**ZÁKLADY JAZYKA HTML**

## Co je to jazyk HTML?

**HTML** je základním kamenem každé webové stránky na internetu. Jedná se o jednoduchý **značkovací jazyk**, pomocí kterého definujeme **strukturu pro náš obsah**. Veškeré\*\* odstavce, nadpisy, obrázky či tabulky\*\* jsou na webu zapsány právě pomocí jazyka HTML.

Díky své jednoduchosti je to často první jazyk, se kterým se lidé seznamují. Základní přehled jazyka HTML využívají také marketéři nebo designeři při své práci. Pro front-end vývojáře je pak jeho výborná znalost nezbytnou podmínkou.

HTML je zkratkou pro ***HyperText Markup Language*** (hypertextový značkovací jazyk):

* Spojení **Markup Language** (značkovací jazyk) znamená, že se jedná o jazyk, který ***používá značky (tagy) k vytváření a formátování obsahu***. Význam značek si blíže postupně představíme.
* **HyperText** je metoda procházení internetových stránek. Jednotlivé ***stránky mohou být mezi sebou propojeny tzv. hypertextovými odkazy***. Po kliknutí na daný odkaz se dostaneme na další stránku.

A logo of a website

Description automatically generated

## Anatomie HTML element

Základní jednotkou značkovacího jazyka HTML jsou tzv. **HTML elementy**. Tyto elementy nám určují **strukturu webu a definují význam samotného obsahu**. Pojďme se podívat na to, jak se zapisují.

Schéma následujícího obrázku představuje ukázku toho, jak zapisujeme HTML element pro odstavce:

A black background with blue text and red text

Description automatically generated

Zápis začíná ***otevírací značkou*** <p>. Poté následuje samotný textový obsah (*Dobrý den*). Aby bylo zřejmé, kde celý element končí, tak ho uzavřeme pomocí ***uzavírací značky*** </p>.

Význam jednotlivých částí z obrázku je následující:

* **HTML element** - Základní jednotka jazyka HTML. Skládá se z otevírací a uzavírací značky, mezi nimiž je umístěn textový obsah, popř. jiný HTML element. Existují však i výjimky, tzv. *prázdné elementy*, které nemají obsahovou část a zapisujeme je bez uzavírací značky. V našem případě celý nápis:<p>Dobrý den</p>.- **HTML tag** - Tag (značka) je reprezentována svým názvem, který určuje jeho význam. Abychom odlišili značku od běžného textu, uzavíráme její název mezi ostré závorky < a >. Ukázka nahoře obsahuje značku <p> pro zápis odstavce. Znak p představuje zkratku pro odstavec (z anglického slova *paragraph*). Jiné elementy budou reprezentovány pomocí jiných znaků uzavřenými mezi < a >.
* **Otevírací značka** - Otevírací značka určuje začátek celého HTML elementu. V našem případě <p>.
* **Uzavírací značka** - Konec HTML elementu poznáme pomocí uzavírací značky. Jeho zápis obsahuje kromě špičatých závorek také lomítko / umístněné hned po první závorce. Na naší ukázce patří k otevírací značce <p>uzavírací značka </p>.
* **Obsah** - Informace umístěná mezi otevírací a uzavírací značku (text, popř. jiný HTML element). V našem případě je to nápis Dobrý den, ale můžeme mít např. i takovouto strukturu:

<p>

<b>Dobrý den</b>

</p>

V tomto případě je pak obsahem elementu paragraph další HTML element "tučný text", který obsahuje text Dobrý den.

### Párové a nepárové tagy

Tagy (značky) bývají velmi často **párové** - pro otevírací značku existuje vždy její uzavírací značka. V naší ukázce vystupují tagy <p> a </p> jako párové.

Protože ne každý HTML element musí mít vždy svou obsahovou část, existují také **nepárové tagy**. Mluvíme potom o tzv. prázdných HTML elementech. U nich nemá význam používat uzavírací značku. Vodorovnou čáru v jazyce HTML zapíšeme pomocí:

<hr>

případně pomocí

<hr />

Všimněte si dvou různých zápisů. Toto je dané historickým vývojem jazyka HTML, kdy v minulosti bylo vyžadováno, aby každý HTML element byl náležitě zakončen. Jednalo se o předchozí standard [XHTML](https://www.w3schools.com/html/html_xhtml.asp). Používal se proto speciální zápis, kdy nepárový tag obsahoval mezeru a lomítko před poslední závorkou (<hr />). V moderním zápisu jazyka HTML5 již tato podmínka neplatí a můžeme psát klidně <hr>.

### Základní shrnutí o HTML

V jazyce HTML máme k dispozici velké množství různých tagů (značek). Některé slouží k organizaci obsahu, jiné k formátování textu. Některé HTML tagy nám umožní vkládat netextový obsah, jako obrázky či videa. Možností je hodně, ale nemějte obavu. Není třeba znát vše. Pro běžné používání si vystačíme se základním přehledem. Ale k tomu se postupně dostaneme.

## Atributy HTML elementu

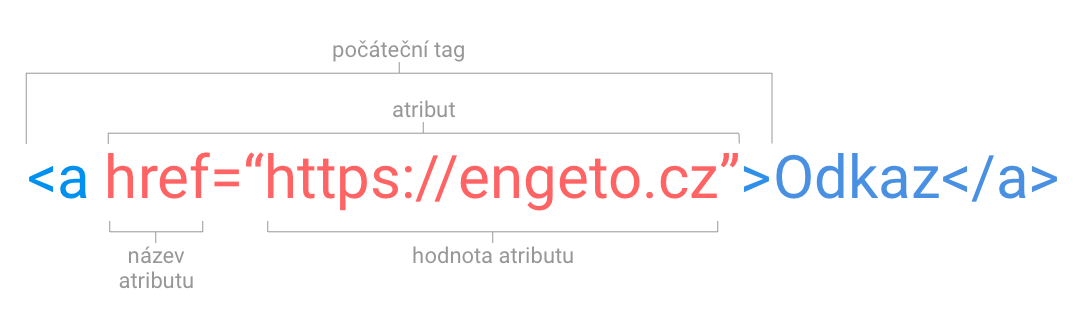
Nyní už víme, co je to **HTML element** a že v jeho zápise hraje důležitou roli **HTML tag**. Řekli jsme si také, že už samotný název tagu určuje jeho význam. Ukázka použití p-tagu pro zápis odstavce:

<p>Toto je text našeho odstavce</p>

U odstavce je zápis v jazyce HTML jednoduchý. Existují však HTML elementy, kde si tímto nevystačíme. Občas potřebujeme přidat další informace, aby prohlížeč věděl, jak se má chovat. Příkladem může být odkaz. Pokud na odkaz klikneme, webový prohlížeč nám zobrazí jeho cílovou stránku. Dodatečnou informaci zde představuje adresa cílové stránky. Bez ní by postrádal odkaz svůj význam. Pojďme se nyní podívat na to, jak dodatečné informace zapíšeme v jazyce HTML.

### Atributy a jejich zápis

Pokud chceme rozšířit možnosti nějakého HTML elementu, můžeme to v jazyce HTML udělat pomocí tzv. **atributů**. Atribut představuje dodatečnou hodnotu, kterou přiřazujeme k počátečnímu tagu.



Atribut se skládá ze dvou částí:

* **název atributu** - Název atributu nám říká, co vlastně u HTML elementu nastavujeme.
* **hodnota atributu** - Konkrétní hodnota měněné vlastnosti.

Již zmíněný odkaz zapisujeme v jazyce HTML pomocí tagu <a>. Jeho atribut href (zkratka hypertext reference) nám určuje, kam se uživatel po kliknutí na něj dostane. V tomto případě na stránku https://engeto.cz. Tag s atributem pak zapíšeme jako <a href="https://engeto.cz">.

HTML tagy mohou obsahovat i více atributů. Dokážeme tak nastavit další chování HTML elementu. Odkazům a dalším atributům se budeme blíže věnovat ke konci dnešní lekce.

<a href="https://engeto.cz">Odkaz</a>

Atributy HTML jsou důležité, protože poskytují další informace o prvku HTML, například o jeho chování nebo vzhledu nebo o tom, jak má spolupracovat s ostatními prvky na stránce.

Atributy lze použít k:

* přidání metadat k prvku

<meta name="keywords" content="gardening, tips, tricks, plants, flowers, vegetables">

* určení alternativního textu pro obrázky

<img src="cesta-k-obrazku" alt="alternativni text" />

* definování odkazů nebo cílů

<a href="https://engeto.cz">Odkaz</a>

* a k mnoha dalším účelům.

Pomocí atributů v jazyce HTML mohou vývojáři vytvářet složitější a interaktivnější webové stránky, které uživatelům poskytují lepší zážitek. Kromě toho mohou atributy využívat vyhledávače a další nástroje, které pomáhají indexovat a pochopit obsah webové stránky, což může zlepšit její viditelnost a přístupnost.

# Základy formátování textu

## Odstavce

První HTML element, který si ukážeme, je odstavec. Jedná se o základní element sloužící k formátování textu. Oceníme ho především u delších textů, kde členění do odstavců napomáhá k lepší přehlednosti. Význam odstavců určitě znáte ze svých mladých let. Ve slohových pracích na základní škole se používal ke stejným účelům.

### Zápis v jazyce HTML

Pro zápis odstavce slouží tag <p>. Znak p je zástupný název pro odstavec (z anglického slova **paragraph**).

Původně se jednalo o nepárový tag, ale postupně se začala používat i značka </p>, která tag ukončuje. Novější specifikace už ukončovací tag přímo vyžadují.

<p>Toto je odstavec textu, který se zobrazí na stránce. Může obsahovat informace o webové stránce, úvod do obsahu nebo jiné typy informací.</p>

<p>Zde je další odstavec textu. Může obsahovat další informace nebo být součástí samostatné sekce webové stránky.</p>

### Praktická ukázka vlozit do htlm

<p>V chaloupce u lesa je jeden domeček, v něm bydlí s kocourem kouzelný dědeček. Kouzelný dědeček má tajemnou moc, vypráví pohádky na dobrou noc.</p>

<p>Skřítkové je dětem po světě roznáší, pohádky roznáší, ve kterých nestraší. Skřítkové na další pohádku čekají, "Dědečku, vyprávěj!", na něho volají.</p>

<p>Sotva ten dědeček pohádku pověděl, skřítek ji popadl a k dětem uháněl.</p>

<p>Za devatero horami až na konci světa, prý černý krkavec do polí létá, lišky tu dávají dobrou noc, jezevec ani nevystrčí nos a ještě kousek dál za černou tmou už běží skřítek s pohádkou.</p>

<p>Venku už je tma a děti jdou spát, tu pohádku s námi můžeš poslouchat!</p>

<p>Adélka byla za celý den moc unavená a už se těšila do postýlky na další pohádku. Ještě si umyla zoubky a popřála mamince a tatínkovi dobrou noc. Pak už se jen přitulila ke svému polštářku, který ji dnes voněl rozkvetlou loukou, a poslouchala tichý hlas pohádkového skřítka…</p>

A screenshot of a computer

Description automatically generated

## Ruční zalomení řádku

Zatím jsme si ukázali, jak lze formátovat delší text do odstavců. Prohlížeče odstavce zobrazují tak, že je od sebe vizuálně oddělují pomocí mezery: <p> </p>

Mohou však nastat situace, kdy potřebujeme jenom zalomit řádek textu na konkrétním místě a ne začínat nový odstavec. K tomuto účelu slouží nepárový tag <br>. br je zástupný název pro zalomení (z anglického slova **break**). Text obsahující tento HTML element bude pokračovat na dalším řádku hned za místem jeho výskytu.

### raktická ukázka

Krásným příkladem, na kterém se dá ukázat ruční zalomení textu, jsou verše básně. Jednotlivé verše se vyskytují vždy na novém řádku, přičemž se stále jedná o text stejného odstavce. Odstavce pak představují jednotlivé sloky básně.

p>Zemřela matka a do hrobu dána,<br> siroty po ní zůstaly;<br> i přicházely každičkého rána<br> a matičku svou hledaly.</p>

<p>I zželelo se matce milých dítek;<br> duše její se vrátila<br> a vtělila se v drobnolistí kvítek,<br> jím mohylu svou pokryla.</p>

<p>Poznaly dítky matičku po dechu,<br> poznaly ji a plesaly;<br> a prostý kvítek, v něm majíc útěchu,<br> mateřídouškou nazvaly. -</p>

<p>Mateřídouško vlasti naší milé,<br> vy prosté naše pověsti,<br> natrhal jsem tě na dávné mohyle -<br> komu mám tebe přinésti?</p>

<p>Ve skrovnou já tě kytici zavážu,<br> ozdobně stužkou ovinu;<br> do šírých zemí cestu ti ukážu,<br> kde příbuznou máš rodinu.</p>

<p>Snad se najde dcera mateřina,<br> jí mile dech tvůj zavoní;<br> snad že i najdeš některého syna,<br> jenž k tobě srdce nakloní!</p>

A screenshot of a computer

Description automatically generated

### Vyhněte se nesprávnému použití

Ruční zalomení řádku by se mělo používat pouze tam, kde to dává význam. Před chvilkou jsme si ukázali, že <br> zalomí text na další řádek. Pokud ale tento tag použijeme několikrát za sebou, můžeme tím vizuálně napodobit funkci odstavce. Mezi dvěma odstavci je totiž vždy viditelná mezera. Mezery docílíme ale i pomocí <br><br>. Nicméně je tu jeden podstatný rozdíl. My pomocí <br><br> **nevytváříme nový odstavec, ale jen tu mezeru**.

Nesnažme se tedy nahradit funkci odstavce tím, že umístíme více tagů <br> za sebou. Později si v kurzu ukážeme, jak lze zvětšit odsazení mezi dvěma odstavci. K tomuto účelu je nutné, aby každý odstavec byl definovaný správně. Tzn. pomocí tagů, které jsou k tomu určené <p></p> a ne pomocí násobného použití <br>.

<!-- NESPRÁVNÉ POUŽITÍ: zalomení řádku by nemělo nikdy nahrazovat funkci odstavce -->

<p>V chaloupce u lesa je jeden domeček, v něm bydlí s kocourem kouzelný dědeček. Kouzelný dědeček má tajemnou moc, vypráví pohádky na dobrou noc.<br><br>Skřítkové je dětem po světě roznáší, pohádky roznáší, ve kterých nestraší. Skřítkové na další pohádku čekají, "Dědečku, vyprávěj!", na něho volají.</p>

<!-- SPRÁVNÉ POUŽITÍ: správně definované 2 odstavce -->

<p>V chaloupce u lesa je jeden domeček, v něm bydlí s kocourem kouzelný dědeček. Kouzelný dědeček má tajemnou moc, vypráví pohádky na dobrou noc.</p>

<p>Skřítkové je dětem po světě roznáší, pohádky roznáší, ve kterých nestraší. Skřítkové na další pohádku čekají, "Dědečku, vyprávěj!", na něho volají.</p>

A screenshot of a computer

Description automatically generated

Zatímco prvek HTML <br> může být užitečný pro vytváření zalomení řádků v textu, obecně se nedoporučuje používat jej pro formátování celých odstavců nebo bloků textu. Zde je několik důvodů:

**Přístupnost**: Používání <br> k oddělení řádků textu může ztížit přístup k obsahu zrakově postiženým uživatelům, kteří se spoléhají na čtečky obrazovky. Čtečky obrazovky často oznamují zalomení řádků jako pauzy, což může způsobit, že text bude znít nesouvisle a bude obtížné jej sledovat.

**Úprava**: Používání <br> k formátování textu může v budoucnu ztížit údržbu a aktualizaci vašeho obsahu. Pokud potřebujete provést změnu v textu, možná budete muset ručně upravit zalomení řádků, abyste zajistili, že obsah zůstane čitelný.

**Sémantika**: HTML je navrženo tak, aby poskytovalo sémantický význam obsahu a použití   
k formátování textu může vést k obsahu, který je méně sémanticky smysluplný. Místo toho zvažte použití prvků HTML, jako jsou <p>, <ul> a <ol> ke strukturování obsahu a poskytnutí smyslu textu.

Zatím se řiďte tím, že použití elementu <br> je špatně! Jedná se o element, který je v jazyce HTML spíše z historických důvodů (ještě z dob psacích strojů) a v moderním kódu by se neměl vyskytovat, protože působí problémy při rozšiřování a udržování kódu a navíc často nefunguje tak, jak byste si mysleli. Proto **vždycky** zkuste přijít s možností, jak se bez tohoto elementu obejít.

## Zvýraznění a zdůraznění textu

V psaných textech občas nastávají situace, kdy chceme "vypíchnout" nebo odlišit určité části textu. Jazyk HTML s tímto počítá. K dispozici zde máme hned několik HTML tagů.

### Zvýraznění a zdůraznění textu

Všechny tagy, které si nyní budeme ukazovat, jsou párové - vždy pomocí nich označujeme vybranou část textu.

* Tag <strong> slouží k zvýraznění důležitého textu. strong znamená v angličtině silný.
* Tag <em> slouží k zdůraznění vybrané části textu. em je zástupný název pro důraz (z anglického slova emphasis).

Při zobrazení v prohlížeči pak oba elementy snadno rozlišíme. Každý z nich je zvýrazněný trochu jiným stylem:

* Text uvnitř tagů <strong></strong> je v prohlížeči zvýrazněn **tučným písmem**.
* Text uvnitř tagů <em></em> je v prohlížeči zvýrazněn kurzívou.

<strong>zvýraznění důležitého textu</strong>

<em>zdůraznění vybrané části textu</em>

V praxi to pak vypadá takto:

<p><strong>Město Brno</strong> je počtem obyvatel i rozlohou <em>druhé největší</em> město v České republice, <em>největší město</em> na Moravě a <em>bývalé hlavní město</em> Moravy.</p>

A screenshot of a computer

Description automatically generated

### Jiná alternativa pro tučný text a kurzívu

Můžeme se setkat ještě s variantou zvýraznění, kdy **tučný text** zapisujeme pomocí tagů <b></b> a kurzívu pomocí tagů <i></i>.

* Tag <b> slouží k vykreslení tučného textu. b je zástupný název pro tučný text (z anglického slova bold).
* Tag <i> slouží k vykreslení textu kurzívou. i je zástupný název pro kurzívu (z aglického slova italic).

I když prohlížeč zobrazuje obě varianty stejně, je mezi tímto a předchozím zápisem rozdíl. Zápis pomocí tagů <b></b> pouze říká, že vybraný text bude tučný. Nic víc. Zápis pomocí tagů <strong></strong> nese navíc informaci, že se jedná o důležitý (významný) text. Přidává tedy vybranému textu kromě vzhledu i další význam. Odborně se tato vlastnost nazývá jako **sémantický význam textu**. Stejný rozdíl platí mezi zápisem pomocí tagů <i></i> a <em></em>, kde <i> představuje pouze text vykreslený kurzívou, kdežto <em> je text s důrazem.

<b>slouží k vykreslení tučného textu</b>

<i>slouží k vykreslení textu kurzívou</i>`

### Význam sémantiky textu

Možná vám to přijde až zbytečné, takto rozlišovat text. Je však nutné si uvědomit, že váš text nebude číst pouze člověk. Internetové vyhledávače (Google, Seznam,...) automaticky procházejí obsah, aby jej pak mohli zařadit do výsledků svých vyhledávání. Sémantický význam je pro ně velice důležitý. Vhodně zvolené HTML tagy jsou pro ně užitečným pomocníkem při rozpoznávání významu textu.

Druhým argumentem, proč dávat přednost HTML tagům se sémantickým významem, jsou nejen zrakově handikepovaní čtenáři. Tito uživatelé jsou odkázáni na své čtečky obrazovky s hlasovým výstupem. Jejich nástroje přebírají HTML kód a snaží se ho strojovým způsobem přečíst. Pokud například poznají, že část textu je označena jako důležitá (označená pomocí <strong></strong>), tak ji zdůrazní i při čtení.

## Nadpisy

Pokud se podíváme na plnohodnotný dokument, téměř vždy obsahuje nadpisy. Jedná se o prvky na stránce, které slouží k lepšímu strukturování obsahu. Známe je také z novin, kde zastávají roli titulků u jednotlivých článků. Jejich snahou je pomoci čtenáři **získat rychlý přehled** o obsahu textu, který bude následovat. Běžného čtenáře totiž snáze zaujmou výrazné prvky na stránce (což nadpisy rozhodně jsou). A z nich se pak snaží vypozorovat, zda následující obsah stojí za to číst.

Nadpis nemusíme použít pouze ve spojení s textem. Na webu může nadpis uvozovat téměř libovolný obsah (např. video, formulář).

### Zápis v jazyce HTML

Nadpisy zapisujeme v jazyce HTML pomocí párových tagů - vymezují, kde text nadpisu začíná a kde končí. K dispozici máme šest různých druhů nadpisů, které se liší svou úrovní. K zápisu používáme tagy <h1>, <h2>, <h3>, <h4>, <h5> nebo <h6>. Znak h je zástupný název pro nadpis (z anglického slova heading). Číslovka 1 až 6 pak značí úroveň nadpisu.

1. <h1> - Používá se jako hlavní nadpis. Všechny ostatní nadpisy nižší úrovně představují podnadpisy.
2. <h2>
3. <h3>
4. <h4>
5. <h5>
6. <h6> - Nadpis s nejmenší důležitostí.

V prohlížeči od sebe jednotlivé úrovně poznáme podle různé velikosti písma. Nejdůležitější nadpisy bývají zobrazeny jako výrazný prvek na stránce (největší velikost písma, případně nejvýraznější barva). S klesající úrovní nadpisu se velikost písma zmenšuje.

V praxi většinou nevyužijeme všech šest úrovní. Různé úrovně nadpisu, nám však umožňují hezky strukturovat náš obsah. Jednotlivé úrovně by měly jít postupně za sebou. Na následující ukázce je dobře vidět hierarchie textu rozdělená díky vhodně zvoleným nadpisům. Důležitější text následuje pod hlavním nadpisem (<h1>). Podrobnější informace najdeme až pod nadpisy nižší úrovně (<h2>, <h3>).

<h1>Jára Cimrman</h1>

<p>Jára Cimrman je fiktivní postava univerzálního českého génia, vytvořená Jiřím Šebánkem a Zdeňkem Svěrákem.</p>

<p>Poprvé se objevil roku 1966 v rozhlasovém pořadu Vinárna U Pavouka v Československém rozhlase, a to jako řidič parního válce a sochař. Ještě téhož roku však bylo autory a dalšími spolupracovníky založeno celé Divadlo Járy Cimrmana, které dodnes uvádí hry přisuzované Járovi Cimrmanovi a zároveň zpracovávalo fiktivní život a dílo člověka, jenž je popisován jako génius a všeuměl. Jára Cimrman má v české kultuře značný ohlas a dokonce získal nejvyšší počet hlasů v celonárodní anketě Největší Čech (2005).</p>

<h2>Umělecké zpracování</h2>

<h3>Vznik postavy</h3>

<p>První zpráva o existenci českého velikána Járy Cimrmana se objevila 16. září 1966 v humoristickém pořadu (Nealkoholická) Vinárna U Pavouka v Československém rozhlase. Byl popsán jako řidič parního válce a sochař, neúspěšně vystavující v nafukovacím pavilonu předměty zdeformované jejich přejetím parním válcem. Reportáž vedl Zdeněk Svěrák. Na motivy jeho údajného života a díla byla vytvořena podstatná část repertoáru Divadla Járy Cimrmana.</p>

<p>Jméno Cimrman vymyslel zřejmě Jiří Šebánek podle chomutovského hokejisty Cimrmana, jehož jméno slyšel dotyčný v rozhlase a zalíbilo se mu. Křestní jméno „Jára“ bylo dodáno, aby vzbuzovalo zdání uměleckosti. Jako hlavní předobrazy samotné postavy uvádí autoři dobrušského technologického nadšence a naivistického malíře Aloise Beera (1833–1897).</p>

<h3>Umělecké formy</h3>

<p>Myšlenku založit Divadlo Járy Cimrmana dostali Šebánek, Čepelka, Smoljak a Svěrák v říjnu 1966. Ke hře Akt, která byla pro celovečerní představení příliš krátká, přidali herci ještě fingovanou přednášku o autorovi J. Cimrmanovi, která se následně stala pevnou součástí všech představení a setkala se s značným úspěchem. Herci se tak stylizovali do odborníků (vědců), kteří interpretují dílo „mistra“ jako součást své vědecké práce. Veřejnosti jsou nejznámější autoři her Ladislav Smoljak a Zdeněk Svěrák.</p>

### Komentáře v kódu

V rámci kódování se setkáte i se zakomentovanými částmi kódu, které poznáte podle zelené barvy daného bloku. Zakomentovaný kód ve výsledku není viditelný pro uživatele/čtenáře webu, ale prohlížeč ho "vidí". Zakomentovaný kód vypadá např. takto:

<!-- text, který chceme mít zakomentovaný -->

Když se teď podíváme do kódu v prohlížeči, zadaný text se tu nezobrazuje. Když si ale zobrazíme vývojářské prostředí, kód tu vidíme opět zeleně.

Komentáře se nepíší jen v HTML, ale ve všech jazycích. Slouží jednak pro popis kódu a jeho funkcionalit, ale i při samotném kódování, kdy máte nějaký kód a zkoušíte si, jak bude vypadat v různých zápisech.

#### Tip pro Visual Studio a další textové editory

Každý jazyk má svůj způsob zápisů komentářů, který se liší od těch ostatních. Abychom si to nemuseli pamatovat a také pro jednodušší používání, existuje ve Visual Studiu a ostatních editorech klávesová zkratka **Ctrl+C+K** pro zakomenářování označeného textu a **Ctrl+C+U** pro jeho odkomentářování. Vzykoušejte si to a používejte pro rychlejší psaní kódu.

# Odkazy

## Odkaz a jeho základy

V úvodu dnešní lekce jsme se dozvěděli, že HTML je zkratkou pro **HyperText Markup Language** (hypertextový značkovací jazyk). **HyperText** je metoda procházení internetových stránek, kdy jednotlivé stránky jsou mezi sebou propojeny tzv. **hypertextovými odkazy**. Po kliknutí na odkaz se dostaneme na další stránku. Dostáváme tak síť propojených stránek, kde návštěvník může v ideálním případě procházet celým webem.

Odkaz je tedy velice důležitou součástí webové stránky. V našem kurzu je to první **interaktivní prvek**, na který lze kliknout a prohlížeč poté vykoná nějakou akci. Díky němu dokážeme propojit jednu stránku s druhou. Nemusíme však zůstat pouze u stránek. Pomocí odkazu můžeme ukázat na soubory, které si může čtenář stáhnout. Odkázat můžeme uživatele dokonce i na e-mailovou adresu. Všechny tyto způsoby si názorně ukážeme v této lekci.

### Zápis v jazyce HTML

Odkaz zapisujeme v jazyce HTML pomocí párového tagu <a>. Znak a je zástupný název pro odkaz (z anglického slova anchor).

Text, na který bude návštěvník klikat, uzavíráme mezi počáteční a ukončovací tag.

<a>Text odkazu</a>

Tento zápis však nevytvoří funkční odkaz. Aby bylo zřejmé, kam se má uživatele po kliknutí dostat, musí prohlížeč znát cíl odkazu (např. adresu webové stránky). Pro připojení této informace k odkazu poslouží HTML atribut href (hypertext reference). Jeho hodnota představuje cílovou adresu. Celý element spolu s atributem href pak může vypadat takto:

<a href="cílová adresa">Text odkazu</a>

Následující živá ukázka demonstruje zápis odkazu **Odkaz na Engeto**, který zavede uživatele na adresu <https://engeto.cz>:

A screenshot of a computer

Description automatically generated

### Další případy použití

Odkaz může být umístěn i uvnitř delšího textu.

<p>Vzdělávání v 21. století musí být inovativní. V <a href="httsp://engeto.cz">ENGETO Academy</a> se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

A screenshot of a computer

Description automatically generated

Část odkazu, na kterou bude návštěvník v prohlížeči klikat, nemusí být nutně jen text. Mezi tagy <a></a> lze v podstatě uzavřít cokoli. Až se naučíme, jak v HTML zapsat obrázek, dokážeme podobným způsobem vytvořit klikací obrázky.

## Otevření do nového okna

Takto vypadá obyčejný odkaz:

<a href="https://engeto.cz/">Odkaz na Engeto</a>

Když na takový odkaz v prohlížeči klikneme, otevře se nám webová stránka https://engeto.cz. Technicky řečeno budeme přesměrováni na novou cílovou stránku. Existuje však způsob, jak prohlížeči sdělit, aby nám cílovou stránku otevřel v novém okně. Pojďme si ukázat, jak na to.

### Otevření do nového okna

Otevírání odkazu do nového okna probíhá v moderních prohlížečích tak, že se nám v rámci akutálního prohlížeče otevře **nová záložka**. Obsahem záložky bude daná cílová stránka. Starší prohlížeče neměly funkci záložek, obsah se otevíral do nového okna. Záložky slouží k lepší organizaci a snadněji se mezi nima přepíná.

Vyplatí se ale otevírat odkazy do nového okna? Jsou zajisté situace, kdy ano. Pokaždé když klikneme na odkaz, opustíme původní stránku a otevře se nám stránka nová. Otevírání odkazu do nového okna nám zajístí, že původní stránka zůstane stále otevřená a cíl odkazu se nám objeví v novém okně (popř. záložce). Toto může být užitečné hlavně v situacích, kdy by odkaz návštěvníka odvedl pryč z našeho webu.

**TIP:** Pokud máš myš s kolečkem uprostřed, můžeš kliknutím kolečka na odkaz otevřít cílovou stránku v novém okně. Ne každý uživatel webu tento trik zná. Pokud nechceš opustit původní web, používej tuto funkci při otevírání odkazu do nového okna.

### Zápis v jazyce HTML

Informaci o tom, kde se má cíl odkazu otevřít, změníme pomocí nového HTML atributu target. Atribut může mít jednu z následujícíh hodnot:

* \_self - Výchozí hodnota. Pokud má atribut tuto hodnotu, odkaz se otevře do stejného okna.
* \_blank - Pokud má atribut tuto hodnotu, odkaz se otevře do nového okna.

Pokud atribut target nepoužijeme, odkaz se otevře ve stejném okně. Platí tedy, že běžný odkaz typu:

<a href="https://engeto.cz/">Odkaz na Engeto</a>

bude fungovat stejně jako v případě následujícího zápisu.

<a href="https://engeto.cz/" target="\_self">Odkaz na Engeto</a>

Pokud chceme mít odkaz, kde se cílová adresa bude otevírat do nového okna, musíme připojit atribut target="\_blank". Vše si můžeme vyzkoušet na následující živé ukázce:

## E-mailové odkazy

Cílem odkazu nemusí být jen webová stránka. Cílem může být i e-mailová adresa. Pokud návštěvník klikne na podobný odkaz, prohlížeč to pozná a pokusí se otevřít dostupného e-mailového klienta (program na psaní emailových zpráv). To může být užitečné v situacích, kdy chceme čtenářům usnadnit práci. Takto jim stačí kliknout na odkaz a rovnou mohou psát zprávu na daný email. Odpadá tak různé přepisování nebo kopírování e-mailové adresy.

Aby prohlížeč věděl, že má tu čest s e-mailovým odkazem, musíme v cílové adrese uvést magické slovo mailto:. Ihned poté následuje e-mailová adresa. Celý zápis vypadá takto:

<a href="mailto:info@engeto.com">Kontaktujte nás</a>

Uvedený odkaz se na pohled neliší od ostatních "obyčejných" odkazů. Jediný rozdíl je v tom, že cílem odkazu není webová stránka, nýbrž emailová adresa, kterou se prohlížeč pokusí otevřít v dostupném programu.

# V čem zapisujeme kód

## Vývojová prostředí

Nejen HTML, CSS, ale i JavaScript a další programovací jazyky zapisujeme v tzv. **vývojovém prostředí** (zkratka IDE, anglicky Integrated Development Environment). Těchto IDE existuje celá řada. V našem kurzu budeme pracovat v prostředí **VS Code**, které je zdarma a využívá ho celá řada programátorů.

Kromě tohoto programu existuje např. také velmi rozšířený Webstorm, Sublime, Atom a další. Některé IDE jsou určené primárně pro určité programovací jazyky (např. PHPstorm nebo IntelliJ IDEA). Ne všechna prostředí jsou ale bezplatná. Postupně si můžete vyzkoušet více IDE a vybrat si ten, který vám bude nejvíce vyhovovat.

Vývojové prostředí umožňuje kromě psaní kódu také spoustu dalších věcí, kterým se budeme postupně věnovat. Na začátek zmíním různé pluginy, které programátorům zrychlují a zjednodušují práci. Např. automatická kontrola chyb nebo chytré vyhledávání napříč kódem. Postupně si budeme tyto pluginy ukazovat.

**Stažení VS Code**: <https://code.visualstudio.com/>

## VSC rozšíření

**Extensions** nebo rozšíření ve VSC jsou doplňky, pomoci kterých můžeme rozšířit funkce editoru.

### 1. Prettier

Prettier je populární formátovač kódu, který dokáže konzistentním a standardizovaným způsobem automaticky formátovat kód.

Výhodou použití rozšíření Prettier je schopnost udržet konzistentní styl kódu v celém projektu, zlepšit čitelnost kódu a ušetřit čas tím, že se vyhneme ručnímu formátování kódu.

Podrobný popis jak si nainstalovat a nakonfigurovat Prettier najdete v následujícím videu.

### 2. Live Server

Live Server je další populární rozšíření, které umožňuje vytvořit lokální server a zobrazit náhled webových stránek v reálném čase. Rozšíření automaticky obnovuje prohlížeč při provádění změn v kódu HTML, CSS nebo JavaScriptu, čímž poskytuje pohodlné vývojové prostředí.

Ulohy:

Na následující ukázce je text, který je vizuálně rozdělen do dvou částí. Situace vypadá, jako by obsahoval dva odstavce. Nicméně není tomu tak. Přejděte do editoru. Opravte kód s použitím tagů <p>, </p> a vytvořte tak **dva skutečné odstavce**.

<p>Vzdělávání v 21. století musí být inovativní. V ENGETO Academy se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.<br /><br />Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

## Úkol č.2

Na následujícím příkladu máme větu, kde chceme zvýraznit některá slova. Text je pak čitelnější a čtenář ví, které pojmy jsou ve větě důležité. Použijte HTML element pro zdůraznění textu a zvýrazněte jméno **Praotec Čech**.

<p>Praotec Čech je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována Čechy.</p>

## Úkol č.3

Z této lekce už známe, že nadpisy nám pomáhají strukturovat obsah. Úrovně nadpisů by měli jít za sebou. Na následující ukázce je text, kterému však chybí hlavní nadpis. Údělejte v HTML kódu následující úpravy:

1. Na úplný začátek textu přidejte hlavní nadpis Legenda o praotci Čechu.
2. V kódu vidíme několik nadpisů třetí úrovně <h3>. Aby text dával větší význam a jeho čtení bylo přehlednější, přidejte před nadpis Kosmova kronika úplně nový nadpis druhé úrovně s textem Dostupné prameny.

<p>Praotec Čech je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována Čechy.</p>

<h3>Kosmova kronika</h3>

<p>V latinsky psané Kosmově Kronice Čechů se vypráví o příchodu bezejmenného lidu do dnešních Čech.</p>

<h3>Dalimilova kronika</h3>

<p>V česky psané veršované Kronice tak řečeného Dalimila z počátku 14. století se poprvé objevuje motiv Charvátské země, vraždy, výstup na Říp.</p>

## Úkol č. 4

Blok týkající se odkazů je za námi a je tu další úkol. Pro jeho splnění musíš udělat následující kroky:

1. Do následujícího editoru zapiš odkaz. Text odkazu bude ENGETO Academy a cílová adresa bude směřovat na https://engeto.cz.
2. Jakmile bude odkaz hotový, zobrazí se nám jeho náhled v pravé části editoru.
3. Poté odkaz uprav tak, aby se nám jeho cílová adresa otevírala do nového okna.

## Úkol č. 5

Použijte libovolně tyto html tagy h1, h2, h6, p, br, em, strong, v následující ukázce.

Vítejte na mé webové stránce

Toto je odstavec textu. Lze ho formátovat pomocí různých HTML tagů jako zdůrazněný text a tučný text.

Nadpis 2

Můžete použít nadpisy k organizaci obsahu. Existují šest úrovní nadpisů, od největšího po nejmenšího.

Na svou webovou stránku můžete také přidat odkazy pomocí tagu anchor tag. Například můžete odkazovat na Google.

Nakonec můžete přidávat řádkové zlomy pomocí tagu. To je užitečné, pokud chcete rozdělit odstavec textu na menší kousky.

## Bonusový úkol

Vytvoř jednoduchou stránku, která bude splňovat tyto požadavky:

1. Stránku vytvořte v interaktívnim editoru.
2. Přidej 3 odstavce textu se zvýrazněnými (libovolnými) částmi textu pomocí tagů strong a em. Každý odstavec bude začínat nadpisem s tagem **H2**.
3. Přidej odkaz s atributy href a target s hodnotou otevření v novém okně.
4. Vytvoř alespoň jeden komentář.

# Základní tagy

## Základní (nečíslovaný) seznam

Už zvládneme vložit text do odstavců a vytvářet jednotlivé kapitoly pomocí nadpisů. V této lekci si ukážeme možnosti organizace pomocí dalších prvků.

Nejdříve se podíváme na **seznamy**, které se nám hodí hlavně v situacích, kdy potřebujeme zobrazit výčet nějakých položek. Obrovskou výhodou je snadná čitelnost. Takto vypadá nečíslovaný nákupní seznam, kde jsou jednotlivé položky označeny puntíkem:

* rohlík
* šunkový salám
* čokoláda
* mouka
* pivo

Takový zápis je lépe čitelný než následující: rohlík, šunkový salám, čokoláda, mouka a pivo.

### Zápis v jazyce HTML

V jazyce HTML vytvoříme nečíslovaný seznam pomocí párového tagu <ul>. ul je zástupný název pro nečíslovaný seznam (z anglického výrazu unordered list). Element <ul></ul> **nemůže obsahovat** rovnou samotný text. Tento element nám pouze ohraničuje, kde seznam začíná a kde končí. Nic víc. Dovnitř <ul></ul> pak umístíme HTML zápis jednotlivých položek seznamu, a to pomocí párového tagu <li>. li je zástupný název pro položku seznamu (z anglického výrazu list item). Obsahem položky už může být text.

Pokud se pokusíme uvedený nákupní seznam vyjádřit v jazyce HTML, jeho kód bude vypadat následovně:

<ul>

<li>rohlík</li>

<li>šunkový salám</li>

<li>čokoláda</li>

<li>mouka</li>

<li>pivo</li>

</ul>

## Číslovaný seznam

Nečíslovaný seznam používá k označení svých položek černé puntíky. Pokud ale máme položek více nebo chceme zdůraznit jejich pořadí (např.: 1., 2., 3.), je lepší zvolit **číslovaný seznam**. Ten namísto černého puntíku označuje položky číslem.

### Zápis v jazyce HTML

Číslovaný seznam zapíšeme v jazyce HTML pomocí párového tagu <ol>. ol je zkratka pro číslovaný seznam (z anglického výrazu ordered list). Jednotlivé položky pak vytvoříme stejným způsobem, jako u nečíslovaného seznamu - k jejich zápisu použijeme tag <li>.

Číslovaný seznam má jednu užitečnou vlastnost. Jeho položky se číslují automaticky při výpisu v prohlížeči a číslo se nemusí nikam do textu vypisovat. Pokud třeba v budoucnu přeházíme pořadí položek, seznam se sám přečísluje a odpadá tak ruční opravování pořadí.

Zápis v jazyce HTML vypadá takto:

<ol>

<li>Sběr hroznů</li>

<li>Odzrnění</li>

<li>Lisování</li>

<li>Kvašení (fermentace)</li>

<li>Jablečno - mléčná fermentace</li>

<li>Školení vína</li>

<li>Lahvování</li>

</ol>

## Obrázek a jeho základy

Všechno, co jsme se až doteď naučili, má jednu společnou vlastnost. Každý zmíněný element (ať už odstavec, nadpis, odkaz či seznam) má textový základ. Web se ale neskládá jenom z textu. Znáš to z vlastních zkušeností. Stránky bývají také plné obrázků.

### Zápis v jazyce HTML

Většinu probíraných elementů jsme vyjádřili pomocí párových tagů. Počáteční a koncový element ohraničoval text a dával mu určitý význam (např. nadpis). V případě obrázku není potřeba nic ohraničovat. Obrázek zapíšeme v jazyce HTML pomocí nepárového tagu <img />. img je zkratka pro obrázek (z anglického slova **image**).

Tag <img> má povinný atribut src (zkratka z angl. **source** = zdroj). Jeho hodnota představuje **adresu umístění obrázku**. Bez tohoto atributu by prohlížeč nevěděl, co vlastně chceme zobrazit. Obrázek je vždy uložen jako soubor (typicky jako soubor s koncovkou **.jpg** nebo **.png**). Adresa obrázku nám pak říká, kde tento soubor v rámci internetu najdeme.

Příklad zápisu v HTML:

<img src="logo.jpg">

Druhý příklad zápisu v HTML:

<img src="https://engeto.cz/logo.jpg">

Ukázka obrázku zapsaného v jazyce HTML vypadá takto:

<img src="https://www.google.com/images/branding/googlelogo/1x/googlelogo\_color\_272x92dp.png">

V tomto případě je důležité upozornit na to, že v atributu **src** používáme odkaz na obrázek uložený na internetu. V našem případě se jedná o logo google, které nalezneme na adrese [www.google.com](http://www.google.com). Když si tuto stránku otevřeme ve vývojářských nástrojích (devtools), uvidíme adresu, na které je obrázek uložený. Tuto adresu jsme si zkopírovali a použili v našem příkladu. Nicméně v editoru (v našem případě VS Code) nemusíme používat jen obrázky z internetu, ale můžeme odkázat na obrázek, který máme uložený u sebe v počítači. To si nyní ukážeme.

**Postup:**

1. Na libovolném místě (např. na ploše) si vytvoříme složku s názvem Engeto.
2. Otevřeme si VS Code.
3. Klikneme na **File** -> **Open Folder** a najdeme složku Engeto, kterou jsme si vytvořili a klikneme na Vybrat složku.
4. Tato složka se nám otevře v editoru. Při najetí myší na tuto složku, se nám zobrazí možnosti přidání souborů. Z možností vybereme **New File** a pojmenujeme jej **index.html**. Tento soubor představuje souhrn všech HTML tagů, které na stránce použijeme. Podle přípony **.html** prohlížeč pozná, o který typ souboru se jedná a otevře jej jako webovou stránku.
5. V souboru **index.html** vytvoříme úvodní část html kódu (hlavičku, tělo a další nezbytnosti). Pomůžeme si tím, že na 1. řádku napíšeme pouze vykřičník a stiskneme klávesu enter. Celá hlavička se nám tu automaticky doplní.
6. Mezi otevírací a uzavírací značky body, doplníme tag <img> s atributem src.
7. Ještě si potřebujeme stáhnout obrázek. Otevřeme si prohlížeč, zadáme adresu [www.google.com](http://www.google.com), kliknutím pravého tlačítka myši na logo google a výběru **Inspect** (česky Prozkoumat prvek) si otevřeme vývojářské nástroje, v záložce **elements** si najdeme tag <img> a najedeme na něj myší. Otevře se nám informace o obrázku i s umístěním obrázku. Klikneme na něj a obrázek se nám otevře v novém okně.
8. Najedeme si myší na obrázek, opět klikneme pravým tlačítkem myši a dáme stáhneme si obrázek do naší složky Engeto. Pokud se vám obrázek automaticky stáhne do složky Uložené (Downloads), stačí když si ho přetáhnete nebo zkopírujete do naší složky Engeto.
9. Vrátíme se do VS code, kde do atributu **src** doplníme název obrázku, v tomto případě **googlelogo\_color\_272x92dp** a nesmíme zapomenout doplnit koncovku .png. Uložíme, vrátíme se do složky Engeto a dvojklikem otevřeme soubor index.html. Tímto si zkontrolujeme, že se nám logo opravdu zobrazuje.
10. Nyní se vrátíme do naší složky s názvem Engeto a dvojklikem levého tlačítka myší si otevřeme soubor **index.html**. Soubor se nám automaticky otevře v okně prohlížeče a můžeme se podívat, jestli jsme vše udělali správně a obrázek se nám tu zobrazuje.

## Atribut alt

Obrázek má kromě atributu src ještě jeden povinný atribut. Jedná se o atribut alt (z angl. **alternative text**), který představuje zástupný text. Jeho hodnota by měla představovat **popis obrázku**.

<img src="zahrada.jpg" alt="Rozkvetlá zahrada plná červených růží">

Atribut alt je užitečný hned v několika případech:

* Jakmile dojde k chybě a obrázek se nepodaří načíst, prohlížeč tento stav zobrazí jako ikonku (popř. červený křížek). Vedle pak uvidíme text alt atributu. Dostaneme tak alespoň nějakou informaci o obrázku, který se měl na daném místě nacházet.
* Hodnotu alt atributu uvidíme také v situaci, kdy načítání obrázku trvá příliš dlouho, např. z důvodu pomalého internetového připojení.
* Zrakově hendikepovaní uživatelé používají při procházení webu speciální čtečku (odečítač obrazovky). Zástupný text alt atributu je pro ně jediná cesta, jak zjistit, co se na obrázku nachází.
* Pro internetové vyhledávače (Google, Seznam,...) jsou obrázky dalším zdrojem informací. Aby bylo možné vůbec rozpoznat, co se na obrázku nachází, je důležité mít správně nastavené alt atributy.

Následuje ukázka nefunkčního obrázku s nastaveným alt atributem

<img src="zahrada.jpg" alt="Rozkvetlá zahrada plná červených růží">

Pokud je obrázek v pořádku, atribut alt nevidíme. I přesto je ale vhodné jej používat.

<img src="https://www.google.com/images/branding/googlelogo/1x/googlelogo\_color\_272x92dp.png" alt="Logo společnosti Google">

### Prázdný atribut alt

Pokud obrázek neposkytuje návštěvníkům žádnou smysluplnou informaci, můžeme ponechat alt atribut prázdný. Takový případ může nastat např. u článku, kde je pouze ilustrační fotografie, která nepopisuje nic konkrétního. Pro uživatele čteček tedy neposkytuje žádnou relevantní informaci jen zahlcuje uživatele nadbytečnými informacemi. Jak už bylo zmíněno výše, alt je povinný atribut, nemůžeme ho tedy vynechat, ale v případě potřeby ho ponecháme prázdný alt="".

<img src="ikonka.png" alt="">

## Základní tabulka

Tabulka je element, který slouží k zobrazení dat v přehledné formě. Díky její struktuře (řádky, sloupce) se v ní lidé dobře orientují. Je to tedy vhodný nástroj i pro zobrazení většího množství dat.

### Zápis v jazyce HTML

Základní tabulka se v jazyce HTML skládá z elementů <table>, <tr> a <td>.

* <table> - Tento párový tag označuje, kde celá tabulka začíná a kde končí. table je označení pro tabulku (z anglického slova **table**).
* <tr> - Řádek tabulky zapíšeme pomocí párového tagu <tr>. tr je zkratka z angl. **table row**.
* <td> - Buňky tabulky vyjádříme pomocí párového tagu <td>. td je zkratka z angl. **table data cell**.

Pro vytvoření tabulky musíme pochopit, jak tyto tři elementy vhodně poskládat dohromady. Potřebný zápis pak bude mít strukturu podle následujícího vzoru:

* tabulka
  + řádek
    - buňka
    - buňka
    - buňka
  + řádek
    - buňka
    - buňka
    - buňka

### Ukázka jednoduché tabulky

Následuje ukázka, která obsahuje jednoduchou tabulku se třemi řádky a třemi sloupci. Její HTML zápis splňuje zmíněné pravidlo, že neprve zapisujeme řádky a až v rámci řádků definujeme jednotlivé buňky. I přesto, že u zobrazené tabulky nevidíme grafické rámečky, můžeme si povšimnout, jak se data hezky strukturují do společných sloupců:

<table>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

<td>37 let</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Pardubice</td>

<td>30 let</td>

</tr>

<tr>

<td>Ondřej Vlezdopole</td>

<td>Staré Město u Uherského Hradiště</td>

<td>29 let</td>

</tr>

<tr>

<td>Martin Pešout</td>

<td>Brno</td>

<td>34 let</td>

</tr>

</table>

### Ukázka tabulky se zvýrazněnou mřížkou

Až se v navazujícím kurzu naučíme přiřazovat elementům vlastní styl, budeme schopní zvýraznit mřížku tabulky. Stejný HTML zápis pak může vypadat takto:

HTML:

<table>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

<td>37 let</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Pardubice</td>

<td>30 let</td>

</tr>

<tr>

<td>Ondřej Vlezdopole</td>

<td>Staré Město u Uherského Hradiště</td>

<td>29 let</td>

</tr>

<tr>

<td>Martin Pešout</td>

<td>Brno</td>

<td>34 let</td>

</tr>

</table>

Css.

table {

border-collapse: collapse;

}

td {

padding: 5px 8px;

border: 1px solid #ccc;

}

## Složitější tabulka

Tabulka může obsahovat celou řadu dalších elementů. Díky tomu dokážeme vytvořit složitější strukturu a vyjádřit mnohem víc než u základní tabulky. Velice zajímavou možnost poskytují hlavičky tabulek, protože ty v sobě nesou informace o údajích v jednotlivých sloupcích. Podíváme se teď na to, jak lze data v tabulce popsat.

Všechny elementy, které si tu ukážeme, jsou nepovinné.

### Hlavička tabulky

Buňky tabulky jsou vždy definované uvnitř řádku. Samotná buňka však nemusí být zapsaná pouze pomocí tagu <td>. Existuje ještě jeden typ, pomocí kterého lze vytvořit hlavičku. Hlavičková data označíme pomocí tagu <th>. th je zkratka z angl. table header cell.

Pojďme si to ještě krátce shrnout. Řádek tabulky může v sobě nést 2 různé druhy buněk:

* <th> - obsahuje informaci pro hlavičku,
* <td> - obsahuje v sobě data tabulky.

Prohlížeč obsah tagu <th> zobrazí výrazněji. Standardně tučným písmem zarovnaným na střed.

<table>

<tr>

<th>Jméno</th>

<th>Bydliště</th>

<th>Věk</th>

</tr>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

<td>37 let</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Pardubice</td>

<td>30 let</td>

</tr>

<tr>

<td>Ondřej Vlezdopole</td>

<td>Staré Město u Uherského Hradiště</td>

<td>29 let</td>

</tr>

<tr>

<td>Martin Pešout</td>

<td>Brno</td>

<td>34 let</td>

</tr>

</table>

### Nadpis tabulky

V ideálnim případe by měla tabulka obsahovat nádpis (hlavičku), který zapisujeme pomoci tagu <caption> . V HTML zápisu nasleduje za tagem <table> a zobrazuje se jako text nad tabulkou. Úlohou nádpisu je poskytnou informace o obsahu tabulky, který je zejména pro hendikepované uživatele. Procházení tabulky odečítačem obrazovký může byt v případe několik set řádku nároční a proto nádpis poskytuje relevantní informace.

<table>

<caption>Zaměstnanci Engeta</caption>

<tr>

<th>Jméno</th>

<th>Bydliště</th>

<th>Věk</th>

</tr>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

<td>37 let</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Pardubice</td>

<td>30 let</td>

</tr>

<tr>

<td>Ondřej Vlezdopole</td>

<td>Staré Město u Uherského Hradiště</td>

<td>29 let</td>

</tr>

<tr>

<td>Martin Pešout</td>

<td>Brno</td>

<td>34 let</td>

</tr>

</table>

## Doplňkové elementy tabulky

V rámci tabulky existují ještě **další 3 elementy**, které lze v tabulce použít. Jedná se o tagy:

* <thead> = sada řádků definujících záhlaví sloupců tabulky
* <tbody> = tělo tabulky
* <tfooter> = sada řádků shrnujících sloupce tabulky

Když se podíváme na některou již existující tabulku pomocí DevTools, uvidíme, že prohlížeč si sám tyto elementy do kódu doplnil. Ne vždy však doplní všechny tyto elementy. Např. v ukázce v Codepenu výše, prohlížeč doplnil pouze <tbody>, ale nerozpoznal a nedoplnil <thead>.

Tyto elementy v tabulce existují především pro **lepší sémantický význam**, tzn. že pomocí těchto elementů dokážeme přesněji určit hlavičku, tělo a patičku tabulky.

Používání těchto elementů není nezbytně nutné, tabulka nám bude fungovat i bez těchto elementů. Pokud bychom však chtěli tabulku přizpůsobit i lidem, kteří pro prohlížení webu **používají čtečku**, bylo by vhodnější tyto elementy doplnit, aby čtečka rozpoznala jednotlivé části tabulky a uživatelům podala lepší přehled o řádcích a sloupcích.

## Ukázka tabulky s použitím doplňkových elementů

<table>

<thead>

<tr>

<th>Položka</th>

<th>Částka</th>

</tr>

</thead>

<tbody>

<tr>

<td>Nákup potravin</td>

<td>551 Kč</td>

</tr>

<tr>

<td>Oběd v restauraci</td>

<td>150 Kč</td>

</tr>

<tr>

<td>Plavání</td>

<td>140 Kč</td>

</tr>

<tr>

<td>Pojištění auta</td>

<td>1135 Kč</td>

</tr>

</tbody>

<tfoot>

<tr>

<th>Celkem</th>

<th>1976 Kč</th>

</tr>

</tfoot>

</table>

# Blokové vs. řádkové elementy

## Typy elementů

V předchozích kapitolách jsme objevili spoustu HTML elementů. Vybavuješ si jejich chování při zobrazení v prohlížeči? Některé elementy mohou být součástí textu a jiné naopak vytváří nové sekce (kolem nich se automaticky zalamují řádky). Pojďme se na toto rozdělení podívat blíže.

## Blokové elementy

Blokový element je typ HTML elementu, který je v prohlížeči zobrazeny jako blok. Představit si to lze tak, že kolem podobného elementu máme neviditelný obdélník v celé dostupné šířce.

### Účel elementů

HTML blokové elementy se používají k uspořádání rozvržení webové stránky a vytváření vizuálně odlišných sekcí obsahu. Obvykle obsahují další prvky, jako je text, obrázky, další značky HTML a blokové nebo inline elementy. Ovlivňují také tok obsahu na webové stránce, protože blokové elementy ve výchozím nastavení zabírají celou dostupnou šířkou a proto jsou naskládány svisle na sebe.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Cumque exercitationem molestias excepturi recusandae accusamus facilis nemo atque.

Příklady HTML blokových prvků zahrnují nadpisy (h1, h2, h3, atd.), odstavce (p), seznamy (ul, ol), tabulky (table) a divy (div).

Blokové prvky mohou obsahovat další blokové prvky nebo inline prvky. Často se používají k uspořádání rozvržení webové stránky a vytváření vizuálně odlišných sekcí obsahu.

### Zalomeni řádku

To znamená, že jakýkoli obsah, který následuje za blokovým elementem, se zobrazí na novém řádku pod ním. Před i za takovým blokovým elementem se nám **vždy zalamují řádky**. Blokové elementy tak nikdy nemohou být součástí textu, protože pokaždé text rozdělí do více částí.

<div>

<div>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</div>

<div>

Cumque exercitationem molestias excepturi recusandae accusamus facilis nemo

atque.

</div>

</div>

**Výsledek příkladu:**

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Cumque exercitationem molestias excepturi recusandae accusamus facilis nemo atque.

### Výška a šířka

Výchozí šířka blokového elementu je **100 % šířky dostupné v jejim rodičovském elementu, ve kterém se daný element nacházi**.

// V tomto příklade je <ul> rodičovský element pro <li> elementy, které označujeme jako potomky.

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

To znamená, že blokový element zabíra celou šířku obrazovky nebo svého rodičovského elementu. Nicméně, šířka blokových prvků může být změněna pomocí CSS stylů.

Blokové elementy nemají výchozí výšku stanovenou prohlížečem. **Výška blokového prvku je určena obsahem uvnitř** něj a jakýmkoli CSS styly aplikovaným na element. Pokud není výška určena pomocí CSS, blokový element si upraví svou výšku tak, aby odpovídala jeho obsahu.

### Úprava výšky a šířky

Výchozí šířka může být změněna pomocí vlastnosti width. Výška může být změněna pomoci vlastnosti height.

<div>

<div style="width: 150px; height: 150px">Lorem ipsum dolor sit amet, consectetur adipisicing elit.</div>

<div style="width: 200px; height: 200px">

Cumque exercitationem molestias excepturi recusandae accusamus facilis nemo

atque.

</div>

</div>

A screenshot of a computer

Description automatically generated

### Seznam blokových elementu

Seznam nejčastěji používaných blokových elementů:

* <div>
* <p>
* <h1>-<h6>
* <ul>
* <ol>
* <li>
* <table>
* <tr>
* <td>

Souhrní seznam blokových elementů najdete na stránce [mdn web docs](https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements#elements).

Následuje živá ukázka, kde je zvýrazněna oblast, kterou každý prvek zabírá:

<h1>Nákupní seznam</h1>

<p>Až půjdeš do města, nakup prosím následující potraviny. Klidně přidej do seznamu další věci, které nám schází na víkend.</p>

<ul>

<li>rohlík</li>

<li>šunkový salám</li>

<li>čokoláda</li>

<li>mouka</li>

<li>pivo</li>

</ul>

Css:

h1, h2, p, ul, li {

border: 1px dotted #FF6363;

}

## Řádkové elementy

Řádkový nebo také inline element, je typ HTML elementů, který je uzpůsoben tak, aby byl v souladu (inline) s okolním obsahem. To znamená, že element nebude zalomovat řádek ale bude plynut s textem a okolními inline elementy.

### Účel elementů

Účelem inline elementů v HTML je **formátovat text uvnitř řádku obsahu**. Inline elementy se obvykle používají k **stylizaci nebo zdůraznění konkrétních slov nebo frází v odstavci, jako jsou tučné nebo kurzívou psané texty**.

<p>Lorem <em>ipsum</em> dolor sit amet, <strong>consectetur<strong> adipiscing elit</p>

**Výsledek příkladu:**

Lorem ipsum dolor sit amet, **consectetur adipiscing elit**

### Zalomeni řádku

Inline elementy **nevytvářejí nové řádky ani zlomy v obsahu** a obvykle jsou vykreslovány vedle dalších inline elementů nebo uvnitř blokového elementu.

<span>Inline element 1</span><span>Inline element 2</span><span>Inline element 3</span>

Inline elementy mají v tomto příkladě červený rámeček a jsou umístěny vedle sebe.

**Výsledek příkladu:**



### Výška a šířka

**Výchozí výška a šířka inline elementu v HTML závisí na obsahu uvnitř elementu a stylech**, které jsou na něj aplikovány. Inline elementy **mají výchozí šířku pouze takovou, jakou potřebují k zobrazení svého obsahu**, aniž by vytvářely zlomy v řádku nebo nové řádky v textu.

Například výchozí šířka inline elementu, jako je <span>, bude určena délkou obsahu uvnitř něj a výchozí výška bude určena velikostí písma a meziliniovým prostorem rodičovského elementu nebo jakýmkoli styly aplikovaným na inline element samotný.

### Úprava výšky a šířky

Je však důležité si uvědomit, že tyto **výchozí hodnoty mohou být přepsány použitím stylů jako například velkost fontu, border, line-height a jiné**, což způsobí, změnu šířky a výšky elementu. Na nasledujícím odstavci můžete vidět výsledek takto upraveného textu.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Neque ullam dolor accusantium alias consectetur, nobis ipsa quibusdam? Eligendi vero exercitationem illo sit officia aliquid tempora. Obcaecati ipsa dolorum quia! Ipsum.

Pozor, výška a šířka inline elementů **nemůže byt ovlivněna vlastnostmi width a height**, protože element zabíra pouze tolik místa, kolik je nutné k zobrazení jejich obsahu.

Lorem <strong style="width: 150px; height: 150px">ipsum dolor</strong> sit amet consectetur adipisicing elit.

V tomto příklade jsme nastavili width a height na 150px ale k žádne změne výšky a šířky na elementu nedošlo.

Lorem **ipsum dolor** sit amet consectetur adipisicing elit.

### Seznam inline elementu

Příklady nejčastěji používaných inline elementů:

* <a>
* <span>
* <strong>
* <em>
* <img>
* <input>
* <label>
* <button>
* <select>
* <textarea>
* <br>

Souhrní seznam inline elementu najdete na stránce [mdn web docs](https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements#list_of_inline_elements).

Následuje živá ukázka, kde je schválně zvýrazněná oblast, kterou každý prvek zabírá:

<p><strong>Město Brno</strong> je počtem obyvatel i rozlohou <em>druhé největší</em> město v České republice, <em>největší město</em> na Moravě a <em>bývalé hlavní město</em> Moravy.</p>

strong, em {

border: 1px dotted #FF6363;

}

## Blokový vs. inline element

|  | **Blokový element** | **Inline element** |
| --- | --- | --- |
| **Použití** | Používají se k uspořádání rozvržení webové stránky a vytváření vizuálně odlišných sekcí obsahu. | Účelem inline elementů v HTML je formátovat text uvnitř řádku obsahu. Používají se například k stylizaci nebo zdůraznění konkrétních slov nebo frází v odstavci, jako jsou tučné nebo kurzívou psané texty. |
| **Šířka** | Výchozí šířka je 100 % šířky dostupné v jejim rodičovském kontejneru. | Výchozí šířka závisí na obsahu uvnitř elementu, t.j. element ma šířku pouze takovou, jakou potřebuje k zobrazení svého obsahu. |
| **Změna šířky** | Výchozí šířka může být změněna pomocí vlastnosti width. | Neumožňují nastavit šířku pomocí vlastnosti width. |
| **Výška** | Výška je určena obsahem uvnitř něj a jakýmkoli CSS styly aplikovaným na element. Pokud element nemá obsah nebo výška není nastavena pomocí CSS vlastnosti, vnitřek elementů se na strance nezobrazí. | Výchozí výška závisí na obsahu uvnitř elementu, t.j. element ma výšku pouze takovou, jakou potřebuje k zobrazení svého obsahu. |
| **Změna výška** | Výchozí výška může být změněna pomocí vlastnosti height. | Neumožňují nastavit výšku pomocí vlastnosti height. |
| **Zalomeni řádku** | Zalomuje řádek. Jakýkoli obsah následující za blokovým elementem se objeví na novém řádku. | Nemá odřádkování a zobrazuje se vedle textu. |
| **Uzavřeny pomocí párových značek** | Musí být vždy uzavřeny pomocí párových značek. | Inline elementy jako např. <i>, <em> a <strong> obaluji svůj obsah mezi otevírací a uzavírací značku. Týto elementy jsou párové. Na druhou stranu inline elementy jako např. <img>, <input>, <br> a <hr> jsou samouzavírací elementy nebo také tzv. prázdne elementy, což znamená, že nepoužívají uzavírací značku, protože nemají obsah a nemá to u nich význam. |

## Replaced elements

Některé inline elementy se mohou chovat jako blokové elementy a lze je stylovat pomocí šířky a výšky. Tyto elementy jsou známe jako **"replaced elements"**.

**"Replaced elements"** jsou inline elementy, které jsou nahrazeny externím zdrojem například obrázkem nebo videem. Důvodem, proč řadíme <img> nebo <video> mezi tyto elementy, je ten, že nemají žádný vlastní obsah. Namísto toho je jejích obsah závislý na externím zdroji (souboru s obrázkem). Tím se liší od jiných elementů, jako jsou <p> nebo <div>, které mají definovaný svůj vlastní obsah.

Příkladem nejčasteji používanych elementu jsou <img>, <video>, <audio> a <input>.

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

<img src="example.jpg" alt="Example Image" width="500" height="300">

<video src="example.mp4" width="640" height="360" controls></video>

## Inline-block element

Kromě blokových a inline elementů existují také inline-block elementy, které kombinuji vlastnosti obou typu. Podobně jako **inline** element, bude se zobrazovat **vedle ostatního textu na stejném řádku**, a jako **blokový** element dovoluje **určit jeho šířku a výšku**.

Použití inline-block elementu může být užitečné v případě, že chceme zobrazit elementy vedle sebe, a zároveň chceme mít možnost jim nastavit konkrétní rozměry.

Chceme-li vytvořit inline-block element, použijeme CSS vlastnost display s hodnotou inline-block. (Co je CSS a jak můžeme nastavit CSS vlastnosti si ukážeme v následující kapitole.)

Na následujícím příkladě si ilustrujeme jak můžeme přepsat blokový element na inline-element a proč se nám to můžeme hodit.

V předcházející kapitole jsem si představili blokový element seznam <ul>, který je tvořený <li>. V tomto příkladě navážeme na seznam a vytvoříme pomocí něho navigaci. V navigaci na desktopové stránce chceme mít položky seřazené vedle sebe s konkrétně definovanými rozměry.

V prvním kroku vytvoříme seznam s třemi položkami. Položkám nastavíme červený rámeček aby jsme viděli jak se mění jejich výchozí nastavení

<ul>

<li style="border: 1px solid red">Item 1</li>

<li style="display: inline-block; border: 1px solid red">Item 2</li>

<li style="display: inline-block; border: 1px solid red">Item 3</li>

</ul>

A red lines on a white background

Description automatically generated

Aby se nám elementy seřadili vedle sebe přepíšeme v následujícím kroku <li> blokové elementy na inline-block elementy pomoci vlastnosti display s hodnotou inline-block.

<ul>

<li style="display: inline-block; border: 1px solid red">Item 1</li>

<li style="display: inline-block; border: 1px solid red">Item 2</li>

<li style="display: inline-block; border: 1px solid red">Item 3</li>

</ul>



V posledním kroku nastavíme výšku a šířku <li> elementů.

<ul>

<li style="display: inline-block; width: 100px; height: 50px; border: 1px solid red">Item 1</li>

<li style="display: inline-block; width: 100px; height: 50px; border: 1px solid red">Item 2</li>

<li style="display: inline-block; width: 100px; height: 50px; border: 1px solid red">Item 3</li>

</ul>

A red rectangle with black text

Description automatically generated

Na tomto příkladě jsme si ukázali jak můžeme změnit výchozí nastavení blokového elementu aby jsme docílili seřazení elementu vedle sebe a nastavení jejich rozměru. To samé můžeme aplikovat na inline elementy, v případě kdy chceme aby měli definovanou výšku a šířku.

Nejčastější situace kdy chceme změnit inline element na inline-block je při vytváření tlačítka, které nás přesměruje na jinou stránku.

Pro vytvoření odkazu použijeme inline element <a>, kterému nastavíme cílovou adresu pomoci atributu href. Protože je <a> inline element, nemůžeme mu nastavit rozměry a plnohodnotně nastylovat jako tlačítko.

<a href="https://engeto.cz/" style="border: 1px solid red">Odkaz</a>

A close-up of a red rectangle

Description automatically generated

Lze shrnout, že inline-block elementy představují flexibilní způsob zobrazení prvků na webové stránce. Ať už vytváříme jednoduché navigační menu nebo tlačítko, inline-block elementy nabízejí vhodny nástroj pro ovládání zobrazení a rozložení elementů na stránce.

## Neutrální elementy

HTML elementy mívají zpravidla svůj význam. Například <h1> představuje hlavní nadpis, pomocí <a> tvoříme odkazy, <p> zas označuje odstavec. Existují ale dva další elementy <div> a <span>, které jsou zcela neutrální. Svému obsahu nepřiřazují žádný sémantický význam a ani prohlížeč jim při zobrazení nedává žádný styl.

Oba elementy jsou sice neutrální, ale je mezi nimi drobný rozdíl:

* <div> - neutrální blokový element,
* <span> - neutrální řádkový element.

Význam těchto prvků nyní ještě plně nedoceníme. V další lekci **Úvod do jazyka CSS** si však ukážeme, jak lze těmto neutrálním elementům přiřadit vlastní vzhled. Budeme tak schopni tvořit strukturu celé webové stránky a jednotlivé části webu pozicovat nebo obarvit podle vlastních představ. To si pak už nevystačíme se základními HTML elementy. Právě díky <div> a <span> tak dokážeme vytvořit složitější struktury.

### Zápis v jazyce HTML

* Blokový neutrální element zapíšeme pomocí párového tagu <div>. div je zkratka pro sekci nebo blok (z anglického slova division).
* Řádkový neutrální element zapíšeme pomocí párového tagu <span>. span je zkratka pro interval nebo rozpětí (z anglického slova span).

Takto může vypadat ukázka použití těchto elementů na webu:

<h1>Osnova kurzu</h1>

<div>

<h2>Úvod</h2>

<p><span>Úvodní část kurzu</span> je zaměřena na obecné představení jazyka HTML. Vysvětlíme si, kde všude nachází tento jazyk svá uplatnění. Poté si ukážeme, co je to tag a jak se využívá při zápisu výsledného kódu.</p>

<div>

<h3>Intro</h3>

<h3>Co je to jazyk HTML?</h3>

<h3>Anatomie HTML elementů</h3>

<h3>Atributy HTML elementů</h3>

</div>

</div>

<div>

<h2>Základy formátování textu</h2>

<p>První <span>praktická část kurzu</span>, která se zaměřuje na formátování textu. Ukážeme si, jak můžeme definovat odstavce, jak zvýraznit vybrané části textu a také, jak náš text lze doplnit o nadpisy.</p>

<div>

<h3>Odstavce</h3>

<h3>Ruční zalomení řádku</h3>

<h3>Zvýraznění a zdůraznění textu</h3>

<h3>Nadpisy</h3>

</div>

</div>

Z ukázky je zřejmé, že použité elementy <div> a <span> nejsou při zobrazení vůbec vidět. Všimni si však zápisu HTML kódu, kde <div> element přispívá k lepší organizaci díky seskupení společných prvků do bloků.

# Skládání elementů

## Skládání elementů

HTML elementy, které jsou vyjádřeny pomocí párových tagů, mají vždy svou obsahovou část. Doteď jsme se učili, že obsahem elementu může být text. Textový obsah ale není nutno podmínkou. My jsme ho zvolili kvůli jednoduchosti. Obsahem elementu může být však i jiný element. Díky této možnosti dokážeme sestavit i složitě strukturovaný dokument.

A close-up of a text

Description automatically generated

V tomto příklade máme element seznam <ul>, který v sobě obsahuje dvě elementy <li>. Když element v sobě obsahuje jeden nebo víc elementu, nazýváme jej **rodičovským elementem**. Element, který je vnořený do rodičovského elementu, nazýváme **potomkem**.

Vztah **rodič-potomek** mezi elementy je důležitý, protože umožňuje seskupovat elementy a aplikovat styly nebo jiné vlastnosti na skupinu jako celek.

V našem příklade se seznamem můžeme nastavit rodičovy barvu pozadi a ta se aplikuje na jeho potomky.

<ul style="background-color: red;">

<li>Položka 1</li>

<li>Další položka</li>

</ul>

Existuje pravidlo, podle kterého se řídí skládání HTML elementů:

* **Blokové elementy** mohou obsahovat jiné **blokové elementy**, **řádkové elementy** nebo *text*.
* **Řádkové elementy** mohou obsahovat pouze **řádkové elementy**, nebo **text**.

Možná to zní po přečtení složitě, když se nad tím ale zamyslíš, dává to smysl. Pokud máme například odstavec (blokový element), může tento element obsahovat čistý text, tučný text nebo klidně i obrázek. Naopak pokud máme odkaz (řádkový element) asi nebude moc moudré dovnitř umisťovat prvky jako jsou nadpis nebo odstavec textu.

Je důležité zdůraznit, že elementy mohou být vnořený do více úrovní, přičemž každá úroveň vnoření vytváří nový vztah rodič-potomek. To umožňuje vytvářet složité a vysoce strukturované webové stránky.

<div>

<div>

<h1>Article Title</h1>

</div>

<div>

<p>Introductory paragraph.</p>

</div>

<div>

<div>

<h2>Section Title</h2>

</div>

<div>

<p>Section content.</p>

</div>

<div>

<div>

<h3>Aside Title</h3>

</div>

<div>

<p>Aside content.</p>

</div>

</div>

</div>

<div>

<div>

<h2>Another Section Title</h2>

</div>

<div>

<p>Another section content.</p>

</div>

<ul>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3

<ul>

<li>Subitem 3.1</li>

<li>Subitem 3.2</li>

</ul>

</li>

</ul>

</div>

<div>

<p>Copyright information.</p>

</div>

</div>

# SÉMANTIKA HTML

## Co je SEO a proč je důležité?

Při vytváření webových stránek nesmíme zapomínat na kvalitu obsahu, která se zde bude zobrazovat. Kromě základního obsahu, který chceme na stránkách mít, je důležité myslet i na to, aby naše stránky a obsah, byly snadno dohledatelné. K tomu slouží **SEO (Seach Engine Optimization)**, v překladu optimalizace pro vyhledávače.

Vyhledávače, k pochopení obsahu webové stránky a určení její relevance pro dotazy uživatelů, analyzuji HTML elementy a zdrojový kód stránky. Už jsme zmiňovali elementy nadpisů (h1 až h6), odstavce <p>, popisky obrázků alt nebo zvýraznění pomocí <strong> atd. Všechny tyto elementy přímo souvisí s tím, jak dobře bude vyhledávač náš web procházet a jaké výsledky bude zobrazovat. Kromě elementu se v rámci SEO řeší také další věci jako je používání klíčových slov, vhodné URL adresy jednotlivých stránek a podstránek nebo odkazy z jiných zdrojů.

## Sémantické elementy

S pojmem **sémantika** se můžeme setkat napříč různými obory. Sémantiku chápeme jako **nauku o významu jazykových jednotek**. Zjednodušeně řečeno, sémantika popisuje, jaký význam mají jednotlivá slova, v našem případě tagy nebo různé příkazy. Z toho vyplývá, že i pro web bude hrát sémantika důležitou roli.

Ideální webová stránka by tedy měla obsahovat jeden H1 nadpis, který představuje hlavní informaci o stránce. V případě dalších nadpisů použijeme hierarchii dalších nadpisů, nějčastěji <h2>, <h3>, <h4>. Text následně dělíme do odstavců <p>. To vše jsme si již říkali. Nicméně sémantických elementů existuje mnohem více a pro nás bude důležité s nimi od začátku pracovat.

Existuje několik důvodu proč je používani sémantických elementu výhodné:

1. **Accessibility / Přístupnost**: Sémantické HTML pomáhá zlepšit přístupnost webového obsahu tím, že poskytuje popisnější a smysluplnější značky, které mohou asistenční technologie, jako jsou například čtečky obrazovky, používat k lepší interpretaci a prezentaci obsahu uživatelům s postižením.
2. **SEO**: Sémantické HTML může také zlepšit optimalizaci webových stránek pro vyhledávače (SEO). Vyhledávače používají značky HTML k pochopení obsahu webové stránky a určení její relevance pro dotazy uživatelů.
3. **Čitelnost a udržovatelnost**: Díky sémantickému HTML je kód také čitelnější a lépe udržovatelný. Díky použití popisných značek je snazší pochopit účel a strukturu webové stránky, což usnadňuje její údržbu a budoucí aktualizaci.

Kromě sémantických elementů pro popis obsahu existuje celá řada elementu pro popis struktury stránky.

<header> - párový tag, který reprezentuje kontejner pro úvodní obsah a navigaci (odkazy).

<nav> - párový tag, který definuje navigaci stránky, tedy odkazy na jednotlivé podstránky na našem webu.

<article> - párový tag, který specifikuje nezávislý, samostatný obsah. Příkladem mohou být příspěvky na blogu, fóru nebo reportáž.

<aside> - párový tag, který definuje vedlejší (dodatečný) obsah stránky. Příkladem mohou být odkazy na další podobné články, další informace nebo reklamy.

<section> - párový tag, který definuje určitou konkrétní část

<footer> - párový tag, který definuje patičku stránky

Tyto elementy umožňují jasně a sémanticky strukturovat obsah webové stránky do logických oddílů a nespoléhat se pouze na nesémantické prvky, jako jsou <div> nebo <span>.

Použití sémantických elementů pro strukturování webových stránek může celkově přinést řadu výhod, včetně lepší organizace a čitelnosti kódu, lepší přístupnosti pro uživatele se zdravotním postižením a lepší viditelnosti a hodnocení ve vyhledávačích.

Příklad použití sémantických elementů pro strukturováni layoutu stránky.

A screenshot of a web page

Description automatically generated

V následujícím úkole máme jednoduchou stránku, která je rozdělena do několika sekci. Vaši úlohou je přepsat dany kód za použití sémantických elementu. Při řešení úkolu můžete plně vycházet z předcházejícího obrázku. Pro kontrolu můžete využít referenční řešení na konci teto sekce.

<!DOCTYPE html>

<html>

<head>

<title>My Webpage</title>

</head>

<body>

<div>

<h1>Welcome to My Webpage</h1>

</div>

<div>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact</a></li>

</ul>

</div>

<div>

<h2>Blog Posts</h2>

<div>

<h3>My First Blog Post</h3>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis

imperdiet sed ante eu consectetur. Maecenas dignissim, est ac laoreet

malesuada, dolor nibh ultricies tortor, sit amet aliquet dolor lectus

vel augue.

</p>

</div>

<div>

<h3>My Second Blog Post</h3>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis

imperdiet sed ante eu consectetur. Maecenas dignissim, est ac laoreet

malesuada, dolor nibh ultricies tortor, sit amet aliquet dolor lectus

vel augue.

</p>

</div>

</div>

<div>

<h2>Product Listings</h2>

<div>

<h3>Product A</h3>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis

imperdiet sed ante eu consectetur. Maecenas dignissim, est ac laoreet

malesuada, dolor nibh ultricies tortor, sit amet aliquet dolor lectus

vel augue.

</p>

</div>

<div>

<h3>Product B</h3>

<p>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis

imperdiet sed ante eu consectetur. Maecenas dignissim, est ac laoreet

malesuada, dolor nibh ultricies tortor, sit amet aliquet dolor lectus

vel augue.

</p>

</div>

</div>

<div>

<h3>My Sidebar</h3>

<ul>

<li><a href="#">Link 1</a></li>

<li><a href="#">Link 2</a></li>

<li><a href="#">Link 3</a></li>

</ul>

</div>

<div>

<p>&copy; 2023 My Webpage</p>

</div>

</body>

</html>

## Úkol č.1 - Nečíslovaný seznam

Na následující ukázce máme nákupní seznam. Přidej na konec seznamu další položku s textem banány.

<ul>

<li>rohlík</li>

<li>šunkový salám</li>

<li>čokoláda</li>

<li>mouka</li>

<li>pivo</li>

</ul>

## Úkol č. 2 - Číslovaný seznam

Pojďme se vrhnout na další část testu. Nyní si vyzkoušíme naše HTML znalosti ohledně číslovaných odkazů. Do prázdného editoru přepiš následující (číslovaný) seznam. Použij HTML elementy pro číslovaný seznam.

1. Vytvoření plánu a rozpočtu
2. Návrh architektury
3. Návrh grafického designu
4. Tvorba webových stránek
5. Schválení a zveřejnění webu

Jakmile budeš mít HTML zápis hotový, přidej do seznamu ještě jednu položku. **Na úplný začátek** přidej bod Úvodní analýza. Všimni si, jak se výsledný seznam automaticky přečísluje.

## Úkol č. 3 - Obrázek

Pojďme si vyzkoušet vložení nového obrázku do hotového textu. Na následující ukázce máme HTML kód obsahující nadpisy a odstavce.

1. **Nad nadpis** <h2> s textem V Česku chybí 20 tisíc ajťáků vlož nový obrázek. Jako zdrojový soubor obrázku použij následující adresu https://engeto.cz/wp-content/uploads/2019/01/engeto-square.png.
2. Pokud se ti to podaří, uvidíš logo ENGETO Academy.
3. Každý obrázek musí mít definovaný svůj zástupný text. Doplň do HTML elementu obrázku vhodný atribut a popiš v něm, co na obrázku vidíš.

<h1>ENGETO Academy</h1>

<p>V Česku chybí 20 tisíc ajťáků. Tvoříme nové IT talenty.</p>

<h2>V Česku chybí 20 tisíc ajťáků</h2>

<p>Vzdělávání v 21. století musí být inovativní. V ENGETO Academy se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

img {

width: 100px;}

## Úkol č. 4 - Tabulka

Pojďme si vyzkoušet, jestli rozumíme struktuře tabulky v jazyce HTML. Začneme jednoduchým úkolem. Do tabulky na následující ukázce přidej na konec nový řádek, který ponese jméno Radek Svoboda v první buňce a hodnotu Praha v buňce druhé.

<table>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Pardubice</td>

</tr>

<tr>

<td>Ondřej Vlezdopole</td>

<td>Staré Město u Uherského Hradiště</td>

</tr>

</table>

## Úkol č. 5 - Tabulka II

Doplňování řádku bychom měli. I v druhém úkolu máme tabulku se seznamem jmen. Obsahuje tři jména smyšlených osob a jejich bydliště. Tvůj úkol:

1. Přidej do tabulky třetí sloupec s hlavičkou **Povolání**.
2. Hodnoty ostatních buněk v nově vznikajícím sloupci budou následující: Jan Novák má povolání **Technik**, Hana Šeredová má povolání **Asistentka**, František Dobrota má povolání **Programátor** a Ondřej Vlezdopole má povolání **Překladatel**.
3. Jakmile bude tabulka hotová, přidej ještě její popisek Účastníci kurzu.

<table>

<tr>

<th>Jméno</th>

<th>Bydliště</th>

</tr>

<tr>

<td>Jan Novák</td>

<td>Brno</td>

</tr>

<tr>

<td>Hana Šeredová</td>

<td>Kladno</td>

</tr>

<tr>

<td>František Dobrota</td>

<td>Moravská Třebová</td>

</tr>

</table>

## Bonusový úkol

V rámci domácího úkolu zkombinujeme většinu věcí, které jsme se dnes naučili.

1. Vytvoř si složku a pojmenuj ji dle libosti (např. **Cestování**).
2. Otevři si složku ve VS Code a v této složce si vytvoř soubor **index.html**
3. Pomocí vykřičníku a klávesy enter si doplň hlavičku dokumentu.
4. Doplň nadpis <h1> s textem: Dovolená.
5. Doplň nadpis <h2> s textem: Co si sbalit
6. Vytvoř nečíslovaný seznam alespoň s 5 položkami.
7. Přidej libovolný obrázek, nezapomeň na atribut alt.
8. Doplň nadpis <h2> s textem: Náklady
9. Vytvoř složitější tabulku i s hlavičkou, která bude obsahovat alespoň tři řádky <tr> a v každém řádku tři sloupečky <th>. Představ si tabulku jako program zájezdu: Datum, název položky, cena (21.10., Letenka, 2699 Kč).

# Základy CSS

## Co je to CSS?

HTML je základním komunikačním jazykem každého webu. Pomocí něho dokážeme vytvořit různé elementy a poskládat tak obsah celé stránky. Jenže takto vzniklý obsah je pouhá kombinace obyčejného textu, nadpisů a obrázků. Jak sami asi už tušíte, samotné HTML nám k vytvoření plnohodnotného webu nestačí. Webové stránky jsou totiž plné barev, tvarů a různých grafických prvků a tyto designové vlastnosti již nejsme schopni vyjádřit na úrovni HTML. Koncem minulého století tak začaly vznikat první pokusy o grafickou úpravu webových stránek. Kolem roku 1997 vše vyústilo ve vznik nového jazyka CSS.

**CSS** je zkratkou pro Cascading Style Sheets, česky **kaskádové styly**:

* Styly jsou skupiny pravidel, které umožňují webovým vývojářům měnit vzhled jednotlivých HTML elementů.
* Přívlastek kaskádové pak znamená, že je možné tyto pravidla na sebe vrstvit podle určité priority.

Pokud tedy hledáte způsob, jak například změnit barvu textu či pozadí, jak upravit styl písma nebo jak definovat umístnění jednotlivých elementů, pak jazyk CSS představuje v současnosti jedinou možnou cestu, kterou lze na webu použít. Hned po HTML je to další jazyk, kterému rozumí každý moderní webový prohlížeč.

# Jak zapisujeme CSS

## Vlastnosti vzhledu

Pokud se snažíme popsat vzhled nějakého elementu, je nejlepší se zaměřit na jeho vlastnosti. Vlastnost nám říká například:

* Jak bude vypadat text? - barva textu, velikost písma, tloušťka (řez) písma, ...
* Má element nějaký pevný rozměr? - šířka, výška, ...
* Jaký je vzhled celého elementu? - pozadí, průhlednost, rámeček, zaoblené rohy, mezery, ...
* Jaká je pozice elementu na celé stránce? - vzdálenost od levého horního rohu stránky, ...

Když si pak vybereme určitý HTML element na webu, dokážeme vyjmenovat vlastnosti, které popisují jeho vzhled. Příkladem může být následující ukázka:

A blue text on a white background

Description automatically generated

Vlastnostmi takového prvku jsou: modrý text, podtržený text, větší velikost písma a šedé pozadí.

## Deklarace stylů

Jazyk CSS představuje v současnosti jedinný způsob, jak lze nastavit vzhled webového obsahu. Při zápisu píšeme jednoduché **deklarace**. A každá taková deklarace pak **popisuje jednu vlastnost** vybraného HTML elementu.

A screenshot of a computer

Description automatically generated

Deklarace vlastnosti se skládá z **identifikátoru**. Pomocí něho řekneme, jakou vlastnost elementu chceme změnit. Poté následuje dvojtečka :. Dále pak zapíšeme **hodnotu** upravované vlastnosti. A celý zápis zakončíme středníkem ;.

Pokud upravujeme více vlastností, můžeme jednotlivé deklarace psát za sebou. Následující příklad ukazuje zápis několika deklarací (jejich význam bude vysvětlen později):

color: blue;

background-color: grey;

text-decoration: underline;

font-size: 34px;

Řádek color: blue; je deklarace samotná, kde color je identifikátor vlastnosti a blue její hodnota.

<h1>Zámek Jezeří</h1>

<p>Jezeří je hrad přestavěný na zámek, který se nachází na svazích Krušných hor v okrese Most v Ústeckém kraji. Zámek stojí přímo nad Lomem ČSA, necelých 500 metrů od jeho hrany.</p>

p {

color: blue;

background-color: grey;

text-decoration: underline;

font-size: 34px;

}

## Přímý zápis stylu

I přesto, že CSS a HTML jsou úplně odlišné jazyky, existuje jednoduchý způsob, jak lze CSS deklaraci přiřadit přímo vybranému HTML elementu. Docílíme toho pomocí HTML atributu style, který připojíme k počátečnímu tagu. Hodnotu atributu style tvoří jednotlivé CSS deklarace. Zápis pak může vypadat například takto:

<p style="color: blue;">Ukázkový text</p>

Podobným způsobem lze přiřadit i více CSS deklarací k elementu:

<p style="color: blue; text-decoration: underline;">Ukázkový text</p>

Významem uvedených CSS deklarací se teď netrapte. Zanedlouho se k nim vrátíme a vše si podrobně vysvětlíme.

#### Výhody tohoto zápisu:

* Ihned vidíme, jaké CSS deklarace jsou přiřazeny k elementu.
* Styly se načítají spolu s HTML kódem.
* Jdou použít ve všech případech, kdy zapisujeme HTML kód.

#### Nevýhody tohoto zápisu:

Kvůli tomu, že styly vypisujeme zvlášť ke každému HTML elementu, tak nedokážeme nadefinovat jeden sdílený vzhled. Co si pod tím představit? Kdybychom například chtěli, aby všechny <h2> nadpisy vypadali stejně, budeme muset napsat CSS deklarace zvlášť ke každému tagu <h2>. To je ale velmi nepraktické. Představte si, že se později rozhodneme změnit barvu písma. Takové rozhodnutí by znamenalo, že bude nutné projít všechny HTML elementy, kterých se to týká, a ručně upravit potřebné CSS deklarace.

Tento problém je naštěstí snadno řešitelný. Hned na další stránce si ukážeme jiný způsob zápisu.

# Pravidla

## Pravidla v CSS

Pokud nechceme přiřazovat CSS deklarace přímo k HTML elementům (pomocí atributu style), tak existuje ještě jeden způsob zápisu. V jazyce CSS můžeme vytvářet tzv. **pravidla**. Pomocí pravidel dokážeme seskupit dohromady i několik CSS deklarací. A tyto skupiny stylů poté přiřazujeme vybraným HTML elementům. Výsledný kód zapisujeme podle následujícího vzoru:

selektor {

identifikátor-vlastnosti-A: hodnota1;

identifikátor-vlastnosti-B: hodnota2;

}

Každé pravidlo se skládá ze svého **selektoru** a bloku jednotlivých **deklarací**.

* Selektor je důležitá část každého pravidla. Jak již název napovídá, selektory nám umožňují **označit** (podle angl. výrazu select = označit) **elementy** na webové stránce podle určitých kritérií. Takto vybrané elementy jsou pak ostylované pomocí CSS deklarací uvedených uvnitř závorek { ... }.
* Význam deklarací jsme už probrali na začátku lekce. Jen zde zmíním, že každé pravidlo musí obsahovat **minimálně jednu deklaraci** a pokud je deklarací více, tak jsou mezi sebou oddělené středníkem.

### Ukázka

Konkrétní zápis nějakého pravidla může vypadat například takto:

h3 {

color: blue;

text-decoration: underline;

}

V našem příkladu v Codepen můžeme vidět, že nastavení pravidla v CSS pomocí tagu <h3> nám obarvil a podtrhl všechny nadpisy úrovně **h3**, které se v HTML dokumentu nacházejí. V tomto případě je to výhodné, protože chceme všem nadpisům nastavit stejné styly. Nicméně pokud bychom chtěli některému z nadpisů nastavit jinou barvu, museli bychom k zápisu stylů přistoupit pomocí selektoru tříd. Více v kapitole **Selektory tříd**.

<h3>Podzim</h3>

<p>Podzim začíná podzimní rovnodenností v září.</p>

<h3>Zima</h3>

<p>Zima začíná zimním slunovratem v prosinci.</p>

<style>

h3 {

color: blue;

text-decoration: underline;

}

/\*Nastavení pravidla pomocí tagu h3 nám nastavil barvu a podtržení na všechny tagy h3, které se v HTML dokumentu nacházejí.\*/

</style>

## Pravidla umístěná v kódu

U přímého zápisu stylů bylo jasné, jaké styly přiřazujeme k HTML elementům. Do kódu stačilo přidat atribut style=" ... " a do něho vepsat jednotlivé CSS deklarace. U CSS pravidel je situace trochu jiná. Pravidlo samotné není součástí HTML kódu. Jediné, co ho pojí s HTML elementy, je selektor - což je předpis, podle kterého vybíráme HTML elementy na stránce.

Pokud tedy vytváříme CSS pravidla, zbývá nám dořešit, jakým způsobem je připojíme k dokumentu. Jazyk HTML s tím počítá a do hlavičky stránky je možné vložit speciální element <style> (z angl. slova style = styl). Tento element zapíšeme pomocí párových tagů <style> ... </style> a obsahuje výčet všech CSS pravidel.

<head>

<style>

</style>

</head>

Po přidání <style> tagů do hlavičky stránky, můžeme začít psát potřebný CSS kód (význam jednotlivých CSS deklarací bude vysvětlen později):

<head>

<style>

.red-text {

color: red;

}

.underline-text {

text-decoration: underline;

}

</style>

</head>

### Ukázka

Reálná ukázka s pravidly umístěnými uvnitř tagu <style> může vypadat například takto:

<!DOCTYPE html>

<html>

<head>

<title>ENGETO academy</title>

<style>

.blue-text {

color: blue;

}

.underline-text {

text-decoration: underline;

}

</style>

</head>

<body>

<h1 class="blue-text underline-text">ENGETO Academy</h1>

<p>V Česku chybí 20 tisíc ajťáků. Tvoříme nové IT talenty.</p>

<h2 class="blue-text">V Česku chybí 20 tisíc ajťáků</h2>

<p>Vzdělávání v 21. století musí být inovativní. V <strong>ENGETO Academy</strong> se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

</body>

</html>

## Pravidla umístěná v souboru

Předchozí řešení nám dovoluje napsat CSS pravidla přímo k dokumentu. Tento přístup je velice jednoduchý. Rád bych ale zmínil několik problémů, které to s sebou přináší:

* Jakmile začneme mít pravidel více, celý kód stránky se nám značně "natáhne".
* Představte si, že váš web má několik stránek a každá z nich obsahuje definici stylů pomocí <style>...</style> ve své hlavičce. Všechna CSS pravidla, která budou společná pro více stránek, se budou mnohem hůře udržovat. Jakákoliv úprava společných stylů se tak musí provést na více místech.

Použití <style>...</style> není však jediná cesta, kterou máme. My totiž můžeme vzít všechna CSS pravidla a uložit je do odděleného souboru. Takovýto soubor musí mít koncovku **.css** (například **style.css**).

### Připojení CSS souboru k dokumentu

Pravidla definovaná tímto způsobem už nebudou součástí HTML kódu. Nyní si ukážeme cestu, jak se dá vytvořený CSS soubor propojit s HTML dokumentem. Pro tyto učely použijeme element <link> (z angl. slova link = odkaz). Tento element umístíme do hlavičky stránky a jeho kód zapíšeme pomocí nepárového tagu. Má dva povinné atributy:

* href — Cílová adresa k souboru, který chceme propojit.
* rel — <link> element nemusí sloužit pouze k propojení CSS stylů s HTML stránkou. Atribut rel nám říká, jaký typ vazby vzniká mezi propojenými soubory. Protože v našem případě připojujeme kaskádové styly k HTML dokumentu, bude mít atribut hodnotu stylesheet (z angl. slova stylesheet = stylopis).

Celý zápis elementu pak může vypadat takto:

<link href="./style.css" rel="stylesheet">

### Jaké to má výhody?

Největší výhodou bezesporu je, že všechny styly máme na jednom místě - v odděleném souboru. Stránka pak ví, že má své styly načítat pouze z tohoto zdroje. Pokud má web více stránek, tak nám odpadá opakované psaní CSS deklarací. Všechny totiž můžou načítat stejný CSS soubor. Když se pak v budoucnu rozhodneme upravit nějakou vlastnost, nemusíme tak šahat do kódu jednotlivých stránek.

### Ukázka

Následuje reálná ukázka HTML kódu, kde styly se načítají z odděleného souboru style.css:

### Jaké to má výhody?

Největší výhodou bezesporu je, že všechny styly máme na jednom místě - v odděleném souboru. Stránka pak ví, že má své styly načítat pouze z tohoto zdroje. Pokud má web více stránek, tak nám odpadá opakované psaní CSS deklarací. Všechny totiž můžou načítat stejný CSS soubor. Když se pak v budoucnu rozhodneme upravit nějakou vlastnost, nemusíme tak šahat do kódu jednotlivých stránek.

### Ukázka

Následuje reálná ukázka HTML kódu, kde styly se načítají z odděleného souboru style.css:

<!DOCTYPE html>

<html>

<head>

<title>ENGETO academy</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1 class="blue-text underline-text">ENGETO Academy</h1>

<p>V Česku chybí 20 tisíc ajťáků. Tvoříme nové IT talenty.</p>

<h2 class="blue-text">V Česku chybí 20 tisíc ajťáků</h2>

<p>Vzdělávání v 21. století musí být inovativní. V <strong>ENGETO Academy</strong> se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

</body>

</html>

CSS:

.blue-text {

color: blue;

}

.underline-text {

text-decoration: underline;

}

### Best practice

V kapitole **Pravidla** jsme si ukázali všechny možnosti zápisu stylů. Občas se setkáte se všemi těmito možnostmi, nicméně v drtivé většině případů budete používat **samostatný CSS soubor**, který bude nalinkovaný do HTML. Je to přehlednější a schůdnější varianta. I v našem kurzu budeme nadále k zápisu stylů používat externí CSS soubor, který budeme do HTML linkovat.

# Selektory

## Typové selektory

Jak jsme již zmínili, bez selektoru se žádné pravidlo neobejde. Právě díky němu dokážeme označit určitou skupinu HTML elementů a na ně pak aplikovat definované styly. Selektory mohou být různého druhu. Liší se hlavně způsobem, jakým vybíráme HTML elementy na stránce.

Jako první si ukážeme tzv. **typový selektor**. Pomocí něho dokážeme označit **konkrétní typ HTML elementů**. Postup je velice snadný. Při zápisu selektoru stačí uvést název HTML elementu.

Pojďme si to vysvětlit na jednoduchém příkladu. Mějme následující pravidlo:

h3 {

color: blue;

text-decoration: underline;

}

Zápis obsahuje selektor h3, kterým označíme všechny HTML elementy zapsané pomocí tagu <h3>. Na ně se pak aplikují deklarace uvedené uvnitř závorek { ... }. V našem případě bude obsah všech h3 elementů obarven na modro a text bude podtržený (pomocí color: blue; a text-decoration: underline;).

### Ukázka

Následující ukázka obsahuje HTML kód a jeho výsledné zobrazení. Můžeme si povšimnout, že všechny <h3> elementy jsou obarveny právě pomocí našeho pravidla (obsahujícího typový selektor). V Codepen se podívej do sloupečku CSS (tam se nachází náš style.css soubor):

<h1>Legenda o praotci Čechu</h1>

<p>Praotec Čech je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována Čechy.</p>

<h2>Dostupné prameny</h2>

<h3>Kosmova kronika</h3>

<p>V latinsky psané Kosmově Kronice Čechů se vypráví o příchodu bezejmenného lidu do dnešních Čech.</p>

<h3>Dalimilova kronika</h3>

<p>V česky psané veršované Kronice tak řečeného Dalimila z počátku 14. století se poprvé objevuje motiv Charvátské země, vraždy, výstup na Říp.</p>

h3 {

color: blue;

text-decoration: underline;

}

### Procvičení

V připraveném HTML souboru v rámci Codepen si procvičíme, jak zapisovat styly pomocí typových selektorů. Rozklikněte si Codepen a do sloupečku CSS doplňte následující styly:

1. Nadpisu <h1> nastavte zelenou barvu.
2. Nadpisům <h2> nastavte podtržení.
3. Odstavcům <p> nastavte velikost písma na 18px.

<h1>Kraje ČR</h1>

<h2>Jihomoravský kraj</h2>

<p>Jihomoravský kraj je vyšší územně samosprávný celek České republiky, který vznikl v roce 2000 na jižní a středozápadní Moravě.</p>

<h2>Moravskoslezský kraj</h2>

<p>Moravskoslezský kraj (od ledna 2000 do května 2001 Ostravský kraj) je jedním ze 14 vyšších územních samosprávných celků v Česku.</p>

<h2>Olomoucký kraj</h2>

<p>Olomoucký kraj je vyšší územně samosprávný celek, jehož území je tvořeno pěti okresy v západní části někdejšího Severomoravského kraje a okresem Prostějov v severní části územního Jihomoravského kraje.</p>

<h2></h2>

<p></p>

## Selektory tříd

Typové selektory nám umožňují označit všechny HTML elementy určitého typu (např. všechny <h3> nadpisy). Tato možnost je užitečná, ale v praxi často potřebujeme vybrat elementy podle přísnějšího kritéria (např. jen <h3> nadpisy, které se nachází v hlavičce stránky). Další možností, kterou lze u CSS selektorů využít, je necílit na celé elementy, ale pouze na konkrétní atributy HTML elementů.

Jedním takovým atributem je tzv. **třída**. V HTML jí zapíšeme přidáním atributu class (třída = z angl. slova class) k počátečnímu tagu libovolného elementu. Říkáme, že **element zařazujeme do nějaké třídy**. V jazyce CSS pak dokážeme jednoduchým zápisem vybrat pouze ty elementy, které obsahují třídu s konkrétní hodnotou.

Pojďme se podívat na ukázku HTML zápisu pro nadpis:

<h3 class="main-title">Vítejte u nás na webu</h3>

Kód obsahuje HTML element <h3> tvořící nadpis třetí úrovně. Nadpis pak má přiřazený atribut class, jehož hodnota je hlavni-nadpis. V jazyce CSS pak dokážeme tento konkrétní element vybrat pomocí selektoru, který cílí na hodnotu atributu class:

.main-title {

/\* sem pak umístíme jednotlivé CSS deklarace... \*/

}

Aby bylo možné v CSS vybrat konkrétní třídu (tj. element s class atributem), musíme přidat tečku (.) před hodnotu dané třídy. V příkladě nahoře máme třídu main-title a odpovídajícím CSS selektorem je .main-title.

Teď už nejspíš tušíte, že výběr elementů, které budeme chtít ostylovat pomocí pravidla .main-title { ... }, je v našich rukou. Stačí pouze k HTML elementu přidat třídu class="main-class" a na takový prvek bude použito naše CSS pravidlo. A je jedno, zda se jedná o nadpis nebo odstavec. Na typ elementu se nyní nehledí. Jediné, co u takové selektoru rozhoduje, je atribut class a jeho hodnota.

### Ukázka

Následující ukázka představuje použití třídy .highlighted-text. Jedná se o pravidlo, které obarví text na modro. Tento připravený styl je pak použit na třech místech v kódu, tj. u každého elementu, který má atribut class="highlighted-text".

Naše CSS třída obsahuje následující deklaraci:

.highlighted-text {

color: blue;

}

HTML kód a jeho výsledek pak vypadá takto:

<h1>Legenda o praotci Čechu</h1>

<p><span class="highlighted-text">Praotec Čech</span> je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována Čechy.</p>

<h2>Dostupné prameny</h2>

<h3 class="highlighted-text">Kosmova kronika</h3>

<p>V latinsky psané Kosmově Kronice Čechů se vypráví o příchodu bezejmenného lidu do dnešních Čech.</p>

<h3 class="highlighted-text">Dalimilova kronika</h3>

<p>V česky psané veršované Kronice tak řečeného Dalimila z počátku 14. století se poprvé objevuje motiv Charvátské země, vraždy, výstup na Říp.</p>

.highlighted-text {

color: blue;

}

## Násobné použití tříd

V minulé kapitole jsme si řekli, že styly dokážeme přiřadit konkrétním elementům, které mají HTML atribut class. Tento přístup je užitečný. Můžeme si tak vytvořit pravidla, které pak budeme přiřazovat podle potřeby. Na následující ukázce máme 2 jednoduchá pravidla. První pravidlo .red-text obarvuje text na červeno a druhé pravidlo .underline-text podtrhne zobrazený text čárou.

.red-text {

color: red;

}

.underline-text {

text-decoration: underline;

}

Při zápisu HTML kódu pak zapisujeme konkrétní class atributy:

<p><span class="underline-text">Praotec Čech</span> je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována <span class="red-text">Čechy</span>.</p>

Tímto způsobem ostylujeme část textu červenou barvou a část textu bude podtržena čárou. Jenže, co kdybychom chtěli oba styly na některých místech zkombinovat, tj. v našem případě vytvořit červený podtržený text? Jednou z možností by bylo zapsat úplně nové CSS pravidlo, které bude obsahovat zmíněnou kombinaci stylů. Existuje však ještě jeden přístup. HTML atribut class nemusí nutně obsahovat pouze jeden název třídy. Jako jeho hodnotu můžeme uvést i názvy několika tříd.

<span class="red-text underline-text"> ... </span>

Pokud jako hodnotu atributu class uvádíme více tříd, tak jedinou nutnou podmínkou je **oddělovat jejich názvy mezerou**.

### Ukázka

<h1>Legenda o praotci Čechu</h1>

<p><span class="red-text underline-text">Praotec Čech</span> je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována <span class="underline-text">Čechy</span>.</p>

<h2>Dostupné prameny</h2>

<h3 class="red-text">Kosmova kronika</h3>

<p>V latinsky psané Kosmově Kronice Čechů se vypráví o příchodu bezejmenného lidu do dnešních Čech.</p>

<h3 class="red-text">Dalimilova kronika</h3>

<p>V česky psané veršované Kronice tak řečeného Dalimila z počátku 14. století se poprvé objevuje motiv Charvátské země, vraždy, výstup na Říp.</p>

.red-text {

color: red;

}

.underline-text {

text-decoration: underline;}

## ID selektory

HTML elementy můžeme označit ještě dalším způsobem, pomocí tzv. **jedinečného identifikátoru**. Ten v HTML zapíšeme přidáním atributu id k počátečnímu tagu libovolného elementu. Jedinečný se nazývá proto, že hodnota tohoto atributu by měla být **unikátní v rámci celé stránky** - tj. žádný jiný HTML element nemá identifikátor (atribut id) se stejnou hodnotou.

Na následující ukázce je vidět zápis jedinečného identifikátoru v jazyce HTML:

<h1 id="introduction-title">ENGETO Academy</h1>

Identifikátor id se z hlediska CSS chová stejně jako třída class. Jediný rozdíl je v zápisu samotného selektoru. U třídy se přidává tečka (.) před hodnotu třídy. U jedinečného identifikátoru musíme uvést jiný znak. Selektor obsahuje křížek (#) následovaný hodnotou hledaného identifikátoru. V příkladě nahoře máme atribut id="introduction-title". Odpovídajícím selektorem bude #introduction-title:

#introduction-title {

/\* sem pak umístíme jednotlivé CSS deklarace...\*/

}

### Ukázka

<h1 id="introduction-title">Legenda o praotci Čechu</h1>

<p><span id="praotec-cech">Praotec Čech</span> je legendární prapředek, který podle českých pověstí o svém původu přivedl svůj lid do země, která po něm byla pojmenována Čechy.</p>

<h2>Dostupné prameny</h2>

<h3 id="kosmova-kronika">Kosmova kronika</h3>

<p>V latinsky psané Kosmově Kronice Čechů se vypráví o příchodu bezejmenného lidu do dnešních Čech.</p>

<h3 id="dalimilova-kronika">Dalimilova kronika</h3>

<p>V česky psané veršované Kronice tak řečeného Dalimila z počátku 14. století se poprvé objevuje motiv Charvátské země, vraždy, výstup na Říp.</p>

Použitá CSS pravidla jsou:

#introduction-title {

color: blue;

}

#praotec-cech {

color: red;

text-decoration: underline;

}

#kosmova-kronika {

color: red;

}

#dalimilova-kronika {

color: red;

}

Protože hodnota ID atributu by měla být jedinečná v rámci celé stránky, musíme přiřadit unikátní id atribut každému elementu, který chceme stylovat. Všimněte si, že v případě selektoru tříd nám stačila pouhá dvě CSS pravidla (viz ukázka z minulé kapitoly) a tento sdílený styl šlo využívat opakovaně.

Na každodenní bázi v HTML a CSS pracujeme především s atributy class. Atribut id využíváme např. při odkazování v rámci stránky (tzv. kotvy). Ukážeme si později v kurzu. S častějším využití id se setkáme např. v JavaScriptu, kdy pomocí id odkazujeme na daný element. Pokud máme např. tlačítko na odesílání formuláře, chceme, aby JavaScript reagoval opravdu jen na tohle konkrétní tlačítko, proto má tohle tlačítko jedinečný identifikátor.

### Procvičení

1. Rozklikněte si Codepen a k elementu <div>, který obaluje všechny stupně obtížnosti, doplňte jedinečný identifikátor.
2. Tomuto id v CSS nastavte velikost písma na **18px**.
3. Lichým položkám v HTML (jednotlivé stupně obtížnosti ferrat) přidejte třídu.
4. Této třídě nastavte barvu písma na libovolnou barvu. Např. hnědou.

<h1>Ferraty v Rakousku</h1>

<h2>Popis jednotlivých obtížností</h2>

<div>

<div>A – Zajištěné, jednoznačně vedené turistické cesty a velmi lehké zajištěné cesty – ferraty. </div>

<div>B – Mírně strmý skalní terén. Umělé zajištění v podobě ocelových lan, kramlí, kolíků a žebříků.</div>

<div>C – Strmý skalní terén. Umělým jištěním (ocelová lana, kramle, kolíky, žebříky) je opatřena převážná část celkové délky cesty</div>

<div>D – Velmi strmý skalní terén, místy kolmé partie, místy dokonce převislé.</div>

<div>E – Místy extrémně exponované cesty via ferrata, vedené v kolmém skalním terénu s minimem přirozených stupů. </div>

<div>F - Tyto úseky bývají dlouhé např. do 50 m. Jedná se o cestu v převisu kde je natažené jen jistící ocelové lano. </div>

</div>

## Kdy použít který selektor

Jak jsme si ukázali výše, máme 3 základní selektory: typový, třída, ID. Jaké jsou mezi nimi rozdíly?

Z pohledu CSS vlastně nijak velké. Kaskádovým stylům je jedno, ke kterému selektoru je připíšete a budou fungovat stejně.

Z celkového pohledu na web už to ale není tak jednoduché:

**Typové selektory** jsou svázané s HTML elementy, které jsou na stránce používané. Tzn. pokud tušíme, jak bude celková stránka vypadat, je výhodné si pomocí typových selektoru nastylovat úplný základ - např. pokud víme, že nadpisy <h1> budou vždy červéné, má smysl použít typový element h{ }, pokud víme, že většina odstavců (element <p>) bude mít velikost písma 14px, má smysl použít typový element p{ }.

**ID selektory** se naopak nejvíc využívají v Javascriptu a je pro ně typické, že jsou unikátní. Tedy, že pokud přiřadíme id nějakému elementu, pak to samé id nebude mít jiný element na celých stránkách. Pro statické css styly (tedy takové, které se nepřiřazují pomocí javascriptu - i to jde, ale o tom až o hodně lekcí později) se prakticky nepoužívají.

**Třídy** naopak jsou tím nejpoužívanějším selektorem pro kaskádové styly. Často se celé css soubory píší jen pomocí nich (tedy i bez používání typových selektorů). Potřebujeme střídat barvu pozadí v řádcích tabulky? Jednoduše si vytvoříme 2 třídy: .tr-sude{ } a .tr-liche{ }, ty přířadíme správným řádkům a přidáme požadované vlastnosti. Potřebujeme nastylovat nadpis nebo obrázek? Vytvoříme třídu .header{ } a třídu .picture{ }...

**Poznámka:** Výše zmíněná pravidla nejsou vynucovaná samotnými jazyky html a css, ale jsou běžně dodržovaná mezi programátory.

## Kdy použít který selektor

V tomto kurzu jsme si ukázali několik CSS pravidel. Uvedené příklady měli však jedno společné - každé pravidlo obsahovalo jeden selektor.

selektor {

identifikátor-vlastnosti-A: hodnota1;

identifikátor-vlastnosti-B: hodnota2;

}

V jazyce CSS ale není nutné, aby pravidlo mělo jen jeden selektor. Vytvářet lze i tzv. **hromadné deklarace**. Jednotlivé selektory seskupíme pomocí čárek. Deklarace uvedené v závorkách { ... } pak budou platit pro všechny vyjmenované selektory.

### Ukázka

Mějme situaci, kdy se rozhodneme nastavit barvu všech nadpisů na webu. Pomocí starého přístupy by zápis CSS pravidel vypadal takto:

h1, h2, h3, h4, h5, h6 {

color: blue;

}

## Hierarchie selektorů

V jazyce HTML můžeme libovolné párové HTML elementy skládat do sebe. Díky této vlastnosti dokážeme vytvořit i složitě strukturovaný dokument. Zde je malá ukázka několika do sebe zanořených HTML elementů:

<div class="hlavicka">

<h1><strong>ENGETO Academy</strong> Vás vítá</h1>

</div>

<div class="obsah">

<p>Vzdělávání v 21. století musí být <em>inovativní</em>. V <a href="httsp://engeto.cz">ENGETO Academy</a> se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – <strong>praktické znalosti aplikovatelné na projektech</strong>.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

</div>

Při vytváření CSS pravidel, konkrétně při zápisu CSS selektorů, můžeme této struktury HTML dokumentu využít a psát pravidla, která například cílí **pouze** na element <strong>, který se nachází uvnitř nadpisu <h1>. Ostatní elementy <strong> pak nebudou takovýmto pravidlem vůbec ovlivněny.

Při psaní CSS pravidel postupujeme tak, že zapíšeme za sebe více selektorů a tyto selektory budou **od sebe oddělené pouze mezerou**. Oddělíme-li totiž v definici dva selektory X a Y mezerou, znamená to, že se pravidlo X Y { ... } bude aplikovat pouze na elementy typu Y, které se nachází uvnitř HTML elementů typu X.

### Ukázka

Využijeme HTML kód uvedený výše. CSS pravidlo, které by cílilo na element <strong> umístěný uvnitř hlavního nadpisu h1, by vypadalo následovně:

h1 strong {

color: red;

}

Jiné CSS pravidlo, které by umožňovalo nastavit styl pouze těch odkazů, které se nachází uvnitř obsahové části, by vypadalo takto:

.obsah a {

color: green;

}

<div class="hlavicka">

<h1><strong>ENGETO Academy</strong> Vás vítá</h1>

</div>

<div class="obsah">

<p>Vzdělávání v 21. století musí být <em>inovativní</em>. V <a href="httsp://engeto.cz">ENGETO Academy</a> se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – <strong>praktické znalosti aplikovatelné na projektech</strong>.</p>

<p>Ty se učíš, my analyzujeme. Každé dokončené cvičení nebo úkol vyhodnocujeme a díky tomu ti povíme, co ti jde a kde je třeba zabrat.</p>

</div>

CSS:

h1 strong {

color: red;

}

.obsah a {

color: green;

}

## Další CSS selektory

Popsali jsme si už tři různé selektory: **typové selektory**, **selektory tříd** a **ID selektory**. Pro pochopení základů CSS nám tato znalost stačí. Je však dobré vědět, že **v jazyce CSS máme k dispozici selektorů mnohem víc**. Existují totiž situace, kdy potřebujeme označit HTML elementy podle složitějších kritérií. Jedním takovým příkladem může být zbarvení odkazu po přejetí kurzorem myši. Pro označení tohoto stavu HTML elementu, bude zajisté potřeba složitější CSS selektor.

Tyto selektory se nazývají **pseudo-elementy** a **pseudo-třídy**. Těmito selektory se v našem kurzu příliš zabývat nebudeme. Ukážeme si prozatím jen pseudo-třídy týkající se odkazů. V rámci stránky **w3schools.com** se můžete podívat na ostatní [pseudo-třídy](https://www.w3schools.com/css/css_pseudo_classes.asp) a [pseudo-elementy](https://www.w3schools.com/css/css_pseudo_elements.asp).

## Pseudo-třídy používané u odkazů

V následující ukázce jsme si vytvořili **dva odkazy**, kterým budeme chtít nastylovat **pseudo-třídy**. V rámci odkazů běžně potřebujeme např. barevně rozlišit, **jak má odkaz vypadat za různých okolností** Např. pokud na něj najedeme kurzorem myši, změní se nám barva a barva se nám mění i v případě, že odkaz byl již navštívený.

Zkusíme si nejprve navštívit jen první odkaz na Engeto a vrátíme se do Codepen. Vidíme, že první odkaz na Codepen má nyní **červenou barvu** oproti **původní modré barvě**. Docílili jsme toho tak, že jsme v CSS nastavili k odkazu pseudo-třídu :visited.

Pokud na odkaz najedeme kurzorem myši, změní se nám barva na hnědou pomocí pseudo-třídy :hover.

Další možností, kterou v CSS máme nastavenou, je **aktivní odkaz**. To znamená moment, kdy jsme na odkaz kliknuli a stále držíme zakliknutý kurzor na místě odkazu. Nastavujeme pomocí pseudo-třídy :active.

Odkaz: <a href="https://engeto.com/">Engeto</a>

<br>

Odkaz: <a href="https://seznam.cz/">Seznam.cz</a>

body {

font-size: 50px;

text-align: center;

margin-top: 30px;

}

/\* navštívený odkaz \*/

a:visited {

color: red;

}

/\* přejetí kurzorem myši nad odkazem \*/

a:hover {

color: brown;

}

/\* vybraný odkaz - klikneme na něj a držíme zakliknutý kurzor \*/

a:active {

color: yellow;

}

# Oficiální dokumentace

## Dokumentace k HTML & CSS

V rámci HTML i CSS existuje velké množství tagů a stylů, které můžeme v rámci kódování webu používat. V průběhu našeho kurzu si spoustu tagů i stylů ukážeme. Nicméně se rozhodně nedostaneme ke všem. Velmi **užitečným pomocníkem** je oficiální dokumentace, která poskytuje nejen **přehled dostupných možností**, ale zobrazují i **příklady použití** včetně možnosti **otestování na příkladech**.

Dokumentace je běžnou každodenní pomůckou každého kodéra i programátora. Doporučujeme se na níže přiložené dokumentace podívat a zkusit si např. najít nějaký tag nebo styl, který jsme se už v kurzu naučili a podívat se, jaké jsou k nim v dokumentaci informace.

V rámci HTML a CSS doporučujeme dokumentaci **W3schools**, která je velmi intuitivní a obsahuje opravdu velké množství informací. Nicméně dokumentací existuje víc, zmíníme např. ještě **MDN Web Docs**.

**W3schools dokumentace**:

[HTML](https://www.w3schools.com/html/)[CSS](https://www.w3schools.com/css/)

**MDN Web Docs dokumentace**:

[HTML](https://developer.mozilla.org/en-US/docs/Web/HTML)[CSS](https://developer.mozilla.org/en-US/docs/Web/CSS)

# Úkoly

## Postup při plnění úkolů

Postup při plnění úkolů je jednoduchý:

1. U každého úkolu máme jednoduché zadání.
2. Některé úkoly můžete zpracovat přímo v interaktivním prostředí codepen.io, některé vyžadují práci v editoru VS Code.
3. Správné řešení si lze ověřit pod editorem kliknutím na **Zobrazit správné řešení**.

## Úkol č. 1

V prvním úkolu si procvičíme, jak vytvářet styly přímo v HTML kódu. Tato varianta se ve většině případů nepoužívá, ale může nastat situace, kdy ji budeš potřebovat použít nebo se s tímto zápisem setkáš v cizím kódu.

V úkolu máš předvyplněné HTML tagy, které obsahují nadpis, odstavec a položky seznamu. Tvým úkolem bude doplnit pomocí přímého zápisu styly rovnou do HTML kódu:

1. Nadpisu <h1> nastav font-color na hodnotu green.
2. Odstavec <p> zvětši pomocí stylu font-size na hodnotu 20px.
3. Názvy knížek <span> podtrhni pomocí stylu text-decoration: underline.

<h1>Moje knihovnička</h1>

<p>V knihovničce se nachází mé oblíbené knížky:</p>

<ul>

<li>

<span>Zaklínač</span> (Andrzej Sapkowski)

</li>

<li>

<span>Duna</span> (Frank Herbert)

</li>

<li>

<span>Proti proudu času</span> (Robert Silverberg)

</li>

<li>

<span>Mrtvý na pekelném vrchu</span> (Juraj Červenák)

</li>

</ul>

## Úkol č. 2

V dalším úkolu si procvičíme zapisování stylů rovněž v HTML kódu, ale pomocí tagu <style>. Opět se jedná o variantu zapisování stylů, která není příliš častá a v kurzu ji nebudeme používat, nicméně v praxi se s tímto zápisem můžeš občas setkat a je důležité vědět, jak s tím pracovat.

Postup:

1. Otevři si **VS Code** a vytvoř soubor **index.html**.
2. Pomocí **!** a stisknutí klávesy **Enter** si vlož do kódu základ pro HTML soubor.
3. Doplň jeden nadpis <h1>a tři odstavce <p>.
4. Ke každéme z odstavců doplň jiný název třídy (**class**). Např. class="prvni-odstavec"
5. Mezi otevírací a uzavírací tag <head> vlož tag <style>.
6. Mezi otevírací a uzavírací tag <style> vlož styl pro nadpis h1 a tři různé styly pro tři odstavce. Pozn. Nezáleží na stylu, důležité je si tento způsob zápisu vyzkoušet. Klidně stačí, když každému odstavci nastavíš jinou barvu písma (color).
7. Nezapomeň uložit a otevři si soubor v prohlížeči.

## Úkol č. 3

Nyní si konečně vyzkoušíme zápis stylů v odděleném souboru, tedy možnost, která se používá nejčastěji. Do HTML souboru v sekci <head> doplníme odkaz na soubor **style.css** a poté tento soubor také vytvoříme. Následně v tomto souboru zapíšeme několik jednoduchých stylů, abychom si vyzkoušeli, že máme CSS soubor správně napojený.

Postup:

1. Otevři si VS Code a vytvoř soubor index.html.
2. Pomocí ! a stisknutí klávesy Enter si vlož do kódu základ pro HTML soubor.
3. Uvnitř tagu <head> doplň tag <link href="./style.css" rel="stylesheet">.
4. Do HTML doplň jeden nadpis <h1> a jeden odstavec <p>.
5. V souboru CSS přidej styl pro nadpis (např. **font-color: red**) a styl pro odstavec (např. **font-size: 18px**)
6. Nezapomeň uložit a otevři si soubor HTML v prohlížeči. Zobrazily se ti správně styly? Zkontroluj si je pomocí vývojářských nástrojů (devtools).

## Úkol č. 4

Následující ukázka obsahuje část HTML kódu a také některé připravené CSS deklarace. HTML kód zobrazíme kliknutím na tlačítko **HTML**. Kliknutím na tlačítko **CSS** zobrazíme tu část editoru sloužící k editaci CSS kódu. Nyní se netrapte tím, že nerozumíte použitým CSS deklaracím. Pro úspěšné dokončení úkolu to není podstatné.

Uvedené CSS pravidlo má špatně zapsaný selektor. Upravte selektor tak, aby CSS pravidlo cílilo správně na HTML element <h1 class="hlavni-nadpis">Vítejte u nás na webu</h1>.

<h1 class="hlavni-nadpis">Vítejte u nás na webu</h1>

/\* opravte uvedený selektor \*/

#hlavni-nadpis {

font-size: 40px;

font-family: fantasy;

text-align: center;

color: blue;

}

## Úkol č. 5

V domácím úkolu si procvičíme více věcí, které jsme se doposud naučili. Budeme pracovat v editoru VS Code, pomocí kterého si vytvoříme HTML soubor a napojíme do něj CSS soubor. Vytváření HTML souborů už jsme si několikrát vyzkoušeli, v zadání úkolu bude tedy jen postup, jak pracovat s CSS a co v rámci úkolu doplnit.

Postup:

1. Vytvoříme si soubor **style.css**.
2. Do souboru **index.html** si nalinkujeme CSS soubor.
3. V HTML souboru si vytvoříme tabulku hokejových zápasů, která bude vypadat následovně:
   * tabulka bude mít 5 řádků <tr>
   * v prvním řádku (hlavičce tabulky) bude 5 buněk <th>
   * v každém dalším řádku bude 5 buněk <td>
   * hlavička tabulky bude mít buňky (pořadí, tým, zápasy, výhry, prohry)
4. V HTML k názvům týmům doplníme třídu tak, abychom díky ní mohli nastylovat v CSS souboru podtržení názvu týmů.
5. Zároveň k této třídě doplníme v HTML každému týmu (na prvních třech místech) jinou třídu **class**, díky které nastavíme v CSS souboru prvním třem týmům jinou barvu textu (color: gold; color: silver; color: brown;). \*Pozn. Bronzová barva nemá v CSS zastoupení pomocí názvu. O barvách se budeme více bavit v další lekci \*:)
6. Pod tabulkou si vytvoříme nadpis <h2> s textem **Český národní tým**. V HTML si doplníme obrázek vítězného týmu. Obrázek si vyberte jakýkoliv nebo použijte tento: (<https://www.mzv.cz/public/1/57/bf/3363250_2101060_EYOF_Ice_hockey_gold_medallist.jpg>). Nezapomeňte na atribut alt.
7. Pod obrázkem si vytvoříme nadpis <h2> s textem **Soupiska českého týmu** a pod nadpisem vytvoříme nečíslovaný seznam hráčů <ul>, ve kterém budou jednotliví hráči jako položky seznamu <li>. Nemusíte doplňovat celou sestavu, stačí pár jedinců.
8. Na závěr pod seznam hráčů doplníme odkaz <a> s atributem target="\_blank" a atributem href, který povede na stránky hokeje. Např.: (<https://www.olympijskytym.cz/article/male-hokejove-nagano-slavime-zlato-z-olympijskeho-turnaje>). Text odkazu bude: **Oficiální stránka**.
9. Doplníme ještě styly pro nadpisy. Nadpis <h1> bude podtržený. Oba nadpisy <h2> budou mít modrou barvu. Obojí nastavte v souboru **style.css**.

## Bonusový domáci úkol

### Vlastní životopis

Tento úkol můžeš odevzdat lektorům, kteří ti pošlou hodnocení (o odevzdávání bude více níže).

Během lekce jsme vytvářeli životopis. Tvým úkolem je vytvořit vlastní životopis úplně od základů.

#### Podrobnější zadání

Cílem tohoto úkolu je, aby sis procvičil **html** a **css**, které jsme se během první lekce naučili. A protože nejlépe se procvičuje psaním vlastního kódu, vyzkoušíš si založení vlastní stránky. Vzhled může vypadat, jak jen budeš chtít. Pokud bys potřeboval trochu detailnější postup, můžeš se inspirovat strukturou níže. Stejně tak můžeš používat kontrukce, které jsme si ukazovali na lekci. Mysli ale na to, že jen zkopírováním kódu bez toho, abys mu rozuměl, škodíš jen sám sobě. Co je naopak silně doporučeno, je používání Google a dalších vyhledávačů (včetně AI), protože o tom je velká část práce programátora a čím dřív na to přijdeš, tím líp. ;)

Aby sis toho prošel co nejvíc, životopis by měl obsahovat tagy jako jsou nadpisy, odstavce, divy, rozpětí, obrázky, odkazy a tabulky. Pro vizuální rozdělení po stránce zatím klidně používej tabulky (ikdyž později se naučíme i zajímavější postupy). Měli byste také používat základní styly CSS pro formátování textu, obrázků a tabulek.

Základní soubory by se měly jmenovat **index.html** (v něm bude tvoje html) a **style.css** (v něm budou tvé CSS styly).

#### Jak může vypadat struktura

1. **Osobní informace**: Použijte značku <div>, abyste vytvořili kontejner pro sekci osobních informací. Použijte značku <h1> pro své jméno a použijte značku <img> k zobrazení vaší fotografie. Použijte značku <p> pro své kontaktní údaje jako email a discord jméno.
2. **Shrnutí**: Použijte značku <div>, abyste vytvořili kontejner pro sekci shrnutí. Použijte značku <h2> pro nadpis a použijte značku <p> pro text.
3. **Vzdělání**: Použijte značku <table>, abyste vytvořili tabulku pro sekci vzdělání. Použijte značky <tr> a <th> pro hlavičky tabulky a použijte značku <td> pro data tabulky.
4. **Pracovní zkušenosti**: Použijte značku <div>, abyste vytvořili kontejner pro sekci pracovních zkušeností. Pro nadpis použijte značku <h2> a pomocí značek <ul> a <li> uveďte své pracovní zkušenosti.
5. **Dovednosti**: Pomocí tagu <div> vytvořte kontejner pro sekci dovedností. Použijte značku <h2> pro nadpis a pomocí značek <ul> a <li> uveďte své dovednosti.

V této úloze byste také měli použít **CSS styly** k formátování textu, obrázků a tabulek, aby vaše CV vypadalo profesionálně a vizuálně atraktivně. Tyto CSS styly je nejlepší vložit do externího CSS soubor a importovat ho do vašeho HTML dokumentu pomocí značky <link> v hlavičce dokumentu.

#### Odevzání úkolu

Pro odevzdání úkolu budeme využívat [Github](https://github.com/), což je služba určená pro správu verzí a sdílení projektů. Na tomto [linku](https://learn.engeto.com/cs/kurz/front-end-developer-akademie/studium/W_kvztH9QoOoZ1FQ9rioRw/jak-nahrat-a-odevzdat-projekt-pomoci-github?originId=pJzdba9LSZyjvq2z2DZZ_g&originLesson=undefined.1:%20Jak%20nahr%C3%A1t%20a%20odevzdat%20projekt%20pomoc%C3%AD%20GitHub) najdeš návod, jak si vytvořit GitHub účet a nahrát tam svůj projekt. Lektorům pak jen zašleš odkaz na svůj projekt přes discord.

# LEKCIA 2

# Barvy a délkové jednotky

## Barvy

V dnešní lekci se podíváme na další CSS vlastnosti, se kterými se můžeme běžně setkat. Ukážeme si, jak zapisovat barvy, jaké existují délkové jednotky a také jak vybrat a zapsat font.

Nyní si pojďme nejprve vysvětlit, jaké barevné vlastnosti dokážeme na webu vůbec změnit. Barva každého HTML elementu je ovlivněna pomocí těchto dvou složek:

* barva popředí
* barva pozadí

Dejme tomu, že máme následující ukázku textu s barevným pozadím:

A blue letter on a yellow background

Description automatically generated

Barvu popředí si lze na webu představit jako **barvu obsahu**, který se nachází uvnitř samotného elementu. V naší ukázce je to barva textu – tudíž modrá. Pokud nic neměníme, mají ve výchozím stavu všechny elementy nastavenou černou barvu popředí.

Barva pozadí je vyjádřena jako **barva celého prvku**, nehledě na to, jaký obsah se v něm nachází. V naší ukázce je to barva žlutá. Ve výchozím stavu mají všechny elementy průhledné pozadí. Díky nastavené barvě pozadí také můžeme vidět, jaký skutečný rozměr element na stránce zabírá.

### Zápis v jazyce CSS

Barvu popředí a barvu pozadí lze upravit pomocí těchto CSS vlastností:

* color - nastavuje barvu popředí daného elementu
* background-color - nastavuje barvu pozadí daného elementu

Pokud bychom chtěli nastylovat nadpis podobný tomu z ukázky na obrázku, vypadal by zápis CSS pravidla nejspíše takto:

<h1>ENGETO Academy</h1>

<span>blokovy element</span>

<span>blokovy element</span>

Css:

h1 {

color: blue;

background-color: yellow;

}

span {

background-color: red;

}

ýsledek vypadá tak, že barva textu je modrá (color: blue;) a barva pozadí celého prvku je žlutá (background-color: yellow;). Možná se teď ptáte, jaká jsou vlastně pravidla pro samotný zápis barvy? A jak vyjádříme například jiný odstín modré? Toto téma otevřeme hned v následující kapitole.

## Vznik barev

Pro zobrazení barev na obrazovce se používá systém barev označovaný jako **RGB** (z angl. Red Green Blue). Je založený na tom, že každá barva se skládá ze tří základních barev (červené, zelené a modré) a to v různém poměru. Jednotlivé složky barev se sčítají a vzniká tak světlo větší intenzity.

A group of multicolored circles

Description automatically generated

Základem je černá a k ní se přičítají další barvy. Čím více barev budeme skládat, tím blíže budeme bíle. Naopak čím méně barev budeme skládat, tím blíže budeme černé. Na úrovni jazyka CSS pak jednotlivé barevné složky vyjadřujeme číslem. Zde tedy bude platit, čím větší číslo, tím více se přiblížíme světlejší barvě a naopak.

## Pojmenované barvy

Než se ale vrhneme do zápis pomocí jednotlivých složek, ukážeme si jeden mnohem jednodušší způsob: přímo pomocí anglického názvu barvy. Do jazyka html byly totiž ty "nejpoužívanjěší" implementovány, takže je můžeme volat přímo.

Ačkoliv se dnes uvádí, že máme k dispozici 140 pojmenovaných barev, je dobré držet se základních šestnácti windowsáckých barev. Jsou to:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| white | blue | red | green | yellow | gray | black | fuchsia |
| silver | teal | purple | maroon | aqua | lime | navy | olive |

### Ukázka

Na následující ukázce je barva textu upravena pomocí pojmenované barvy:

<h1>ENGETO Academy</h1>

h1 {

color: blue;

}

Možností, jak zapisovat barvy v css stylech je spoustu (a o to víc, když se mezi sebou začnou kombinovat). My si tu ukážeme ty 2 nejčastější způsoby: **desítkový zápis barev pomocí funkce rgb()** a **hexadecimální zápis barev**.

## Desítkový zápis barev

U desítkového zápisu barev používáme funkci rgb(), které dáme 3 čísla jako parametry. Tato čísla jsou od 0 do 255 a každé odpovídá jedné složce světla (první červené, druhé zelené a třetí modré). Čísla můžeme ekvivalentně zapsat i procenty (100% odpovídá číslu 255), jedná se ale o méně častý zápis. V praxi nám pak vznikne třeba: rgb(25,78,191).

h1 {

color: rgb(0, 0, 255);

}

## Hexadecimální RGB zápis

Poslední způsob vyjádření všech třech barevných složek je pomocí tzv. hexadecimálního zápisu barev. V CSS zapíšeme barvu pomocí tohoto vzoru:

**#RRGGBB**

Jednotlivé barevné složky jsou zde **RR** (červená), **GG** (zelená) a **BB** (modrá). Zapisujeme je čísly tentokrát v šestnáctkové číselné soustavě a jejich hodnoty mohou být v rozmezí **00** a **FF**. 00 odpovídá 0% a FF je 100%. Modrou barvu bychom tímto způsobem zapsali jako #0000FF (případně je možné psát místo velkých znaků i malé, tzn. #0000ff).

h1 {

color: #0000FF;

}

## Hexadecimální zápis čísel (šestnáctková soustava)

Hexadecimální soustava je číselná soustava základu 16. V hexadecimální soustavě jsou používány znaky 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, doplněné o A, B, C, D, E, F, kdy písmenné znaky reprezentují hodnoty 10–15. Zápis v hexadecimální soustavě bývá doplněn o znak H nebo h použitý jako dolní index za poslední číslicí, případně zkratkou HEX.

Pokud vás zajímá více informací o šestnáctkové soustavě, doporučujeme mrknout na video z [Khanovy akademie](https://khanovaskola.cz/video/52/362/2889-sestnactkova-hexadecimalni-ciselna-soustava), kde se tato soustava rozebírá. Video je v angličtině s českými titulky.

## Složitost zápisů barev

Uvedené způsoby zápisu barev mohou působit celkem komplikovaně. Pokud si navíc představíme, že u každé barvy bude nutné přemýšlet o tom, jak ji poskládat, tak se vůbec nedivím případnému rozčarování. Mohu vás ale ujistit, že barvy téměr nikdy sami skládat nebudete. Zmíněné vyjádření barev totiž používá naprostá většina editorů a grafických programů. Požadovanou barvu tak většinou vybíráme myší a údaj jednoduše zkopírujeme do CSS kódu.

Nejčastěji se můžeme setkat s desítkových a hexadecimálním RGB zápisem barev. Výběr vhodného zápisu barvy je však pouze jen na vás. Všechny uvedené způsoby jsou plně podporovány v běžných webových prohlížečích.

### Procvičení barev na příkladu

Zadání:

1. Otevřete si **Codepen**. V levé části HTML máte připravené tři odstavce, u kterých jsou již doplněné třídy (class).
2. Do prostředního sloupečku doplňte **styly pro jednotlivé odstavce**. Každý odstavec bude mít jinou **barvu písma** a jinou **barvu pozadí**.
3. Výběr barev je čistě na vás. Barvy si můžete vybrat pomocí nástroje **color picker** např. zde: (<https://www.w3schools.com/colors/colors_picker.asp>)

## Průhledné barvy

### rgba(barva, alfa)

Průhledné barvy jsou v css velmi jednoduché, protože je jen jedna možnost, jak jich dosáhnout, a to pomocí funkce **rgba()**. Vychází z toho, že jí zadáme **barvu** a pak tzv. kvocient **alfa**.

#### alfa

Kvocient alfa uvádíme jako desetinné číslo od 0 do 1, které se rgba() zadaná jako poslední parametr a určuje průhlednost barvy. Čím je nížší, tím je daná barva průhlednější.

V praxi to pak vypadá takto:

A screenshot of a cell phone

Description automatically generated

#### barva

Barvu můžeme rgba funkci zadat stejným způsobem jako u rbg() - tedy desítkovými čísly zadáme hodnotu jednotlivých složek.

* např: takto: rgba(0,50,89,0.1)

### Ukázka

Ukázku poloprůhledné barvy si můžeme prohlédnout na následujícím přikladu. První boxík má poloprůhlednou barvu pozadí. Druhý boxík má poloprůhlednou pouze barvu textu.

<div class="box-1">

Tento box má poloprůhledné barevné pozadí

</div>

<div class="box-2">

Tento box má poloprůhlednou barvu textu

</div>

CSS:

.box-1 {

padding: 50px 20px 50px 20px;

margin-bottom: 20px;

width: 400px;

text-align: center;

color: rgb(0, 0, 0);

background-color: rgb(5, 255, 176, 0.3);

}

.box-2 {

padding: 50px 20px 50px 20px;

width: 400px;

text-align: center;

color: rgb(0, 0, 0, 0.3);

background-color: rgb(5, 255, 176);

}

### ak udělat průhlednou barvu z hexadecimálního zápisu?

Občas se nám může stát, že barvu dostaneme v hexadecimálním zápisu a k tomu kvocient průhlednosti. Zároveň jsme si ale před chvílí řekli, že pro průhledné barvy jde použít pouze rgba zápis s desítkovými čísly. Jak to převést?

**Zadání**: barva: #48a2d9,

**Cíl**: použít ji jako průhlednou barvu

Říkali jsme si, že každá hexadecimální barva má svůj rgb() ekvivalent. Stačí tedy převést naši hexadecimální barvu na její rgb() ekvivalent. Na to stačí jednoduchý nástroj z internetu. Třeba tento: [převaděč barev](https://www.rapidtables.org/cs/convert/color/hex-to-rgb.html). Pomocí něj zjistíme, že ekvivalentní k naší barvě je **rgb(72, 162, 217)**. Její škálování pak vypadá takto:

A blue and black text

Description automatically generated

# PROMĚNNÉ V CSS

## Proměnné v css

S barvami v css úzve souvisí používání proměnných (variables) v css, o kterých tu uděláme krátkou odbočku. Hodí se v místě, kdy tu samou hodnotu používáme na několika místech, což se nejčastěji stane právě u barev.

Průběžný příklad: V grafickém návrhu máme použít na pozadí nadpisu h1 barvu rgb(54, 141, 84) a také na border-color u všech tabulek barvu rgb(54, 141, 84). (Cože není zas tak nereálný příklad, protože webové stránky mají často definované pár barev, které se používají všude, aby stránka barevně ladila.)

### Inicializace proměnné

:root{

--jmeno\_promenne: hodnota;

}

:root{

--special\_green: rgb(54, 141, 84);

}

To, co děláme, je, že v pseudotřídě :root\* (což je třída, kterou obsahuje každý element -> proměnné z ní budou přístupné všude) inicializujeme proměnnou --special\_\_green, jejíž hodnota je rgb(54, 141, 84)

\*kromě pseudotřídy :root můžeme použít symbol \* (ten označuje všechny elementy), nebo proměnné přidat i jen některým třdám (což pak ale ničí jejich využití coby globálních veličin).

### Volání proměnné

element{

vlastnost: var(--jmeno\_promenne);

}

Pro náš příklad by to vypadalo takto:

h1 {

background-color: var(--special\_green);

}

table{

border-color: var(--special\_green);

}

Co jsme tu udělali: naspáním var() jsme elementu řekli, že má hledat proměnnou (variable), která se jmenuje --special\_green. Když ji najde, tak zjistí, že její hodnota je rgb(54,141,84) a vlastně je to to samé jako bycho tam napsali rgb(54,141,84).

### Výhody používání proměnných

Jako programátor nemusíte pořád dokola vypisovat složitější zápisy a nahradit je jednodušším.

Pokud se změní tato barva, v celém dokumentu ji změníte jen na jednom místě. Např. pokud by nám grafik zadal, že zelená už není rgb(54, 141, 84) ale rgb(67, 118, 45), tak změnu provede jenom na jednom místě (v elementu :root) a nikde jinde.Kód je tak dlouhodobě udržovatelnější.

Když proměnných máte víc, tak si na proměnné můžete uložit do speciálního souboru.

# DÉLKOVÉ JEDNOTKY

## Úvod do délkových jednotek

Nyní si uděláme malou odbočku. Aby bylo možné pracovat s některými dalšími CSS vlastnostmi, je třeba mít přehled o tom, jakým způsobem můžeme na webu zadávat rozměry. Použitelných délkových jednotek je v CSS celá řada: px, mm, cm, in, pt, pc, rem, em, ex, ch, vw, vh, vmin a procenta. Význam některých z nich si přiblížíme v této lekci.

### Zápis údajů

V CSS mezi číslem a značkou jednotky nepíšeme mezeru (např. 16px, 100%). To stejné pravidlo platí pro záporné hodnoty, kde znaménko mínus neoddělujeme od hodnoty mezerou (např. -12px).

Pokud se rozhodneme zadávat desetinné hodnoty, **píšeme tečku jako oddělovač celočíselné a desetinné části**, nikoliv čárku, jak bývá zvykem v českém jazyce. Příklad takového zápisu: 1.25em.

### Pixel

Základní délkovou jednotkou je tzv. **pixel**. Lze si ho představit jako jeden svítící bod na obrazovce, respektive jeden bod obrázku (charakteristický svým jasem a barvou a definovaný např. ve formátu RGB). Když se pak díváme na obrazovku, tak je celá složená z čtvercové sítě pixelů. Každý bod lze pak jednoznačně identifikovat podle jeho souřadnic.

V dnešní době už není pixel z pohledu jazyka CSS to stejné jako pixel na obrazovce (hardwarový pixel). V minulosti tomu tak bývalo. S nástupem nových technologií se však objevili displeje s vícenásobným rozlišením (Retina, Amoled, ...). Prohlížeče tak hardwarové rozlišení přepočítají do takzvaného CSS rozlišení. Laicky řečeno, zabraňují tomu, aby se prvky webu definované v pixelech jevili miniaturní na nových displejích. Retina displeje mají dvojnásobné rozlišení. Když na nich v CSS vykreslíme element velký 1 pixel, bude ve skutečnosti zabírat 4 hardwarové pixely (půjde o mřížku 2 × 2 obrazových bodů).

**V jazyce CSS používáme pro jednotku pixel označení px.** Pokud například budeme chtít vyjádřit rozměr o velikosti 40 pixelů, zapíšeme ho v CSS jako 40px.

### Rozdělení měrných jednotek

Měrné jednotky dělíme do dvou skupin:

* Absolutní jednotky
* Relativní jednotky

## Absolutní měrné jednotky

Absolutní jednotky představují rozměry známé z reálného světa (mm, cm, in, apod.). Špatně se však u nich odhaduje, jak se ve výsledku zobrazí. Lze je použít pouze tehdy, jsou-li předem známy rozměry výstupního média. Zde je uvádíme pouze pro přehled. Při psaní stylů na webu se téměř nepoužívají a svá uplatnění nachází maximálně při tisku.

| **Měrná jednotka** | **Název** | **Převod** |
| --- | --- | --- |
| pt | Typografický bod | 1pt = 0,3528 mm |
| pc | Typografická jednotka "pica" známá také jako "cicero" | 1pc = 0,0294 mm |
| mm | Milimetr |  |
| cm | Centimetr |  |
| in | Anglický palec (angl. inch) | 1in = 25,4 mm |

## Relativní měrné jednotky

| **Měrná jednotka** | **Název** | **Význam** |
| --- | --- | --- |
| em | "eM" | Rozměr 1em je stejný jako velikosti písma daného HTML elementu. Pokud má element velikost písma 16px, platí, že rozměr 2.5em (použitý uvnitř takového elementu) bude stejný jako 40px. |
| ex | "eX" | Podobná jednotka jako em s tím rozdílem, že se její rozměr neodvozuje z velikosti písma, ale z výšky malého písmene x. |
| rem | Kořenové "eM" | Rozměr jednotky rem se počítá podobně jako u jednotky em. V obou případech se odvozuje z velikosti písma. Velikost 1em je stejná jako velikost písma daného HTML elementu. Rozměr 1rem spočítáme jako velikost písma kořenového HTML elementu celého webu (což bývá typicky element <html>). |
| px | Pixel | Tato jednotka je občas milně považována za absolutní. Ve skutečnosti se však jedná o relativní jednotku. Je to dáno tím, že prohlížeč na obyčejné obrazovce chápe 1px jako jeden obrazový bod. Pokud ale 1px zobrazíme na obrazovce s vyšším rozlišením (např. Retina displej), zabere nám jeden pixel ve skutečnosti 4 obrazové body. |
| % | Procento | Pokud jako měrnou jednotku použijeme údaj v procentech, počítá se výsledná hodnota jako procentuální poměr z velikosti rodičovského HTML elementu. |
| vw | "vw" | Zkratka z anglického výrazu "viewport width". 1vw označuje 1% šířky viewportu (zjednodušeně šířky okna). |
| vh | "vh" | Zkratka z anglického výrazu "viewport height". 1vh označuje 1% výšky viewportu (zjednodušeně výšky okna). |
| vmin | "vmin" | Zkratka z anglického výrazu "viewport minimum". 1vmin reprezentuje menší hodnotu z porovnání 1vw a 1vh. |
| vmax | "vmax" | Zkratka z anglického výrazu "viewport maximum". 1vmax reprezentuje větší hodnotu z porovnání 1vw a 1vh. |

## Ukázky použití relativních jednotek

Konkrétní použití relativních jednotek závisí vždy na dané situaci. Nejčastěji se na webu setkáme s jednotkami jako px, %, em a rem. Nyní si na několika jednoduchých ukázkách si přiblížíme různé situace, kde dokážeme těchto jednotek využít:

### Jednotka pixel

Jednotka px nachází uplatnění převážně tam, kde potřebujeme zadat stabilní neměnný rozměr. Používá se při tvorbě kostry (layoutu) celé stránky. Dokážeme tak například nadefinovat, že určitý element bude mít pevnou šířku o velikosti 400px.

### Jednotka procento

Údaj v procentech využijeme tam, kde chceme například říci, že rozměr elementu bude poloviční vzhledem k rozměrům svého rodiče. V pixelech sice dokážeme zapsat poloviční velikost, ale museli bychom znát přesný rozměr rodiče, nový rozměr si spočítat a doufat, že se velikost rodiče se již nezmění. V procentech ale poloviční velikosti zapíšeme snadno. Stačí uvést hodnotu 50%.

### Jednotka em

Definice zní: Rozměr 1em je stejný jako velikosti písma daného HTML elementu.

Toho se dá využít například při určování velikosti písma vnořených elementů uvnitř dokumentu. Způsoby nastavování písma sice probereme až v následující lekci, nicméně princip jednotky em si můžeme teoreticky ukázat již nyní. Představte si modelovou situaci, kdy máme dokument obsahující odstavce a nadpisy.

1. Nadpisům lze díky em říci, že velikost písma bude vždy 1.5× větší než velikost písma nadřazeného elementu. Velikost nadpisů zapíšeme jako 1.5em.
2. Pokud někdy v budoucnu změníme velikost písma dokumentu, velikosti nadpisů se nám přizpůsobí.

Když pak zanoříme do sebe elementy používající velikosti definované v em, budou platit následující dědičnosti:

<div class="element-1">

Velikost textu je 18px

<div class="element-2">

Velikost textu je 1.25em (stejná jako 22.5px)

<div class="element-3">

Velikost textu je 1.25em (stejná jako 28.125px)

<div class="element-4">

Velikost textu je 1.25em (stejná jako 35.15625px)

</div>

</div>

</div>

</div>

CSS:

.element-1 {

font-size: 18px;

}

.element-2 {

font-size: 1.25em;

padding: 15px;

background-color: #05ffb0;

}

.element-3 {

font-size: 1.25em;

padding: 15px;

background-color: #ff3860;

}

.element-4 {

font-size: 1.25em;

padding: 15px;

background-color: #ffdd57;

}

.element-1 {

font-size: 18px;

}

.element-2 {

font-size: 1.25em;

padding: 15px;

background-color: #05ffb0;

}

.element-3 {

font-size: 1.25em;

padding: 15px;

background-color: #ff3860;

}

.element-4 {

font-size: 1.25em;

padding: 15px;

background-color: #ffdd57;

}

### Jednotka rem

Definice zní: Rozměr jednotky rem se počítá podobně jako u jednotky em. V obou případech se odvozuje z velikosti písma. Velikost 1em je stejná jako velikost písma daného HTML elementu. Rozměr 1rem spočítáme jako velikost písma kořenového HTML elementu celého webu (což bývá typicky element <html>).

Laicky řečeno, pokud se chceme vyhnout dědičnosti velikostí viditelné na předchozí ukázce, tak můžeme využít jednotky rem. Ztratíme tím ale určitou flexibilitu. Velikosti v rem se totiž neodvozují z velikosti písma uvnitř konkrétního elementu, nýbrž budou vždy relativní k velikosti písma na elementu <html>.

Webový vývojáři často přepočítávají jednotky do pixelů, aby mohli porovnat výsledné velikosti elementů v prohlížeči a ujistit se, zda odpovídají představám designera. Přepočet em do pixelů je složitější díky již zmíněné dědičnosti. U velikosti v rem je přepočet o poznání snadnější a například velikost 1.25rem bude všude na webu představovat stejný rozměr:

Kořenový <code>&lt;html&gt;</code> element má velikost písma 18px

<div class="element-1">

Velikost textu je 1.25rem (stejná jako 22.5px)

<div class="element-2">

Velikost textu je 1.25rem (stejná jako 22.55px)

<div class="element-3">

Velikost textu je 1.25rem (stejná jako 22.55px)

</div>

</div>

</div>

CSS:

html {

font-size: 18px;

}

.element-1 {

font-size: 1.25rem;

padding: 15px;

background-color: #05ffb0;

}

.element-2 {

font-size: 1.25rem;

padding: 15px;

background-color: #ff3860;

}

.element-3 {

font-size: 1.25rem;

padding: 15px;

background-color: #ffdd57;

}

# TYPOGRAFIE

## Zarovnání textu

Pomocí CSS dokážeme ovlivnit vzhled textu hned několika způsoby. První vlastnost, kterou se naučíme měnit, je zarovnání textu.

Většina HTML elementů má ve výchozím stavu zarovnaný text k levému okraji stránky. Příkladem takového chování může být následující ukázka, kde máme text bez jakýkoliv dodatečných stylů:

Výchozí chování je však možné upravit. V CSS nám k tomu poslouží vlastnost text-align. Ta nemění pouze zarovnání textu, ale ovlivňuje zarovnání libovolného obsahu uvnitř elementu.

Text dokážeme zarovnat čtyřmi různými způsoby. Vlastnost text-align může nabývat jedné z následujících hodnot:

* left - Text bude zarovnaný v rámci elementu doleva.
* center - Text bude zarovnaný na střed.
* right - Text bude zarovnaný doprava.
* justify - Text bude zarovnaný do bloku (jednotlivá slova jsou rovnoměrně roztažena tak, aby vyplnila celou šířku řádku). Toto zarovnání se používá především v tištěných dokumentech. Použití na webu není příliš vhodné, protože zatím nedokážeme automaticky rozdělovat dlouhá slova na konci řádku. Při roztahování textu tak dochází k vzniku až nezvykle velikých mezer mezi slovy.

### Ukázka zarovnání textu doprava

h1, h2, p {

text-align: right;

}

### Ukázka zarovnání nadpisů na střed

h1, h2 {

text-align: center;

}

### Ukázka zarovnání textu do bloku

Všimněte si jednotlivých odstavců. Aby bylo vůbec možné jejich text zarovnat do bloku, dochází k roztahování jednotlivých slov. Mezi slovy tak vznikají rozdílné velikosti mezer

p {

text-align: justify;

}

## Styl písma

Pokud hovoříme stylu písma, máme z pohledu webové typografie k dispozici dva různé styly:

* normální písmo
* kurzívu (neboli skloněné písmo)

V CSS změníme styl textu pomocí vlastnosti font-style. Ta může nabývat jedné z těchto hodnot:

* normal - Slouží po nastavení normálního stylu písma.
* italic - Text bude formátovaný kurzívou.

### Ukázka

Texty nadpisů na následující ukázce jsou ostylovány tak, aby se zobrazovali kurzívou:

h1, h2 {

font-style: italic;

}

Pomocí CSS dokážeme zrušit kurzívu u těch elementů, u kterých je použita jako jejich výchozí styl písma:

em {

font-style: normal;

}

## Tučnost písma

Další vlastnost písma, kterou dokážeme ovlivnit, je jeho tučnost. Webové prohlížeče znají dvě základní rozdělení:

* Normální písmo
* **Tučné písmo**

Ke změně tučnosti textu slouží v CSS vlastnost font-weight. Ta může nabývat [různé hodnoty](https://www.w3schools.com/cssref/pr_font_weight.asp). Např.:

* normal - Slouží pro nastavení normálního (netučného) písma. U většiny HTML elementů je netučné písmo součástí výchozího vzhledu textu.
* bold - Text bude zobrazen tučným písmem.

### Různé stupně tučnosti

Kromě uvedených dvou možností normal a bold, můžeme jako hodnotu vlastnosti font-weight uvést také číselný údaj. Povolené jsou násobky 100 v rozmezí 100 až 900 (tj. 100, 200, 300, ..., 900). Takové číslo nám umožnuje přesněji nastavit různé stupně tučnosti písma. 100 by znamenalo nejslabší písmo, 900 naopak nejtučnější. Zde je však nutné podotknout, že takto jemné škálování tučnosti nepodporuje každé písmo na webu. Obvykle v prohlížečích fungují pouze hodnoty 400 (které tučností odpovídá hodnotě normal) a 700 (které tučností odpovídá hodnotě bold). Případný zápis v CSS by vypadal takto:

.tucny-text {

font-weight: 700;

}

### Ukázka

Na následující ukázce máme část HTML kódu, kde jsou názvy měst obaleny elementem <span class="vyrazny-text">Brno</span>. V CSS je pak definované pravidlo, které třídě .vyrazny-text nastavuje tučný vzhled písma. Slova **Brno** a **Ostrava** jsou díky tomu zobrazeny tučně. Názvy ostravských řek máme obaleny elementem <span class="tucny-text"> Odry, Opavy, Ostravice a Lučiny</span>. Tučnost zde určujeme pomocí číselné hodnoty **700**. Výsledná tučnost pak vypadá stejně.

h1 {

font-weight: normal;

}

.bold-text {

font-weight: bold;

}

.tucny-text {

font-weight: 700;

}

## Velikost textu

Výchozí velikost textu se na webu liší element od elementu. Stačí třeba jen porovnat několik nadpisů a text běžného odstavce. Na první pohled můžeme vidět, že hlavní nadpis <h1> bývá vypsaný větším textem, aby vynikla jeho důležitost. Jakmile ale začneme přizpůsobovat vzhled webu, přestanou nám většinou výchozí velikosti textu stačit.

Naštěstí velikost textu se dá snadno přizpůsobit. V CSS k tomu slouží vlastnost font-size. Hodnotu této vlastnosti můžeme zadat několika způsoby:

* **Číslo s uvedenou jednotkou** - Velikost textu můžeme zadat pomocí délkových jednotek, které jsou podporované v jazyce CSS. Např. px, em, rem, apod.
* **Procentem** - Velikost textu bude procentuální poměr vzhledem k velikosti textu nadřazeného elementu. Tzn. font-size: 200% znamená, že velikost textu bude dvakrát větší než je velikost textu rodiče.

### Ukázka

Velikost textu zadaná v pixelech:

p {

font-size: 24px

}

Následující ukázka zobrazuje velikost textu zadanou v jednotkách em. Jak můžeme vidět, výsledná velikost je vždy relativní vzhledem k velikosti písma nadřazeného prvku:

<div class="element-1">

Velikost textu je 18px

<div class="element-2">

Velikost textu je 1.25em (stejná jako 22.5px)

<div class="element-3">

Velikost textu je 1.25em (stejná jako 28.125px)

<div class="element-4">

Velikost textu je 1.25em (stejná jako 35.15625px)

</div>

</div>

</div>

</div>

.element-1 {

font-size: 18px;

}

.element-2 {

font-size: 1.25em;

padding: 15px;

background-color: #05ffb0;

}

.element-3 {

font-size: 1.25em;

padding: 15px;

background-color: #ff3860;

}

.element-4 {

font-size: 1.25em;

padding: 15px;

background-color: #ffdd57;

}

Následující ukázka zobrazuje velikost textu zadanou v jednotkách rem. Zde se naopak velikost textu odvozuje od velikosti písma kořenového <html> elementu:

Kořenový <code>&lt;html&gt;</code> element má velikost písma 18px

<div class="element-1">

Velikost textu je 1.25rem (stejná jako 22.5px)

<div class="element-2">

Velikost textu je 1.25rem (stejná jako 22.55px)

<div class="element-3">

Velikost textu je 1.25rem (stejná jako 22.55px)

</div>

</div>

</div>

html {

font-size: 18px;

}

.element-1 {

font-size: 1.25rem;

padding: 15px;

background-color: #05ffb0;

}

.element-2 {

font-size: 1.25rem;

padding: 15px;

background-color: #ff3860;

}

.element-3 {

font-size: 1.25rem;

padding: 15px;

background-color: #ffdd57;

}

Velikost textu zadaná v procentech:

<div class="element-1">

Velikost textu je 22px

<div class="element-2">

Velikost textu je 85% (stejná jako 18.7px)

<div class="element-3">

Velikost textu je 85% (stejná jako 15.895px)

<div class="element-4">

Velikost textu je 85% (stejná jako 13.51075px)

</div>

</div>

</div>

</div>

.element-1 {

font-size: 22px;

}

.element-2 {

font-size: 85%;

padding: 15px;

background-color: #05ffb0;

}

.element-3 {

font-size: 85%;

padding: 15px;

background-color: #ff3860;

}

.element-4 {

font-size: 85%;

padding: 15px;

background-color: #ffdd57;

}

## Výška řádku

Výšku řádku si lze představit jako množství prostoru, který zabírají jednotlivé řádky (typicky řádky textu) uvnitř elementu. Někdy je tato vlastnost nazývaná také jako řádkování.

Výška řádku je graficky vyznačena i na následujícím diagramu:

A blue and red text on a black background

Description automatically generated

Výška řádků se dá snadno změnit. Správně nastavená výška řádku dokáže výrazně **zlepšit čitelnost dlouhých textů**. Takto může vypadat dlouhý text, který má původní (nezměněnou) výšku řádku:

Pokud ale o něco zvýšíme výšku řádku, začne stejný text působit "vzdušnějším" dojmem. Takový text je pro člověka lépe čitelnější. Však porovnejte sami:

p {

font-size: 16px;

line-height: 1.2;

}

### Nastavení v jazyce CSS

V CSS upravíme výšku řádku pomocí vlastnosti line-height. Její hodnota může být:

* normal - Je to výchozí hodnota. Výška řádku bude stejná, jako je výška nejvyššího prvku na řádku.
* Desetinné číslo bez jednotky (např. 1.2). Výška řádku se pak spočítá jako násobek uvedeného čísla a velikosti fontu.
* Číslo s uvedenou jednotkou (např. 18px). Výška řádku je pevně určená tímto rozměrem. Použít lze jakýchkoliv podporovaný rozměr v CSS (tj. v jednotkách px, %, em, rem, ...). Velikost samotného písma nijak neovlivňuje celkovou velikost řádku.

V současnosti se doporučuje u hodnoty line-height používat čísla bez jednotek. Nevzniká totiž žádná závislost na velikosti fontu. Pokud v budoucnu dojde ke změně velikosti písma, výška řádku se nám díky tomu automaticky přepočítá.

### Ukázka

Následuje ukázka nastavené výšky řádku pomocí hodnoty 1.5. Protože velikost fontu je 16px, bude výsledná hodnota spočítaná jako násobek 1.5 \* 16px. Lze tedy říci, že rozměr line-height: 1.5; bude v tomto případě stejný jako line-height: 24px;.

p {

font-size: 16px;

line-height: 1.2;

}

## Podtržené písmo

Vybraný text je možné zvýraznit pomocí podtržení. V praxi se nejčastěji setkáme s podtrženým textem u odkazů. Nic však nebrání tomu, aby byl takto "ozdoben" i text, který odkazem není. V CSS toho docílíme pomocí vlastnosti text-decoration. Ta může nabývat [různé hodnoty](https://www.w3schools.com/cssref/pr_text_text-decoration.asp). Nejčastěji se však používají tyto:

* none - Obyčejný text bez podtržení. Jedná se o výchozí vlastnost většiny elementů.
* underline - Text bude podtržený souvislou čárou v barvě písma.

Na následující ukázce jsou všechny nadpisy podtržené a naopak všem odkazům bylo jejich výchozí podtržení zrušeno. V praxi ale doporučuji podtržení u odkazů zachovat. Řada uživatelů se podle toho orientuje a mohlo by být pro ně složitější rozpoznat klikatelné odkazy.

h1, h2 {

text-decoration: underline;

}

a {

text-decoration: none;

}

## Textové transformace

Pomocí CSS dokážeme ovlivnit velikost písmen v textu. Slouží nám k tomu vlastnost text-transform, která určuje způsob transformace textu na velká nebo malá písmena. Nabývat může jedné z následujících hodnot:

* none - Na vybraný text se nepoužije žádná textová transformace. Jedná se výchozí nastavení textu.
* uppercase - Všechna písmena budou převedena na velká.
* lowercase - Všechna písmena budou převedena na malá.

V samotném HTML kódu nám zůstává původní nezměněný text. Celá transformace se děje až v prohlížeči. Možná se ptáte, proč rovnou text nepřepíšeme do velkých písmen v HTML? Dejme tomu, že budeme mít všechny hlavní <h1> nadpisy napsané velkými písmeny. Nebude nutné psát žádnou potřebnou CSS deklaraci. Na druhou stranu, co se stane, když se v budoucnu rozhodneme, že hlavní nadpis bude vypadat jinak a bez všech velkých písmen? V takovém případě nám nezbyde nic jiného, než vše ručně přepsat. Kdyby textová transformace byla řešena jen pomocí CSS, celá změna vzhledu by byla nesrovnatelně jednodušší.

Na následující ukázce můžeme vidět použití vlastnosti text-transform: uppercase;, která převádí text nadpisů na velká písmena:

h1, h2 {

text-transform: uppercase;

}

# RODINA PÍSMA

## Co je to rodina písma?

Pokud jste někdy použili textový editor (například Word), tak jste se zajisté setkali s možností změny typu písma. Tento "typ písma" pak v typografii nazýváme jako **rodina písma**. Pojem "rodina" je zde použitý z toho důvodu, protože sdružuje dohromady různé řezy stejného písma (řezy písma jsou: normální písmo, tučné písmo, kurzíva).

Na následujících ukázkách můžeme vidět porovnání tří různých rodin písma. Vyznačeny jsou zde také odlišné řezy písma, které jsou vždy k dispozici v rámci jedné rodiny:

See the Pen by ENGETO () on .

.sans-serif-family {

font-family: Arial;

}

.serif-family {

font-family: "Times New Roman";

}

.monospace-family {

font-family: "Courier";

}

### Nastavení rodiny písma v CSS

Pro nastavení jiné rodiny písma slouží v CSS vlastnost font-family. Hodnotu této vlastnosti lze zadat třemi způsoby:

* Název konkrétní rodiny písma
* Název obecné rodiny písma
* Lze uvést dokonce několik rodin písma najednou

## Konkrétní rodina písma

Konkrétní rodinu písma vybereme tak, že jednoduše uvedeme celý její název. Například:

Ukázka kódu7

ZKOPÍROVAT KÓD

font-family: Arial;

Jestliže má rodina víceslovný název, tak je třeba tuto hodnotu uzavřít do uvozovek. V opačném případě se může stát, že prohlížeč nerozpozná uvedenou CSS deklaraci:

Ukázka kódu8

ZKOPÍROVAT KÓD

font-family: "Times New Roman";

Mějme na paměti, že pokud v CSS nastavujeme jinou rodinu písma, tak nová volba nemusí být k dispozici všem návštěvníkům. Při prohlížení webu se totiž **používají písma dostupná pouze na zařízení uživatele**, který si momentálně web prohlíží (tj. na jeho počítači, tabletu či mobilu). Existuje však několik rozšířených rodin písma (viz dále), které by měli být nainstalované na většině dnes používaných zařízeních a my je tak můžeme bez obav použít. Pokud si s touto nabídkou nevystačíme, lze pomocí složitějších postupů načíst úplně nové písmo. Načítání nového písma však přesahuje rámec tohoto základního kurzu, proto se tomuto tématu nebudeme nyní věnovat.

Následující rodiny písma by měli být dostupné na běžně dostupných zařízeních:

<h3>Arial</h3>

<p style="font-family: Arial;">Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec vitae arcu. Nunc tincidunt ante vitae massa.</p>

Rozsáhlejší seznam rodin písma nalezneme na adrese <https://www.cssfontstack.com/>, kde je možné vyčíst i údaj o jejich průměrné dostupnosti.

Většina prohlížečů má jako svou výchozí rodinu písma nastavnou volbu Times New Roman. To se vyznačuje především patkovým stylem písma. Pokud tedy v CSS definici font-family vynecháme, bude veškerý obsah vypsán právě pomocí výchozího písma Times New Roman.

## Obecná rodina písma

Pokud až tak netrváme na konkrétním písmu, ale zajímá nás spíš obecný typ (patkové, bezpatkové, ozdobné, apod.), můžeme jako hodnotu font-family uvést název **obecné rodiny písma**. Obecné rodiny písma představují několik skupin, kde každá taková skupina obsahuje typově podobná písma. Výhodou je, že se nemusíme starat o to, jestli bude vybrané písmo podporované na koncovém zařízení návštěvníka webu. Prohlížeč prostě použije první dostupný font z vybrané obecné rodiny. Musíme se však smířit s tím, že například návštěvníkům s operačním systémem Windows se nakonec zobrazí jiné písmo než naříklad lidem, kteří si prohlíží web na zařízeních od Apple. Styl písma však zůstane pokaždé zachovaný.

Obecné rodiny písma máme následujícího typu:

#### 1. Patková písma

* Tato skupina obsahuje písma, která se vyznačují tzv. patkami - tah písma je vždy příčně zakončen.
* Zápis v CSS: font-family: serif;
* Obsahuje rodiny písma jako: Times New Roman, Georgia, ...

html {

font-family: serif;

}

#### 2. Bezpatková písma

* Jedná se o opak patkového písma. Tato skupina obsahuje písma, u kterých není tah písma zakončený patkou.
* Zápis v CSS: font-family: sans-serif;
* Obsahuje rodiny písma jako: Arial, Verdana, Helvetica, ...

html {

font-family: sans-serif;

}

#### 3. Neproporciální písma

* Skupina obsahující písma, u kterých platí, že každý znak má stejnou šířku. Tyto písma nachází svá uplatnění v situacích, kdy potřebujeme na webu vypsat text, kde budou jednotlivé znaky vždy pod sebou. Např. výpis zdrojového kódu.
* Zápis v CSS: font-family: monospace;
* Obsahuje rodiny písma jako: Courier, Courier New, Lucida Console, ...

html {

font-family: monospace;

}

#### 4. Kaligrafovaná písma

* Písma v této skupině mají tah linky, která vytváří dojem ručně psaného textu. Jednotlivé znaky působí, že na sebe úplně nebo částečně navazují.
* Zápis v CSS: font-family: cursive;
* Obsahuje rodiny písma jako: Comic Sans, Brush Script MT, ...

html {

font-family: cursive;

}

#### 5. Ozdobná písma

* Speciální skupina obsahující dekorativní písma, které se vyznačují především hravým pojetím jednotlivých znaků.
* Zápis v CSS: font-family: fantasy;
* Obsahuje rodiny písma jako: Western, Papyrus, ...

html {

font-family: fantasy;

}

## Zápis více rodin písma najednou

Protože nikdy není 100% jistota, že konkrétní písmo bude koncové zařízení podporovat, je nejlepším řešením vybrat více rodin písma a ty pak uvést v rámci jedné společné definice font-family. Pří zápisu je pak od sebe oddělujeme čárkou. Příkladem může být následující CSS deklarace:

Ukázka kódu9

ZKOPÍROVAT KÓD

font-family: Arial, Helvetica, sans-serif;

Prohlížeč pak uvedenou deklaraci interpretuje **zleva doprava**. Pokud narazí na dostupné písmo, tak ho použije. Pokud ne, tak postupuje v hledání dál. V našem případě zkouší postupně použít rodinu Arial, pak Helvetica a pokud neuspěje, tak na závěr vybere nějaké písmo z obecné rodiny sans-serif.

Při zápisu font-family bývá nejlepším řešením, když nejprve zapíšeme jednu až dvě konkrétní rodiny písma, které jsou podobného vzhledu, a pak na závěr uvedeme obecnou rodinu písma. Pokud se vrátíme k situaci z naší ukázky, tak Arial i Helvetica jsou bezpatková písma, která jsou si navzájem velmi podobná. Obecná rodina pro bezpatková písma má označení sans-serif. CSS deklarace pro bezpečný výběr písma Arial je pak poskládaná z těchto možností: font-family: Arial, Helvetica, sans-serif;.

html {

font-family: Arial, Helvetica, sans-serif;

}

## Rodina písma z Google Fonts

V běžné praxi se často stává, že si s vestavěnými rodiny písem nevystačíme. Pokud se setkáme s tím, že nám UX designer nebo grafik předá návrh webu, kde je jiná rodina písma, než kterou můžeme v rámci **CSS** běžně použít, musíme si písmo do projektu natáhnout ručně. Nejjednodušším způsobem je využití služby **Google Fonts**, která nabízí velké množství rodin písma zdarma.

Web <https://fonts.google.com/> již na úvodní stránce uvádí seznam, ze kterých můžeme vybírat. Existuje tu i možnost **vyhledávání**, pokud už víme, kterou rodinu písma potřebujeme. Zároveň si tu můžeme vyzkoušet, jak písmo vypadá v různých velikostech pomocí **posuvníku velikostí** nebo si lze vyzkoušet napsat **vlastní text**, který se tu rovnou zobrazí a odhalí, která písma neumí diakritiku.

### Postup získání písma

Pokud se pro některé písmo rozhodneme, budeme ho chtít do projektu natáhnout. V našem případě vybereme např. rodinu písma **Roboto**.

1. Klikneme na zvolené písmo a otevře se nám stránka s detaily písma.
2. Zobrazí se informace o tvůrci, přehled znaků, licence a další informace.
3. Níže se zobrazují řezy písma, tedy různé tloušťky a kurzíva. Pomocí **+ Select this style** si zvolíme, o které řezy máme zájem. Může jich být víc, mohou být všechny. Záleží, jaké poždavky jsou od designéra webu.
4. V našem příkladu jsme vybrali tloušťky Light 300, Medium 500 a Bold 900.
5. V pravém sloupečku se zobrazí náš výběr.
6. Zbývá nám překopírovat kód ze sekcve **Use on the web**. Kód na šedém pozadí v možnosti <link> zkopírujeme do HTML. V **Codepen** ho umístíme úplně nahoru. Ve VS Code editoru bychom ho umístili do sekce <head> před uzavírací tag </head>.
7. Vrátíme se na stránku s Google Fonts a překlikneme se pomocí "puntíku" na možnost **@import**. Zkopírujeme opět kód na šedém pozadí a v Codepen ho umístíme nahoru na začátek CSS.
8. Nyní už stačí na naši stránku nastavit jen font-family: 'Roboto', sans-serif; V případě, že chceme mít rodinu písma **Roboto** použitou na celý náš web, nastavíme font-family rovnou na tag <body>. Zápis bude vypadat takto:

Ukázka kódu10

ZKOPÍROVAT KÓD

body {

font-family: 'Roboto', sans-serif;

}

1. Nyní si na jednotlivé části textu (odstavce) můžeme vyzkoušet tloušťky 300, 500, 900, které jsme si vybrali.

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;500;900&display=swap" rel="stylesheet">

body {

font-family: 'Roboto', sans-serif;

}

# ÚKOLY

## Úkol č.1 - Barvy

V tomto úkolu si procvičíme zápis barev a zároveň si zopakujeme některé HTML tagy z minulých lekcí.

**Postup:**

1. K tagům <h2> doplníme třídu dle vlastní volby a v sloupečku CSS této třídě nastavíme jakoukoliv barvu pomocí hexadecimálního zápisu barev.
2. Tagům s odkazy <a> nastavíme v CSS jakoukoliv barvu, tentokrát pomocí RGB zápisu. Odkazům můžeme rovněž nastavit třídu (class) nebo jim můžeme styl nastavit přímo pomocí **typového selektoru**. Odkazy jsou zde jen pro ilustrační příklad, reálně nikam neodkazují, proto je atribut href prázdný.

<h1>Nejnovější informace</h1>

<h2>Zprávy z domova</h2>

<ul>

<li><a href="">Letošní Vánoce budou opět na blátě</a></li>

<li><a href="">ČT uvede nové seriály a řadu dokumentů</a></li>

<li><a href="">Praha se umístila mezi nejlepšími městy</a></li>

</ul>

<h2>Zprávy ze zahraničí</h2>

<ul>

<li><a href="">Řidička sjela do vody u Niagarských vodopádů</a></li>

<li><a href="">Na Novém Zélandu plánují v roce 2025 úplný zákaz kouření</a></li>

<li><a href="">Britský premiér Johnson se stal opět otcem</a></li>

</ul>

<h2>Sport</h2>

<ul>

<li><a href="">Obojživelnice Ledecká chystá změnu!</a></li>

<li><a href="">V NHL hrozí skandál. Pustí Vejmelku do haly? </a></li>

<li><a href="">Překvapení v nominaci na mistrovství světa</a></li>

</ul>

## Úkol č. 2 - délkové jednotky

V tomto úkolu si procvičíme základní délkové jednotky (% a px), které budeme v tomto kurzu používat nejvíce.

**Postup:**

1. V HTML máme připravenou kartu žáby, která obsahuje nadpis, obrázek, odstavec a element <div>.
2. Nejprve si k obrázku <img> doplníme třídu, pomocí které následně v CSS nastavíme šířku width na hodnotu **20%**.
3. Dále si k odstavci <p> také doplníme třídu a v CSS pak odstavci nastavíme šířku na **400px**.
4. Na závěr doplníme k tagu <div> třídu, které v CSS nastavíme velikost písma na 18px. Díky tomu, že jsme použili tag <div>, který nyní "obaluje jak text, tak odkaz, tak se nám nastavená velikost písma bude aplikovat na obojí.

## Úkol č. 2 - délkové jednotky

V tomto úkolu si procvičíme základní délkové jednotky (% a px), které budeme v tomto kurzu používat nejvíce.

**Postup:**

1. V HTML máme připravenou kartu žáby, která obsahuje nadpis, obrázek, odstavec a element <div>.
2. Nejprve si k obrázku <img> doplníme třídu, pomocí které následně v CSS nastavíme šířku width na hodnotu **20%**.
3. Dále si k odstavci <p> také doplníme třídu a v CSS pak odstavci nastavíme šířku na **400px**.
4. Na závěr doplníme k tagu <div> třídu, které v CSS nastavíme velikost písma na 18px. Díky tomu, že jsme použili tag <div>, který nyní "obaluje jak text, tak odkaz, tak se nám nastavená velikost písma bude aplikovat na obojí.

<h1>Kuňka východní</h1>

<img src="https://www.stoplusjednicka.cz/sites/default/files/obrazky/2020/09/02\_01\_kunka\_vychodni.jpg" alt="Kuňka východní">

<p>Kuňka východní je žába dorůstající délky 4 – 6 cm, vzhledem připomíná kuňky žijící v našich oblastech. Hřbet těla má bradavičnatý povrch s jedovými žlázami (z nichž vylučuje mléčnou tekutinu, která dráždí sliznici tlamy a oči nepřítele) a je zbarven do zelených případně hnědých odstínů v kombinaci s černou. Bříško má oranžové až červené s černými skvrnami.Toto výstrařné zbarvení může být u odchovaných jedinců méně výrazné. Samička je většinou robustnější než sameček, který má delší a silnější přední končetiny.</p>

<div>Zdroj obrázku:

<a href="https://www.stoplusjednicka.cz/sites/default/files/obrazky/2020/09/02\_01\_kunka\_vychodni.jpg">www.stoplusjednicka.cz</>

</div>

<h1>Počasí</h1>

<p><span class="tucny-text">Počasí</span> je okamžitý stav v <span class="kurziva">ovzduší</span> na určitém místě. Je dáno stavem všech atmosférických jevů pozorovaných na určitém místě a v určitém krátkém časovém úseku nebo okamžiku. Tento stav se popisuje souborem hodnot <span class="kurziva">meteorologických prvků</span>, které byly naměřeny <span class="podtrzeny-text">meteorologickými přístroji</span> nebo zjištěny pozorovatelem (např. teplota vzduchu, stav oblačnosti, rychlost a směr větru, déšť, sněžení apod.)</p>

<p>Změny počasí jsou způsobeny především <span class="tucny-text">zemskou rotací</span>. Ohromné masy vzduchu a vody vlivem <span class="kurziva">zemské rotace</span> mají na severní polokouli tendenci pohybovat se ve směru hodinových ručiček. Na <span class="podtrzeny-text">jižní polokouli</span> se tyto masy pohybují opačným směrem.</p>

<p>Počasí je jedním jedinečným stavem atmosféry. Počasí je obvykle chápáno jako stav <span class="tucny-text">troposféry</span>, protože ta je člověku nejblíže a bezprostředně ho obklopuje. Obecný typ počasí v oblasti se nazývá podnebí nebo klima. Počasí se může měnit velmi rychle, změna klimatu je obvykle velmi pozvolná.</p>

<p>Kromě počasí se také používá, respektive používalo, termínu „povětří“ či „povětrnost“. Oba termíny jsou však <span class="tucny-text">zastaralé</span> a mohou mít také jiný význam.</p>

<p>Velká pozornost je věnována <span class="podtrzeny-text">předpovědi počasí</span>, protože počasí ovlivňuje všechny lidské činnosti. Počasím se zabývá <span class="kurziva">meteorologie</span>, respektive fyzika atmosféry.</p>

<div class="zdroj-textu">Zdroj textu: <a href="https://cs.wikipedia.org/wiki/Po%C4%8Das%C3%AD">Wikipedie</a>

</div>

## Bonusový úkol

V rámci úkolu se pokusíme nastylovat úryvek článku z cestovatelského blogu, doplníme fotky a odkazy. V zadání naleznete předpřipravené HTML, do kterého však bude tentokrát potřeba doplňovat věci dle zadání níže. CSS bude čistě na vás.

**Postup:**

1. Na celý dokument nastavíme v CSS pomocí tagu body rodinu písma **Helvetica** a druhá zástupná rodina písma bude **sans-serif**. Zároveň celému dokumentu nastavíme výšku řádku na hodnotu **1.5**.
2. Nadpisům <h1> a <h2> nastavíme pomocí desítkového zápisu libovolnou barvu.
3. K nadpisům <h2> v HTML doplníme třídu, které pak v CSS nastavíme transformaci textu na **velká písmena**.
4. Do HTML k obrázkům doplníme atributy src, tedy odkazy na obrázky. Můžeme použít fotky přímo z blogu z tohoto článku. Nezapomeneme na doplnění atributu alt. K obrázkům rovnou doplníme i třídu.
5. V CSS následně obrázkům pomocí zvolené třídy nastavíme **šířku** na **25 %**.
6. Najdeme si v textu v HTML názvy měst, obalíme je tagem <span> a doplníme k nim třídu .tucny-text. V CSS těmto třídám nastavíme tloušťku na hodnotu **bold**.
7. V textu se nachází několik údajů o počtu kilometrů. Najdeme tyto počty (např. 350 km), celý tento údaj obalíme tagem <span> doplníme k nim třídu .podtrzeny-text. V CSS těmto třídám nastavíme **podtržení**.
8. V úryvku článku se nachází dva odkazy <a>. V HTML k odkazům doplníme **atribut target** tak, aby se nám odkaz otevíral v novém okně. V CSS odkazům nastavíme **červenou** barvu (jakýkoliv odstín červené) pomocí **RGB zápisu** barev.
9. Pod oběma obrázky se nachází popisky k těmto obrázkům. Doplníme k nim v HTML názvy tříd a v CSS jim nastavíme styl písma na **kurzívu** a velikost písma na **14px**.
10. Na konec doplníme v HTML třídu ke -zdroji textu\* a této třídě v CSS nastavíme styl písma na **kurzívu**.

<h1>Po Jizerských horách na kolech</h1>

<p>Výlet do Jizerských hor jsme měli v plánu už loni na podzim, kdy jsme měli vyhlídnuté ubytování a naplánované atrakce. Nicméně jak jsme zmiňovali už v článku o Kutné Hoře, počasí tomu chtělo jinak. Přece jen pět propršených dní z pěti na horách nemá moc smysl.</p>

<p>K programu Jizerek jsme se tedy opět vrátili až letos, kdy jsme řešili, kam se podíváme během červencových svátků. Volno jsme si prodloužili i o zbylé tři dny v týdnu a našli jsme si ubytování přes <a href="https://www.booking.com/" >booking.com</a> ve vysoce hodnoceném penzionu Věra v >Tanvaldu. Vzhledem k tomu, že nás z Brna čekalo přes 350 km, zvolili jsme odjezd až v sobotních dopoledních hodinách. Cesta ubíhala bez problémů, provoz byl plynulý, jen s pár zastávkami na kafe a protažení. Posléze jsme zaparkovali přímo před penzionem, kde jsme narazili na pana domácího. Ten nás rovnou provedl objektem, objednali jsme si snídani na další den a začali jsme plánovat menší vycházku po okolí.</p>

<h2 >Singletrek pod Smrkem</h2>

<p>Z postele nás každodenně spolehlivě vytáhla brzká snídaně, která byla od 8 do 9. Sešli jsme pár schodů dolů do přízemí, kde na nás čekalo to, co jsme si den předtím objednali. Tedy míchaná vajíčka, toasty, párky nebo musli s jogurtem a k tomu dobrá káva. Díky brzkým snídaním jsme tedy i na výlety vyráželi na naše poměry relativně brzy. V neděli nás čekal přejezd do necelých 30 km vzdálených Lázních Libverda, odkud jsme měli naplánovaný start trailů. Cesta autem nebyla zas tak dlouhá, nicméně v Desné byla rozkopaná silnice a na semaforu jsme se dost zdrželi. Další zpomalení nás čekalo v úzkých serpentinách směrem na Bílý Potok, kde se nedalo jet rychleji než třicítkou. Zaparkovali jsme asi za 70 Kč na parkovišti u Obřího sudu, kde nás paní ujistila, že nám nosič na kola bude hlídat. Ještě jsme stihli nanuka a vyrazili jsme.</p>

<p>Lokalita čítá opravdu velké množství červených, černých i modrých stezek, které se táhnou až do sousedního Polska. Cesty vedou krásnými lesy, úseky nejsou nijak zvlášť technické, naopak jsou poměrně přírodní. V některých případech je to však i nevýhoda, pokud se zde projíždíte zrovna po deštích, kdy se na cestách tvoří menší i větší kaluže. Nemalá část cest vede také kopcovitým terénem, takže sjezdy nejsou úplně zadarmo. V rámci modré trasy jsme se dostali i na území Polska, kde jsme náhodou objevili bývalou astronomickou věž, která později sloužila také jako restaurace. Celkově hodnotíme traily na výbornou. Najeli jsme tu bezmála 45 km, ale rozhodně jsme neotestovali veškeré trasy.</p>

<img src="">

<div>Obrázek č. 1 - Lesní trail pod Smrkem</div>

<h2>Bikepark Tanvaldský Špičák</h2>

<p>Po jednodenní cyklistické přestávce, kterou jsme využili na návštěvu Liberce a jeho okolí, jsme se vydali opět na kola. Tentokrát jsme chtěli otestovat bikepark, který je umístěný přímo na sjezdovce na Tanvaldském Špičáku. Tentokrát to nebylo tak daleko, takže jsme zvolili variantu dojet na místo rovnou na kolech. Čekalo nás nějaké to převýšení, ale nahoru jsme se pak vyvezli lanovkou a ušetřili tak nohy na odpoledne. Dají se tu koupit buď samostatné jízdy nebo pak po třech a více. S koly nám na lanovce pomohli a my si užívali poměrně dlouhou zhruba 15 minutovou jízdu nahoru. Vzhledem k blížící se polední přestávce jsme se tu dlouho nerozhlíželi a jeli rovnou dolů. Trail byl divočejší než pod Smrkem, plný klopenek a skoků, ale v přiměřené rychlosti téměř bezproblémový. Při další jízdě už jsme tolik nespěchali, takže jsme si vychutnali krásné výhledy z kamenné vyhlídky poblíž vrcholu, kde je mimo jiné i stejnojmenná rozhledna. Po krátké pauze a svačině jsme se vydali na druhý trail, který byl velmi podobný tomu prvnímu.</p>

<p>Dnešní cyklovýlet však tímto nekončil. Pokračovali jsme dál dlouhým stoupáním na protrženou přehradu, což je místo, kde se před lety protrhla hráz a voda zatopila níže položenou vesničku Desná a přilehlé okolí. Nicméně nyní je to krásné slunné místo s pěknou vegetací a přilehlou hospůdkou vhodnou k občerstvení po vyčerpávajícím stoupání. Kopce zdaleka neskončily a mi si vyšlápli horskou stezkou další várku výškových metrů až na sklářskou osadu Jizerka (850 m. n. m.), která je nejvýše položenou obcí v ČR. Den se pomalu posunoval k podvečerním hodinám a my měli v plánu ještě zdolání vrcholu Hvězda s rozhlednou Štěpánka. Krásná kamenná rozhledna z roku 1892, která se tyčí nad obcí Příchovice nabízí úchvatný pohled na okolní vesničky i pohoří a rozhodně stojí za návštěvu. Ještě nám zbývalo posledních pár kilometrů, které už naštěstí byly jen z kopce dolů.</p>

<img src="">

<div>Obrázek č. 2 - Lanovka na Tanvaldský Špičák</div>

<div> Zdroj textu: <a href="https://www.cibulka-na-cestach.cz/po-jizerskych-horach-na-kolech/">cibulka-na-cestach.cz</a>

</div>

# Šířka, Výška, Okraj a Odsazení

## Šířka a výška

Obsah elementu má dva rozměry: **šířku** a **výšku**. Nicméně celkový prostor vyhraněný právě pro obsah nebývá pokaždé stejný. Záleží zde hlavně na typu HTML elementu. Ty máme buď blokové nebo řádkové. Každý z nich se pak chová na webu trochu jinak:

* **Blokové elementy** zabírají celou šířku dostupného prostoru. Výchozí výška je pak definovaná velikostí samotného obsahu.
* U **řádkových elementů** je výška i šířka vymezena pouze skutečným obsahem.

Lépe to lze vidět na ukázce. Blokové elementy jsou vyznačeny žlutou barvou a řádkové jsou modré. Šířka blokových elementů je vždy 100% (zabírají celou šířku prostoru). A to i v případě, že text uvnitř nezabírá celou šířku stránky.

A screenshot of a computer

Description automatically generated

### Úprava rozměrů v CSS

Rozměr elementu (resp. jeho obsahové části) lze přizpůsobit v CSS pomocí vlastností height a width. **Zde ale pozor! Nastavit vlastní rozměr dokážeme pouze u blokových elementů**. Řádkové elementy mají ze své podstaty vždy stejné rozměry, jako je jejich obsahu. Proto jakýkoliv pokus o změnu šířky či výšky se u řádkových elementů nijak neprojeví.

* height - Vlastnost určující výšku elementu.
* width - Vlastnost určující šířku elementu.

Hodnotu vlastnosti lze zapsat pomocí délkových jednotek, které jsou podporované v jazyce CSS. Např. px, em, rem, %, apod.

### Ukázka

Na následující ukázce byla omezena šířka odstavce na velikost 400px. Výchozí výška odstavce je beze změn, tudíž se přizpůsobí velikosti svého obsahu:

<p>Vzdělávání v 21. století musí být inovativní. V ENGETO Academy se budeš učit na unikátní výukové platformě a jen to, co skutečně potřebuješ – praktické znalosti aplikovatelné na projektech.</p>

p {

width: 400px;

background-color: #05ffb0;

}

Druhá ukázka obsahuje element, kde byla šířka omezena na 50% (tj. na půlku dostupného prostoru) a výška je nastavena na 300px:

p {

width: 50%;

height: 300px;

background-color: #05ffb0;

}

## Vnější a vnitřní okraje

Každý element může mít kolem svého obsahu nadefinované okraje. U webových prvků rozlišujeme dva druhy okrajů: vnější a vnitřní.

### Vnitřní okraj

**Vnitřní okraj** nazýváme anglickým výrazem **padding** a vyskytuje se mezi samotným obsahem a případným rámečkem. Padding je vlastnost, která upravuje prostor mezi obsahem elementu a jeho rámečkem/okrajem. Umožňuje přidat prostor uvnitř elementu a lze ji nastavit na různé hodnoty pro každou stranu elementu.

### Vnější okraj

**Vnější okraj** nese anglický název **margin** a vyskytuje se na vnější straně elementu, kolem případného rámečku. Ovlivňuje prostor mezi elementem a jeho sousedními elementy. Umožňuje přidat prostor kolem vnější strany elementu a lze jej nastavit na různé hodnoty pro každou stranu daného elementu.

A yellow rectangle with blue text

Description automatically generated

Představte si rám obrazu na stěně - obraz v rámu je od vlastního rámu oddělen bílým prostorem. Tento bílý prostor je jako padding v HTML a CSS. Poskytuje určitý prostor mezi obsahem a rámečkem elementu. Podobně pokud máte na zdi více obrázků, můžete je od sebe oddělit tím, že mezi nimi necháte nějaký prostor. To je jako margin v HTML a CSS. Poskytuje určitý prostor mezi rámečkem prvku a okolními elementy. Úpravou velikosti paddingu a marginu můžete kontrolovat rozestupy mezi elementy a vytvořit vizuálně příjemnější rozvržení.

V CSS vytvoříme okraje pomocí těchto vlastností:

* margin - CSS vlastnost určující velikost vnějšího okraje.
* padding - CSS vlastnost určující velikost vnitřního okraje.

Hodnotu obou vlastností lze zapsat pomocí délkových jednotek, které jsou podporované v jazyce CSS. Např. px, em, rem, %, apod.

Pokud bude mít element barevné pozadí, bude vnitřní okraj ještě součástí obarvené části. Pro vnější okraj to už neplatí, ten se už nachází z venkovní strany elementu.

<div class="item-1">Lorem ipsum dolor sit amet</div>

<div class="item-2">Lorem ipsum dolor sit amet</div>

<div class="item-3">Lorem ipsum dolor sit amet</div>

<div class="item-4">Lorem ipsum dolor sit amet</div>

.item-1 {

border: 1px solid red;

padding: 1rem;

margin: 1rem;

background-color: #FDFD96;

}

.item-2 {

border: 1px solid red;

padding: 2rem;

margin: 2rem;

background-color: #FDFD96;

}

.item-3 {

border: 1px solid red;

padding: 1rem;

margin: 1rem;

background-color: #FDFD96;

}

.item-4 {

border: 1px solid red;

padding: 1rem;

margin: 1rem;

background-color: #FDFD96;

}

#### Jedna hodnota

Pokud u vlastnosti margin nebo padding uvedeme jednu jedinou hodnotu, nastavíme tak **stejný okraj kolem celého obsahu**.

#### Více uvedených hodnot

Pokud u vlastnosti margin nebo padding uvedeme více hodnot za sebou (oddělených mezerou), nastavíme tím okraj **postupně pro horní, pravou, dolní a levou stranu elementu stranu elementu** (začínáme od horní strany a postupujeme po směru hodinových ručiček).

Hodnota padding: 20px 10px 30px 40px; tak znamená, že horní vnitřní okraj bude mít velikost 20px, pravý vnitřní okraj bude 10px, dolní okraj 30px a levý okraj 40px. Stejné pravidlo by platilo i u vnějšího okraje margin.

.box {

padding: 20px 10px 30px 40px;

margin: 10px 0 20px 0;

width: 300px;

border: 1px solid blue;

background-color: #05ffb0;

}

#### Okraj konkrétní strany

Okraj nemusíme nutně nastavovat pro všechny strany elementu. Definici lze provést i pro jednu konkrétní stranu. Toho se dá docílit přidáním anglického slova left (nalevo), right (napravo), top (nahoře) nebo bottom (dole) k názvu CSS vlastnosti, kterou hodláme měnit.

Například pomocí vlastnosti padding-left nastavíme levý vnitřní okraj. margin-bottom nám poslouží k nastavení dolního vnějšího okraje

### Slučování vnějších okrajů

Vlastnost margin má jednu zvláštní vlastnost. Když dáme vedle sebe dva elementy a každý z nich bude mít nějaký vnější okraj, **dojde ke sloučení těchto okrajů**.

Pojďme si to ukázat na jednoduchém přikladě. Na následující ukázce máme 2 elementy. První z nich má dolní vnější okraj o velikosti 20px a druhý element má horní vnější okraj 40px. Tyto elementy na sebe "narazí" a jejich vnější okraje se sloučí. Mezera mezi nimi bude nakonec 40px (vezmeme okraje na stranách, kterými se elementy vzájemně dotýkají a použijeme to větší číslo).

### Procvičení

V následujícím úkolu si procvičíme nastavení vnitřního a vnějšího odsazení a také si vyzkoušíme nastavit šířku elementu.

1. Elementu s třídou **jaro** nastavte šířku na 300px, barvu pozadí nastavte dle libosti pomocí **slovního označení** barvy.
2. Elementu s třídou **leto** nastavte šířku na 150px, barvu pozadí nastavte dle libosti pomocí **hex** zápisu.
3. Elementu s třídou **podzim** nastavte šířku na 400px, barvu pozadí nastavte dle libosti.
4. Elementu s třídou **zima** nastavte šířku na 200px, barvu pozadí nastavte dle libosti pomocí **RGB** zápisu.
5. Elementy jaro a podzim budou mít **vnitřní odsazení** 20px a **vnější odsazení** 30px.
6. Elementy leto a zima budou mít **vnitřní odsazení** 5px a **vnější odsazení** 10px.

### Inline elementy

Inline elementy mají jenom vodorovné vnější (margin) a vnitřní (padding) okraje, které lze nastavit pomocí vlastností margin-left, margin-right, padding-left a padding-right.

A screenshot of a computer

Description automatically generated

Pozor, inline elementy nemají svislé okraje, to znamená, že nemůžeme nastavit pomocí vlastností margin nebo padding prostor nad nebo pod elementem. Vlastnosti margin-top, margin-bottom, padding-top a padding-bottom nebudou mít žáden vplyv na inline element. To znamená, že pokud na inline element nastavíme svislé okraje, neovlivní to okolní text a prohlížeč je bude ignorovat a nezobrazí je.

Pokud ale přidáme inline elementu rámeček nebo barvu pozadí, a nastavíme vertikální okraje, ovlivní to rámeček a barvu ale ne samotný obsah elementu a okolní text.

A screenshot of a computer

Description automatically generated

Pokud chceme přidat svislý prostor kolem inline elementu, můžete použít vlastnost line-height. Například můžeme zvýšit výšku řádku, aby jsme vytvořili více vertikálního prostoru kolem inline elementu.

A screenshot of a computer

Description automatically generated

<https://codepen.io/yadahi/pen/GRYQKpY>

### Shrnutí

* **Blokové elementy**: vnější a vnitřní okraje lze použít na všechny čtyři strany elementu, a to jak horizontálně, tak vertikálně. Lze je použít k vytvoření prostoru kolem a uvnitř elementu.
* **Inline elementy**: vnější a vnitřní okraje lze nastavit pouze horizontálně, a to na levou a pravou stranu elementu. Okraje se nepřidávají nad ani pod element a neovlivňují rozvržení obsahu a okolních elementu.

Obecně se blokové elementy používají k vytváření větších struktur na webové stránce, jako jsou záhlaví, sekce a sloupce, zatímco inline elementy se používají ke úpravě menších částí obsahu v rámci těchto struktur, jako jsou odkazy, zvýraznění a obrázky.

# RÁMEČEK

## Rámeček

Rámeček si lze představit jako čáru, kterou dokážeme zobrazit kolem samotného elementu. Vzhled této čáry nám na webu určují tři vlastnosti:

#### 1. Tloušťka čáry

* Linka rámečku může být různé tloušťky.
* Zápis v CSS: border-width
* Tloušťku čáry lze vyjádřit pomocí dostupných délkových jednotek (px, pt, em, ...). Druhou možností je použití jednoho z klíčových slov thin, medium nebo thick. Tyto klíčová slova pak představují tři různé stupně tloušťky čáry.

#### 2. Styl čáry

* Čára rámečku nemusí být nutně jen plná. Na webu dokážeme zobrazit rámeček například i v tečkovaném, čárkovaném nebo dokonce i v 3D stylu.
* Zápis v CSS: border-style
* Ukázka dostupných stylů:

#### 3. Barva čáry

* Barvu linky lze přizpůsobit vlastním představám.
* Zápis v CSS: border-color
* Zápis barvy čáry provedeme podobně jako u barvy textu či pozadí. Jako hodnotu lze použít [RGB zápis barvy](https://engeto.com/cs/kurz/tvorba-webu-1-html-css/studium/wHRzPaZPR3i_msgLV9mguw/barvy-a-delkove-jednotky/rgb-zapis-barvy/hexadecimalni-rgb-zapis), popřípadě můžeme uvést jednu z [pojmenovaných webových barev](https://engeto.com/cs/kurz/tvorba-webu-1-html-css/studium/r8ci4O16QBupoTtj2_pPSQ/barvy-a-delkove-jednotky/pojmenovane-barvy).

### Ukázka

Na následující ukázce je definice rámečku o šířce 2 pixely. Barva linky je zde nastavena na #e63144.

A screenshot of a computer

Description automatically generated

## Rámeček podél specifické strany

Každý HTML element má tvar boxíku. Když potom definujeme rámeček, vzniká tak linka kolem dokola celého objektu. Na webu ale není podmínkou, aby rámeček šel kolem všech čtyř stran. Vybrat se dá i jedna konkrétní strana. Toho docílíme přidáním anglického slova left (nalevo), right (napravo), top (nahoře) nebo bottom (dole) k názvu CSS vlastnosti, kterou hodláme měnit.

**Tloušťka rámečku:**   
border-left-width (tloušťka rámečku na levé straně), border-right-width (tloušťka rámečku na pravé straně), border-top-width (tloušťka horního rámečku) nebo border-bottom-width (tloušťka dolního rámečku).

**Styl rámečku:**   
border-left-style (styl rámečku na levé straně), border-right-style (styl rámečku na pravé straně), border-top-style (styl horního rámečku) nebo border-bottom-style (styl dolního rámečku).

**Barva rámečku:**   
border-left-color (barva rámečku na levé straně), border-right-color (barva rámečku na pravé straně), border-top-color (barva horního rámečku) nebo border-bottom-color (barva

### Ukázka

Takto může vypadat situace, kdy se rozhodneme definovat pouze dolní stranu rámečku:

.barevny-ramecek {

border-bottom-width: 2px;

border-bottom-style: dashed;

border-bottom-color: #e63144;

border-top-width: 1px;

border-top-style: dashed;

border-top-color: blue;

border-left-width: 4px;

border-left-style: solid;

border-left-color: #cccccc;

border-right-width: 4px;

border-right-style: solid;

border-right-color: #999;

}

## Zakulacené rohy

I přesto, že rámečky kolem elementů mají obdélníkový tvar, tak existuje v CSS vlastnost, díky níž dokážeme rohy rámečků zakulatit. Otevírají se nám tím nové možnosti stylování.

Zakulacené rohy vytvoříme v CSS pomocí vlastnosti border-radius. Hodnota této vlastnosti pak udává poloměr zakulacení jednotlivých rohů. Hodnotu lze zapsat pomocí délkových jednotek, které jsou podporované v jazyce CSS. Např. px, %, apod.

#### Jedna hodnota

Element na následující ukázce má nastavené zakulacení svých rohů o poloměru 5 pixelů:

p {

width: 300px;

background-color: #05ffb0;

border-radius: 5px;

}

#### Více uvedených hodnot

Zakulacovat můžeme i pro každý růžek zvlášť. Je však nutné použít zápis, kdy uvedeme míru zakulacení pro jednotlivé rohy. Postupně tak uvedeme zakulacení pro levý horní roh, pravý horní roh, pravý dolní roh a levý dolní roh (začínáme vlevo nahoře a postupujeme po směru hodinových ručiček). Jednotlivé hodnoty pak od sebe oddělujeme mezerou.

p {

width: 300px;

background-color: #05ffb0;

border-radius: 10px 5px 0 40px;

}

#### Zakulacení konkrétního rohu

Zakulacení nemusíme nutně nastavovat všem rohům najednou. Pro nastavení zakulacení jednoho konkrétního rohu slouží v CSS tyto vlastnosti:

* border-top-left-radius - zakulacení levého horního rohu
* border-top-right-radius - zakulacení pravého horního rohu
* border-bottom-right-radius - zakulacení pravého dolního rohu
* border-bottom-left-radius - zakulacení levého dolního rohu

### Kruh

Pomocí zakulacených rohů dokážeme docílit i dokonale kulatého vzhledu elementu. Trik spočívá v tom, že míru zakulacení vyjádříme v procentech. Hodnotu border-radius pak nastavíme o polovinu menší, než jsou rozměry celého elementu - tj. nastavíme hodnotu na 50%:

.kruh {

width: 300px;

height: 300px;

background-color: #05ffb0;

border-radius: 50%;

}

## Zkrácený zápis

Jak již bylo zmíněno, definice rámečku se provádí pomocí vlastností border-width, border-style a border-color. Tyto vlastnosti se vyskytují většinou společně. V CSS proto můžeme uvádět i tzv. **zkrácený zápis**, kdy do jedné definice zahrneme naráz všechny tři vlastnosti rámečku.

Zkrácený zápis vyjádříme v CSS pomocí vlastnosti border. Hodnota se pak obsahuje **tloušťku, styl a barvu rámečku**, kde jednotlivé hodnoty jsou **zapsané za sebou a oddělené mezerou**. Na pořadí hodnot nezáleží.

Pro představu, namísto tří deklarací

border-width: 2px;

border-style: dashed;

border-color: #e63144;

lze zkráceně past

border: 2px dashed #e63144;

# Box model

## Box model

Aby bylo možné v budoucnu přistoupit k rozmisťování jednotlivých prvků po stránce, je nutné prvně pochopit principy toho, jak dnes na webu definujeme rozměry samotného HTML elementu. Jakýkoliv element je při zobrazení interpretován jako boxík. Nejlépe je to viditelné v momentě, kdy prvku přiřadíme pomocí CSS nějaké barevné pozadí. U blokových HTML elementů zabírá takový boxík celou šířku dostupného prostoru, u řádkových HTML elementů je šířka definovaná pouze samotným obsahem.

Každý obsah na webu se tedy vyskytuje uvnitř nějakého boxíku. Celý model, o kterém se budeme v této lekci bavit, dostal právě díky tomu název **box model** (používáme anglický výraz, který nemá ryze české synonymum).

Pojmem **box model** nazýváme soubor několika vlastností, které mají vliv na celkový rozměr HTML elementu na webu. Jedná se o následující vlastnosti:

1. **Šířka a výška** - Tento rozměr definuje prostor pro samotný obsah uvnitř elementu, například text nebo obrázek.
2. **Vnitřní okraj (padding)** - Představuje dodatečný prostor mezi samotným obsahem a rámečkem.
3. **Rámeček** - Rámeček (pokud je definovaný) má svou tloušťku čáry a styl. Samotný obsah elementu a jeho vnitřní okraje se nachází uvnitř rámečku.
4. **Vnější okraj (margin)** - Množství vnějšího prostoru kolem samotného rámečku.

Všechny prvky HTML si lze představit jako obdélníkové boxy s těmito čtyřmi částmi a každou z nich lze stylovat pomocí CSS a řídit tak vzhled a rozložení prvku. Pochopení box modelu je nezbytné pro vytváření efektivních a responzivních webových návrhů.

Znázornění jednotlivých vlastností box modelu můžeme vidět na následujícím diagramu:

A yellow rectangular object with black text

Description automatically generated

### Ukázka

V následující ukázce jsme nastavili prvnímu divu **vnitřní odsazení** na 30px a druhému divu **vnější odsazení** na 30px. Všimněte si rozdílů.

A screenshot of a computer

Description automatically generated

### Skutečná šířka a výška

Teď už víme, že existuje nějaký box model a jaké vlastnosti ho reprezentují. K čemu je to ale dobré? My si díky box modelu dokážeme spočítat skutečný rozměr elementu. Ten netvoří pouze prostor vyhraněný pro samotný obsah. Rozměr elementu ovlivňují také okraje a tloušťka rámečku. Celkový rozměr elementu spočítáme takto:

**Celková šířka elementu:**   
levý vnější okraj + tloušťka levého rámečku + levý vnitřní okraj + šířka obsahové části + pravý vnitřní okraj + tloušťka pravého rámečku + pravý vnější okraj

**Celková výška elementu**   
horní vnější okraj + tloušťka horního rámečku + horní vnitřní okraj + výška obsahové části + dolní vnitřní okraj + tloušťka dolního rámečku + dolní vnější okraj

A diagram of a number

Description automatically generated with medium confidence

Grafika (se kterou se můžeme pravidelně setkat v **DevTools** v prohlížeči) nám ukazuje informace o velikostech. Modrý obdélník zobrazuje velikost daného elementu (šířka x výška). Následuje vnitřní odsazení (padding) 20px v zeleném obdélníku. Dále je tu také rámeček (border), který má 2px a nakonec vnější odsazení (margin), které má 30px.

**Celková šířka** elementu je tedy 30 + 2 + 20 + 256 + 20 + 2 + 30 = 360px.

**Celková výška** elementu je 30 + 2 + 20 + 156 + 20 + 2 + 30 = 260px.

V dalších kapitolách této lekce se podíváme podrobněji na jednotlivé vlastnosti box modelu a naučíme se je přizpůsobovat vlastním potřebám.

V CSS se **box model** nastavuje pomocí vlastnosti box-sizing. Vlastnost box-sizing umožňuje určit, zda má šířka a výška prvku zahrnovat pouze jeho obsah, nebo i **obsah, výplň (padding) a ohraničení (border)**.

selector {

box-sizing: content-box; /\* default value \*/

}

Výchozí hodnota vlastnosti box-sizing v CSS je content-box. To znamená, že **šířka a výška prvku jsou vypočteny pouze na základě obsahové oblasti a veškerý vnitřní okraj a vnější okraj jsou k této šířce a výšce připočteny**.

Existuje však další hodnota pro vlastnost box-sizing, která se nazývá border-box.

selector {

box-sizing: border-box;

}

Pokud je hodnota box-sizing nastavena na border-box, je **šířka a výška prvku vypočítána na základě obsahové oblasti, vnitřního okraje a vnějšího okraje dohromady**. To znamená, že **vnitřní okraj a vnější okraj jsou zahrnuty do šířky a výšky prvku**.

Použití border-box může být užitečné při práci s responzivním designem, protože umožňuje přesnější a konzistentní velikost prvků bez ohledu na jejich vnitřní a vnější okraj.

Porovnáni box-content a border-box na [příkladu](https://miro.medium.com/v2/resize:fit:3450/1*WUkflCFQ1R238tbX67Ok4A.jpeg).

# POZICOVÁNÍ

## Pozicování prvků na stránce

Další důležitou součástí CSS je **pozicování prvků** na stránce. Prvky na stránce můžeme **pozicovat, posouvat** nebo **umisťovat na konkrétní souřadnice** na stránce. Pro pozicování prvků je nezbytné nastavení vlastnosti position na daný prvek.

Zásadní pro pochopení pozicování elementu je pochopení normálního toku dokumentu v HTML a CSS. Pozicování je totiž v podstatě manipulace s výchozím umístěním elementu v normálním toku.

## Normálni tok dokumentu

V HTML je normální tok dokumentu daný výchozím umístněním a rozvržením elementu na webové stránce bez použití jakýchkoliv dalších vlastnosti pro nastavení pozice.

Jinými slovy, normální tok dokumentu je pořadí, v jakém jsou prvky na stránce standardně umístěny bez jakéhokoli pozicování. Elementy ve výchozím nastavení jsou seřazeny za sebou v pořadí, v jakém se objevují v kódu HTML.

Každý blokový element zabírá celou šířku svého kontejneru a jeho výška je určena jeho obsahem, pokud není nastaveno jinak. Inline elementy se rozmístí vedle sebe vodorovně a v případě nedostatku místa se zalamují na další řádek. Pozice každého elementu je určena jeho vztahem k elementům, které jsou v dokumentu před ním a za ním.

## Position: static;

Vlastnost position s hodnotou static je výchozím nastavení prvků. Není třeba jej nastavovat, prvky mají tuto hodnotu nastavenou automaticky, i když se u prvku nezobrazuje. To si můžeme ověřit i pomocí DevTools v prohlížeči.

## Position: relative;

Hodnota relative již umožňuje s prvky "hýbat". Když na elementu nastavíme vlastnost position: relative, řekneme prohlížeči, aby element pozicoval **vzhledem k jeho původní pozici** v normálním toku dokumentu. To znamená, že element je stále součástí toku dokumentu a ostatní prvky budou pozicovány tak, jako by byl na své původní pozici.

Nicméně hodnota relative sama o sobě prvek neposune. Vlastnosti top, right, bottom a left pak lze použít k posunu prvku vzhledem k jeho původní pozici. K těmto vlastnostem pak přiřadíme číselný údaj např. v **px**.

* top
* right
* bottom
* left

### Příklad

V uvedeném příkladu jsme nastavili vlastnost position na hodnotu relative. Prvku jsme poté nastavili, že se **vysune o 30px shora** ze své původní pozice a o **30px zleva** ze své původní pozice.

div {

position: relative;

top: 30px;

left: 30px;

}

V Codepenu můžeme vidět dva rámečky. Čárkovaný rámeček pro lepší představu znázorňuje výchozí pozici našeho <div> elementu s třídou **box**. Druhý <div> s třídou **box** již znázorňuje vysunutí o zmíněných 30 px shora a 30px zleva.

<https://codepen.io/engeto/pen/OJxgqNb>

Možná si v tuto chvíli nedokážeme představit, proč by se nám tato vlastnost mohla hodit. Během této lekce i po čas celého kurzu budeme pracovat s více úkoly a pozicování si pořádně procvičíme :).

## Position: absolute;

Další hodnotou vlastnosti position je absolute. Při absolutním pozicování je **element vyjmut z normálního toku** dokumentu.

Představme si to tak, že nyní existuje v další vrstvě nad zbytkem stránky. Na stránce přestane existovat. **Na jeho místo se tedy může posunout obsah dalšího elementu**. Pomocí absolutního pozicování **můžeme prvek napozicovat vůči celé stránce** nebo **vůči rodičovského elementu**. Nejprve si ukážeme, jak nastavíme pozicování vůči stránce.

Nastavení souřadnice je pak stejné, jako u předchozí hodnoty. Zápis tedy bude vypadat takto:

div {

position: absolute;

top: 30px;

right: 50px;

}

### Ukázka

V dané ukázce vidíme, že jsme na <div> nastavili absolutní pozicování. Odshora jsme nastavili osazení 30px, zprava 50px. Náš <div> se napozicoval s uvedenými údaji vůči celé stránce, která v Codepenu představuje bílé místo pod černými boxy HTML a CSS. Znamená to, že CSS vypočítalo, aby se <div> odsadil od horního okraje 30px a od pravého okraje 50px; O **responzivním designu** budeme mluvit až v další lekci, nicméně si už teď můžeme ukázat, že pokud prvku nastavíme absolutní pozicování, bude tato **pozice zůstávat stejná** i v případě, že se bude zmenšovat nebo zvětšovat šířka okna.

<https://codepen.io/engeto/pen/rNGwRpP>

### Pozicování vůči rodičovskému elementu

Pozicování vůči rodičovskému elementu využijeme možná častěji než pozicování vůči celé stránce. Často totiž potřebujeme umístit např. tlačítko k pravému dolnímu okraji formuláře nebo obrázek do levého horního okraje rodičovského elementu.

V takovém případě je zapotřebí dvojího nastavení. Nejprve musíme vybrat rodičovský prvek, vůči kterému chceme náš prvek pozicovat. Tomuto rodičovskému prvku musíme nastavit vlastnost position na hodnotu relative. Tím prohlížeči řekneme, že k tomuto prvku chceme nějaký další prvek napozicovat. Danému prvku poté nastavíme position na hodnotu absolute a dál postupujeme stejně. Tedy nastavíme souřadnice (top, right, bottom nebo left).

V naší ukázce máme nejprve uvedený rodičovský div, uvnitř kterého je umístěné tlačítko. V prvním případě nemáme nastavené žádné pozicování a vidíme, kam se tlačítko automaticky umístilo.

V druhém případě již vidíme, že na rodičovský <div> je nastavená vlastnost position: relative; a na tlačítko vlastnost position: absolute;. Pomocí hodnot bottom: 15px; a right: 15px; jsme tlačítko umístili 15px; od spodního okraje a 15px od pravého okraje rodičovského elementu.

<https://codepen.io/engeto/pen/rNGwRQg>

### Pozor na častou chybu

Častou chybou při absolutním pozicování elementu vůči jinému elementu je, že zapomeneme nastavit pozicování rodičovského elementu na position: relative. Pokud nezadáme pozici rodičovského elementu, bude podřízený element pozicován absolutně vzhledem k nejbližšímu předkovi, který má nastavenou pozici. To může často vést k neočekávaným výsledkům a element se nemusí zobrazit tam, kde jsme ho zamýšleli umístit. Proto je důležité vždy nastavit pozici rodičovského elementu na position: relative, aby bylo zajištěno správné umístění podřízeného elementu.

Představme si, že máme rodičovský element div s nějakým textem uvnitř a chceme umístit potomka absolutně do pravého horního rohu rodičovského elementu. Zápis může vypadat následovně:

A screenshot of a computer

Description automatically generated

Všimněme si, že jsme u rodičovského elementu správně nastavili position: relative, což nám umožňuje umístit potomka absolutně vzhledem k rodiči. Při zobrazení stránky v prohlížeči vidíme potomka umístěného v pravém horním rohu rodičovského elementu, jak bylo zamýšleno.

Nyní, odstraníme position: relative z rodičovského elementu a podívejme se co se stane s potomkem.

A screenshot of a computer

Description automatically generated

Při zobrazení stránky v prohlížeči vidíme, že potomek je umístěn absolutně vzhledem k dalšímu rodičovskému elementu v toku dokumentu, kterým je v tomto případě element body. Je to proto, že když má rodičovský element position: static (což je výchozí nastavení), jsou jeho podřízené elementy s position: absolute pozicovány vzhledem k dalšímu rodičovskému elementu v řetězci, který má nestatickou pozici.

Jde o častou chybu při pozicování elementů, která může způsobit neočekávané chování rozvržení, pokud si nedáme pozor. Proto je důležité při absolutním pozicování podřízeného elementu vždy nastavit position: relative na rodičovském elementu.

Zde je příklad, který ukazuje, jak může zapomenutí nastavení pozice na relativní u bezprostředního rodiče způsobit, že absolutně pozicovaný element bude pozicován vzhledem k předkovi vyšší úrovně.

<https://codepen.io/yadahi/pen/GRYQvNE>

V tomto příkladu máme element .wrapper, který slouží jako relativně pozicovaný element pro absolutně pozicovaný element .child. Zapomněli jsme však nastavit pozici bezprostředního rodiče .parent na relativní. V důsledku toho bude element .child pozicován relativně vůči elementu .wrapper namísto elementu .parent. To může způsobit neočekávané problémy s pozicováním a rozvržením. Abychom se tomuto problému vyhnuli, měli bychom se vždy ujistit, že bezprostřední rodičovský element absolutně pozicovaného elementu má vlastnost position nastavenou na relative.

## Position: fixed;

Poslední hodnotou vlastnosti **position**, kterou si v rámci kurzu představíme, je hodnota fixed. Prvek pomocí fixního pozicování pozicujeme fixně vůči oknu prohlížeče. To znamená, že prvky, které mají nastavený position: fixed; zůstávají na místě, zatímco scrollujeme po stránce. To znamená, že je vyjmut z normálního toku dokumentu a neovlivňuje rozložení ostatních elementů.

Nejčastějšími příklady využití fixního pozicování je **hlavička stránky s navigací** nebo **tlačítko na vyscrollování nahoru**. Nastavení je opět velmi podobné jako v předchozích příkladech:

div {

position: fixed;

right: 30px;

bottom: 30px;

}

Vzhledem k tomu, že v Codepenu **nevidíme kýžený efekt**, můžeme se na použití fixního pozicování **tlačítka scroll up** podívat např. na stránkách [Webstudia](https://webstudio.team/). **Fixně nastavené menu** můžeme zkontrolovat přímo na stránkách [Engeta](https://engeto.cz/).

<https://codepen.io/engeto/pen/qBPjwVm>

## Externí materiál k pozicování

Pozicování prvků může být zvlášť ze začátku náročnější na pochopení. Kromě příkladů v této lekci a několika úkolů na procvičení, doporučujeme ještě externí zdroj, který velmi detailně rozebírá jednotlivé hodnoty vlastnosti position a používá názorné ukázky.

**Odkaz na materiál**: [Freecodecamp.org](https://www.freecodecamp.org/news/css-position-property-explained/)

# VIDITELNOST ELEMENTŮ

## Skrývání elementů

Při vývoji webových stránek je běžné, že chceme před uživatelem skrýt určité elementy. Důvody mohou být různé - element třeba není pro aktuální stránku relevantní nebo obsahuje informace, které by se měly zobrazit pouze za určitých podmínek. Skrývání pomocí CSS nachází svá uplatnění u dynamických elementů stránky, které bude možné zobrazit například kliknutím na vybrané tlačítko nebo odkaz. Ať už je důvod jakýkoli, v HTML a CSS existuje několik způsobů, jak prvky skrýt.

1. display: none - Tím se tento element zcela skryje a odstraní z toku stránky. Běžně se používá ke skrytí elementů, které by se neměly zobrazovat vůbec.
2. visibility: hidden - Tím se element skryje, ale stále zabírá místo v layoutu. Element není viditelný, ale stále ovlivňuje rozvržení ostatních elementů v jeho okolí.
3. opacity: 0 - Tím se element stane zcela průhledným. Element stále zabírá místo v layoutu a jeho obsah je stále přítomen, ale není viditelný.

## display: none

Jedním z nejběžnějších způsobů skrytí elementu v HTML a CSS je nastavení jeho vlastnosti display na hodnotu none. Tím se element zcela odstraní z toku stránky a stane se pro uživatele neviditelným a nepřístupným. Elementy, které jsou skryty pomocí display: none, nezabírají na stránce žádné místo a nelze s nimi nijak interagovat.

selector {

display: none;

}

### Ukázka

Ukázku skrytého prvku můžeme vidět na následujícím příkladu, kde je pomocí display: none; schován celý element <span class="radkovy-element">řádkový HTML element.</span>:

A screenshot of a computer

Description automatically generated

## visibility: hidden

Dalším způsobem skrytí elementů v HTML a CSS je nastavení jejich vlastnosti visibility na hodnotu hidden. Tato vlastnost skryje element ze zobrazení, ale na rozdíl od vlastnosti display: none stále zabírá místo na stránce. Elementy, které jsou skryté pomocí vlastnosti visibility: hidden, jsou neviditelné, ale stále ovlivňují rozvržení stránky.

A screenshot of a computer

Description automatically generated

<https://codepen.io/yadahi/pen/VwEQLoR>

## Průhlednost

Pro elementy na webu je možné definovat jejich průhlednost. Takový element se vykreslí částečně průhledný a to včetně svého pozadí i obsahu. V CSS nastavíme průhlednost pomocí vlastnosti opacity. Hodnotu uvádíme jako desetinné číslo v rozmezí od nuly do jedné:

* 0 - Element **je zcela průhledný**. Takový HTML prvek není sice vidět, ale na stránce stále zabírá své místo.
* 1 - Element **není vůbec průhledný**. Jedná se o výchozí hodnotu. Viditelnost prvku je tak stejná, jako kdyby se nic nenastavilo.
* 0.5 - Průhlednost elementu je nastavena na 50%.

### Ukázka

Na následující ukázce jsou tři elementy stejné barvy umístěné pod sebou. První z nich má nastavenou 100% průhlednost. Všimněte si, že i když není vidět, tak stále zabírá prostor na webu. Druhý prvek má nulovou průhlednost (opacity: 1;) a je tedy zobrazen zcela beze změny. Třetí prvek je průhledný ze 70%.

<https://codepen.io/engeto/pen/Jeqbxr>

## Shrnutí

Stručně řečeno, display: none zcela odstraní element ze stránky, visibility: hidden skryje element a zároveň mu umožní zabrat místo a opacity: 0 zprůhlední element tak, že je stále viditelný, ale zdá se být neviditelný.

V následující ukázce máme tři elementy, z kterých má každý nastavený jiný způsob skrytí. Element můžete skrýt nebo zobrazit kliknutím na tlačítko. Zaměřte se na to, jak se jednotlivé elementy chovají a jak se mění rozvržení a flow elementů.

<https://codepen.io/yadahi/pen/MWPQaYE>

## Průhledná barva

Existuje ještě jeden způsob nastavení průhlednosti, který však neovlivňuje viditelnost celého HTML elementu. U všech CSS vlastností nastavujících barvu (tj. color, background-color, border-color, ...) můžeme jako hodnotu uvést i tzv. **poloprůhlednou barvu**. Toho se dá využít například v situaci, kdy se elementu rozhodneme přiřadit poloprůhledné pozadí, ale zbytek obsahu plánujeme ponechat viditelný.

V tomto kurzu jsme se učili, že pro zobrazení barev na obrazovce používáme barevného modelu označovaného jako RGB. Každá barva pracuje se třemi barevnými složkami a výslednou barvu získáme složením těchto složek. V CSS takovou barvu vyjadřujeme zápisem:

**rgb(RED, GREEN, BLUE)**

Poloprůhledné barvy existují v rámci **modelu RGBA**. Ten vychází z klasického barevného modelu RGB, který je navíc rozšířen o tzv. alfakanál (informaci o průhlednosti barvy). Zápis RGBA barvy provedeme v CSS pomocí následujícího zápisu:

**rgba(RED, GREEN, BLUE, ALFA)**

Jednotlivé barevné složky (RED, GREEN a BLUE) můžeme zapsat buď procenty (0 až 100%) a nebo číslem od 0 do 255. Tento způsob zápisu byl podrobné rozebrán v lekci týkající se [zápisu barev](https://engeto.com/cs/kurz/tvorba-webu-1-html-css/studium/X26dPCxbQeiXJDhle1XK4A/barvy-a-delkove-jednotky/vznik-barev). Čtvrté číslo je alfakanál a jeho hodnota se zapisuje jako desetinné číslo v rozmezí od nuly do jedné (podobně jako tomu je u opacity).

## Průhledná barva

Existuje ještě jeden způsob nastavení průhlednosti, který však neovlivňuje viditelnost celého HTML elementu. U všech CSS vlastností nastavujících barvu (tj. color, background-color, border-color, ...) můžeme jako hodnotu uvést i tzv. **poloprůhlednou barvu**. Toho se dá využít například v situaci, kdy se elementu rozhodneme přiřadit poloprůhledné pozadí, ale zbytek obsahu plánujeme ponechat viditelný.

V tomto kurzu jsme se učili, že pro zobrazení barev na obrazovce používáme barevného modelu označovaného jako RGB. Každá barva pracuje se třemi barevnými složkami a výslednou barvu získáme složením těchto složek. V CSS takovou barvu vyjadřujeme zápisem:

**rgb(RED, GREEN, BLUE)**

Poloprůhledné barvy existují v rámci **modelu RGBA**. Ten vychází z klasického barevného modelu RGB, který je navíc rozšířen o tzv. alfakanál (informaci o průhlednosti barvy). Zápis RGBA barvy provedeme v CSS pomocí následujícího zápisu:

**rgba(RED, GREEN, BLUE, ALFA)**

Jednotlivé barevné složky (RED, GREEN a BLUE) můžeme zapsat buď procenty (0 až 100%) a nebo číslem od 0 do 255. Tento způsob zápisu byl podrobné rozebrán v lekci týkající se [zápisu barev](https://engeto.com/cs/kurz/tvorba-webu-1-html-css/studium/X26dPCxbQeiXJDhle1XK4A/barvy-a-delkove-jednotky/vznik-barev). Čtvrté číslo je alfakanál a jeho hodnota se zapisuje jako desetinné číslo v rozmezí od nuly do jedné (podobně jako tomu je u opacity).