

Opakování

SSPŠ
MVOP4 Webové inženýrství
2025 / 2026
Ing. Matěj Cajthaml



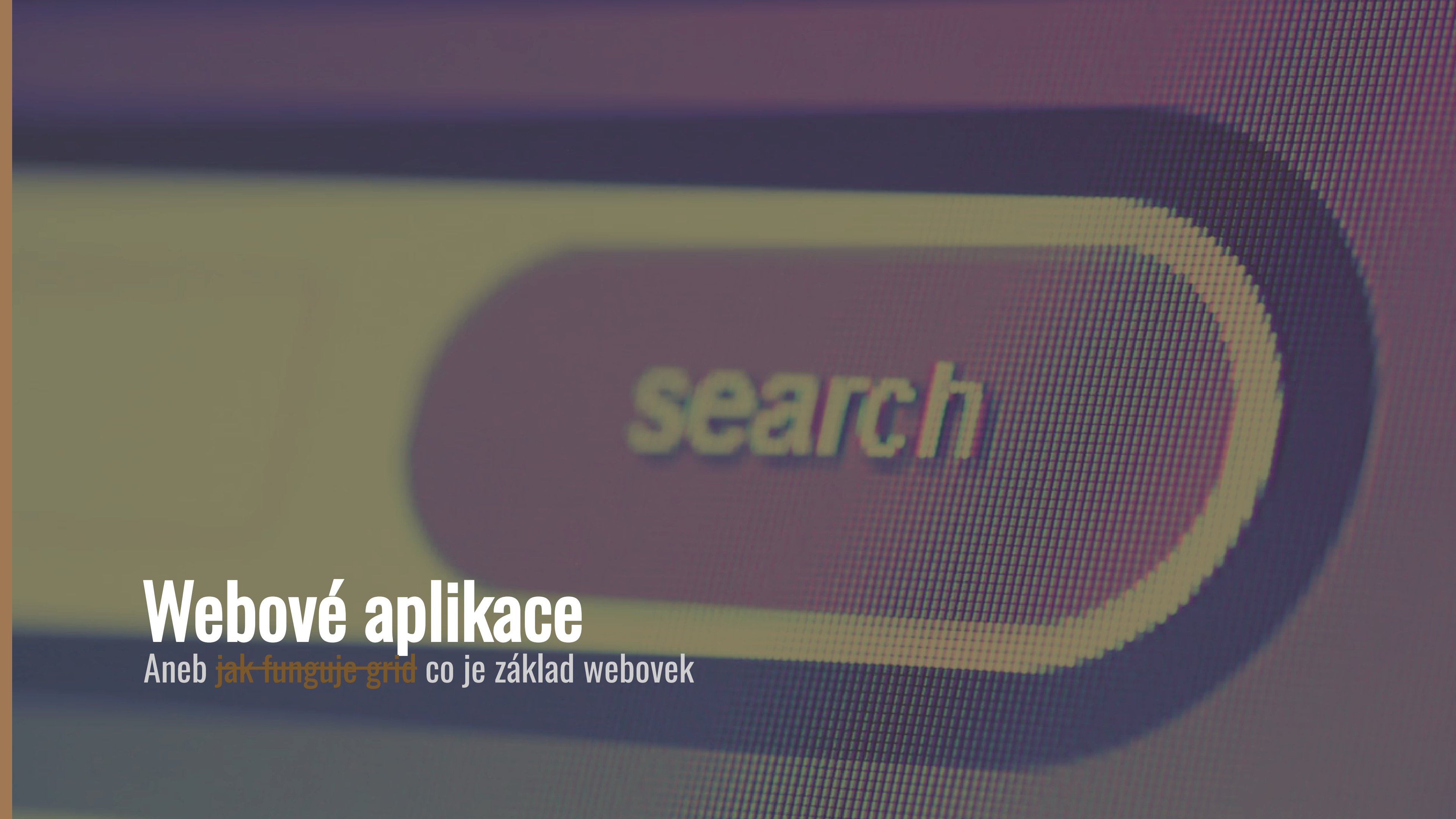
CO MÁTE DĚLAT?

Jak brát opakování?

Co při tom dělat?

Co si zapsat?

Co?



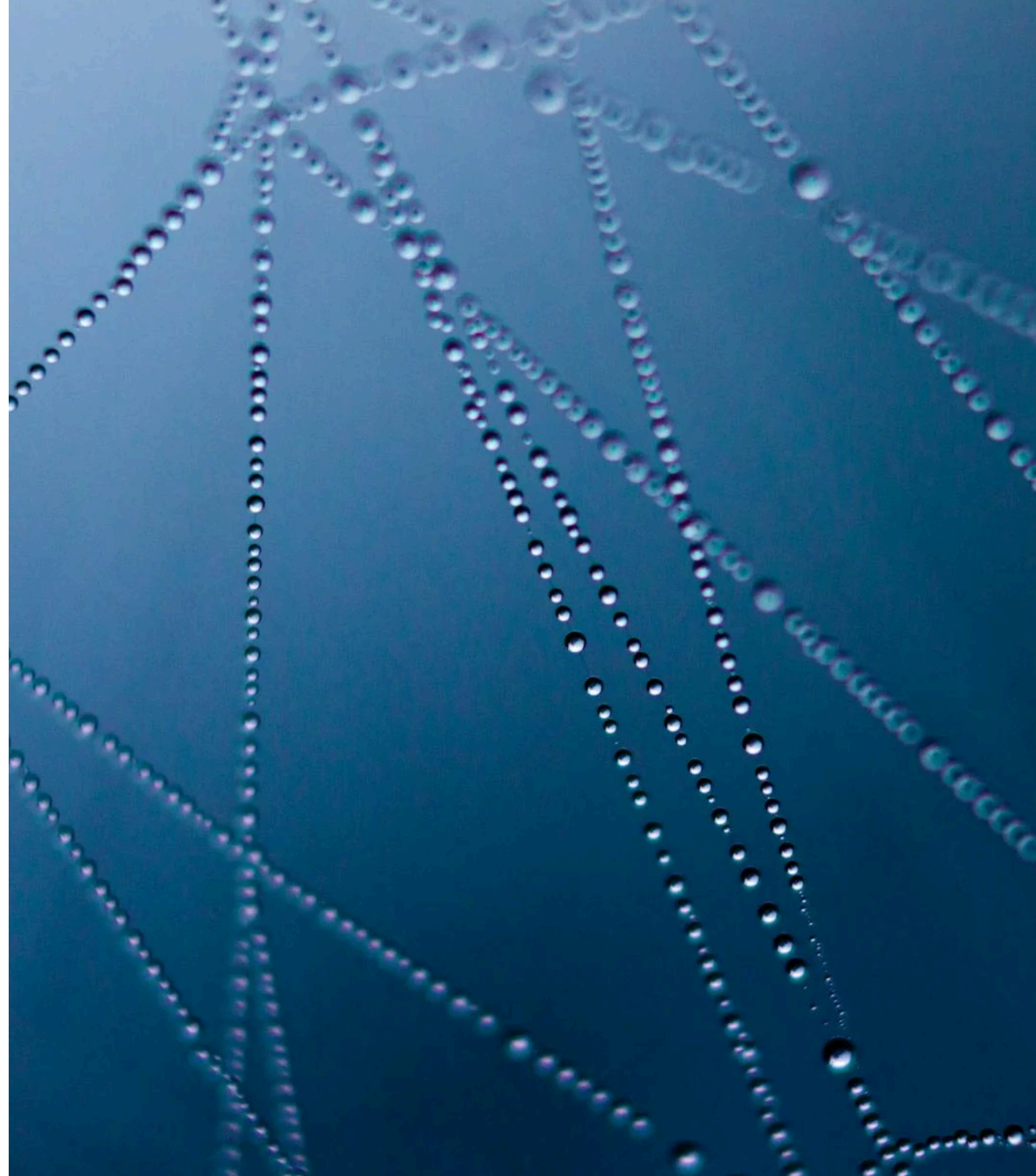
search

Webové aplikace

Aneb ~~jak funguje grid~~ co je základ webovek

Internet, WWW a další

- Internet
- World Wide Web
- Webová stránka
- Hypertextový dokument
- Hyperlink
- Doména
- URL a URI
- Webový vyhledávač
- Malware
- Antivir, antimalware
- Digitální stopa
- Webový prohlížeč
- Certifikáty



(Webové) protokoly

HTTP(S)

FTP

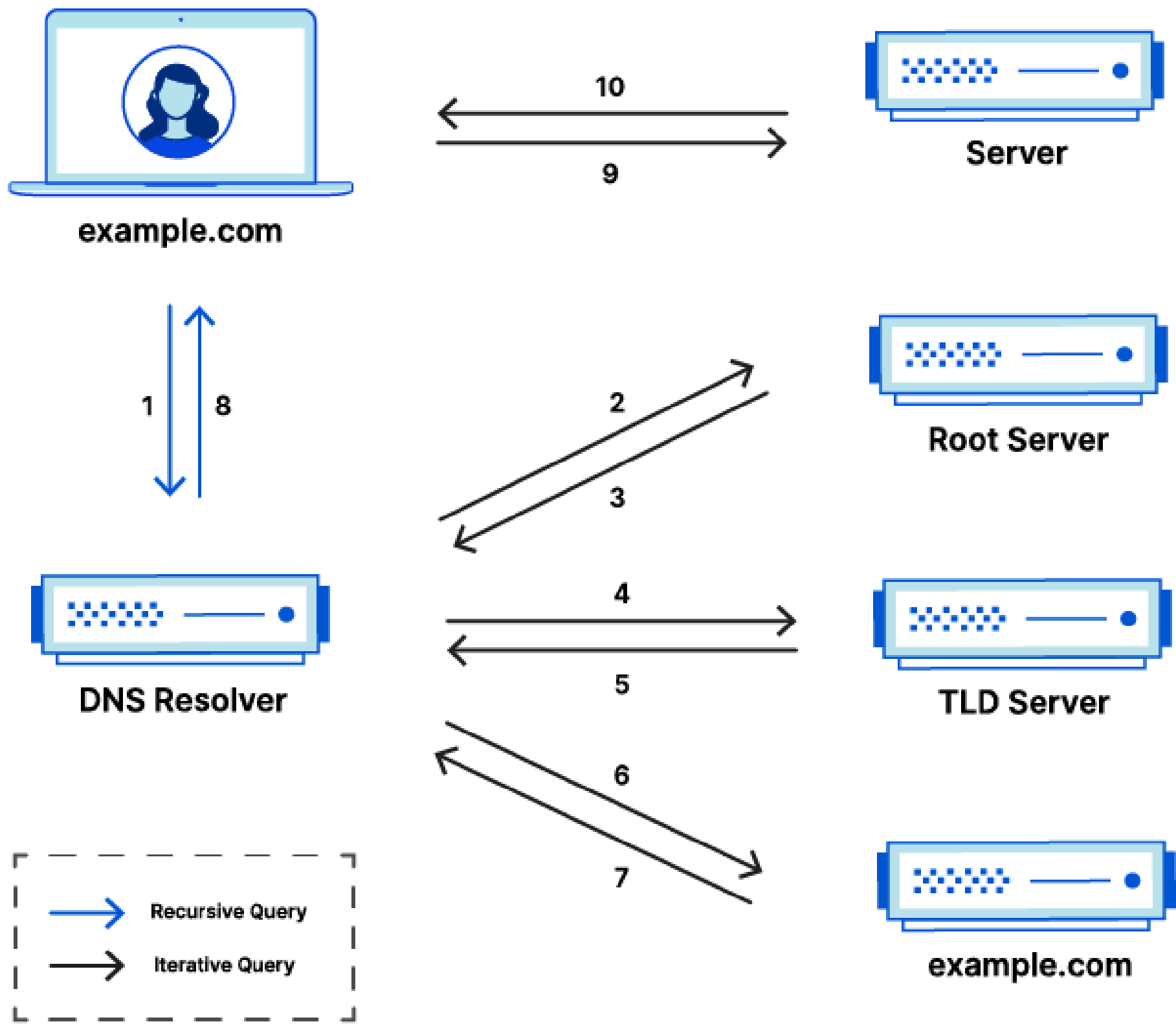
TCP/IP

DNS

SSH

WebSockets

Complete DNS Lookup and Webpage Query



HTML

- Tag, formát
- HTML5
- Sémantika a důležitost
- Neplatnost HTML
- Základní struktura
- Spojitost s XML

HTML tagy

- `html, head, body,`
- `title, link, meta, style`
- `article, section, nav, aside,`
- `header, footer, main`
- `p, pre, h1-h6`
- `span, div, hr`
- `b, u, i, s, a`
- `strong, small, q, br`
- `ins, del`
- `ol, ul, dl, li, dt, dd,`
- `img,`
- `video, audio,`
- `table, thead, tbody, tfoot`
- `caption, tr, td, th,`
- `form, label`
- `input[type=`
`text, hidden, email,`
`password, range, number,`
`file, checkbox]`
- `select, option, button,`
`textarea`
- `script`

CSS

- Význam
- Pravidlo
- Selektor
- Vlastnost
- Hodnota
- Kaskáda
- Specifita selektoru

Tagy (typové selektory) a pseudoelementy (specifita: 0-0-0-1)

`div { ... }, ::before, ::after`

Třídy, atributové selektory, pseudotřídy (specifita: 0-0-1-0)

`.class { ... }, [type="text"], :hover`

ID selektory (specifita: 0-1-0-0)

`#id { ... }`

Inline styly (specifita: 1-0-0-0)

`<div style="color:red">`

!important

přebíjí vše ostatní, bez ohledu na specifitu



Typy zobrazení

- Blokové
- Řádkové
- Blokově-řádkové
- Flex
- Grid
- Vlastnost `display`

display: inline

A

B – dlouuuuuuhý text, aby bylo vidět zalamování

C

display: block

Prvek 1 (zabere celý řádek a další jde pod něj)

Prvek 2

Prvek 3

display: inline-block

Box 1
200×100 px

Box 2
200×100 px

Box 3
200×100 px

Vlastnosti

- display, width, height
- padding, margin, border
- outline, box-shadow, background
- overflow, box-sizing,
- aspect-ratio
- color, text-shadow
- font-weight, font-family
- text-align, text-transform,
- text-decoration, font-size
- line-height
- letter-spacing, word-spacing
- animation, transition
- position
- top, left, right, bottom,
- content
- cursor, visibility, opacity
- filter, transform, scale
- z-index

Jednotky

- relativní vs. absolutní
- číslo
- px, rem, em, %
- ch, vh, vw
- cm, mm, in, pc, pt
- deg, turn, rad,
- s, ms

Funkce

- calc, minmax, clamp, min, max
- repeat
- translate, rotate, scale, skew
- blur, brightness, contrast, grayscale
- hue-rotate, invert, saturate
- rgb, hsl, color-mix
- light-dark
- linear-gradient, radial-gradient
- url
- linear, cubic-bezier, steps

Selektory, at-pravidla a pseudotřídy

- *
- tag
- tag > tag
- tag tag
- tag, tag
- tag ~ tag
- tag + tag
- .tag
- #tag
- tag[attr]
- @import
- @keyframes
- @media
- @font-face
- :checked
- :link, :visited,
- :root
- :nth-child, :last-child, :first-child
- :nth-of-type, :last-of-type, ...
- :hover, :active, :focus
- :is(), :not(), :has()

PRÁCE

Vymyslete a vytvořte stránku (proof of concept), ve které rozumně využijete selektoru sourozence.

Pseudoelementy a proměnné

- `::first-line`
 - `::first-letter`
 - `::selection`
 - `::before`
 - `::after`
 - `::marker`
-
- `:root`
 - `--name: value`
 - `var(--name)`
 - `var(--name, def)`

Ukázka pseudoelementu

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <title>A</title>
6      <style>
7          * {
8              box-sizing: border-box;
9              padding: 0;
10             margin: 0;
11         }
12
13         body {
14             height: 100vh;
15             font-size: 5em;
```



Title

Rozložení

FLEX

- `display: flex`
- `flex-direction`
- `justify-content`
- `align-items`
- `flex-wrap`
- `flex-shrink`
- `flex-grow`
- `flex-basis`
- `gap`
- `order`
- `align-self`

GRID

- `display: grid`
- `grid-template-columns`
- `grid-template-rows`
- `gap, row-gap, column-gap`
- `align-items`
- `align-content`
- `justify-content`
- `justify-items`
- `grid-column`
- `grid-row`

Soubory

- Index soubor
- .css, .js, .html
- ./ relativní cesty
- / absolutní cesty
- .. rodičovská cesta



JE TO VALIDNÍ ZÁPIS RESPONZIVITY?

@media (min-width: 300px)

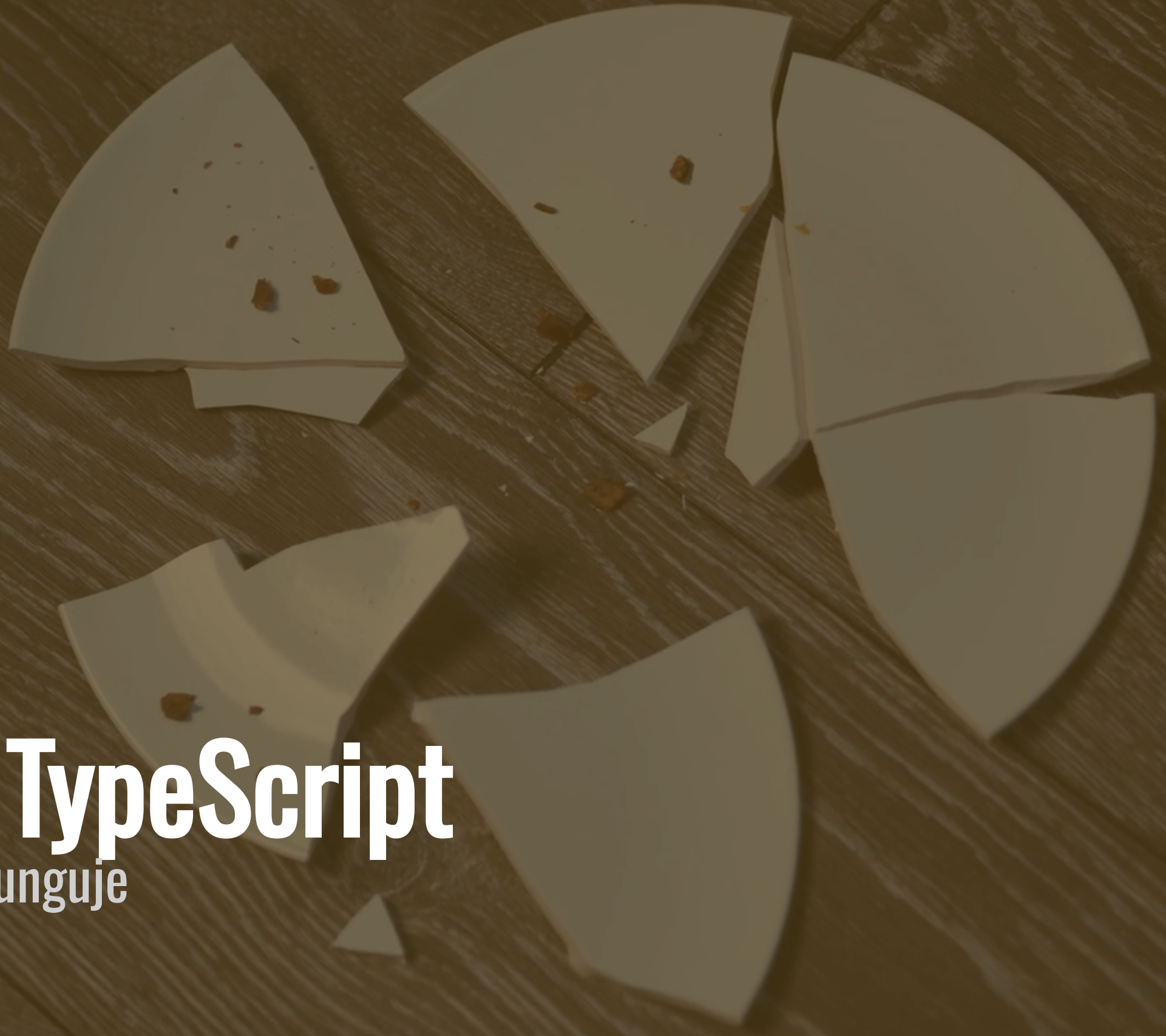
@media (print)

@media (height: 300px)

@media (min-width: 200px) and (min-height: 300px)

@media (width <= 300px <= 600px)

@media only screen and (max-width:420px) and
(orientation:portrait) and (max-aspect-ratio: 9/16)

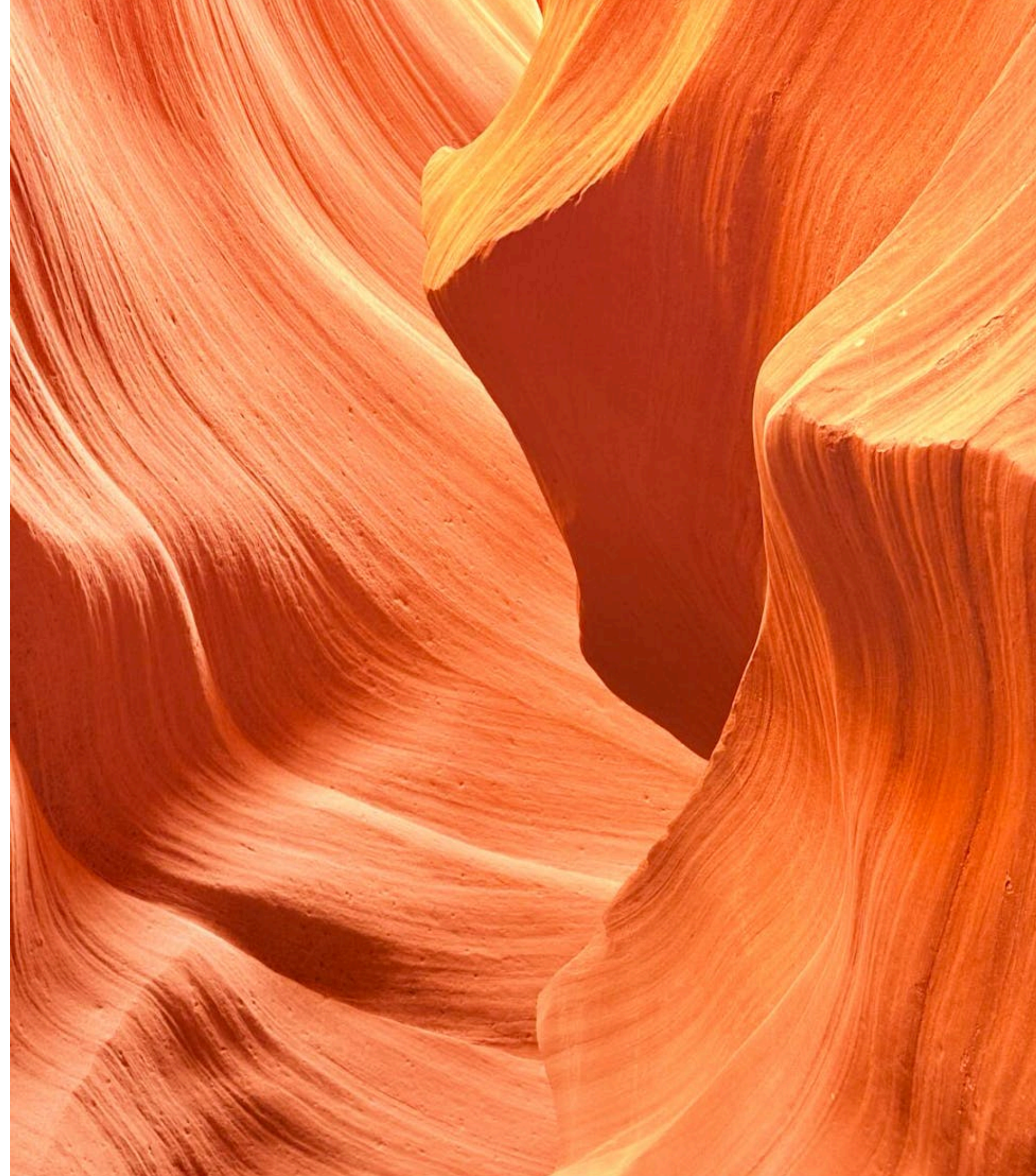


JavaScript a TypeScript

Aneb proč to všechno **ne**funguje

JavaScript

- Skriptovací jazyk
- Imperativní jazyk
- Objektově orientovaný jazyk
- Zejména pro webové stránky
- Existují runtime => Node.js, Deno, Bun
- Programovací jazyk
- Dynamicky typovaný jazyk
- Velmi limitující počet typů
- Porovnávání a striktní porovnání
- DOM



Datové typy

- number
 - string
 - boolean
 - bigint
 - null
 - undefined
 - (symbol)
-
- object
 - function

```
1  const n1 = 3;  
2  const n2 = 3.31;  
3  const n3 = Infinity;  
4  const n4 = NaN;  
5  const s1 = "Ahoj";  
6  const s2 = 'Ahoj';  
7  const s3 = `Ahoj x\${n4}  
8  ;)`;  
9  const o1 = null;  
10 const o2 = undefined;
```

Struktury

- Objekt
- Pole
- Set
- Map

```
1  const array = [3];
2
3  const object = { name: "Test" };
4
5  const set = new Set();
6  set.add(3);
7  set.add(3);
8
9  const map = new Map();
10 map.set("321123", "Yoko");
11 map.set("1923", "Bambi");
```

Konstrukty

- Proměnné: `let`, `const`, `var`
- Podmínka: `if`, `else if`, `else`
- Cykly: `for`, `while`, `for..in`, `for..of`, `continue`, `break`
- Aritmetické operátory
- Logické operátory
- Přiřazovací operátory

```
1 3 + 2;  
2 3 - 2;  
3 3 / 2;  
4 3 * 2;  
5 3 % 2;  
6 3 ** 2;  
7 "3" + "2";  
8 true && false;  
9 true || false;  
10 1 ^ 3;  
11 1 | 3;  
12 1 & 2;  
13 1 >> 3;  
14 i++;  
15 ++i;  
16 3 == 2;
```

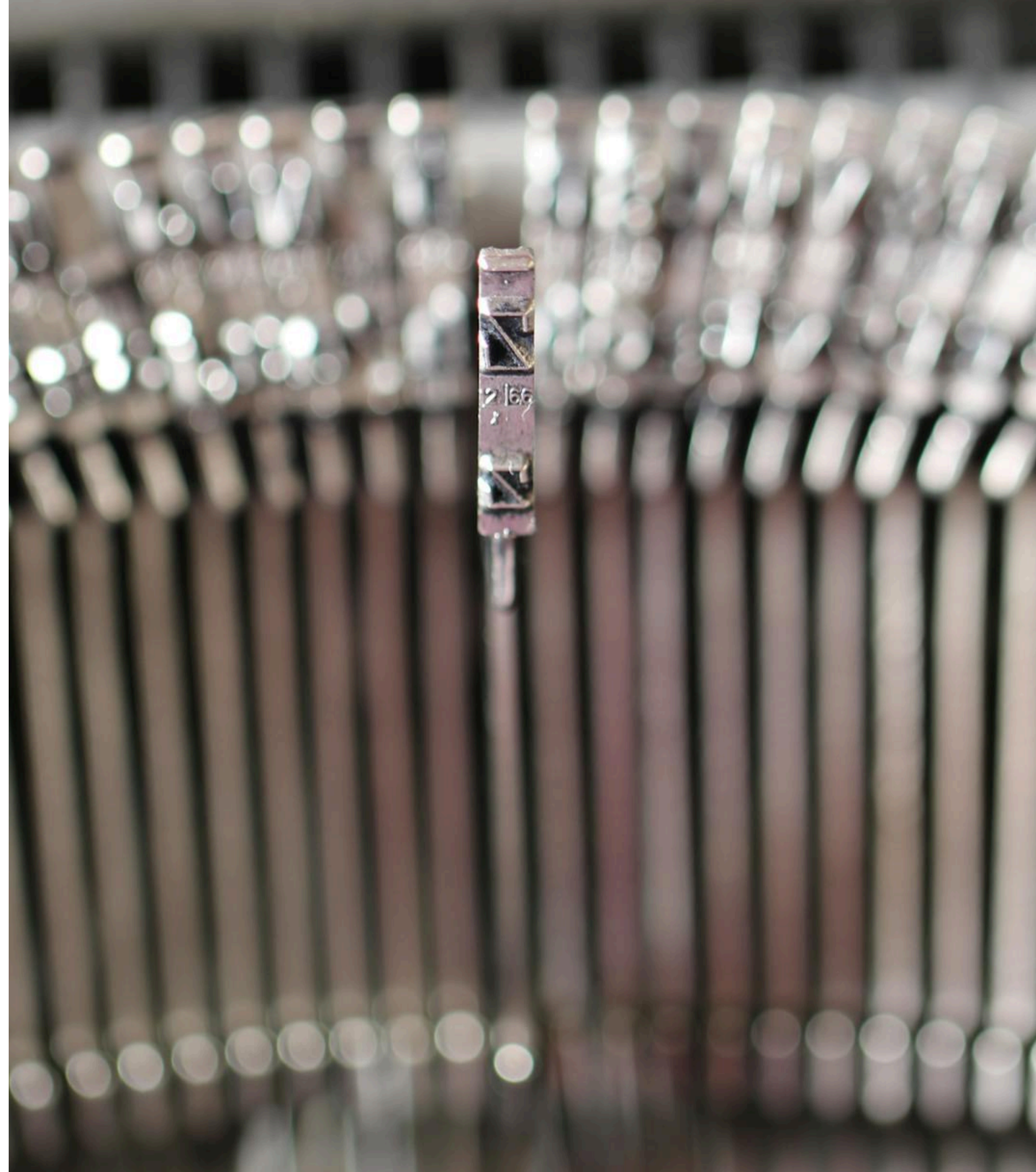
Funkce

- Vstup => výstup
- Nemusí mít vstup a ani výstup
- Klíčové slovo return
- Kontext dle this
- Deklarovaná => hoistovaná, this podle volání – window, objekt
- Anonymní => this stejně jako u deklarované
- Arrow => this dle kontextu

```
1 function a (arg) {  
2     console.log("a");  
3 };  
4  
5 const b = function (arg) {  
6     console.log("b");  
7 };  
8  
9 const c = (arg) => {  
10     console.log("c");  
11 };
```

Paradigma

- Způsob struktury a organizace programu
- Způsob programování



Objektově orientované programování

- Používá objekty k posouzení reálných objektů
- Třídy jsou šablony
- Instance (objekt) je přiřazen ke konkrétní třídě a obsahuje svá data
- Metody modifikují instance
- Klíčové slovo `this`
- Konstruktory

```
1 class AuthService {  
2     users;  
3  
4     constructor() {  
5         this.users = [];  
6     }  
7  
8     addUser(name) {  
9         this.users.push(name);  
10    }  
11 }
```

Přístupové modifikátory

- `static` - ukládá se na třídě
- `#` - pouze uvnitř třídy
- Vkládají se na metody a vlastnosti

```
1 class AuthService {
2     static KEY = "AUTH";
3     #users;
4
5     constructor() {
6         const data = localStorage.getItem(KEY);
7         this.#users = data ? JSON.parse(data) : [];
8     }
9
10    addUser(name) {
11        this.#users.push(name);
12        localStorage.setItem(KEY, JSON.stringify(this.#users));
13    }
14 }
```

Dědičnost a polymorfismus

- Zjednodušení kódu
- Dědí se metody a vlastnosti
- Problém s konstruktory, opakované volání a proč
- Omezení dědění na jednu nadtřidu
- Polymorfismus dovoluje abstraktně přistupovat k volání různých metod pod stejným jménem

```
1 class Block {
2     x, y;
3
4     constructor(x, y) {
5         this.x = x;
6         this.y = y;
7     }
8
9     move(x, y) {
10         this.x = x;
11         this.y = y;
12     }
13 }
14
15 class WatermarkBlock extends Block {
```

Asynchronní programování

- Neplést s vícevláknovým programováním
- Dovoluje přepínat kontext, kdy by program musel čekat
- Často I/O operace, čekání na čas
- Často v UI
- V JS pomocí slibů
- Vázání pomocí then a catch
- Syntax sugar pomocí async a await

```
1  const promise = new Promise((res, rej)
2    setTimeout(() => res("ahoj"), 3000
3  ));
4
5  promise.then((val) => console.log(val))
6
7  const getUsers = async () => {
8    const response = await fetch("api/
9
10   const json = await response.json()
11
12   console.log(json);
13 }
```

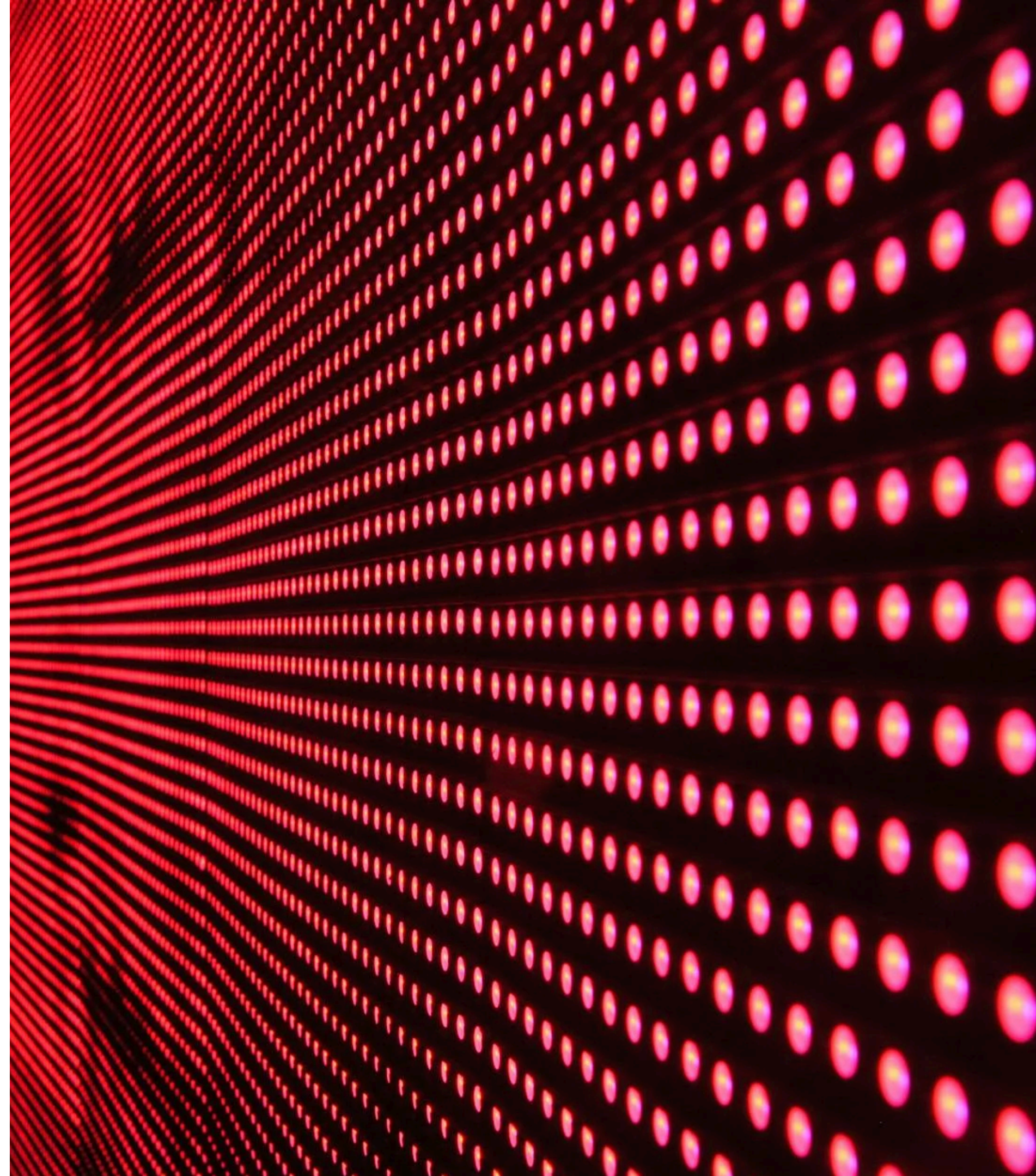

Výjimky

- Nastávají když se program dostane do nečekané situace
- Často zneužívané na předávání jakýchkoliv chyb
- Možné zachytit – try catch
- Vlastní výjimky – new Error()
- Hodně drahé / pomalé

```
1 // `getUser` z předešlého slidu
2
3 const load = async () => {
4     try {
5         const user = await getUser();
6         // TODO
7     } catch (e) {
8         sentry.log(e);
9     }
10 }
```


TypeScript

- Snaha udělat JS staticky typovaným
- Přidává typy přímo do kódu
- Nutná kompilace do JS
- Přidává různé další funkcionality, _
zejména do OOP návrhu
- Abstraktní třídy
- Modifikátory přístupu
- Rozhraní
- Generiky
- A další



Typování a typy

- Implicitní typování
- Výčtový typ

```
1  const a: string = "text";
2  const b = "text" as string;
3
4  type Rank = "OWNER" | "CUSTOMER";
5
6  const rank: Rank = "OWNER";
7
8  type Arr = string[];
9  type Complex = {
10     salary: Number,
11     ranks: Rank[],
12     user: User | undefined
13 }
```

Typ třídy a rozhraní

```
1 interface Copyable<T> {
2     copy(): T;
3 }
4
5 class Block implements Copyable<Block> {
6     private x: number;
7     private y: number;
8
9     constructor(x: number, y: number) {
10         this.x = x;
11         this.y = y;
12     }
13
14     public copy() {
15         return new Block(this.x, this.y);
16     }
17 }
```


A close-up, sepia-toned photograph of a mechanical watch movement. The image shows intricate gears, a balance wheel, and a portion of the watch dial with Roman numerals. The lighting is dramatic, highlighting the metallic textures and the complex arrangement of the internal components.

Design

Aneb jak udělat *stránku opravdu* pěknou a krásnou a **ÚŽASNOU** a
ZAPAMATOVATELNOU

FE, BE, UI a UX

- Frontend
- Backend
- Full-stack
- User Interface
- User Experience
- Nutné řešit vše dohromady!



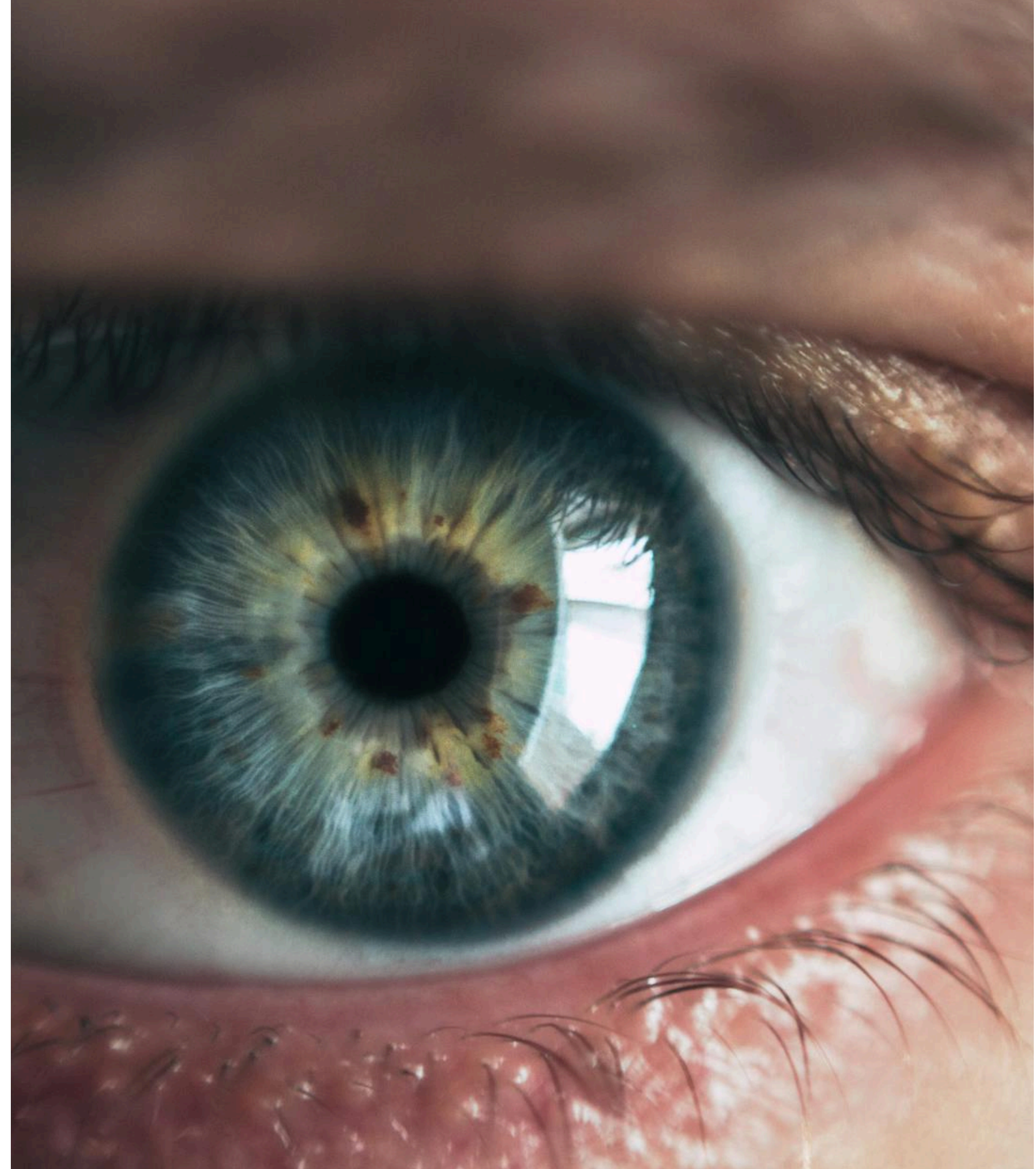
Co hlídat

- Zásady
- Princip minimálního překvapení
- Konzistence
- Přívětivost
- Stručnost
- Zpětná vazba
- Standard
- Příjemnost
- Efektivita
- Očekavatelnost



Jak hlídat

- Testování
- Sledování požadavků
- Sledování uživatelů
- Nutnost informování
- Uživatelské testování



Psychologie

- Design
- Barvy
- Font
- Typografie
- Obrázky a multimédia
- Místo



Tvůj no

YOU - Tvůj person

ohli nabídnout person

om rádi ukládali anebo načítali informace o vašem zařízení (např. adresu IP, ID uživatele, informace o prohlizeči, identifikatory zařízení) a používají

následujícím účelům: pro identifikaci na webových stránkách třetích stran (také s využitím pseudonymizované e-mailové adresy a telefonního

ování osobně přizpůsobených reklam s přizpůsobeným obsahem, zde i na stránkách třetích stran ("personalizované reklamy a obsah"), pro měření

am a obsahu, k lepšímu porozumění vašich potřeb ("průzkum trhu") a abychom pro vás mohli vyvíjet ještě lepší produkty a služby ("vývoj

od s tím nesouhlasíte (klikněte na "Odmítnout"), omezíme se na následující činnosti a to na soubory cookie a technologie, které jsou nezbytně nutné

spolehlivému používání našich stránek a to ve formě, kterou preferujete ("výkon") – bohužel vám ale nemůžeme v této variantě nabídnout plný,

uživatelský zážitek na ABOUT YOU. Svůj souhlas můžete kdykoliv s účinností do budoucna odvolat kliknutím na "Nastavení". [Informace o ochraně](#)

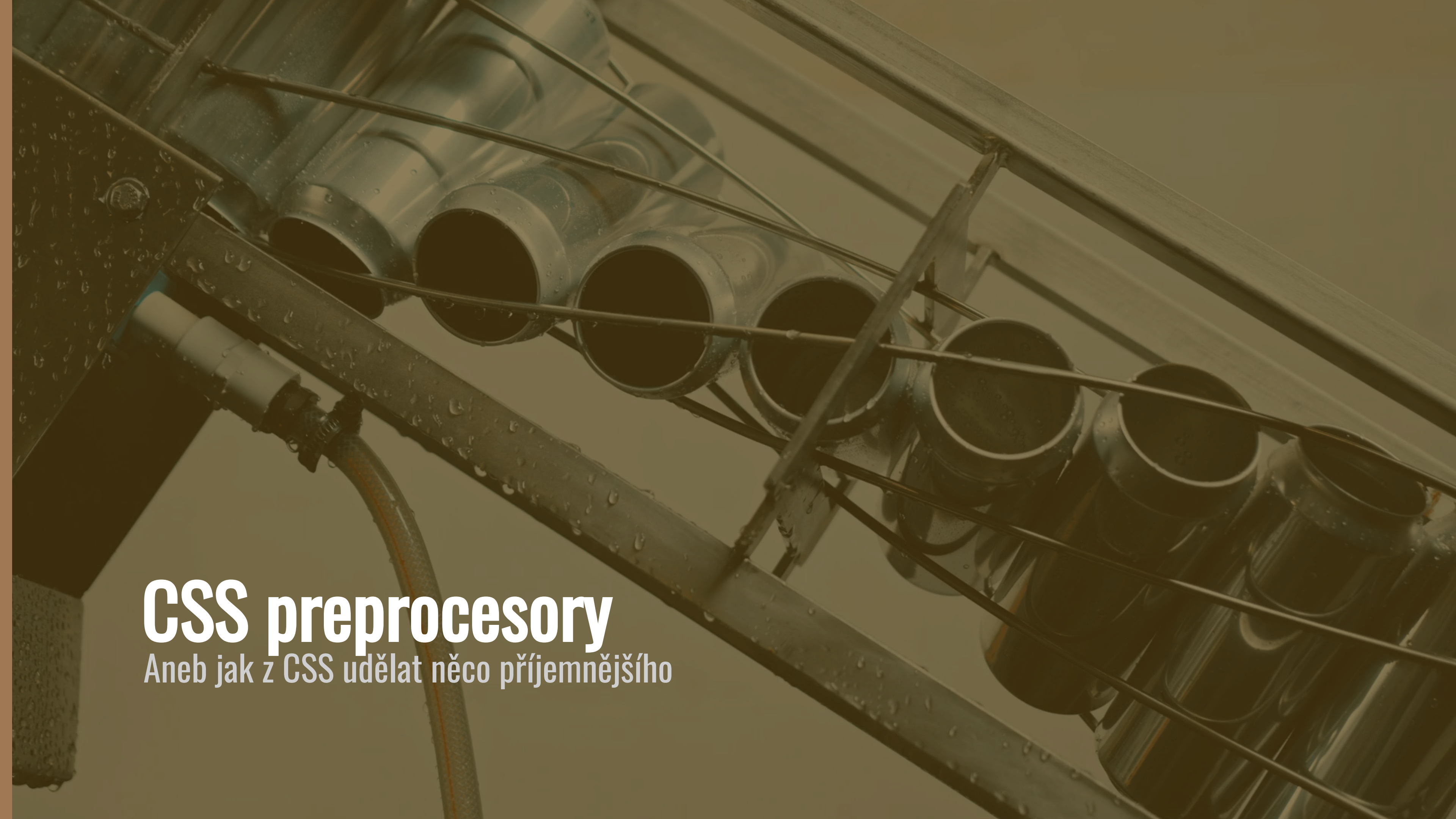
Ok

Zamítnout

[Nastavení](#)

PRÁCE

Prozkoumejme a zhodnoťme stránku
internetového obchodu s oblečením
aboutyou.cz

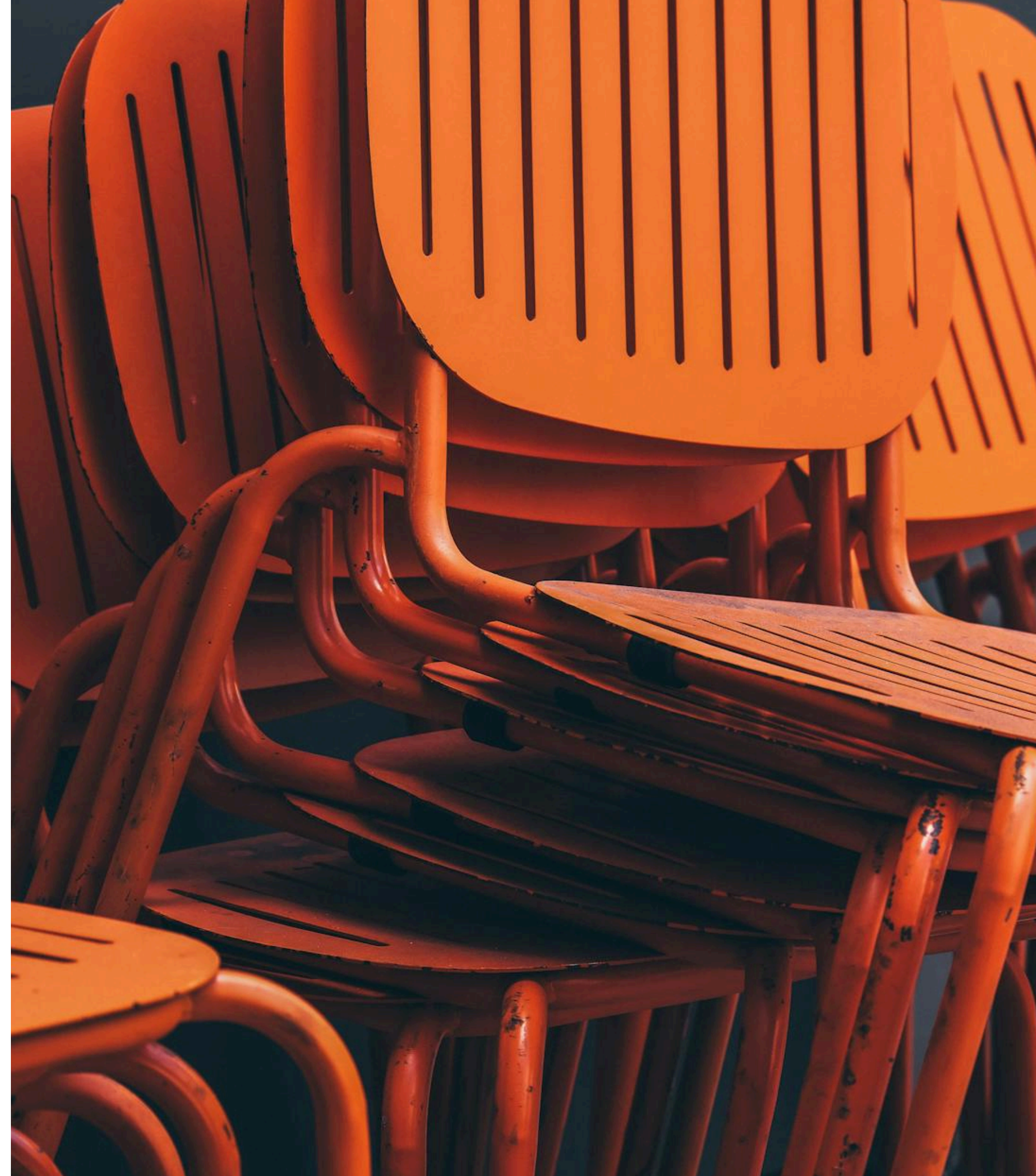


CSS preprocesory

Aneb jak z CSS udělat něco příjemnějšího

CSS preprocessory

- Vstup je nějaký syntax
- Výstupem je CSS
- Kompilátor
- Podobně jako TS
- Nejznámější: SASS, Less
- Přidávají možnost generace kódu



Novinky CSS

- Proměnné (~ 2018)
- Vnořování (~ 2023)
- Color mixing (~ 2022)
- Další mat. funkce (~ 2020)
- Funkce a mixiny (~ 2026)



OTÁZKA

Když spoustu věcí již CSS zahrnuje, proč přesto
preprocesory používáme?

SASS

- Formát SCSS a SASS
- SCSS -> rozšiřuje CSS
- SASS -> spíše "Python" zápis
- SASS -> CSS
- Proměnné, vnořování, funkce
- Funkce, mixiny, moduly

```
1 $primary-color: #2a7ae2;
2 $secondary-color: #e26f2a;
3 $font-stack: 'Inter', sans-serif;
4
5 $spacing-unit: 1rem;
6 $border-radius: 0.5rem;
7
8 @function theme-shade($color, $amount) {
9   @if ($amount > 0) {
10     @return lighten($color, $amount);
11   } @else {
12     @return darken($color, abs($amount));
13   }
14 }
15
16 @mixin respond($breakpoint) {
```


OTÁZKA

Proč a kde používat SASS?



Verzování

Aneb jak se naučit používat čtyři příkazy

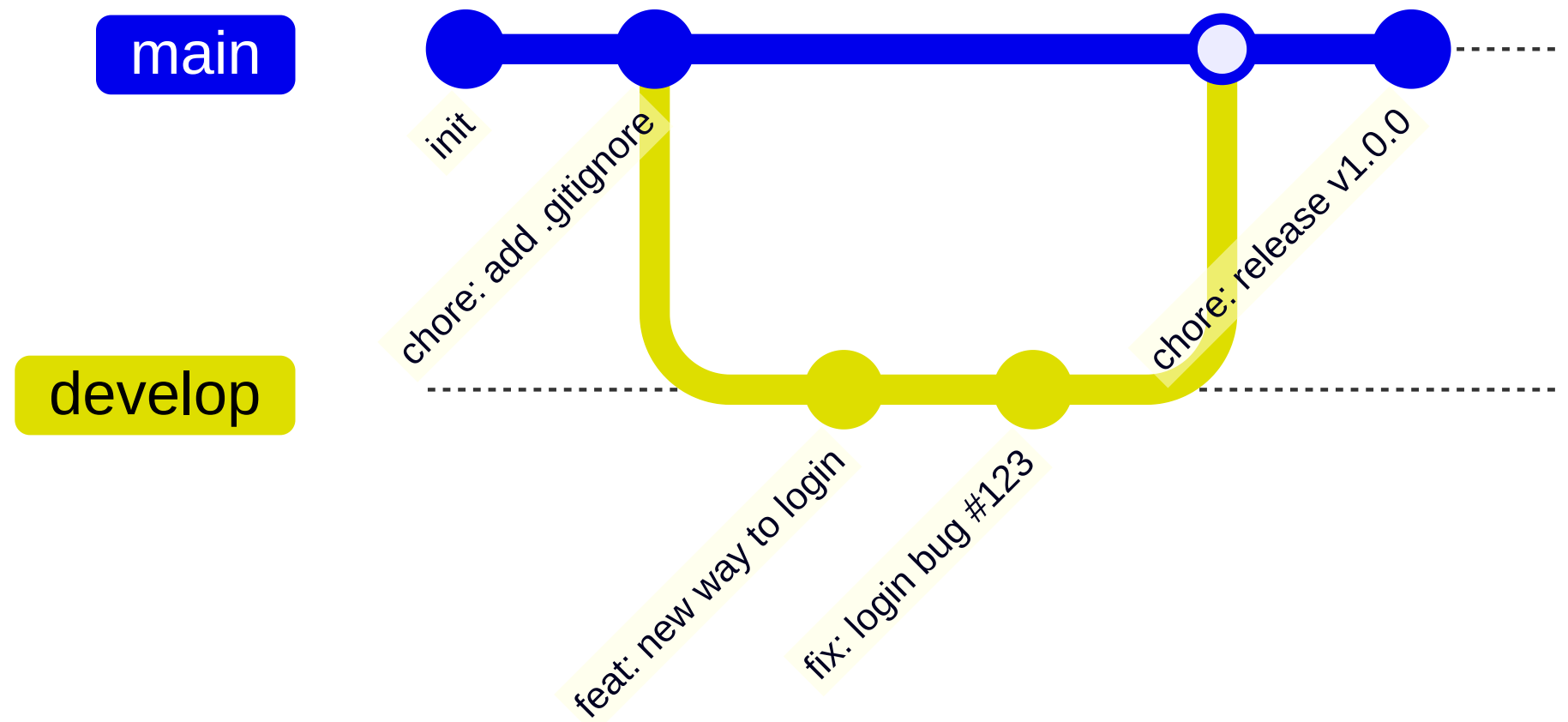
Verzování

- Revize, verze
- Rychlé vrácení, zjištění příčiny, historie
- Merge vs. locking



Git

- Distribuovaný systém
- Repozitář
- Commit
- Větev
- Gitové služby
- Soubor .gitignore
- Spolupráce => merge conflict



The background is a dark purple gradient with a complex network of thin, light purple lines connecting various nodes. The nodes are represented by small icons, including envelope symbols, '@' symbols, and circular icons with a plus sign. Some nodes are highlighted with a soft, glowing purple aura. The overall effect is a sense of interconnectedness and digital communication.

Vue

Aneb jak vytvořit velkou aplikaci, u které neuplavou vlasy

Vue

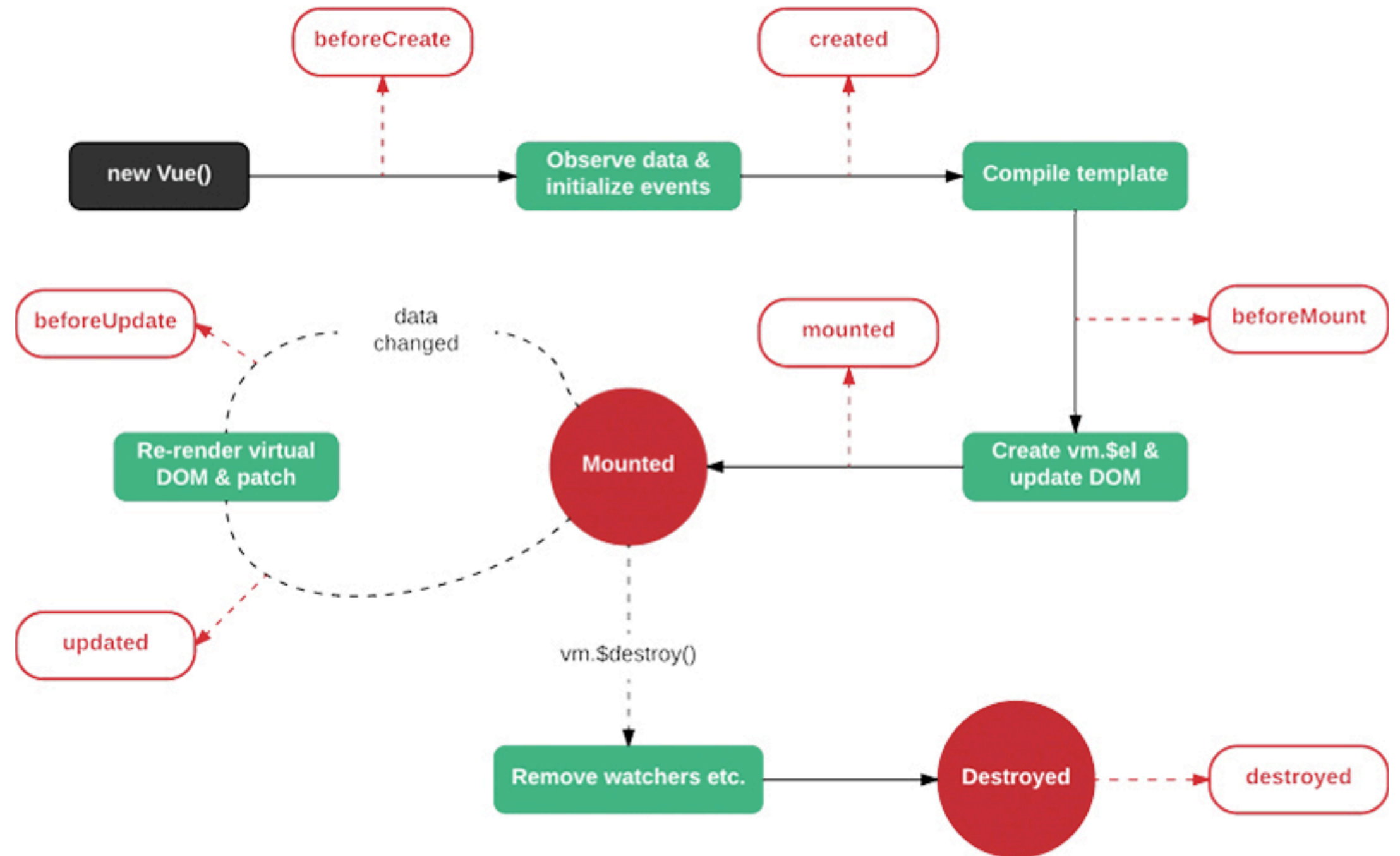
- Webový framework
- Virtuální DOM
- Komponenty
- Direktivy
- Projekty, rozložení složek
- Options API vs. Composition API

Direktivy

- `v-bind / :`
- `v-model`
- `v-on / @`
- `v-if, v-else, v-else-if`
- `v-for`
- `v-slot`
- `v-html`
- `v-text / {{{}}`

Komponenta

- Data / ref
- Computed
- Methods / functions
- Watch
- Props
- Components / import
- Životní cyklus
- Události



Knihovny ve Vue

SASS

- Instalace knihovny
- `lang="scss"`

PINIA

- Globální umístění dat
- Store
- Kompozice

ROUTER

- Vícestránkové aplikace
- `router.js`
- Jednotlivé routy
- Jmenné routy
- Vnořování
- `router-link`, `router-view`
- Způsoby směrování

Kompilace

- Po sestavení se očekává kořenový adresář
- Případně nutné změnit složku (base) v nastavení Vite
- Složka `public/` se pouze kopíruje
- `assets/` dle referencí v kódu
- Při módu web history je nutné nastavit server na přesměrování na `index.html`



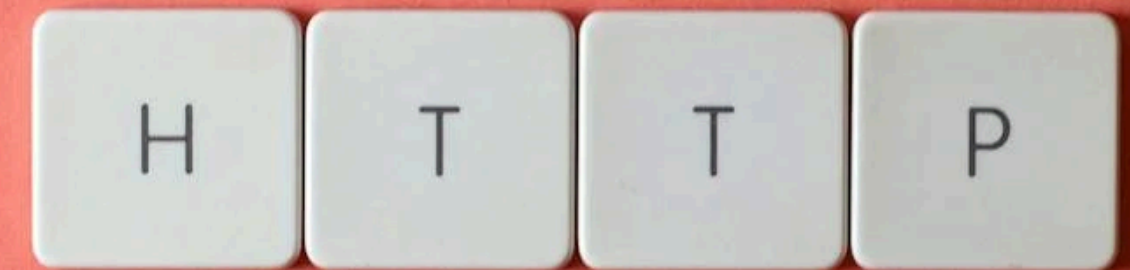
A low-angle photograph of a person in a forest, looking up at tall, thin trees. The image has a warm, brownish tint. The person is in the foreground, slightly out of focus, looking towards the right. The trees are tall and thin, reaching towards the top of the frame.

Backend

Aneb jak bezpečně předávat data

HTTP

- Komunikace na WWW
- Dnes již na cokoliv
- Textový formát
- TCP
- Dělí se na informace, hlavičky a tělo
- HTTP/3 a budoucnost



HTTP

POŽADAVEK

- Žádost o něco
 - Metoda
 - Cíl (ne celá URL!)
-
- HEAD, OPTIONS
 - GET, POST, PUT
 - PATCH, DELETE

ODPOVĚĎ

- Odpověď na žádost
 - Status kód a text
 - Odpověď
-
- 1XX - informační
 - 2XX - úspěch
 - 3XX - další akce
 - 4XX - chyba klienta
 - 5XX - chyba serveru

Nejdůležitější stavové kódy

- 200 – OK
- 201 – Created
- 301 – Moved permanently
- 400 – Bad Request
- 401 – Unauthorized
- 404 – Not found
- 418 – I am a teapot
- 500 – Internal server error

Ukázky komunikace

```
1 GET /user HTTP/1
2 Authorization: Bearer 312
3 Accepts: */*
```

```
1 HTTP/1 200 OK
2 Content-Type: application/json
3
4 {"name": "matej"}
```

```
1 POST /user HTTP/1
2 Authorization: Bearer 312
3
4 {"name": "kocek"}
```

```
1 HTTP/1 201 Created
2 Content-Type: application/json
3
4
```


Server / Backend

- Fyzický server – HW
- Webový server – komunikuje pomocí HTTP
- Backend – komplexní systém včetně komunikace (ne nutně HTTP)



API

- Jasná definice rozhraní
- Různé způsoby a protokoly komunikace
- Např. v GPU, komponenty, ale i web
- Pro web definujeme endpointy
- Endpoint = metoda a část url
- Dokumentace a přehlednost



Formáty

JSON

- Objekty v JS
- Omezené datové typy

XML

- Superset HTML
- Nedefinuje skoro žádné tagy

AJAX

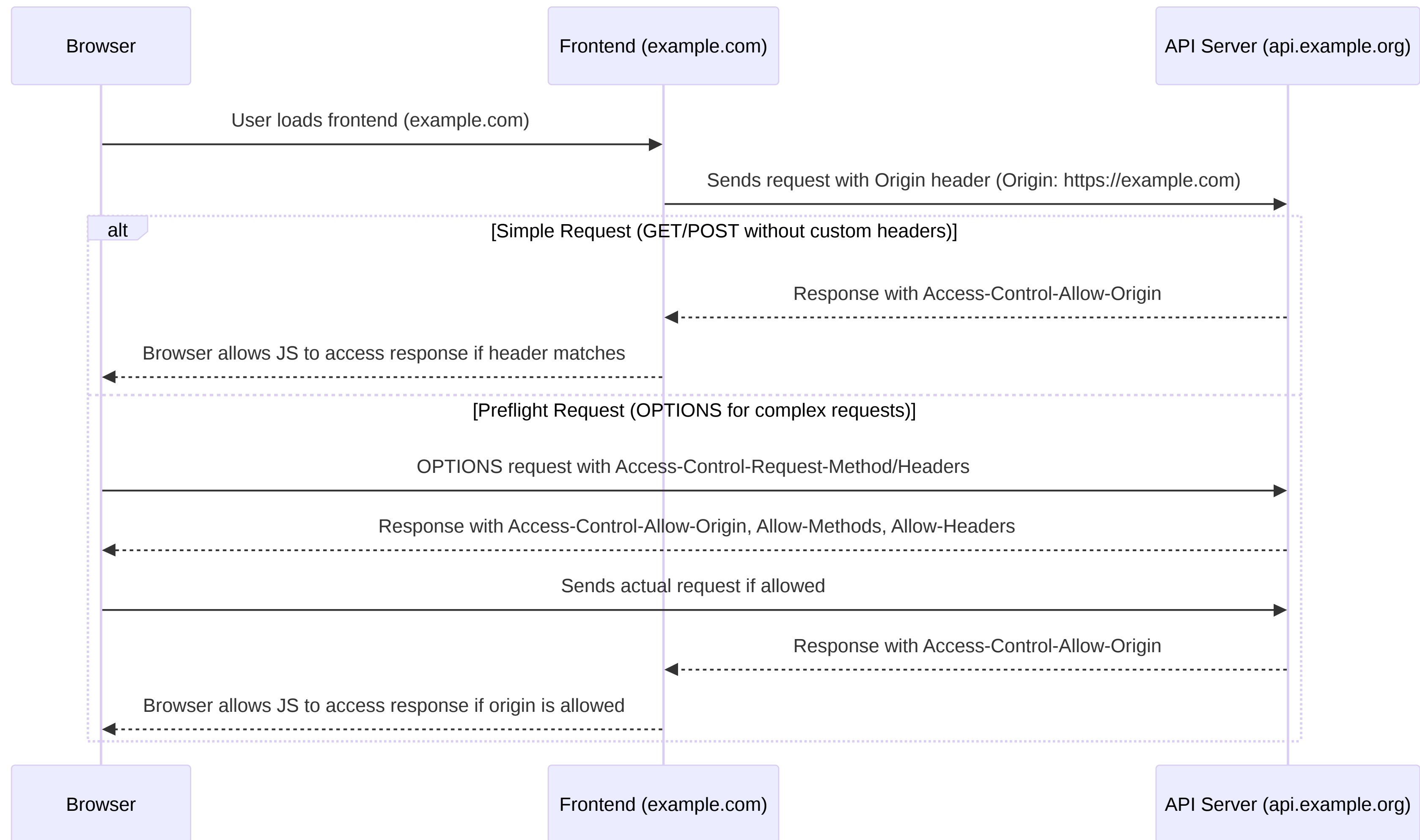
- Konkurence volání požadavků na webu
- Dovoluje vykreslovat UI během toho, co se něco děje



CORS

- Forma zabezpečení uživatele
- Resp. jeho dat
- Zamezení volání stránek ze zdroje, který není povolen
- Kontroluje prohlížeč





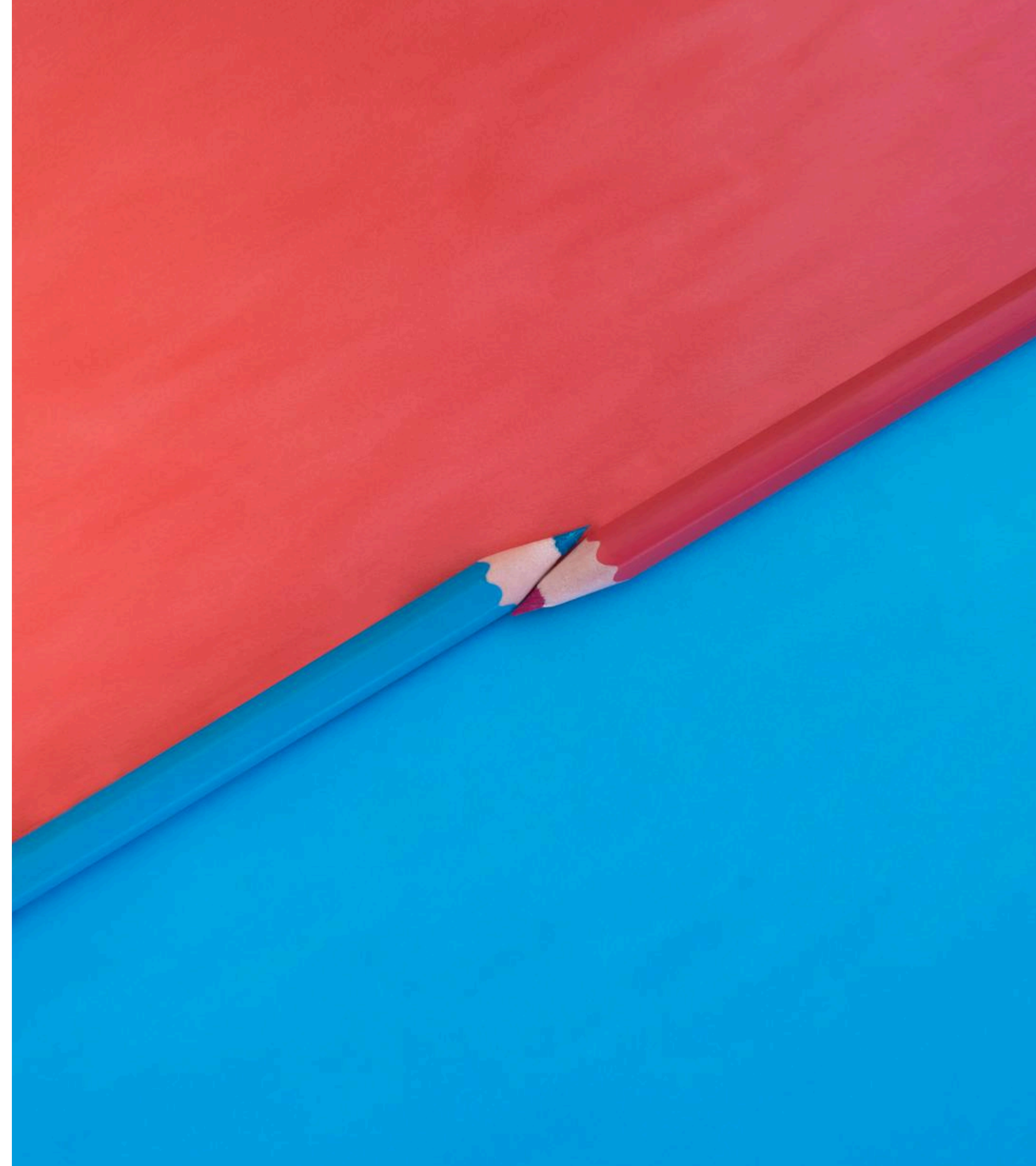
Express

- Knihovna pro JS
- Velmi jednoduché
- Alternativní knihovna – Fastify
- Stojí na middleware
- Body parser – zpracování vstupu dle hlavičky Content-Type



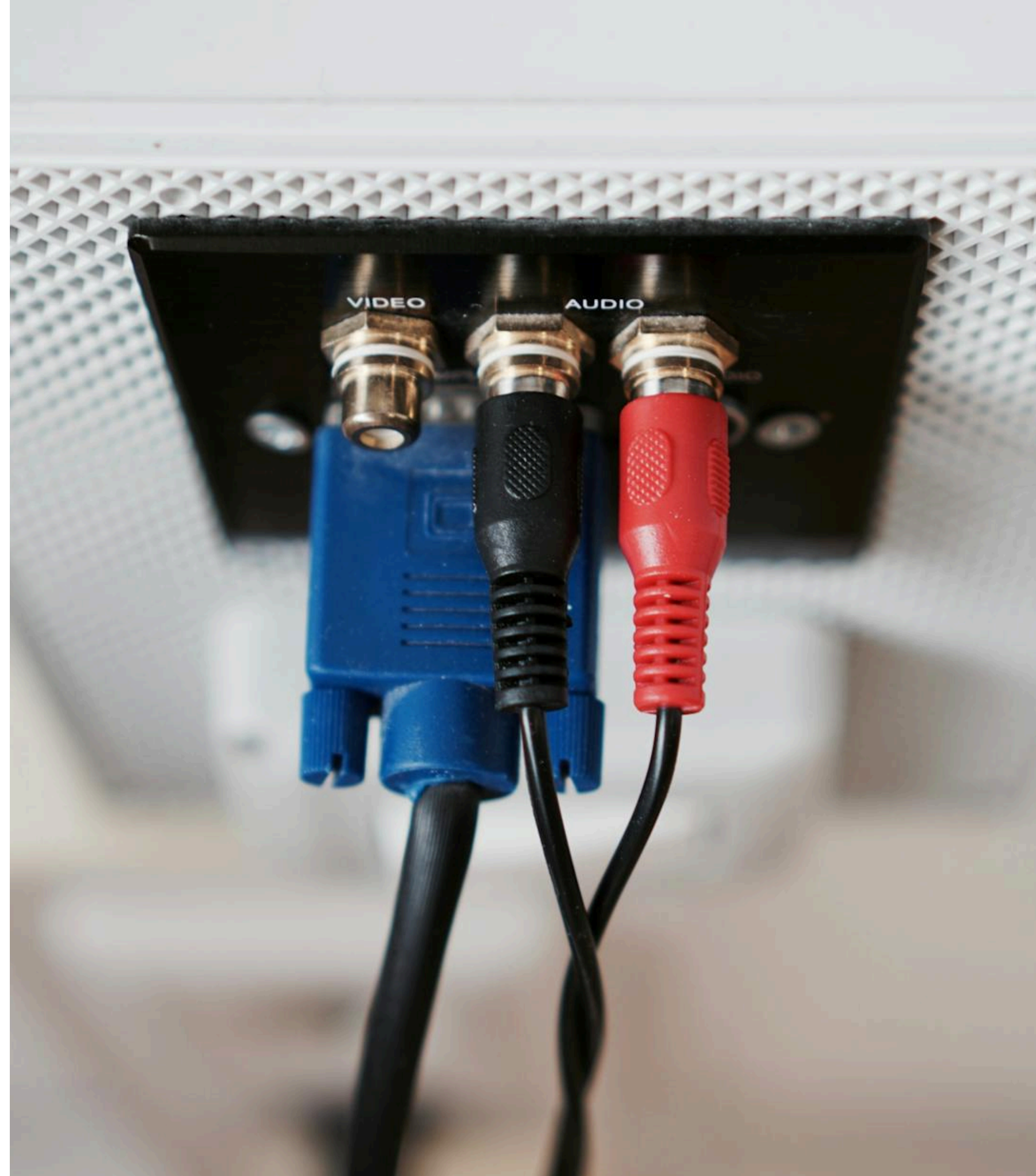
Middleware

- Funkce, která přijímá parametry
- Požadavek
- Odpověď
- Další funkce - nemusí se použít, požadavek se přestane zpracovávat
- Volání v řetízkách



Vstup

- Nutné validovat
- Nikdy nevěřit
- Validace - zod, joi, regexy, ...



Databáze

- Nutné znát relační databáze
- SQL
- Postgresql
- Tabulka, řádek, sloupec
- SELECT, UPDATE, INSERT, DELETE
- Agregace
- Joiny
- Použití v JS, knihovna PG
- Pozor na SQL Injection



Reverzní proxy

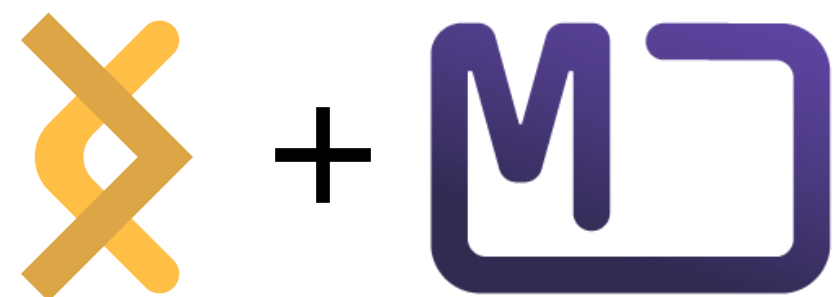
- Služba
- Loadbalancing, bezpečnost a další
- Často pro WWW
- Protokol očekává vše na portu (443), ale chceme na jednom serveru více stránek
- Po přijmutí přesměrujeme na další server, ale na jiném portu



Děkuji za pozornost

matej.cajthaml@ssps.cz

ssps.cajthaml.eu



Vytvořeno pomocí aplikace Materalist

© 2025 Matěj Cajthaml