

# From JavaScript to TypeScript

## Experiences from a Java project

**Sönke Sothmann**  
sso@tonbeller.com

# Agenda

- ▶ Introduction to TypeScript
- ▶ TypeScript Features
- ▶ Experiences
- ▶ Outlook

# Introduction to TypeScript

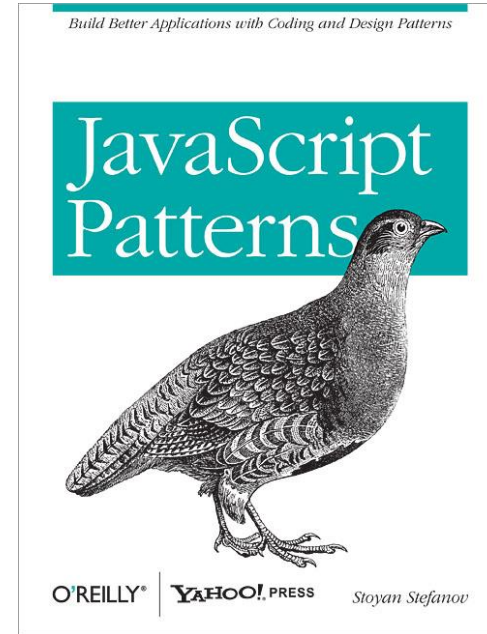
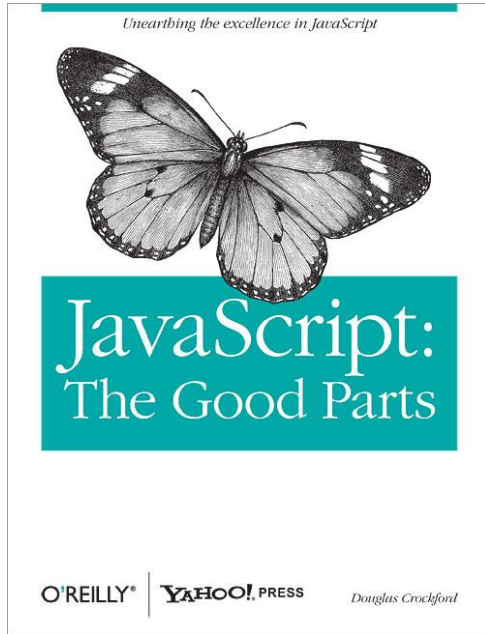
# A tale of woe

- ▶ GWT SPA → JS SPA
- ▶ Marginal IDE support
  - ▶ Autocompletion, refactorings, compiler checks
- ▶ Code is hard to navigate

# JavaScript as ByteCode

```
on hXg(){hXg=CEh;Swg=new dXg} function AXg(){AXg=CEh;jXg=new pXg} function BXg(){BXg=CEh;kXg=new xXg} fl
Eh;yah=new Cah} function Ebh(){Ebh=CEh;bbh=new fbh} function Hbh(){Hbh=CEh;Gbh=new xDb} function ich(){i
on Fkh(){FKh=CEh;qkh=new ukh} function Wkh(){Wkh=CEh;Hkh=new Lkh} function llh(){llh=CEh;Ykh=new alh} fl
gKe.call(this)} function Swg(){_M();mJf.call(this)} function Vwg(){_M();mJf.call(this)} function $wg(){_
ion FGb(){GGb.call(this,false)} function Q7(a){this.i=a;this.g=Nt()} function Ybb(a,b){this.b=a;this.c=t
.b=a;this.c=b} function uGe(a,b){this.b=a;this.c=b} function ccf(a,b){this.b=a;this.c=b} function mcf(a
ion yje(a,b){this.c=a;this.b=b} function ihf(a,b){this.c=a;this.b=b} function mhf(a,b){this.b=a;this.c=t
.b=a;this.c=b} function aLf(a,b){this.b=a;this.c=b} function wLf(a,b){this.b=a;this.c=b} function oMf(a
ion ofg(a,b){Nb.call(this,a,b)} function UPe(){WPe.call(this,false)} function DGe(){this.p=new lee(this)
.b=a;this.c=b} function LAg(a,b){this.b=a;this.c=b} function IIG(a,b){this.b=a;this.c=b} function mLG(a
ion p8g(a,b){this.b=a;this.c=b} function R8g(a,b){this.b=a;this.c=b} function Ieh(a,b){this.b=a;this.c=t
.b=a;this.c=b} function qnh(a,b){this.c=a;this.b=b} function bfh(a,b){Nb.call(this,a,b)} function Mlh(a
tion N8(a,b){return uWd(a,b,b,0)} function bN(a,b,c){return cN(a,b,c)} function $4f(a,b){return a.d.Td(t
(a,p,b);Uq(b,a)} function hg(a,b){Lqh(b,b,a,f)&&jg(a)} function mN(a,b){fi(b.Ec,a.g.n,null)} function Tl
l(this,'SUB',1)} function kJ(){bJ.call(this);this.i=10} function IR(a){wt.call(this);this.b=a} function
a.i.g}{return)m8(a)} function M8g(a){XSf(new pch);Iqf(a.b)} function hfh(a,b){pWd(a.k,b);ifh(a,b)} funct
p.Wd(new F1(c),b)} function yK(a,b,c){a.Xb=true;BM(b,c)} function Km(a,b){a.c=!b?(ed(),dd):b} function >
rn new xx(a,c,b)} function aN(a,b){return bN(a,b,a.Tb.c)} function M7(a){return evb(a)?evb(a):a} functic
this,a,1,20)} function xcf(a){A9e.call(this,a,2,28)} function SIF(a){TIF.call(this,a,false)} function Sz
d(a,DSd(a)))} function Bih(a,b){IQd(b,xSd(a,DSd(a)))} function Kih(a,b){Mih(b,xSd(a,DSd(a)))} function i
$(a.g,a.f);$$ (a.e,a.d)} function $Jf(a,b){uSd(a,b,b);uSd(a,b,c)} function Vrf(a,b){m$F(a.i,b,new lSf(a)
}De(a){gOe.call(this);this.e=a} function EJe(a){this.c=a;KN.call(this)} function OJe(a){this.d=a;CW.call(
zg(a){this.b=a;Szg.call(this)} function Czg(a){this.b=a;DJf.call(this)} function FVG(a){this.b=a;V9e.ca
Jg(){JJg=CEh;hr());IJg=new OJg} function Uhg(a,b){b.b=pSd(a);b.c=pSd(a)} function NJg(a,b){a.b=b;OE(a.B)
unction Y7g(a,b){f8g(a,b,b);f8g(a,d,b)} function f8g(a,b){sVd(a,b,b);sVd(a,c,b)} function hyd(a,b){CLd(a
f(a,b))} function Z3d(a,b,c){var d;d=c;$3d(a,b,d)} function ZAd(a,b,c){$Ad.call(this,a,b,c)} function Cc
t.call(this);this.kc=true} function ox(a,b){nx.call(this,a);this.k=b} function DN(a,b){VG();this.b=a;WG(
function t(a){return !a.k?null:Avb(a.k)} function Ecb(a){return a=null?xHh:Uk(a)} function Bl(a,b){ret
jF(a.e,I8(a.s,b),false)} function jKf(a){Igb(a,a,kf(a).c!=0,false)} function jwg(){jwg=CEh;hr();iwg=Sph(
function lqe(a,b){hr();iqe.call(this,a,b)} function XRe(a,b){VRe();this.c=a;this.b=b} function bSe(a,b)
){mSe();this.b=a;this.c=b} function KDF(a,b){JDf();this.b=a;this.c=b} function jNf(a,b){hNf();this.b=a;t
function qIf(a,b){SHF.call(this,a,b,true)} function Kxf(a,b){pEd.call(this,a,null,b)} function Pgf(a,b)
){a.g.cm(a.g.Xl()-1);Gch(a)} function O7(a){if(a.e){a.e.Fd();j.a.Fd()}} function P8e(a){B8e(a,b,a.b.i.b)
} function _Kg(a){UMg();rOg(TMg,new gLg(a))} function kJf(a,b){NJ(a);a.qA(b,new qJf(a))} function tjh(a
a){OE((BIg(),AIg().B);SKf(a.b)} function bLe(a){XSf(new WRe(a.b.b.b.c.j))} function zEe(){zEe=CEh;yEe=Pb(
){Z$F=CEh;X$F=_$f());Y$F=a_f()} function Pjh(){PJh=CEh;Njh=Rjh());Ojh=Sjh()} function Trf(){Trf=CEh;SrF=(
t(a.d.z,b,c);Ur(a.d.v)} function Itf(a,b){oBb.call(this,b);this.b=a} function jtF(a,b){oBb.call(this,b);
```

# JavaScript as a serious programming language



# Wish List for a JavaScript Alternative

## Java Developer

- ▶ Static Typing
  - ▶ Instant feedback
- ▶ Tooling
  - ▶ Autocompletion, Refactoring
- ▶ Familiar language features
  - ▶ Modules, classes, interfaces, enums, generics, dependencies

## JavaScript Developer

- ▶ Preserve elegance of JS
- ▶ Preserve syntax compatibility with JS
- ▶ No loss in performance
  - ▶ for runtime type checking
- ▶ No cryptic generated code
- ▶ Ability to use JS libs
  - ▶ hassle-free

# Some TypeScript Facts

- ▶ „large scale application development“
- ▶ Language, not a framework
- ▶ Cross-compiles to JavaScript
- ▶ Produces idiomatic JavaScript



# Some TypeScript Facts

- ▶ Doesn't try to replace JavaScript
- ▶ Offers missing language features
  - ▶ Anticipates ES6
- ▶ Is a superset of JavaScript
  - ▶ Promises easy migration of JS code
- ▶ JS libs can be used easily
- ▶ (Optional) static typing

# Some TypeScript Facts

- ▶ Developed by Microsoft
- ▶ Open Source
  - ▶ Apache 2 License
- ▶ Hosted on Github
  - ▶ <https://github.com/microsoft/typescript>
- ▶ Compiler implemented in TypeScript



# Language Features

- ▶ Type annotations
- ▶ Type inference
- ▶ Compile time type checking
- ▶ Optional, default and rest parameters
- ▶ Classes
- ▶ Interfaces
- ▶ Structural typing
- ▶ Arrow function expressions
- ▶ Enums
- ▶ Generics
- ▶ Modules
- ▶ Tuple types
- ▶ Union types and type guards

# Tooling

- ▶ **Compiler**
  - ▶ Supports different ECMAScript target versions
  - ▶ Supports different module systems
  - ▶ May produce SourceMaps
- ▶ **IDE Support**
  - ▶ Based on Language Service

# IDE Support

- ▶ TypeScript Playground
- ▶ Visual Studio, VS Express for Web
- ▶ Visual Studio Online
- ▶ Eclipse
- ▶ JetBrains IntelliJ / WebStorm / PHPStorm
- ▶ Text Editors
  - ▶ Sublime Text, Vim, Emacs
- ▶ Cloud9 IDE
- ▶ ...

# TypeScript Team

The TypeScript team includes

- ▶ Anders Hejlsberg
- ▶ Steve Lucco (Chakra)
- ▶ Luke Hoban (ECMA)

# Some Prominent Products using TypeScript

- ▶ TypeScript Compiler
- ▶ Mozilla Shumway
  - ▶ a Flash VM and runtime written in TypeScript/JavaScript
- ▶ Hawt.io
  - ▶ a modular web console for managing your Java stuff
- ▶ Visual Studio Online

# History

10/2011 *DART revealed*

**10/2012** TypeScript revealed (0.8)

**04/2014** TypeScript 1.0

**10/2014** TypeScript 1.1

10/2014 *AtScript revealed*

**11/2014** TypeScript 1.3

11/2014 *Flow revealed*

**01/2015** TypeScript 1.4



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

Douglas Crockford

# TypeScript Features

# Type Annotations

```
var num = 42;
```

```
var num: number = 42;
```

```
function mult(a: number, b: number) : number {  
    return a*b;  
}
```

```
var x: any;
```

# Optional Arguments

```
function mult(a: number, b?: number) {  
  if (!b) {  
    return a;  
  }  
  return a * b;  
}
```

# Default Values

```
function mult(a: number, b: number = 1) {  
    return a * b;  
}
```

# Rest Parameters

```
function mult(...factors: number[]) {  
    var result = 1;  
    for (var i = 0; i<factors.length; i++) {  
        result *= factors[i];  
    }  
    return result;  
}
```

```
mult(2, 3, 4);
```

# Overloading

```
function someFunc(arg: string): void;
```

```
function someFunc(arg: number): void;
```

```
function someFunc(arg: any): void {  
    // implementation goes here  
}
```

# Classes

```
class Content {  
    private title: string;  
  
    constructor(title: string) {  
        this.title = title;  
    }  
  
    toString(): string {  
        return this.title;  
    }  
}
```



# Interfaces

```
interface Printable {  
    print(): void;  
}  
  
class Foo implements Printable {  
    print(): void {  
        console.log("Hello from Foo");  
    }  
}
```

# Structural Typing

- ▶ Languages like Java have a **nominal type system**
- ▶ TypeScript has a **structural type system**
- ▶ Two types are **compatible** if they share the **same structure**
  - ▶ A private fields makes a class incompatible with everything else except its super class
- ▶ <https://github.com/Microsoft/TypeScript/wiki/Type%20Compatibility>

# Structural Typing - Example

```
interface Point {  
  x: number;  
  y: number;  
  z?: number;  
}  
  
function acceptPoint(point: Point) {  
  // ...  
}  
  
acceptPoint( {x:1, y:2} );  
acceptPoint( {x:1, y:2, z:5} );
```

# Inheritance and Polymorphism

```
class Book extends Content {
    constructor(title: string, public isbn: string) {
        super(title);
    }

    toString() {
        return super.toString() + " (ISBN: " + this.isbn + ")";
    }
}
```

# Overload on constants

```
interface Document {  
    createElement(tagName: string): HTMLElement;  
    createElement(tagName: "canvas"): HTMLCanvasElement;  
    createElement(tagName: "div"): HTMLDivElement;  
    createElement(tagName: "span"): HTMLSpanElement;  
    // ...  
}
```

```
document.createElement("button"); // HTMLButtonElement
```

# Modules

```
module con {  
  export class Content {  
    // ...  
  }  
}  
  
///  
// <reference path="Content.ts" />  
import Content = con.Content;  
  
new Content("Some fancy Content");
```

# Enums

```
enum Color {  
    BLUE, GREEN, RED, WHITE, BLACK  
}
```

```
var color = Color.RED;
```

# Generics

```
class ContentHolder<T> {  
    public content: T;  
};
```

```
var c = new ContentHolder<number>();  
c.content = 42;
```



# Ambient Declarations

- ▶ Provide static type information for JavaScript libraries
  - ▶ E.g. statically typed jQuery
  
- ▶ Popular source: Definitely Typed
  - ▶ <https://github.com/borisyankov/DefinitelyTyped>

# Arrow Function Expressions

lexical scoping of this

```
class ArrowFunctionDemo {  
    private message = "Hello from Arrow Function";  
    run() {  
        setTimeout( () => {  
            console.log(this.message);  
        }, 1000 );  
    }  
}
```

```
new ArrowFunctionDemo().run();
```

# Tuple Types

- ▶ Provide typing for the elements of an array

```
var tuple: [number, string] = [42, "Fred"];
```

# Union Types

- ▶ Value can have one of multiple types

```
var commandline: string[]|string;
```

```
commandline.length === 0
```

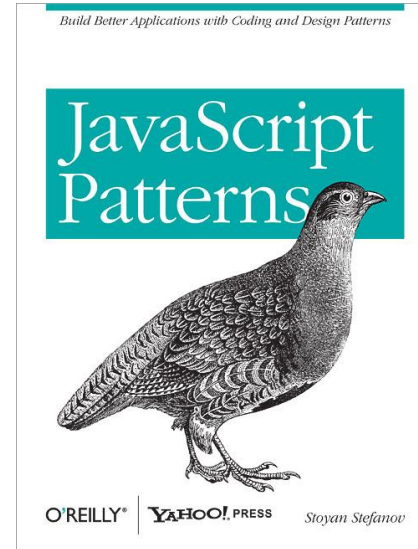
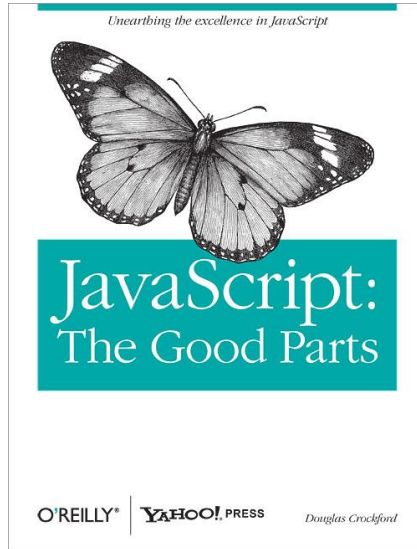
# Type Guards

```
function formatCommandline(c: string[]|string) {  
    if(typeof c === 'string') {  
        return c.trim();  
    } else {  
        return c.join(' ');  
    }  
}
```

# Experiences

# Employee Training

- ▶ No JavaScript training necessary?
  - ▶ A solid JavaScript foundation is a must!

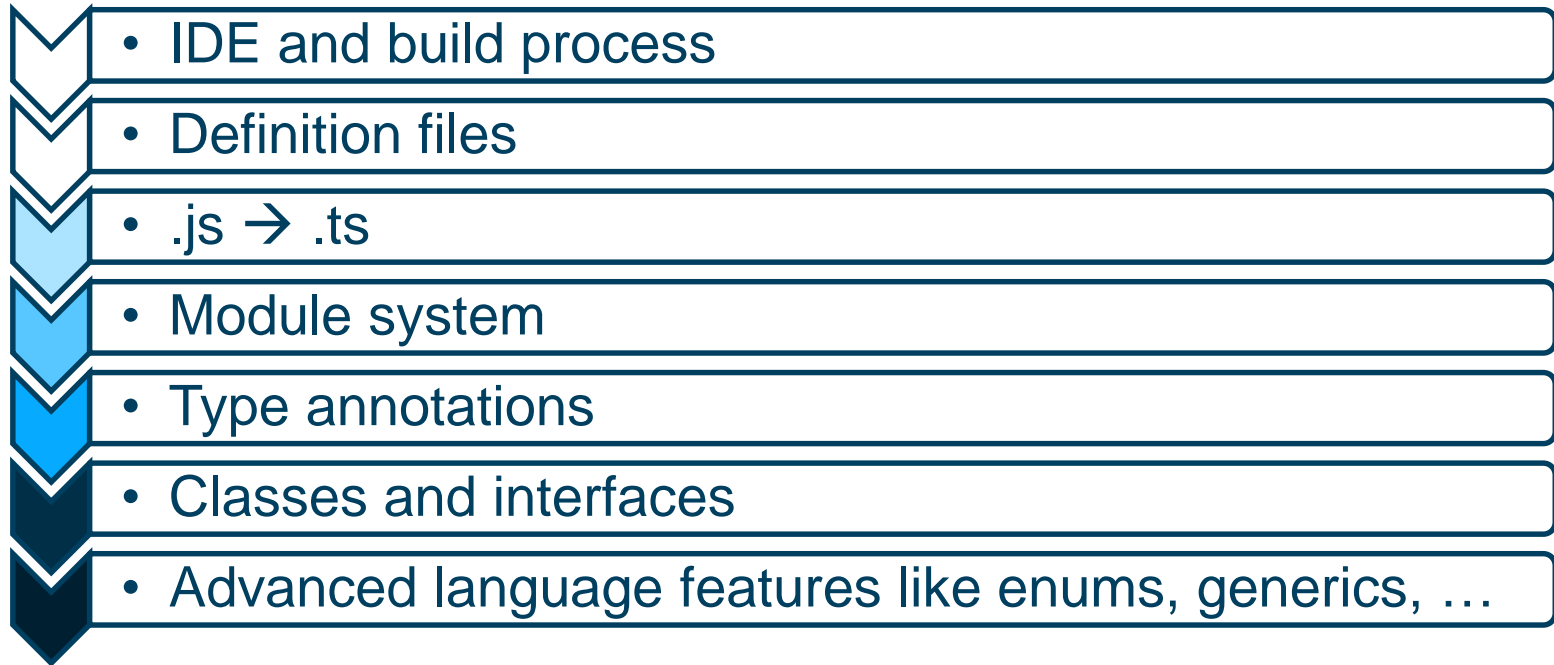


# Migration of existing code

- ▶ Mix of TS and JS code possible
- ▶ Code may be migrated step by step
- ▶ You can go as far as you wish



# Migration Strategy

- 
- IDE and build process
  - Definition files
  - .js → .ts
  - Module system
  - Type annotations
  - Classes and interfaces
  - Advanced language features like enums, generics, ...

# Integration into the Build-Process

## ▶ TypeScript Gradle Plugin

▶ <https://github.com/sothmann/typescript-gradle-plugin>

1. Compile TypeScript code
2. Concatenate (+optimise?) JavaScript code
3. Include in WAR file

```
war {  
    into("js") {  
        from compileTypeScript.outputs  
    }  
}
```

# Developer Satisfaction

- ▶ Features and tooling
- ▶ Feels lightweight
- ▶ Developers become more open to frontend development
- ▶ Java is still preferred by most

# any: use with caution

- ▶ the *any* type annuls type checking
- ▶ Solves problems with typing easy and fast
  - ▶ It's alluring to use it
- ▶ Is your best friend while migrating the codebase to TS
- ▶ Creates problems, because developers trust on the type system
  - ▶ No compile errors -> Code is error free
- ▶ Put a TODO on these code lines

# Be afraid of the implicit any

- ▶ Use compiler flag `noImplicitAny`
  - ▶ *“Warn on expressions and declarations with an implied 'any' type”*

# Don't trust your types!

(when you are communicating with the backend)

- ▶ Remote communication is likely a point for the type system to break
- ▶ Backend may provide types that doesn't match the expected types in the frontend
- ▶ Introduce runtime type checking / type conversion
  - ▶ `_.isString()`, `_.isBoolean()` and `_.isDate()` (Underscore) may be helpful

# Type Conversion in Backbone Models

```
class MyModel extends Backbone.Model {  
  getComment(): string {  
    return this.get("comment");  
  }  
  
  getDate(): Date {  
    return moment(this.get("date")).toDate();  
  }  
}
```

# Outlook



# A Common Standard?

“The TypeScript team is working with both the Flow and AtScript teams to help ensure that resources that have already been created by the JavaScript typing community can be used across these tools. There's a lot these projects can learn from each other, and we're looking forward to working together going forward and creating the best tools we can for the JavaScript community. In the long term, we will also be working to fold the best features of these tools into ECMAScript, the standard behind JavaScript.”

Jonathan Turner, Microsoft

# Is this the future of the web?



**wonderwhy-er** · 3 months ago

Hmm, when Facebook, Google and Microsoft are working together it kinda looks like future of the web...

2 ^ | v · Reply · Share ›

# Conclusion

- ▶ Productivity improved
- ▶ Maintainability improved
- ▶ Developer satisfaction improved
- ▶ Lack of a shared codebase is sometimes annoying

# Thank You

**Sönke Sothmann**  
sso@tonbeller.com