21 October 2020

TypeScript: Coding JavaScript without the Pain

Agus Kurniawan

Microsoft MVP Data Platform

Agus Kurniawan

- Lecturer, Researcher and IT Consultant
- Networking, Security System, Machine learning, and Embedded System
- Sigma Balicamp, Intimedia, Astra International, Hewlett-Packard
- Franhoufer, Samsung R&D Indonesia Institute
- Faculty of Computer Science, Universitas Indonesia
- Freie Universität Berlin, Germany
- Author of several computer books (200+)
- Microsoft Most Valuable Professional (MVP) – 16 years in a row
- https://www.linkedin.com/in/agusk/ and @agusk2010

Outline

- What is TypeScript?
- What is the problem in JavaScript
- Walkthrough: TypeScript
- Q&A

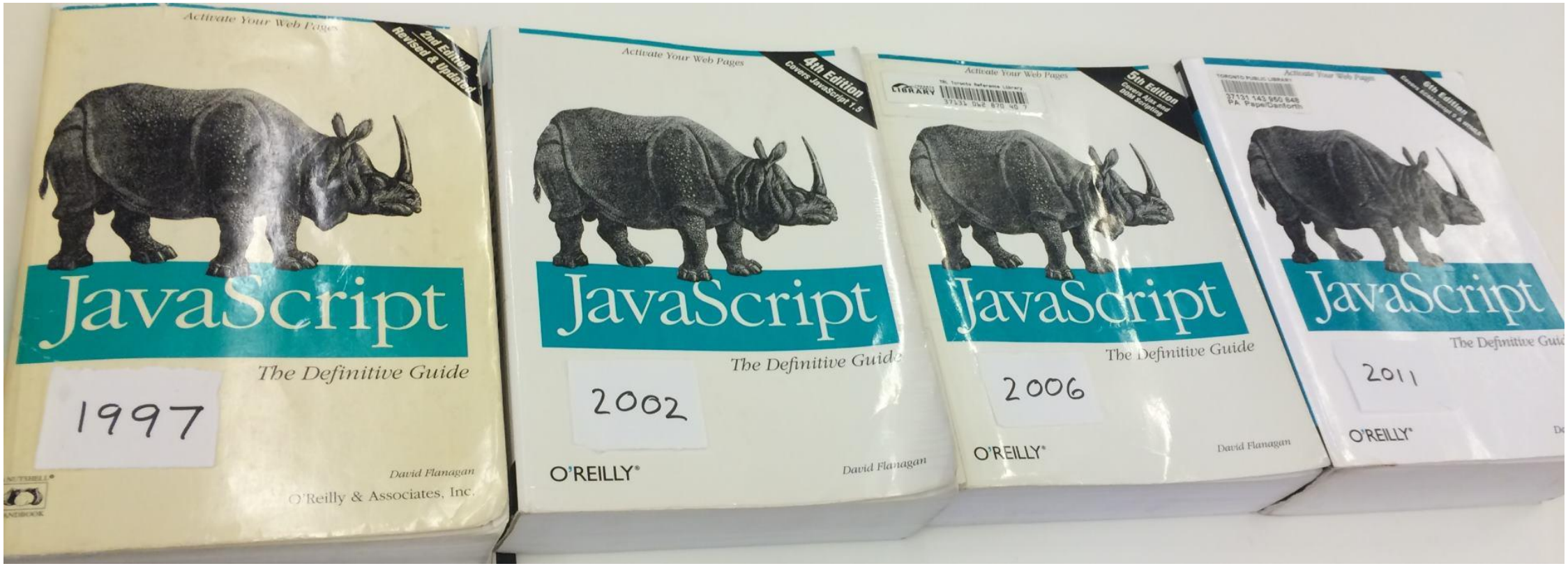
What is TypeScript?

- Free and open source, strongly supported by Microsoft
- Based on ecmascript 4 + ecmascript 6
- Created by the father of C# Anders Hejlsberg
- A superset of JavaScript

- To answer why we need JavaScript+, we need to understand what's wrong with vanilla JavaScript

What is the problem

- Why do people hate working in JavaScript?



Problem - dynamic types

- Variables are untyped and dynamic. They are flexible
- Bad because it is so easy to get wrong
- var x = 1; var y = x + 1;
  // OK, type is inferred. can assume x and y are both numbers.

- var x = 1; x = "hello";
  // NOT OK, type is mixed up. We can't assume what type is x.

- // I am most guilty too - var i, j, k, x, y, z, a, b, c, i1, i2;
  // JS is interpreted. There are no design-time intellisense or compile-time assistance to help you point out errors

Problem - scope

- JavaScript's scope looks like C#, but does not work at a block level. It is at the function level.
- It is so easy to get wrong

```
var i = 1;
if (i == 1) {
    var i = 2;
}
var y = function { var i = 3; }
```

Problem - object inheritance is hard

- Based on object extension. Not class inheritance (at a syntax level)

```
var animal = {
    var name;
};

var cat = jQuery.extend( animal,
    var claw = function() { /*claw*/ };
});
```

- //Syntax complicated, so nobody really does it.

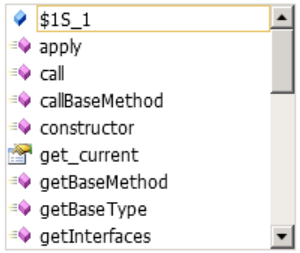
Problem - multiple files

- Last problem for today.
- JavaScript doesn't understand multiple files.
- VS.NET helps with <reference>, but doesn't help you check the correctness of your reference code

```
/// <reference path="C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\MicrosoftAjax.js" />
/// <reference path="C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\SP.debug.js" />
ExecuteOrDelayUntilScriptLoaded(test, "sp.js");

function test() {
    this.ClientContext = SP.ClientContext.get_current();

}
```



Walkthrough: TypeScript

Q&A