Marian Benčat

Angular

Konzultace a školení – Frontend, Backend

Aktuální webový vývoj a minulost

Server rendering

- Java, .NET, PHP,…

Client rendering

- SPA aplikace – Angular, Vue, React

Nový tooling

- Node.js a NPM
- Webpack
- Typescript

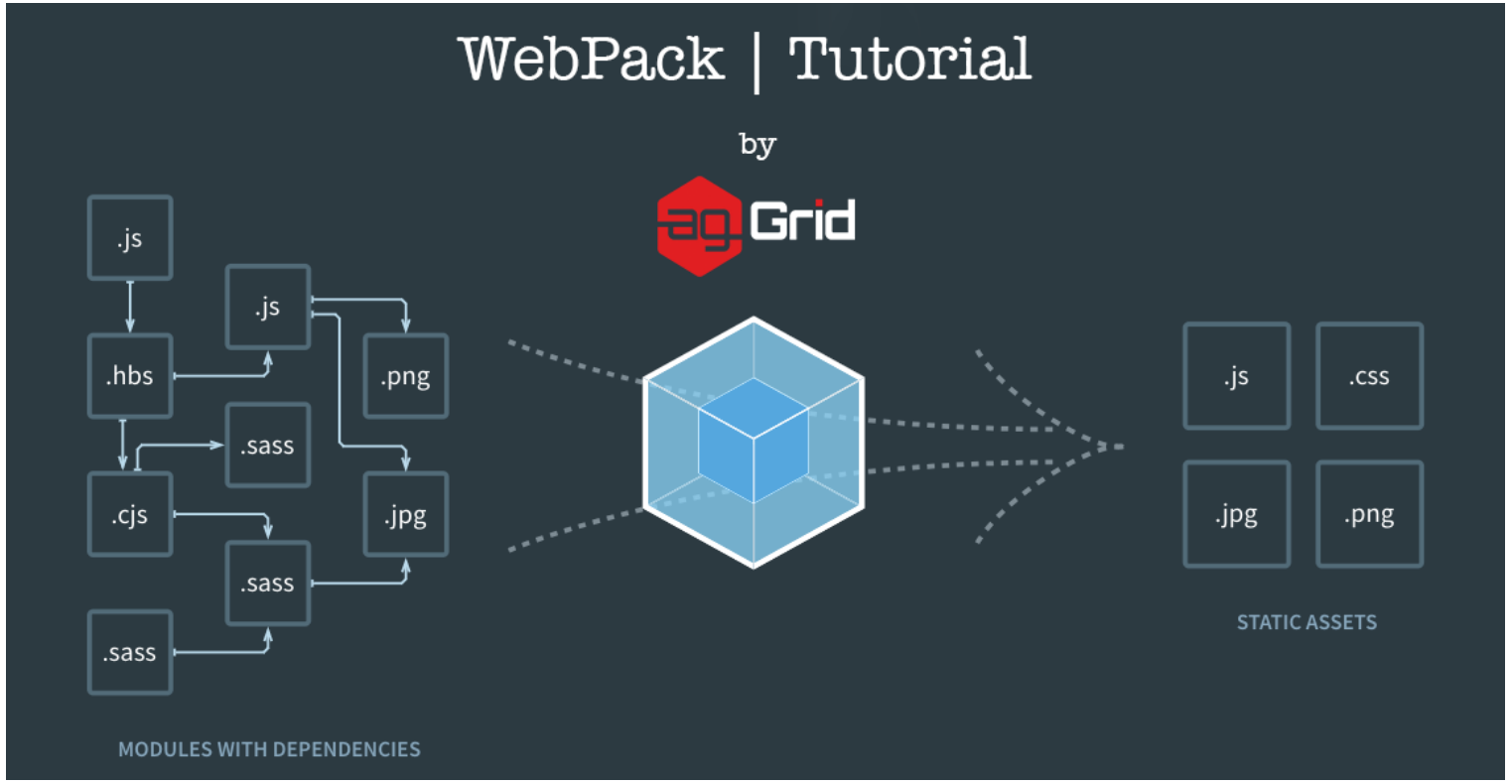
Node, NPM

JAVASCRIPT EVERYWHERE (NO ESCAPE ☺)

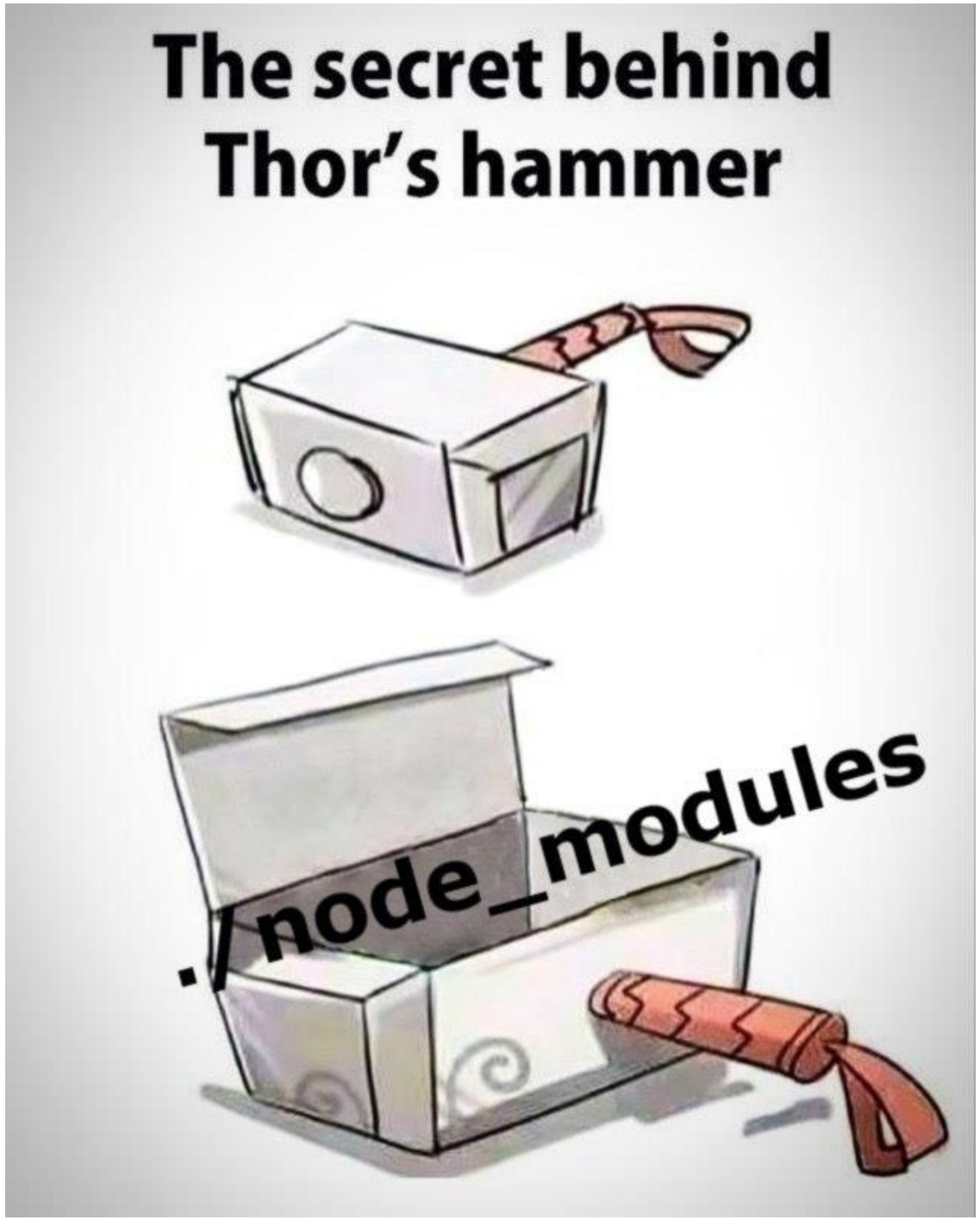
NPM BALÍČKOVAČ, PACKAGE.JSON, PACKAGE-LOCK.JSON

YARN

Webpack


WebPack | Tutorial
by
ag Grid
.js
.hbs
.cjs
.sass
.sass
.js
.sass
.png
.jpg
MODULES WITH DEPENDENCIES
.js
.css
.jpg
.png
STATIC ASSETS

Tree shaking



Vývojové prostředí

- Node.js a NPM
- Visual Studio Code
- Angular CLI
  - npm install –global @angular/cli
- .gitignore
  - node_modules or R.I.P.
- ng new NAZEV
- ng serve

Základní stavební kameny

- Angular.json
- Index.html
- Src folder

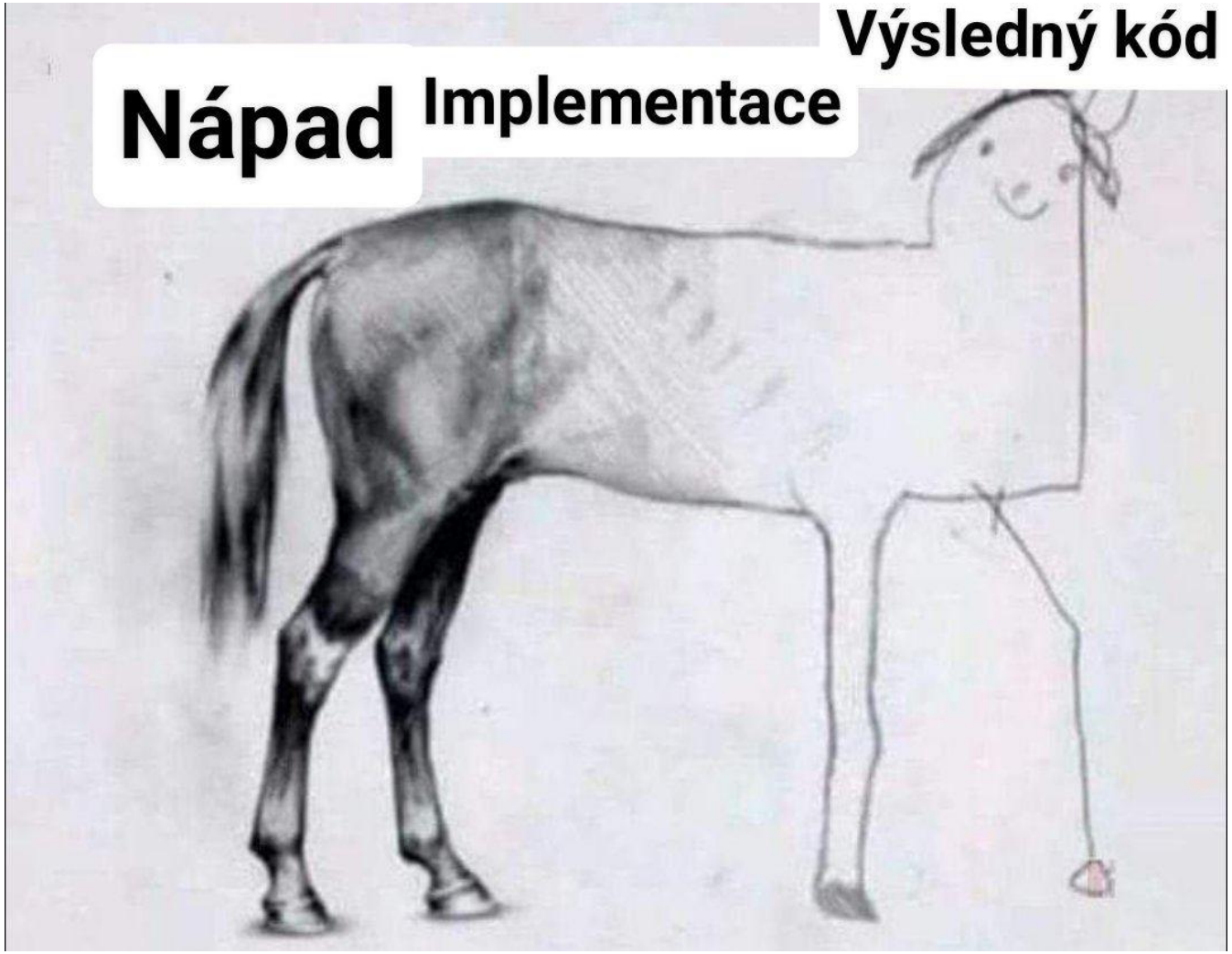
- Building blocks
  - Components
  - Modules
    - Importy a exporty
    - CommonModule, BrowserModule, RouterModule,…
  - … services, pipes,… o tom později

- Javascript rendering a jeho nevýhody
  - Server pre-render, SSR

Doporučená struktura Angular aplikace

- /src
  - /app
    - /FEATURE_NAME
      - /components
        - /MY_COMPONENT
          - MY_COMPONENT.component.ts
          - MY_COMPONENT.component.html
          - MY_COMPONENT.component.scss
      - /pipes
      - /services
        - MY_SERVICE.service.ts
      - …
      - FEATURE_NAME.module.ts

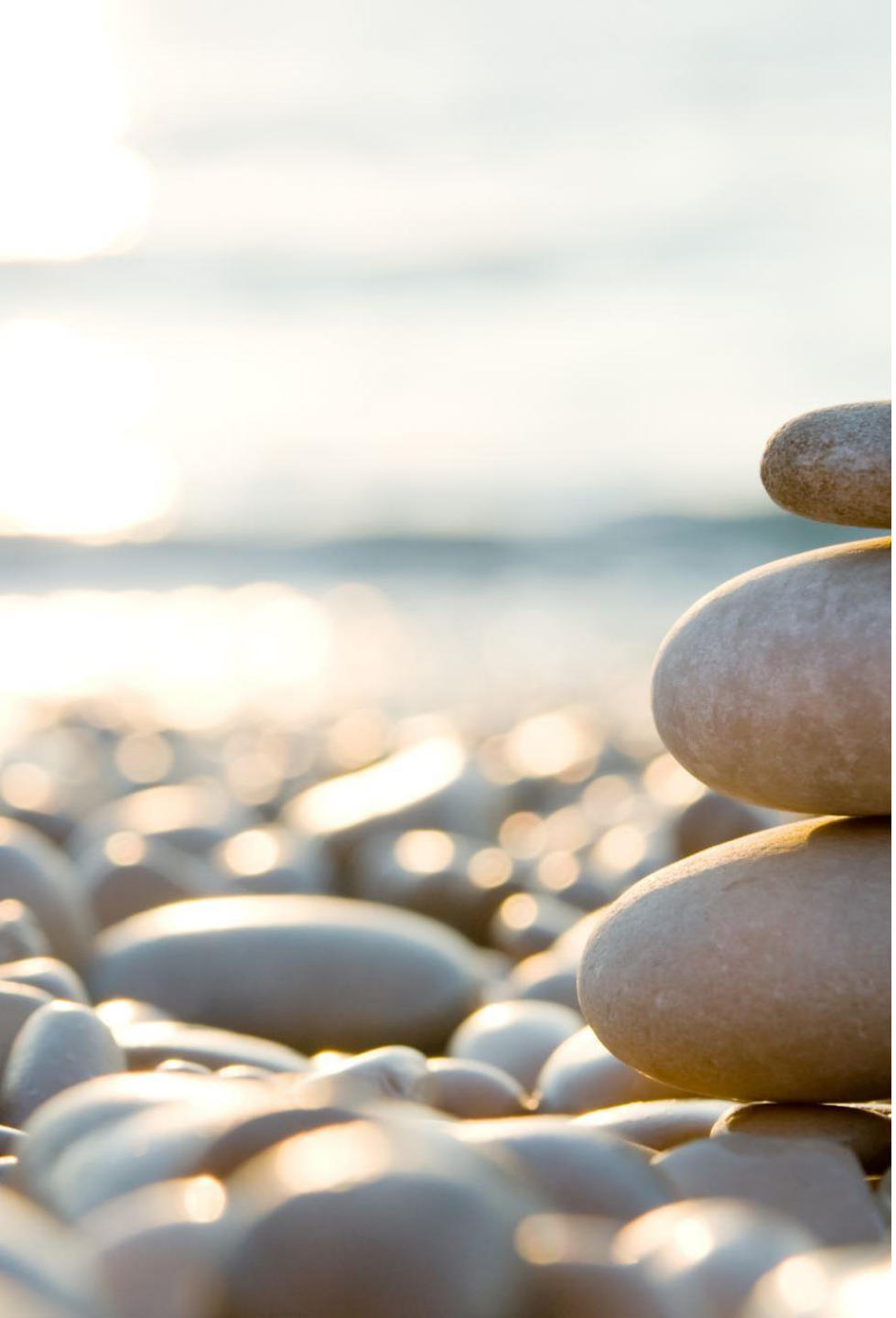
Vývojový proces



Typescript a C#/Java

Typescript a C#/Java – hlavní rozdíly

- Mnohem méně ukecaný
- Velmi jednoduchý na naučení díky jemné konstrukci
- **Typovost je hlídána v COMPILE TIME, v runtime jde o klasický javascript!!!**
- Mnohem více multiparadigmatický než Java / C#
- Vnímání interfaceu / třídy (otypováním se nevolá žádná konstrukce objektu)
  - Model binding na backendu?
- Osobně ho vnímám jako zdaleka nejdokonalejší jazyk současnosti



Na výuku Typescriptu

- https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html
- https://stackblitz.com/

Typescript a ES Moduly

Import

Export

Barel files a namespacy (paths)

Tree shaking a co mu brání

Komponenty 1

- Co je komponenta
- Popis dekorátoru
- Registrace do NgModulu a použití
- Template, CSS, code-behind
- Interpolace a základní logické konstrukce
  - ngIf
  - ngFor
- Event binding, 2-way data binding
- Pipes

Komponenty 1

- CSS scoping
- (keyup)=„0“ v tutoriálech
- Child componenty, Inputy a Outputy

Change tracking - základy

- Monkey patching
- Zone.js
- Single thread apartment
  - Desktopové frameworky

Komponenty – kompozice a SOC

- NG ADD
- Manuální přidání
- Angular.json a Přidání modulů

Dependency injection v angularu

- Depencency injection
  - IOC container
  - Constructor injection
- Angular má hierarchický DI (o tom později)
  - Registrace do containeru
  - Získání instance
    - Singleton? Multiton? Factory?
  - Registrace
    - Servisy
    - Hodnoty
    - Objektu

Formuláře

Template driven formuláře

- Old-school
- Jednoduché
- Příliš se nepoužívají

Reactive forms

- Složité
- Velmi mocné
- Bohužel né zase tolik reaktivní

Jednotlivé typy

- AbstractControl
- FormControl
- FormGroup
- FormArray
- FormBuilder

Formuláře

Direktivy

FormControl

FormGroup

OnSubmit

CSS classy

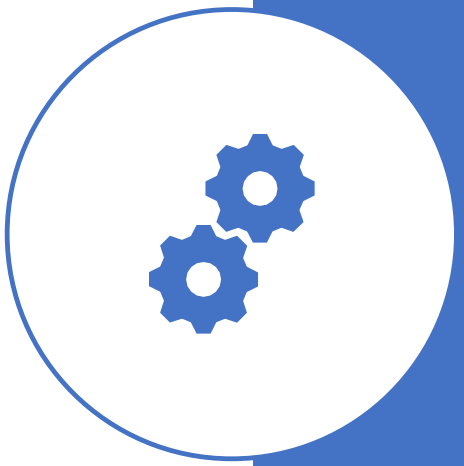
Ng-*

Eventy

Validace a reakce na ně

Routing

- Routing a routování – motivace
- Strategie routování
  - Hash stratégy
  - HTML5 routing a History API
- RouterModule
  - Definice routy
  - Skládání routingu
  - Layouty (masterviews)
  - Navigace a direktivy

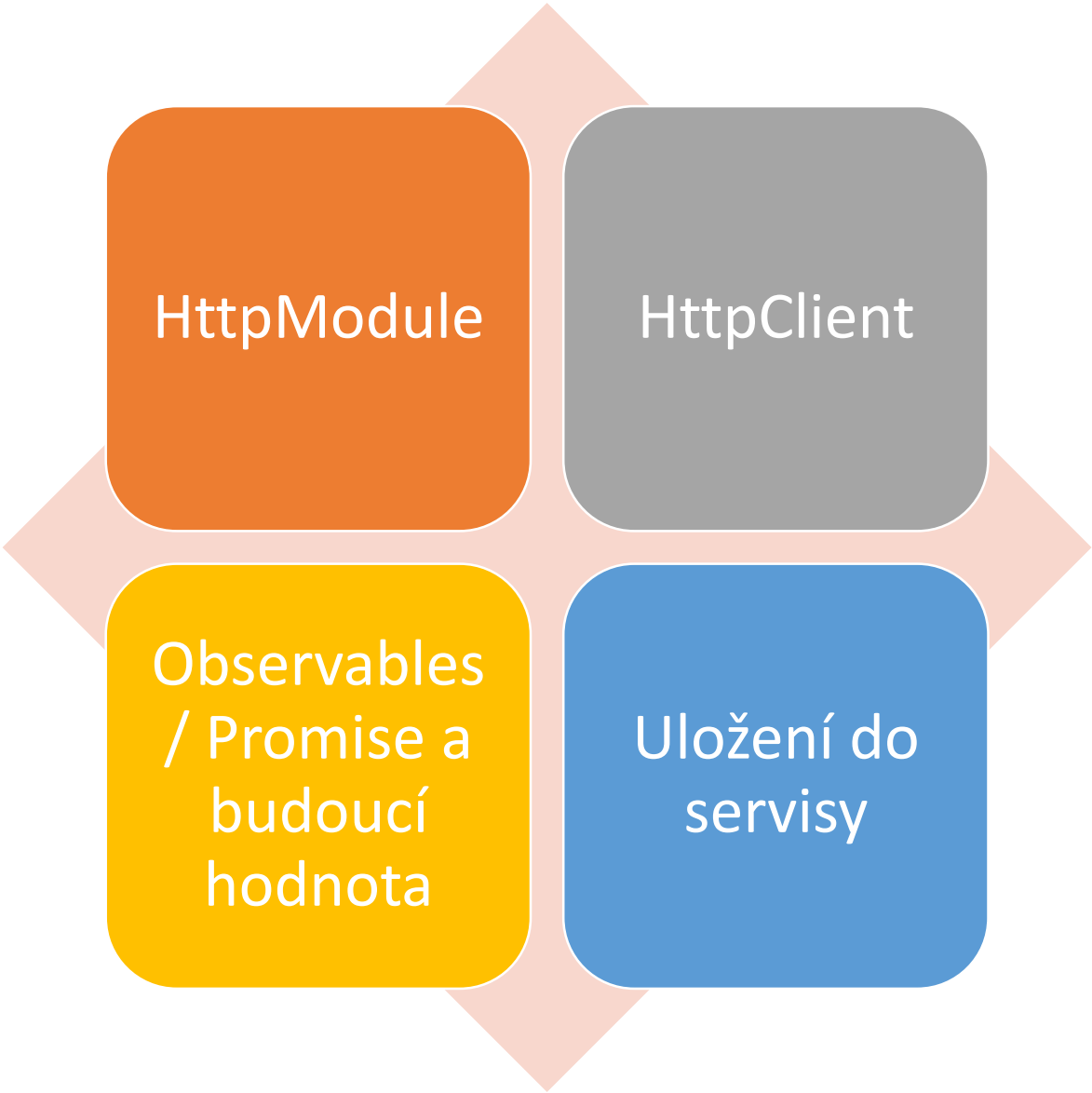


Moduly podruhé a lazy loading

Modul jako IOC container

Lazy loaded moduly

Komunikace se serverem



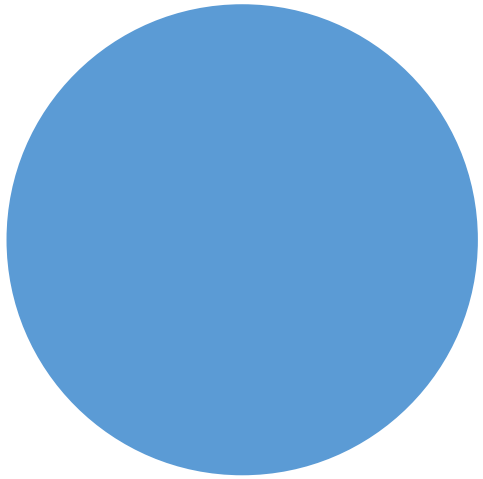
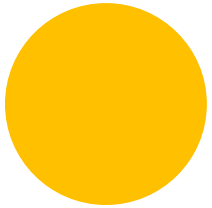
Komponenty podruhé

- Problémy s výkonem kvůli změnám stavu
- OnPush a immutabilita
- Zopakování angular change detection
- ChangeDetectorRef, markForCheck, detectChanges, appTick
- Async pipe a zretezeni pomoci ngIf as

Komponenty potřetí

- ViewChild, ViewChildren, templateRef
- ContentChild, ContentChildren, ng-content

- Templating a skládání komponent, loading komponenta

Formuláře podruhé

Control value accessor a custom input

- Guardy
- Resolvery
- QueryParams
- Params
- Reakce na změny routy

Routing podruhé

Prod build

DEV build vs PROD build a balíky

Hashování a cache busting

Index.html

Doporučené nastavení web serveru

RxJS – reaktivní programování

Funkcionální programování

Reaktivní programování

Callback a eventy

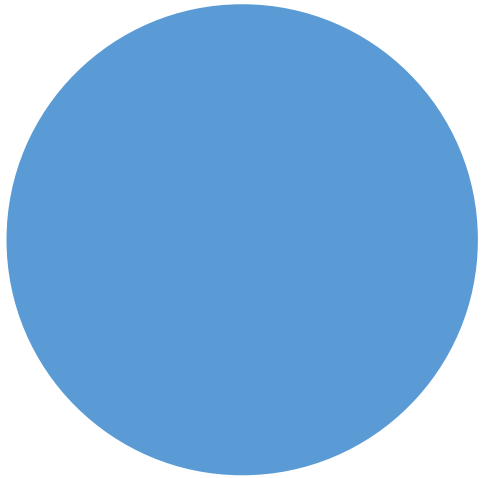
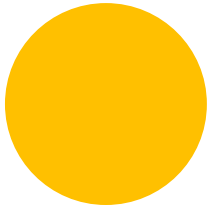
Async / await

Základní typy observable a popis implementace

Closures

Základní operátory (map, filter, take,..)

Subscribe a Unsubscribe, asyn pipe

RxJs 2

Formulář, rxjs a komunikace se serverem

Způsob předávání dat mezi komponentami

- Input / Output
- Observable service
- Router a route
- Čunačinky (window, storage,…)
- State management systém
  - Redux
  - Mobx
  - Akita
  - Custom

NGRX/Platform

Redux

Actions

Reducer

Effects

RouterStore

RxJS 3

- Složitější operátory a jejich správné použití
- Custom operátor

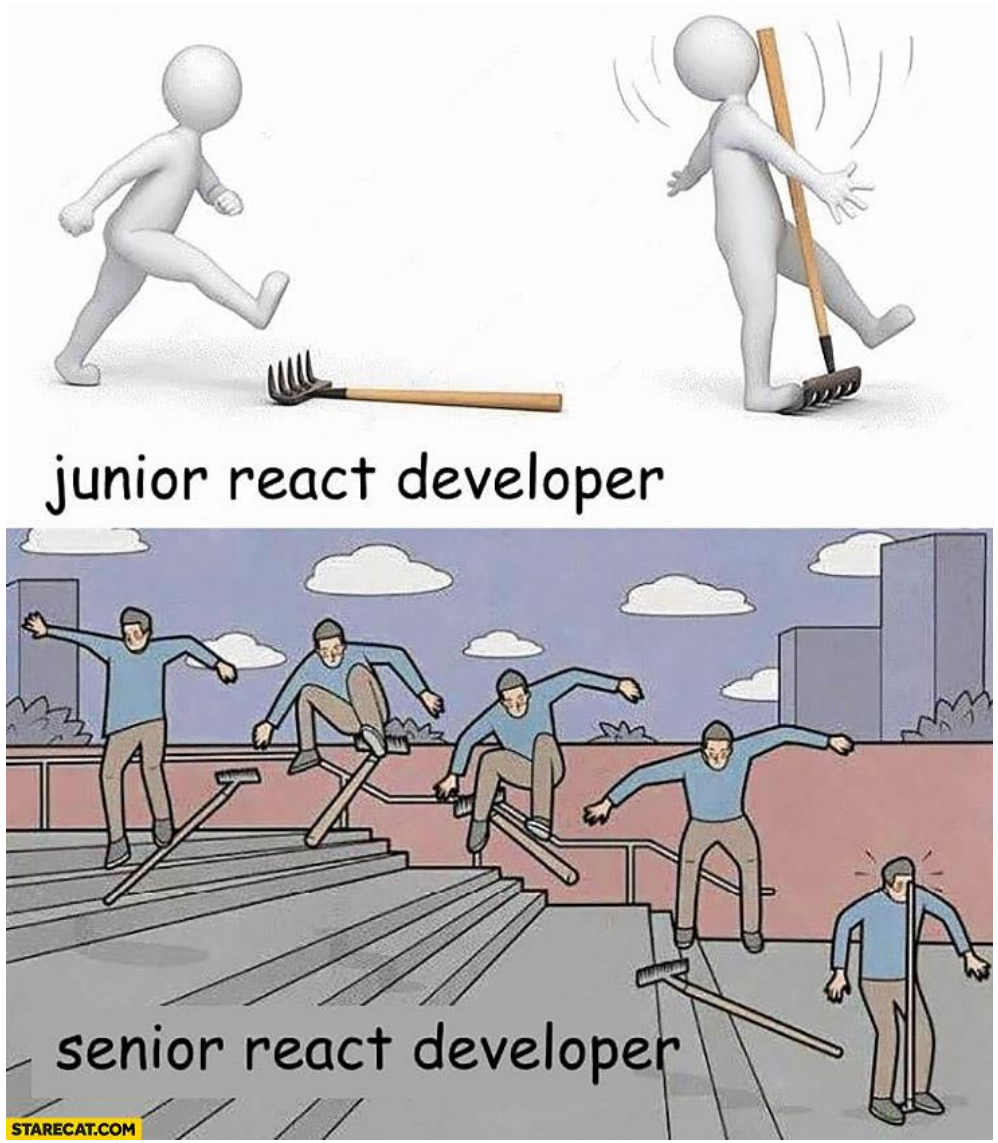
Optimalizace

- Zones a jejich efektivní použití
- Zone.less
- Manuální change tracking

- Lepší code splitting
- Preload dat a částí aplikace

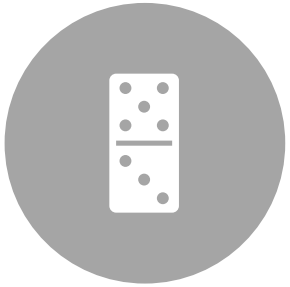
Nx Workspace

- **Používejte to, než to autoři dodrbají potřebou mít to univerzální i s Reactem**
- Monorepo
- Rozdělení kódu více do knihoven
- Sdílení napříč projekty
- Dependency graph



Unleash the power

Vlastní datová knihovna na práci s daty NGRX/Mára (Data)



Queryování dat pomocí Pipe

Dynamické loadění komponent

Data connectory