Udite 3

jarek.kligl7

June 2024

1 Introduction

2 Služba WWW

World Wide Web Nejznámější ("nejpoužívanější") služba poskytovaná v síti Internet. Umožňuje prohlížení, ukládání a odkazování dokumentů.
ve skutecnosti je to DNS protoze lide se nedostavaji na webove stranky pomoci ip ale domenovych jmen ergo pri kazdem vyuziti WWW se vyuzije DNS a DNS ma vice vyuziti jeste

Sir Timothy "Tim" John Berners-Lee (*8. června 1955 Londýn)

- Konec 80. let informačního systému ENQUIRE pro **CERN**
  navrhl system sdilenych dokumentu ktere propojil odkazama, a stoho vznikl Internet
  Tim je autorem prvniho prohlizece a prvni webove stranky
- Vznik jazyka HTML, protokolu HTTP, URL a prvního prohlížeče
- Počátek informační revoluce
- 1991 → služba WWW
- "Otec webu" - on to zalozil a dodnes se o to stara
- ma Turnigovu cenu

URL
Identifikace souborů pomocí URL (Uniform Resource Locator)
Zjednodušená podoba URL:

```
protokol://domenove-jmeno:port/umisteni-na-serveru
```

Například:

`http://www.inf.upol.cz/adresa/index.html`

Speciální význam `index.html` a další. - z historickejch duvodu , kdyz nacitame adresar tak se nacte soubor index.html

Absolutní vs. relativní URL.
Absolutni - cele umisteny adresy
Relativni - Vzhledem k umisteni konkretniho souboru na konkretnim serveru

WWW = HTTP(S).
kdyz byl Internet navrhovany tak nebylo pocitano s bezpecnosti proto byl pak vyroben HTTPS ktery pridal sifrovani

HTTP neřeší bezpečnost.

URL, URI ,URN
Často zaměňováno:
URI = identifikátor zdroje, například ISBN.

URL = umístění (lokace) zdroje, například adresa. - "zpusob jak se k tomu dostaneme"

URL a URI lze u služby WWW zaměnit. - URI je ve webovem prostredi v podstate ta adresa bez toho protokolu
URN popisuje jméno zdroje (striktní formát). - formalni format URN:namespace s timto se setkavame nejmene casto

protokol HTTP
Verze HTTP/1.1 - 70 procent webu, muzu pouzivat jenom jeden dotaz zaroven
HTTP/2 - zbilejch 30 procent, muzu posilat vice dotazu zaraz (rychlejsi)
HTTP/3 (HTTP over QUIC) - snazi se utlacit protokoly transportni vrstvy svim - vymysli to GOOGlE , ale je nespolehlivej a nespojovej
Pokud pouzivate GOOGLe chrome tak pouzivate HTTP/3

Bezstavový protokol = neuchovává informace
ani jedna strana nevi o histrorii → cookies - nastroj jak na klientskej pc ulozit nejake textove informace, aby protokol byl mozny pracovat s minulosti spojeni jde to vypnout a uz se to moc nepouziva
Nezamenovat s EUro Cookies

Metody GET, POST
GET:

```
GET /pro-zajemce-o-studium HTTP/1.1
Host: www.inf.upol.cz
Connection: keep-alive
User-Agent: Mozilla/5.0
Accept-Language: cs
```

GET je uchovavan v historii prohlizece a v cashe je pristupny
omezeni na velikost spravy, delka URL adresy je omezena
Klient posila pozadavek a server mu odpovi:

```
HTTP/1.1 200 OK
Date: Mon, 09 Sep 2019 09:14:40 GMT
Server: Apache/2.4.10 (Debian)
Content-Type: text/html; charset=UTF-8

data
```

POST: je tam misto GET POST, libovolne data, libobolne delky, neuklada se historie

3 Struktura webové stránky

www stránka, HTML stránka, web stránka, web, stránka

Dříve dokument v síti Internet, dnes nelze přesně vymezit - treba i Visual Studio code je webova stranka

Tvořena řadou webových technologií

pro nas: webova stranka ktere jsou umisteny na webovych servrech

Umístěna na webovém serveru, ale není nutné

- "běžný" počítač v síti
- muzeme pouzti localhosst - coz je loopbackova adresa
  pro testovaci uceli, nebo na lokalni siti
- specializovaný software → webový server
- bezne web-server v ramci virtualizaci - webovy server nepotrebuje veliky vykon , lepsi je pouzit vic malich pc (rozdrobit to virtualizaci)

Komunikace: klient-server architektura

Základní rozdělení:

- statická
- dynamická

Dalsi deleni:

- webove stranky
- webove aplikace

Dalsi deleni z hlediska UI
Treba OnePage stranky

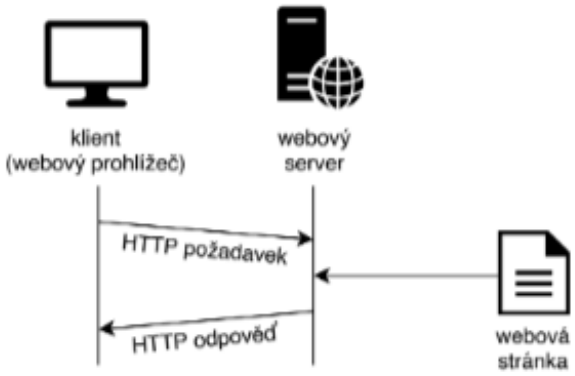

Figure 1: Enter Caption

Staticka stranka Staticky obsah ktery dal neni jinak menen
Uzivatel pokud chce stranku zobrazit posle http pozadavek
Server celou stranku vezme a posle ji v odpovedi v Http odpovedi
kazdy soubor je dorucovan v separatnich pozadavcich a separatnich odpovedi
(HTML - jeden pozadavek, CSS - druhy, JS - treti ...)
HTTP 1.1 - to dela sekvence dostane 1 pozadavek posle 1 ...
HTTP 2 = jsou poslany vsechny pozadavky a dostaneme zpatky vsecky (konsolidace)
http 2 ma pokrocilejsi kompresy a posila je ve binarni formne ne v textove jak HTTP 1.1

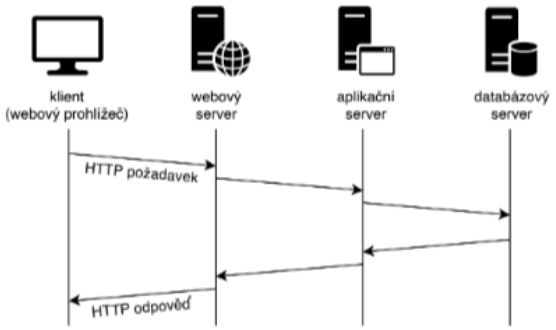

Figure 2: Enter Caption

dynamická stranka

Webova stranka je sestavena podle pozadavku
Priklad: Internetovy obchod - data jsou ulozeny v databasy
je tam nejaka obsluha ktera vezme data s database a posle je
Aplikacni server - scripty na strane serveru - kontaktuje databasi, vygeneruje stranku da do ni obsah ktery vzal z database a posle ji klientovy

Webove technologie
Obecně jakékoliv technologie spojené se službou WWW - zobrazeni , vytvareni

Dnes → prostředky pro tvorbu webových stránek
Zahrnuje enormní množství technologií
Celá řada klasifikací:

- základní
- pokročilé
- klientské
  na strane klienta (webovy prohlizec)
- serverové
  server-side
- ...

Etapy vývoje webu: Web 1.0, Web 2.0, Web 3.0 (web3)
Základní klientské (front-end)

- **HTML**, AMP, XML → **popis sémantiky**
- HTML - Bez toho se neobejde zadna webova stranka!!
- AMP - puvodne GOOGLE , vyrazne zrychleni
- CSS → vizualizace stránky
- JavaScript, WebAssembly → **skripty na straně klienta**

Základní serverové (back-end) (webovy server, aplikacni server, db server)

- webový server (Apache, IIS, nginx, Node.js, ...)
- skriptovací jazyk (PHP, .NET, Python, JavaScript, **LISP** ...) → skripty na straně serveru, cast co realizuje **Aplikacni server**
  generovani stranky je prace s textem
- databázový server (MySQL, MariaDB, MongoDB, ...)
  misto kde jsou ulozena data, lze to redukovat na textovy soubor

Jsou i balickove programy
(LAMP - Apache + PHP + MariaDB)

Front-end, back-end, full-stack vývojář/programátor
full-stack je jich cim dal mene, je to tezke
vyvojar - clovek co technologie pouziva
programator - clovek co technologie vytvari

prohlížeč	jádro
Google Chrome	Blink (založeno na WebKit)
Safari	WebKit
Microsoft Edge	Blink (od verze 79)
Firefox	Gecko
Opera	Blink
IE	Trindent

Figure 3: Enter Caption

Webovy prohlizec

Uživatelské hledisko → Nezajímavé
"renderuje" webové stránky. Renderování webové stránky = složitý proces
Rozdíl mezi prohlížeči → **Renderovací jádro** program napsany vetsinou v jazyce C
Zobrazi tu stranku uzivately ve webovim prohlizeci
rozdily v prohlizeci jsou velike, i kdyz maji stejne jadro
kazdy prohlizec to jadro modifikuje
Boj mezi jadrami - rust webovy technologii

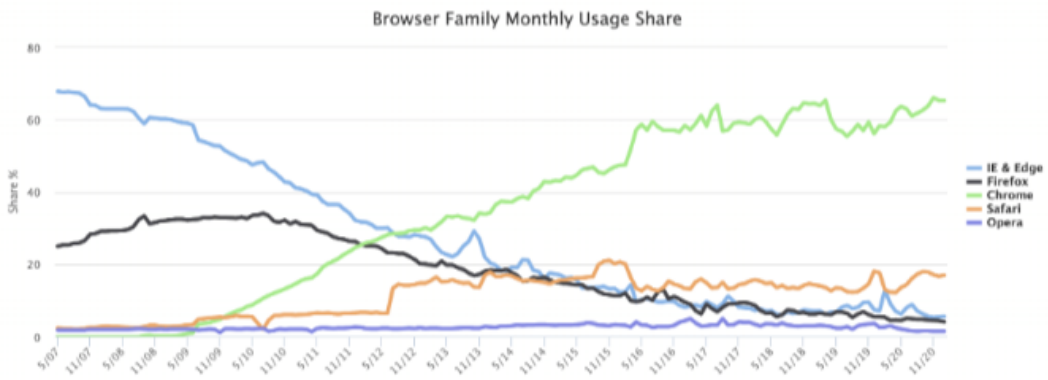

Figure 4: Enter Caption

Statistiky pouzivani Masivni narust prohlizece Google Chrome, dnes nejpouzivanejsi
ale berme to s nadhledem
v nejake zemi to muze byt jinak
V USA je safary vyrazne dominantnejsi nez ve vychodnich zemich
Kdyz delame webove technologie , zoohledneme jaky prohlizec je pouzivanej v tom kruhu ktere chceme aby ano nevim je kurva 6 rano a nejsem schopny doformulovat zaver tehle vety ale msg te vety snad jde pochopit

Webove Vyhledavace
"Privetiva" cesta k webové stránce
drive byly webove katalogy, pak byly ty katalogy moc velke
YAHOO - prisel s indexaci

Indexace = zanesení stránky do databáze vyhledávače
Proces indexace

- Stránku navštíví robot (web crawler)
  program ktery zbira informace o webovych strankach, musi byt na ni odkaz od nekud jinud
- Stáhne její obsah
- Stažený obsah je analyzován (ohodnocení stránky, klíčová slova, různé metriky)
  kazdy vyhledavac to ma skryte , vy se ze stranka je ohodnocovana treba podle rychlosti (cibulka.net by dostal spatne ohodnoceni), klicove slova - nejcastejsi slova atd...
- Výsledek uložen do databáze

Optimalizace pro vyhledávače (SEO)

optimalizace pro webove vyhledavace, jak co nejlepe zprijemnit robotovy at se tam dostane

da se to zneuzivat (linkove farmy)

za to aby ta webova stranka byla dobre vyhledavatelna se da vydelat dost dobre penezii

Odbocka: Google Základem je algoritmus PageRank:

- Důležitost stránky (0–10)
  kazdre strance prirazje dulezitost na zaklade odkazatelnosti jinich stranek (pokud na nasi stranku odkazuje kvalitni stranka tak nase je pak taky kvalitnejsi
  kvantyta vs kvalita
  na webovou stranku musi vest odkazy a musi z ni vest odkazy aby mela dobre hodnoceni (i pro SEO)
- Core algoritmus = PageRank + neznámý algoritmus (zvažující řadu kriterií)
  google ma zhruba 250 kriterii pricemz 1/3 je znama nebo odhadovana

Na výsledek jsou aplikovány filtry:

- **Panda** (2011): Filtrace stránek s nízkou kvalitou
- **Penguin** (2012): Filtrace stránek s mnoha odkazy
- **Hummingbird** (2013): Filtrace stránek obsahující spam (analýza textu), částečně nahrazeno RankBrain (2015) a BERT (2019)
  RankBrain a BERT jsou neuronove site
- **E-A-T** (2014): Autoritativní domény
- 2016–2017: Orientace na mobilní vyhledávání
- 2020: Pandemie
- **Page Experience** (2021), aka UX: Uživatelská zkušenost

skryte - protoze vyvojari webovych stranek se muzou chovat neeticky (a taky chovaji) aby se jim zobrazovala jejich webova stranka
spam slova - treba pornografie

Core algoritmus a každý filtr je průběžně aktualizován. Jiné vyhledávače mají podobnou architekturu.
rozdil hlavne ve filtraci

Pristupnost
Uživatelé (nejen) s handicapem (treba zrak):
treba i poruchy barvocitu, nebo sluchu

- ctecky webovych stranek
- Přístup = zpřístupnění webové stránky těmto uživatelům
- Dáno různými standardy
- Státní instituce musí (s výjimkami) splňovat.

- tohle se da prenest na jakykoliv software co budeme delat
- statni instituce tohle musi splnovat (smernice EU uz to knam prosakuje taky)

Webove standarty

Konsorcium (organizace):

- **W3C** (World Wide Web Consortium), `http://www.w3.org/`
  stalo na zacatku sluzbu WWW, reditel je samotny Sir Tim bernes LEE
  mezinarodni bezne do toho promlouvaji velke spolecnosti jako Apple a Google
- Konsorcium **WHATWG** (Web Hypertext Application Technology Working Group), `https://whatwg.org/`

Specifikace (standardy):

- Složitý životní cyklus specifikace (`https://www.w3.org/2020/Process-20200915/`)
- aha ted budeme delat vec X zalozi se pracovni skupina a ta vyplodi *working draft*
  takovych standartu je hodne
  nachazi se tu hodne dlouhou dobu
  Az to dospeje do posledni faze recommendation
  pak uz se na nich nepokracuje max vyvyji nova verze

Důležité:

- Rozpracovaný standard → status *working draft*
- Hotový standard → status *recommendation* (doporučení)

Webové prohlížeče implementují tyto standardy.
Jadro implementuje standarty

Realita: ...(standarty jsou jenom doporuceni) , a nebo webove prohlizece implementuji working draft , nebo to implementuje "svym zpusobem"

Ukázka: `https://caniuse.com/#feat=webp`
mapuje ruzne technologie a jejich pouziti ve ruznych prohlizecich

Novější standardy nahrazují předchozí.

4 Jazyk HTML

Nový způsob standardizace:
Vyvoj HTML5 trval 10 let - velky milnik , takhle to dal nejde (trvat to tak dlouho) + patentove zalezitosti
neni mozny aby vysla webova technologie ktera je patentovana - nastal by konec internetu
28. ledna 2021 WHATWG *reviewer draft* → status *recommendation* W3C
tak nikdy to nedokoncime a budeme to porad vyvyjet, W3C dava doporuceni jenom konkretni verzy live standartu, velka sance ze uz nebudou cisla
5.2 html posledni vydany
ted mame jenom verzy Live HTML a konkretni verze jazyka uz nikdy nebude
Textový soubor s příponou .html (není nutné)

Značky přiřazují význam obsahu **(udávají sémantiku**).

Značky HTML neříkají nic o vzhledu (vizualizaci) obsahu (poněkud nepřesně).
Webová stránka obsahuje:

- Obsah
- Odkazy
- Značky jazyka HTML

Q: Je sémantika nutná? A: ANO!
Sémantika má vliv na celou řadu klíčových atributů:

- SEO
- Přístupnost, pro ty ctecky
- Strojové zpracování, semanticky web

Znacky nerikaji nic o vzhledu , **ale o obsahu**!!! Priklad: Noviny (pekny priklad webovy stranky)
kdyz odstranim semantiku tak mi zbydou jenom data
nevim ze toto je autor ze toto je text ze toto je nazev clanku
Jazyk html rekne toto je nadpis toto je odstavec toto je hlavni sekce toto je navigace... **SEMANTIKA** ano uz to rekl 50 tak v jedne minute
Je tam vychozi vizualizace

Kategorie značek (`https://www.w3.org/TR/html52/dom.html#kinds-of-content`):

- Metadata
- Flow
- Sectioning
- Heading

- Phrasing
- Embedded
- Interactive

Zastaralé dělení značek (Nemelo by se vubec pouzivat protoze rikaji i neco o vzhledu a to nechceme):

- Reflektuje výchozí zobrazení značky
- Blokové – jejich použití způsobí vytvoření bloku
- Řádkové (inline) – nepřekročí rozsah řádku

4.1 Syntax HTML

(Normalni) element →element, drive parovy element
Syntaxe:

```
<znacka [atribut_1="hodnota_1" ... atribut_n="hodnota_n"]>
obsah elementu
</znacka>
```

Příklad:

```
<a href="https://cs.wikipedia.org/Douglas_Adams" rel="external" title="více o Adamsovi">
Douglas Adams
</a>
```

Prazdny Element
Prázdný element, dříve nepárový element
Syntaxe:

```
<znacka [atribut_1="hodnota_1" ... atribut_n=" hodnota_n"]>
<znacka [atribut_1="hodnota_1" ... atribut_n=" hodnota_n"] />
```

V HTML 5 již není znak / povinný (ani zakázaný)
Příklad:

```
<img src="pruvodce.jpg" alt="Obálka průvodce" width="200">
```

Při renderování jsou ignorovány:

- Komentáře `<!-- 42 -->`
- Vícenásobné mezery - zobrazi se jenom jedena
- Zalomení řádků - zobrazi se jenom jedeno
- Tabulátory - zobrazi se jenom jeden
- Neznámé značky - fylozofie prohlizecu kdyz nevi tak preskoci

Terminologie:

- Element HTML = součást jazyka HTML → značky, například element `p`, tedy `<p></p>`
- Element (webové stránky) = element HTML a jeho obsah
- HTML značka = jedna konkrétní značka, například `</p>`

Atributy
Povinné, doporučené, volitelné, globální
pro konkretni element je to ruzne

Několik způsobů zápisu:

```
<input type="text" disabled>
<input type="text" value=yes>
<input type="text" value='no'>
<input type="text" value="yes or no">
```

disabled - je boolovsky atribut pokud je uveden tak ano pokud neni tak ne
nemusime pouzivat "" " kdyz je tam jenom jedno slovo

Zanorovani elementu
Vytváří vztah potomek-rodič. Značky musí být uzavírány v **pořadí LIFO** (Last In, First Out).

```
<!-- správné zanoření -->
<p> ... <em>trilogie v pěti dílech</em> ... </p>

<!-- chybné zanoření -->
<p> ... <em>trilogie v pěti dílech ... </p></em>
```

teb druhy zpusob je nevalidni ale jinak se zobrazi normalne
nastane problem az treba u CSS

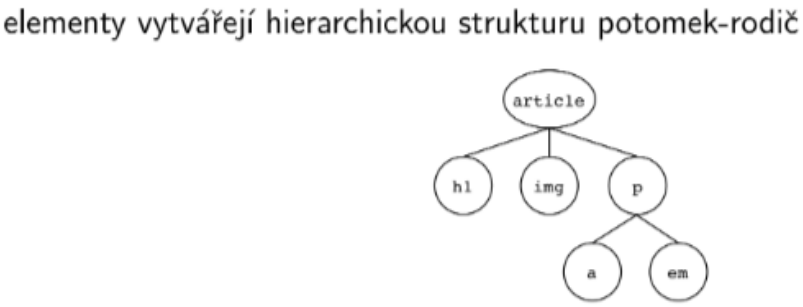

Figure 5: Enter Caption

Vztah: potomek & rodic

```
<article>
    <h1>Kniha Stopařův průvodce Galaxií</h1>
        <img src="pruvdce.jpg" alt="Obálka průvodce">
        <p>...
            <a href="odkaz" rel="external" title="více o Adamsovi">
                Douglas Adams
            </a>
            ...
            <em>trilogie v pěti dílech</em>.
        </p>
</article>
```

- h1 je potomek article
- p je rodicem prvku em & a

Primi rodic X obecny
dualne potomek

Pojmenovani elementu

Pojmenování **jedinečným názvem** = atribut `id`

```
<div id="content">
    <article>
        ...
    </article>
</div>
```

jenom jeden element muze mit stejny id

Pojmenování **opakujícím se názvem** = atribut `class`

```
<div class="navigation active">
    <article>
        ...
    </article>
</div>
```

Názvy by měly být smysluplné. - existuje cela rada doporucenych postopu muze mit vice nazvu (oddeluje se mezerou)

Zakladni struktura webove stranky

```
<!DOCTYPE html>
<html lang="cs">
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
    </body>
</html>
```

Analogie prazdneho papiru
$<!DOCTYPE>$ - udava standart jazyka ve kterem ta stranka je vytvarena pokud napisu jenom html - tak myslim tu nejnovejsi co ma status recomendation
Bez ni muzou byt problemy
$< HTML >$ - Vymezuje celou webovou stranku, **lang je nepoviny ale sile doporuceny**, mely by jsme to pouzit
$< HEAD >$ - hlavicka cast co obsahuje metainformace urcene pro prohlizec, uzivatel vidi jenom obsah elementu $< TITILE >$
MetaInformace - informace pro webove prohlizece , charset, autor, ... , nezobrazuji se
taky se tam davaji externi odkazy , na js , css atd...
$< BODY >$ - cast co urzivatel skutencne vidi v tom okne (obsah webove stranky)

Nadpisy

`h1`, . . . , `h6`

```
<h1>nadpis</h1>
<h6>nadpis</h6>
```

Nadpisy urcuji semantiku ne vzheld
Postupně klesající **důležitost**. Používáním nadpisů vytváříme ***outline***!! (osnovu) webu. Výskyt nadpisů musí **být postupný**.
velky vyznam pro prohlizece a pro pristupnost!!

Strukturalni elementy
Strukturální elementy
Sectioning elements
Elementy určující (základní) strukturu stránky.
emoznuje popsat jednotlive casti stranky

Strukturální elementy:

- `header` - hlavicka stranky, ne head!!!
- `nav` - pro vymezeni nejdulezitejsi navigace
  jenom pro tu jednu hlavni, duvod pristupnost
- `footer` - pro paticku
- `article` - pro uceleny uplny obsah
- `section` - sekce ktera obsahuje nejaky casti (politika , sport)
- `main` - pro vymezeni nejhlavnejsiho obsahu , to o cem ta webova stranka skutecne je
- `aside` - dodatecny obsah, velmi casto se pouziva pro "side bar", ale nemusi byt

Na strance muze byt jenom 1 main
Treba v elemtnu Article muze byt hlavicka a paticka
Kontextova zavyslost - vyznam elemtu je dan kontextem, mistem kde se pouzije
Restartují outline webu. - s vyjimkou main
uvnitr strukturalniho elementu se pouzivaji nadpisy znova od H2
Dobre pouzivani jazyka HTML pouzit jenom 1 H1
Obsah H1 je dulezity pro SEO, zmateme SEO pokud pouzijeme vice H1

Odkaz
Charakteristika pro webove stranky (hypertextovy odkaz)
Element `a` a atributy:

- `href`, URL adresa destinace (relativní, absolutní)
- `title`, popisek (důležitý význam)
  Globalni atribute , kdyz najedu kurzorem mysi na nejaky prvke , tak se zobrazii tooltip
  Velmi dulezity vyznam, uzivatel by mel vedet kam ten odkaz vede kdyz na to kliknu
  Velky vyznam v pristupnosti

Kotva (odkaz na element):

```
<a href="#stoparuv-pruvodce">Stopařův průvodce Galaxií</a>
```

Adresovani skrz element ID

Posunuti stranky na konkretni element aby byl zarovnan v levem hornim rohu, je to skokove presunuti zadna plynula animace
muze dojit k lehkemu spozdeni

...

```
<article id="stoparuv-pruvodce">
    ...
</article>
```

Obrazek
Prázdný element `img`, různé formáty (WebP, JPG, PNG, GIF, ...)

Atributy:

- `src`, URL adresa obrázku (relativní, absolutní)
  Analogicky jako atribute href
- `alt`, popisek obrázku (má vliv na SEO a v případě nedostupnosti obrázku, popřípadě pro čtečky stránek)
  pouzije se pokud se obrazek nepodari nacist a pristupnost
  clovek co nevidi ten obrazek nevidi
  ALT je poviny kdyz ho tam nenapiseme dopustime se sintakticke chyby

Obrázky by měly být vždy optimalizovány pro web.
uzivatel musi treba stahovat 12mb fotografii spatne, musi se zmensit velikost obrazku
aby se webpage nacitala co nejrychleji
Příklad:

```
<img src="smiley.gif" alt="Smiley face" height="42" width="42">
```

Atributy pro vysku a sirku obrazku, ridi se v tom pomer obrazku
s pohledu fotografa nejhorsi co muzete udelat je zdeformovat pomer stran

Text v HTML
Jazyk Html obsahuje kolem 250 znacek
`p`, odstavec, žádný (skutečný) text by neměl být mimo odstavec
`strong`, zvýrazněný text, **sémantická důležitost**
nepjekna vlasnost je ze se ten element zobrazi tucne ale to je **VYCHOZY VIZUALIZACE**
dulezetost i pro prohlizec `em`, zdůrazněný text
kdyz neco cteme tak zvysime hlas `small`, krátký dodatečný komentář (např. autorská práva)

`i`, `b`, `u`, jiný typ nebo jiná výslovnost (např. jiný jazyk, technický pojem), tučný (bez ovlivnění sémantiky), stylistika (píše se jinak, text obsahující záměrnou chybu)
do verze jazyka html 5 se pouzivaly pro vizualici dnes nepripustne!!
krome elemnentu B ten je opravdu jen na vizulizaci, pri sazbe matematiky a konkretne maticich
U - jina stylistika
`br`, prázdný element, řádkový zlom (nese to semantickou informaci)
`hr`, **tématická změna**, (nikoliv vodorovná čára!)

Seznamy
`ul`, seznam s odrážkami
`ol`, číslovaný seznam - mohne ovlivnit pomoci atributu nebo CSS
`li`, položka seznamu
`dl` - hodnota
`dt` (term), termín - klic
`dd` (obecně data ve tvaru název-hodnota, např. otázky a odpovědi)
seznam definic - sezban typu klic hodnota

Dalsi elementy
`audio`
`video`
`canvas` (2D kreslící plátno) - prostor do ktereho je mozne pomoci JS kreslit a vyzulizovat vektorovou grafiku, revoluce ve vizualizaci, umoznil nahradu flesh, HTML5 hry
Prisli s HTML5
Prehravac Audia a videa prehrava Webovy prohlizec
Ne kazdy prohlizec dokaze kazdy format prehrat
proto se tam dava vice ve formatu a kdyz prohlizec nezna tak to preskoci
prohlizec vzdy vybere to prvni

```
<audio controls>
    <source src="horse.ogg" type="audio/ogg">
    <source src="horse.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
</audio>

<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
    Your browser does not support the video tag.
</video>
```

Formulare

Elementy pro tvorbu formulářů a jejich prvky vytváří obsah, **nepřiřazují sémantiku** (až na několik výjimek). Lze programovat vlastní prvky (HTML + CSS + JS).
Formulare generuji obsah
INPUT - atributte type - urcuje typ inputu
pomoci struktury formulare je mozne ovlivnit "validaci" vztupu
Nutna ale validace na server-side (da se upravit ve zdrojovem code

```
<form method="post" action="zpracuj-formular.php">
    <fieldset>
        <legend>Osobní informace</legend>

        <!-- Pro oddělení je použito <p>, lze i jinak -->
        <p>
            <label for="jmeno">Jméno:</label>
            <input type="text" name="jmeno" id="jmeno">
        </p>

    </fieldset>
</form>
```

Atributy metod bud POST nebo GET
Action url adresa skriptu nebo obsluhy toho formulare
name - identifikace na strane serveru pod tim jmenem

Tabulky

Zobrazení tabulkových dat či dat tabulkové povahy v webovém prostředí: využití tabulek pro layout. - dnes neni potreba delat
dnes Vyhradne pro zobrazeni tabulkovych dat

Syntax:

- `table`
- `thead`
- `tbody`
- `tfoot`
- `tr` - radek
- `th` - bunka hlavicky tabulky
- `td` - bunka tabulky
- `caption`

Attributes:

- `rowspan`: Roztažení buňky přes řádky.
- `colspan`: Roztažení buňky přes sloupce.

Udeltat dobre tabulku aby byla responzivni tak neni jednoduche

```
<table>
    <caption>Popis tabulky</caption>
    <thead>
        <tr>
            <th scope="col">Hlavička sloupce 1</th>
            <th scope="col">Hlavička sloupce 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Rádek 1, Buňka 1</td>
            <td>Rádek 1, Buňka 2</td>
        </tr>
        <tr>
            <td>Rádek 2, Buňka 1</td>
            <td>Rádek 2, Buňka 2</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="2">Paticka tabulky</td>
        </tr>
    </tfoot>
</table}
```

Elementy div a span

genericke konterjnery **(bez Semantickeho vyznamu)**
historie znacky div - naduzivani casta chyba
DIV - orgie
Stranka se da napsat jen pomoci A a DIV ale neeee nedelat
dulezite pro visualizaci

HTML entity

Stránky by měly být v UTF-8 kódování (existují výjimky). Možnost zapisovat speciální znaky přímo v HTML.

```
&nazev_entity;
```

```
&copy;
&raquo;
&amp;
  <!-- Nejdůležitější no breaking space -->
```

Validni HTML

Validní = syntakticky (a částečně sémanticky) korektní HTML kód vzhledem **k aktuálnímu DOCTYPE**. Nevalidní HTML může být renderováno nečekaným způsobem, **což má velký vliv na SEO**.

Automatická validace: `http://validator.w3.org/`
od W3C konsorcia

HTML kód by měl být vždy validní!

Sémantická správnost → nelze ověřit.
aby to slo musel by tu stranku ten validator pochopit tak daleko jeste nejsme

Nespravne pouzivani HTML

Jazyk HTML dává "volnost"

- Sémanticky nesprávný kód se zobrazí špatně.
- Syntakticky nesprávný kód se zobrazí špatně.
- Špatné HTML může vést k **horší přístupnosti a SEO**.
- Složitá HTML struktura zpomaluje renderování stránky.
- **Horší vizualizace.**

Semantyka souvisi s praktycnosti te webove stranky

5 Jazyk CSS

HTML určuje sémantiku.

Sémantika částečně určuje vzhled (např. nadpis).

Dodatečná specifikace (ignorována v většině prohlížečů) určuje výchozí vzhled každého elementu.

Vizualizace stránky = Cascading Style Sheets (CSS), textový soubor s CSS pravidly, určující vzhled elementů.

Obecný princip: oddělení obsahu (HTML) od jeho prezentace (CSS).
oddeleni semantiky od vizualizace

verze CSS
formálně nic takového neexistuje
verze CSS 2.1 Odebraly status redomendation a dali tomu status Warking Draft
verze CSS 3.0 - byla rozdelena na ty moduly specifikace CSS = sada modulů (specifikací)
každý modul má svoji verzi označenou jako level
rozšíření předchozí funkcionality = vyšší číslo (např. selektory level 3, level 4)
zcela nová funkcionalita (FlexBox level 1, FlexBox level 2)
→ potřeba caniuse.com
CSS - je uplne prvni standart , standart na HTML byl az pozdeji

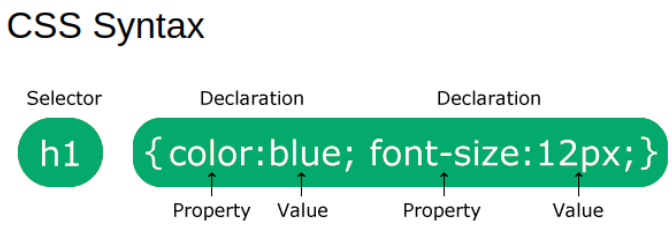

Figure 6: Enter Caption

Syntaxe CSS pravidel

Terminology:

- CSS pravidlo (CSS rule) - to cele
- Selektor (Selector) - vybyra co
- Deklarační blok (Declaration block)
- Deklarace (Declaration)
- Vlastnost (Property)
- Hodnota (Value)

Example CSS rule structure:

```
selektor {
    vlastnost_1: hodnota_x;
    ...
    vlastnost_n: hodnota_z;
}
```

Poslední středník je volitelný (je lepší jej psát), a bílé znaky neovlivňují syntaxi.

Komentáře:

```
/* a */
```

CSS je třeba udržovat přehledné a organizované!
Formátování CSS pravidel, strukturování pravidel
Hodne se vracim ke starym pravidel
vsechno ma globalni platnost

Zacleneni CSS do HTML

1. Externí soubor (přípona `.css`):

```
<head>
    <link rel="stylesheet" href="file.css">
</head>
```

   Kdyz je stranka rendorvana

   (a) prvni je stazen soubor HTML
   (b) podiva se do hlavicky
   (c) pozada pres HTTP protocol o jejich stazeni
   (d) zacne se soubor cist a vyzualizovat

   \+ Udrzeni rozdeleni mezi CSS a HTML

2. Přímo v HTML stránce (embedded), element `style`:

```
<style>
    img {border: 1px solid black;}
</style>
```

   Obvykle v head ale neni nutne
   cim pozdeji je code style uvedeny tim pozdeji se to vyzualizuje Poznámka:
   Lze použít i `@import url(file.css)`, ale je pomalejší a je lepší ho nepoužívat.
   \+ mene HTTP requestu

3. Inline definice pomocí globálního atributu `style`:

```
<img src="dark.jpg" alt="Dark Vader" style="border: 1px solid black">
```

   pise se osah declaracniho bloku jako hodnota atributu style
   velice nepjekny zpusob, cesta do pekla, mala udrzitelnost, duvod je manipulace JS

Otázka: Jaký je mezi nimi rozdíl?
Odpověď: Rozdíl mezi nimi je zásadní.

5.1 Konstrukce selektoru

Pomoci Jmena elementu

Zápis jednoduchého CSS pravidla:

```
h1 {
    color: gold;
}
```

Zápis zřetězením více selektorů (and):

```
h1, h2 {
    color: gold;
}
```

ID & Class

```
/* elementy které mají class="nadpis" */
.nadpis {
    color: gold;
}

/* element který má id="clanek" */
#clanek {
    color: gold;
}

/* elementy em které mají class="nadpis" */
em.nadpis {
    color: gold;
}
```

Konstrukce selektorů: Kontext elementu

Kontext elementu = místo potomek-rodič hierarchie
zápis pomocí mezery
Tzv **DOM** struktura
obecny rodic potomek

HTML kód:

```
<!--
HTML kód
-->
<h1>Stopařův <span>průvodce</span> po <span class="podnadpis">galaxii</span></h1>
<p>je prvním dílem pentalogie označované jako <span>trilogie v pěti dílech</span></p>
```

CSS kód:

```
/*
CSS kód
*/
h1 span {
    color: red;
}
h1 span.podnadpis {
    color: blue;
}
```

Primi potomek (rodic)
Zápis znakem ¿

```
<!-- HTML kód -->
<article>
  <h1>Nadpis</h1>
  <p>A</p>
  <p>B</p>
  <p>C</p>
  <section>
    <p>D</p>
    ...
  </section>
</article>

/* CSS kód */
article > p {
  color: gold;
}
```

Vybere jen prvky `<p>`, které jsou přímými potomky `<article>`.

Pseudo-element a pseudo-trida
pseudo-element → část stránky, která není určena žádným elementem (první písmeno, první řádek), ale chováme se k ní jako k elementu, zápis s ::
pseudo-třída → elementy identifikované na základě jejich pozice v HTML nebo vlastnosti, zápis : (jakoby určené třídou)
ma rozdil od pseudo-elemntu to urcuje elementy ktere tam vzkutku jsou

```
/* pseudo-element, výběr prvního řádku elementu p */
p::first-line {
  color: gold;
}

/* pseudo-třída, výběr prvního potomka elementu p */
p:first-child {
```

```
  color: gold;
}

/* pseudo-třída, výběr elementů a na kterých je kurzor */
a:hover {
  color: gold;
}
```

Dedicnost
Klicovy koncept v CSS

```
<!-- HTML kód -->
<p>Stopařův průvodce Galaxií, jehož autorem je Angličan Douglas Adams, je prvním dílem stejn
<em>trilogie v pěti dílech</em>.</p>


/* CSS kód */

p {
  color: gold;
}
```

potomci přejímají vlastnosti svých rodičů
mění se p i em
ne vše se dědí
vynucená dědičnost: inherit
dědičnost usnadňuje práci je klicova, lepsi udrzitelnost

5.1.1 Kaskada

Hrozne dulezita vec

více zdrojů CSS, muzu jich linknout vice **elementy mohou být vybírány různými selektory**

```
<!-- HTML kód -->
<article id="prvni-clanek"> <!-- id by se nemělo používat -->
<h2 class="nadpis"></h2>
</article>


/* CSS kód */
h2 {...}
article h2 {...}
.nadpis {...}
#prvni-clanek h2 {...}
```

kolize CSS pravidel
řešení kaskáda → uvedené způsoby nejsou ekvivalentní

deklarace různých vlastností → spojené
deklarace **stejných vlastností → pravidlo (princip) kaskády**:

priorita: Prohlizec < Uzivatel < Autor

1. **specifičnost**, více specifický selektor má přednost před méně specifickým selektorem (dáno standardem, začátečníci o něm mnohdy ani nevědí)
2. **pořadí**, později uvedené pravidlo má přednost
3. **Dulezitost**, lze obejít pomocí vlastnosti `!important`(ideálně nepoužívat! protoze ty 2 kroky predtim staci)

Vypocet specificnosti
Specifičnost = číslo ve tvaru a b c (lexikální uspořádání):

- a: počet id v selektoru
- b: počet class, výběru atributů a pseudo-tříd v selektoru
- c: počet elementů a pseudo-elementů v selektoru

Prosta sumace tech v selektroru

Inline styl má větší specifičnost než jakýkoliv jiný autorský styl (pro jednoduchost 1 a b c).

Použití `!important` obejde specifičnosti, největší specifičnost (pro jednoduchost 1 0 a b c).

Univerzální selektor (*): 0 0 0 0 0.

Kaskada Priklad

```
<!-- HTML kód -->
<div class="druhy prvni"></div>

/* CSS kód */
.prvni {
    color: blue;
}
.druhy {
    color: red;
}
```

1. specificnost obojiho je stejna (0 0 0 1 0)
2. rozhodne poradi

aplikuje se druhé pravidlo (rozhoduje pořadí)

```
<!-- HTML kód -->
<article id="prvni-clanek">
<!-- id by se nemělo používat -->
<h2 class="nadpis"></h2>
</article>


/* CSS kód */
/*specifičnost: 0 0 1 0 1 */
#prvni-clanek h2 {
    color: blue;
}

/*
specifičnost:
0 0 0 1 0
*/
.nadpis {
    color: red;
}
```

aplikuje se první pravidlo (rozhoduje specifičnost)
poradi uz nehraje roly
častý zdroj chyb (dedicnost a pravidlo kaskad)

Zapis hodnot

- Hodnota z výčtu
  - Výčet přípustných hodnot pro každou vlastnost, např. bold, block, ...
- Textový řetězec
  - Apostrofy, uvozovky → výběr dle výskytu druhého symbolu – lze provádět escape pomocí \
- Číslo (celé, desetinné, zaporne)
  - Faktor, bez jednotky, udává změnu, např. 1.1
    vetsinou u velikosti pisma
  -
- Délka
  - S jednotkou, např. 12em, případně záporné hodnoty (pokud to má smysl)
- Absolutní vs. **relativní jednotky**
  - 0 se píše vždy bez jednotky
- Barva
  - Nazev
  - RGB model
  - HSL model

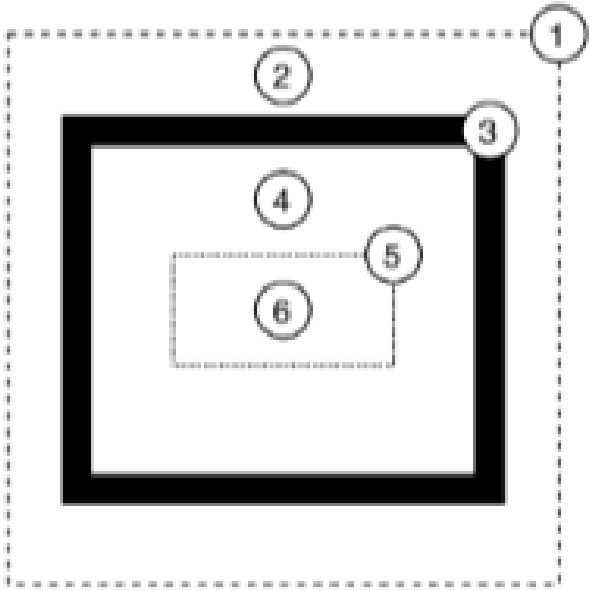

Figure 7: Enter Caption

CSS box model urcuje velikost elementu

Obrázek Box model elementu.:

1. Venkovní hrana elementu,
2. **Venkovní okraj** (margin) elementu,
3. **Rámeček** (border) elementu,
4. **Vnitřní okraj** (padding) elementu,
5. Vnitřní hrana elementu, - neni upravovatelny
6. **Obsah elementu**

Ne kazdy element se chova defualtne jako mox bodel
vlastnost Display to lze prenastavit

Zakladni vlastnosti

- box model
  drive problmey ve zobrazeni na IE6

- layout - rozvrzeni stranky a komponent
  CSS-Grid a Css-Flexbox , nejlepsi 2 specifikace ktere byly (v ramci reakci komunity)
- pozicování
  misto kde se to nachazi v HTML
  Z jednoho mista na druhy
  - absolutni - vzhledem k oknu weboveho prohlizece souradni 0,0 je levy horni okraj
  - relativni- pocatek soustavy souradnic je levy horni okraj vychoziho mista kde se ten element nachazi
- text
  font, velikost, rez

Responzivni design
2007 začátek revoluce (uvedeni prvniho iphone)
různá zobrazovací zařízení - rozliseni displaye
dnes nutnost
adaptivní vs. responzivní design

- Adaptivni
  Je nadmnozina responzivniho designu, ruzne HW naroky, Priklad:
  telefon ma mensi obrazovku a horsi prenosovou rychlost
  zarizeni jsou pomalejsi atd
  Adaptivni design resi aby se to rychle nacetlo na pomalejsich zarizeni
- Responzivni
  Vyhradne vizualizace

fluidní layout - stranka se plne prizpusobuje velikosti okna
jenom ale jednoduche stranky breakpointy - bod zlomu
nejaky bod ktery rika do tohoto okamziku se to zobrazuje takto , od toho okamziku plati tyto pravidla, hrube rozdeleni kombinace
naprosty ideal Obcas se zapomina i na obrovske monitory v teto dobre - Mobile-first CSS

CSS preprocesory
pomalý vývoj CSS → CSS preprocesory skriptovací jazyk (preprocesor) → transpilace → CSS
byl to obrovsky trend ktery spise ted opada
přináší

- funkce

- proměnné
- řízení běhu programu
- pohodlnější syntaxi

příklad: SASS, LESS, Stylus
dnes spíše konzervativní technologie
nový trend postprocesing
mam samotne CSS a postprocesing ho vylepsi
PostCSS, NextCSS

6 Veci co nepatri pod zasnou otazku ale pta se na ne jak se mu zachce

6.1 Javascript

Původní název LiveScript

Plnohodnotný skriptovací jazyk

Primárně určen pro skriptování na straně klienta (pouze část jazyka)

Velmi populární (např. Node.js - webovy server, Adobe Acrobat - prf prohlizec, vývoj aplikací)

Interpretován webovým prohlížečem - interpretovany jazyk

ECMAScript standard jazyka, JS implementace ECMAScript - specifikace

Byla verze 5.1 a pak se 4 roky cekalo na verzy 6 (nejlepsi vylepseni)
od te dobre se cisluje rokem takze javascript2023 (ecmascript2023) Každoroční update, číslovány dle roku

Prohlížeče implementují standardy (analogická situace jako u HTML a CSS)

- Základní funkcionalita
  - Manipulace s HTML a CSS
  - Události (eventy) - onclick...
  - rozsiteni staticke stranky o dinamicke prvky (blikajici text nebo neco horsiho)
  - AJAX
- Nemá žádné grafické schopnosti
  historicke duvody a ani to nema duvod
  - **Umí generovat HTML**
  - Spojení s elementem `canvas`
    JS je nastroj jak do nej kreslit
- Omezení na straně klienta
  - duvod bezpecnost
  - Částečně potlačeno HTML 5 API
    programove rozhrani ktere dokaze komunikovat s klintskym OS
  - dokaze skrz to API ulozit a nacist klientska data

Zacleneni JS do HTML

- Externí soubor (přípona `.js`)

```
<head>
    <script src="file.js"></script>
</head>
```

  atribut SRC
  jde to nacitat libovolne ve strance, rozdil kdy se to zacne spracovavat

- Přímo v HTML stránce (embedded), element `script`

```
<script>
    alert("Hello world");
</script>
```

Pouziti JS na webove strance
Webová stránka reprezentovaná jako **DOM** (Document Object Model) - DOM je specifikaze jak je webova stranka reprezentovana vnitrne
DOM - zachycuje i obsah jako atributy a komentare
HTML-DOM

- Stromová struktura
- JS má stránku přístupnou v objektu `document`
- JS disponuje metodami pro výběr prvků na stránce, jejich přidávání a odebírání

priklad GetElementByID(document.body.h1) ...

Casovace, slideshows, validace formulare ...

6.2 WebAssembly

WASM

JS ma monopol na webove prohlizece

- Transpilace z programovacího jazyka (C, C#, Rust, ...) do instrukční sady virtuálního stroje prohlížeče
  Aplikacni vyrtualizace, vyhoda: Odost Rychlejsi
  Nevyhoda: Binarni soubor - odklon on fylozofie prohlizecu

- "Binární soubor" pro webové prohlížeče
- Velká rychlost
- Není náhrada za JS
  spise doplnek k Javascriptu
  JS zprostredkovava spojeni mezi instrukcni sadou prohlizece a tim co vidi uzivatel weboveho prohlizece
  Js ten zkompilovany code spusti a pak jenom si vezme vystup

6.3 Webove aplikace

Obecně stránky s masivním nasazením skriptů (na straně klienta nebo serveru)

- Nelze jednoduše vymezit
  priklad Webovy obchod nekdo rekne ano nejkdo rekne ne

- Myšlenka: aplikace je webová stránka
  - Mobilní aplikace (např. Apache Cordova, **React Native**)
    staci mit webovy prohlizec, skorapka - nativni aplikace, ve vnitr je jadro weboveho prohlizece a tam stranku
    duvod programatori jsou lini lide , uz maji webovou stranku tak z ni udelaji aplikaci, Visual-studio code, MS teams (vyjebana picovina)

  - Desktopové aplikace (např. Electron)
    staci vytvorit aplikaci pro webovy prohlizec , dobry nastroj pro multy-platformovost
    uz ani nemusim byt na internetu pro use
  - Progresivní webové aplikace (PWA)
    dokazi fungovat i v Ofline rezimu, dramaticky nabiraji na popularite StarBucks - vytvorily objednavkovy system skrz tohodle a diky tomu se jim zvysili trzby

- Platformová nezávislost
- → Horší UI a UX
  nativni aplikace maji vyhodu v tom ze jsou privetivejsi , pouzivaji OS vrstvu (UI)

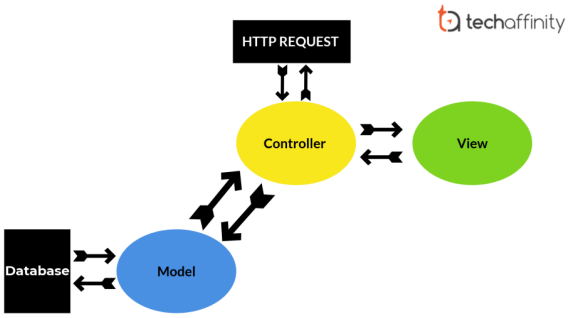

Figure 8: MVC architektura

6.4 Strana serveru

Strana serveru

- Webový server
  muzeme si napsat vlastni (treba za technologie NODE.js)
  spracovani pozadavku
- Skriptovací jazyky
  Vykova obsluhu (to je ta prostredni carka v tom dinamickem rozdeleni)
  programovani jako kazdy jine Pehablee, nebo PYTHON dot.neti technologie
- Webové databáze
  misto pro ulozeni dat
- MVC architektura

  1. prijde HTTP pozadavek na server
  2. je kontaktovan kontroler
  3. kontraktuje model (ulozeni dat (DB))
  4. model (DB) mu ty data vrati
  5. ty data preda do view (to je nejaka sablona)
  6. view ty data vrati (jiz sestavena)
  7. poslano na server
  8. poslano v odpovedi klientovy

  ModelViewControl - popisuje nepresne architekturu beznejh webovejch aplikacich na strane serveru

- REST API (zavadi to Abstrakci)
  spojeno s HTTP pozadavkem
  rozbit na co nejmensi casti vsechny operace by se vykonavali srkz to API
  treba: pridej polozku do kosiku, odeber atd...
  Moznosti ruznejch uzivatelskejch rozhrani
  Treba veskera komunikace NETFLIXU jde skrz jejich API
  - GET
  - PUT
  - POST
  - DELETE

Ajax
Asynchroni Javascript
Jedina technologie ktera se MS povedla (jeho vtip ale actually to ja jako vtip neberu to je smutna realita mrdky z MS)

Pokazde kdyz chceme zmenit WebPage , tak nutne je vyslan HTTP pozadavek a server mu odpovi a nacte se cela stranka znovu

Modifikace části stránky bez nutnosti jejího celého načtení
psle se HTTP pozadavek ale neceka se na odpoved a v moment ale co prijde tak asynchrone to zachyti JS a dinamicky zmeni tu stranku (generovanim HTML)
Nutna podpora prohlizece - dnes nastesti opravdu dobra

Asynchronní komunikace se serverem + JS

Socialni site , (taky to je webova aplikace)
vyuzivaji hojne AJAX a diky tomu to vybouchlo

6.5 Knihovny a frameworky

Odbočka: Knihovny a frameworky

- programatori jsou lini a nechcou psat veci znova
- Poskytují pokročilejší funkcionalitu
- Vystavěné na základních webových technologiích
- Přirozený vývoj
- Například:

- Bootstrap, miniCSS, Materialize, ...
  CSS knihovny, responzivita s breakpointy
  Materialize - hezke materialy od GOOGLU
  Univerzalnost takze je tam tooho mnoho vice nez je potraba
- jQuery, Prototype, ...
  usnadneni trivilani prace
- React, Angular, Vue, Lit (Google Polymer), ...
- React Native, Meteor, Tauri, ...
  dalsi knihodny pro JS
- Nette, Symphony, Laravel, Falcon, Django, Flask, ...
  Servrove FrameWorky, knihovny (za me doporucuji Flask , krasny framework (Lepsi nez PHP na 10000000000000000000000000000 procent))

CMS systemy
Content Management Systems
Systemy pro zpravu obsahu

- Systémy pro správu obsahu
- Blogy, e-shopy
- WordPress, Drupal, Joomla!
- nevyhoda univerzalnost, trpy bezpecnostinimi riziky, podari se najit chybu a je mozne napadnout hodne webu, nizsi rychlost
  dost casto nejde neco udelat co je mimo architekturu toho CMS systemy

6.6 Semanticky web

Původně označován jako Web 3.0
Web 2.0 - je od te doby co se zacal pouzivat JS
Tim Berners-Lee (2001): web = obrovské množství stránek
predstavil to v reci: "I have a dream " , neplest z reci od Martina Luthera Kinga

Aby i stroj pocopil vyznam te stranky
Sémantický web = přidání významu obsahu pomocí metainformace

Zápis pomocí: mikroformáty, mikrodata, RDF, OWL, JSON-LD
umoznuje do jazyka HTML pridat dalsi informace
Ctecky RSS - Mikrodata ktere nesou vyznamovou informaci
Je mozne zachycovat i vztahy mezi entitamy (treba tohle je fotka s italie protoze tam jsou piramidy a to patri do teto kolekce (Czechcloud level zemepis, btw Trnecka rekl normalne Egypt))