TypeScript vs JavaScript

PB138 Moderní značkovací jazyky
Duben 2021

Tomáš Pitner

TypeScript – Motivation

- JavaScript – was a decent simple language for manipulation with objects in web pages
- Eventually, it became a universal language for both **client & server side** (Node.js), for general **scripting**, for defining **queries** and **views** in NoSQL databases
- **1,500,000+** projects at npmjs.com

JavaScript – dynamic weak typing

- Issues with **weak and dynamic typing**
- Urgent need for a **static type system** emerged

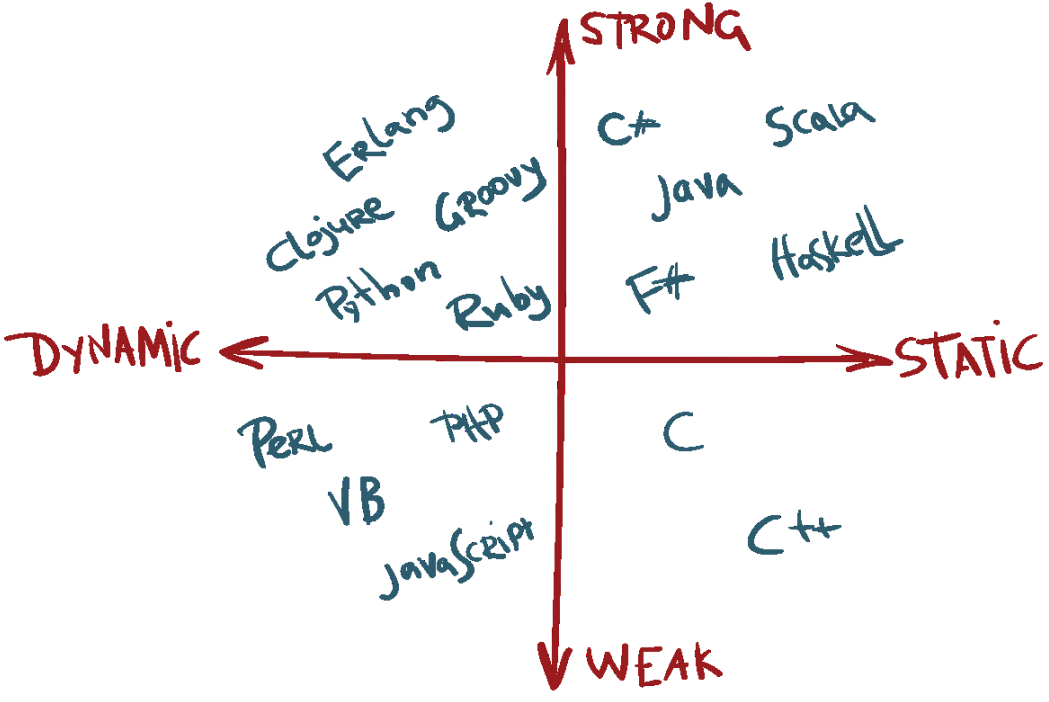
JS – **dynamic** typing

- JS is a **dynamically typed** language
- Objects behavior is determined in runtime – objects offer & behave according to what they have in runtime not by declaration
- **Type checking in runtime**
- JS does not (did not) have classes
- Generaly about language typing, see https://medium.com/android-news/magic-lies-here-statically-typed-vs-dynamically-typed-languages-d151c7f95e2b

JS – **weak** typing

- JS is a **weakly typed** language – opposite of **strong typing** where more or less allowing only those automatic type conversions that *do not lose information*), one can refer to the process as ***Strongly*** typed, if not, as **We*akly*** typed.

Typing in Languages



Java, Python vs PHP

- **Java** – strong static typing

```
String temp = “Hello World!”;
temp = 10; // error in compile time
```

- **Python** – strong dynamic typing

```
temp = “Hello World!”
temp = temp + 10; // error in RT
```

Java, Python vs PHP

- **PHP** – weak dynamic typing

```
$temp = “Hello World!”;
$temp = $temp + 10; // no error as type coercion occurs
in RT
```

JS – **issues** with typing

- Type **coersion** (enforcement) – type must be converted in order to do certain operation
  - Ie. String concatenation in JS

```
var value1 = “5“; // string
var value2 = 9; // number
let sum = value1 + value2; // must be all strings
// result: sum = „59“
```

JS – **issues** with typing

- Type **conversion** – in static languages does not loose information while in dynamic may loose information
- May be done implicitly (more error prone) or explicitly (more under control, done consiously)

```
// implicit
let a = 5 + 2.0;
// 2.0 (floating) is converted to 2 (integer)
// explicit
sum = Number(value1) + value2; // both numbers
// result: sum = 14
```

JS vs TS (strong typing)

- Absurd expressions still evaluate in JS (but not TS):

```
// JS: coerces [] to 0 (array length)
console.log(4 / []); // gives Infinity (as 4/0)

// TS: checks types and produces error:
console.log(4 / []);
```

```
The right-hand side of an arithmetic operation must be of
type 'any', 'number', 'bigint' or an enum type.
```

TS = Static Types over JS

- The type system in TypeScript has different levels of strictness when working with a codebase:

1. A type-system based only on inference with JavaScript code
2. Incremental typing in JavaScript via JSDoc
3. Using `// @ts-check` in a JavaScript file
4. TypeScript code
5. TypeScript with strict enabled

TypeScript

- very popular open-source language maintained and developed by Microsoft
- superset of JavaScript
- **static type definitions**
- **kind of arguments** & **returned** values from functions
- exact *structure* of **objects**
- see https://nodejs.dev

Example – type definition

```
type User = {
        name: string;
        age: number;
};
function isAdult(user: User): boolean {
        return user.age >= 18;
}
const justine: User = {
        name: 'Justine‘,
        age: 23
};
const isJustineAnAdult: boolean = isAdult(justine);
```

Example – type definition

```
type User = {
      name: string;
      age: number;
};
function isAdult(user: User): boolean {
      return user.age >= 18;
}
const justine: User = {
      name: 'Justine‘,
      age: 23
};
const isJustineAnAdult: boolean = isAdult(justine);
```