TypeScript

Ein "Superset" von JavaScript

... entwickelt von Anders Hejlsberg (Microsoft)

erweitert ECMA2015 / ECMA2016 um statische Typisierung

TypeScript: Website



http://www.typescriptlang.org/

Aktueller Stand von TypeScript:

Erste stabile Version: **Typescript 1.0** (Mai 2012)

(Prä-implementierung geplanter ECMA6 features.)

Milestone 1: TypeScript 1.6 (Dez. 2016)

(Unterstützung von ECMA6 classes, ECMA6 Generators)

Milestone 2: Typescript 2.0 (Dez. 2016)

(Typanalyse nun auch basierend auf Controlflow möglich.)

Aktuelle Version: **TypeScript 2.6** (November 2017)

ECMAScript 5

ECMAScript 6

classes, inheritance default, rest, spread arguments

arrow functions modules, loaders

template strings promises let, const, destructuring

etc...

TypeScript

type annotations enums

> type checking decorators

introspection generics

> interfaces etc...

JavaScript vs. TypeScript:

- Wie wir eben gesehen haben ist beides quasi dasselbe...
- TypeScript stellt einen (optionalen) Überbau in Form der Typisierung bereit.

Vorschlag: Wir betrachten die Syntaxstrukturen von ECMA6 zunächst in "Reinform" und betrachten dann erst die "Dreingaben" von TypeScript in den jeweiligen Fällen.

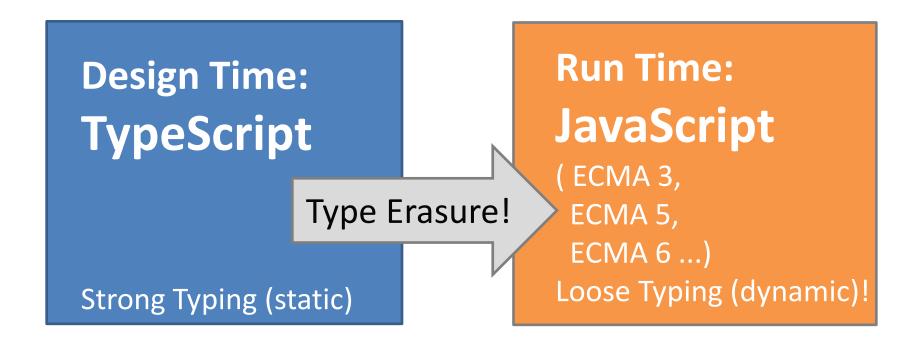
Ziele von TypeScript:

- Potentielle Fehler statisch erkennen
- "Strongly typed"
- Static type checking zur Compile time
- Superset: jedes valide JavaScript ist valides TypeScript
- Features wie Types verfügbar zur Design time
- Kompiliert nach ECMA3, ECMA5 oder ECMA6
- Open Source Apache license

Der Compiler...

- Parst den Code, vollzieht das Typechecking und kompiliert den TS-Code nach JavaScript
- Stellt im Editor die Sprachunterstützung bereit (IntelliSense, Typinfos etc.)
- Integriert sich für die "Design Time" in die IDE

Compilevorgang...



... die Typen haben wir also nur im Editor! Nicht mehr im Browser!!!

JavaScript vs. TypeScript:

- TypeScript ist <u>keine Run-Time-Sprache</u>. Es bleibt uns nichts anderes übrig, als zu *kompilieren*, wenn wir unser Programm arbeiten sehen wollen.
- Wir kompilieren nach ECMA6. Im Rahmen der vorgestellten Syntaxstrukturen wird dies in aktuellen Browsern lauffähig sein.

Ausnahme: Aktuell werden ECMA6-Module noch nicht ausreichend unterstützt! Wir werden daher einen externen Loader verwenden (SystemJS).

Typen in TypeScript...

Static types (während des Codens): "design time"

Dynamic types (loose) nach Compile in JS: "run time"

TS-Typing arbeitet mit

- Type annotation (Typen werden explizit zugewiesen)
- Type inference (Typen werden vom Kontext abgeleitet)

... ähnlich wie bei C#!