Hello AngularJS!

Aplikacja „od zera” w jedno popołudnie.

Marcin Piekarski

COMARCH

Czym jest AngularJS?

- otwarta biblioteka języka JavaScript…

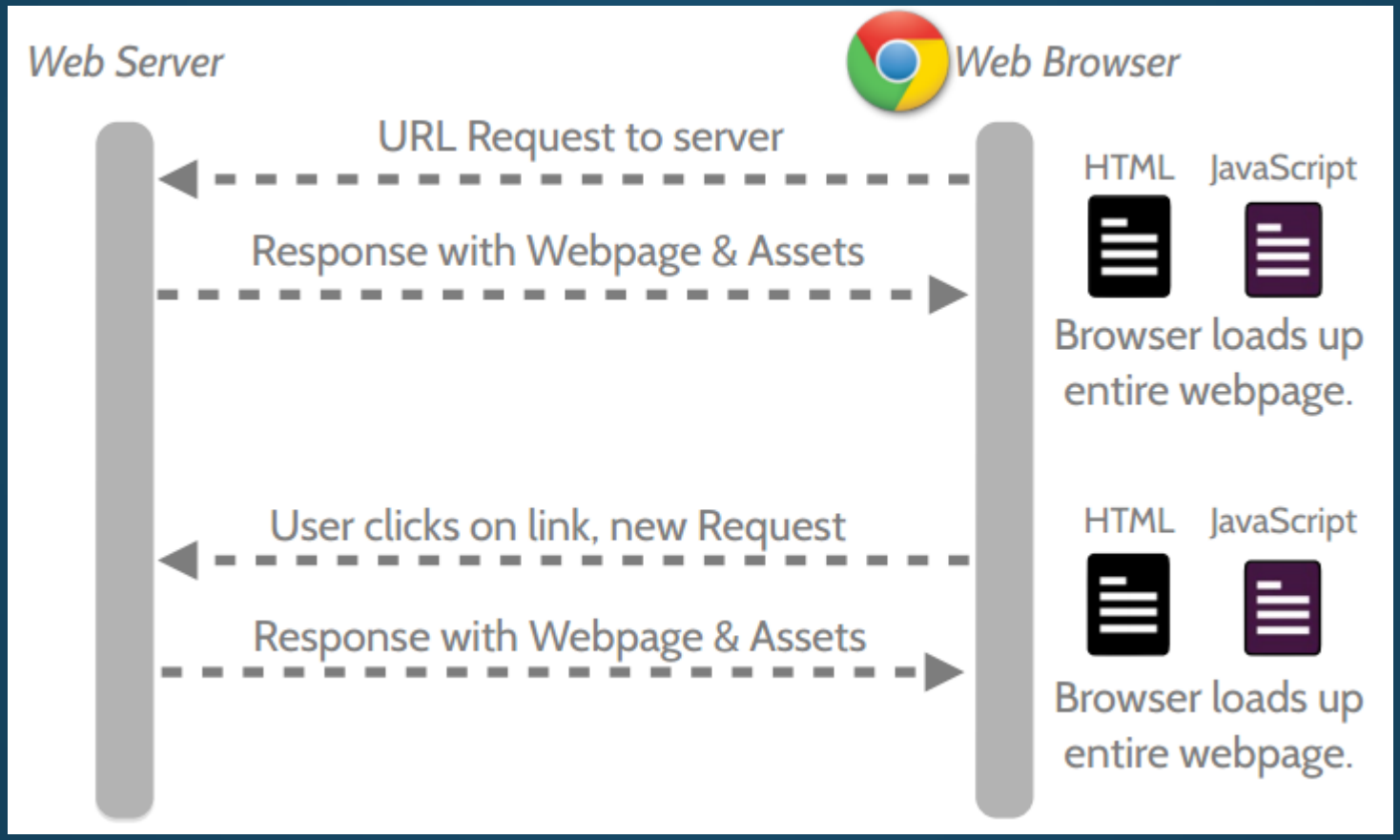
Dlaczego AngularJS?

- HTML jest świetny…dla statycznych widoków, ale…
- …dla dynamicznych treści nie nadaje się

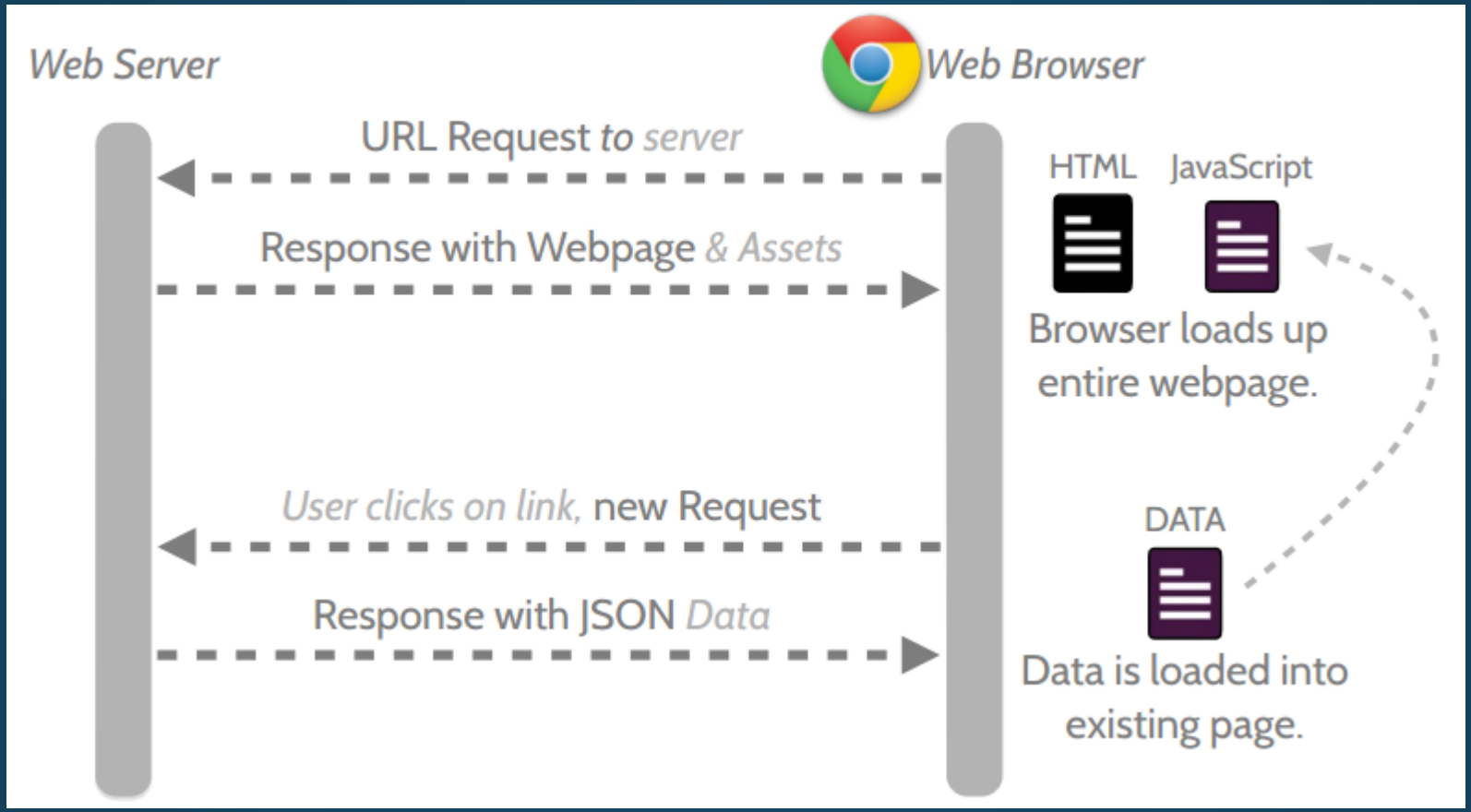
- rozszerzenie składni HTML – elementy dynamiki i interaktywności - dyrektywy
- data binding
- manipulacje w DOM? Tylko na własne życzenie ☺

- oddzielenie warstwy klienckiej aplikacji od warstwy serwerowej
- wzorzec MVC / MVVM / MVW
- dependency injection

Klasyczny refresh strony



Dynamiczne ładowanie zawartości



Podstawowe elementy aplikacji

- Moduł aplikacji
- Kontrolery
- Dyrektywy
- Filtry
- Serwisy: service, factory, provider
- Configi i run block’i

Dyrektywy

- tagi oraz atrybuty
- komponenty reużywalne
- komponenty funkcyjne lub strukturalne
- własne dyrektywy

ng-app – attach the Application Module to the page

```
<html ng-app="store">
```

ng-controller – attach a Controller function to the page

```
<body ng-controller="StoreController as store">
```

ng-show / ng-hide – display a section based on an Expression

```
<h1 ng-show="name"> Hello, {{name}}! </h1>
```

ng-repeat – repeat a section for each item in an Array

```
<li ng-repeat="product in store.products"> {{product.name}} </li>
```

Moduły

- Kawałki naszej aplikacji
- Definiowanie zależności (Dependency Injection)

```
var app = angular.module('store', [ ]);
```

Kontrolery

- model danych + logika

```
var app = angular.module('store', [ ]);

app.controller('StoreController', function(){

});
```

Serwisy

- provider
- service
- factory
- value/constant

- komunikacja z backend’em (dobra praktyka!)
- komunikacja między kontrolerami
- współdzielenie funkcjonalności

Filtry

- filtrują/formatują content
- wbudowane filtry
- własne filtry

```
{{ data* | filter:options* }}
```

```
{{'1388123412323' | date:'MM/dd/yyyy @ h:mma'}}
```

12/27/2013 @ 12:50AM

```
{{'octagon gem' | uppercase}}
```

OCTAGON GEM

```
{{'My Description' | limitTo:8}}
```

My Descr

```
<li ng-repeat="product in store.products | limitTo:3">
```

Walidacje

- gotowce – required, minlength, maxlenght, pattern, itp.
- customowe (własne typy)
- proste mechanizmy informujące, co jest nie tak:
  - Klasy CSS (ng-valid, ng-invalid, ng-pristine, ng-dirty, ng-touched, itp.)
  - Identyfikatory w obiektach $error

Dependency Injection

- nie muszę wiedzieć, gdzie żyje dana zależność
- nie muszę dbać o jej zainicjalizowanie
- zamiast tego mówię: daj mi „to”…a Angular sam załatwia za mnie brudną robotę i dba jeszcze o czas życia danej zależności!

```
app.controller('SomeController', [ '$http', '$log', function($http, $log){

} ]);
```

Czas na odrobinę praktyki ☺

NodeJS + npm

- platforma/środowisko uruchomieniowe dla JavaScript'u…
- …poza przeglądarką ☺
- moduły (np. http, file system, buffer, itp.)
- npm – menadżer pakietów (globalna biblioteka)

Bower

- menadżer pakietów klienckich, czyli…
- …wszystko to, z czego korzystamy w naszej aplikacji, możemy ściągnąć za pomocą bower'a

Gulp

- system automatyzacji pracy
- mnóstwo wtyczek npm ułatwiających życie programiście ☺

Krok 1 - zadanie

- stworzyć katalog „main”
- dodać w nim plik skryptowy o nazwie ToDoController.js
- przenieść do tego pliku zawartość kontrolera jako funkcję nazwaną: ToDoController
- zaimportować plik ze skryptem w pliku index.html
- podpiąć funkcję nazwaną ToDoController jako ‘ToDoController’ w module aplikacji w module aplikacji

Krok 2 - zadanie

- Zainstalować przy pomocy bower'a Angular UI Router z flagą –save
- Zaimportować wymagane skrypty w index.html
- Podpiąć zależność w module aplikacji

https://github.com/angular-ui/ui-router/tree/legacy

Krok 3 – zadanie*

- Stworzyć nowy kontroler w katalogu main/addToDo/ o nazwie AddToDoController.js z funkcją nazwaną AddToDoController
- Przypisać w tym kontrolerze do zmiennej obiekt this (**var ac = this**;)
- zadeklarować w zmiennej pusty obiekt todo (ac.todo = {…}) *[obiekt todo powinien zawierać 3 pola: text {string – ''}, description {string – ''} oraz priority {int - 0}]*
- stworzyć listę priorytetów w zmiennej (**ac.priorityList**). Każdy element tej listy powinien składać się z pól: id {int} oraz name {string}. Lista powinna zawierać 3 priorytety – Low, Medium, High
- Napisać funkcję save, która będzie na razie powracała do ekranu z listą todo. Należy dodać serwis **$state** jako zależność do kontrolera i wykorzystać do tego celu funkcję **$state.go** z tego serwisu, np. **$state.go('state_name')** *[deklaracja funkcji - patrz Krok 0 - kontroler]*

- Stworzyć nowy template HTML w katalogu main/addToDo/ o nazwie addToDoTemplate.html. W szablonie tym stworzyć formę z 3 polami: input dla wartości text obiektu todo, textarea dla wartości description obiektu todo oraz select dla priorytetu *[podpiąć w ng-model odpowiednie pola].*
- zasilić listę select listą zdefiniowanych priorytetów poprzez atrybut ng-options [https://docs.angularjs.org/api/ng/directive/ngOptions]
- w szablonie tym dodać również przycisk (element button) z wywołaniem funkcji save (**ng-click**) powracającej na ekran listy
- w analogiczny sposób stworzyć funkcję **addToDo** w kontrolerze ToDoController do przechodzenia
- Zaimportować kontroler w index.html
- Podpiąć zależność w module aplikacji ***todoApp***.**controller**(**'AddToDoController'**, AddToDoController);

Krok 4 – instalacja MongoDB

- Instalacja MongoDB: https://www.mongodb.org/downloads#production
  Uruchamianie https://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/
  - stworzyć katalog z bazą na dysku twardym np. D:\mdb\
  - w linii poleceń przechodzimy do katalogu, gdzie zainstalowane zostało MongoDB *[zazwyczaj C:\Program Files\MongoDB\Server\3.2\bin]*, następnie wywołujemy:

    mongod.exe --dbpath d:\mdb\

- Uruchomienie serwera node.js z backendem:
  - w linii poleceń w katalogu ***/express*** wywołujemy

    npm install

  - następnie:

    node server.js

    [może być konieczna zmiana portu, zależy na jakim baza danych się odpaliła]

Krok 4 – zadanie*

- Dla zadanego API:
  - dodawanie todo [**POST**]: $http.post('http://localhost:8080/api/todos', toDo)
  - usuwanie todo [**DELETE**]: $http.delete('http://localhost:8080/api/todos/' + toDoId)

  utworzyć pozostałe funkcje w serwisie – **addToDo** i **deleteToDo** na wzór **getTodos**.
- Dla deleteToDo w serwisie stworzyć funkcję w kontrolerze ToDoController, które ją wywołuje. Pamiętać o tym, że funkcja w serwisie jest parametryzowana identyfikatorem todo. Uwaga: usługa DELETE zwraca komplet danych po usunięciu rekordu – należy podmienić model z listą todo *[patrz funkcja getTodos]*
- Dodać w szablonie toDoTemplate.html pole input o typie checkbox wewnątrz dyrektywy ng-repeat [np. w znaczniku <td>]. Na akcję ng-click podpiąć stworzoną w kontrolerze funkcję do usuwania todo [pamiętać o przekazaniu identyfikatora todo: td.deleteToDo(todo._id)].

Krok 5 – zadanie

- Dodać pole input nad listą z todo w szablonie dyrektywy z listą (**toDoListDirectiveTemplate**) z atrybutem **ng-model=”searchToDo”**.

- Do **ng-repeat** dodać filtr na pole text obiektu todo według modelu wpisywanego w dodane pole input

```
ng-repeat="todo in tdd.todoList | filter: {text: searchToDo}"
```

Krok 6 – zadanie

- Analogicznie do przedstawionego materiału wykonać walidację dla pola z opisem (**description**) – pole powinno być obowiązkowe.
- Pole powinno się oznaczać kolorem czerwonym
- Dodatkowo pod polem powinien prezentować się komunikat walidacyjny w przypadku błędu walidacji.

COMARCH

JUTRO?

[STAŻ]CA

STAŻ WAKACYJNY

1 LIPCA - 30 WRZEŚNIA 2016

COMARCH

STAŻ PROGRAMISTYCZNY

STUDENCI II, III, IV lub V roku
studiów informatycznych i pokrewnych

Technologie do wyboru
(2 z 4):

Java | C/C++ | .NET
bazy danych

COMARCH

STAŻ PROGRAMISTYCZNY

GDAŃSK | GLIWICE | KATOWICE
KRAKÓW | LUBLIN | ŁÓDŹ
POZNAŃ | RZESZÓW
WARSZAWA | WROCŁAW

COMARCH

JUTRO? [STAŻ]CA

FORMULARZ ONLINE

Wypełnij do

21 KWIETNIA

na staz.comarch.pl