Web Frameworks Typescript

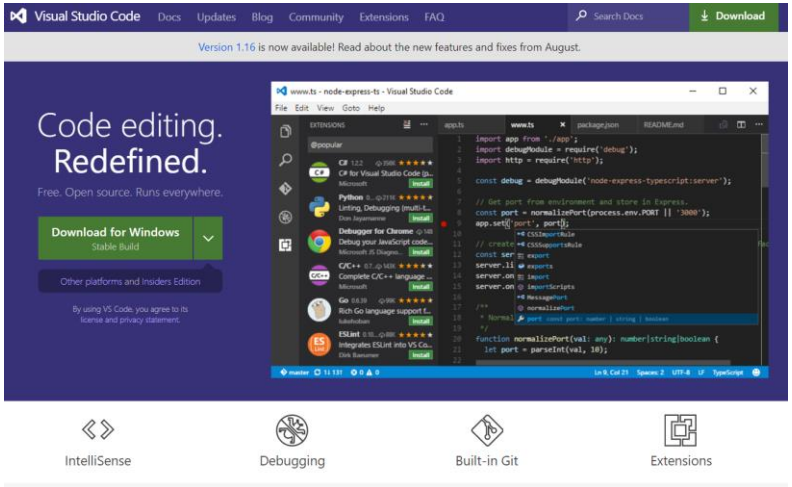
Elektronica – ICT
Sven Mariën
(sven.marien01@ap.be)

AP HOGESCHOOL ANTWERPEN

AP.BE

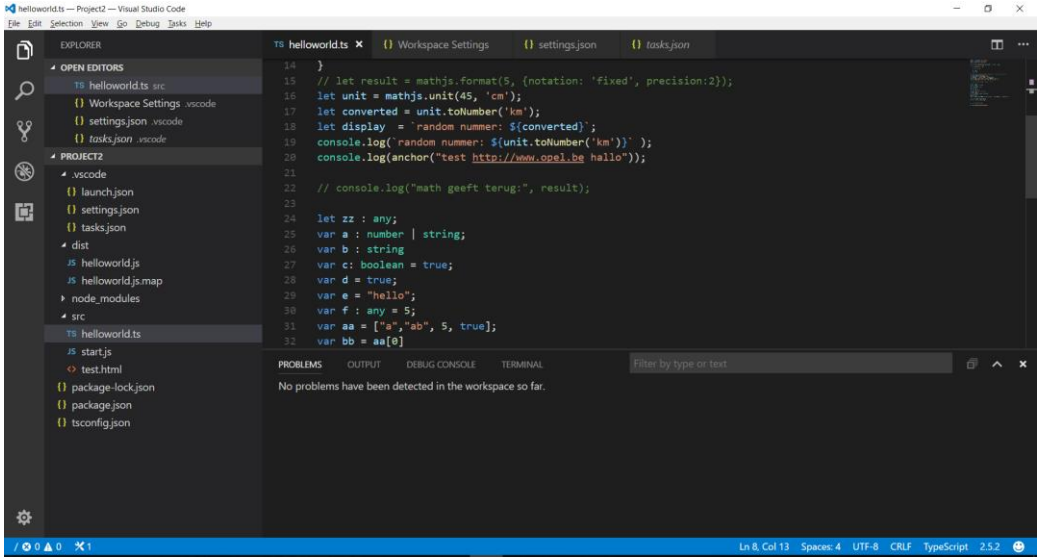
VS Code

- ‘Lightweight’ ontwikkel tool
- Uitgebracht begin 2016 (Microsoft)
- Cross platform (window, mac, linux)
- Heel geschikt voor het ontwikkelen van Web applicaties (ondersteuning voor javascript, typescript, CSS, HTML, JSON…) met “rich intellisense”, “debugging support”,..
- Verschillende “extensions” beschikbaar
- Koppeling met Git (en andere providers)
- https://code.visualstudio.com/



VSCode / gebruik

- Editor split (CTRL + µ)
- Code in commentaar (CTRL-K + CTRL-C)
- Code uit commentaar (CTRL-K + CTRL-U)
- Thema instellen (CTRL-K + CTRL-T)
- User settings en/of Workspace settings(CTRL ,)
- Opzoeken van commands (CTRL-SHIFT-P)
- …



ECMA Script

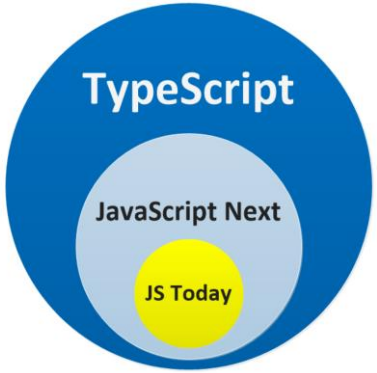
- ECMA Script ?
  - = de specificatie (standaard) voor javascript
  - javascript werd eind 1995 ontwikkeld door Netscape.
  - In 1997 kwam de eerste versie van de ECMA Script standaard.

Compatibiliteit met browsers http://kangax.github.io/compat-table/es6/ (ondertussen niet meer up-to-date)

Versie	Datum	Belangrijkste toevoegingen
ES3	Dec. 1999	Regular expressions, try/catch,..
ES4	Werd niet verder uitgewerkt	
ES5	Dec. 2009	Strict mode, JSON support, …
ES6 / **ES2015**	Juni 2015	Classes/Modules, arrow functions, typed arrays, collections, …
ES7 / **ES2016**	Juni 2016	Array.includes(), ** operator
ES8 / **ES2017**	Juni 2017	Async/await,..
ES9 / **ES2018**	Juni 2018	Async on for loops, new regex features,..
ES2019	Juni 2019	Object.fromEntries() , flatMap(),…
ES2020	Juni 2020	BigInt, Dynamic import, Nullish Coalescing, Optional Chaining,..

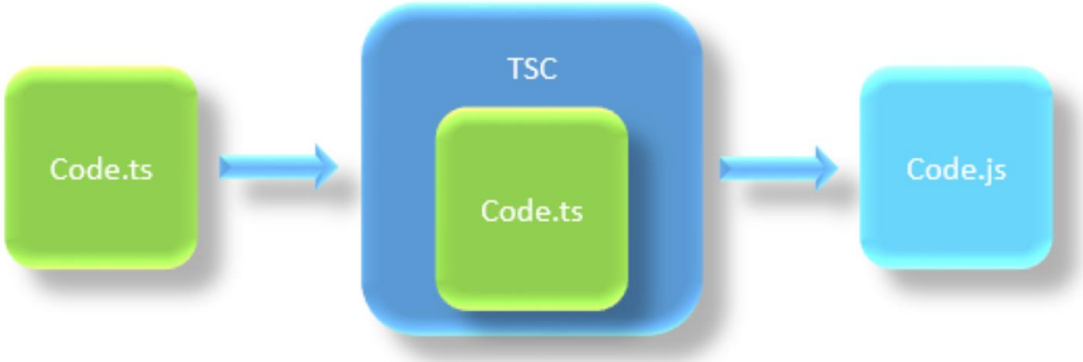
Typescript

- Typescript ?
  - Ook wel eens “javascript of the future” genaamd.
  - Bevat alle “toeters en bellen” uit de laatste ECMAscript versie
  - + ….
  - + voegt “static types” toe aan javascript (vandaar de naam Typescript)

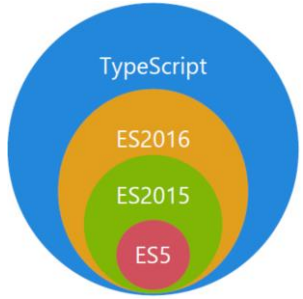


Typescript (2)

- Hoe werkt dit nu juist ?
- We ontwikkelen onze web toepassing in Typescript, zodat we alle “features” kunnen gebruiken van de laatste ECMAScript versie + …
- Maar hoe kan onze code dan werken in de hedendaagse browsers die enkel javascript ES5, .. ES6 ondersteunen (en uiteraard geen typescript ondersteunen) ?
- De oplossing hiervoor is “transpiling” naar javascript (source to source compiler)
- We kunnen aan de “transpiler” meegeven welke ES versie we wensen te bekomen als “output” javascript



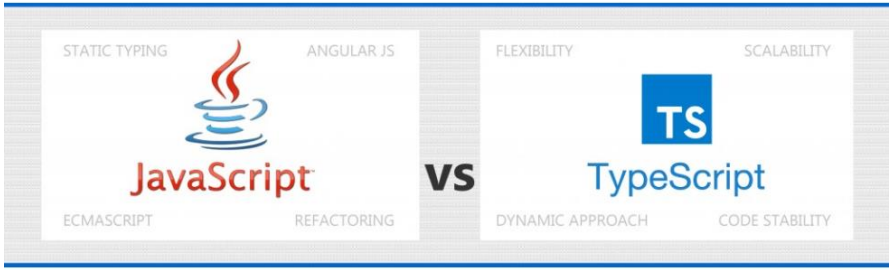
Typescript (Microsoft)



- ≈ Javascript
  - “Superset” van Javascript
  - De allerlaatste ECMAScript versie + enkele extra features
  - De ES2015 versie wordt nog niet (volledig) ondersteund door alle browsers
  - Oplossing: “Transpiling” (=compiling) naar oudere versies van javascript (ES 3, 5,..)
- Enkele voordelen:
  - “Static type support” (via explicit types of type inference)
  - “object oriented” (classes, inheritance, overloading, interfaces,…)
  - Concept van “namespaces” en modules
- Documentatie:

  https://www.typescriptlang.org

  https://basarat.gitbooks.io/typescript/



Installatie

1. VS Code
   - https://code.visualstudio.com/
   - “Out-of-the-box support” voor “typescript language”, maar bevat geen “typescript compiler”
2. Node Package Manager (npm)
   - Tool om javascript componenten te installeren en beheren
   - Wordt meegeleverd met Node.js
   - https://nodejs.org/en/
   - Kies de LTS version => v12.xx
   - Gebruik via “command prompt” (of vanuit VS Code terminal)
3. Typescript compiler
   - Via Node.js “command prompt”
   - **npm install -g typescript**
   - -g => “global installation”
   - tsc –version (versie opvragen)
   - tsc –help (mogelijke opties)

Typescript – type system

- Static type support

Type	*Expliciet*	*Impliciet (Type inference)*
Number	var a : number ;	var b = 5; var c = b + 1;
Boolean	var d : boolean;	var e = false;
String	var f : string;	var g = “GoForIt”; let h = ‘Green’;
Array	var i : number[];	var j = [“A”,”b”,”D”,”rt”]; **var k = [1, True, “OK”];**
Tuple	var l : [number, string, string];	
Enum	enum color {Red = 1,Green, Blue}; var m : color; var n = color.Red;	
Any	var o : any;	var p; let q;

Typescript – Type system (2)

- **null** en **undefined**
  - Niet echt nuttig om als afzonderlijk type te gebruiken, wel in combinatie met een ander type.
  - Gebruik bij voorkeur de compileroption: "strictNullChecks": true (tsconfig.json)
    - Static types zijn bijgevolg niet “nullable”
    - Tenzij expliciet als dusdanig gedefinieerd:

```
let lastName : string | null;
let age : number | null;
```

Typescript - functies

- Een functie kan uiteraard ook static types gebruiken
  - Alle parameters zijn ‘by default’ verplicht
  - ‘optional’ parameter(s):
    - Enkel als laatste parameters
    - Deze krijgen de waarde ‘undefined’
  - ‘default’ waarde:
    - Als deze niet als laatste voorkomen moet de aanroeper de waarde ‘undefined’ meegeven om de ‘default’ waarde te gebruiken
  - Rest Parameters:
- ‘Named’ of ‘anonymous’ functies
- Lambda (fat arrow) functies

```
function CalculateLength(text : string) : number {
    return text.length;
}

let Calc = function (text : string) : number {
    return text.length;
}
```

```
function ReverseText(text: string, cuttOffLength? : number) : string {···
}

function SplitText(text : string, separator: string = "-") : string[] {···
}
```

```
function buildName(firstName: string, ...restOfName: string[]) {
    return firstName + " " + restOfName.join(" ");
}
```

Typescript - functies

- lambda expressions / fat-arrow functions
  - = Verkorte notatie van anonymous functions

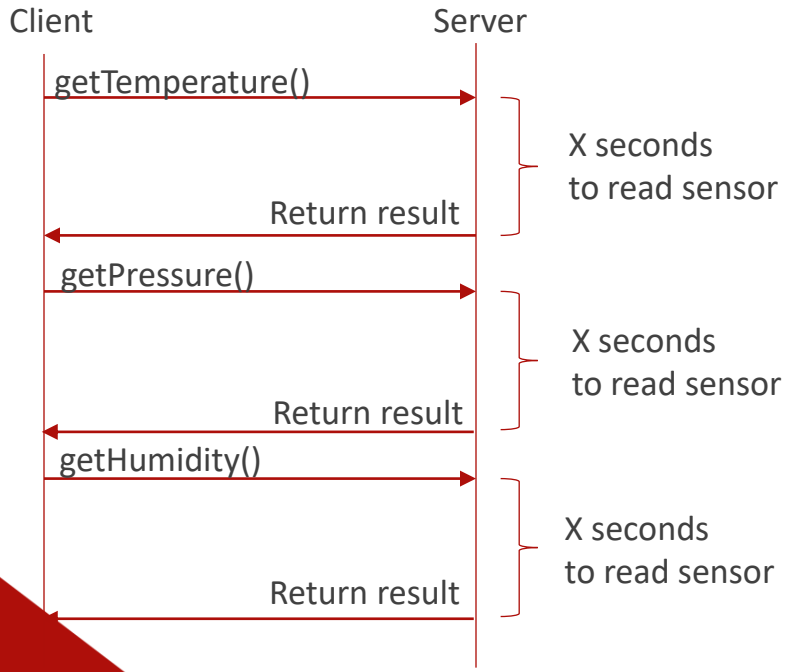
```
function Sum(a: number, b: number) {
    return a + b
}
var sum = function (a: number, b: number) {
    return a + b
}
var sum = (a:number, b:number) => {
    return a+b;
}

var sum = (a: number, b:number) => a+b;
```

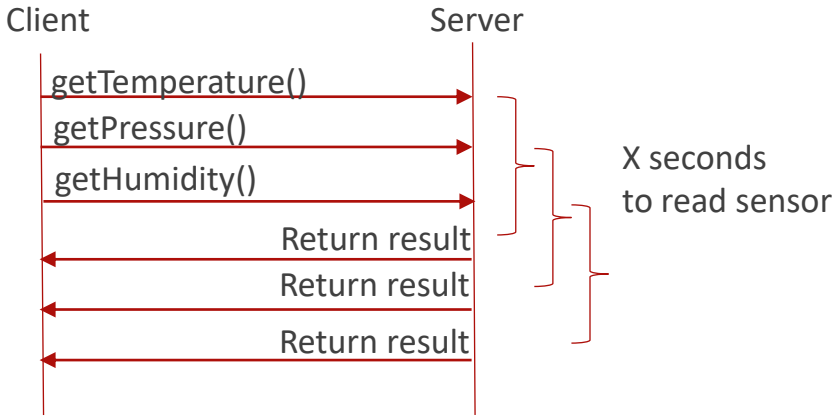
“Callback” functies

- Wat zijn callback functies ?
  - Een **functie** die als **parameter** wordt meegegeven aan een **andere functie**….
- Waar wordt dit gebruikt ?
  - Als we vanuit functie x en andere functie y wensen te kunnen aanroepen
    - Bv. Array.sort, Array.forEach,…
  - Als we in een functie het resultaat niet onmiddellijk kunnen teruggeven
    - Bv. Asynchrone operatie (AJAX calls,..)

```
list.forEach((val, idx)=> console.log(val + " " + idx));
```

Synchroon versus Asynchroon





Synchroon: totale tijd = X * 3

Asynchroon: totale tijd = X

Typescript – Object oriented programming

- Gebruik van Classes
  - Constructor, properties, methods (**public** 'by default', private and protected)
  - '**this**' keyword om eigen 'members' aan te roepen binnen de klasse
  - **Readonly** properties
  - **Parameter** properties !
  - **Get/set** properties
  - **Static** properties & methods
  - **Abstract** members en classes
- Gebruik van **Inheritance** (subclassing)
  - Mbv. '**extends**' keyword
  - '**super**' keyword om de 'base' class aan te roepen.
  - '**overriding**' behoeft geen keyword
- Gebruik van **interfaces**
  - Implementeren mbv. '**implements**' keyword

```
class Maths
{
    private _getal1 : number;
    private _getal2 : number;

    constructor(getal1 : number, getal2: number){
        this._getal1 = getal1;
        this._getal2 = getal2;
    }

    Add() : number {
        return this._getal1 + this._getal2;
    }

    Subtract(): number { return this._getal1 - this._getal2 };
```

Typescript - Modules aanmaken

- Laat toe om je applicatie **op te delen**
- Enerzijds opdelen in afzonderlijke bestanden en anderzijds ook opdelen in “namespaces”
- Een module is gewoon een apart (.ts) bestand waarin je klassen, functies kan onderbrengen.
- Elke module heeft zijn eigen ‘scope’
- Via het ‘export’ statement geef je aan wat je wil publiek maken vanuit een bepaalde eigen geschreven module:

```
export class Maths
{
    private _getal1 : number;
    private _getal2 : number;

    constructor(getal1 : number, ge
        this._getal1 = getal1;
```

```
export { Maths, Helpers as MathHelpers };

class Maths {···
}

class Helpers {
}
```

- Er bestaan ook reeds vele 3rd party (javascript) modules die je kan downloaden en gaan gebruiken.

```
export function CalculateChecksum(accountNr : string): string
{
```

Typescript - Modules gebruiken

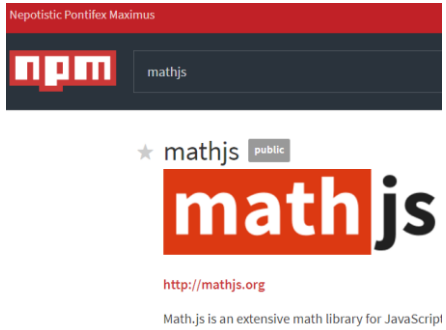
- Via het ‘import’ statement bestaande modules linken
  - Merk op: **geen extensie** (.ts, .js) vermelden
  - Laat de **naam voorafgaan door** “**./**” indien de module zich in dezelfde map bevindt.
- Ofwel alles importeren (import * as …)
  - Verplicht lokale ‘namespace’ opgeven
- Ofwel enkel bepaalde items (klasse, functies, ..) importeren
  - ‘Alias’ (as …) is niet verplicht

```
import * as MathLib from "./Maths"
let z = new MathLib.Maths(10,35);
```

```
import {Maths as MathsLib}  from  "./Maths"
let q = new MathsLib(10,35);
```

Typescript – 3rd party modules

- Via Node Package Manager (npm)
  - Zoek een package op via: https://www.npmjs.com
  - Vervolgens package installeren via VSCode
    - Naam/Versie toevoegen als “dependencies” in package.JSON
      - VSCode stelt automatisch de recentste versie voor.
    - Via ‘terminal’ window: “**npm install**” uitvoeren




```
"dependencies": {
    "mathjs": "3.16.3"

}
```

```
PS D:\AP\Typescript\projects\test1> npm install
npm WARN test1 No description
npm WARN test1 No repository field.
npm WARN test1 No license field.

added 8 packages in 6.297s
```

  - Installeren van npm packages kan desgewenst ook via de command line
    - bv. “npm install mathjs – save”

Typescript – 3rd party modules

- Om “intellisense support” te bekomen bij die javascript modules:
  - Ondersteunt typescript de zogenaamde “**Type declaration**” bestanden
  - Deze hebben een extensie: **“.d.ts”**
  - Voor vele 3rd party modules bestaan reeds .d.ts bestanden -> installeren via npm
  - Je kan desgewenst eerst zoeken via : http://microsoft.github.io/TypeSearch/
  - De naam is per afspraak hetzelfde als de module, voorafgegaan door **“@types/”**

TypeSearch


```
"dependencies": {
    "mathjs": "3.16.3",
    "@types/mathjs": "0.0.35"
}
```

```
PS D:\AP\Typescript\projects\test1> npm install
npm WARN test1 No description
npm WARN test1 No repository field.
npm WARN test1 No license field.

added 1 package in 1.83s
PS D:\AP\Typescript\projects\test1>
```

Typescript – 3rd party modules gebruiken

- Locatie van de modules:
  - Npm modules worden geïnstalleerd onder de map '**node_modules**
  - Npm type declaration modules worden geïnstalleerd onder de submap **'@types**'
- Gebruik van de npm modules:
  - Via het **import** statement (zoals bij eigen gemaakte modules)
  - Locatie echter **niet voorafgegaan** door "./"

```
import * as MathJS from "mathjs"
```

  - **Sommige modules** moeten anders worden geïmporteerd:
    - Als de export in de modules is gedefinieerd als **export =**
    - Gebruik dan het "require" statement (bovenstaande manier zal immers niet lukken)

```
import Pad = require("pad")
```

Typescript - decorators

- Is wat te vergelijken met “class attributes” in c#
- Kan worden gebruikt om extra ‘metadata’ te leveren of de functionaliteit aan te passen van een:
  - Klasse, Methode, Property of Parameter
- Is in feite een functie die vooraf wordt aangeroepen
- Worden in typescript voorafgegaan door een @
- Decorators zitten nog in een “proposal” fase om te kunnen worden toegevoegd aan de ECMAScript standaard, bijgevolg moet deze compiler setting worden toegevoegd:
  - **experimentalDecorators**: **true**

tsconfig.json:

```
{
    "compilerOptions": {
        "target": "ES5",
        "experimentalDecorators": true
    }
}
```

```
public class Class1
{
        [Obsolete]
        public void Method1()
        {
        }
        public void NewMethod1()
        {
        }
}
```

```
static void Main(string[] args)
{
    Class1 o = new Class1();

    o.Method1();
}
```

[deprecated] void Class1.Method1()

Warning:
'ConsoleApplication48.Class1.Method1()' is obsolete

Typescript - decorators

- Kan gebruikt worden om functionaliteit toe te voegen:
  - bv. Loggen wanneer een class, member,.. Wordt aangeroepen.
- Kan gebruikt worden om metadata toe te voegen:

```
@logClass
class ExampleClass {
  constructor(name : string) {
    console.log(`Yo ${name}`);
  }

  get(a, b, c) {
    return 3;
  }
}
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'appy2';
}
```

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

AP HOGESCHOOL ANTWERPEN

AP.BE