TypeScript Fundamentals

Bogdan Manate

Agenda

- Introduction
- Motivation
- Installation
- Features
- Configuration file
- Type definition files
- Bundlers
- Write your own application
- Conclusion

Introduction

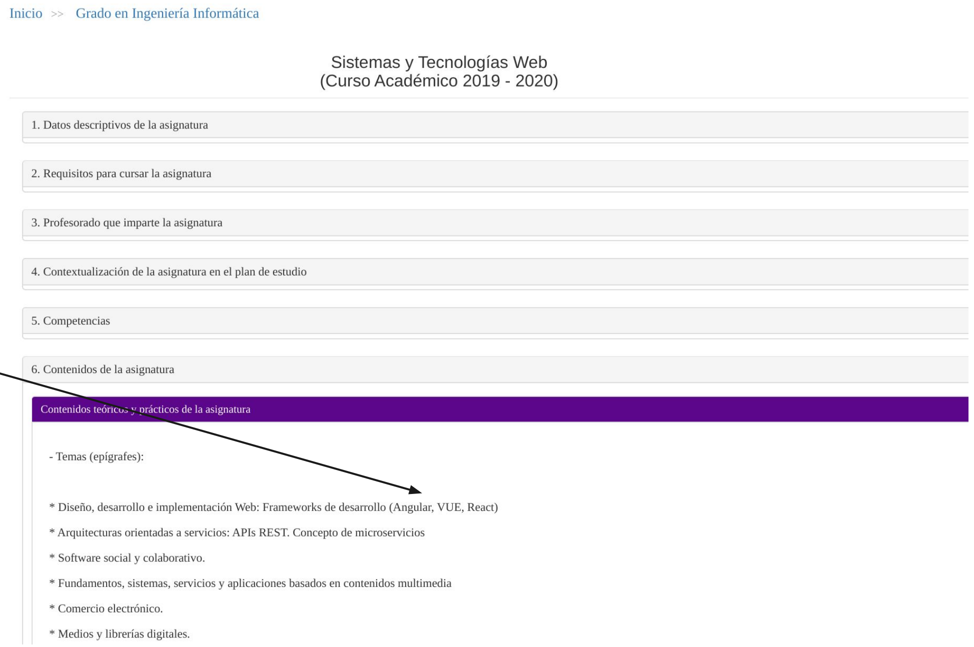
- TypeScript (TS) is a typed superset of JavaScript (JS) that compiles to plain JavaScript.
- Syntax based on ECMAScript 4 & ECMASCript 6 proposals.
- TS is first and foremost a superset of JS -> Any regular Javascript is valid TypeScript Code.
- ...and is developed by Microsoft.

Motivation

- Compile time error reporting
- Strong typing
- Type definitions (.d.ts files that provide easy integration for js projects)
- Encapsulation provided by classes
- Private, protected and public accessors

Motivation

Angular, Vue or React



Inicio >> Grado en Ingeniería Informática

Sistemas y Tecnologías Web
(Curso Académico 2019 - 2020)

1. Datos descriptivos de la asignatura

2. Requisitos para cursar la asignatura

3. Profesorado que imparte la asignatura

4. Contextualización de la asignatura en el plan de estudio

5. Competencias

6. Contenidos de la asignatura

Contenidos teóricos y prácticos de la asignatura

- Temas (epígrafes):

* Diseño, desarrollo e implementación Web: Frameworks de desarrollo (Angular, VUE, React)

* Arquitecturas orientadas a servicios: APIs REST. Concepto de microservicios

* Software social y colaborativo.

* Fundamentos, sistemas, servicios y aplicaciones basados en contenidos multimedia

* Comercio electrónico.

* Medios y librerías digitales.

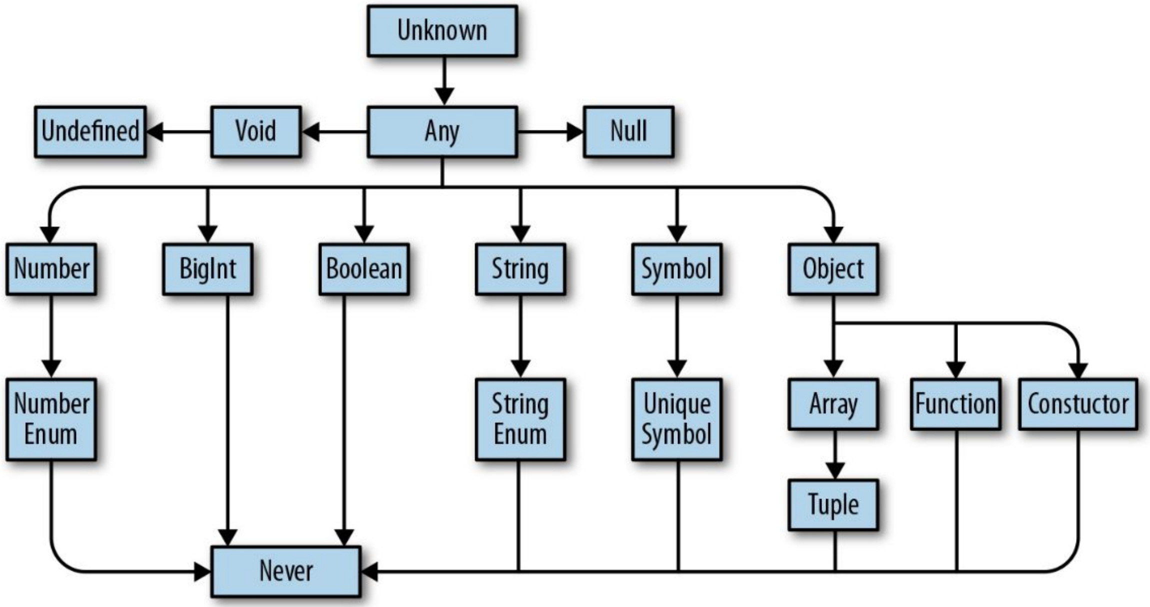
Installation

- Run: **npm install -g typescript**
- Make sure the installation was successful: **tsc -version**

Features

- Supported data types
- Functions
- Classes
- Interface
- Namespace
- Modules
- Decorators

Features: Supported data types



*O’Reilly **Programming with Typescript** - Boris Cherny

Features: Functions

- Functions are the fundamental building block of any application in JS.
- In JS you build up layers of abstraction, mimicking classes, information hiding, and modules.
- In TS, while there are classes, namespaces, and modules, functions still play the key role in describing how to do things.

Features: Interfaces

- Declared using **interface** keyword.
- There is no corresponding type in JS, therefore no JS code will result from compilation.
- Errors being shown when interface signature and implementation doesn’t match.

Features: Classes

- Can implement interfaces
- Inheritance
- Instance methods/members
- Static methods/members
- Single constructor
- ES6 class syntax

Features: Namespaces vs. Modules

Namespaces

- Namespaces are a TypeScript-specific way to organize code.
- Namespaces are simply named JavaScript objects in the global namespace.
- Namespaces that can span multiple files.
- Use them when you don't want to use a module loader.

Modules

- Modules can contain both code and declarations.
- Modules also have a dependency on a module loader (CommonJS/Require)
- Each file that has a import or export statement is considered a module.
- The most common way to organise code.

Features: Decorators

- Decorators are available as an experimental feature of TypeScript.
- The **experimentalDecorators** compiler option should be enabled.
- Can be attached to a class declaration, method, accessor, property, or parameter.
- Decorators use the form @expression.
- **expression** must evaluate to a function that will be called **at runtime** with information about the decorated declaration.

Configuration file

- The presence of a **tsconfig.json** file in a directory indicates that the directory is the root of a TypeScript project.
- The **tsconfig.json** file specifies the root files and the compiler options required to compile the project.
- A project can be compiled:
  - By invoking tsc with no input files - the compiler searches for the tsconfig.json (current dir and parent chain).
  - By running tsc with no input files and a --project (or just -p) and a path to a config file.
- A tsconfig.json file can be generated using **tsc --init**

https://www.typescriptlang.org/docs/handbook/tsconfig-json.html

Type definition files

- What about using plain JS libraries inside TS projects?
- The type definition files (*. d.TS) help you to overcome this problem.
- You don't have to define them yourself -> https://microsoft.github.io/TypeSearch/
- Installation: npm install --save-dev @types/lodash
- Usage: import * as _ from "lodash";

Bundlers

- Node.js
  - Configure TS to output commonjs modules.
- Webbrowser
  - Requirejs (old) - module: amd
  - Systemjs (old) - module: systemjs
  - Webpack -module: es2015 or higher
  - Rollup -module: es2015 or higher
  - Browserify -module: commonjs
  - Parcel: -module: esnext

Write you own application

- Conway's Game of Life
- https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- https://github.com/bogdanmanate/mvp-ts

Conclusion

- TS is not a completely new language, so you still have to know the JS quirks.
- Optional static typing (the key here is optional)
- Type Inference, which gives some of the benefits of types, without actually using them explicitly
- Access to ES6 and ES7 features, before they become supported by major browsers
- The ability to compile down to a version of JavaScript that runs on all browsers
- Great tooling support with IntelliSense