Orientation in Objects

TypeScript

Das JavaScript der Zukunft?

Orientierungspunkt 12/2013

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.0

Ihr Sprecher

Orientation in Objects

Sönke Sothmann

Trainer, Berater, Entwickler

Schwerpunkte
Agile Entwicklungsprozesse,
Entwicklung von Web-Anwendungen,
Web als Plattform

© 2013 Orientation in Objects GmbH

Gliederung

- JavaScript
- Überblick über JavaScript Alternativen und Abgrenzung
- Vorstellung TypeScript
- Marktakzeptanz

© 2013 Orientation in Objects GmbH

Gliederung

- **JavaScript**
- Überblick über JavaScript Alternativen und Abgrenzung
- Vorstellung TypeScript
- Marktakzeptanz

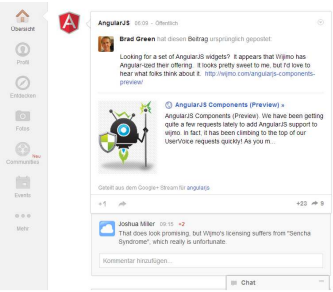
© 2013 Orientation in Objects GmbH

Das Web heute - aus Anwendersicht

Orientation in Objects

- Benutzerfreundlich
- “Rich User Experience”
- Hohe Geschwindigkeit, geringe Latenz
- Gute Ideen

- Messlatte: z.B. Google-Dienste



Webtechnologien erobern neue Domänen

Orientation in Objects

- Desktop Web Anwendungen
- Mobile Web Anwendungen
- „Native“ Mobile Anwendungen
  - iOS, Android, Windows Phone, Blackberry, …
- Desktop Anwendungen
  - Windows Store Apps
  - Chrome / ChromeOS Apps
  - Ubuntu Apps
  - …
- Server-Anwendungen
  - Node.js

Die Bedeutung von JavaScript

Orientation in Objects

- JavaScript läuft überall
- „Write once, run anywhere“ für Frontends
- Wird uns noch Jahre begleiten

© 2013 Orientation in Objects GmbH

Programmiermodell im Web

Orientation in Objects

- Entwicklung ist
  - Aufwändig / teuer
  - Fehleranfällig
  - Nervenzehrend
- Verantwortlich dafür
  - Schwächen der Sprache JavaScript
    - **Keine Klassen, Module, Abhängigkeiten**
    - **Keine statische Typisierung**
  - Tooling
  - Browserinkompatibilitäten
  - Kein standardisiertes Komponentenmodell

© 2013 Orientation in Objects GmbH

JavaScript Highlights

- Objektliterale
- Objekte als Maps nutzen
- Array Notation (vergleichbar mit Java List)
- Closures

© 2013 Orientation in Objects GmbH

JavaScript Codebeispiele

```
move({x: 3, y: 5});

var map = {a:1, b:2}
map["a"]
map.a

var arr = ["Hallo", "JavaScript"];
arr.push("Welt")
arr.join(" ");

button.click(function() {
   // …
});

var myFunction = …
button.click(myFunction);
```

© 2013 Orientation in Objects GmbH

JavaScript

- Vorteile
  - Elegante Sprachkonstrukte
- Nachteile
  - Fehlende statische Typisierung
    - **Schwaches Tooling**
  - Ungewohntes Vererbungskonzept
    - **Keine Klassen**
    - **Prototyp. Vererbung ist weniger deklarativ**
  - Keine Abhängigkeiten, Module
    - **Müssen über Bibliotheken nachgerüstet werden**
  - Magere Funktionsbibliothek
  - Viele Stolperfallen
- JavaScript wird uns noch ein Weilchen begleiten…

© 2013 Orientation in Objects GmbH

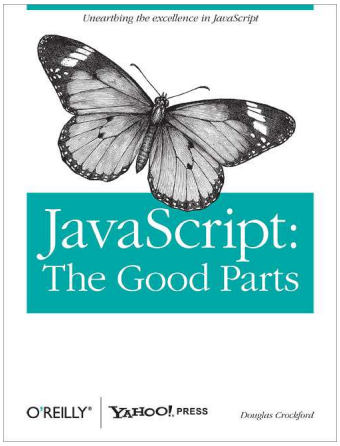
ECMAScript 6

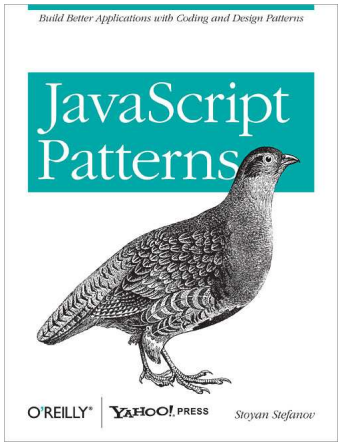
- Kommender JavaScript Standard
- Bringt
  - Klassen
  - Module
  - Abhängigkeiten
- Aber…
  - Noch nicht fertig spezifiziert
    - **Noch kein nennenswerter Browser Support (http://kangax.github.io/es5-compat-table/es6/)**
  - Statische Typisierung nicht auf der Roadmap

© 2013 Orientation in Objects GmbH

JavaScript als Programmiersprache

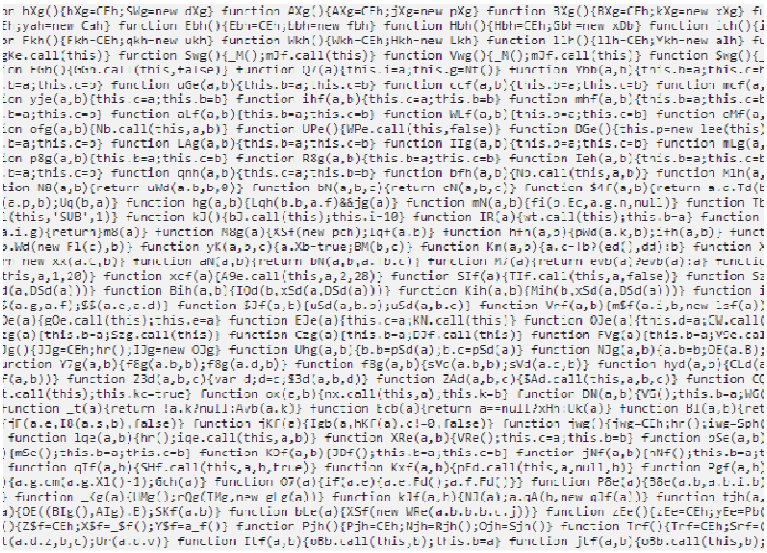
Orientation in Objects





JavaScript als „ByteCode“

Orientation in Objects



JavaScript Alternativen

Orientation in Objects

Readable JavaScript
Easy JS integration

JavaScript as Bytecode
Cumbersome JS integration

CoffeeScript

DART

TypeScript PREVIEW

HAXE

Script#

emscripten

CoffeeScript

Orientation in Objects

- Ruby ähnliche Syntax
  - Beliebt unter Ruby on Rails Entwicklern
- Syntactic sugar
- Bietet Klassen
- Erzeugt ideomatisches JavaScript
- Sprache, kein Framework
- Keine statische Typisierung

CoffeeScript

GWT

- Weit entfernt von JavaScript
  - JavaScript als ByteCode, hochoptimiert
- Einbindung externer Bibliotheken umständlich und fehleranfällig
  - Eigenes Ökosystem mit nativen GWT Bibliotheken
  - JavaScript Bibliotheken nur wenn unbedingt nötig
- Sprache
  - Java, damit statische Typisierung , gutes Tooling
- Framework
- **Code-Sharing zwischen Front- und Backend**
- Kein Drop-in Replacement

© 2013 Orientation in Objects GmbH

DART

- Enthüllt 10/2011 von Google
- Sprache, kein Framework
- Ausführung
  - In Laufzeitumgebung
  - Cross-compiled zu JavaScript

© 2013 Orientation in Objects GmbH

Ziele / Vorteile von DART (1)

- Strukturierter / „large scale application development“
- Bietet
  - Statische Typisierung
  - Klassen
  - Module
  - Abhängigkeiten

© 2013 Orientation in Objects GmbH

Ziele / Vorteile von DART (2)

- Soll langfristig **JS** als Sprache des Webs **ersetzen**
- Für Client und Server
- Gutes Tooling (Eclipse)

© 2013 Orientation in Objects GmbH

Nachteile von DART

- Neue Plattform
- Fehlende Bibliotheken (3rd Party)
- Fehlende Community
- Kann JS aufrufen, ist aber umständlich

- Keine Unterstützung der VM durch andere Browserhersteller

DART VM wird es schwer haben

"I guarantee you that Apple and Microsoft (and Opera and Mozilla, but the first two are enough) will never embed the Dart VM."

- Brendan Eich (Mozilla), Inventor of JavaScript

TypeScript

- Enthüllt 10/2012 von Microsoft
- Sprache, kein Framework
- Cross-compiled zu JavaScript
  - **Generiert idiomatisches JavaScript**

Ziele / Vorteile von TypeScript (1)

- Strukturierter / „large scale application development“
- Bietet
  - Statische Typisierung
    - **Optional, Type Inference**
  - Klassen
  - Interfaces
  - Enums
  - Module
  - Abhängigkeiten

Ziele / Vorteile von TypeScript (2)

Orientation in Objects

- **JavaScript nicht ersetzen**
  - Schwächen ausgleichen
- **Superset** von JavaScript
  - Projekte **leicht migrierbar**, bestehende **JS Bibliotheken nutzbar**
- Declaration Files
  - Bestehende **JavaScript Bibliotheken nachträglich typisieren**
- Implementiert zukünftige ES6 Features schon heute, hält sich an Spec
- Gutes Tooling (Visual Studio, Eclipse)

© 2013 Orientation in Objects GmbH

Schöpfer von TypeScript

Orientation in Objects

- Anders Hejlsberg
- Steve Lucco (Chakra)
- Luke Hoban (ECMA)
- (Erich Gamma)

© 2013 Orientation in Objects GmbH

Warum gerade TypeScript? (1)

Orientation in Objects

- Statische Typisierung + Tooling
- Alle fehlenden Features vorhanden
- Sehr dicht an JS = Investment Protection
  - Migration von Bestandscode
  - Integration von Bestandscode
  - Integration von JS Bibliotheken
  - JavaScript Wissen bleibt erhalten / kann genutzt werden

© 2013 Orientation in Objects GmbH

Warum gerade TypeScript? (2)

Orientation in Objects

- Drop-In Replacement
  - Überall, wo wir schon JS haben, oder
  - Überall, wo Kunde JavaScript will
- Generiert ideomatisches JavaScript
  - Exit Strategie
    - **TypeScript fallen lassen, mit den generierten JS Files weiterentwickeln**
- Nicht die sauberste, aber die aktuell **pragmatischste** Lösung, um das Programmiermodell zu verbessern

© 2013 Orientation in Objects GmbH

Ist die Zeit reif für TypeScript?

Orientation in Objects

- Sprache noch „Preview“
- Schon 0.9, nächster Sprung wird auf 1.0 sein
- Gutes Tooling vorhanden
- Noch gibt es einige Bugs
  - Man ist im Moment noch Early Adopter

© 2013 Orientation in Objects GmbH

Gliederung

Orientation in Objects

- JavaScript
- Überblick über JavaScript Alternativen und Abgrenzung
- **Vorstellung TypeScript**
- Marktakzeptanz

© 2013 Orientation in Objects GmbH

Tooling

- Compiler / Kommandozeile
  - Kann SourceMaps produzieren
  - Kann Declaration Files produzieren
  - Kompilieren vieler TS Dateien in eine große JS Datei möglich
  - EcmaScript Zielversion wählbar
    - **Aktuell ES3 oder ES5**
  - Kein implizites any erzwingbar
- IDEs
  - Language Service

© 2013 Orientation in Objects GmbH

IDEs

- (TypeScript Playground)
- Visual Studio 2012/2013
  - Web Essentials Plugin
- Visual Studio Express 2012/2013 for Web
- Visual Studio Online
- Eclipse
- JetBrains IntelliJ / WebStorm / PHPStorm
- Text Editors
  - **Sublime Text, Vim, Emacs**
- Cloud9 IDE

© 2013 Orientation in Objects GmbH

IDEs - Visual Studio

Orientation in Objects

```
class Greeter {
    element: HTMLElement;
    span: HTMLElement;
    timerToken: number;

    constructor(element: HTMLElement) {
        this.element = element;
        this.element.innerHTML += "The time is: ";
        this.span = document.createElement('span');
        this.element.appendChild(this.span);
        this.span.innerText = new Date().toUTCString();
    }

    start() {
        this.timerToken = setInterval(() => this.span.innerHTML
            = new Date().toUTCString(), 500);
    }

    stop() {
        clearTimeout(this.timerToken);
    }

}

window.onload = () => {
    var el = document.getElementById('content');
    var greeter = new Greeter(el);
    greeter.start();
};
```

```
var Greeter = (function () {
    function Greeter(element) {
        this.element = element;
        this.element.innerHTML += "The time is: ";
        this.span = document.createElement('span');
        this.element.appendChild(this.span);
        this.span.innerText = new Date().toUTCString();
    }
    Greeter.prototype.start = function () {
        var _this = this;
        this.timerToken = setInterval(function () {
            return _this.span.innerHTML = new Date().toUTCString();
        }, 500);
    };

    Greeter.prototype.stop = function () {
        clearTimeout(this.timerToken);
    };
    return Greeter;
})();

window.onload = function () {
    var el = document.getElementById('content');
    var greeter = new Greeter(el);
    greeter.start();
};
//@ sourceMappingURL=app.js.map
```


IDEs - Eclipse

Orientation in Objects

```
class Greeter {
    private greeting: string;
    constructor(message: string) {
        this.greeting = message;
    }
    greet() {
        return "Hello, " + this.greeting;
    }
}

var greeter = new Greeter("world");

var button = new HTMLButtonElement();
button.
```

Typ Annotationen

```
var i = 1;

var i: number = 1;


function f(i: number) : number {
   // …
}


var x: any;
```


Optionale Argumente, Default-Werte und Rest-Parameter

```
function multiply(val: number, factor?: number = 1) {
   return val * factor;
}



function add(...numbers: number[]) {
   var result = 0;
   for (var i = 0; i<numbers.length; i++) {
       result += numbers[i];
   }
   return result;
}
```

Überladen von Funktionen

```
function doIt(a: string): void;

function doIt(a: number): void;

function doIt(a: any): void {
  // …
}
```


Klassen

```
class Content {
  private title: string;

  constructor(title: string) {
      this.title = title;
  }

  toString(): string {
      return this.title;
  }
}
```

Klassen - Konstruktor Magie

```
class Content {
  constructor(public title: string) {
  }

  toString(): string {
      return this.title;
  }
}
```


Vererbung und Polymorphie

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

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

Interfaces

```
interface Printable {
  print(): void;
}

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


Overload on constants

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

Structural Typing

```
interface Point {
  x: number;
  y: number;
  z?: number;
}

function acceptPoint(point: Point) {
  // …
}

acceptPoint( {x:1, y:2} );
acceptPoint( {x:1, y:2, z:5} );
```

© 2013 Orientation in Objects GmbH

Arrow Function Expressions

```
var add = (a: number, b: number) => {
  return a + b;
}

var print = s => {
  console.log(s);
}

var doubleIt = (x:number) => x*2;

var doIt = () => { console.log("hello world!"); }
```

© 2013 Orientation in Objects GmbH

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();
```


Methodensignaturen erzwingen - am Beispiel von Callbacks

```
class MyDialog {
  private handler: { (arg: string): number; };

  addOkHandler(handler: { (arg: string): number; } ) {
      this.handler = handler;
  }

  run() {
      this.handler("foo");
  }
}
```

Methodensignaturen erzwingen - am Beispiel von Callbacks (2)

Orientation in Objects

```
var dialog = new MyDialog();
dialog.addOkHandler(function (arg: string) {
   console.log(arg);
   return 1;
});

// NOT OK - wrong type
dialog.addOkHandler(function (arg: number) {
   return 1;
});

// OK: the consumer is not forced to accept the arg
dialog.addOkHandler(function () {
   return 1;
});
```

© 2013 Orientation in Objects GmbH

Module

Orientation in Objects

```
module cai {
    export class Content {
        // …
    }
}



/// <reference path="Content.ts" />
import Content = cai.Content;

new Content("Test");
```

© 2013 Orientation in Objects GmbH

Enums

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


Generics

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

var c = new Container<number>();
c.content = 5;
```

Ambient Declarations

- JavaScript Bibliotheken mit statischen Typinformationen versehen
- Z.B. von Definitely Typed

```
1  /// <reference path="jquery.d.ts" />
2
3
4  $("
5      ▲ 8 von 8 ▼ $(selector: string, context?: any): JQuery
6
```


Ambient Declarations - für klassenartige Bibliotheken

```
declare module Library {
  export class AClass {
      doIt(arg: string): void;
  }
}

declare var lib: Library;
```

Ambient Declarations - für nicht klassenartige Bibliotheken

```
interface Library {
  AClass: {
      doIt(arg: string): void;
  };
}

declare var lib: Library;
```


Migration

- Mix aus JS und TypeScript möglich
- Bestehender JS Code
  - kann Schritt für Schritt migriert werden
  - die Migration kann unterschiedlich weit getrieben werden

Mögliche Migrationsstrategie

Orientation in Objects

1.
   - Dateiendung ändern
     - .js => .ts
     - erste Fehler durch Type Inference
2.
   - Typinformationen einstreuen (primitive Datentypen)
3.
   - Umstellen auf Interfaces und Klassen
4.
   - Umstellen auf TypeScript Sprachkonstrukte wie Enums und Generics

© 2013 Orientation in Objects GmbH

Gliederung

Orientation in Objects

- JavaScript
- Überblick über JavaScript Alternativen und Abgrenzung
- Vorstellung TypeScript
- **Marktakzeptanz**

© 2013 Orientation in Objects GmbH

Meinung von Douglas Crockford

Orientation in Objects

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

Douglas Crockford

Marktakzeptanz von TypeScript

Orientation in Objects

- Erfahrungsberichte von Early Adoptern überwiegend sehr positiv
- Noch kein Hype
  - Wenig in den Medien Präsent (z.B. Magazine)
  - Wenig Vorträge auf Konferenzen

Firmen, die TypeScript nutzen / investieren

Orientation in Objects

Microsoft®

- Visual Studio Online (Project Lead Erich Gamma)
- Team Foundation Server
- BING Bildersuche
- Xbox Music
- Touchdevelop.com (Touchscreen IDE für App Entwicklung)
- Socl
- TypeScript Compiler

© 2013 Orientation in Objects GmbH

Firmen, die TypeScript nutzen / investieren (2)

Orientation in Objects

- Palantir Technologies
- hawt.io (HTML5 web management UI)
- 3D-/Game-Engines (StarlingJS, Turbulenz, Away3D)
- featuremap.co (Online Project management software)
- zud.io (Windows Azure Management Tool)
- HittaHem.se (Real Estate)
- jRIAppTS (RIA Framework)

© 2013 Orientation in Objects GmbH

Definition Files für JavaScript Bibliotheken

- Viele JS Bibliotheken haben schon Definition Files
  - https://github.com/borisyankov/DefinitelyTyped
  - Für über 200 Projekte
  - Über 300 Contributors
- Emöglicht besseren Tool Support und Typsicherheit
- Liefern JS Frameworks zukünftig TS Definition Files mit?
  - Bisher keine Anzeichen

© 2013 Orientation in Objects GmbH

TypeScript Aktivität in Foren

- Official Forum, über 5000 Posts
  - http://typescript.codeplex.com/discussions
- Auf Stackoverflow >1500 tagged questions
  - zum Vergleich, Dart hat 1800, ist aber schon knapp ein Jahr älter und schon Version 1.0

© 2013 Orientation in Objects GmbH

Meinung zu TS in den Communities der JS Frameworks

Orientation in Objects

- Viele finden es sehr interessant
- Einige haben bereits TS zusammen mit „ihrem“ Framework genutzt
  - größtenteils sehr angetan
  - Step by step Migration tatsächlich einfach
  - Generierter JS Code ist tatsächlich „schön“
    - **Exit Strategie: Zurück zu JavaScript auf Grundlage des Kompilates**
- Besonders viel Interesse in der Microsoft Welt
- Skepsis
  - Viele warten erstmal auf 1.0
  - Es kommt von Microsoft

© 2013 Orientation in Objects GmbH

Fazit

Orientation in Objects

- Leichte Migration und Integration
- Mit beliebigen JS Frameworks verwendbar
- Bewährte Sprachkonstrukte nachgerüstet
- Statische Typisierung
  - So viel statische Typisierung wie möglich
    - **Dynamische Typisierung wo es Sinn macht**
  - Schnelles Feedback -> Fehler schnell erkennen
  - Besseres Tooling
- Investitionssicherheit (Knowhow, Exit-Strategie)
- Entscheidung FÜR JavaScript
  - Wir rüsten uns nur fehlende Features nach

© 2013 Orientation in Objects GmbH

Links

Orientation in Objects

- Real world projects written in TypeScript
  - http://typescript.codeplex.com/discussions/430577
- (Mostly) Painlessly Migrating a 3D Game Engine to TypeScript
  - http://hardcodeded.blogspot.jp/2013/02/mostly-painlessly-migrating-3d-game.html
- DefinitelyTyped
  - https://github.com/borisyankov/DefinitelyTyped

© 2013 Orientation in Objects GmbH

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de