Technologie internetowe
Angular

Paweł Rajba
pawel@cs.uni.wroc.pl
http://pawel.ii.uni.wroc.pl/

Plan wykładu

- Wprowadzenie
- Get Started
- Architektura aplikacji
- Moduły
- Kompontenty
- Szablony
- Data binding
- Usługi

Wprowadzenie

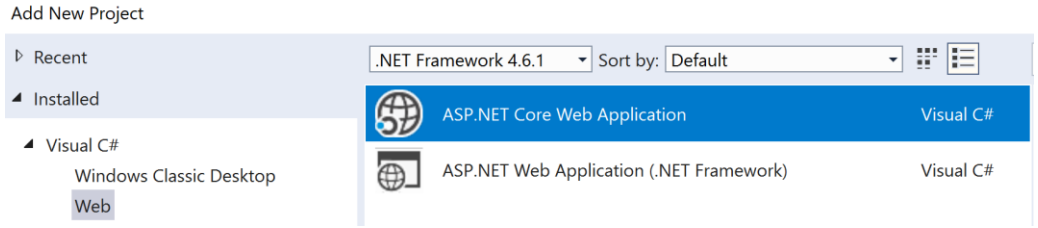
- Biblioteka, framework od Google
- Umożliwia tworzenie aplikacji SPA
- Tworzone na zasadach OpenSource
- Najnowsza wersja stabilna: 5.0.5
- Strona główna: https://angular.io/
- Wersja początkowa 1.6.x
  - https://angularjs.org/
- Wsparcie wielu narzędzi
  - Dalej będziemy korzystać z Visual Studio

Get started

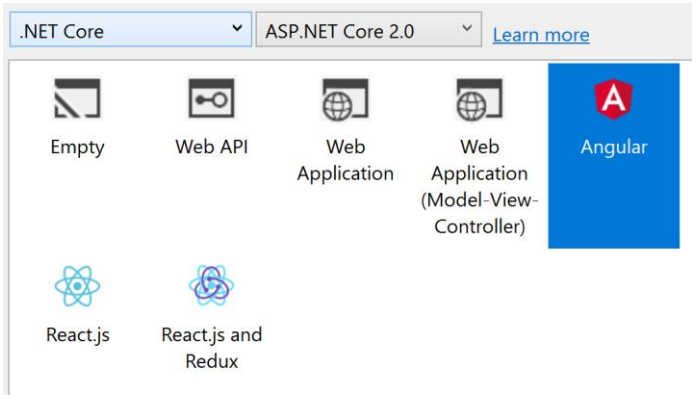
- Jako punkt wyjścia możemy skorzystać ze struktury (seed):
  - Dostępnej w Visual Studio 2017
  - Wbudowanej w WebStorm
  - Publikowanej przez różnych dostawców
- QuickStart
  - https://angular.io/guide/quickstart
- Podstawowy przykład
  - https://angular.io/generated/zips/cli-quickstart/cli-quickstart.zip

Get started: Visual Studio 2017

- Upewniamy się, że mamy dostępny node.js
  - Jeśli nie → https://nodejs.org/en/download/
- Tworzymy nowy projekt i wybieramy:



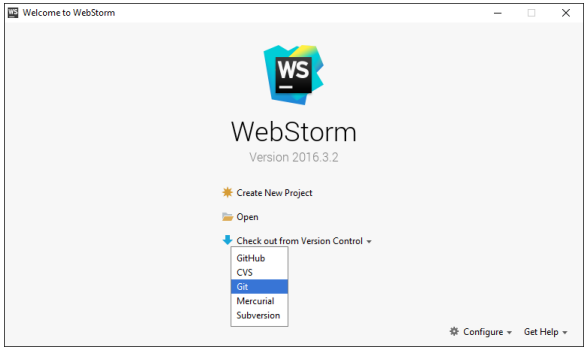
- A następnie:



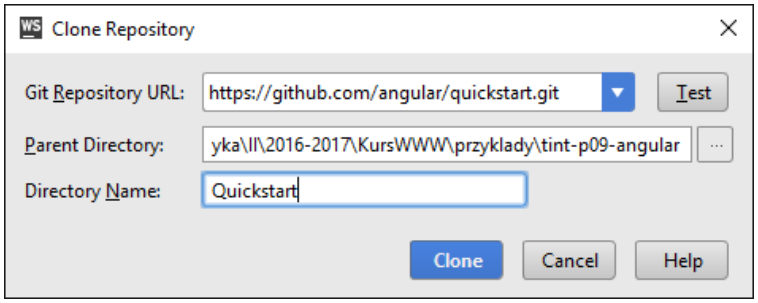
Get started: WebStorm

(może być nieaktualne)

- Uruchiamy WebStorm
- Wybieramy opcję na poniżej



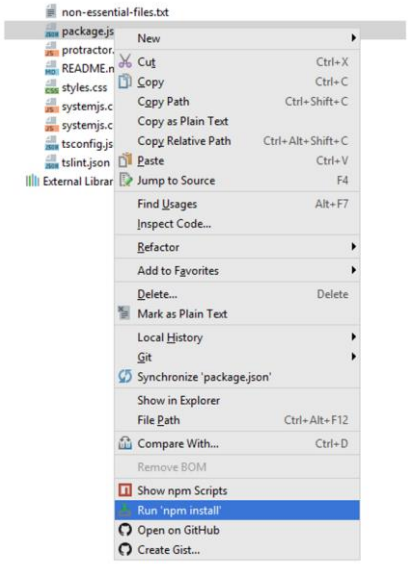
- Do formularza wprowadzamy:



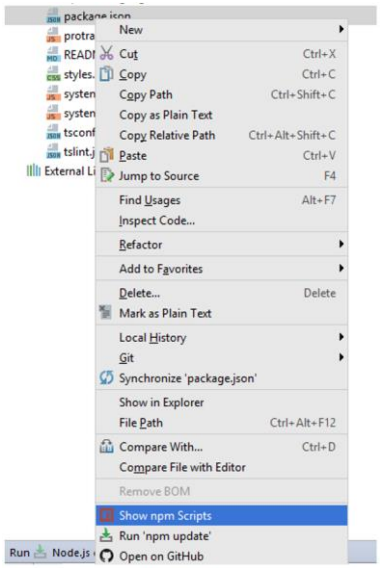
Get started: WebStorm

(może być nieaktualne)

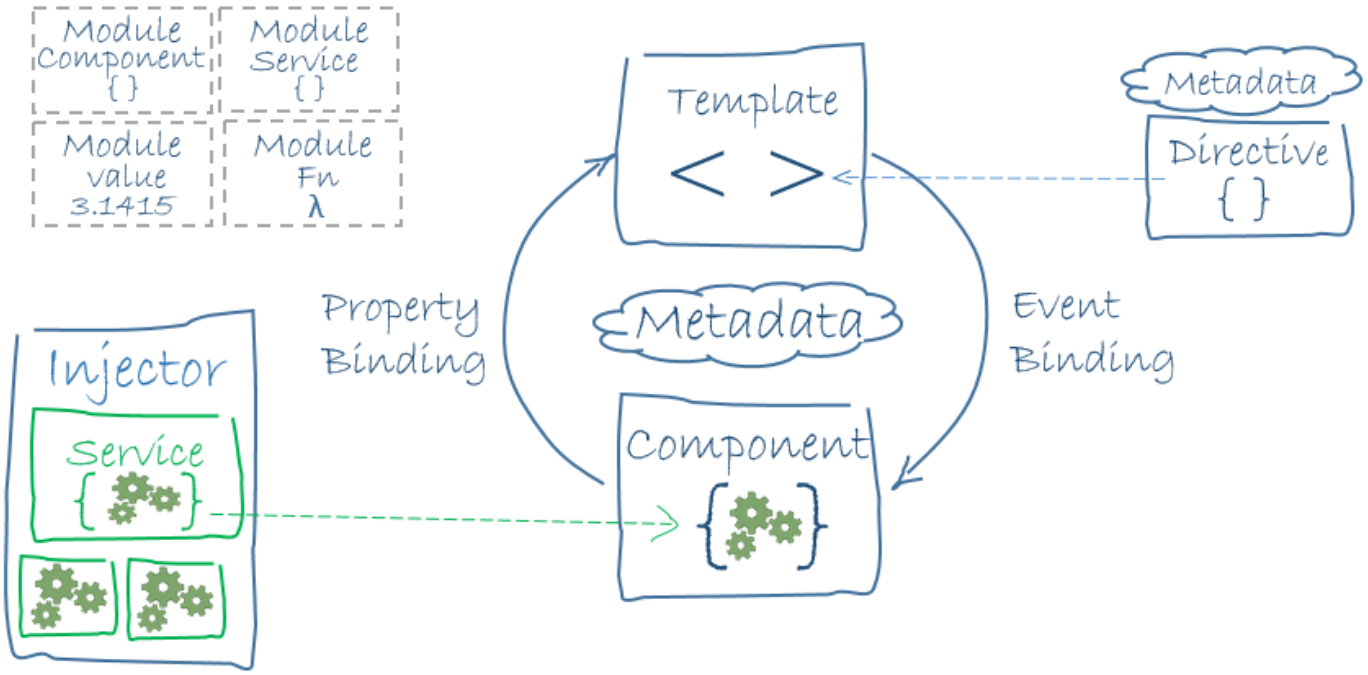
- Wybieramy Run „npm install” na packages.json



- Wybieramy opcję Show npm Scripts, a następnie klikamy skrypt „start”



Architektura aplikacji



Przygotowane z pomocą: https://angular.io/docs/ts/latest/quickstart.html

Moduły

- Główne składowe aplikacji
- Co najmniej jeden główny moduł (root module), zwykle zwany AppModule
- W większych aplikacjach jest więcej modułów odpowiadających np. wybranym funkcjonalnościom czy procesom
- Moduł to klasa definiowany przez dekorator @NgModule

Komponenty

- Odpowiada za zachowanie jakiegoś fragmentu aplikacji
  - Sterują zachowaniem widoków
- Składa się z dwóch części
  - Metadanych, które „mówią” Angularowi o tym komponencie
  - Klasie, które definiuje zachowania komponentu
- Metadane definiowane są przez dekorator @Component wraz parametrami:
  - selektor – wskazuje obszar w HTML, w ramach którego komponent zostanie wyrenderowany
    - analogia do selektorów CSS
  - template – tekst szablonu
  - templateURL – jeśli template mamy w osobnym pliku
  - providers – tablica usług, które dostarczają funkcjonalności do komponentu
    - dostępne przez dependency injection

Szablony

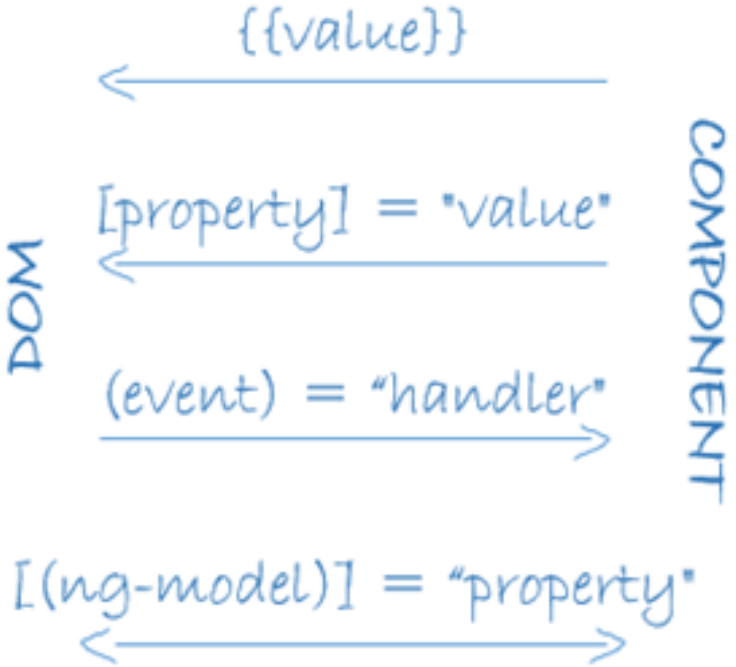
- Składają się z HTML-a z dodatkowymi elementami do Angulara
- Pełna składnia języka szablonów dostępna:
  - https://angular.io/guide/quickstart
- Definiują, jak komponenty będą wyglądać
- Mogą zawierać komponenty (szablony) zagnieżdżone

Data binding

- Zarządzanie wymianą danych pomiędzy komponentem i widokiem wymagałoby sporo uwagi i wprowadzało spore ryzyko błędów
  - Dlatego dostępny jest mechanizm data binding
- Polega on na powiązaniu elementów komponentu i widoku w taki sposób, że następujące automatyczna aktualizacja danych.
- Data binding może być
  - Jednokierunkowy (w obu kierunkach)
  - Dwukierunkowy

Data binding

- Definicja może być zrealizowana na 4 sposoby:



Usługi

- Dodatkowe funkcjonalności dostępne dla aplikacji
- Mogą pełnić bardzo różne role od technicznej po logikę biznesową
- Przykłady:
  - Usługi danych (np. pobranie z RESTful service lub local storage)
  - Integracja z innymi aplikacjami
  - Wyliczenia dla aplikacji (kalkulator roczny pensji brutto do netto)
- Udostępniane za pomocą mechanizmu depedency injection i przekazywane do np. konstruktorów komponentów (analogicznie jak w ASP.NET Core)

Materiały dodatkowe

- Quick Start
  - https://angular.io/guide/quickstart
- Tutorial
  - https://angular.io/tutorial
- Inny ciekawy tutorial dla wersji 2
  - http://learnangular2.com/
- Getting started nie od Google dla wersji 2
  - http://onehungrymind.com/build-a-simple-website-with-angular-2/
- Konfiguracja Angular2 w WebStorm
  - https://egghead.io/lessons/angular-2-webstorm-setting-up-angular-2