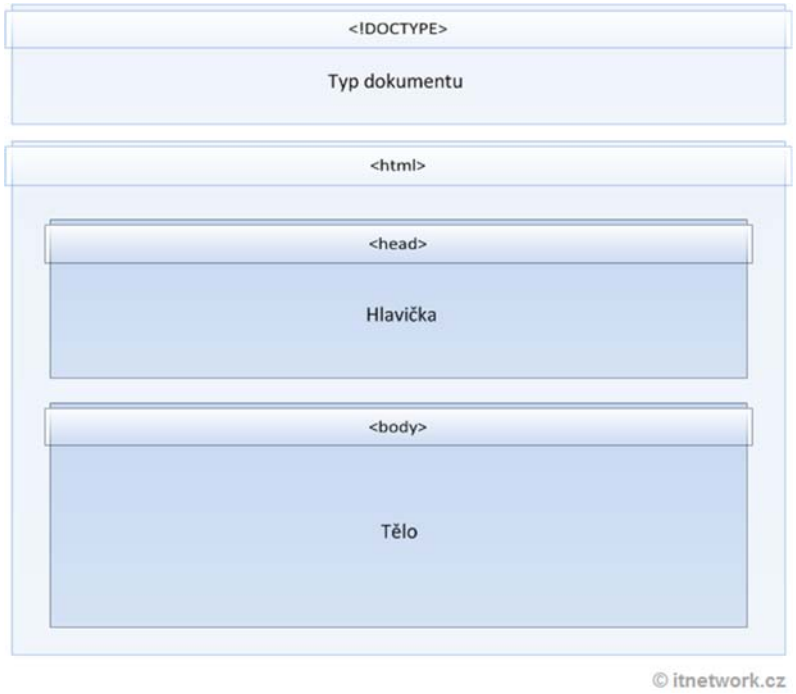
HTML - obsah

Responsivní web – obsah

CSS – obsah

Základní struktura HTML



<!DOCTYPE>
první element stránky, který obsahuje informaci o typu dokumentu
zápis: <!DOCTYPE html>

<html> *(párový)*
obaluje celý HTML dokument a skládá se ze dvou částí: první obsahuje informace pro prohlížeč (říká se ji hlavička), druhá obsahuje informace pro uživatele - obsah stránky (říká se ji tělo).
Atributy:

- ***manifest*** – obsahuje umístnění manifestu pro cache
- ***lang*** – specifikace jazyka stránky (česká = cs-cz)
- ***xmlns*** – pokud chceme používat XML synaxi

<head> *(párový)*
obaluje celou lavičku HTML dokumentu s informacemi pro prohlížeč.

<body> *(párový)*
obaluje celé tělo HTML dokumentu s informacemi pro uživatele.

Tagy v hlavičce HTML

<title> *(párový)*
povinný údaj, který specifikuje titulek sránky a je velmi důležitý pro vyhledávač z hlediska SEO. titulek by měl obsahovat klíčová slova a zobrazí se jako nadpis záložky v prohlížeči.

<meta> *(nepárový)*
Poskytuje meta data, informace, které jsou určeny pro vyhledávače a jsou důležitá pro SEO.
Atributy:

- ***charest*** – specifikuje znakovou sadu dokumentu (nejčastěji UTF-8)
- ***content*** – specifikuje parametr pro následující dva údaje:
  - ***http-equiv*** – umožňuje simulovat hlavičku http odpovědi
  - ***name*** – specifikuje název metadat

<base> *(nepárový)*
umožňuje nastavit kořenovou složku pro relativní odkazy v dokumentu.
Atributy:

- ***herf*** – specifikuje kořenovou složku pro všechna relativní URL v dokumentu
- ***target*** – specifikuje cílové umístění pro všechny odkazy

<link> *(nepárový)*
umožňuje provázání dokumentu s externím souborem.
Atributy:

- ***href*** - specifikuje umístění připojovaného souboru
- ***hreflang*** - specifikuje jazyk připojovaného dokumentu
- ***media*** - specifikuje typ zařízení, pro které je připojovaný dokument určen
- ***rel*** - specifikuje vztah mezi dokumenty
- ***sizes*** - umožňuje specifikovat velikost ikony (žádný prohlížeč tento atribut ale nepodporuje)
- ***type*** - MIME typ připojovaného souboru (dokumentu)

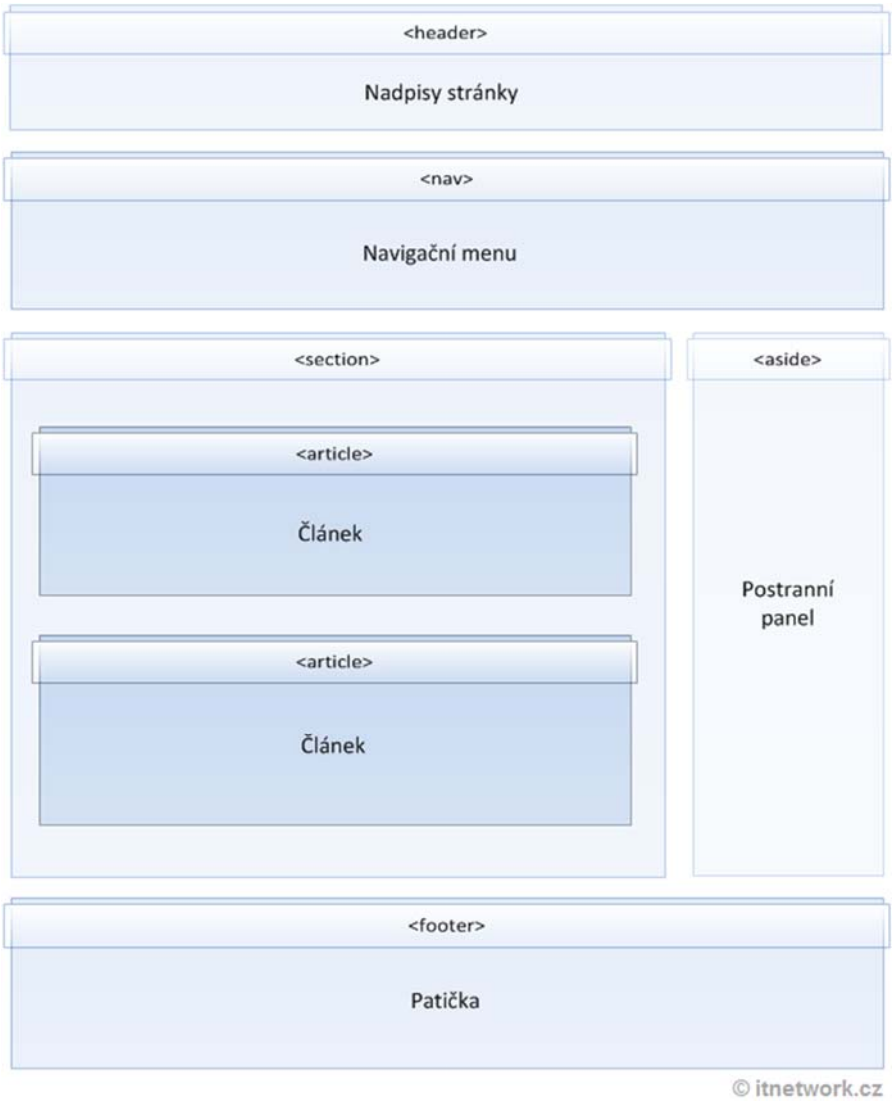
<style> *(párový)*
slouží k stylování přímo do HTML dokumentu, ale příliš se nepoužívá, jelikož se preferují styly v externím souboru.
Atributy:

- ***type*** - specifikuje typ stylu (jako hodnotu uvedeme text/css)
- ***media*** - specifikuje typ zařízení, pro který je styl optimalizován
- ***scoped*** - pokud je atribut uveden, může tag stát mimo hlavičku

<script> a **<noscript>** *(párový)*
používají se pro skriptovací jazyk JavaScript

Tagy v těle HTML

obsah HTML dokumentu se nachází v elementu <body> a to dále rozdělujeme do několika sekcí s různým významem, které tvoří tzv. layout:



<header> *(párový)*
definuje hlavičku stránky a zpravidla obsahuje nadpis [h1] (případně 2 nadpisy sloučené s tagem [hgrup].
váže se k celému webu, ale můžeme jej vložit i do dalších sekcí stránky.
Header dále zpravidla obsahuje obrázek s logem, vyhledávací pole, nebo např. možnost výběru jazyka

<nav> *(párový)*
obsahuje navigační prvky. může být součástí header, nebo stát samostatně pod ním, nebo kdekoliv na stránce (ale zvykem je mít navigaci pod hlavičkou a obvykle obsahuje nasylovaný seznam [ul] jehož položky reprezentují odkazy na jednotlivé podstránky)

<footer> *(párový)*
patička obvykle obsahuje copyright, informace o autorovi a případně další odkazy

<section> *(párový)*
se používá zejména k označení těla dokumentu (mezi hlavičkou a patičkou) a nalézá se zde hlavní obsah

<article> *(párový)*
sem píšeme samotný článek ten opět může obsahovat header-section-footer, což je pak přínosné pro vyhledávače, jelikož vidí na jakou část webu se zaměžit.

<aside> *(párový)*
tag pro postranní panel, který je zpravidla chápán, jako soubor doplňujících informací k článku a jsou zde obvikle umístěny odkazy na další podobné články, informace, nebo reklamy.

<main> *(párový)*
je tag pro unikátní obsah – tedy ne pro obsah, který se na stránce opakuje (hlavička, zápatí,…), ale měl by v něm být každý článek – ten je totiž na síti unikátní. Tag by měl být na stránce pouze jeden a neměl by být součástí zvýrazňujících tagů [article], [aside], [footer], [headerr] a [nav].

<time> *(párový)*
slouží k lepší indexaci stránky a případně pro upomínkovač

<summary> a **<details>** *(párový)*
provázané tagy, kdy tag [summary] označuje titulek sekce s detaily a [details] by měl být shrnutím skrytých detailů

<figure> a **<figcaption>** *(párový)*
tag [figure] označuje ilustraci týkající se článku, a nemusí to být jen obrázek, ale i tabulka nebo zdrojový kód, k doplnění hlavního obsahu a tag [figcaption] pak k tomu nastavuje popisek

Textové tagy v HTML

<p> *(párový)*
paragraph – označuje odstavec textu

<h1> *(párový)*
nadpis nejvyšší úrovně a smí být v HTML dokumentu jenom jednou
obsahuje název webu nebo nějaký slogan nebo i obrázkové logo

<h2> *(párový)*
nadpis druhé úrovně, používá se jako nadpis podstránky nebo článku
leží vždy pod nadpisem <h1>

<h3, h4, h5, h6> *(párový)*
další 4 nadpisy jsou určeny pro lepší orientaci v textu (poslední 2 úrovně se příliš nepoužívají)
Atribut nadpisů:
align – pro zarovnávání (již není validní, neb zarovnáváme v CSS)

<q> *(párový)*
quote – označuje krátkou řádkovou citaci a používá se pro citaci z jiného zdroje
prohlížeč při základním nastavení vykreslí citovaný text v uvozovkách
Atribut:
cite – označuje zdroj z kterého citace pochází

<blockquote> *(párový)*
jako <q>, pro blokové citace

<cite> *(párový)*
používá se pro označení názvu díla – může se jednat o knihu, článek, hru, …
nepoužívá se k označení jména autora

<ins> a **<del>** *(párový)*
tag <ins> označuje podtržením text, který byl přidán po jeho editaci
tag <del> označuje přeškrtnutím text, který byl odebrán
Atributy:
cite – obsahuje URL dokument, který objasňuje změnu
datetime – obsahuje datum změny ve formátu YYY-MM-DD hh:mm:ss

<sub> a **<sup>** *(párový)*
tag <sub> označuje dolní index a tag <sup> označuje horní index

<wbr> a **
** *(nepárový)*
word break – používáme když dlouhé slovo chceme zalomit, např. aby se nám vešlo do rámečku
break – pouříváme když cheme zalomit text řádku

<ruby> *(párový)*
definuje tzv. ruby anotace, které se používají v asuijské typografii aby pomáhali s výslovností
(symboly se vkládají do <ruby> a vysvětlivky do <rt>

<rt> a **<rp>** *(párový)*
tag <rt> označuje danou informaci o výslovnosti, tag <rp> není povynný a uvnitř elementu <ruby>
označuje text, který se zobrazí v případě, když prohlížeč ruby notace nepodporuje.

Frázové tagy v HTML

jedná se o řádkové elementy, které se vyskytují v odstavcích <p> a umožňují dodat části textu nějaký význam. Těchto tagů si všímají vyhledávače a jsou důležité z hlediska SEO optimalizace a dělíme je do základních tří skupin: Základní, Počítačové (programátorské), Prezentační

Základní frázové tagy (vyskytují se v běžném textu):

<abbr> *(párový)*
abbrevivation – označuje zkratku v textu a má atribut ***title*** v kterém píšeme celý význam zkratky

**** *(párový)*
emphasis – označuje část textu, která má větší význam, než okolní text
text je pak ve výchozím nastavení vykreslen jako kurzíva

**** *(párový)*
je významově podobný , ale označuje ještě silnější zdůraznění
tohoto textu si vyhledávač všímá nejvíce

<dfn> *(párový)*
definition – se používá k označení části textu, kde definujeme nějaký pojem

Tabulky v HTML

<table> *(párový)*
obaluje celou tabulku a má jediný atribut:

border – označuje šířku rámečku (výchozí hodnota je 0 = bez rámečku)
vzhledem k tomu, že formátování se přesunulo do CSS má podle nového standardu obsahovat pouze prázdné uvozovky pro rozvržení layout (nesématické) ,
nebo 1 (v uvozovkách), pokud v ní máme data

<tr> *(párový)*
table row – řádek tabulky – obaluje jednotlivé buňky

<td> a <th> *(párový)*
[td] table data – data tabulky – buňka může obsahovat text, obrázky a další elementy.
výchozí formátování je vlevo

[th] table head – hlavička tabulky. Vkládá se namísto <td> do hlavičky
výchozí formátování je tučné písmo zarovnané na střed

Atributy tagů <td> a <th>

colspan – pro sloučení buněk v rámci řádku – kromě první, další propojené buňky už se nepíší
rowspan – pro sloučení buňek v rámci sloupců - kromě první, další propojené buňky už se nepíší
headers – atribut udává hlavičky, ke kterým se buňka váže – pro hlasové čtečky
scope – tag určuje zd se popisek v hlavičce vztahuje k sloupci nebo řádku – pro hlasové čtečky
(pouze pro <th>)
col – hlavička patří sloupci
row – hlavička patří řádku
colgrup – havička patří více sloupcům
rowgroup – hlavička patří více řádkům

Pokročilé formátování tabulky
<thead> *(párový)*
obaluje hlavičkový řádek (obsahuje buňky <th>)

<tbody> *(párový)*
obaluje tělo tabulky (obsahuje buňky <td>)

<tfoot> *(párový)*
obaluje patičku tabulky (obsahuje buňky <td>)

<caption> *(párový)*
obaluje nadpis tabulky a vkládá se za tag <table>

<colorgroup> *(párový)*
obaluje tagy <col> pro definici formátování skupin sloupců v CSS

<col> *(párový)*
definují styl sloupců ve skupině a má atribut:

span – uvádí kolik sloupců element <col> představuje

Seznamy v HTML

unordered list – neuspořádaný seznam:
<ul> *(párový)*
položky se standardně zobrazují s odrážkami, které pomocí atributu [type] můžeme změnit

<menu> a **<dir>** *(párový)*
starší verze tagů nahrazena tagem

<li> *(párový)*
list item - označuje jednu položku ze seznamu

ordered list - uspořádaný seznam
<ol> *(párový)*
liší se od neuspořádaného seznamu tím, že prvky jsou řazeny podle nějakého klíče a místo odrážek se nám v základním stylu zobrazí písmena, čísla , římské číslice, atd.
Atributy:

reversed – položky budou číslovány v opačném pořadí
start – určuje první číslo v seznamu
type – nastavuje typ číslování:
1 – pro číslice
A/***a*** – pro písmena
I/***i*** – pro římské číslice

<li> *(párový)*
list item - označuje jednu položku ze seznamu
Atributy:

value – označuje číslo dané položky, ostatní se automaticky číslují od této hodnoty

Definition list – slovníček pojmů
<dl> *(párový)*
na rozdíl od ostatních obsahuje 2 typi položek, tagy [dt] a [dd], které se střídají

<dt> a **<dd>** *(párový)*
do [dt] (definition term) se zapisuje zpravidla na novém řádku u levého okraje stránky a obsahuje heslo, ke kterému pak [dd] (definition definition) obsahuje popis, či vysvětlení

<datalist> *(párový)*
se používá k označení skupiny možností, které jsou po tom navrhovány v elementu [input], jedná se tedy o tzv. našeptávač

Počítačové (programátorské) tagy v HTML

tagy vytvořené za účelem zobrazení zdrojových kódů a dokumentování aplikací

<code> *(párový)*
označuje úryvek zdrojového kódu, který se pak zobrazí
často se vkládá do tagu <pre>, který zachovává bílé znaky

<samp> *(párový)*
sample output – slouží k označení textu, který vypíše aplikace, tedy jejího výstupu

<kdb> *(párový)*
keyboard – označuje text, který má být uživatelem vložen z klávesnice

<var> *(párový)*
variable – slouží k označení proměnné v textu

Prezentační tagy v HTML

jsou tagy, které byli zavedeny ve starších verzích HTML pro stylování textu a následně byli nahrazeny výše zmíněnými tagy. V HTML5 byl jejich význam redefinován, nicméně se nedoporučují používat, protože jejich význam je matoucí ve smyslu stylování pomocí tagů namísto CSS

<b> *(párový)*
bold – označuje tučný text, ale zároveň není důležitý (alternativa - ,)

<i> *(párový)*
italic – označuje text který je řečený jiným přízvukem, nebo v jiné náladě, text je vykreslen kurzívou

<u> *(párový)*
underline – podtržení - prakticky se nepoužívá, protože je zažité, že na stránce jsou podtržené jen odkazy

<s> *(párový)*
strike – přeškrtnutý text, označující neaktuálnost, či nekorekntost (plést se může s <del>)

<mark> *(párový)*
při standardním nastavení žluté zvíraznění textu. Používá se pro aktuální potřeby uživatele, nebo např, při zvýraznění části citace, která je klíčová

Zastaralé tagy v HTML

následující tagy byli odebrány z HTML specifikace a neměli bychom je tedy používat

Stylovací tagy:
do HTML dokumentů již nepatří, protože jsou nahrazeny CSS variantou

<big>
vykreslil větší text než normální (řešíme v CSS tagem)

<small>
tento tag byl ponechán a slouží k označení textu s nízkým významem

<strike>
vykreslí text jako přeškrtnutý (alternativa: tag <s>, nebo <del>)

<basefont>
se dříve používal jako nadřazený element, který nastavil typ, velikost a barvu všem potomkům, elementům, které byli pod ním (nyní již řešíme v CSS)

<font>
umožňoval nastavit typ velikost a barvu písma (nyní již řešíme v CSS)

** **
sloužil k centrování elementů (nyní již řešíme v CSS)

<applet>
byl používán pro vložení pluginu appletu (alternativa: <video>)

<acronym>
se používal jako vysvětlivka zkratky (alternativa: <abbr>)

<command>
měl původně sloužit pro vložení příkazu do kontextového menu, kde mohl mít podobu radiobuttonu, checkboxu, nebo tlačítka.

Rámec:
se používali k zobrazení podstránek a z HTML byli odebrány ze dvou důvodů, zaprvé vyhledávače s nimi měli problém, a pak taky nemožnost dostat se na určitou stránku pomocí URL adresy

<frameset>
obsahoval soubor rámů a do HTML stránky se vkládalo na místo <body> (alternativa: <iframe>)

<frame>
byl rámec, ve kterém se zobrazovala samotná HTML stránka (alternativa: <iframe>)

<noframes>
do tohoto tagu se vkládal obsah, který se zobrazil v případě, že daný prohlížeč rámec nepodporoval (alternativa: <iframe>)

Multimédia v HTML

<img> *(nepárový)*
image – tag pro vkládání obrázků
Atributy:

scr – cesta k souboru s obrázkem (je dobré mít všechny obrázky v jedné složce)
srcset – podobný jako src, akorát používá více druhů rozlišení obrázku (pro mobil a PC)
výhoda je rychlejší načítání na menších obrazovkách, nevýhodou potřeba více verzí obrázku
alt – alternativní text popisující, pro vyhledávače a hlasové čtečky, co je na obrázku
width – pro šířku obrázku v pixelech nebo procentech
height – pro výšku obrázku v pixelech nebo procentech
(vždy je ale lepší nahrávat obrázek v požadované velikosti)
isamp – po kliknutí na obrázek odesílá na server souřadnice, kam uživatel na obrázek klikl

<map>
umožňuje definovat mapu na obrázku, několik oblastí s různými tvary, a ty budou rovnou fungovat, jako odkazy a má jeden atribut:

name – specifikuje jméno mapy, pomocí které ji potom přiřadíme k obrázku
souvisí s tagem <area>

<area>
označuje jednu oblast na mapě obrázku, které se pak zneaktivní a fungují jako odkazy
Atributy:

alt – alternativní text k oblasti z obrázku
coords – souřadnice oblasti
href – odkaz na stránku, která se otevřže po kliknutí na odkaz
herflang – kód jazyka stránky
media – možnost specifikovat zařízení, pro které je výsledná stránka optimalizovaná
rel – vztah mezi aktuálním a cílovým dokumentem
shape – tvar oblasti
- ***default*** – celý obrázek
- ***rect*** – obdelník (souřadnice začínají levou horní)
- ***circle*** – kruh (souřadnice jsou xy středu a poloměr
- ***poly*** – mnohoúhelník (souřadnice jednotlivých bodů)

target – kam se má cílový dokument otevřít
type – mime typ cílového URL

<canvas>
umožňuje do HTML vložit kreslící plátno, samotné kreslení se potom provádí scripty (např. JavaScript)
Atributy:

height – výška plátna v pixelech
width – šířka plátna v pixelech

<embed>
tag se v HTML5 používá pro vložení externí aplikace nebo plug-inu (např. Flash)
namísto tohoto tagu je však více doporučován tag <object>
Atributy:

height – výška aplikace v pixelech
widht – šířka aplikace v pixelech
scr – URL adresa aplikace
type – MIME typ obsahu

<object>
používá se pro vložení multimediální aplikace přímo do HTML dokumentu (Flash, Java, PDF, …)
vloženým objektům můžeme předávat parametry pomocí <param>
Atributy:

data – URL vkládaného obsahu
form – ID formuláře do kterého element patří
name – název objektu
type – MIME typ vkládaného obsahu
usemap – umožňuje připojit k objektu klikací mapu
height – výška objektu v pixelech
widhr – šířka objektu v pixelech

<param>
umožňuje vkládat atributy do objektu, které se následně předávají vkládané aplikaci
Atributy:

name – název parametru
value – specifikuje hodnotu parametru

<audio>
slouží k vložení zvukové stopy do HTML dokumentu
Atributy:

autoplay – automaticky spustí zvuk
controls – zobrazí přehrávač s ovládacími prvky
loop – přehrává zvukovou stopu stále dokola
preload – umožňuje nastavit načítání médií do paměti (zrychlí start přehrávání, zpomalí načítání stránky)
- ***auto*** – po načtení stránky se automaticky načte celý soubor
- ***metadata*** – načte se pouze hlavička
- ***none*** – soubor se do paměti nenačte

scr – specifikuje umístění zvukového souboru (alternativa je tag <source>)

Podpora audio formátů:

Kodek	IE	Edge	Firefox	Chrome	Safari	Opera Mini	Opera
MP3	Ano	Ano	Ano	Ano	Ano	Ne	Ano
OGG	Ne	Ano	Ano	Ano	Ne	Ne	Ano
WAV	Ne	Ano	Ano	Ano	Ano	Ne	Ano

Když prohlížeči nabídneme více formátů, vybere si ten první, který bude fungovat, proto je dobré jako první dávat ty s nyšší komprecí a menším datovým tokem.

<source>

představuje zdroj média pro tagy <audio> a <video>. Do těchto tagů se vkládá a umožňuje specifikovat několik zdrojových souborů, tak aby si mohl prohlížeč vybrat.

Atributy:

- ***media*** – specifikuje typ zařízení pro které je médium určeno
- ***scr*** – udává zdrojový soubor média
- ***type*** – MIME typ média

<video>

slouží k vložení videa do HTML dokumentu a je podporován všemi moderními prohlížeči
(ve starších se objeví jen alternativní text)

Atributy:

- ***autoplay*** – automaticky spustí video
- ***controls*** – zobrazí přehrávač s ovládacími prvky
- ***height*** – nastavení výšky přehrávače
- ***widht*** – nastavení šířky přehrávače
- ***loop*** – opakování videa ve smyčce
- ***muted*** – vypnutí zvuku při přehrávání
- ***poster*** – URL obrázku, který se zobrazí na pozadí přehrávače
- ***scr*** – specifikuje umístění zvukového souboru
- ***preload*** – umožňuje nastavit načítání médií do paměti (zrychlí start přehrávání, zpomalí načítání stránky)
  - ***auto*** – po načtení stránky se automaticky načte celý soubor
  - ***metadata*** – načte se pouze hlavička
  - ***none*** – soubor se do paměti nenačte

Podpora video kodeků:

Kodek	IE	Edge	Firefox	Chrome	Safari	Opera Mini	Opera
MP4	Ano	Ano	Ano	Ano	Ano	Ne	Ano
OGV	Ano	Ano	Ano	Ano	Ne	Ne	Ano
WebM	Ne	Ano	Ano	Ano	Ano*	Ne	Ano**

* Pouze částečná podpora

** Pro mobilní prohlížeč Opera je pouze částečná podpora.

<track>

slouží k vložení textové stopy a používá s k vložení titulků a dalších textových popisků v videu a i audiu
jeho jediným formátem je WebVTT (.vtt), který se podobá formátu .str, musí aly být kódován v UTF-8

Atributy:

- **default** – pokud je uveden, je brána daná stopa jako výchozí, nehledě na nastavení jazyka
- **kind** – specifikuje druh popisků
  - **captions** – popis dialogů a zvuků ve videu pro neslyšící
  - **chapters** – popisky kapitol pro lepší orientaci cve videu
  - **description** – textový popis videa
  - **metadata** – popis videa pro scripty (uživatelé ho nevydí)
  - **subtitles** – titulky
- **label** – popisek textové stopy
- **srclang** – specifikace jazyku titulků
- **src** – cesta ke zdroji titulků, či videa

<inframe>

inline frame – řádkový rámec – pomocí tohoto elementu můžeme odkázat např. na You Tube video

Atributy:

- ***height*** – udává výšku rámu v pixelech
- ***widht*** – udává šířku rámu v pixelech
- ***name*** – udává název rámu
- ***sandbox*** – umožňuje zakázat některé akce na stránce v <inframe>
  - ***allow-same-origin*** – povolí stejný původ stránky vloženého a vkládaného dokumentu
  - ***allow-top-navigation*** – povolí obsahu rámce, aby se navigoval na stránku ve které je vložen
  - ***allow-forms*** – povolí odesílání formulářů
  - ***allow-scripts*** – povolí spouštění scriptů
- ***seamless*** – zobrazuje dokumentu v <iframe> jako by byl součástí hlavního dokumentu
- ***scr*** – URL adresa zdrojového dokumentu
- ***srcdoc*** – umožňuje přímo specifikovat vložený dokument (obsahuje HTML kód)

Formuláře v HTML

<form>

základní tag pro vkládání formulářů, obaluje celý formulář

Atributy:

- ***accept-charset*** – povolení pouze určitých znakových sad (nejčastěji UTF-8)
- ***action*** – určuje URL adresu skriptu, který bude formulář spracovávat
- ***method*** – určuje metodu odesílání dat (POST, GET)
- ***autocomplete*** – nabízí uživateli při vyplňování údajů našeptávač
- ***name*** – přiřazuje formuláři název
- ***novalidate*** – specifikuje, zda se má formulář před odesláním zvalidovat
- ***target*** – specifikuje cílové umístění odpovědi formuláře
- ***enctype*** – určuje způsob kódování obsahu formuláře

<input>

používá se ve formulářích, která umožňují uživateli vkládat data a může mít mnoho podob, která se liší různým nastavením atributů. Atribut definujeme pomocí typů:

checkbox	Zaškrtávací pole (checkbox)
radio	Přepínací tlačítko (radiobutton)
text	Zadání krátkého textu
password	Zadání hesla
submit	Tlačítko k odeslání formuláře
image	Tlačítko s obrázkem k odeslání formuláře
reset	Vyprázdní formulář
button	Tlačítko
color	Výběr barvy (color picker)
time	Zadání času
week	Zadání týdnu a roku
month	Výběr měsíce a roku
date	Výběr data (date picker)
datetime-local	Výběr data a času (datetime picker)
email	Zadání emailové adresy
url	Zadání URL adresy
tel	Zadání telefonního čísla
number	Zadání čísla
range	Zadání čísla posuvníkem
file	Nahrání souboru (upload)
search	Vyhledávací pole
hidden	Skryté pole

Atributy společné pro všechny typy inputů:

- ***autocomplete*** – nabízí uživateli našeptávač, který zobrazuje dříve zapsané hodnoty
- ***disable*** – pokud je uveden, nelze měnit hodnotu elementu a ten je zašedlý
- ***list*** – položky listu slouží jako předdefinované možnosti elementu <input>
- ***readonly*** – pokud je uveden, nelze hodnoty v daném poly editovat
- ***name*** – definuje jméno elementu
- ***value*** – přednastavená hodnota pole
- ***autofocus*** – pokud je uveden, element se zaktivní po načtení stránky
- ***form*** – slouží k připojení definice <input> učiněné mimo formulář
- ***pattern*** – obsahuje regulární výraz, podle kterého se má hodnota v poli validovat
- ***placeholder*** – definuje text, který se zobrazí v případě, že je pole prázdné
- ***required*** – označuje povinná pole k vyplnění, pro odeslání formuláře

[typ] checkbox

je podobný radiobutton, ale je možné zaškrtávat více možností. Pokud spolu checkboxy souvisí, můžeme je združit do pole, v atributu name budou mít hranaté závorky.

Atribut:

- ***checked*** – pokud je uveden, je políčko zaškrtnuto

[typ] radio

vytvoří přepínací tlačítko, které můžeme seskupovat tak, že jim dáme stejný název (atribut name) při změně volby se původní volba označí, odesláno potom bude jen jedno vybrané tlačítko

Atribut:

- ***checked*** – pokud je uveden, je políčko zaškrtnuto

[typ] text

jedná se o krátký text na jeden řádek, často tento [typ] můžeme vidět v kombinaci s typem password

Atribut:

- ***size*** – počet znaků – výchozí hodnota je 20
- ***maxleng*** – maximální délka zadávaného textu

[typ] password

[typ] pro zadávaní hesla – funguje stejně jako [text], ale místo znaků jsou zobrazeny hvězdičky

[typ] submit

jedná se o odesílací tlačítko, které vyvolá odeslání hodnot na server a měl by být součástí každého formuláře, který může obsahovat více typů [submit]. Atribut [value] nastavuje popisek tlačítka

Atribut:

- ***formaction*** – změní atribut [action] formuláře
- ***formenctype*** – změna kódování formuláře
- ***formmethod*** – změna metody formuláře
- ***formnovalidate*** – změna validace formuláře
- ***formtarget*** – změna cíle pro odeslání formuláře

[typ] image
slouží pro zobrazení grafického tlačítka [submit] (obrázek namísto tlačítka) – pokud nám prohlížeč obrázek nezobrazí,zobrazí se popis zadaný v atributu [alt]
Atributy jsou stejné jako u [submit] + tyto dva:
width – pro šířku obrázku v pixelech
height – pro výšku obrázku v pixelech

[typ] reset
zobrazí resetovací tlačítko, které po stisknutí obnoví výchozí hodnoty ve formuláři

[typ] button
funguje jako jednoduché tlačítko, které můžeme naprogramovat tak, aby ovládalo určitou funkci, nejčastěji [onclick]. Tento [typ] nahradil novější element <button>

[typ] color
formulář může taky obsahovat výběr barvy a k tomu se používá tento typ

[typ] time
tento [typ] pužijeme například pro výběr času schůzky, či pro čas návštěvy u lékaře (podporují jen některé prohlížeče)

[typ] week
tento [typ] použijeme pro výběr týdne v daném roce (podporují jen některé prohlížeče)

[typ] month
tento [typ] použijeme pro výběr měsíce v daném roce (podporují jen některé prohlížeče)

[typy] date, datetime-local
typ [date] vytvoří vstupní pole pro výběr námi zvoleného data, výsledek obsahuje rok, měsíc a den
typ [datetime-local] představuje místní datum a čas, což nemusí být schodné s datumem a časem uživatele (podporují jen některé prohlížeče)
Atributy:
min – nejdřívější datum, které můžeme vybrat
max – nejvýše možné datum, které můžeme vybrat
step – specifikuje krok, např „2“ = každý druhý den

[typy] email, url
typ [email] vytvoří pole pro zadání emailu – před odesláním je ověřena validita
typ [url] vytvoří pole pro zadání URL – před odesláním je ověřena validita
Atribut:
maxlength – maximální počet znaků
minlength – minimální počet znaků
multiple – povolení pro zadání více emailů oddělených čárkou
size – velikost pole – počet znaků (výchozí hodnota je 20)

[typ] tel
pole pro zadání telefonu, vzhledem k různorodosti telefonních čísel se nekontroluje, využití ale najde především v mobilních prohlížečích, které po výběru pole typu [tel], zobrazí numerickou klávesnici
Atribut:
maxlength – maximální počet znaků
minlength – minimální počet znaků
size – velikost pole – počet znaků (výchozí hodnota je 20)

[typ] number
umožňuje uživately zadat číslo, nebo jej inkrementovat, či dekrementovat pomocí šipek v poli.
V mobilních aplikacích se nám zobrazí numerická klávesnice
Atributy:
max – maximální možné číslo
min – minimální možné číslo

[typ] range
jedná se o slider, který přesně specifikuje rozmezí, které můžeme vybrat, typické pro snížení a zvýšení hlasitosti, nevýhodou je, že uživatel nepozná přesně jakou hodnotu zadal
Atributy:
max – maximální možné číslo
min – minimální hodnota
step – specifikuje o kolik se hodnota zvýší, či sníží posunutím

[typ] file
umožňuje k formuláři přiložit soubor, ten se po tom odešle spolu s formulářem
Atributy:
accept – umožňuje určit MIME souborů, které je přípustné vybrat
multiple – umožňuje vybrat více souborů

[typ] search
pokud náš prohlížeč podporuje [typ] [search], tak se nám po zadání textu zobrazí vpravo malý křížek, kterýnám smaže zadaný text v tomto poli

[typ] hidden
označuje skryté pole a používá se ve specifických případech, kdy chceme s daty uživatele odeslat i nějakou další informaci, o které uživatel nemusí vědět, nebo ji nemá měnit.

Tagy k tvorbě formulářů v HTML

<label>
slouží k popisu polí – text vedle pole, který udává, k čemu pole slouží
Atributy:
for – obsahuje id elementu ke kterému patří popisek
form – slouží k připojení definice <label> učiněné mimo formulář

<output>
používá se k výsledku nějaké počáteční operace, např výstup JavaScriptu
Atributy:
for – obsahuje id elementu ke kterému patří popisek
form – slouží k připojení definice <label> učiněné mimo formulář
name – jméno elementu

<keygen>
používá se k bezpečnému odeslání formuláře pomocí certifikátu (po odeslání formuláře je vygenerován pár klíčů – privátní a veřejný (tento tag není mnoha prohlížeči podporován)
Atributy:
autofocus – pokud je uveden, element se zaktivní po načtení stránky
challenge – umožňuje nastavit chaleng string
disable – zneaktivňuje element, ten je pak zašedlý
form – vložení elementu do formuláře přes ID
keytype – algoritmus ke generování klíče
name – jméno elementu

<option>
je součástí tagů [select] a [datalist] a reprezentuje jednu z možností, které může uživatel vybrat
Atributy:
disable – pokud je uveden, možnost nelze vybrat
label – udává zkrácený popisek možnosti, který se zobrazí v seznamu
selected – pokud je uveden, je daná možnost předem vybrána
value – udává hodnotu, která se má poslat servru

<select>
umožňuje vybrat jednu, nebo více možností z uvedených hodnot, jednotlivé možnosti vkládáme pomocí tagu [option]
Atributy:
autofocus – pokud je uveden, element se zaktivní po načtení stránky
disable – zneaktivňuje element, ten je pak zašedlý
multiple – pokud je uveden, je možnost zaškrtnout více možností
size – určuje počet zobrazených možností, pokud má hodnotu 1 je zobrazen jako vyjíždějící combox, pokud je uvedeno vyšší číslo, je zobrazený jako rozbalený seznam možností.
form – vložení elementu do formuláře přes ID
name – jméno elementu

<optgroup>
tímto tagem můžeme seskupovat možnosti v elementech [select] a [datalist]
Atributy:
label – popisek skupiny možností
disabled – pokud je uveden, možnost v dané skupině nelze vybrat

<fieldset>
slouží k seskupení souvisejících polí ve formuláři, díky tomu vypadá formulář přehledněji
Atributy:
disable – zneaktivňuje element, ten je pak zašedlý
form – vložení elementu do formuláře přes ID
name – jméno elementu

<legend>
tag označuje nadpis pro [fieldset]

<textarea>
slouží k zadání libovolně dlouhého textu s více řádky do formuláře, jako výchozí vykreslení je použito neproporciální písmo
Atributy:
autofocus - pokud je uveden, element se zaktivní po načtení stránky
cols - počet sloupců, znaků na řádek. (spíše se nastavuzje v CSS vlastnost width)
rows - počet řádků pole. (spíše se nastavuzje v CSS vlastnost height)
disabled - pokud je uveden, nelze do pole vkládat text a je obvykle zašedlé
form - propojuje element, který je definován mimo formulář
maxlength - udává maximální počet znaků pro pole
name - definuje jméno elementu
placeholder - definuje text, který se zobrazí v případě, že je pole prázdné
readonly - pokud je uveden, nelze text uvnitř pole editovat
required - pokud je uveden,¨znamená to, že pole musí být pro odeslání formuláře vyplněné
wrap - udává, jak se má text v poli zalamovat (hard / soft)

<button>
umožňuje vložit tlačítko, které nemusí být bezprostředně součástí formuláře
Atributy:
autofocus - pokud je uveden, element se zaktivní po načtení stránky
disabled - pPokud je uveden, nelze na tlačítko kliknout a je zašedlé
form - propojuje element, který je definován mimo formulář
formaction - změní atribut action formuláře
formenctype - změna enctype
formmethod - změna method
formnovalidate - změna validate
formtarget - změna target
name - definuje jméno elementu
type - specifikuje typ tlačítka
value - hodnota tlačítka, která se po kliknutí odešle

Ostatní tagy v HTML

** **
nemá žádný sémantický význam a používá se k seskupování logicky souvisejících blokových elementů, ty se pak jeho pomocí mohou stylizovat v CSS (podobný význam má v HTML tag [span], který se používá k seskupení elementů řádkových. Od HTML5 se tento tag již nepoužívá na layout stránky.

****
se používá k seskupování logicky souvisejících řádkových elementů a funguje tedy podobně jako

<a>
anchor – kotva – se používá k definici hypertextového odkazu, můžeme se tak přesouvat mezi jednotlivými stránkami našeho webu, nebo odkázat na jiný web
Atributy:

- ***herf*** – specifikuje URL adresu odkazované stránky
- ***herflang*** – specifikuje jazyk odkazovaného dokumentu
- ***download*** – po kliknutí se začne stahovat soubor
- ***media*** – specifikuje pro která zařízení je obsah určen
- ***rel*** – specifikuje vztah mezisoučasným a odkazovaným dokumentem
- ***referrfrpolicy*** – určuje které informace o odkazu se mají odeslat společně s odkazem
- ***ping*** – po kliknutí odešle krátký požadavek HTTP POST na námi definovanou adresu
- ***target*** – definuje cíl odkazu
- ***type*** – specifikuje MIME typ odkazovaného dokumentu

<template>
je používán jako obsah, který je po načtení stránky skrytý před uživatelem, obsah se zobrazí až po nějaké akci, kterou vyvolá uživatel (tipicky se jedná o skryté informace o produktu, atd)

<dialog>
je používán jako takzvané dialogové okno, nebo podokno, a usnadňuje zobrazení vyskakovacích oken
Atribut:

- ***open*** – pokud uveden, bude dialogové okno vyditelné

<address>
poskytuje kontaktní informace k dokumentu, nebo jeho části, neobsahuje adresu, ale např. jméno autorů, a často se dává do patičky webu

<bdi>
bi-directional isolation – označuje text, který může být púsán jiným směrem než text okolní

<bdo>
bi-directional override – označuje přepsání aktuálního směru textu
Atribut:

- ***dir*** – specifikuje směr použitého textu

<hr>
označuje v HTML dokumentu konec, nebo změnu tématu, prohlížeč ho vykreslí jako vodorovnou oddělovací čáru

<meter>
v HTML5 označuje měřič, který zobrazuje určitou část z celku, je funkčně podobný [progress], ale nepoužívá se k zobrazení vykonané práce, ale všeho ostatního (např. volné místo ve složce, či počet dnů prémiového účtu)
Atributy:

- ***form*** – určuje do kterého formuláře element patří
- ***high*** – určuje od kolika je hodnota považovaná jako vysoká
- ***low*** - určuje od kolika je hodnota považovaná jako nízká
- ***max*** – maximální hodnota
- ***min*** – minimální hodnota
- ***optimum*** – optimální hodnota
- ***value*** – současná hodnota měřiče

<progress>
slouží k vložení progressbaru, významově by měl element reprezentovat vykonanou práci a není vhodný k použití jeko měřič, k tomu slouží výše zmíněný [meter]
Atributy:

- ***max*** – maximální hodnota (např. 16 úloh, nebo 100 procent)
- ***value*** – aktuální hodnota (např. hotovy 4 úlohy, nebo 25 procent)

<!—komentář --!>
označuje komentář, ve zdrojovém kódu stránky, tato komentáře slouží pro autory stránky, jako poznámka pro lepší orientaci v kódu, prohlížeč si komentáře nevšímá
často se používá k popisu složitých struktur, kdy je zanořeno více do sebe

HTML - obsah

Responsivní web – obsah

CSS – obsah

CSS – propojení s HTML

K propojení CSS s HTML máme 4 možnosti:

- Přímé propojení
- Interní CSS
- Připojení CSS souboru k HTML souboru
- Připojení externího CSS souboru z CDN (Cloud Delivery Network)

Přímé připojení
(inline style) je nejméně používaný a je dobré se tomuto způsobu vyhnout.
styly vkládáme za pomocí atributu [style] přímo k HTML prvku:

<h1 style="color: blue;">Učím se na ITnetworku.</h1>

Interní CSS
je zápis stylů do hlavičky HTML dokumentu mezi prvky <head>
styly vkládáme mezi tagy <style>, kde atribut [type] s hodnotou [text/style] říká, že se jedná o CSS

```
<head>

    <style type="text/css">
        h1{ color: blue; font-size:50px; }
    </style>
</head>
```

Připojení CSS souboru k HTML souboru
nejpoužívanější způsob, připojení externího souboru CSS (style.css) do HTML souboru je definováno v hlavičce HTML skrze tag <link> a patřičné atributy :

```
<head>

    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

atribut [**rel**] s hodnotou „**stylesheet**“ oznamuje, že tenhle CSS dokument určuje vzhled HTML
atribut [**type**] s hodnotou „**text/css**“ oznamuje, že se jedná o CSS soubor
atribut [**herf**] s hodnotou „**style.css**“ oznamuje, kdenje soubor uložen vzhledem k umístění HTML dokumentu (v tomto případě HTML a CSS jsou ve stejné složce)

Připojení externího CSS souboru z CDN
jedná se o připojení již napsaných stylů uložených někde na webu (např. bootstrap)

Selektory v CSS

slouží pro vybrání a označení částí HTML dokumentu, které chceme stylovat a zde jsou uvedeny ty nejpoužívanější, soubor CSS se čte od zhora dolu, takže pokud chceme přepsat již deklarovaný styl, vkládáme jej pod něj.

Všechny elementy
pro vybrání všech elementů dokumentu slouží znak hvězdičky [*]

**** {color: green;}*** obarví veškerý text na zeleno

Typový selektor
vybere všechny elementy námi vybraného typu a přidá jim styl

p {color: green;} obarví veškerý text odstavců na zeleno

Třídy
vybere všechny elementy vztahující se do námi deklarované třídy, v HTML dokumentu se třída deklaruje pomocí atributu [class] s názvem třídy, v CSS se třída deklaruje pomocí tečky [.] a názvu třídy

HTML = *** První odstavec *** (třídy je možné i slučovat zápisem názvů vedle sebe s mezerou)
CSS = ***.zelena {color: green;}***

Identifikátor
má větší prioritu než třída a používá se spíše pro JavaScript nebo odkazy na stránce, v HTML dokumentu je vložen za pomoci atributu [id] za kterým následuje název ID, v CSS jej deklarujeme pomocí hashtagu [#]

HTML = *** První odstavec. *** (odstavec může obsahovat i třídu i identifikátor)
CSS = ***#cervena {color: red;}***

Pseudo-selektory v CSS

jsou předdefinované styly (funkce), které můžeme použít na jakýkoliv HTML element a zapisují se mezi selektor s dvojtečkou a první vlnitou závorku: selector: pseudoselektor {vlastnost: hodnota;}

:hover
můžeme použít prakticky na jakýkoliv element, po najetí kurzorem myši na tento element se změní ty vlastnosti, které hoverem deklarujeme, například změna barvy

HTML = ***<h1 class="zmen-barvu">CSS3 selektory</h1>***
CSS = ***.zmen-barvu:hover {color: green;}*** – při najetí na nadpis se změní jeho barva z původní na zelenou

:firs-child
najde první element daného typu a přidá mu nějaký styl

HTML = ***obalíme všechny odstavce do ***
CSS = ***div :first-child {color: red;}*** – při použití se první element v tagu obarví na červeno

:nth-child(x)
říká, kolikátý element daného typu vybíráme (využívá se často u tabulek pro barevné rozlišení řádků)

HTML = ***obalíme všechny odstavce do ***
CSS = ***div p:nth-child(2n+0) {color: green;}*** – 2n říká že vybíráme 2. prvek, a 0 označuje začátek počítání
výsledkem je zelené písmo pro každý druhý odstavec, můžeme také využít hodnot [odd] lichý a [even] sudý

Jednotky v CCS3

Jednotky se dělí na dvě hlavní skupiny: absolutní a relativní; desetiné čísla oddělujeme tečkou (ne čárkou) a mezi číslem a jednotkou se nedělá mezera, stejně tak u mínusu u záporných hodnot, za nulovou hodnotu není jednotka povinná

Absolutní jednotky: cm, mm, in, px, pt, pc
Relativní jednotky: em, ex, ch, rem, vw, vh, vmin, vmax, %

Absolutní jednotky

jsou pevně dané a délka vyjádřená pomocí těchto jednotek se nám bude jevit přesně tak, jak je zapsána

Pixel [px]
nejedná se o pixel displaye, ale o referenční (tzv. CSS) pixel, skoro ve všech prohlížečích je základní velikost písma nastavená na 16px

p {font-size: 24px;}

Ostatní absolutní jednotky
se používají spíše výjmečně a tak jen pro zajímavost, tabulka přepočtů:

```
h1 { font-size: 0.5in; }        /* palce - 48px */
h2 { font-size: 1cm; }          /* cm - 37.79px */
h3 { font-size: 10mm; }         /* mm - 37.79px */
h4 { font-size: 12pt; }         /* body - 16px */
h5 { font-size: 1pc; }          /* picas - 16px */
h6 { font-size: 16px; }         /* pixel - 16px */
```

Relativní jednotky

u relativních jednotek se výsledná délka mění na základě jiné délky, nejčastěji rodiče elementu

EM
udává poměrnou velikost vůči velikost písma rodičovského elementu

p {font-size: 75%; padding: 1em;} – zarovnání okraze textového rámečku

REM
vychází z velikosti písma v dokumentu, obsahuje hodnotu výchozí velikosti písma nastavenou autorem a případně upravenou uživatelem, nebo prohlížečem (např. při zvětšení náhledu stránky uživatelem)

p {margin: 1rem;} - změníli se výška písma, změní se i výsledná velikost okraje textového rámečku

Procenta [%]
první způsob použití je procentuální hodnota z rodičovského elementu (např. u tabulek)
druhý způsob je procentuální hodnota z přirozených rozměrů elementu (100% u fontu je aktuální velikost)
třetím způsobem je procentuální šířka stránky, nebo výška okna prohlížeče (100% je u šířky okna maximální velikost)

viewportWidht (vw) & viewport Height (vh)
viewport units - procenta velikosti prohlížeče se odvozují od šířky (vw) a výšky (vh) prohlížeče, viewport je oblast prohlížeče, která zobrazuje obsah stránky bez lišt a nástrojů

div {width: 50vw; height: 50vh;} značí poloviční velikost divu vůči oknu viewportu prohlížeče

Text a písmo – Barva a typ písma

color [color: hodnota;]
nastavujeme barvu písma, kterou můžeme zadat několika způsoby:

- **<u>předdefinovanou hodnotou</u>** – zadává se slovně názvem barvy (HTML podporuje 140 názvů)
- ***RGB*** – zadává 3mi hodnotami (0-255) pro červenou, zelenou a modrou barvu
- ***RGBA*** – jako RGB, jen je zde ještě možnost průhlednosti
- ***HEX*** – zadává se hexadecimálním zápisem barvy v šestnáctkové soustavě (#rrggbb)
- ***HSL*** – reprezentuje odstín, sytost, svítivost
- ***HSLA*** – jako HSL, jen je zde ještě možnost průhlednosti

font-Family [font-family: hodnota;]
nastavuje font daného elementu, jako hodnotu můžeme zadat i několik písem (oddělujeme čárkami a pokud je název víceslovný, tak ho dáme do uvozovek), pokud uživatelský počítač nebude písmo umět zobrazit, použije druhé uvedené, atd., proto by poslední písmo mělo být generické, můžeme také použít externí fonty (např. od Googlu), ty musí být v uvozovkách pokaždé, do hlavička HTML dokumentu pak vložíme před propojení s CSS lin na font a v CSS použijeme font přes vlastnost font-family

HTML = ***<link href="https://...com/css2?family=Ubuntu:ital,wght@0,400;0,700;1,400&display=swap" rel="stylesheet">***

CSS = ***body {font-family: 'Ubuntu', sans-serif;}***

Text a písmo - Velikost, styl, dekorování, výška řádku

font-size [font-size: hodnota;]
udává velikost písma daného elementu, jako hodnotu můžeme použít slovní spojení (small, medim, large, …) nebo číselnou hodnotu v různých jednotkách (px, em, …), aby uživatelé mohli změnit velikost textu v prohlížeči, mnoho vývojářů používá em (doporučeno standardem W3C) namísto pixelů,

1em = aktuální velikost písma (= 16px), hodnotu em tedy vypočítáme vzorcem: pixel/16

font-style [font-style: hodnota;]
používá se pro nastavení stylu písma, tedevší pro nastavení kurzívy

hodnoty:

normal – normální písmo
italic – kurzíva
oblique – šikmé písmo
inherit – zdědění vlastnosti od rodičovského elementu

text-transform [text-transform: hodnota;]
používá se na převedení textu z malých na velká písmena a naopak
hodnoty:

none (výchozí) – bez transformace textu
capitalize – počáteční písmeno v každém slově je velké
uppercase – celý text velkými písmeny
lowercase – celý text malými písmeny
inherit – zdědění vlastnosti od rodičovského elementu

line-height [line-height: hodnota;]
udává výšku řádku textu
zadávání hodnot:

normal – výchozí nastavení
číslo bez jednotky – poměr k normalu - např. 1.6
délku (s hodnotou) – např. 20px, 2em
procenta

text-decoration [text-decoration: hodnota;]
styluje text různými styli čar
hodnoty:

none – výchozí, normální text
underline – podtržený text, čára pod textem
overline – čára nad textem
line-through – přeškrtnutý text, čára přes text
blink – blikající text (není moc prohlížeči podporován)
inherit – zdědění vlastnosti od rodičovského elementu

pomocné vlastnosti:

text-decoration-color - zajišťuje barvu dekorace textu
text-decoration-line - zajišťuje konkrétní typ dekorace
text-decoration-style - zajišťuje styl dekorace

syntaxi je pak možné zapsat dvěma způsoby:

p {text-decoration: overline red wavy;}
p {text-decoration: overline; text-decoration-color: red; text-decoration-style: wavy;}

Text a písmo - Zarovnání, stínování, tučnost písma, mezery

t**ext-align** [text-align: hodnota;]
zarovnává text v HTML elementu
hodnoty:

left – zarovnání textu naleva
right – zarovnání textu napravo
center – vycentrování textu
justify – zarovnání textu do bloků
inherit – zdědění vlastnosti od rodičovského elementu

t**ext-align-last** [text-align-last: hodnota;]
smyslově pojí se s [text-align] a zarovnává poslední řádek odstavce
hodnoty:

auto – výchozí – dle vlastnosti z [text-align]
left – řádek se zarovná doleva
right – řádek se zarovná doprava
justyfy – řádek se zarovná do bloku
center – řádek se vycentruje
start – v případě nastavení hodnoty [rtl] u vlastnosti [direction] doprava, nastaví zarovnání textu doprava
end – jako u [start], akorát doleva (na stejnou stranu jako v nastavení [direction])
initial – nastaví vlastnost na výchozí hodnotu
inherit – zdědění vlastnosti od rodičovského elementu

text-shadow [text- shadow: x y rozostření barva;]
přidá textu stín
hodnoty:

x – vodorovná pozice stínu vůči textu
y – svislá pozice stínu vůči textu
rozostření – nepovinný parametr udávající rozostření v délhových jednotkách
barva – nepovinný parametr udávající barvu stínu

font-weight [font-weight: hodnota;]
nastavuje tučnost písma v HTML elementu
hodnoty:

normal – výchozí – normální písmo
bold – tučné písmo
bolder – tučnější písmo
lighter – tenčí písmo
inherit – zdědění vlastnosti od rodičovského elementu

dále můžeme místo slovních hodnot uvést číselné hodnoty:

100, ***200***, ***300***, ***400*** (jako normal), ***500***, ***600*** (jako bold), ***700***, ***800***, ***900***

letter-spacing [letter-spacing: hodnota;]
snižuje nebo zvyšuje rozestoupení znaků v textu
hodnoty:

jako hodnotu doplňujeme libovolnou hodnotu s jedotkou délky (px, em,),
můžeme jít i do záporných hodnot
můžeme ale i použít slovní hodnotu ***normal*** což je výchozí rozestup mezi písmeny

Text a písmo - Směr, šíře, varianty, závěsná interpunkce

@font-face

nejedná se o vlastnost, ale o pravidlo, pomocí něhož definujeme na stránkách fonty pro text, na to potřebujeme odkaz na uložený font, ten po té uložíme pod proměnou, kterou budeme v našich třídách používat, u adresy fontu je potřeba ještě specifikovat formát používaného fontu,
podporované formáty jsou: tft/oft, woff, woff2, svg a eot (některé však fungují pouze na určitých prohlížečích)

zápis:

@font-face {font-family: "Název proměnné"; src: url("") format("");}

(v názvu proměné je dobré pro přehlednost používat názvy fontů)

v těle pravidla můžeme dále specifikovat:

font-stretch – jak blízko bude u sebe písmo (nepodporuje ji žádný prohlížeč)
font-style – styl písma fontu
font-weight – tučnost písma fontu

direction

pomocí této vlastnosti nastavujeme směr textu v HTML elementu, vlastnost se dědí a používá se hlavně v jazycích, které se píší zprava doleva

hodnoty:

ltr – výchozí - zleva doprava
rtl – zprava doleva
inherit – zdědění vlastnosti od rodičovského elementu

font-size-adjust

pomocí této vlastnosti nastavujeme velikost písma, tato vlastnost dává větší kontrolu nad nastavení stejné velikosti rozdílných rodin písma (podporuje pouze prohlížeč motzila a nemá dnes již reálné využití)

hodnoty:

none – výchozí – vlastnost se nepoužívá
číslo – hodnota poměru k použití
initial – nastaví vlastnost na výchozí hodnotu
inherit – zdědění vlastnosti od rodičovského elementu

font-stretch

pomocí této vlastnosti nastavujeme, zda budou písmena v textu užší, nebo širší, tuto vlastnost musí podporovat jak prohlížeč, tak vybraný font (dnes se již takév podstatě nikde nepoužívá)

hodnoty:

ultra-condensed - udělá text co nejužší.
extra-condensed - udělá text užší než hodnota condensed ale méně jak ultra-condensed.
condensed - udělá text užší než hodnota semi-condensed ale méně jak extra-condensed.
semi-condensed - udělá text jenom trochu úzký.
normal - výchozí - vykreslí text normálně.
semi-expanded - udělá text jenom trochu široký.
expanded - udělá text širší než hodnota semi-expended ale méně jak extra-expanded.
extra-expanded - udělá text širší než hodnota expanded ale méně jak ultra-expanded.
ultra-expanded - udělá text co nejširším.
initial - nastaví vlastnost na výchozí hodnotu (tedy na normal).
inherit - hodnota se zdědí z rodičovského elementu.

font-variant

pomocí této vlastnosti nastavujeme variantu písma v HTML elementu

hodnoty:

normal – výchozí – normální písmo
small-caps – kapilárky
inherit – zdědění vlastnosti od rodičovského elementu

hanging-punctuation

pomocí této vlastnosti nastavujeme jak bude umístěné interpunkční znaménko (podporuje pouze prohlížeč safari a v praxi se s touto vlastností moc nesetkáváme)

hodnoty:

none - výchozí - interpunkční znaménko může být umístěno kdekoliv v elementu (podle textu)
first - znaménko může být mimo začátek prvního řádku
end - znaménko může být mimo konec posledního řádku
allow-end - interpunkční znaménko je vždy mimo element
force-end - interpunkční znaménko je mimo element, pokud je nedostatek místa na řádku
initial - nastaví vlastnost na výchozí hodnotu (tedy na none)
inherit - hodnota se zdědí z rodičovského elementu

Text a písmo - Zrcadlení, zalamování, rozestupy

unicode-bidi

vlastnost [direction] v kombinaci s touto vlastností, řekne prohlížeči, aby text vygeneroval zrcadlově převrácený

hodnoty:

normal - výchozí - text se vykresluje stejně jako před použitím této hodnoty.
embed - vytvoří další úroveň vkládání.
bidi-override - vykreslí text zrcadlově převrácený (u hodnoty rtl k vlastnosti display).
isolate - element se vykreslí podle vlastnosti display, ale jeho obsah nikoliv (je separován od ostatní sourozeneckých elementů)
isolate-override - vykreslí jak element, tak jeho obsah zrcadlově převrácený.
plaintext - text se vykreslí s ohledem na nastavení rodičovského elementu, směr textu je určen za použití pravidel p2 a p3 obousměrného algoritmu Unicode.
initial - nastaví vlastnost na výchozí hodnotu (tedy na normal).
inherit - hodnota se zdědí z rodičovského elementu.

příklad použití v kódu:

.bidi {unicode-bidi: bidi-override; direction: rtl;}

word-wrap

pomocí této vlastnosti nastavujeme zalomování slov v textu, je možné zalomovat celá slova, nebo můžeme povolit rozdělování slov i tam, kde to není povoleno, vlastnost se dědí

hodnoty:

normal – výchozí – zalamuje pouze celá, nebo dělitelná slova
break-word – zalomuje slovy vždy, když je to třeba
inherit – zdědí vlastnost z rodičovského elementu
initial – nastaví vlastnost na výchozí
příklad použití kódu:
.wrap {word-wrap: break-word; width: 100px;}

word-break
pomocí této vlastnosti nastavujeme rozdělování slov v textu, vlastnost je podobná [word-warp], akorát se zde mluví o tzv. CJK textu(čínských, japonských, korejských znacích)
hodnoty:
normal - výchozí - zalamuje slova podle standardních pravidel.
break-all - zalamuje slova na kterémkoli znaku.
keep-all - nezalamuje v CJK textech, pro ostatní texty se chová jako hodnota normal.
break-world - stejná pravidla jako výchozí, pouze s tím rozdílem, že při zalomení nerozděluje slova.
initial - zalamuje slova podle standardních pravidel (stejně jako u výchozí hodnoty).
inherit - hodnota se zdědí z rodičovského elementu.

word-spacing
pomocí této vlastnosti nastavujeme rozestupy mezi jednotlivými slovy v textu
hodnoty:
normal – výchozí – výchozí mezera mezi slovy
hodnota – délka je specifikovaná libovolnou jednotkou (px, em) a může být i záporná
inherit - hodnota se zdědí z rodičovského elementu.

white-space
pomocí této vlastnosti nastavujeme chování mezer v textu, můžeme docílit, aby bylo zachováno několik mezer vedle sebe, nebo aby se text, podle nich zalomoval, ovlivníme i zalomování řádků
hodnoty:
normal - výchozí - více mezer za sebou se nahradí jedinou mezerou. Text se zalamuje pouze pokud je to potřeba a konce řádků jsou ignorovány.
nowrap - více mezer za sebou se nahradí jedinou mezerou. Text se zalomí pouze explicitně značkou
.
pre-text se zobrazí přesně tak, jak je zapsán (chová se jako tag <pre>).
pre-line - více mezer za sebou se nahradí jedinou mezerou. Text se zalomí podle zalomení v původním textu a také, když je to potřeba.
pre-wrap - text se zobrazí tak, jak je zapsán. Zalomí se podle zalomení v původním textu a také, když je to potřeba.
inherit - zdědí vlastnost z rodičovského elementu.

quotes
pomocí této vlastnosti definujeme, jakým způsobem budou uvedeny uvozovky okolo tagu <q>, jako hodnotu uvádíme pár ohraničený uvozovkami, v případě že zapíšeme více párů, budou ovlivňovat vnitřní citace
hodnoty:
text text(„!“ „!“) – výchozí hodnota jsou anglické uvozovky
initial – nastaví vlastnost na výchozí hodotu
inherit - hodnota se zdědí z rodičovského elementu.

Text a písmo - Sloupce, tabulátory, odsazení, zalamování

Vlastnosti sloupců
pomocí těchto vlastností členíme text do sloupců tak, jako například v novinových článcích

column-count
tato vlastnost určuje počet sloupců, do kterých bude text rozdělen

column-fill
tato vlastnost určuje jak bude vyvážen sloupec
hodnoty:
auto – pokud se text vleze do prvního sloupce, ostatní zůstanou prázdné
balance – text bude rovnoměrně rozdělen do všech sloupců

column-gap
tato vlastnost nastavuje šířku mezi sloupci, jako hodnota slouží číslo (např. 200px)

column-rule
tato vlastnost nastaví, co bude mezi sloupci (lze rozdělit na color, style, width)

column-rule-color
tato vlastnost nastaví barvu mezery mezi sloupci

column-rule-style
tato vlastnost nastaví styl mezery mezi sloupci

column-rule-width
tato vlastnost nastaví šířku mezery mezi sloupci

column-span
tato vlastnost určuje danému elementu, zda má být roztažený přes celou šířku rodičovského elementu nebo jen přes 1 sloupec
hodnoty:
none – výchozí – element se vygeneruje v rámci jednoho sloupce
all – element se vygeneruje přes všechny sloupce

column-width
tato vlastnost nastavuje šířku každého sloupce

Příklad použití:

```
.sloupce { column-count: 3;
        column-width: 100px;
        column-rule-width: 10px;
        column-rule-style: dotted;
        column-rule-color: blue;
```

column-gap: 200px;
column-fill: auto;
height: 500px; }

tab-size
tato vlastnost nastavuje velikost znaku tabulátoru, funguje pouze pro elementy [textarea] a [pre] (ostatní elementy převedou tabulátor na jednu mezeru)
hodnoty:
číslo – výchozí hodnota je 8, znamená počet mezer pro jeden znak tabulátoru
číslo jednotka – tuto hodnotu nepodporuje žádný prohlížeč
initial – nastaví vlastnost na výchozí hodnotu
inherit – hodnota se zdědí z rodičovského elementu

text-indent
v této vlastnosti můžeme nastavit odsazení prvního řádku odstavce, odsazení může být i záporné
hodnoty:
délka – výchozí je 0, uvádí se délka odsazení v px, em, pt, %
inherit - vlastnost se zdědí z rodičovského elementu

text-owerflow
v této vlastnosti nastavujeme chování při vytékání textu z HTML elementu
hodnoty:
clip – výchozí – text je zkrácen na nejvyšší možnou délku, která se do elementu vejde
ellipsis – jako clip, akorát je text zakončen trojtečkou
„zástupný text“ – můžeme uvést zástupný text, který se zobrazí místo původního textu, v případě, že se do elementu celý nevejde

Seznamy v CSS3

list-style [list-style: typ pozice obrázek;]
pomocí této vlastnosti nastavujeme styl odrážek seznamů,
vlastnost je shrnující a umožňuje nastavit 3 vlastnosti odrážek: typ, pozici a obrázek:
list-style-type – nastavení typu odrážek
list-style-position – nastavení pozice odrážek
list-style-image – nastavení obrázku odrážek

Nabídka možností v ***list-style-type***:
disc - výchozí - vyplněný kruh
none - bez odrážek
circle - kruhové
square - čtvercové
decimal - číslované
decimal-leading-zero - číslované na 2 místa (před čísla menší než 10 přidá nulu)
lower-alpha - malá písmena latinské abecedy
lower-greek - malá písmena řecké abecedy
lower-latin - malá písmena latinské abecedy
lower-roman - římské číslice malými písmeny
upper-alpha - velká písmena latinské abecedy
upper-greek - velká písmena řecké abecedy
upper-latin - velká písmena latinské abecedy
upper-roman - římské číslice velkými písmeny
armenian - armenské číslice
cjk-ideographic - ideografické číslice
georgian - gregoriánské číslice
hebrew - hebrejské číslice
inherit - zdědí typ odrážek od rodičovského elementu

Nabídka možností v ***list-style-image***:
url – url obrázku s odrážkou
none – výchozí – odrážky nemají obrázek a vykreslují se podle list-style-type
inherit – obrázek odrážek se zdědí z rodičovského elementu

Nabídka možností v ***list-style-position***:
outside – odrážky jsou mimo okraje prvků
inside – odrážky jsou uvnitř prvků

Vzhled a nastavení tabulek v CSS

table-layout
pomocí této vlastnosti nastavujeme algoritmus pro výpočet šířky sloupců v HTML tabulce, šířka sloupců se může nastavovat podle jejich obsahu, nebo může být vypočtena jen podle obsahu tabulky
hodnoty:
auto – výchozí – šířka sloupců se určuje na základě obsahu buněk, sloupec je tedy široký, podle nejdelší hodnoty v něm
fixed – šířka sloupců se určuje podle šířky tabulky
inherit – vlastnost se zdědí od rodičovského elementu

border-spacing
pomocí této vlastnosti nastavujeme rozestupy mezi rámečky okolo buněk HTML tabulky, tato vlastnost funguje pouze pokud má tabulka nastavenou vlastnost [border-collapse], při jejím nastavení můžeme použít buď 2 hodnoty pro vodorovný a svislý rozestup, nebo jednu, která platí pro oba
hodnoty:
délka – rozestup v libovolných jednotkách (px, cm, pt, …)
inherit – vlastnost se zdědí od rodičovského elementu

border-collapse
tato vlastnost umožňuje nastavit styl rámečku tak, aby se vykresloval jako jednoduchá čára, (výchozí nastavení je rámeček okolo každé buňky, ve výsledku je tedy dvojitý), tato vlastnost se nastavuje buňkám, tedy elementům <td> a <th>
hodnoty:
collapse – oraničení je spojeno ze dvou čar sousedních buňek do jedné

separate – ohraničení je vykreslováno okolo každé buňky
inherit – vlastnost se zdědí od rodičovského elementu

caption-side
tato vlastnost nastavuje umístění nadpisu tabulky, nadpis vložíme pomocí tagu <caption>, ten může být zobrazen, buď nad tabulkou nebo pod ní
hodnoty:
top – výchozí – nadpis se vykreslí nad tabulkou
bottom – nadpis se vykreslí pod tabulkou
inherit - vlastnost se zdědí od rodičovského elementu

empty-cells
pomocí této vlastnosti nastavujeme, zda se má skrýt rámeček a pozadí prázdných buněk HTML tabulky
hodnoty:
show – výchozí – prázdné buňky se vykreslují
hide – prázdné buňky se nevykreslují
inherit - vlastnost se zdědí od rodičovského elementu

Okraje prvků: margin a padding v CSS3

tyto okraje nejde vidět, takže jde o způsob, jakým odsadit daný prvek od ostatních prvků, ani jeden okraj se nepočítá do velikosti elementu, takže jeho velikost zvětšují

Box model

Margin (vnější okraj)
pokud cheme všechny okraje nastavit na stejnou velikost, uvedeme jen jednu hodnotu, pokud cheme nastavit okrajům různou velikost, použijeme jako hodnotu 4 čísla (levý, pravý, dolní, horní),
případně můžeme použít některou z rozšíření možností:
margin-left - levý vnější okraj.
margin-right - pravý vnější okraj.
margin-top - horní vnější okraj.
margin-bottom - dolní vnější okraj.

Padding (vnitřní okraj)
pokud cheme všechny okraje nastavit na stejnou velikost, uvedeme jen jednu hodnotu, pokud cheme nastavit okrajům různou velikost, použijeme jako hodnotu 4 čísla (levý, pravý, dolní, horní),
případně můžeme použít některou z rozšíření možností:
padding-left - levý vnitřní okraj.
padding-right - pravý vnitřní okraj.
padding-top - horní vnitřní okraj.
padding-bottom - dolní vnitřní okraj.

Okraje a rámečky v CSS3

border
pomocí vlastnosti border nastavujeme rámeček kolem elementu, samotná vlastnost border má více variant a v základním formátu se nastavuje: ***border: šířka_čáry styl_čáry barva_čáry,*** které můžeme zadat v libovolném pořadí, nebo můžeme nastavit konkrétní hodnoty jednotlivě:

border-width
vlastnost nastavuje šířka rámečku
hodnoty:
thin – tenká rámeček (1px)
medium – výchozí – středně široký rámeček (3px)
thick – tlustý rámeček (5px)
šířka – námi zadaná šířka např. v PX
inherit – hodnota je zděděna z nadřazeného elementu

border-color
vlastnost nastavuje barva čáry rámečku
hodnoty:
barva – specifikujeme barvu
transparent – průhledná barva (rámeček se nevykresluje)
inherit – hodnota je zděděna z nadřazeného elementu elementu

border-style
vlastnost nastavuje styl čáry rámečku
none - bez rámečku
hidden - rámeček se nevykreslí (stejný efekt jako u none)
dotted - tečkovaný rámeček
dashed - čárkovaný rámeček
solid - celistvá plná čára
double - dvojitá čára
groove - 3D rámeček s efektem groove (drážky)
ridge - 3D rámeček s efektem ridge
inset - 3D rámeček s efektem inset (jako stlačené tlačítko)
outset - 3D rámeček s efektem outset (jako nestlačené tlačítko)
inherit - zdědí styl čáry rámečku nadřazeného elementu

Pokud ke každé vlastnosti napíšeme jen jednu hodnotu, je pak brána pro všechny 4 strany, nebo můžeme zapsat jen 2 hodnoty a ty jsou pak brány pro okraje a hořejšek se spodkem, nebo můžeme zapsat 4 hodnoty a ty jsou pak rozebrány jednotlivými stranami, anebo můžeme nastavit vlastnosti jen pro konkrétní stranu:

border-bottom – dolní strana
border-left – levá strany
border-right – pravá strana
border-top – vrchní strana

Pokročileší okraje a rámečky v CSS3

border-radius
tato vlastnost nastavuje poloměr zakulacení okrajů okolo HTML elementu, opět můžeme nastavit 1 hodnotu pro všechny rohy stejnou, nebo jen vybrat které rohy a jak moc (1 – 4 čísla), případně se dají zadat i 2 sady 4 čísel, kde pak každé číslo odpovídá za polovinu rohu

outline
pomocí této vlastnosti nastavujeme obrys okolo HTML elementu, obrys je velmi podobný rámečku, ale na rozdíl od něj, nepřidává elementu na jeho velikosti, syntaxe je stejná jako u vlastnosti [border]

border-image
pomocí této vlastnosti nastavujeme místo barvy rámečku obrázek, prohlížeč vezme námi zvolený obrázek, ořízne jeho okraje a ty použije, v základním nastavení se nastavuje:
border-image: zdroj_obrázku ořízntutí_obrázku šířka_rámečku spadávka typ_opakování
a nebo je můžeme nastavit i jednotlivě:

border-image-source - odkaz na místo, kde je obrázek uložen
border-image-slice - tato vlastnost naám určuje jak se ořezávají rohy obrázku, při jejím zápisu, můžeme kromě číselné hodnoty použít také hodnotu [fill], což způsobí, že oříznutý obrázek se také nastaví jako pozadí obsahu rámečku
border-image-widt - nastavení šířky rámečku
border-image-outset - nastavuje jak daleko se má zobrazit rámeček od obsahu
border-image-repeat - pomocí vlastnosti nastavujeme ´, jak se bude chovat vzor obrázku na stranách
stretch – výchozí – obrázek rámečku se roztáhne kolem elementu
repete – obrázek se opakuje, aby zaplnil celý prostor
round – stejné jako [repete], akorát se malinko deformuje velikost obrázku, tak aby nikde nebyl oříznutý
space – stejné jako [repete], akorátv místě, kde by byl obrázek zkrácen, vloží se mezera

Okraje a rámečky v CSS3 – Animace

box-shadow
při vytváření nefunguje jako okraj elementu, ale jako další element (elementy) umístěný buď od pozadí, nebo popředí, díky tomu se na stránce nechová jako další element, ale pouze jako rozšiřující část už existujícího, jeden element může obsahovat více různých stínů
hodnoty (***box-shadow: 1px 2px 3px 4px red inset***) :
1px - Posunutí podle osy x (doleva nebo doprava)
2px - Posunutí podle osy y (nahoru nebo dolů)
3px - Velikost rozmazání
4px - Velikost stínu
Nastavení barvy
Specifikace umístění (za elementem / v elementu)

Border-block
touto vlastností nastavujeme okraj protilehlých stran okolo HTML elementu, používá se ale zřídka
hodnoty:
border-block-width – šířka rámečku
border-block-style – styl rámečku
border-block-color – barva rámečku
pokud chceme aby byl text vertikálně, přidáme vlastnost ***writing-mode: vertical-lr;***

Rozměry prvků v CSS3

box-sizing
pomocí této vlastnosti nastavujeme, co bude šířka a výška elementu zahrnovat, vlastnost se dědí
hodnoty:
content-box – výchozí – výška a šířka elementu zahrnuje pouze obsah elementu
border-box – výška a šířka elementu zahrnuje obsah vlastnosti [padding] a [border]
initial – nastaví vlastnost na výchozí hodnotu
inherit – hodnota se zdědí z rodičovského elementu

width a **height**
pomocí těchto vlastností nastavujeme výšku a šířku HTML elementu, výšku není třeba hlídat, ale šířka by neměla být širší než rozlišení zobrazovací plochy, a obsah by přetékal do strany, pokud nastavíme pouze jednu hodnotu, druhá se dopočítá automaticky
hodnoty:
auto – výchozí – šířku, či výšku spočítá prohlížeč
délka – šířka či délka je specifikovaná v libovolných délkových jednotkách
% - šířka, či délka, je vyjádřena jako část rodičovského elementu
inherit – hodnota se zdědí z rodičovského elementu

max-width a **max-height**
pomocí těchto vlastností nastavujeme maximální hodnotu šířky a výšky HTML elementu, nastavit maximální šířku je vhodné zejména u obrázků,
hodnoty:
none – výchozí – šířka nebo výška není ničím omezena
délka – šířka či délka je specifikovaná v libovolných délkových jednotkách

% - šířka, či délka, je vyjádřena jako část rodičovského elementu
inherit – hodnota se zdědí z rodičovského elementu

min-width a **min-height**
pomocí těchto vlastností nastavujeme minimální hodnotu šířky a výšky HTML elementu, nastavit maximální šířku je vhodné zejména u obrázků, nebo layoutu, kdy chceme omezit minimální šířku, či výšku stránky
hodnoty:
délka – šířka či délka je specifikovaná v libovolných délkových jednotkách
% - šířka, či délka, je vyjádřena jako část rodičovského elementu
inherit – hodnota se zdědí z rodičovského elementu

Vykreslování, překrývání a svislé zarovnání prvků v CSS3

display
pomocí této vlastnosti nastavujeme jakým způsobem má být HTML element vykreslován
hodnoty:
none - element nebude vykreslen a prohlížeč se bude chovat tak, jako že ve stránce není. Bude tedy přesněji ze stránky vyjmut. Pokud bychom chtěli element pouze skrýt, použijeme vlastnost [visibility], kterou se budeme zabývat níže.
flex - element se vykreslí jako flexbox.
block - element se vykreslí jako blokový. Těmi jsou např. odstavce nebo nadpisy. Bude se roztahovat na nejvyšší možnou šířku a elementy se budou řadit pod sebe.
inline - element se vykreslí jako řádkový. Řádkové elementy se vkládají vedle sebe (na řádek) a výška řádku je určena podle nejvyššího elementu na řádku. Řádkovým elementem je např. .
inline-block - element je vložen na řádek, ale chová se jako blok.
inline-flex - element je vykreslen jako řádkový flexbox.
inline-table - element je vykreslen jako řádková tabulka.
list-item - element je vykreslen jako položka seznamu.
table - element je vykreslen jako tabulka.
table-caption - element je vykreslen jako nadpis tabulky.
table-cell - element je vykreslen jako buňka tabulky. Toto chování se používá zejména kvůli svislému centrování, protože v tabulkách funguje vlastnost vertical-align, na kterou se podíváme níže.
table-column - element je vykreslen jako sloupec tabulky.
table-column-group - element je vykreslen jako skupina sloupců (<colgroup>).
table-footer-group - element je vykreslen jako skupina patiček tabulky.
table-header-group - element je vykreslen jako skupina hlaviček tabulky.
table-row - element je vykreslen jako řádek tabulky.
table-row-group - element je vykreslen jako skupina řádků tabulky.
inherit - vlastnost display bude zděděna od rodičovského elementu.

visibility
pomocí této vlatnosti můžeme skrýt HTML element, skrytý element není vykreslován, ale je ve stránce přítomen a je místo něj vykresleno prázdné místo, pro úplné vajmutí použijeme vlastnost [display]
hodnoty:
visible – výchozí – element je vykreslován
hidden – element není vykreslován, je skytý – na stránce je místo něj vykreslováno prázdné místo
collapse – vlastnost slouží pouze pro tabulkové elementy (výměna dvou čar sousedních buněk za jednu)
inherit – vlastnost je zděděna od rodičovského elementu

z-index
pomocí této vlastnosti nastavujeme hloubku HTML elementu na stránce, pokud se více elementů překrývá, můžeme pomocí této vlastnosti nastavit pořadí, v jakém se mají vykreslovat
hodnoty:
auto – výchozí – pořadí vykreslování prvků je založeno na rodičovském elementu
číslo – číslo určující pořadí, ve kterém se má element vykreslit (čím vyšší číslo, tím výše je prvek)
inherit – vlastnost bude zděděna od rodičovského elementu

vertical-align
pomocí této vlastnosti nastavujeme svislé zarovnání HTML elementu, tato vlastnost funguje pouze u in-line elementů, nebo u tabulek (nelze přes ní svisle centrovat text v bloku)
hodnoty:
délka - posune element o danou délku nahoru oproti výchozí výšce nadřazeného elementu. Pro posun opačným směrem můžeme uvést zápornou hodnotu.
% - posune element dolů o danou část z výšky řádku. Opět můžeme použít zápornou hodnotu.
baseline - umístí element na řádek.
sub - umístí element na pozici dolního indexu.
super - umístí element na pozici horního indexu.
top - umístí element co nejvýše.
text-top - umístí element co nejvýše tak, aby se vrchol fontu dotýkal vrcholu elementu.
middle - vycentruje element svisle.
bottom - umístí obsah na dno elementu (co nejníže).
text-bottom - umístí obsah na dno elementu tak, aby se dna dotýkal konec fontu.
inherit - vlastnost vertical-align se zdědí od rodičovského elementu.

Pozice prvku v CSS3

position
pomocí této vlastnosti nastavujeme, jakým způsobem se má element zobrazovat, resp. jakým způsobem se má určovat jeho pozice

!!!tato lekce je uzamčená!!!

Plovoucí obsah v CSS3

float

pomocí této vlastnosti se nastavuje, zda má být element plovoucí – tyto elementy jsou vyjmuty z klasického rozložení stránky a jsou neplovoucím obsahem obtékány (např. obrázek v textu)

hodnoty:

- ***left*** - element plave (je obtékán) po levé straně a je neplovoucím obsahem obtékán zprava.
- ***right*** - element plave po pravé straně a je neplovoucím obsahem obtékán zleva.
- ***none*** - element není plovoucí. Pokud je tedy vložen v textu, zobrazí se přesně na místě, kde je vložen.
- ***inherit*** - vlastnost float bude zděděna od rodičovského elementu.

clear

pomocí této vlastnosti nastavujeme v které části dokumentu není povolen plovoucí obsah a tím plovoucí obsah zastavíme

hodnoty:

- ***left*** - na levé straně elementu není povolen plovoucí obsah.
- ***right*** - na pravé straně elementu není povolen plovoucí obsah.
- ***both*** - na obou stranách elementu není povolen plovoucí obsah.
- ***none*** - výchozí - plovoucí obsah je povolen na obou stranách elementu, tedy zleva i zprava.
- ***inherit*** - vlastnost clear bude zděděna od rodičovského elementu.

Přetékání obsahu v CSS3

overflow

pomocí této vlastnosti nastavujeme chování elementu v situaci, kdy jeho obsah přeteče jeho rozměry

hodnoty:

- ***visible*** - výchozí - vyteklý obsah je vykreslen mimo hranice elementu
- ***hidden*** - obsah je oříznutý tak, aby se vešel do elementu. Co se nevešlo, není zobrazeno
- ***scroll*** - elementu je přidán scrollbar tak, aby bylo možné zobrazit celý jeho obsah, scrollbar je přidán vždy a na obě strany elementu
- ***auto*** - scrollbar je přidán elementu pouze v případě, že se obsah do elementu nevejde
- ***inherit*** - vlastnost overflow bude zděděna od rodičovského elementu

overflow-x a overflow-y

pomocí této vlastnosti nastavujeme chování elementu v situaci, kdy jeho obsah přeteče jen šířku nebo jen výšku elementu, hodnoty jsou stejné jako u [overflow]

Nastavitelná velikost a ořezávání prvků v CSS3

resize

clip

clip-path

!!!tato lekce je uzamčená!!!

HTML - obsah

Responsivní web – obsah

CSS – obsah

RESPONZIVNÍ WEB

Simulace mobilního telefonu v prohlížeči:
1) Zapnout Vývojářské nástroje
2) Kliknout na možnost zobrazení jako na telefonu

CSS pixel vs fyzický pixel
Do doby, než Apple začal používat na telefonu display s čtyřnásobným množstvím pixelů na stejnou plochu obrazovky, byl pixel konstantní jednotkou, kdy rozlišení obrazovky bylo dáno její velikostí. Nyní je ale při tvorbě webu potřeba brát zřetel, na to, že hodnota pixelu není na všech zařízeních shodná a je třeba toto předem ošetřit.

Viewport
<meta name="viewport" content="width=device-width, initial-scale=1"/>
umožňuje nastavit rozměr „okna“, kterým se na web díváme, vkládá se do hlavičky jako meta element a udává, že nechceme CSS pixel převádět na pixel fyzický, ale chceme poměr 1:1 (1px pak odpovídá asi 1/96 palce jak na počítači, tak na mobilním telefonu)

Desktop first vs. mobile first
Desktop first je přístup, kdy stránku vytvoříme nejprve pro web a následně ji upravíme pro mobilní zařízení, a mobile first je přístup, kdy stránku vytváříme nejprve pro mobily. Tato druhá varianta je výhodná zejména proto, že šetří čas a vyžaduje méně kódu.
(U desktop first musíme několik stylů vyresetovat na výchozí hodnotu, která je roztažení přes celou šířku, nevýhodou je, že u složitějších stylech můžeme na něco zapomenout a desing stránky se nám rozsype, nehledě na to, že píšeme více kódu. U mobile first nic resetovat nemusíme a jen nastavíme nové hodnoty pro desktop a pro mobilní zařízení je necháme ve výchozím nastavení)

Breakpointy a media queries
CSS3 nám umožňuje definovat podmínky, tzv. media *queries*, kdy se daný styl aplikuje na elementy. Pomocí *breakpointů* definujeme, od jaké šířky zařízení se má styl aplikovat.

Viewport (podruhé)
<meta name="viewport" content="width=device-width, viewport-fit=cover">
Abychom vůbec mohli webové stránky na mobilních zařízeních spustit, budeme potřebovat upravit ***viewport***. Ten v podstatě nastaví reálnou šířku displeje, kterou si internetové prohlížeče v mobilních zařízeních rády upravují podle sebe.

Box-sizing
nejčastěji se používá s hodnotou border-box, což mění rozpětí šířky elementu a počítá do něj i *padding* a border, můžeme pak nastavit určitou šířku a tu si pak element udrží, a to i včetně paddingu a borderu.

Mobile first
Vždy se budeme snažit vytvářet samostatné komponenty, které nejsou závislé na konkrétním elementu, či pozici v HTML stránce. V této metodě se často pro šířku elementů využívá kombinace responzivnosti a flexibilnosti, takže nenastavujeme pevné hodnoty (px, em, rem, ..), ale využíváme především procenta.
(Existují také takzvané CSS/front-end frameworky, jako např. Bootstrap, které mohou pomoci ke snadnému responzivnímu designu - hledejte grid systém.)

Responzivní menu
(hned po obsahu nejdůležitější prvek naší stránky)

Idea:
nejprve je dobré si určit, jak má naše menu vypadat, co má umět a jak se má chovat, cílem je vytvořit menu, které se bude dobře zobrazovat jak na desktopech, tak na mobilech, či tabletech.
Menu musí fungovat bez JavaScriptu, který se používá jen jako doplněk, který nám poskytuje různé efekty a manipulaci s menu a měl by přicházet na řadu vždy jako poslední, jelikož je to něco navíc a nemusí být všude přítomen, nebo podporován.

Příprava HTML
Využijeme elementu **nav**, který bude obalovat celé menu, seznam (element **ul**), který bude obsahovat jednotlivé položky (elementy **li**) a poté také odkazy (elementy **a**), které budou obsahovat text položky a budeme s nimi moci odkazovat na webové stránky:

```
<nav class="menu">
    <ul>
        <li><a href="#1">První</a></li>
        <li><a href="#2">Druhý</a></li>
    </ul>
</nav>
```

(protože některé starší verze prohlížečů internet explorer nepodporují HTML5 elementy, je potřeba využít html5shiv, který by měl zajistit, že i tam se nám budou vykreslovat elementy správně)
Stáhneme soubor html5shiv-printshiv.js, vložíme do složky (např. js) a do HTML naimportujemedo hlavičky webu pomocí kódu:

```
<!--[if lt IE 9]>
    <script src="js/html5shiv-printshiv.js"></script>
<![endif]-->
```

Stylizace v CSS
Jako první nastylujeme element nav.menu, který obsahuje celé menu:

```
nav.menu {
    display: block;
    width: 100%;
    background: blue;
}
```

(Od kraje obrazovky je mezera. Ta je ale způsobena defaultním nastavením *body*, což pro naše účely ukázky responzivního menu ničemu nebrání. Pokud chcete mezeru odstranit, **vynulujte padding a margin na body.** Také nenastavujeme výšku menu, to proto, že chceme mít celý seznam zabalen, nezávisle na výšce)

Dále upravíme samotný seznam, nejprve odstraníme odrážky, margin a padding, a položky nastavíme na 100% šířku:

```
nav.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

nav.menu ul li {
    width: 100%;
}
```

Dále upravíme odkazy, kterým dáme bílé písmo, odstraníme podtržení a nastavíme font:

```
nav.menu a {
    display: block;
    color: white;
    text-decoration: none;
    font-family: sans-serif;
}
```

Dále nastylujeme jednotlivé položky, nastavíme výšku řádku (2em) a padding, a ztmavení pozadí při hoveru, čehož docílíme pomocí černé barvy v rgba a průhledností 0.1 (tedy 10%) – toto je výhodné zejména proto, že nejsme závislí na určité barvě, ale jen ji ztmavujeme:

```
nav.menu a {
    line-height: 2em;
    padding: 0 15px;
}

nav.menu a:hover {
    background: rgba(0, 0, 0, 0.1);
}
```

Media queries

zajišťuje spuštění specifických stylů pro určité šířky zařízení, např. pro zobrazení menu na desktopu odchytávání šířky s hranicí 600px:

```
@media (min-width: 600px) {
    /* ... kód ... */
}
```

Abychom rozeznali při tvorbě stránky, na jakém jsme zobrazení, je dobré každému dočasně přiřadit jinou barvu pozadí, a to i u položek:

```
@media (min-width: 600px) {
    nav.menu {
        background: #5cba40;
    }

    nav.menu ul li {
        width: auto;
        background: red;
        display: inline-block;
    }
}
```

Mezery mezi položkami v menu

Formátováním HTML vznikne v menu mezera mezi jednotlivými položkami, tu vymažeme tím, že v celém menu (položka nav.menu) vynulujeme velikost fontu - tím, se nám přestanou tvořit mezery, a poté ale musíme nastavit velikost fontu u odkazů (vkládáme do patřičných položek):

```
nav.menu {
    font-size: 0;
}

nav.menu a {
    font-size: 16px;
}
```

Lazení menu pro desktop:
Pokud požadujeme větší výšku menu (např. 50px), nastavíme v bloku @media výšku řádku pro odkazy menu:

```
nav.menu a {
    line-height: 50px;
}
```

Pro vycentrování menu, nastavíme v bloku @media, seznamu menu vlastnost text-align (je třeba zkontrolovat, zda v klasickém nastavení seznamu menu (nav.menu ul) je uvedena šířka na 100%):

```
nav.menu ul {
    width: 100%;
    text-align: center;
}
```

A v tuto chvíli můžeme i smazat pro nás zvýrazňující pozadí položek (li)

Podtržení

Pokud chceme vylepšit zvýraznění :hover, tak abychom zbytečně nemuseli bořit strukturu menu, můžeme využít pseudo element ::after a nastylovat ho. Nejprve musíme nastavit vlastnost conect, která ho propojí s :hover, po té vlastnost display, kde vybereme block, a dále šířku a výšku elementu, barvu nastavíme na 15% průhlednosti černé (pozor sečte se již s nastavenou barvou :hoveru, takže bude tmavší), a nakonec nastavíme pozici bottom a left na 0 a využívat budeme absolutní pozicování, abychom nepřidávali další místo a nezvětšovali menu:

```
nav.menu a:hover::after {
    content: "";
    display: block;
    width: 100%;
    height: 3px;
    background: rgba(0, 0, 0, 0.15);
    bottom: 0;
    left: 0;
    position: absolute;}
```

A jelikož u ::after jsme nastavili position na absolute, musíme se někde stanovit zarážka, což se provede nastavením position: relative na některý rodičovských prvků, a protože ::after nám přidá linku tlačítka, chceme tuto zarážku mít relativně k tlačítku:

```
nav.menu a {
    line-height: 50px;
    position: relative;}
```

Tímto jsme nastavili menu pro desktop, kdybychom chtěli nastavit menu pro mobil, nebo pro tablet, stačí přidat více bloků, které ale bude vhodné ohraničit z obou stran (max-width, min-width)

Responzivní obrázky

Aby se nám nestávalo, že při zobrazení stránky na mobilu se nám na display nevejdou, stačí všem obrázkům (elementům <img>) nastavit maximální šířku na 100% šířky jejich rodičovského elementu, výšku pak dopočítáváme automaticky:

```
img {
    max-width: 100%;
    height: auto;}
```

Rozlišení obrázků

uživatelé procházejí web na zařízeních která mají rozlišení 1x, 2x, 3x, 4x velikosti normálního pixelu, takže obrázek který vypadá dobře na ploše obrazovky, nemusí vypadat dobře na mobilu s větším rozlišením, kde se může jevit rozmazaně.

Ne příliš dobrým řešením je použití velkého obrázku, který na počítači bychom viděly zmenšený a v mobilu v plné velikosti:

```
img {
    width: 500px;}

@media only screen and (max-width: 768px) {
    img {
        width: initial;}}
```

(nevýhodou tohoto řešení je, že se někdy stahují příliš velké obrázky – data – než by bylo potřeba)

Lepším řešením je upravit obrázek do více různých velikostí, které poté poskytneme prohlížeči pomocí speciálního atributu. Prohlížeč potom sám vyhodnotí, který obrázek bude nejlepší použít. Například, máme tři varianty obrázku:

1) img-1x.jpg, šířka 500px
2) img-2x.jpg, šířka 1000px
3) img-3x.jpg, šířka 2000px

A do tagu <img> vložíme následující kód:

```
<img src="img-2x.jpg" srcset="img-1x.jpg 500w, img-2x.jpg 1000w, img-3x.jpg 2000w">
```

S tím, že písmeno ‚w' značí pixely a atribut srcset pak zde nabízí tyto 3 varianty se kterými může prohlížeč pracovat, jeho synaxe je ***srcset="ADRESA šířkaObrázkuVPxw"***

Alternativou je namísto rozlišení (w) použít přímo označení typu rozlišení zařízení (1x, 2x. 3x, 4x):

```
<img srcset="img1x.jpg 1x, img2x.jpg 2x">
```

Pokud uvedeme jen tyto dvě, říkáme prohlížeči, že pro všechny další má použít obrázek s rozlišením 2x. S tím, že různé typy oddělujeme čárkou a atribut src (pro cestu k obrázku) je pak dobré použit pro případ, že prohlížeč srcset nepodporuje.

Dalším řešením, které se ale taky moc nepoužívá **je Art redirection** kde kromě rozlišení dáme ještě prohlížeči za úkol, aby na základě šířky pro obrázek udělal "něco" s velikostí obrázku nebo s obrázkem samotným. Např. do šířky 768px se bude obrázek vykreslovat na 100% šířky displeje, zatímco nad touto hranicí to bude pouze 50%. Použijeme k tomu atribut *sizes*:

```
<img srcset="maly.png 600w, stredni.png 1024w, velky.png 1600w" sizes="(max-width: 768px) 100vw, 50vw">
```

Přepínání obsahu obrázku

Při vkládání obrázku do stránky nemusíme vždy chtít zobrazit obrázek v jeho plné šířce (nebo výšce). Např. na desktopu si můžeme dovolit zobrazit celou postavu i s širokou loukou za ní, zatímco na malém telefonu bude lepší zobrazit pouze detail, tedy výřez postavy, kvůli které obrázek na stránku vkládáme. Pro řešení tohoto problému se používá HTML párový tag *<picture>*, do kterého vložíme tag *<img>*, jako defaoultní obrázek, který se načte, a dále přidáme různé alternativy obrázku pomocí tagu `<source>` s atributem `srcset` a podmínky, kdy se mají zobrazit, zapsané jako atribut `media`:

```
<picture>
    <source media="(min-width: 1024px)" srcset="velky.jpg">
    <source media="(min-width: 512px)" srcset="stredni.jpg">
    <img src="maly.jpg" alt="…">
</picture>
```

Prohlížeč potom vezme první vyhovující řešení, proto je nutné správné řazení, pomocí toho pak můžete docílit přesné změny obrázku na základě šířky displaje

V čem se `<picture>` liší od `<img srcset sizes>`? Příklad, který uvádíme výše, je poměrně zjednodušený. Museli bychom v něm ještě ošetřit různé typy displejů (1x, 2x, ...). To za nás dělá u `<img srcset sizes>` prohlížeč automaticky. Oproti tomu zde si pomocí atributu `media` můžeme sami nastavit *breakpointy* mezi různými obrázky na základě šířky displeje. V `<img srcset sizes>` to vybírá prohlížeč sám na základě nastavení v `sizes`.

Flexbox - Tvorba moderních layoutů

CSS 3 nám dopřálo nové pracování s kontejnery obsahu. Tomuto pozicování se říká **flexbox**. Přineslo mnoho výhod a ulehčilo práci zejména při vytváření layoutů založených na sloupečcích. Dnes se už CSS Flexbox příliš nevyužívá a namísto něj je využíván Bootstrap Grid, nebo Flexbox Grid.

Výhody:
Částečně nahrazuje vytváření sloupečků pomocí float, kde jsme museli vše pracně umísťovat a dopočítávat v %, či jiné jednotce, oproti tomu u flexboxu můžeme nastavovat šířku sloupců pomocí čísel (např. hlavní má 4 a vedlejší 1), délka v pixelech se pak sama vypočítá.
Můžeme také měnit pořadí sloupečků podle sebe, což se může hodit pro vytváření responzivního layoutu, kdy některé sloupce umístěné okolo hlavního textu, můžeme na mobilu dát pod sebe.

Použití:
Pro použití musíme obalit svoje sloupečky nějakým elementem, kterému nastavíme vlastnost displaye na hodnotu flex a po té sloupce stylujeme v CSS vlastnostmi flexboxu:

Nastavení pro třídu flexbox, která obaluje všechny odstavce, kterých se týká:

```
#flexbox {
        display: flex; /*aktivace zobrazení flexboxu*/
        max-width: 960px; /*určení maximální šířky*/
        margin: auto; /*automatické nastavení odsazení objektu od kraje*/
}
```

Nastavení pro jednotlivé sloupce:

```
#flexbox div {
        background-color: #DDDDDD; /*nastavení barvy pozadí*/
}
```

Nastavení pro boční sloupce (třídy menu a třídy panel):

```
#menu, #panel {
        flex: 1; /*nastavení poměru velikosti sloupce*/
}
```

Nastavení pro hlavní sloupec (třídy obsah):

```
#obsah {
        flex: 3; /*nastavení poměru velikosti sloupce*/
}
```

Nastavení přehození sloupců v případěže je okno prohlížeče užší než 700px:

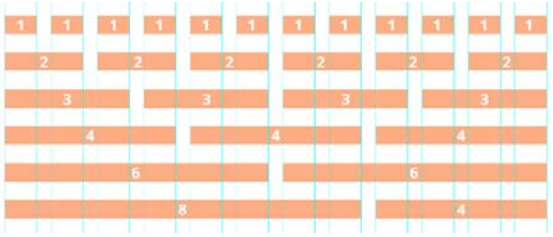
```
@media screen and (max-width: 700px) {
        #obsah {
                order: -1; /*nastavení pořadí*/
        }
}
```

Grid systémy

slouží na rozložení obsahu na stránce webu, dříve se na to používaly tagy <table>, <tr> a <td>, ale vzhledem k nesématické povaze, neresponzivní tabulky se od tohoto způsobu opustilo a pro layout se začaly využívat Grid systémy, to jsou předpřipravené třídy, které přiřazujeme například elementům a ty následně získávají určitou šířku, systém obvykle funguje tak, že rodičovský element rozdělí na 12 sloupců, které je pak možno dělit:

- Element přes celou šířku (velikost 12 sloupců)
- Element přes polovinu šířky (velikost 6 sloupců)
- Element přes třetinu šířky (velikost 4 sloupců)
- Element přes čtvrtinu šířky (velikost 3 sloupců)
- Element přes 1/6 šířky (velikost 2 sloupců)
- Element přes 1/12 šířky (velikost 1 sloupce)

Můžeme nastavovat i násobky těchto hodnot, např. vložit do kontejneru element, který zabere 2/3 obsahu a tudíž bude nastavený na šířku 8 sloupců apod. Naopak nejsme schopni vložit např. 5 elementů vedle sebe, můžeme vložit buď 4 nebo 6. Toto omezení je daní za estetickou pravidelnost danou mřížkou:



Pokud chceme grid systém na svých stránkách používat, máme víceméně dvě možnosti:

- Flexbox Grid od Kristofera Josepha
- CSS framework Bootstrap

Flexbox Grid

pro aktivaci máme dvě možnosti:

1) Stáhnout soubor do svého adresáře a přidat do svého HTML dokumentu link s cestou:
<link rel="stylesheet" href="flexboxgrid-6.3.1/css/flexboxgrid.min.css" type="text/css" />

2) Použít link s cestou na stránky Flexboxu:
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/flexboxgrid/6.3.1/flexboxgrid.min.css" type="text/css" />

pro použití flexboxu grid je jako první nutné definovat řádek, kterým obalíme všechny sloupce, (ten v příkladu definujeme jako *** *** se třídou ***.row,*** sloupce se po té definují také jako *** ,*** kterému přidáme třídu ***.col-X-Y***, kde za ***X*** a ***Y*** dosadíme:

x – může nabývat hodnotu od nejmenšího xs(telefony), sm, md až lg – tímto udáváme od jaké velikosti zařízení začíná styl platit

y – nabývám hodnot 1 až 12 a udává, kolik sloupců bude element zabírat, s tím, že všechny elementy na jednom řádku mohou zabírat maximálně 12 sloupců

příklad:

```
<div class="row">
    <div class="col-xs-3">Čtvrtina</div>
    <div class="col-xs-3">Čtvrtina</div>
    <div class="col-xs-6">Polovina</div>
</div>
```

Breakpointy
ve flexbox grid máme 4 druhy breakpointů:

xs*-Y* - element bude široký Y sloupců na velmi malých zařízeních a větších (tedy na všech)

sm*-Y* - element bude široký Y sloupců na malých zařízeních a větších

md*-Y* - element bude široký Y sloupců na středně velkých zařízeních a větších

lg*-Y* - element bude široký Y sloupců na velkých zařízeních

příklad:

```
<div class="row">
    <div class="col-xs-12 col-md-3">Čtvrtina</div>
    <div class="col-xs-12 col-md-3">Čtvrtina</div>
    <div class="col-xs-12 col-md-6">Polovina</div>
</div>
```

v příkladu je nastaveno: na malých zařízeních (xs a sm) zobrazi řádek přes všechny sloupce a na větších (mg, lg) je rozdělí poměrem 3 – 3 – 6

! Při pouřívání flexboxu bychom neměli používat margin, sloupce mají totiž přesně definovanou šířku a pokud ji narušíme vlastností margin, celková šířka sloupce může přetéct a rozhodit nám obsah stránky, vlastnost margin tedy můžeme používat je na elementy vložené do flexboxu, nebo použít padding !

Automatická šířka
pokud chceme všechny sloupce stejné, nebo jen vyplnit zbytek řádku, pak k tomu můžeme použít automatickou šířku, ta se nastavuje tak, že za .col-x už nenapíšeme žádné číslo počtu sloupců

Změna pořadí
oproti Bootstrapu, kde jde určit přesné místo každému elementu nám flexbox grid dovoluje určit pouze první a poslední element (to vychází pravděpodobně z toho, že uprostřed je hlavní stať, která u větších monitorů bývá zpravidla obklopena bočními texty)

Offsety
občas nějaký sloupec nevyužijeme a chceme místo něj ponechat prázdné místo, to se dělá třídou .col-X-offset-Y, přičemž Y je číslo, které definuje počet sloupců o které se má daný sloupec posunout doprava

Pozicování
flexbox grid vychází z flexboxu, proto využívá některé jeho vlastnosti, jako například pozicování elementů ve volném místě, můžeme pozicovat horizontálně, nebo vertikálně a nebo podle předdefinovaných stylů, hlavní rozdíl je, že třídy se nepíší na sloupec, ale na řádek (.row), definují totiž chování celého řádku

Horizontální pozicování
používají se k němu třídy: ***.start-X***, ***.center-X***, ***end-X***, kde X je breakpoint, tyto třídy pak určují odkud se bude element pozicovat, zda od začátku, ze středu, nebo z konce, každý pak vložený sloupec do stejného řádku se bude pozicovat stejně

Vertikální pozicování
používají se k němu třídy: ***.top-X***, ***.middle-X***, ***.bottom-X***, kde X je breakpoint, tato třídy pak určují odkud se bude element pozicovat, zda od zhora, na střed a nebo ze zdola, každý pak vložený sloupec do stejného řádku se bude pozicovat stejně

Předdefinované pozicování
vycházejí z flexboxu a používají se k němu tyto třídy: ***.around-X*** a ***.between-X,*** kde X je breakpoint, tyto třídy obě používají vertikální pozicování .midle-X a rozložení horizontální by se dalo označit u *around* jako stejně prostoru z obou stran všech sloupců a u *between* je to krajní skloupce na kraji a mezi nima a středním sloupcem (sloupci) stejně volného prostoru

CSS Grid Layout

vznikl pro složitější – dvoudimenzionální rozložení stránky, jeho npřednosti jsou ve snadné manipulaci s elementy, možnost dál stylovat elementy (zarovnání textu atd) a také možnost překrývat elementy, řadit je nebo z nich vytvářet kontejnery a docílit takzvané vložené mřížky (nesting gird)

Příprava:
V HTML zapíšeme jednotlivé položky, objektu přiřadíme třídu, které pak v CSS nastavíme zobrazení grid:
• ***display: grid;*** – definuje vlastnost rozložení položek do gridu
• ***display: inline-grid****;* – řádkové rozložení položek

Rozložení:
grid-template-columns: definuje šířku jednotlivých sloupců (**1fr 1fr 1fr;** pro 3 stejné sloupce)
(šířku můžeme zadat buď pomocí pixelů (není dobré pro responzivní web) nebo podle jednotek **fr**, které reprezentují frakci z dostupného prostoru kontejneru, stránku lze rozdělit na 12 jednotek přičemž poměr jejich šířky může být libovolný)

namísto ***(1fr 1fr 1fr)*** můžeme použít i funkci pro opakování ***repete(),*** např:
grid-template-columns: repeat(3, 1fr);

grid-template-rows: definuje šířku jednotlivých řádek
(pokud obsah přeteče (má více než 12 jednotek) uspořádají se nám elementy do řad)

Řadám i sloupcům pak můžeme nastavit automatickou velikost pomocí vlastností:
• ***grid-auto-rows***
• ***grid-auto-columns***

Funkce minmax()
pomocí této funkce můžeme například nastavit minimální a maximální výšku řady:
grid-auto-rows: minmax(100px, auto);

Další vlastnosti CSS gridu:

gap
neboli gutter, nastavuje mezeru mezi jednotlivými řadami a sloupci,
dá se dále specifikovat pomocí vlastností:
• *row-gap*
• *column-gap*

grid-row a ***grid-column***
tyto vlastnosti přidáváme jednotlivým elementům uvnitř kontejneru a díky nim můžeme umístit daný element na libovolné místo v gridu, využíváme k tomu tzv grid lines, což jsou čáry, které tvoří mřížku grid a číslování se provádí z leva doprava a ze zhora dolu
další možností zápisu je napsatt: *grid-row: 1/3*, kdy tímto zápisem, říkáme že element v mřížce zabere prostor od první po třetí čáru (tedy dvě okénka), takový zápis je v podstatě spojením vlastností:
• *grid-row(nebo colum)-start*
• *grid-row(nebo colum)-end*

Nesting gird
umožňuje z elementu vytvářet kontejner na další podpoložky a vytváříme je tak, že elementu vložíme v CSS vlastnost *display: gird;*
další vlastnosti, jako třeba ohraničení nebo barvu pozadí, nastavujeme ve třídě ***.nested***

Z-index
v CSS se čte kód zezhora dolu a například při překrývání obsahu, položky níže překrývají obsah položek uvedených nad nimi, pokud bychom chtěli ovlivnit toto pořadí použijeme vlastnost z-index, kde pořadí zapisujem pomocí čísla:
• *z-index: 2;*

Bootstrap framework

(framework = je ucelená knihovna, nebo sada knihoven, která dává dohromady celé řešení webových stránek) Bootstrap je tedy velký soubor mnoha CSS & JS kódu

Hlavní výhody:
• ***responzivita*** - Bootstrap styly jsou dokonale přizpůsobené pro mobilní zařízení. Je tedy 100% responzivní.
• ***mobile-first*** - Framework byl ve verzi 3 kompletně přepsán, aby podporoval mobile-first přístup. Jeho kód je tak kompaktnější a podporuje dobré praktiky.
• ***flat design*** - Co dnes není flat? Váš web bude vypadat svěží a in. A když se trend změní, můžete si být jistí, že Bootstrap na to zareaguje. A pokud se vám flat přeci jen nelíbí, můžete si stáhnout jakékoli jiné ze schémat, která jsou pro Bootstrap dostupná (viz dále).
• ***grid*** - Obsahuje podobný dvanáctisloupcový grid systém, který jsme si uvedli minule.
• ***je zdarma*** - Bootstrap je zadarmo i pro komerční účely.

Propojení s HTML:
• buť si ho můžeme celý stáhnout a naimportovat
• nebo můžeme použít odkaz na cloudové úložiště z těchto stránek:
https://getbootstrap.com/docs/4.5/getting-started/introduction/
a kód po té vložíme do hlavičky:
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
(použití linku na cloud má tu výhodu, že prohlížeč uživatele, by tuto knihovnu už mohl mít načtenou v paměti a tím se zrychlí i načítání stránek)

Stejně jako u Flex-box Grid jsou i u Bootstrapu stejně nazvané třídy .col.
Syntaxe je col-X-Y kde:
• ***X*** může nabývat hodnot od nejmenšího sm (telefony), md, lg až xl. Tímto udáváme, od jaké velikosti zařízení začíná styl platit.
• ***Y*** nabývá hodnot 1 až 12 a udává, kolik sloupců bude element zabírat. Všechny elementy v jednom řádku nemohou zabírat dohromady více než 12 sloupců.
(Definujme nyní třídy **od** středně velkých zařízení. Cokoli menšího bude mít vždy šířku 12 sloupců, což je výchozí šířka. Na malých zařízeních se tedy sloupce poskládají pod sebe.)

Co bychom měli vědět:
• elementy sloupců mají kolem sebe ***padding***, tomu se v kontextu Bootstrap grid systému říká ***gutter***
• můžeme jej případně odstranit a to přidáním třídy ***.no-gutters*** elementu se třídou ***.row***
• ***guttery*** jsou implementované pomocí vlastnosti ***padding*** po obou stranách každého sloupce a vlastnosti ***margin*** (záporný) na kontejneru, díky tomu mají sloupce mezery mezi sebou, ale ty krajní mezeru mezi sebou a okolím kontejneru nevytvoří
• sloupcům v ***gridu*** nesmíme přidávat ***margin***, protože by elementy s ním byly dohromady širší než kontejner a zalomily by se
• nic nám ovšem nebrání vložit do sloupce další element a ***margin*** nastavit až tomuto elementu
• aby ***grid*** systém fungoval, musí být vždy všechny přímé podelementy řádků sloupce
• sloupce mohou obsahovat opět řádky, gridy tedy můžeme vkládat do sebe
• některé HTML elementy nelze jako flex kontejnery zatím použít, to jsou např. *<button>* nebo *<fieldset>*

Automatická šířka:
pokud nějakému sloupci nastavíme šířku pomocí třídy ve tvaru ***.col-{breakpoint}-auto***, jeho šířka se nastaví tak, aby se do sloupce vešel jeho obsah, a zbytek šířky bude rozdělen mezi zbývající sloupce podle pravidel definovaných v gridu

Vynucené zalomení řádku:
Pokud bychom z nějakého důvodu potřebovali zalomit řádek dříve, než jej ve skutečnosti ukončíme, můžeme za tímto účelem do gridu vložit sloupec s třídou ***.w-100***

Zarovnání a změna pořadí:
pokud potřebujeme sloupce v gridu nějak zarovnat, existuje na to několik předpřipravených stylů. Elementům se třídou ***.row*** přiřadíme třídy jako ***.align-items-center***, ***.justify-content-center*** nebo **.align-self-end**, stejně tak můžeme měnit skutečné pořadí sloupců v řádku bez ohledu na jejich pořadí v kódu, stačí jim přiřadit jednu z tříd začínající na ***.order-X***, kde X nabývá hodnot 1 - 12

Offsety:
občas nějaký sloupec nevyužijeme a chceme místo něj ponechat volné místo, abychom se vyhnuli jeho zbytečné deklaraci v kódu, nemusíme prázdný sloupec vkládat vůbec a následující sloupec a kus posunout, máme k tomu k dispozici třídy ve formátu ***.offset-{číslo}***, případně ***.offset-{breakpoint}-číslo***, přičemž číslo určuje počet sloupců (z 12), o které se má daný sloupec posunout doprava

Automatický margin:
tato vlastnost má spíše minimální využití, nicméně i přesto by se někdy mohla hodit, pokud sloupci nastavíme třídu ***.ml-auto*** nebo ***.mr-auto***, posune se zbytek sloupců nalevo nebo napravo

Responzivní tabulky

abychom mohli napsat responzivní web, musíme se naučit ho rozložit a nakódovat jednotlivé části, zde to budou tabulky, u kterých si ukážeme 3 způsoby respozivní implementace:

1) Horizontální skrolování
2) Skrolování s pevné ukotveným záhlavím
3) Stocking (neboli stohování)

Horizontální skrolování
Horizontální skrolování (posouvání) je nejlehčím a zároveň nejčastěji používaným způsobem. Cílem je udržet řádek dat pohromadě. Uživatel si může libovolně horizontálně posouvat v tabulce podle vlastní potřeby. Tento způsob se používá pro tabulky s menším počtem řádků i sloupců.

HTML: Kód HTML je obyčejná tabulka s třídami.
CSS: Kromě odsazení a okrajů je pro nás důležitá třída ***.scrollable_table,*** kterou nastylujeme na ***overflow-x: auto;.*** Tato vlastnost nám umožňuje posunutí v tabulce po ose x (horizontálně).

Skrolování s pevně ukotveným záhlavím
Opět se jedná o horizontální skrolování, ale tentokrát s fixním záhlavím. Po posunutí tabulky záhlaví zůstane vždy statické. Tato varianta je vhodná ve chvíli, kdy tabulka obsahuje větší počet sloupců nebo řádků.

HTML: Vytvoříme si taky třídu ***.header,*** která nám seskupí naši statickou hlavičku
CSS: Zbytek úprav v CSS *(viz. tahák)*

Stocking
V češtině se to dá přeložit jako Stohování. Jedná se o velmi šikovný způsob, který nám umožní data přeskládat do tzv. stohu. Nejedná se už o tabulku jako takovou, ale spíše o seskupené sloty. Využívá se v tabulkách, která obsahuje mnoho sloupců i řádků.

HTML: V HTML je nejprve potřeba vložit do hlavičky meta tag pro viewport:
<meta name="viewport" content="width=device-width, initial-scale=1">
(bez tohoto tagu se tabulka sice vykreslí, ale nebude rosponzivní)
CSS: Zbytek úprav v CSS *(viz. tahák)*

Další způsoby
Atypickým způsobem může být tabulka, která se při určitém zmenšení ztratí a následně vyskočí odkaz na tabulku, která se otevře v dalším okně. V případech mobilních telefonů by se tabulka zobrazila uživateli v *landscape* zobrazení.

Vlastní CSS pro tisknutí stránek

pokud na své stránky vkládáme obsah, který by se mohl uživatelovi hodit vytisknout, je dobré takovéto situace ošetřit tak, aby uživatel zbytečně netiskl text a obrázky, které nechce (např. reklamy, menu, atd.)

Máme 2 možnosti, kam psát CSSkód, který se aktivuje pouze pokud stránku budeme chtít vytisknout:
1) psát pomocí tzv. **@media query** přímo do hlavního CSS souboru
2) vytvoření druhého samostatného souboru **print.css**

@media query
slouží pro optimalizaci stránky pro různá zařízení (desktop, tablet, mobil) a můžeme v ní definovat např. velikost fontu, barvu textu, nebo vnější okraj při tisku:

```
@media print {
        p {font-family: Georgia, serif; font-size: 12pt; }
        h1 {font-family: Georgia, serif; font-size: 30px; }
}
```

(a vzhledem k tomu, že tiskne se pouze na velkých zařízení, nemusíme tento kód přepisovat do verze pro mobilní telefon)

print.css
u této varianty nejprve musíme propojit nový CSS s HTML dokumentem pomocí tagu *<link>:*

```
<link rel="stylesheet" href="print.css" type="text/css" media="print" />
```

(použití atributu media na hodnotě "print" nám zajistí, že toto CSS se aktivuje pouze při tisku)

• vlastnosti písma:
abychom dokument co nejlépe připravili pro tisk na tiskárně, použijeme pouze ***černou barvu*** a velikost písma definujeme v ***pt*** (body) a nejčastěji volíme nějaký ***patkový font*** (serif)

• audiovizuální prvky:
tyto prvky je lepší do tisku nezahrnovat, toho dosílíme přiřazením ***ID*** k prvku, nebo ***třídy*** k skupině prvků a vlastností:

```
#sidebar { display: none; }
```

reklami se dají většinou zaměřit přes ***iframe***
obrázky, jsou-li pro tisk nezbytné, je dobré alespoň zmenšit – nastavit jim maximální šířku:

```
img { max-width: 400px; }
```

• odkazy:
je-li pro nás důležité, aby čtenář dostal třeba ke zdroji, nebo námi doporučovaným stránkám, můžeme je zviditelnit:

```
a:after { content: "( " attr(href) " ) "; }
```

• odkazy ve footeru:
footer obsahuje často třeba odkaz na kontaktní údaje, sociální sítě nebo informace o stránce jako

takové. I tyto odkazy je tedy potřeba zviditelnit. Kromě toho je ale důležité, aby byly odkazy co nejčitelnější. Naším záměrem je, aby se daly snadno zapamatovat, a nebo alespoň snadno opsat. Pomocníkem tu může být například doplněk do prohlížeče Google Chrome nebo celá řada webů, které tuto službu (zkracování odkazů) poskytují zdarma. Častými příklady jsou Bit.ly, 1url.cz a další.

• velikost papíru

nastavuje se pomocí querty @page, takto můžete nastavit např. okraje stránky:

@page { margin: 2cm }

Definujeme-li margin takto, aplikuje se na všechny strany. Pokud chceme upravit velikost okrajů na jednotlivých stranách zvlášť, můžeme použít vlastnosti ***margin-left*** a ***margin-right***. Šikovné jsou také pseudotřídy ***:first*** (první strana dokumentu), ***:left*** a ***:right*** (použijeme, pokud chceme ošetřit oboustranný tisk).

Page-break

Není praktické ani estetické rozdělovat nadpis a odstavce, které se k němu vážou, stejně tak nechceme rozdělovat (krátké) tabulky a některé seznamy (třeba výčet ingrediencí, které jsou potřeba na vaření). Ošetříme to pomocí ***page-breaks***, resp. pomocí vlastností ***page-break-before*** (oddělí stránky před elementem) a ***page-break-after*** (oddělí stránky za elementem).
Nastavit jim můžeme následující hodnoty:

- ***auto*** - defaultní hodnota, nezpůsobí oddělení stránky.
- ***always*** - *page-break* se aktivuje vždy (například nadpis h1 a další text začne vždy na nové straně).
- ***avoid*** - předchází zalomení stránky před/za elementem.
- ***left/right*** - způsobí jedno nebo dvě zalomení. Můžeme znát z knih, kdy nová kapitola nezačíná jen na nové stránce, ale například vždy pouze na pravé straně.

Nastavení pro zalomení, pokud začínáme novou kapitolu, a vyhnout se zakončování stránek nadpisy:

h1 { page-break-before: always; }
h1, h2, h3, h4, h5 { page-break-after: avoid; }

Chceme-li se vyhnout například rozdělení tabulky, použijeme vlastnost ***page-break-inside:***

table, figure { page-break-inside: avoid; }

Widows a Orphans

Vlastnost ***widows*** určuje minimální počet řádků, které musí mít odstavec (blok textu) na začátku stránky. Je dobré to ošetřit, abychom předešli například rozdělení pětiřádkového odstavce na čtyři řádky na jedné straně a jeden (nebo jen třeba půl řádku) na straně druhé.
Nastavíme tedy vlastnost ***widows: 3;***, takže nová stránka bude začínat vždy alespoň třířádkovým odstavcem.
Velmi podobně funguje vlastnost ***orphans***, která kontroluje minimální počet řádků na konci stránky.