Notatki Angular

# Poniedzialek (9.03.2020)

## **Czesc 1**

## Wstęp do Visual Studio CODE IDE & Wtyczek

1. Instalacja **wtyczek**
2. Weryfikacja dzialania **Quokki**

* File 🡪 Preferences 🡪 Tree Indent
* Otwieramy js/ts file np. **Sample-module.js** 🡪 F1 🡪 Run quokka on save of current file (F1)
* Ctrl + ~
* PMM na skrypcie 🡪 Open in terminal
* (gdyby nie quokka, musielibyśmy recznie uruchamiac skrypty jsowe)
* Refresh explorer (na folderze glownym)

## Wstęp dot. Node.js

Dzięki temu zaczyna działać jako mocno typowane + uzyskać modularyzacje:

* **Require**(..) – będę korzystał z zewnętrznych zasobów
* **Module**.**exports**() – co eksportuje na zewnątrz dany moduł (np. obiekt)
* **Requirowanie** **pięciokrotne**: wczytuje jeden raz i trzyma w pamięci
* **Node.js** - Program wykonywany w c++
* **Od 2015** – javascript coraz bliżej C# , async, class etc….etc…
* **Let** vs **var** *(dostępny po za petla)* **–** inny scope
* **Prototypowalność vs class** – (nie potrzeba klasy żeby korzystać z obiektu)
* **ES6 –** standard zapisu
* **Vanilla Javascript –** czyściutki javascript, bez bibliotek
* **Array.flat() –** spłaszcza wiele wymiarowe tablicy do listy
* **Transpiler -** (from ts to js)
* **Transpilacja –** dalej mamy poziom abstrakcji, transpiluje się do kodu w innej wersji, który może być dalej modyfikowany (w przeciwieństwie do bajtow, gdy mamy plik wykonywalny)

devDependencies: typescript (mówimy o tym ze node musi wykonać dodatkowy krok, transpilacje)

* **Prototypy** – mam instancje obiektu, probujemy wywolac metode, jeżeli js nie ma jej na instancji to…js biegnie do tego co dziedziczy. Prototyp mieszka w 1 miejscu pamięci 🡪 jeśli jest to do słowa kluczowego this dorzuca ta instancje
* **Polyfill –** dodatkowy kod, który może wykonywać się u klienta gdy chcemy mieć suport dla starszych przeglądarek (dodatkowe funkcje dla starszych wersji)
* **NPM –** tj. nuget dla C#, maven dla javy, npm dopisuje zaleznosci samemu do package.json (nie pobieramy nic już z lapy)
* **Npm install jquery -** (w folderze package.json, resolvuje dependencies do **node**\_**modules** I skrypty)
* **Npm run build** (dist folder z transpilowanym typescriptem, bo jednym z krokow jest tsc)
* **Npm run -** wylistuje skrypty
* **Tsc -** skrypt do transpilacji (skad wie? Tsconfig.json definiuje co budowac) zainstalowany przez nmp’a ( **tsc –watch**)
* **Watchmode** dla node’a zeby mieć dostep do document(?)
* **Node\_modules**  - nie chcemy tego mieć w kontroli wersji, potrzebny jest jedynie package.json, folder src, pliki konfiguracyjne sa na szczycie
* **Dev\_dependencies** - typescript w procesie developmentu
* **Dependencies** - Jquery, rxjs

Podsumowanie: struktura nodem\_modules, skad co pochodzi, quokka

## **Czesc 2**

* **Webpack , the Bundler** (biblioteka, tool) – from development (codes) to production, zadaniem jest zebranie rzeczy z node’a do wersji zmimifikowanej i zbudowanie webpacka na proda
* **Bundler** również dodaje **checksum** do nazwy pliku. Nowa wersja ma nowa checksume i zostanie latwo zaktualizowana.
* <https://webpack.js.org/>
* npm install -g @angular/cli
* ng –version

## Angular CLI : Command Line Interface