ANGULAR 1

Peter Gurský, peter.gursky@upjs.sk

Moderné client-side webové aplikácie

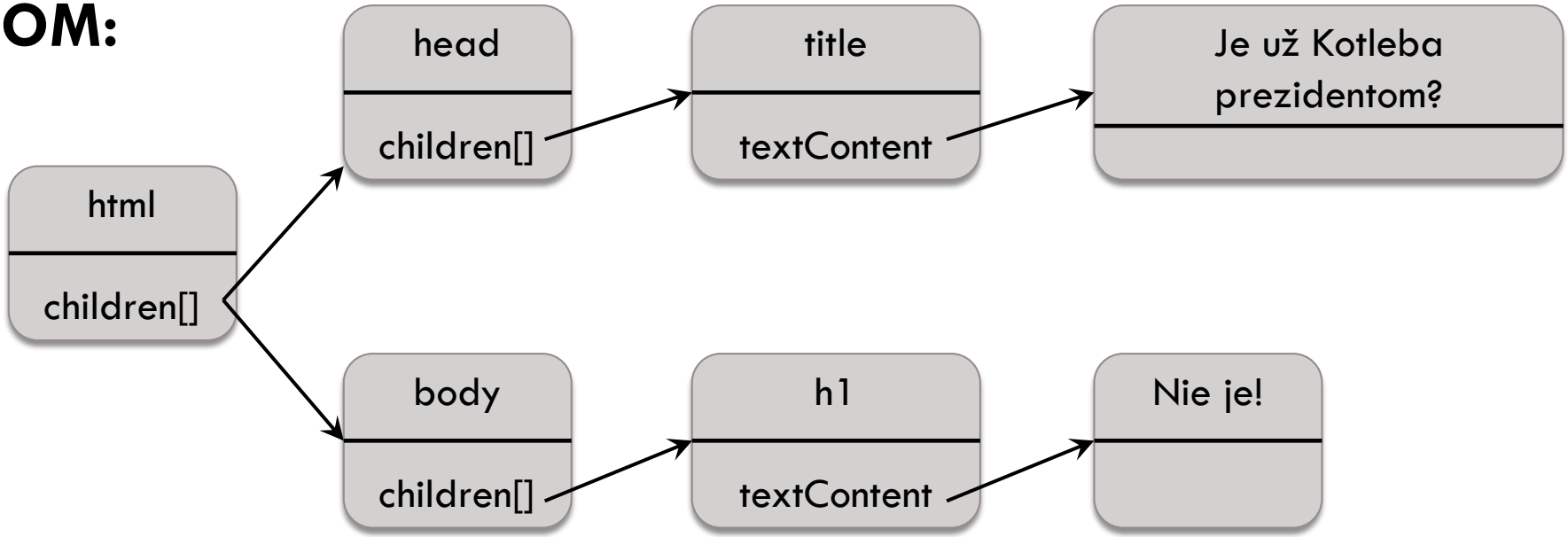
- Čistý JavaScript
- a.k.a. single-page applications, fat (thick) client
- Server poskytuje/prijíma entity – REST volania
  - perzistentná a business vrstva + REST service
  - ľubovoľný jazyk (Java, PHP, JavaScript,…)
- MVC v prehliadači
  - aplikácia modifikuje priamo DOM model zobrazovanej stránky
  - komponenty predstavujú podstromy DOM modelu
  - modely komponentov typicky predstavujú entity poskytované REST serverom
- Angular, Vue, React, Ember, Meteor, ExtJS, Aurelia,..

HTML:

```
<html>
  <head>
    <title>Je už Kotleba prezidentom?</title>
  </head>
  <body>
    <h1>Nie je!</h1>
  </body>
</html>
```

DOM model tvoria JS objekty typu
Node
a okrem textových uzlov aj typu
Element, HTMLElement, ...

DOM:



Angular (od verzie 2)

- Final release verzie 2: 14.9.2016, aktuálne verzia 17
- Využíva jazyk TypeScript
  - Typovaná nadstavba JavaScriptu (od ECMAScript 6)
    - už existujú triedy (syntaktický cukor)
    - moduly
    - lambdy
    - ...
  - dodané anotácie, generiká
- framework = dopĺňame komponenty do štartovacej aplikácie

Vývojové prostredie

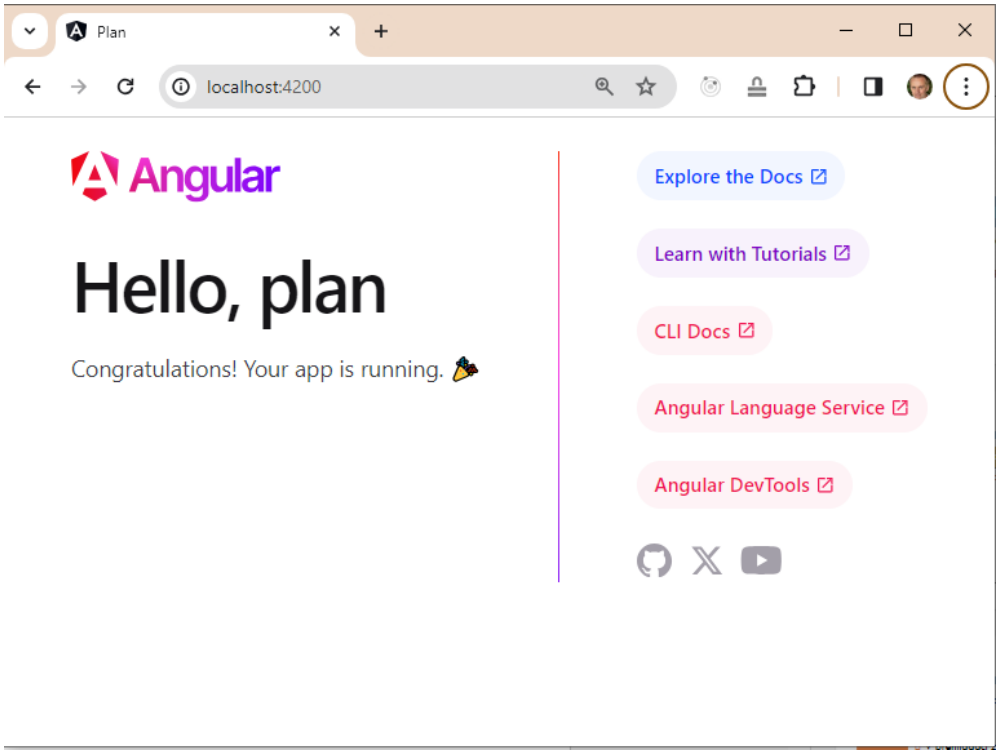
- Mnoho mágie na pozadí
  - Typescript sa kompiluje do javascriptu, ktorému rozumie webový prehliadač
    - Po každej zmene súboru
  - NodeJS server poskytuje skompilované súbory prehliadaču
  - Prehliadačom sa napojíme na NodeJS server na http://localhost:4200/
  - Prehliadač si stiahne súbory a spustí našu aplikáciu
    - Chová sa tak, ako sa bude chovať, keď sa nasadí naostro

Rozbehávame vývojové prostredie

- Nainštalujeme nodejs
  - *https://**nodejs**.org/*
  - *s ním dostaneme aj balíčkovač npm*
- IDE (od Microsoftu):
  - Visual Studio Code: http://code.visualstudio.com/
- Nainštalujeme si zostavovač Angular projektu
  - npm install -g @angular/cli
- Vytvoríme si kostru projektu a vstúpime do projektu
  - ng new PROJECT_NAME
  - cd PROJECT_NAME

Spúšťame projekt

- V koreni projektu spustíme NodeJS server:
  - ng serve
- V prehliadači zadáme URL http://localhost:4200/



Čo nám to vzniklo?

- node_modules
  - ...
- src
  - app
    - **app.component.css**
    - **app.component.html**
    - app.component.spec.ts
    - **app.component.ts**
    - **app.config.ts**
    - **app.routes.ts**
  - index.html
  - main.ts
  - …

kopa modulov/knižníc, ktoré môžeme využiť

tu sa budú nachádzať naše veci

HTML šablóna koreňového komponentu

trieda koreňového komponentu

Konfiguračný súbor aplikácie

Nastavenie router-a

Hlavná stránka, ktorá sa v tele odkazuje na koreňový komponent (telo nemeníme, max. hlavičku)

Spúšťač projektu

Vloženie komponentu do stránky

src/index.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>názov aplikácie</title>
  <base href="/">
  ...
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

src/app/app.component.ts

```
import { Component } from '@angular/core';

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

src/app/app.component.html

```
<h1>
  Hello, {{title}}!
</h1>
```

zobrazenie inštančnej premennej

Vložme vlastný komponent do hlavného

- Ideme do príkazového riadku a v adresári src/app spustíme
  - ng g component users
- Vznikú 4 súbory komponentu
  - Ako selector v src/app/users/users.component.ts sa nastaví **app-users**
- Ak ho chcem vidieť v inom komponente, pridáme
  - do jeho šablóny
    - **<app-users**>info o používateľoch</ **app-users>**
  - do importu
    - `UsersComponent`

Zobrazenie jednotlivých hodnôt

app/users/users-component.ts

```
import { Component, OnInit } from '@angular/core';

@Component(...)
export class UsersComponent implements OnInit {
  title: string = "Zoznam používateľov";
  users: string[] = ["Janko", "Marienka"];
}
```

app/users/users-component.html

```
<h2>{{title}}</h2>
<ul>
    <li>{{users[0]}}</li>
    <li>{{users[1]}}</li>
</ul>
```

Zobrazenie zoznamu cez direktívu *ngFor

app/users/users-component.ts

```
import { CommonModule } from '@angular/common';

@Component(…
    imports: [CommonModule],
…)
export class UsersComponent {
    title: string = "Zoznam používateľov";
    users: string[] = ["Janko", "Marienka"];
}
```

app/users/users-component.html

```
<h2>{{title}}</h2>
<ul>
    <li *ngFor="let user of users">{{user}}</li>
</ul>
```

Zobrazenie zoznamu cez @for

app/users/users-component.ts

```
@Component(…)
export class UsersComponent {
    title: string = "Zoznam používateľov";
    users: string[] = ["Janko", "Marienka"];
}
```

app/users/users-component.html

```
<h2>{{title}}</h2>
<ul>
    @for(user of users; track user) {
        <li>{{user}}</li>
    }
</ul>
```

Nejaká unikátna primitívna hodnota (string, number)

Radšej pracujme s entitami

```
// app/user.ts
export class User {
    constructor(
        public name: string,
        public email: string,
        public id?: number,
        public lastLogin?: Date,
        public password: string = ''
    ) { }
}
```

Vyrobme si triedu entity User

```
// app/user.ts
export class User {
    constructor(
        public name: string,
        public email: string,
        public id?: number,
        public lastLogin?: Date,
        public password: string = ''
    ) { }
}
```

triedu budeme vedieť použiť v iných súboroch

Vyrobme si triedu entity User

```
// app/user.ts
export class User {
    constructor(
        public name: string,
        public email: string,
        public id?: number,
        public lastLogin?: Date,
        public password: string = ''
    ) { }
}
```

nie všetky parametre sa pri volaní konštruktora musia zadať,
id a lastLogin sú defaultne **undefined**,
password je defaultne **prázdny string**

Vyrobme si triedu entity User

```
// app/user.ts
export class User {
    constructor(
        public name: string,
        public email: string,
        public id?: number,
        public lastLogin?: Date,
        public password: string = ''
    ) { }
}
```

z parametrov konštruktora sa stanú verejne prístupné inštančné premenné

```
jano = new User("jano", "jano@jano.sk", 2);
console.log(jano.email);
```

Pre neúnavných pisateľov

```
export class User {
    private _name: string;

    set name(newName: string) {
        this._name = newName;
    }
    get name(): string {
        return this._name;
    }
}
```

```
jano = new User();
jano.name = "Jano";
console.log(jano.name);
```

Iterujeme entity

app/users/users-component.ts

```
export class UsersComponent implements OnInit {
    users: User[] = [new User("Janko","jano@jano.sk"),
                     new User("Marienka","marienka@jano.sk")];
}
```

app/users/users-component.html

```
<ul>
    <li *ngFor="let user of users">{{user.name}}, {{user.email}}</li>
</ul>
```

```
<ul>
    @for(user of users; track user.name) {
      <li>{{user.name}}, {{user.email}}</li>
    }
</ul>
```

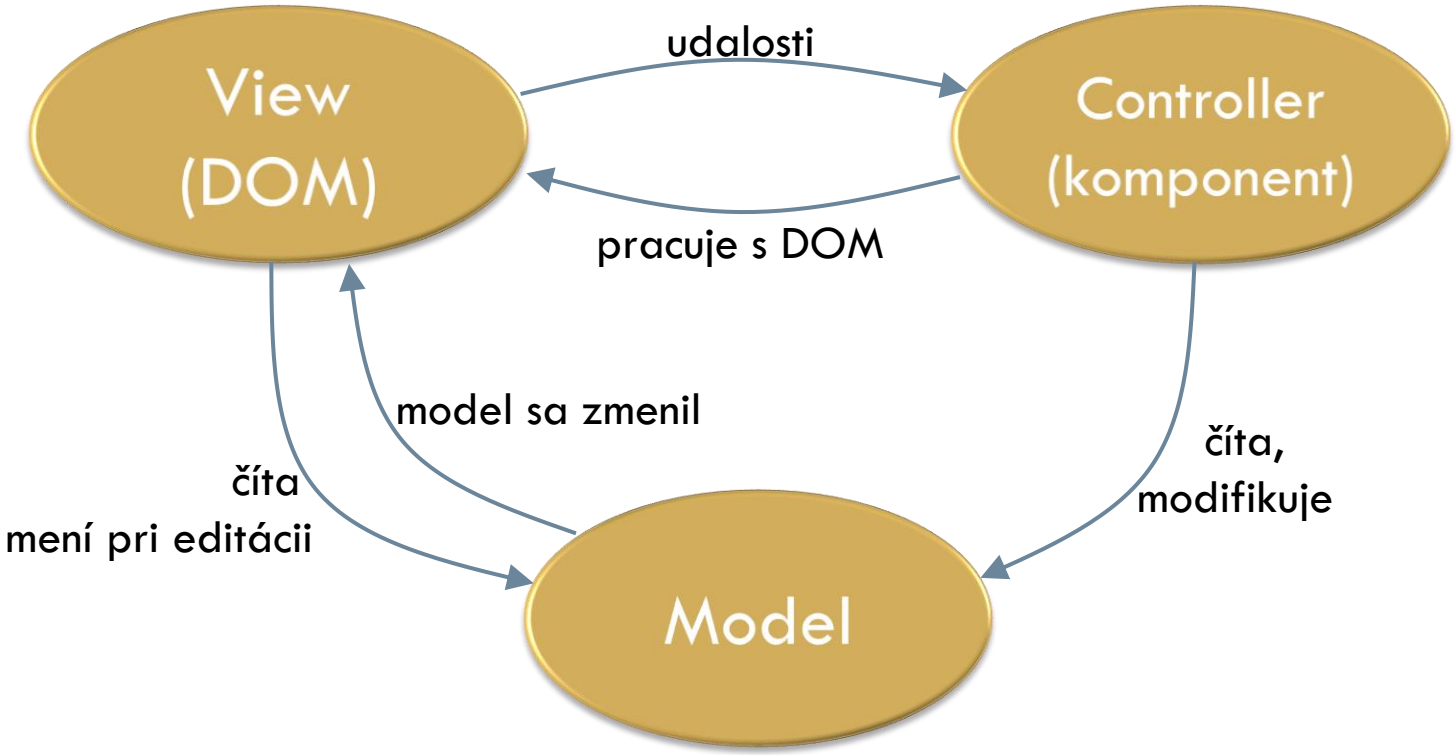
...alebo do tabuľky

```
<table>
    <thead>
        <tr><th>id</th><th>meno</th><th>e-mail</th></tr>
    </thead>
    <tbody>
   @for(user of users; track user.name) {
      <tr>
         <td>{{user.id}}</td>
         <td>{{user.name}}</td>
         <td>{{user.email}}</td>
      </tr>
    }
</tbody>
</table>
```

Model – View – Controller (MVC)

- Odchytáva používateľské akcie
- Prekresľuje GUI, keď sa dozvie zmenu modelu

- Spracováva vstup od používateľa
- Mení alebo vymieňa model



- Uchováva zobrazovaný obsah
- Poskytuje dáta pre View, keď ich potrebuje

Odchytenie udalosti click

app/users/users-component.html (časť)

```
@for(user of users; track user.name) {
  <tr (click)="selectUser(user)">
    …
  </tr>
}
```

app/users/users-component.ts

```
export class UsersComponent implements OnInit {
…
  selectedUser?: User;

  selectUser(user: User) {
      this.selectedUser = user;
  }
}
```

Podmienené časti šablóny

app/users/users-component.html (časť)

```
@for(user of users; track user.name) {
  <tr (click)="selectUser(user)">
    …
  </tr>
}
<p *ngIf="selectedUser">Vybratý používateľ: {{selectedUser.name}} </p>
```

app/users/users-component.ts

```
export class UsersComponent implements OnInit {
…
  selectedUser?: User;

  selectUser(user: User) {
      this.selectedUser = user;
  }
}
```