

~~Adr-Weby~~ příprava  
(Video)

NSW 1153

Proxy server

- ↳ server bere data z internetu
- za uživatele
- "middle-man"
- ↳ proč? 1) souborní, stojí IP adresu

PC - Proxy - Website  
11.22.53.44. → { 22.33.44.55 }

↳ výhlost ~ cache, kde jsou uloženy stránky  
(saver je bandwidth)

3, activity logging ~ ukládá si historii surfování  
a délku  
+ pravidla zakazování  
stránek

Proxy (nesifuje)  
poslona' data

VPN ~ schovává IP a šifruje data,  
zároveň activity logging

# Proxy vs. reverse proxy

proxy ~ někdo/nejco dělá aktivitu za někoho, např. server

Forward Proxy ~ „střed' připojení“  
Uživatelé ↓ vezme data na sebe  
• blokuje špatné sítě  
• maskuje IP, používá „fake“ své  
• jede obejít restriky

PC → sítě

Reverse Proxy ~ „střed' příchozí data“  
Server

PC → chci něco  
z dat./serveru

- použde při DDoS

↳ uživatel poch „nemluví“ přímo se serverem

- ↳ maskuje IP serveru,  
r. proxy má své
- rozdělení požadavků
- LOAD BALANCER**

# VMS vs. (Virtual machines)

- ↳ vše aplikace  
na 1 serveru
- simulace HW/sw

Sah?

- 1) server, Hardware
- 2) hypervisor
  - dovoluje běh  
vše VMs
- 3) VMs (Lin/Win)
- 4) aplikace,

- problémy
  - ↳ hodně paměti  
berou, CPU a RAM
  - ↳ pouze jistý  
(hazda' alpha sůj OS)
  - ↳ licence nula'

OS serveru a aplikace  
nemusí být stejný

# Containers

- ne simuluje  
sboje, pauze  
aplikace

Obsahuje:

- a, soubory
- b, konfigy
- c, závislosti  
(ready to run)

Website → Container  
↳ distribuce

# Docker

- Sah?
- 1) server, Hardware
  - 2) operační systém
  - 3) container engine  
(stejně container soubory,  
předává kernelu OS)
  - 4) aplikace

↳ aplikace nemají  
oddělený OS, OS  
se předává  
↳ všechny -

↳ užívá: crash  
mobil je vše  
OS serveru a  
OS aplikace MUSÍ  
BYT STEJNÝ

(druh)

Business  
↓

Aplikace

na serveru  
(1 na server)  
~ hodně serverů

REACT / Virtual Dom

Model

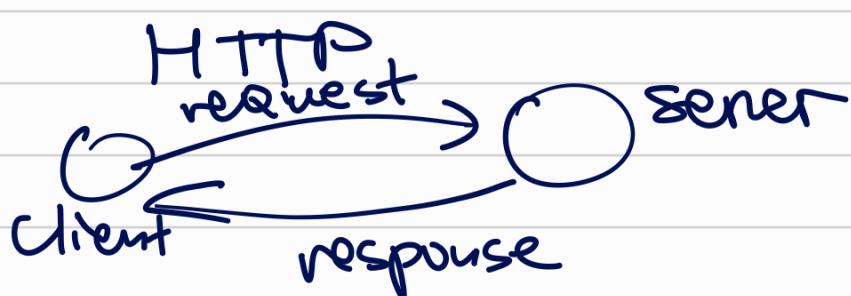
↓ View

View ~ update in place

React, View díí reacu vykreslení,  
pamatuje si to (DOM)  
real

# Web Sockets

- protokol využívá pro oboustrannou web server - browser komunikaci v reálném čase
- ↳ posílání a přijímání dat v jednom TCP



hardy' request = nové propojení

Web Sockets

↓  
full duplex asynchronní  
posílání zpráv

- client i server posílají requesty a responses nezávisle (TCP propojení)  
(weby, ale i např. WhatsApp)

HTTP polling  
↳ AJAX

- hodičky prázdných odpovědí serveru

long polling

request „Hanging GET“  
server prostě delší a  
rah response pošle  
(nebo time Out)  
↳ nezávisle  
(periodické pře-pojování)

1. stage: client připojí k serveru  
pros Websocket „handshake“
  - pokud server Websocket podporuje, pošle odpovídající header
2. stage: nahrazení handshaku  
TCP propojením

3. stage: posílání dat

- Web Socket Handler  
(připojen uživateli)  
↳ sever machine, který  
udržuje „živou“ připojení

použití:

Real time web → Trading  
Chatting  
Gaming  
(jen cast)

# Prezentace Škoda

## 1. PHP

trait → class-like <sup>řeší problem pouze 1-dejíčko</sup>  
 (nejde instancovat) <sup>ností PHP</sup>

trait Om {

} public function Hey() { echo 'Hi'; }

Class a {

} use Om;

}

\$a1 = new a();

\$a1 → Hey()

vrátí 'Hi'

generator → iteratory (yield  
<sup>values</sup>  
 ↳ lze než klasické iterace,  
nenamíš uhládat array)

function doSome () {  
 return 1;  
 return 2;  
 }

vrátí 1,  
exitne

function doSome () {  
 yield 1;  
 yield 2; vrátí 1  
 }

return vs. yield → return exitne rámou,  
 yield vrátí hodnotu  
 a deha'

for (\$i=1; \$i < \$n; \$i++) {

ne-generator

\$numbers[] = \$i

generator

yield \$i;

reflection → inspect class, method, ...

namespaces

namespace App\Model;

errors

↳ try/catch

+

standard php library!

socket. socket() as server-socket

↳ primo esempio

## 2. Sharing is caring

Front Controller → config, initialize beans  
initialize app  
→ routing  
→ dispatch (ber outputs,  
complete request)

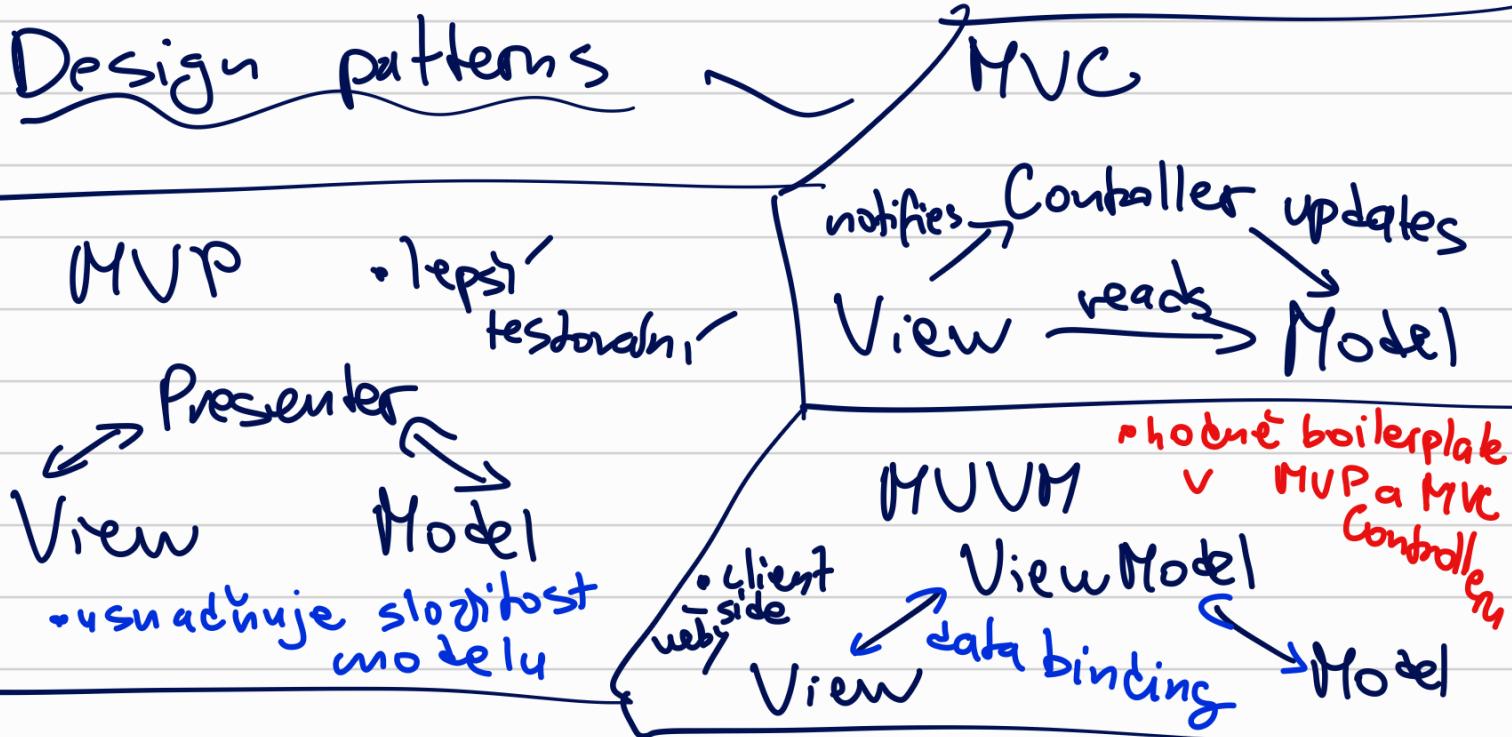
- dependency injection

/\* \*  
\* @ component Welcome  
\*/

```
class WelcomeController implements IController  
/* * @ inject IDatabase */  
public $database;
```

...

## Design patterns



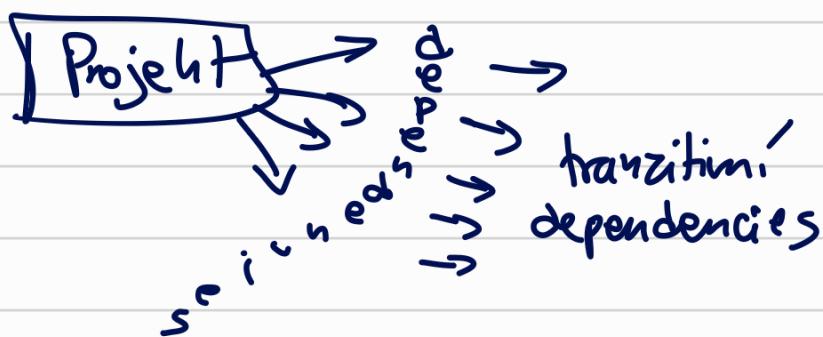
Data model  
↳ prioritizuje repr. dat

ORM – object relational mapping (no SQL)  
relational datab.

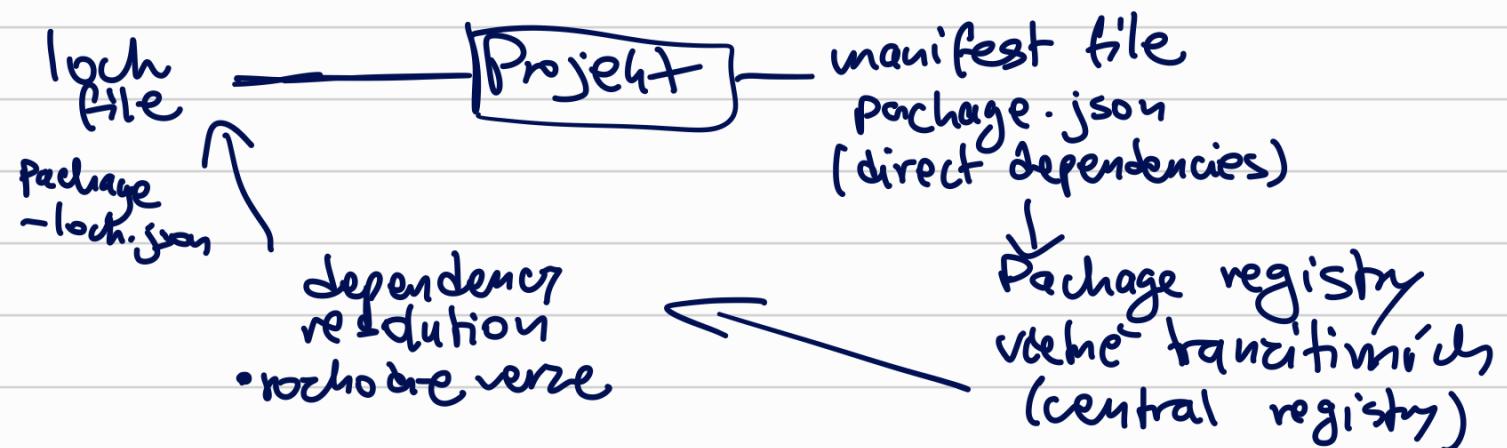
# ↓ Doctrine ~ php

NOT ORM (navezdil od ORM nemusí znát celé DB schema)

Dependency manager  
↳ pip, composer



- declare, resolve  
a manage  
dependencies



Add, remove, install, update

Lock a manifest na GIT !

Prettier ~> deťať hoď hezčí  
(code formatter)

Linter ~> analyzá kódy ES Lint

### 3. JavaScript eco-system

CSS libs

- grid system

CSS preprocessors

(SASS → CSS)      <sup>Magic</sup>      • SASS - syntactically awesome stylesheets

- Leaner CSS (Less)

Package managers

npm, bit

npm

→ Node.js

↓

JS Runtime

yarn

↳ „lepsí“ npm

↳ frameworky, používají ho dle někoho  
jiného (libs, tools)

HTML → JS

JS XML,  
HTML

TypeScript ~ superset of JS

- strict typing  
(JS pro libi preferují 'c' / 'ts')

silne  
typování ↗ types

basic ~ string, num, bool, ..  
arrays  
objects

# ELM

- ↳ functional language
- ↳ forma implementace:
  - kratejš kód
  - static typing

↳ GUIs ve web browserech

(virtual DOM zahrnuje CSS a HTML)

(JS používá portál ~ message exchange)

## 4. JavaScript Ecosystem 2

Předpoklady:

1, kópirování složek

2, použití našejí

3, pouštění testů

→ Dev server, Hot reloading, komprese, ...

Gulp.js = Task runner

• konfigurační soubor (plugging...)  
↳ program

*při každé změně new*

Webpack = bundler (plugging, loader)  
• komplexní config

Parcel.js = zero-config bundler

(JS, HTML, CSS, TypeScript, ...)

Vite → bundler → vice files a dependencies  
to 1 sourcemap  
ESM module nativ.  
(unbundled → better room, nemusíme  
"bundle" project      bundle pred server ready)  
plugin: SSR      ? hot module replacement  
(server side jako Next.js.)      → hned se menu,

React → set up jehož commandem  
npx create-react-app app1  
cd app1  
npm start

Meta-frameworky  
(→ Next.js (react)  
Nuxt.js (vue) ...)

Mono-repository → submodul Gitu  
LERNA      -single repo

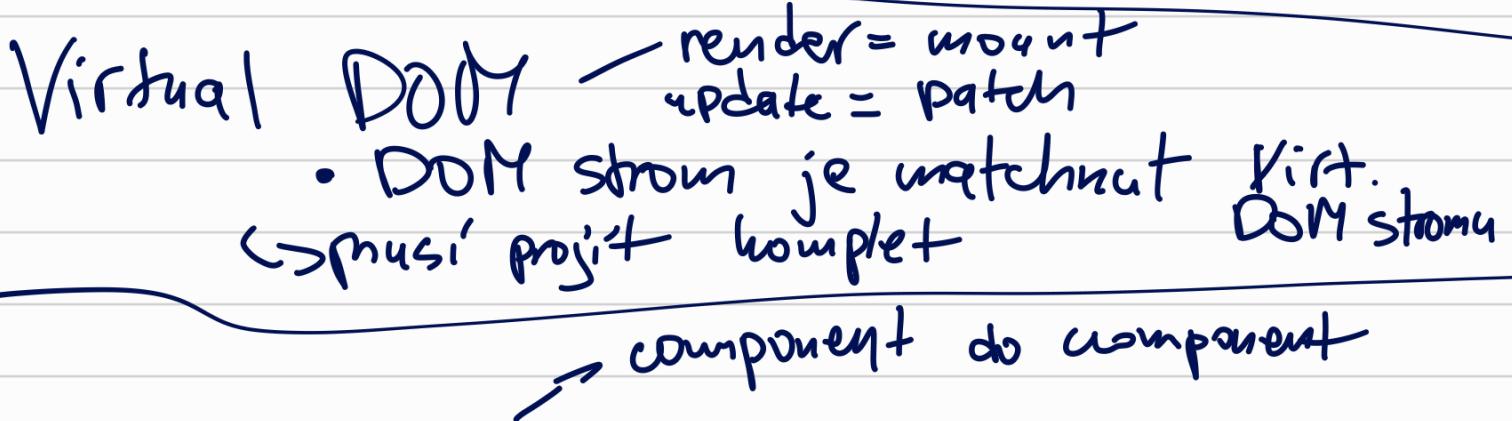
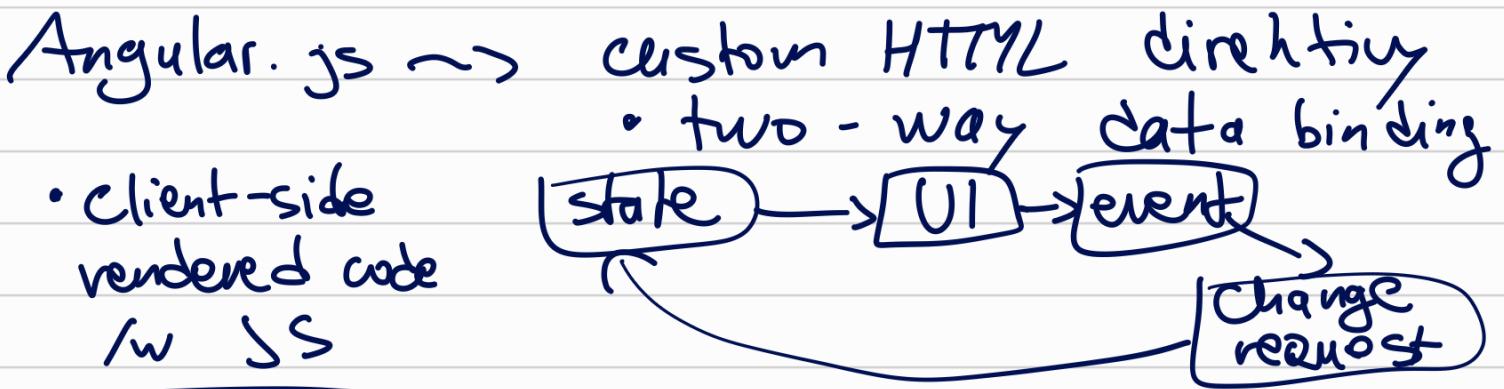
## 5. User Interface Java Script

jQuery → zjednodušení manipulace  
HTML, JavaScript atd.

- 1) Vytvoř UI
  - 2) initialize jQuery  
↳ registrace eventů
  - 3) update data modely
  - 4) update UI přes jQuery
- DOM  
manipulace

D3.js → vizualizační knihovna  
data ~ D3.js

SVG  
scalable vector



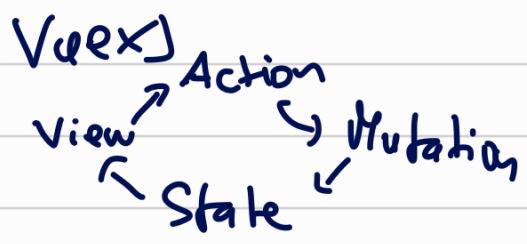
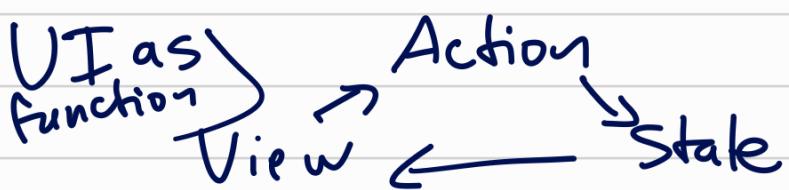
React.js → na' virtual DOM

- JSX
- UI jako funkce

postup:

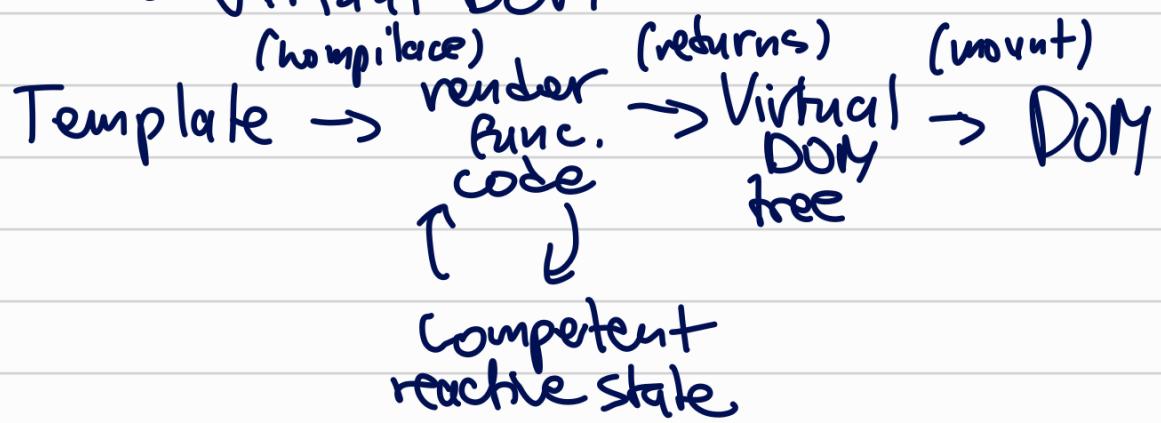
root render, serve hook (useState,...),  
state se event, spust' hook, notifikace  
(hook zavolán), zavola' zas root render,  
reconcilizace, commit (zmeny DOM)

state management



# Vue.js ~ vychází z Angularu

- Virtual DOM



## React

vs.

- client side render
- JSX forma vrt. DOM
- funkce pro hledání eventů
- update / changes přes setter z hooku (immutable)
- ↳ re-run

## Vue

- — || —
- custom HTML syntax
- reactivity
- up./ch. pres reactivity (immutable)
  - ↳ reactive updates

## Server-side render

- ↳ pro: vylejsí přístup k obsahu
- ↳ proti: setup náročnost, server zahření

- routing, UI

místo Virtual DOM ~ Selte

- za běhu optimalizuje

SolidJS ~ react bez Virtual DOM

- ↳ jednon píše JSX render
- ↳ reaktivita: signals, memo s
- optimality ne-rerender, efekty

Alpine.js ~ jQuery pro moderní web

- manipuluje s DOM, ale
- ↳ reaktivita
  - více o HTML (jQuery o JS)

(tímto konkr. verha' 5 i „6“)

## 7 - Server



### Web server

- software s hardwarem přijímající requesty přes HTTP

### Node.js

↳ JS runtime env. V8 engine  
+ npm package management system

- framework: Express
- alternative: Deno, Bun

### Application server

~ zjednodušení -

- software framework, co dává prostředky na tvorbu a hosting aplikace
- ↳ Java Server

## 8- Containers (Deployment)

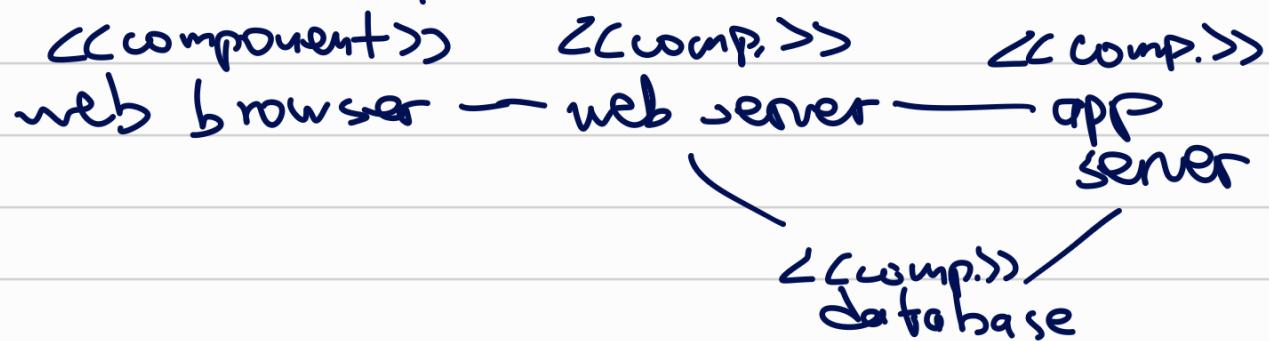
upon run build, Makefile

Ausible → automation platform  
Webhook

konfiguracií soubor → XML | YAML  
(docker compose)

Docker

↳ Dockerfile



WebAssembly

↳ assembly-like language, umožňuje rozbehnut C#, C++ a další na weby

- interface pro: WASI

↳ bezvý v a mimo browser

↳ proc?

↳ JDE PUSTIT VŠUDE

+ Docker císem?

module, memory, table, instance

# q - App. prog. interface

API

RPC

- remote procedure call

<<component>>

Client

Stub

<<component>>

Server

argumenty

odpověď

REST

- representation  
state transfer

client server

↳ stateless, cache

= architektura pro distribuované systémy  
např. GET /list /pic01  
JSON WEBP

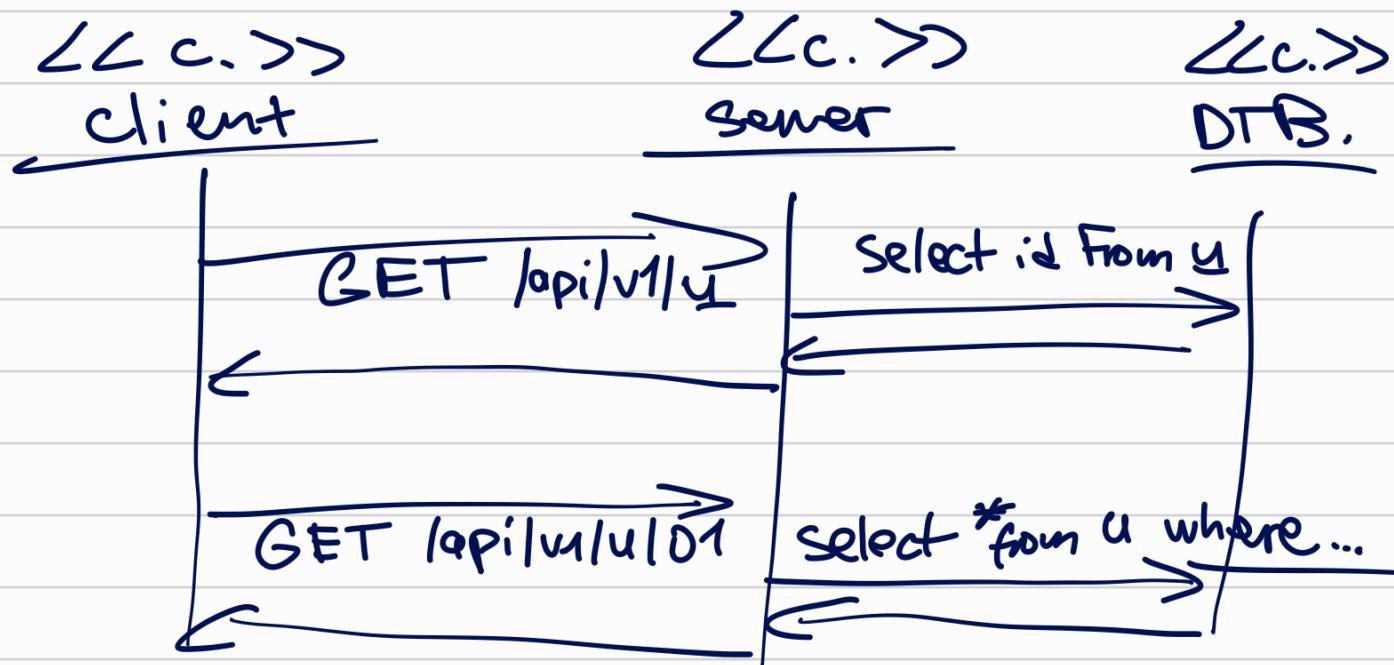
Open API ← Swagger API

- specifikacií jazyk pro YAML či JSON

GraphQL → command driven API

- vše dat jedním dotazem  
přes vše zdrojů  
(REST taky dobrý, ale to je  
vše Queries)
- staticky typované

# 10 - Server



full-stack developer

browser — web server — ~~database~~ user-service

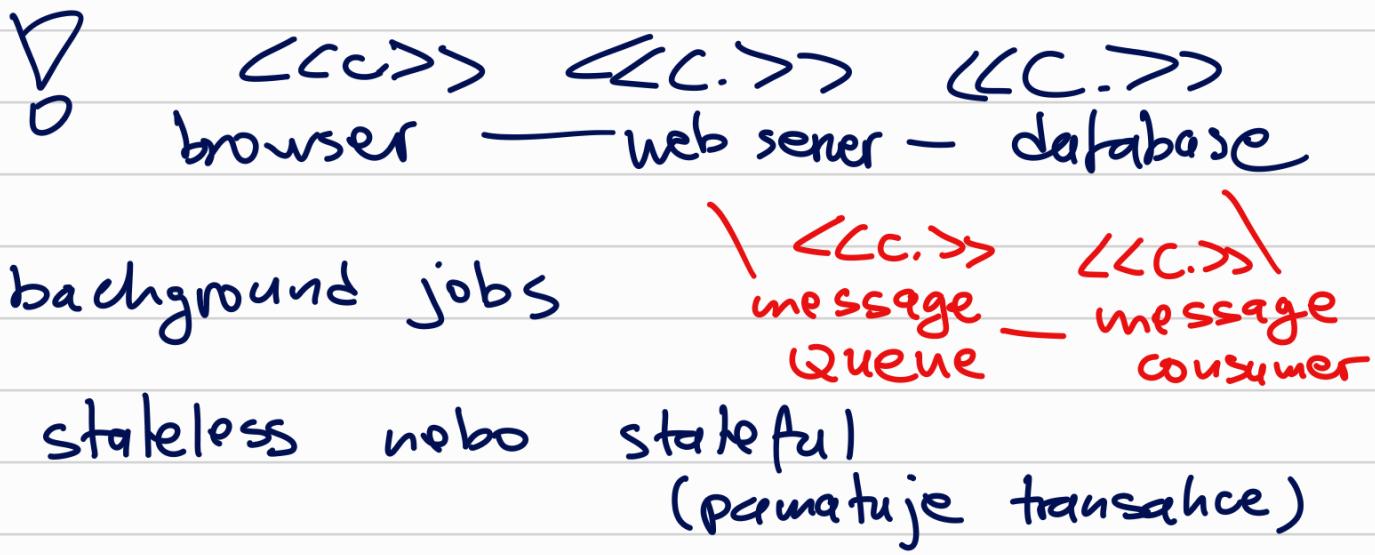
Visual framework  
napt. Angular  
Vuetify

UI f.  
React  
Vue  
Redux

Server lang.  
PHP  
NodeJS  
Java

HTTP Binding  
flash  
Express

Architektur



client - proxy - server - db.

cache ~ client vs server side  
db, RAM, web server

Content delivery network

- oddelejí, že server res'jen dynamické  
věci, CDN statické

Client-side ~ flacítha

# M - Container

- Electron
  - ↳ Node.js + webworkers
- Cordova
  - ↳ js runtime (HTML, CSS, JS..)
- Ionic
  - ↳ framework

# React Native

- ↳ style rendering
- ↳ render via VASHNECH

nativ' w'd:

# Flutter

- ↳ statefull / less

12 - a) Browser API

- visibility State

Canvas → JS drawing

WebSocket → extension HTTPS

b) DAPP

- decentralized app

↳ smart contract language  
as backends with JS frontend

↳ gaming, wallet, news without  
censorship

1. Slab typology JS? Otažky  
↳亟待解决的问题
2. generaliz. DOM  
↳ jsn staty, výroba, uhláddení
3. MVC vs MVMV
4. jsn re-deploy apply prí  
updatu ~ architektury?
5. Dependency injection PHP