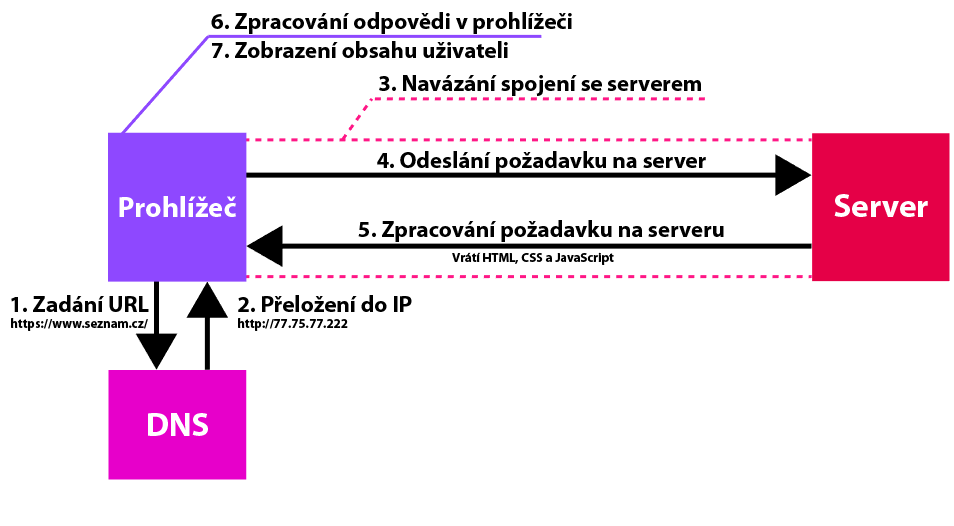
## Základy Frontendu

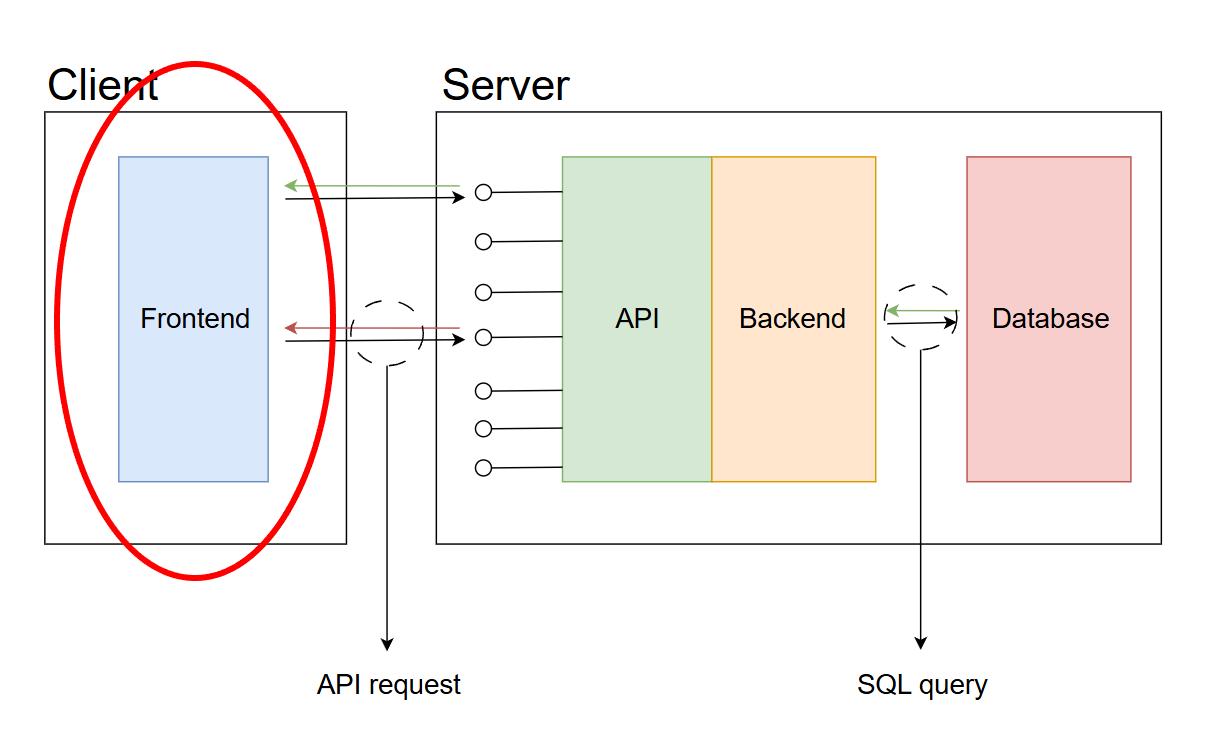
* **Jak funguje web**
* **Webové aplikace vs webová stránka**
* **MPA vs SPA**
* **HTML**
  + Založení dokumentu
  + Struktura dokumentu
  + Základní tagy
  + HTML atributy
* **CSS**
  + Připojení CSS do HTML
  + Selektory, vlastnosti a základní formátování
  + Response Web Design (Flexbox a Grid)
  + Media Queries
* **JavaScript**
  + Interakce s HTML přes DOM
  + Proměnné v Javascriptu
  + Volání API pomocí Javascriptu
* **React, Angular, Vue**
* **Bootstrap**
* **DOM (Document Object Model)**
* **Protokoly**
  + HTTP/HTTPS
  + FTP
  + SMTP
  + IMAP

## Jak funguje web

* **Zadání URL**
  + Například: <https://www.seznam.cz/>
  + URL (Uniform Resource Locator) specifikuje adresu zdroje na internetu. Skládá se z několika částí:
    - **Protokol**: Např. http nebo https (bezpečný přenos dat).
    - **Doména**: Např. example.com (název serveru).
    - **Cesta**: Např. /selenium (specifický obsah na serveru).
* **DNS (Domain Name System)**
  + Prohlížeč nejprve potřebuje zjistit IP adresu serveru, kde je webová stránka uložena.
  + Za tímto účelem se obrátí na DNS server (lze si představit jako telefonní seznam pro webové stránky), který přeloží doménové jméno (např. example.com) na IP adresu (např. <http://77.75.77.222>)
* **Navázání spojení se serverem**
  + Jakmile má prohlížeč IP adresu, naváže spojení se serverem, kde je obsah webu uložen.
* **Odeslání požadavku na server**
  + Prohlížeč odešle na server HTTP požadavek (HTTP request)
* **Zpracování požadavku na serveru**
  + Server přijme požadavek a zpracuje ho. Pokud požadovaná stránka existuje, server ji vrátí jako odpověď (např. HTML soubor)
  + Server klientovi (prohlížeči) vrátí
    - **Status kód**: Např. 200 OK (úspěch) nebo 404 Not Found (chyba).
    - **Obsah**: HTML, CSS, JavaScript nebo jiné soubory.
* **Zpracování odpovědi v prohlížeči**
  + Prohlížeč přijme odpověď od serveru:
    - **HTML**: Prohlížeč vykreslí obsah stránky.
    - **CSS**: Prohlížeč použije styly, aby stránka vypadala vizuálně přívětivě.
    - **JavaScript**: Umožní interaktivitu, dynamické změny obsahu a další funkce.
  + Pokud HTML obsahuje odkazy na další zdroje (např. obrázky, videa, písma, skripty), prohlížeč odešle nové požadavky na server a stáhne je
* **Zobrazení obsahu uživateli**
  + Jakmile prohlížeč stáhne všechny potřebné zdroje, vykreslí stránku na obrazovku uživatele.
  + Uživatel může interagovat se stránkou (klikat, psát do formulářů apod.), což může generovat další požadavky na server.



V této lekci se zaměříme na **Frontend**, tedy tuto část webové aplikace:



## Webové aplikace vs webová stránka

**Webová stránka:**

* Statická nebo dynamická stránka, která obvykle **slouží k poskytování informací** (text, obrázky, videa apod.).
* **Jednosměrná komunikace:** Webová stránka zobrazuje obsah, ale interaktivita je omezená na odkazy a formuláře.
* **Bez nutnosti backendové logiky**: Webové stránky nevyžadují složitou interakci s databázemi nebo servery po každé akci uživatele (kromě běžného načítání obsahu).
* **Příklad**: Osobní blog, firemní prezentační stránka, novinky nebo informační portály.

**Webová aplikace:**

* **Vysoce interaktivní**: Webové aplikace umožňují komplexní interakce s uživatelem, které mohou zahrnovat různé akce, jako je správa dat, interaktivní formuláře nebo real-time aktualizace.
* **Dynamická logika a backend**: Webové aplikace většinou vyžadují složitou logiku na serveru a používají databáze k uchovávání a manipulaci s daty.
* **Například**: Online bankovnictví, e-mailový klienti (Gmail), sociální sítě (Facebook), e-commerce platformy (Amazon).

**Hlavní rozdíly:**

* **Interaktivita**: Webová aplikace je výrazně interaktivnější, zatímco webová stránka je pasivní.
* **Technologie**: Webové aplikace často využívají pokročilé technologie jako AJAX, WebSockets, a další interaktivní prvky pro real-time interakce.
* **Účel**: Webové stránky obvykle slouží k poskytování informací, zatímco webové aplikace se používají pro konkrétní činnosti, jako jsou nákupy, komunikace nebo editace dat.

## MPA vs SPA

**MPA - Multiple Page Application** je návrhový vzor pro webové stránky, kde se každá podstránka plně načítá s celým svým obsahem. Je to služba, kde je hlavní důraz kladen na obsah a počet interakcí s uživatelem je nižší. Důležitý je unikátní obsah a pozice ve webovém vyhledávači.

Jak funguje:

* Každý odkaz načítá nový HTML soubor ze serveru.
* Backend generuje obsah na serveru (server-side rendering) a odesílá plně sestavenou stránku klientovi.

Výhody:

* SEO: Každá stránka má svůj URL a je snadno přístupná vyhledávačům.
* Jednoduchost: Architektura je jednodušší, protože není potřeba složité správy stavu na klientovi.
* Bezpečnost: Tradiční přístup může být bezpečnější, protože dynamika je omezena.

Nevýhody:

* Pomalejší interakce: Každý přechod mezi stránkami vyžaduje nové načítání celého obsahu.
* Uživatelský zážitek: Méně plynulý než SPA.
* Náročnější na server: Vyšší počet požadavků na server kvůli načítání stránek.

Příklady:

* E-commerce stránky (např. Amazon)
* Zpravodajské weby
* Klasické firemní weby

**SPA - Single Page Application** je moderní přístup, kdy se celá stránka načte jednou na začátku aplikace. Díky tomu webová aplikace funguje plynuleji a spotřebovává méně prostředků, zejména méně přenosu dat. Každá podstránka vyžaduje velkou interakci s uživatelem (web e-governmentu, bankovní nebo pojišťovací služba, aplikace pro sociální sítě atd.).

Jak funguje:

* Při prvním načtení se stáhne základní HTML, CSS a JavaScript aplikace.
* Další obsah je načítán a aktualizován prostřednictvím API volání nebo AJAXu.
* Navigace mezi "stránkami" je simulována pomocí JavaScriptu a správy historie prohlížeče (např. s použitím React Router nebo Angular Router). SPA je v současné době trendovým designem.

Výhody:

* Rychlost: Po prvním načtení aplikace je přechod mezi sekcemi velmi rychlý.
* Lepší UX: Uživatelský zážitek je plynulý, bez přerušení načítáním nových stránek.
* Možnosti offline režimu: S využitím cache a service workerů lze vytvořit offline aplikace.
* Podobnější desktopové aplikaci: uživatel může začít pracovat až po úplném načtení aplikace. Ve skutečnosti to lze považovat za nevýhodu (delší doba spuštění), nicméně po načtení aplikace vypadá více jako standardní aplikace.
* Vyšší efektivita: navzdory delšímu startu pracuje po načtení plynuleji.
* Menší režie na straně serveru: méně dotazů na server znamená menší režii serveru a nakonec může zvládnout více souběžných uživatelů.

Nevýhody:

* SEO: Optimalizace pro vyhledávače může být složitější, protože obsah není hned dostupný při načtení (vyžaduje server-side rendering nebo prerendering).
* Komplexita: SPA jsou často složitější na implementaci a údržbu kvůli těsné integraci frontendu a backendu.

Příklady:

* Gmail
* Facebook

**Kdy použít kterou?**

**SPA:**

* Pokud chcete moderní, interaktivní aplikaci s plynulou uživatelskou zkušeností.
* Ideální pro webové aplikace (např. správu dat, emailové aplikace, dashboardy).
* Pokud máte silné technické zázemí na frontendu.

**MPA:**

* Pokud je hlavním cílem SEO a snadné indexování obsahu.
* Ideální pro webové stránky s velkým množstvím obsahu (e-shopy, blogy, magazíny).
* Pokud je požadavek na jednodušší implementaci.

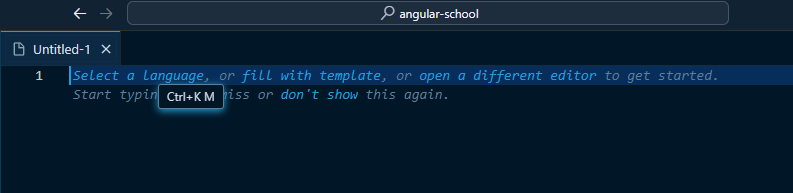
## HTML

### Založení dokumentu HTML (stejný postup pro css)

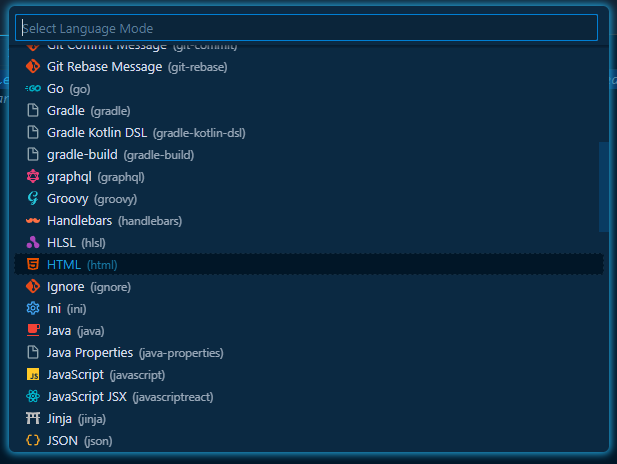
* Otevření Visual Studio Code
* Kliknutí na **File** a poté na **New Text File**

### 

* Klik na **Select a language**



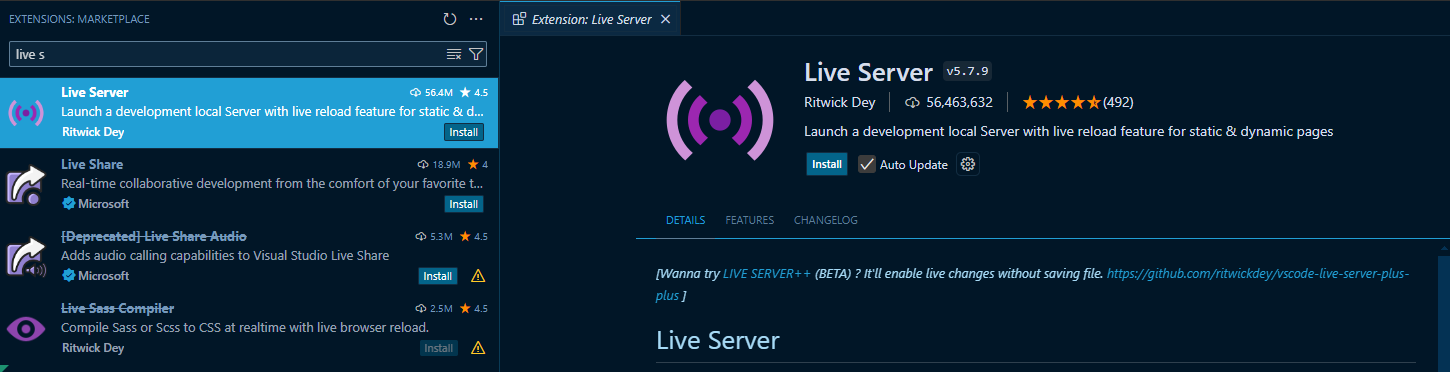
* Vyberte ze seznamu HTML, výsledný dokument pak můžete uložit klikem na **File** a poté na **Save as…**



* Pro pohodlnější práci si můžete přidat rozšíření Live Server. Do rozšíření ve Visual Studio Code dostanete stiskem ctrl+shift+X nebo ikonou čtverců.



* Vyhledáte Live Server a kliknete na install. Stránku spustíte pravým klikem na soubor HTML a následně **Open with Live Server**



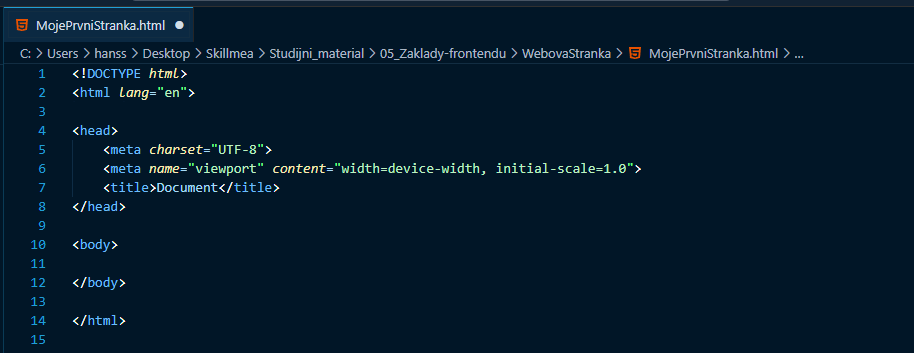
* HTML jde založit i jednodušeji. Stačí vytvořit prázdný dokument a ten uložit (nebo přepsat koncovku na) **.html**. A poté soubor otevřít ve webovém prohlížeči

### Struktura dokumentu

Základní strukturu HTML dokumentu lze pomocí Visual Studio code vygenerovat pomocí znaku „**!**“. Viditelnou část webové stránky pak lze psát mezi tagy <body></body>.

* **<!DOCTYPE html>:** Deklarace typu dokumentu, označuje HTML5.
* **<html>:** Kořenový element, obaluje celou HTML stránku.
* **<head>:** Obsahuje metadata (informace o stránce, odkazy na styly, atd.).
* **<body>:** Obsahuje hlavní viditelný obsah stránky.

### 



### Základní tagy

**Textové tagy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Nadpis | <h1> | <h1>Hlavní nadpis</h1> | Ano |
| Odstavec | <p> | <p>Toto je odstavec textu.</p> | Ano |
| Zvýrazněný text | <strong> | <strong>Důležité slovo</strong> | Ano |
| Kurzíva | <em> | <em>Zvýrazněný text</em> | Ano |
| Zalomí řádek | <br> | První řádek<br>Druhý řádek | Ne |
| Horizontální čára | <hr> | <hr> | Ne |

**Strukturální tagy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Hlavička | <header> |  | Ano |
| Navigace | <nav> |  | Ano |
| Hlavní obsah | <main> |  | Ano |
| Sekce obsahu | <section> |  | Ano |
| Postranní panel | <article> |  | Ano |
| Horizontální čára | <footer> |  | Ano |
| Obecný kontejner | <div> |  | Ano |

**Seznamové tagy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Neuspořádaný  seznam | <ul> | <ul>  <li>Položka 1</li>  <li>Položka 2</li>  </ul> | Ano |
| Uspořádaný  seznam | <ol> | <ol>  <li>Položka 1</li>  <li>Položka 2</li>  </ol> | Ano |
| Definiční  seznam | <dl> | <dl>  <dt>HTML</dt>  <dd>HyperText Markup Language</dd>  </dl> | Ano |

**Tagy pro odkazy a média**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Odkaz | <a> | <a href="https://www.seznam.cz">Navštivte seznam stránku</a> | Ano |
| Obrázek | <img> | <img src="obrazek.jpg" alt="Popis obrázku"> | Ne |
| Video | <video> | <video controls>  <source src="video.mp4" type="video/mp4">  Váš prohlížeč nepodporuje video tag.  </video> | Ano |
| Zvuk | <audio> | <audio controls> <source src="audio.mp3" type="audio/mpeg">  Váš prohlížeč nepodporuje audio tag.  </audio> | Ano |

**Tabulkové tagy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Tabulka | <table> | <table>  <thead>  <tr>  <th>Jméno</th>  <th>Věk</th>  </tr>  </thead>  <tbody>  <tr>  <td>Jan</td>  <td>30</td>  </tr>  </tbody> </table> | Ano |
| Hlavička tabulky | <thead> | Ano |
| Tělo tabulky | <tbody> | Ano |
| Řádek tabulky | <tr> | Ano |
| Buňka v hlavičce | <th> | Ano |
| Buňka v těle | <td> | Ano |

**Formulářové tagy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Význam** | **Tag** | **Příklad** | **Párový tag** |
| Formulář | <form> | <form action="/submit" method="post">  <label for="name">Jméno:</label>  <input type="text" id="name" name="name">  <button type="submit">Odeslat</button> </form> | Ano |

### HTML atributy

* **class** – Slouží k definování CSS tříd, které mohou být použity k nastylování elementu.

<div class="container"></div>

* **id** – Unikátní identifikátor elementu na stránce. Používá se v JavaScriptu nebo CSS pro selekci konkrétního prvku. Každé ID musí být na stránce jedinečné.

<div id="header"></div>

* **style** – Umožňuje přidat inline CSS přímo do elementu.

<p style="color: red; font-size: 16px;">Text</p>

* **href** – Používá se u odkazů (<a>) a definuje cílovou URL adresu, na kterou odkaz směřuje.

<a href="https://example.com">Klikněte zde</a>

* **src** – Používá se u médií, jako jsou obrázky, skripty nebo iframy, a určuje cestu k souboru.

<img src="image.jpg" alt="Popis obrázku">

* **title** – Poskytuje doplňkové informace, které se zobrazí jako nástrojová lišta po najetí myší na prvek.

<button title="Kliknutím odeslat formulář">Odeslat</button>

* **alt** – Používá se u obrázků a poskytuje alternativní text, který se zobrazí, pokud obrázek nelze načíst.

<img src="logo.png" alt="Logo společnosti">

* **placeholder** – Definuje text, který se zobrazí v poli formuláře, dokud uživatel nezačne psát.

<input type="text" placeholder="Zadejte vaše jméno">

* **disabled** – Označuje prvek jako neaktivní, takže uživatel s ním nemůže interagovat.

<button disabled>Neaktivní</button>

* **required** – Určuje, že prvek formuláře je povinný a musí být vyplněn před odesláním.

<input type="email" required>

## CSS

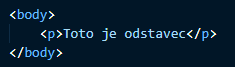
### Připojení CSS do HTML

Je nutné vytvořit nový soubor s koncovkou **.css**. Tento soubor by měl být ve stejné složce jako HTML. V HTML je poté nutno do **head** přidat tento příkaz který nalinkuje HTML s CSS.

<link *rel*="stylesheet" *href*="MojePrvniStranka.css">



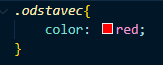
Před aplikací stylů:





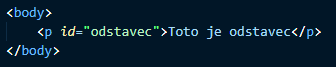
Po aplikací stylů navázání přes class:





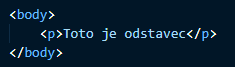


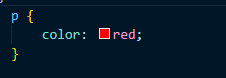
Po aplikací stylů navázání přes id (výsledek je stejný):





Po aplikací stylů na všechny prvky <p> (výsledek je stejný):





### Selektory, vlastnosti a základní formátování

**Barvy a pozadí**

1. **color –** Nastavení barvy textu.
2. **background-color –** Barva pozadí prvku.
3. **background-image –** Obrázek na pozadí.
4. **background-size –** Velikost pozadí (např. cover, contain).
5. **background-repeat –** Opakování obrázku na pozadí.

**Text a fonty**

1. **font-family** – Typ písma (např. Arial, Verdana).
2. **font-size** – Velikost písma (např. 16px, 1em).
3. **font-weight** – Tloušťka písma (normal, bold, apod.).
4. **font-style** – Styl písma (normal, italic, apod.).
5. **line-height** – Výška řádku.
6. **letter-spacing** – Mezera mezi písmeny.
7. **text-align** – Zarovnání textu (left, center, right).
8. **text-decoration** – Dekorace textu (např. underline, none).
9. **text-transform** – Převod textu na velká/malá písmena.

**Rozměry a prostor**

1. **width** – Šířka prvku.
2. **height** – Výška prvku.
3. **max-width** – Maximální šířka prvku.
4. **max-height** – Maximální výška prvku.
5. **margin** – Vnější odsazení (mezi prvky).
6. **padding** – Vnitřní odsazení (uvnitř prvku).
7. **border** – Okraj prvku (tloušťka, styl, barva).
8. **box-shadow** – Stín prvku.

**Rozložení a pozice**

1. **display** – Zobrazení prvku (block, inline, flex, apod.).
2. **position** – Pozice prvku (static, absolute, relative, fixed, sticky).
3. **top**, **right**, **bottom**, **left** – Umístění prvku při použití position.
4. **z-index** – Vrstva prvku (vyšší číslo = výše).
5. **overflow** – Co dělat s přetékajícím obsahem (hidden, scroll, apod.).

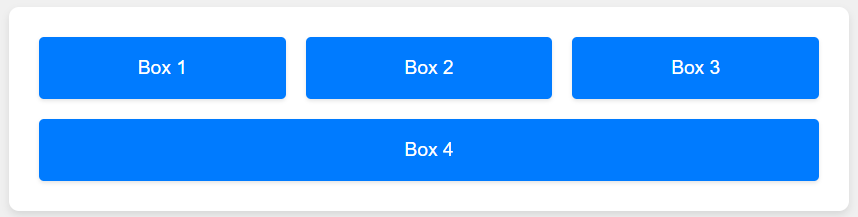
**Flexbox a Grid**

1. **flex** – Nastavení pružnosti prvku ve flexboxu.
2. **align-items** – Zarovnání položek ve flexboxu nebo gridu.
3. **grid-template-columns** – Definování sloupců v gridu.

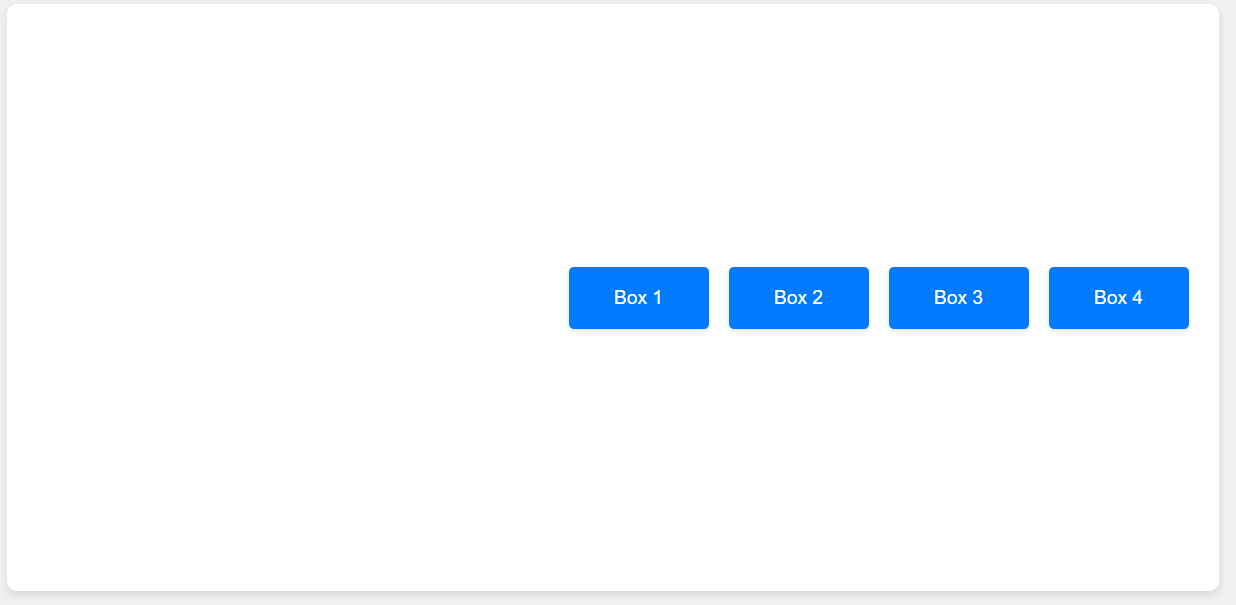
### Flexbox a Grid

**Flexbox (Flexible Box Layout)**

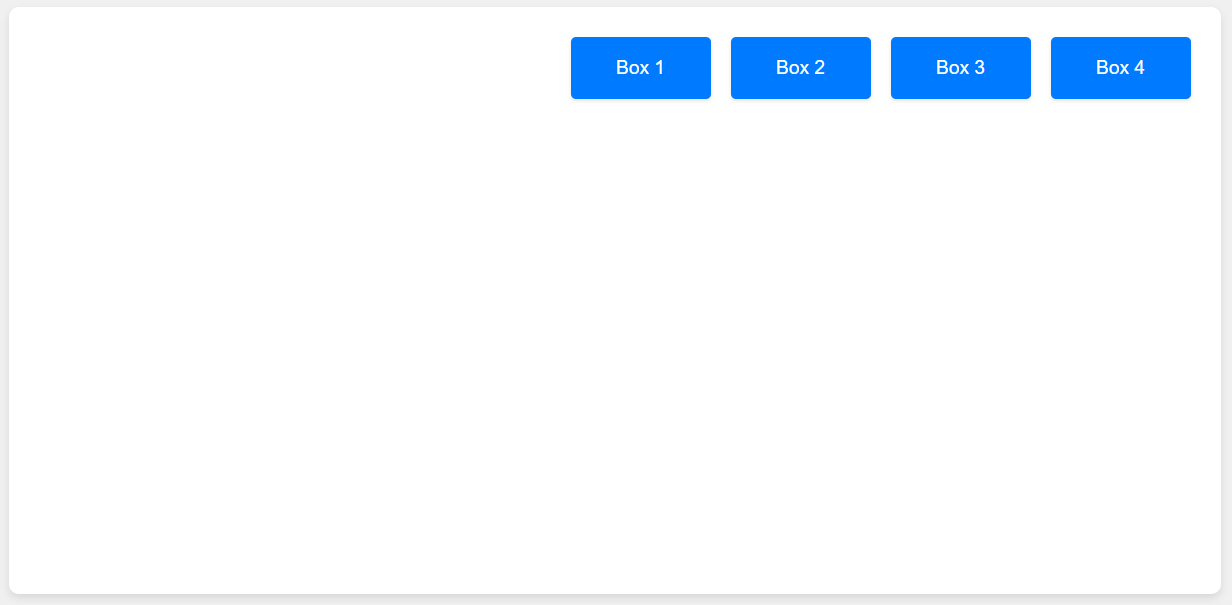
Flexbox je jednorozměrný systém layoutu, který se zaměřuje na uspořádání položek v **řádku** nebo **sloupci**. Hodí se především pro menší struktury, jako je zarovnání prvků na ose nebo tvorba flexibilních navigačních lišt.

****

Příklad použtí justify-content: flex-end

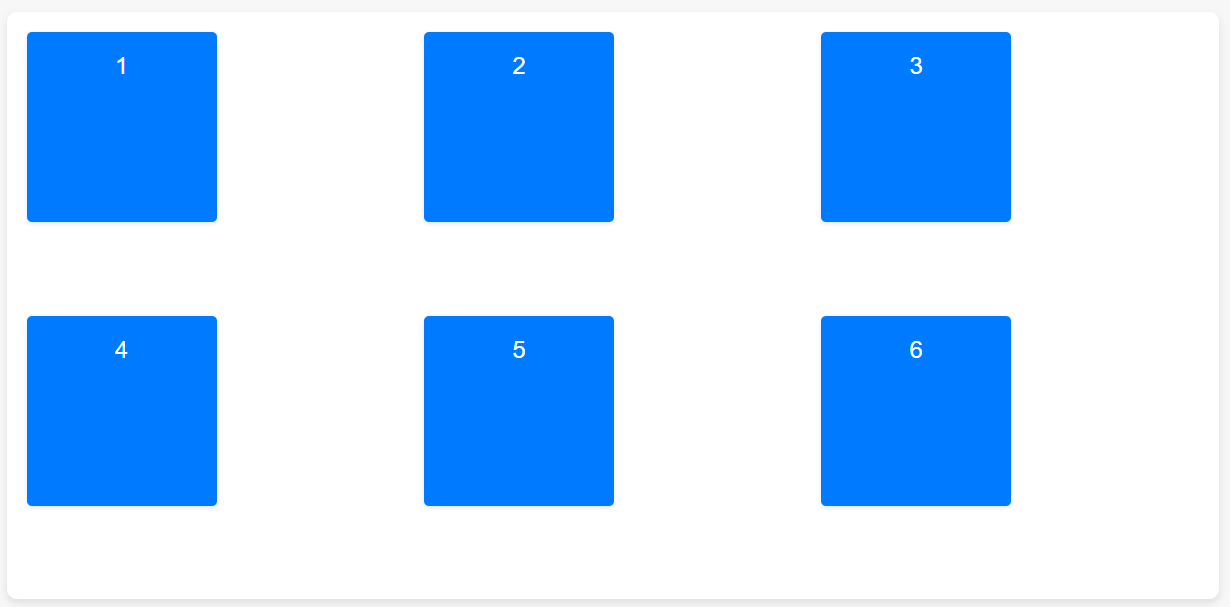
****

Příklad použtí justify-content: flex-end a align-content: flex-start

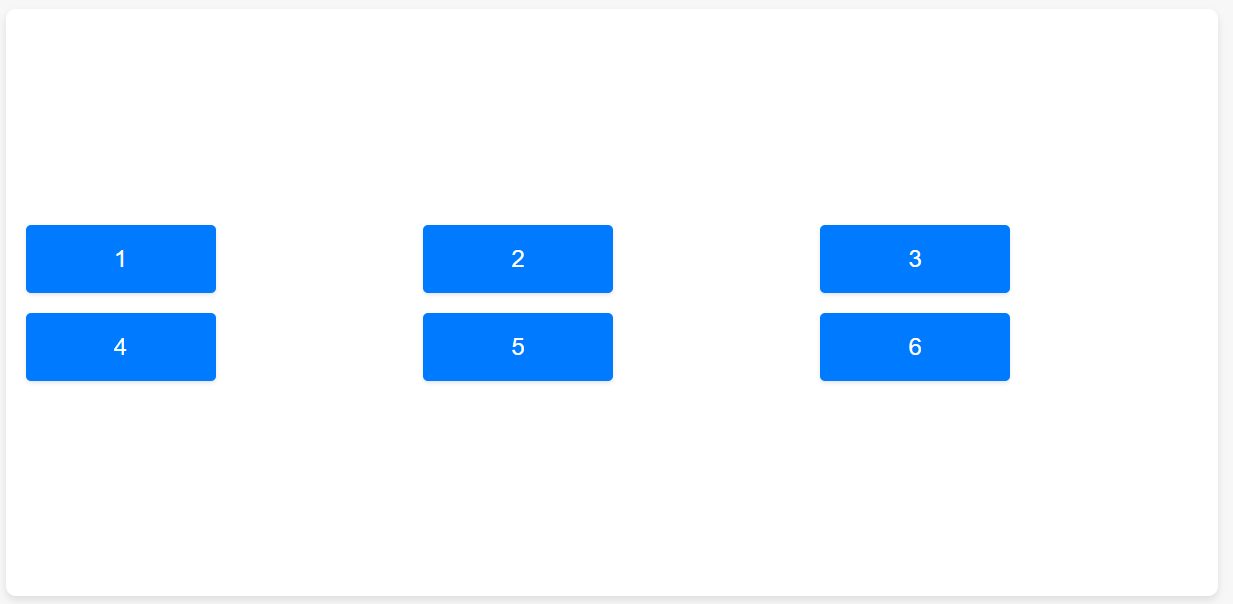
****

**CSS Grid**

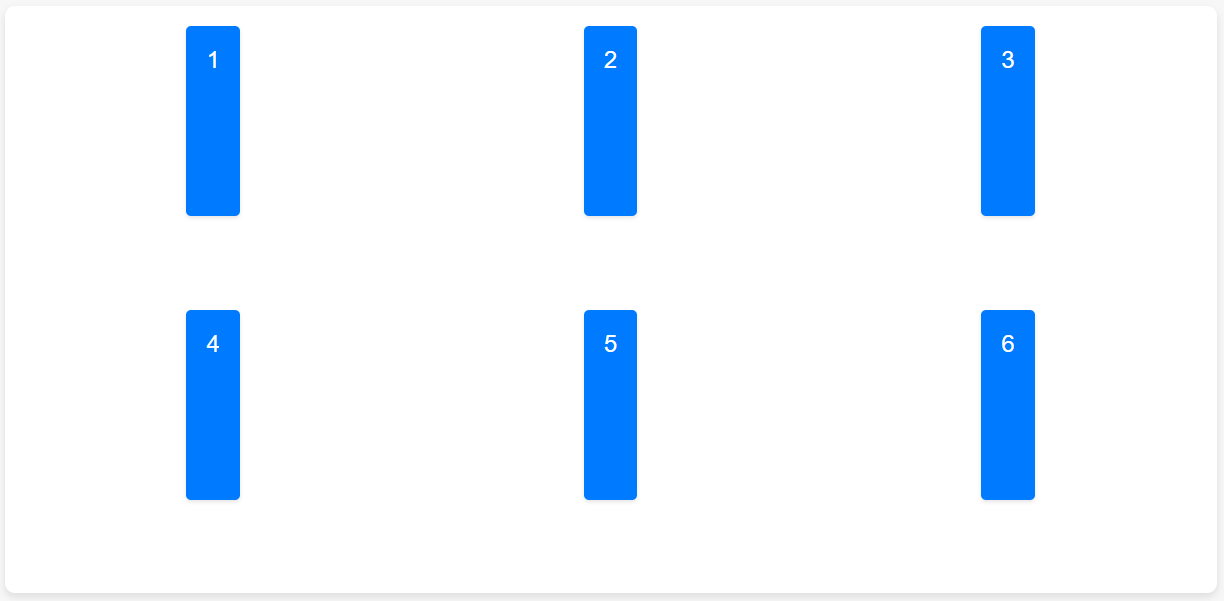
CSS Grid je dvourozměrný systém layoutu, který umožňuje uspořádání prvků ve **sloupcích i řádcích**. Je ideální pro komplexní layouty stránek.



Příklad použtí align-content: center



Příklad použtí justify-items: center

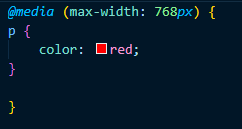


### Media Queries

Responsive Web Design (RWD) je přístup k tvorbě webových stránek, který zajišťuje, že se obsah správně zobrazuje na různých zařízeních a velikostech obrazovky.

Media Queries v CSS se používají k přizpůsobení stylování podle různých vlastností zařízení, jako je šířka, výška, rozlišení, orientace nebo typ zařízení. Jsou základní součástí responzivního designu, který umožňuje webovým stránkám dynamicky reagovat na různé velikosti obrazovek a zařízení.

Příklad. Text v paragrafu zčervená pokud je velikost obrazovky menší než 768px:



@media (max-width: 768px) - Definují maximální šířku, při které se styly aplikují.

@media (min-width: 1024px) - Definují šířku, při které se styly aplikují.

@media (orientation: landscape) - Určuje orientaci zařízení: **landscape** (na šířku) nebo **portrait** (na výšku).

@media (min-resolution: 300dpi) - Zaměřuje se na rozlišení obrazovky

@media (aspect-ratio: 16/9) - Poměr stran obrazovky

@media (device-width: 375px) - Používají se pro šířku a výšku zařízení.

@media (prefers-color-scheme: dark) - Reaguje na preferovaný barevný režim uživatele, například **dark** nebo **light**.

@media (hover: hover) - Zjišťují, zda zařízení podporuje hover (například myš) nebo jak přesný je pointer (například dotykový displej).

@media print – Používá se k definování stylů pro tisk.

@media (min-width: 768px) and (max-width: 1200px) and (orientation: landscape) - Tato Media Query nastaví styl pouze pro zařízení, která mají šířku mezi 768px a 1200px a jsou v orientaci **na šířku**.

## JavaScript

### Interakce s HTML přes DOM

JavaScript je programovací jazyk, který umožňuje interaktivitu na webových stránkách tím, že umožňuje manipulaci s HTML a CSS v reálném čase. Používá se pro různé úkoly, jako jsou dynamické změny obsahu, validace formulářů, animace a interakce se serverem prostřednictvím AJAXu.



**document** – Objekt document představuje celý HTML dokument načtený v okně prohlížeče.

**getElementById("text")** - Metoda objektu document, která vyhledá HTML prvek podle jeho ID.

**style** – Vlastnost DOM elementu, která umožňuje přístup a změnu jeho CSS stylů.

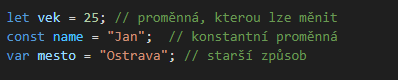
**.color** - Specifická vlastnost CSS, která určuje barvu textu

**= "blue"** - Přiřazení nové hodnoty vlastnosti color

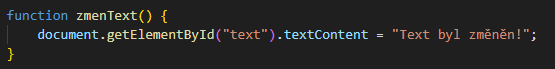
### Proměnné v Javascriptu

Proměnné slouží k ukládání dat v programu. V JavaScriptu se deklarují pomocí klíčových slov:

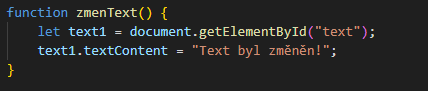
* **let**: Proměnná, jejíž hodnota se může měnit.
* **const**: Konstantní proměnná, jejíž hodnota se pořizuje jednou a už se nemění.
* **var** (zastaralé): Dříve používané pro deklaraci proměnných, ale může způsobit problémy kvůli funkčnímu rozsahu.



Před použitím proměnné:



Po použití proměnné:

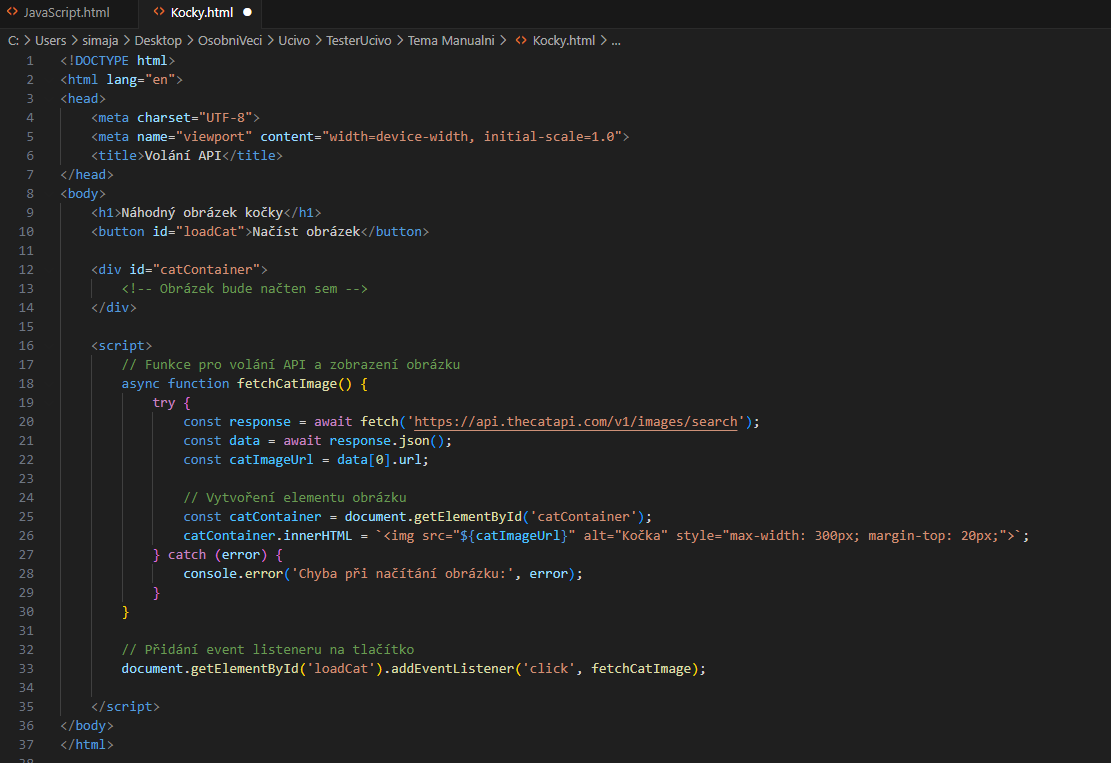


### Volání API pomocí Javascriptu

Volání API pomocí JavaScriptu umožňuje získávat data ze serveru pomocí HTTP požadavků, například s využitím funkce fetch. Tento proces je asynchronní, což znamená, že aplikace může pokračovat v běhu, zatímco čeká na odpověď ze serveru, čímž se zvyšuje efektivita a uživatelský komfort.

Příklad volání API: <https://thecatapi.com/thanks>

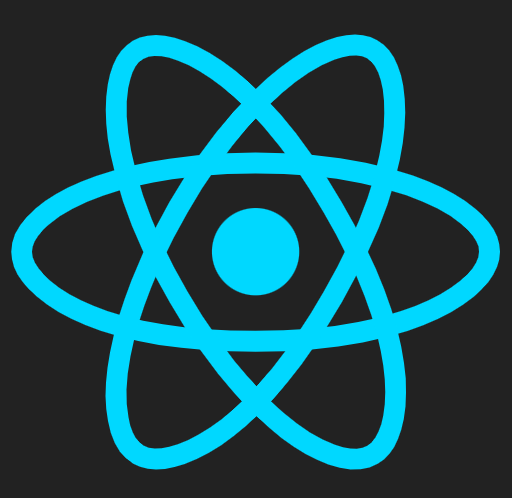
Konkrétně endpoint: https://api.thecatapi.com/v1/images/search



## React, Angular, Vue

React, Angular a Vue.js jsou tři z nejpopulárnějších JavaScriptových frameworků/knihoven pro tvorbu moderních webových aplikací. Liší se především ve filozofii, architektuře a způsobu používání.

**React**



Popis:

* Knihovna pro tvorbu uživatelských rozhraní (UI), nikoliv plnohodnotný framework.
* Vyvinut Facebookem.
* Zaměřuje se na vytváření komponent a používá virtuální DOM.
* React je flexibilní a umožňuje integraci s dalšími knihovnami (např. pro řízení stavu: Redux, Zustand).

Výhody:

* Jednoduchost a flexibilita.
* Velká komunita a ekosystém.
* Virtuální DOM zajišťuje rychlou aktualizaci UI.

Nevýhody:

* Vyžaduje přidání dalších knihoven pro komplexní projekty.
* Náročnější pro začátečníky kvůli JSX (syntaxe kombinující JavaScript a HTML).

Použití:

* Facebook, Instagram, Airbnb, Netflix.

**Angular**



Popis:

* Plnohodnotný framework pro vývoj webových aplikací.
* Vyvinut Googlem.
* Používá TypeScript a poskytuje strukturovaný způsob vývoje.

Výhody:

* Obsahuje vše, co je potřeba (např. dvoucestné vázání dat, dependency injection).
* Podpora TypeScriptu zvyšuje čitelnost a robustnost kódu.
* Skvělé pro rozsáhlé aplikace díky své architektuře.

Nevýhody:

* Složitější na naučení kvůli své komplexnosti.
* Náročnější na výkon při rozsáhlých aplikacích ve srovnání s Reactem či Vue.

Použití:

* Google Ads, Gmail, Upwork, Forbes.

Poznámka:

* Vytvoření nového projektu se provede otevřením konzole a zadáním příkazu: ng new nazev-projektu. Angular musí být nainstalován to se provede příkazem npm install -g @angular/cli.

**Vue.js**



Popis:

* Progresivní framework zaměřený na jednoduchost a snadnou integraci.
* Vyvinut komunitou, iniciátorem byl Evan You (bývalý vývojář Googlu).
* Nabízí jednoduchou syntaxi a intuitivní API.

Výhody:

* Nízká vstupní bariéra – snadné pro začátečníky.
* Skvělá dokumentace a flexibilita.
* Menší velikost než Angular nebo React.

Nevýhody:

* Menší komunita a méně rozsáhlý ekosystém než u Reactu a Angularu.
* Menší podpora v podnikovém prostředí.

Použití:

* Alibaba, Xiaomi, GitLab, Behance.

## Bootstrap

Bootstrap je open-source framework pro vývoj responzivních a mobilních webových stránek, který poskytuje připravené CSS a JavaScript komponenty, jako jsou tlačítka, formuláře, navigace a mřížky. Umožňuje vývojářům rychle vytvářet moderní a funkční uživatelská rozhraní, aniž by museli začínat od nuly s vlastním kódem CSS a JavaScriptu.

<https://getbootstrap.com/>

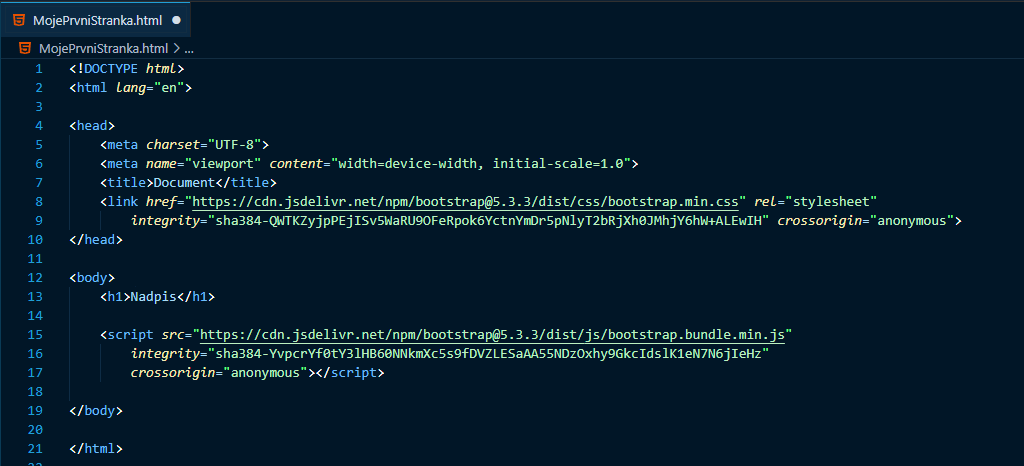
Přídání Bootstrap do projektu:

Pro použití přidejte toto do **head:**

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

A toto do **body**:

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>



<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Bootstrap demo</title>

**<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">**

</head>

<body>

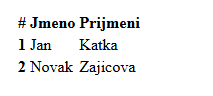
<h1>Hello, world!</h1>

**<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>**

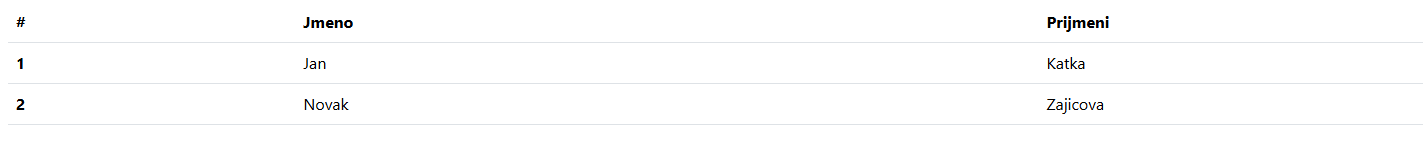
</body>

</html>

Normální tabulka bez formátování:



Tabulka s pomocí Bootstrap:



## DOM (Document Object Model)

DOM neboli Document Object Model funguje jako prostředník mezi JavaScriptem a HTML. Nejde o jiný programovací jazyk, ale o vestavěné API pro práci s JavaScriptem.

Webová stránka je výsledkem spolupráce několika programovacích jazyků, kódů a vývojářů. Pro bezproblémový chod webu bez výkonových problémů je zapotřebí nespočet řádků kódu, frameworků, API a dalších prvků. DOM je jednou z klíčových součástí, která hraje intuitivně nedílnou roli v dynamickém fungování webových stránek.

**DOM** je tedy vnitřní reprezentace HTML nebo XML dokumentu jako objektů, které mohou být manipulovány pomocí JavaScriptu. To znamená, že každý HTML tag, atribut a textový obsah je součástí tohoto modelu a může být upravován dynamicky.



## Protokoly

Webové protokoly jsou souborem pravidel, která umožňují výměnu dat mezi zařízeními v síti, například mezi webovými prohlížeči a servery. Tyto protokoly, jako HTTP, FTP nebo SMTP, definují strukturu a formát komunikace, čímž umožňují efektivní a bezpečnou interakci na internetu.

### HTTP/HTTPS (HyperText Transfer Protocol a HTTP Secure)

HTTP (HyperText Transfer Protocol) je základní protokol pro přenos webových stránek mezi klientem a serverem na internetu. Umožňuje prohlížečům požadovat a přijímat obsah webových stránek a interagovat s různými servery prostřednictvím jednoduchých textových požadavků a odpovědí.

**Co je požadavek HTTP (Request)?**

http požadavek je druh zprávy, kterou klient (ve většině případů webový prohlížeč) odešle serveru a vyžaduje určité specifické zdroje. Skládá se z několika prvků, jako je metoda požadavku (GET, POST atd.), hlavičky a příležitostně tělo nesoucí data. Toto je část požadavku, kde klient nastíní požadavek na server a co chce.

**Co je to HTTP odpověď (Response)?**

Http odpověď je zpráva odeslaná serverem klientovi jako odpověď na požadavek HTTP. Jednoduše řečeno, má stavový kód, který popisuje výsledek požadavku; záhlaví, což je informace o odpovědi; a tělo, což je skutečná odpověď nebo chybová zpráva.

**HTTP vs HTTPS**

Hypertext Transfer Protocol (HTTP) je protokol, pomocí kterého se hypertext přenáší přes web. HTTP je díky své jednoduchosti nejrozšířenějším protokolem pro přenos dat přes web, ale data (tj. hypertext) vyměňovaná pomocí HTTP nejsou tak bezpečná, jak bychom si přáli. [Kryptografické protokoly](https://www.geeksforgeeks.org/cryptography-and-its-types/) jako SSL a/nebo TLS mění HTTP na HTTPS, tj. HTTPS = HTTP + kryptografické protokoly.

FTP (File Transfer Protocol)

Tento protokol slouží k přenosu souborů mezi počítači v síti. Používá se často k nahrávání souborů na webové servery.

SMTP (Simple Mail Transfer Protocol)

Používá se k odesílání e-mailů mezi servery. Zajišťuje, že zprávy dosáhnou svého cílového místa.

IMAP (Internet Message Access Protocol)

Tento protokol slouží k přijímání e-mailů a synchronizaci mezi e-mailovým serverem a klientem, což umožňuje přístup k e-mailům z různých zařízení.