista
  document.query                      s").textContent = costs;
}

document.getElementById("calcButton").addEventListener(updateCosts, "click");
```

TypeScript to the rescue!

TypeScript Overview

What is TypeScript?

- A superset of JavaScript
- ES6 features today
- Type system

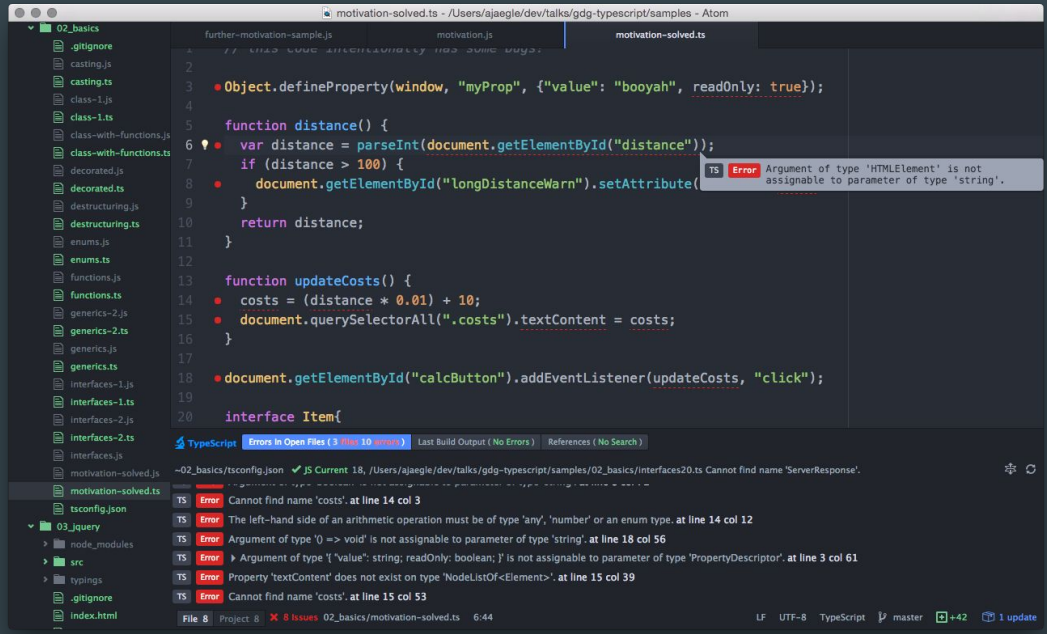
- Transpiler - No Runtime
- Compiles down to ES3 / ES5 / ES6

- Open sourced by Microsoft in 2012
- Version 1.0 in 2014, Current 1.6

Why TypeScript?

- Type Definitions for JavaScript API

- Detecting Bugs
- IDE Auto-Completion
- Code Navigation
- Rename / Refactor



TypeScript Features: Basics

Basic Types

- Basic JavaScript types
- Arrays of these

- Compile-time only

- Explicit type definitions
- Implicit by type inference

```
var age: number;
var name: string;
var vip: boolean;
var emails: string[];
var additionals: Object;

var age = 52;
var name = "John Doe";
var vip = false;
var emails = ["john@doe.com"];
var additionals = {};

var unsure: any = 1337;
unsure = "i'm a string";
```

Type Aliases and composition

- Aliasing for existing types

- Union Types
- “is either A or B”
- Requires typeof checks

- Intersection Types
- “is as well A as B”

```
type mystr = string;
var surname: mystr;

type stringOrStrings =
          string | string[];

var akas: stringOrStrings;
akas = "single";
akas = ["first", "second"];
```

Enum Types

- ES6-Feature
- Mapped to Integers

- Can be compile-time only
- Or: preserveConstEnums

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

let myColor: Color = Color.Blue;
console.log(myColor);         // 2
console.log(Color[myColor]); // Blue


enum Bet { Home = 1, Away = 2, Draw = 0}
let myBet = Bet.Home;

if (myBet === Bet.Home) {
 console.log("win!");
}
```

Any Type

- Supertype of all types
- Opt-out of type checks

- Can get assigned everything
- Can be assigned to everything

- Example: HTTP library

```
let known: string = "really a string";
let probably: any = known;

let unknown: any;

let itsAString: string = unknown;
let itsANumber1: number = unknown;
```

Interfaces

- Compile-time only

- Description of objects' structure
- Required and optional properties

- Strict Object Literal Checking

```
interface MyConf {
  host: string;
  port: number;
  path?: string;
}

let conf: MyConf = {
  host: "localhost",
  port: 3000,

  somethingElse: true
};
```

TypeScript Features: Functions

Functions

- Typed parameters
- Return type
- Optional values
- Default values

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

function logIt(msg: Message,
               verbose: boolean) { ... }

function logIt(msg: Message,
               verbose = false) { ... }
```

Function Types

- Type definition for functions

- As type
- As interface

- both compatible due to duck typing

```
type MyListenerType =
  (event: MyEvent) => void;

let listeners: MyListenerType[] = [];

function addEventListener(listener:
  MyListenerType): void {
  listeners.push(listener);
}

interface MyListenerIf {
  (event: MyEvent): void;
}
```

Arrow Functions

- ES6-Feature
- Shorthand function syntax
- “Lambda Expressions”

- No own “this”
- No more “var that = this;”
- Uses regular scope lookup

```
let squareIt = (n: number) => {
  return n * n;
};

let sq = (n: number) => n * n;

let logit = (s: string) => console.log(s);
let isEven = (n: number) => n % 2 === 0;
```

TypeScript Features: Structuring

Classes

- ES6-Feature
- Data encapsulation

- Constructor
- Default parameters
- Access Modifiers for members and methods

- Inheritance & abstract classes

```
class MostSimpleClass { }

class Greeter {
  constructor(private greeting =
                        "Hello, ") { }

  greet(name: string) {
    console.log(this.greeting + name);
  }
}

new Greeter().greet("Karlsruhe");
```

Namespaces

- aka. “Internal modules”
- Syntactic sugar for IIFEs

- Encapsulation
- Explicit exports

- Order matters!
- Not recommended for larger apps

```
namespace ns1 {
  export var half = 21;
  export function double(a) {
    return a * 2;
  }
}

namespace ns2 {
  function calculateSolution() {
    let x = ns1.half;
    let solution = ns1.double(x);
    return solution;
  }
  console.log(calculateSolution());
}
```

(External) Modules

- Several module systems supported
- ES6-Style import syntax

- Play nice with node.js
- Usable in frontend development with module loaders (e.g. system.js)

- A must for larger apps

// myutil.ts

```
import * as fs from "fs";
export var printFileContent =
   (fileName: string) => { ... };
```

// index.ts

```
import * as myutil from "./myutil";
...
myutil.printFileContent(fileName);
```

TypeScript Features: ES2016

Decorators

- Decorator Pattern
- Aspect-oriented programming

- Annotations
- Attaching metadata

- Heavily used for AngularJS 2

```
class DecoratedClass {
  constructor(public a: number) {}
  @log
  foo(greeting: string) {
    return greeting + this.a;
  }
}

var log: MethodDecorator =
  function(target: Function,
           key: string,
           descriptor: any) {
  ...
  return descriptor;
};
```

Many more...

- Generics
- Intersection Types
- Template Strings
- Destructuring
- TypeGuards
- JSX Support

- Generators
- Async / await

Tooling

TypeScript Compiler

- Compiles TypeScript code to JavaScript
- npm install -g typescript

- tsc --module commonjs --target es6 main.ts
- Project Configuration: tsconfig.json

- tsc --watch

TypeScript Definition Files *.d.ts

- Compiling uses definition files
- “The type information that gets erased during compilation”

- Built-in runtime definition
  lib.d.ts

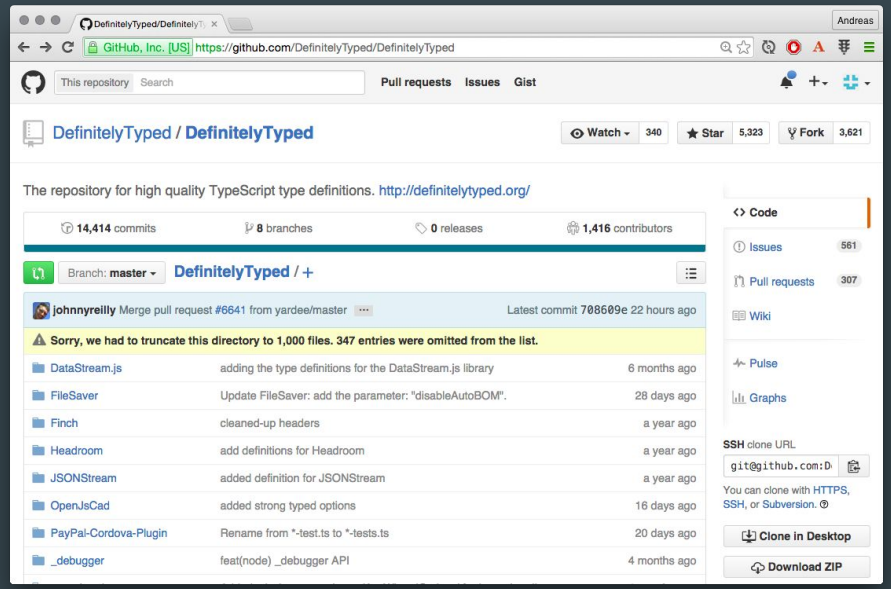
- Third-party definition files
- Provided together with library
- External definition sources

TypeScript Definition Manager

- Manages type definitions
- “npm for type definition files”
- npm install -g tsd

- tsd install jquery --save

- DefinitelyTyped Github Repo



Further tooling

- TSLint
- Static code analysis
- Enforcing code conventions

- Build tool integration
- gulp-typescript
- grunt-ts
- webpack ts-loader

- IDE Support

- Webstorm
- Atom
- SublimeText
- Visual Studio
- VSCode
- Eclipse ...

Demo Time

Conclusion & Outlook

Conclusion

Pros

- Productivity boost
- ES 201X features today
- Getting started is pretty easy
- Great Tooling and Community

- Fun to use technology!

“Cons”

- Definition file conflicts
- Prefer modules over namespaces

- Understanding JavaScript fundamentals is essential!

Outlook

- TypeScript 1.7 about to be released
- More and more libs with typings in npm

- Will gain traction with Angular 2

- Give it a try!

Q & A

Feedback?

mail@ajaegle.de
@ajaegle

Links

- Github Repository: following...

- TypeScript Handbook: http://www.typescriptlang.org/Handbook
- TypeScript Playground http://www.typescriptlang.org/Playground
- TypeScript Deep Dive by Basarat Ali Syed https://basarat.gitbooks.io/typescript/