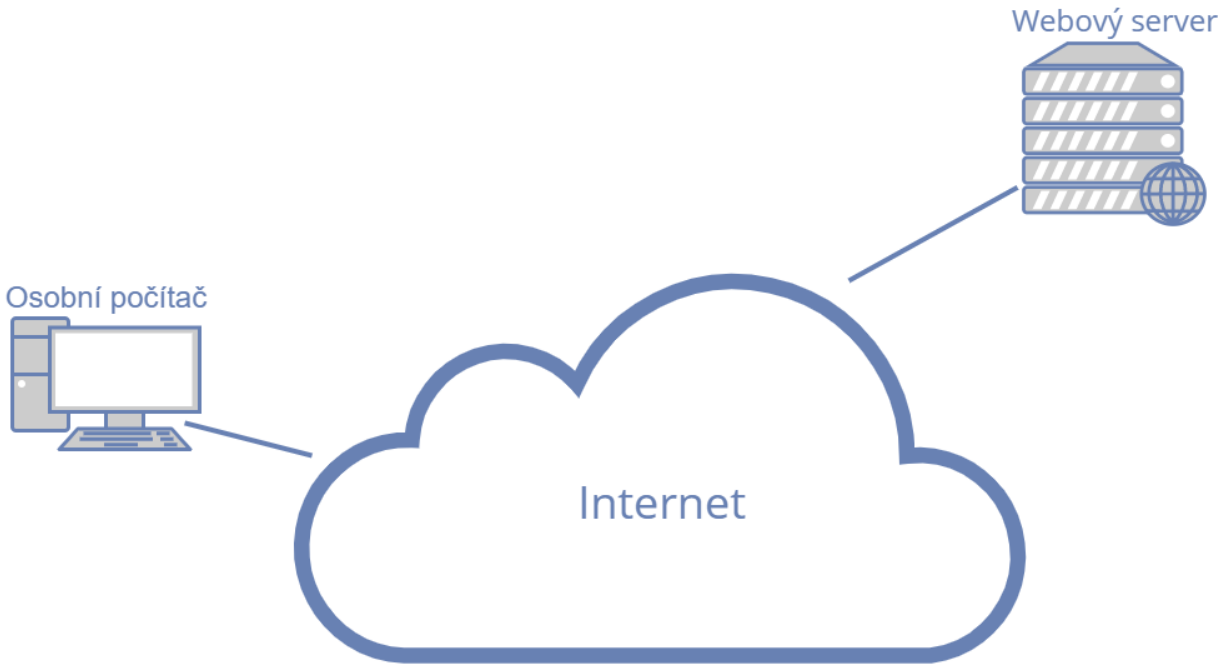
Progresivní technologie v informatice I

Webové technologie

České vysoké učení technické v Praze
Fakulta informačních technologií

Osnova

Architektura klient-server



Obrázek: Základní architektura webu

URL

- *uniform resource locator*
- původní specifikace v RFC 1738[1]
- často zaměňován s URI, URN, ...

Ukázka 1: URL pro vyhledání Nováků

`https://usermap.cvut.cz/search?query=Novák#profile`

- `scheme:[//authority]path[?query][#fragment]`
  - `scheme` → `https`
  - `authority` → `usermap.cvut.cz`
  - `path` → `/search`
  - `query` → `query=Novák`
  - `fragment` → `profile`

[1] https://tools.ietf.org/html/rfc1738

URI

- *uniform resource identifier*
- Předchozí formát platí i pro URI
- URL je podmnožinou URI
  - URL identifikuje „lokaci“ stránky
  - objekty na stránce mohou mít své URI (přednáška REST)

	Jan Novák
URI uživ.	uuid:0d34f3bd-feed-4179-b772-140f269cb240
URL profilu	https://usermap.cvut.cz/profile/140f269cb240
URL fotky	https://usermap.cvut.cz/photo/140f269cb240.jpeg
QR kód	https://usermap.cvut.cz/qrcode/140f269cb240

Tabulka: Ukázka URI a URL popisujících stejnou osobu

Doména

- jednoznačný identifikátor počítače
- většinou *human-readable*
- hierarchický
- druhy: doména země, generická doména
- příklad: `usermap.cvut.cz`

- správce *top-level* domén **ICANN**[2]
- autorita *top-level* domén **IANA**[3] (dříve samostatná, nyní pod ICANN)
- autorita `.cz` domény **CZ.NIC**

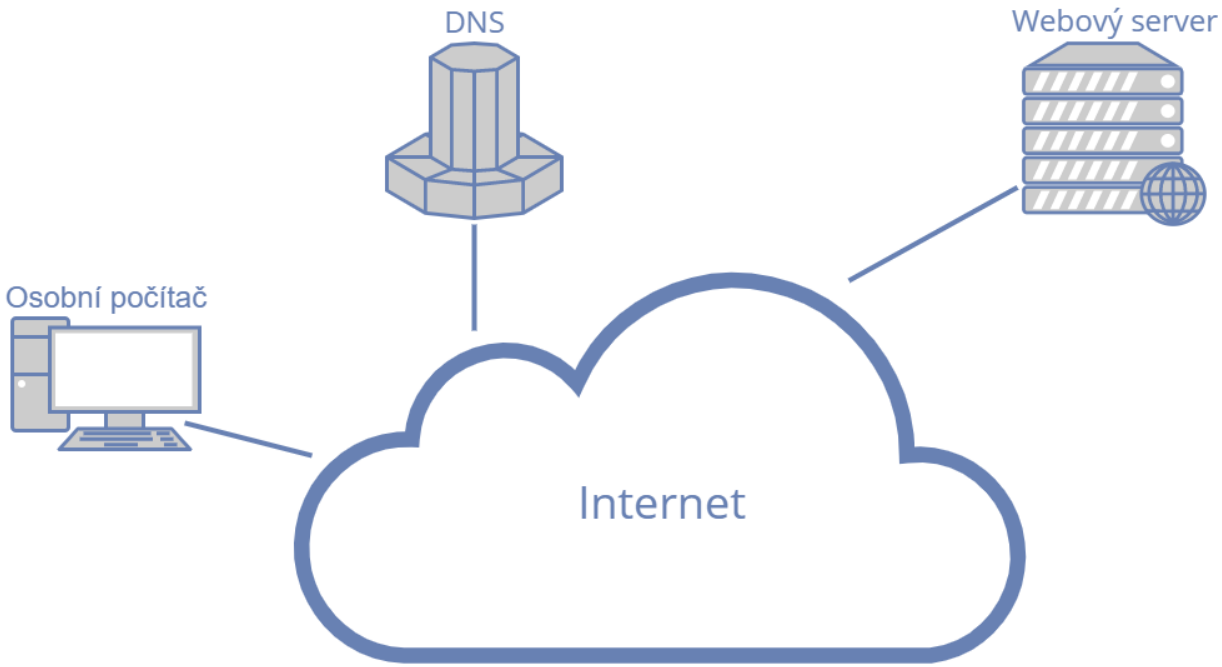
[2] Internet Corporation for Assigned Names and Numbers
[3] Internet Assigned Numbers Authority

DNS

- *domain name system*
- poskytuje metadata k dané URL
- především slouží jako překlad domény na IP adresu

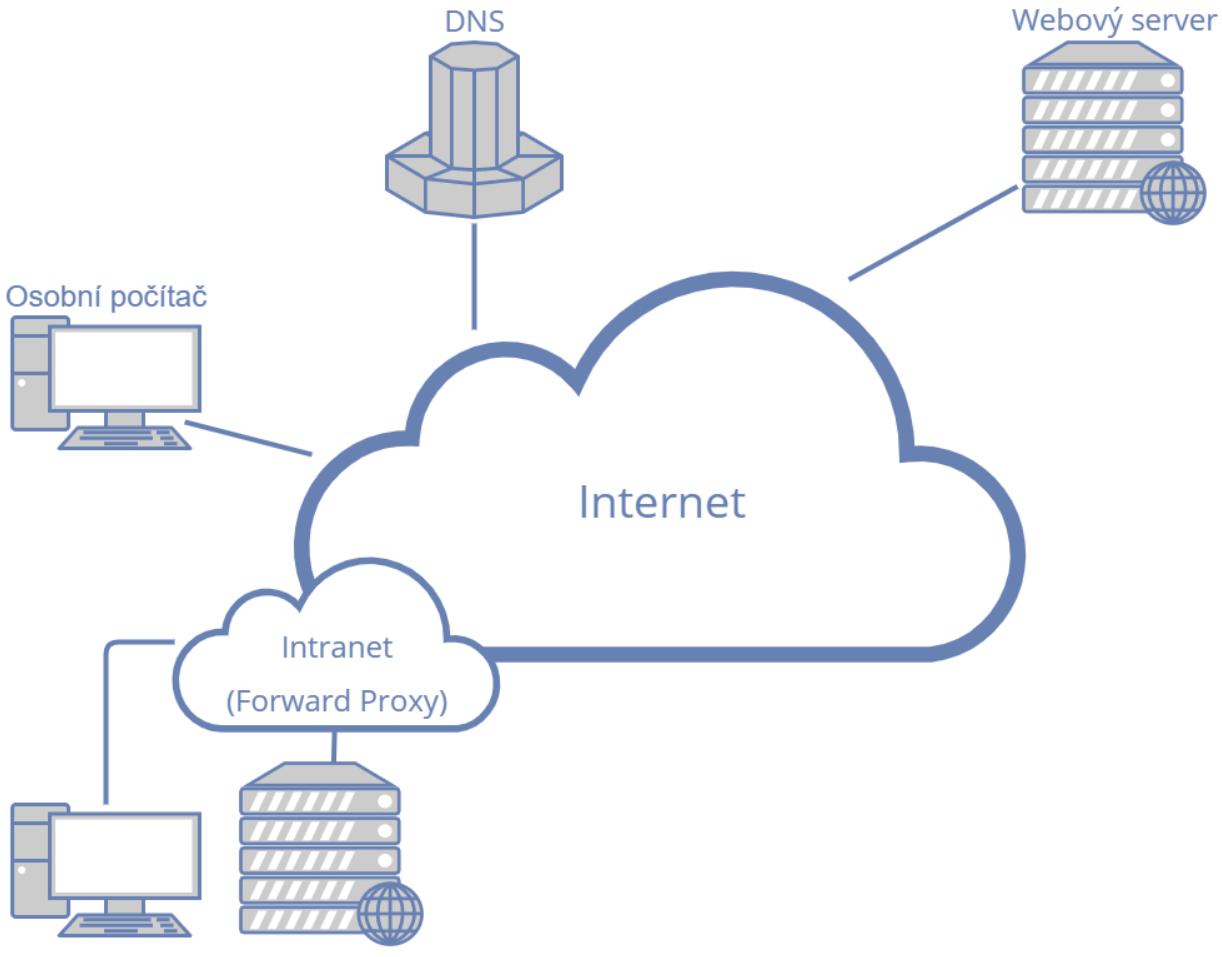
- hierarchický $\rightarrow$ 12 kořenových serverů
- efektivní cachování $\rightarrow$ změny se projeví po čase

DNS



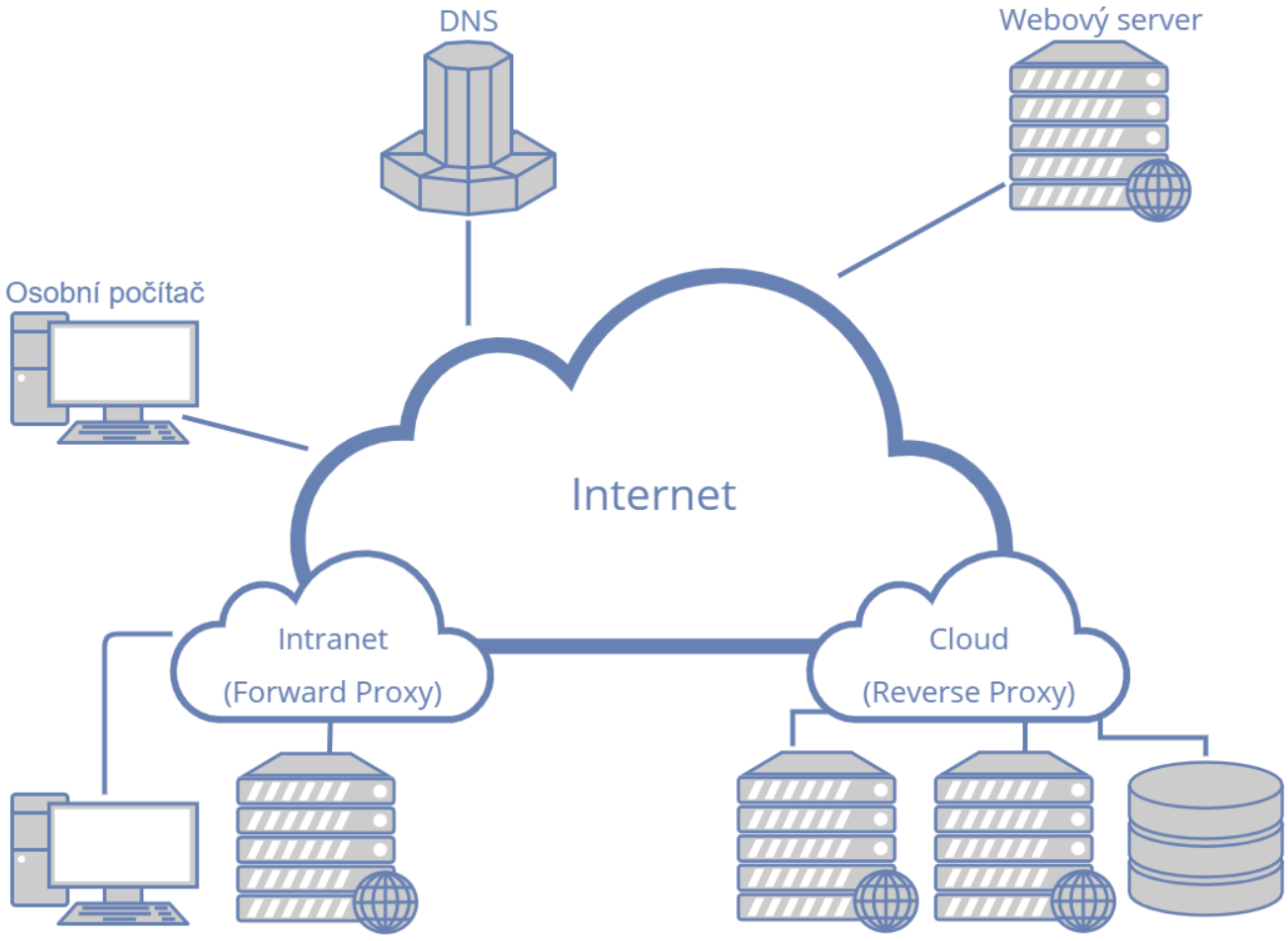
Obrázek: Architektura webu (vč. DNS serveru)

Forward Proxy



Obrázek: Příklad *forward proxy* ve firmě

Reverse Proxy



Obrázek: Příklad *reverse proxy* na webu

HTTP

- protokol aplikační vrstvy
- nad TCP/IP protokoly (obvykle 80, 8080)
- textový protokol[4] (raw data)
- požadavek (*request*) $\rightarrow$ odpověď (*response*)
- **bezestavový** protokol
- verze 0.9, 1, 1.1, 2, 3
- šifrovaná varianta protokolu HTTPs (443, 8443)

[4] HTTP 2.0 podporuje binární přenos

Požadavek (*Request*)

Ukázka 2: Formát požadavku

```
Request ::= RequestLine (Header CRLF)* CRLF [body]
RequestLine ::= Method RequestUri HTTPVersion CRLF
Header ::= Host | Accept | ...
Method ::= GET | HEAD | ...
```

- reprezentuje uživatelovu akci
- metoda specifikuje „akci“
- adresa URL specifikuje „cíl“ akce

Ukázka 3: Příklad požadavku

```
GET /profile/140f269cb240 HTTP/1.1
Host: usermap.cvut.cz
Accept: text/html
```

Základní metody

- **GET** slouží k získání dat (stavu)
  - zobrazení konkrétní stránky
  - vyhledávání a filtrování
- **POST** slouží ke změně dat (stavu)
  - vytváření, upravování, mazání
- *K zamyšlení*
  - Může být akce smazání pouhý odkaz?
  - Kdy POST namísto GET?
  - Kdy GET namísto POST?

Odpověď (*Response*)

Ukázka 4: Formát odpovědi

```
Response ::= StatusLine (Header CRLF)* CRLF [body]
StatusLine ::= HTTPVersion Status CRLF
Status ::= StatusCode ReasonPhrase
Header ::= Content-Type | ...
```

Ukázka 5: Ukázka odpovědi

```
HTTP/1.1 200 OK
Date: Sun, 5 Jul 2020 16:46:06 GMT
Content-Length: 39
Content-Type: text/html

<html>
    <body>Ahoj svete!</body>
</html>
```

Stavové kódy

Informational	**1xx** požadavek přijat, zpracování pokračuje
Success	**2xx** úspěch, požadavek přijat a je/bude zpracován
Redirection	**3xx** přesměrování, další akce nutná pro dokončení
Client Error	**4xx** chyba klienta, špatná syntaxe požadavku, neplatné údaje, nevalidní data, ...
Server Error	**5xx** chyba serveru, údržba, neodchycená výjimka, problém s připojením, ...

Stavové kódy (obvyklé)

200 OK

301 Moved Permanently

400 Bad Request *(obecně neplatný požadavek)*

401 Unauthorized

403 Forbidden

404 Not Found

500 Internal Server Error

Hlavičky (obecné)

Požadavek	Odpověď	Popis
`Host`	-	doména požadavku
`Accept`	`Content-Type`	typ obsahu
`Accept-Language`	`Content-Language`	jazyk obsahu
`Referer`[5]	-	předchozí adresa
`Origin`	-	předchozí doména
-	`Location`	přesměrování

Tabulka: Hlavičky

Další specializované hlavičky si ukážeme v průběhu semestru.

[5] Jedná se o překlep ve standardu.

HTTP/1.1

- povinná hlavička **`Host`**
- možnost poslat odpověď ve více částech
- podpora trvalého spojení (`Connection: Keep-Alive`)
- správné zpracování `100 Continue`
- pokročilejší *cacheování*

HTTP/2

- rozšíření HTTP/1.1
- binární varianta
- multiplexovaný
- stačí jediné spojení se serverem
- efektivní komprese hlaviček
- podpora *server-push*

HTTP/3

- rozšíření HTTP/2
- funguje nad UDP
- zdokonaluje paralelní zpracování
- umožňuje jednoduší přechod mezi sítěmi

HTML

- *HyperText Markup Language*
- značkovací jazyk pro publikování na webu
- strukturuje dokumenty pomocí odkazů
- umožňuje vkládat multimédia (obrázky, videa, hudba, . . .)
- umožňuje komunikovat s uživateli pomocí formulářů[6]
- použití skriptovacích jazyků[7]

[6] přednáška na formuláře
[7] přednáška na JavaScript

Syntaxe

- stromově strukturovaný dokument
- podobný **XML**[8]

Obsahuje:

- elementy a jejich atributy

```
<a href="//localhost/">Homepage</a>
```

- obsah elementů (text)
- deklarace

```
<!DOCTYPE html>
```

- `<-- komentáře -->`

[8] Předpokládaná znalost z předmětu BI-XML

Struktura

```
1  <!DOCTYPE html>
2  <html lang="cs">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="author" content="Jan Novák">
6          <title>Profil uživatele Jan Novák</title>
7          <!-- styly nebo skripty -->
8      </head>
9      <body>
10         <h1 id="profile-name">Jan Novák</h1>
11         <p class="lead">Jan Novák je naším předním
12                 zaměstnancem.</p>
13     </body>
14 </html>
```

Dělení elementů (blokové)

- blokové
  - Nadpisy `h1`, `h2`, `...`, `h6`
  - Odstavce `p`, `blockquote`
  - Formuláře `form`, `fieldset`[9]
  - Seznamy `ol`, `ul`, `dl`
  - Tabulka `table`
  - Formátovaný text `pre`
  - Kontaktní informace `address`
  - Horizontální oddělovač `hr`
- řádkové

[9] Formuláře budou probrány v samostatné přednášce.

Dělení elementů (řádkové)

- blokové
- řádkové

Text	`em`, `strong`, `sub`, `sup`
Definice	`dfn`
Citace	`cite`, `q`
Zkratka	`abbr`, `acronym`[10]
Kódy	`kbd`, `samp`, `var`, `code`
Formuláře	`input`, `select`, `textarea`, `label`, `button`
Odkazy	`a`, `map`
Multimédia	`img`, `object`
Zlom	`br`
Další	`tt`, `i`, `b`, `big`, `small`

[10] HTML5 už má jen `abbr`

Dělení elementů (bezvýznamové)

- blokové `div`
- řádkové `span`

- strukturují dokument
- nenesou žádný význam
- mohou nést další informace ve formě atributů

Text - příklad

```
1  <h1>Základy webových technologií</h1>
2  <p><abbr title="Tvorba webových aplikací">BI-TWA.1</abbr>
3      bude o základech webových technologií&hellip;</p>
4
5  <h2>HTTP</h2>
6  <p>HTTP je <strong>textový protokol</strong>.</p>
7  <img src="images/ukazka-protokolu.png"
8      alt="Uživatel pošle požadavek na server, ten je
9  rozparsován a následně zaslána uživateli odpověď.">
10 ...
11
12 <h2>HTML formát</h2>
13 <p>...</p>
14 ...
15
16 <div class="footer">Zodpovědná osoba:
17     <address>Jan Novák</address>
18 </div>
```

Seznamy - příklad

```
<dl class="contact-info">
    <dt>Jméno</dt>
    <dd>Jan Novák</dd>

    <dt>Kontakt</dt>
    <dd><ol>
        <li><a href="mailto:jan@novak.cz">
            jan@novak.cz
        </a></li>
        <li><a href="tel:+444123456789">
            +444 12 345-6789
        </a></li>
        <li>Novákových 17, Praha - venkov</li>
    </ol></dd>

    <dt>Zájmy</dt>
    <dd><ul class="interests">
        <li>počítačové hry</li>
        <li>hudba</li>
    </ul></dd>
</dl>
```

Tabulky - příklad

```
<table>
    <thead><tr>
        <th>Student</th>
        <th>Počet bodů</th>
        <th>Známka</th>
        <th>Akce</th>
    </tr></thead>
    <tbody>
        <tr>
            <th><a href="/detail/1">Jan Novák</a></th>
            <td>78</td>
            <td>C</td>
            <td><ul>
                <li><a href="/zapocet/1">
                    Neudělit zápočet</a></li>
                <li><a href="/znamka/1">
                    Zapsat známku</a></li>
            </ul></td>
        </tr>
        ...
    </tbody>
</table>
```

HTML 5

- revoluce v HTML
- přidáno velké množství sémantických elementů (tagů)
- sloučení W3C a WHATWG standardů 29. 5. 2019
- definuje jak zpracovávat HTML dokument (*outline*, i chyby)
- staré elementy dostaly nový význam (`i`, `b`, ...)

Struktura

Bloky (slouží pro stanovení *outline*):

- hlavní obsah dokumentu `main`
- celistvá část dokumentu `article` (př. novinka, jeden komentář)
- část `section` (př. blok komentářů u novinky)

Struktura:

- hlavička `header`
- navigace `nav`
- vedlejší informace `aside` (většinou nesouvisející s obsahem, poznámky autora, ...)
- patička `footer`

HTML 5 a další rozšíření

Living standard definuje další vlastnosti:

- možnost využívat microdata
- popis načítání stránek, vykreslování, komunikace, ...
- JS rozšíření: WebAPI, workers, storage, ...

Formuláře

- přidávají prvek interaktivity
- uživatel komunikuje s aplikací pomocí tzv. *named controls*
- skládá se z:
  - ovládacích prvků (*controls*),
  - popisků (*labels*),
  - textové obsahu

Formuláře

- pomocí atributu `action` určuje URL, která formulářový požadavek zpracuje
- atributem `method` specifikujeme druh požadavku

```
<form action="process.php" method="get">
  ...
</form>
```

Formulářové prvky

- tlačítka (*buttons*)
- zatrhávací políčka (*checkboxes*)
- přepínače (*radio buttons*)
- textová pole (*text inputs*)
- nahrávání (*file select*)
- skryté prvky (*hidden controls*)
- ...

Textový vstup

- generický text `<input type="text"/>`
- hesla (soukromé informace) `<input type="password"/>`

HTML5 přidává nové

- email `<input type="email"/>`
- telefon `<input type="tel"/>`
- URL `<input type="url"/>`
- datum `<input type="date"/>`
- čas `<input type="time"/>`
- číslo `<input type="number"/>`
- hodnota z rozsahu `<input type="range"/>`

- doporučování `<input type="text"list="list»` `<datalist id="list"` ... $< /datalist >$

Textový vstup

- navíc můžeme definovat velké textové pole
  `<textarea>...</textarea>`

```
<form ...>
  <label for="field-name">Jméno</label>
  <input id="field-name" name="name" type="text" />

  <label for="field-surname">Příjmení</label>
  <input id="field-surname" name="surname" type="tex

  <label for="field-comment">Komentář</label>
  <textarea id="field-comment" name="comment"></text

  <button type="submit">Okomentovat</buttom>
</form>
```

Popisky

- ke každému ovládacímu prvku patří popisek (*label*)
- navázání přímo

```
<label>Jméno: <input type="text" ... />
```

- navázání pomocí ID elementu

```
<label for="field-name">Jméno</label>
<input id="field-name" type="text" />
```

- při kliknutí na popisek se aktivuje i ovládací prvek

Tlačítka

- odesílací `<button type="submit">...</button>`
- reset `<button type="reset">...</button>`
  - elementy `input`, `output`, `select` a `textarea`
  - resetovací algoritmus
- generické `<button type="button">...</button>`
  - navázané na JavaScript

Přepínače a zatrhávací políčka

- přepínače
  - `<input name="..." type="radio" value="..."/>`
  - pokud je více přepínačů se stejným jménem (`name`), může být aktivní pouze jeden
- zatrhávací políčka
  - `<input name="..." type="checkbox" value="..."/>`
  - pokud je více přepínačů se stejným jménem (`name`), odešle se pouze jeden
  - pro odeslání všech je nutné zvolit jméno se suffixem `colors[]`
- popisek (*label*) se dává ke každému zvlášť

Výběrové pole

Podobné přepínačům.

```
<select size="3" multiple>
  <option value="red">červená</option>
  <option value="green">zelená</option>
  <option value="blue">modrá</option>
</select>
```

Popisek (*label*) se váže vždy k výběrovému poli.

Formuláře v HTML5

- nové formulářové prvky
- nové atributy
  - `required`
  - `pattern`, `min`, `max`, ...
  - `placeholder`, `autocomplete`
  - `form`, `formaction`, `formmethod`

Zpracování formuláře (metody)

- data jsou odeslána na URL uvedenou v atributu `action`
- `method="GET"` data jsou připojena k URL
  - `enctype = application/x-www-form-urlencoded`
  - nesmí mít vedlejší efekty (*idempotentní*)
  - pouze ASCII data
- `method="POST"` data se posílají v těle požadavku
  - může posílat i binární data
  - potom `enctype = multipart/form-data`

Successful Controls

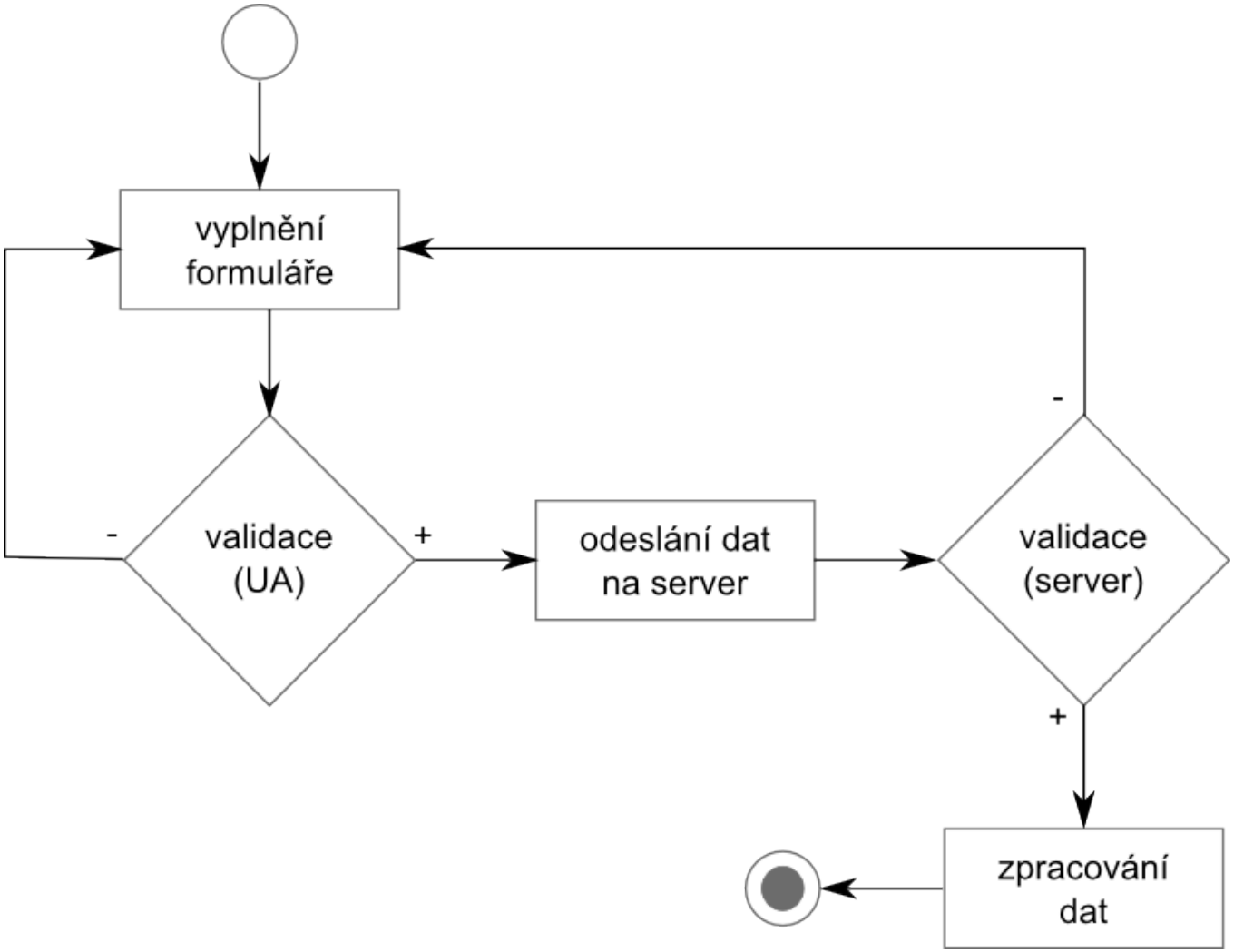
- formulářová pole `disabled` **nemůžou** být successful
- pouze aktivované tlačítko je successful
- všechna zaškrtnutá políčka `type="checkbox"` můžou být successful
- pro přepínače `type="radio"` se stejným `name` pouze zaškrtnuté může být successful
- pouze vybrané `<option>` můžou být successful (pokud není vybraný žádný, nic se neodešle)

Zpracování formuláře (postup)

1. identifikovat *successful controls*
2. vytvořit *form data set* (dvojice klíč-hodnota)
3. zakódovat data
4. odeslat na požadovanou URL

V HTML5 mnohem složitější.

Životní cyklus formuláře



Zdroj: BI-WT1, BI-WT2

Vícenásobné odeslání dat

- problém vytvoření 2 požadavků
  - zboží se vloží dvakrát do košíku
  - vytvoří se dvě objednávky
  - odešle se dvakrát úloha
- ochrana před *nevyžádanou* akcí
  - 2x zachycení eventy
  - tlačítko zpět
  - refresh (F5) stránky

Vícenásobné odeslání dat

`GET /user/create`

1. Vygenerován token (náhodné číslo).
2. Uložen do
   - session (na serveru),
   - skryté pole formuláře (u klienta).
3. Odeslání požadavku (`POST /user`).

Vícenásobné odeslání dat

`POST /user`

1. Kontrola existence tokenu.
   - Pokud neexistuje, požadavek *neplatný*. Přesměrování uživatele `GET /user/create`.
2. Kontrola rovnosti tokenu v session (na serveru) a ve formuláři (u klienta).
   - Pokud se nerovnají, požadavek *neplatný*. Přesměrování uživatele `GET /user/create`.
3. Odstranění tokenu ze session.
4. Zpracování požadavku.
5. *Přesměrování uživatele `GET /user/163`*

Kaskádové styly

- *Cascading Style Sheets*
- jazyk pro popis grafické reprezentace dokumentace
- rozšíření HTML
  - HTML popisuje strukturu
  - CSS popisuje grafickou část
- jednoduchost (př. `background-color: blue`)
- zpětná (*backward*) i dopředná (*forward*) kompatibilita
- *device independent*
- podpora v prohlížečích `caniuse.com`

Syntax

- *style sheet* je seznam pravidel
- pravidlo (*rule*) se skládá ze *selector*u a bloku deklarací
- deklarace se oddělují středníkem a každá se skládá z
  - *property* vlastnosti (př. `color`)
  - *value* hodnoty (př. `red`)

Ukázka 6: Ukázka CSS pravidla

```
div.lead {
    font-size: 2em;
    text-indent: 0;
}
```

Vlastnosti

- seskupování pravidel (př. `h1, h2, h3 { ... }`)
- seskupování vlastností (př. `font`)
- case insensitive (až na vybrané části)
- komentáře `/* ... */`
- lze uvést kódování (př. `@charset "UTF-8"`, proč?)

Integrace do HTML

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

Hodnoty

Délky

- absolutní
  - px
  - in
  - cm
  - mm
  - pt (point, 1/72 in)
  - pc (pica, 12pt)
- relativní
  - em
  - rem
  - ex
  - %
  - vw
  - vh
  - dvw
  - dvh

Elementy nedědí relativní hodnoty, ale hodnoty už vypočtené.

Hodnoty

- Barvy
  - red *(keyword)*
  - #f00
  - #ff0000
  - rgb(255, 0, 0)
  - rgb(100%, 0%, 0%)
  - rgba(255, 0, 0, 1)
  - hsv, lab, oklab, lch, oklch
- Fonty
  - serif *(keyword)*
  - sans-serif *(keyword)*
  - Arial *(string)*
  - "Arial Bold" *(string)*
  - cursive
  - fantasy
  - monospace

Selektory

`*`	univerzální selektor
`E`	selektor elementu
`.class`	selektor třídy
`#id`	selektor identifikátoru
`E.c1.c2`	kombinace elementu a dvou tříd
`E F`	selektor potomka (F je někde pod E)
`E > F`	selektor přímého potomka (F je přímo pod E)
`E + F`	selektor následníka (F následuje za E)
`E:first-child`	pseudo-třída pro prvního potomka
`E:hover`	pseudo-třída pro "aktuálně ukazovaný"prvek
`E:lang(x)`	pseudo-třída pro jazykovou verzi
`E[attr="val"]`	selektor atributu (`=`, `~=`, `|=`)

Pseudo-elementy

- `::first-line`
- `::first-letter`
- `::before`
- `::after`

Nové selektory

- :is

```
div.lead a,
p.lead a,
blockquote.lead a {
    color: blue;
}

/* vs */

:is(div, p, blockquote).lead a {
    color: blue;
}
```

Nové selektory

- `:is`
- `:where`

stejné jako `:is`, ale nulová specificita (později)

Nové selektory

- `:is`
- `:where`
- `:not`

```
/* Každý odstavec, před kterým je jiný odstavec. */
p + p {
    font-size: 0.9em;
}

/* vs */

/* Každý odstavec, který není prvním potomkem. */
p:not(:first-child) {
    font-size: 0.9em;
}
```

Nové selektory

- `:is`
- `:where`
- `:not`
- `:has`

```
/* Styluji "a" */
p.lead a {
    color: blue;
}

/* vs */

/* Styluji kontejner, který obsahuje "a" */
p.lead:has(a) {
    padding: 1rem;
}
/* PERFORMANCE !!! */
```

Dědičnost

Hodnota stylu:

- iniciální
- zděděná (z rodiče)
  některé hodnoty se nedědí (př. pozadí)
- jako výsledek pravidel

Styly mohou být definovány:

- autorem stránky
- uživatelem
- prohlížečem (user agent)

Dědičnost

Pro určení pravidla se vyhodnotí *kaskáda*:

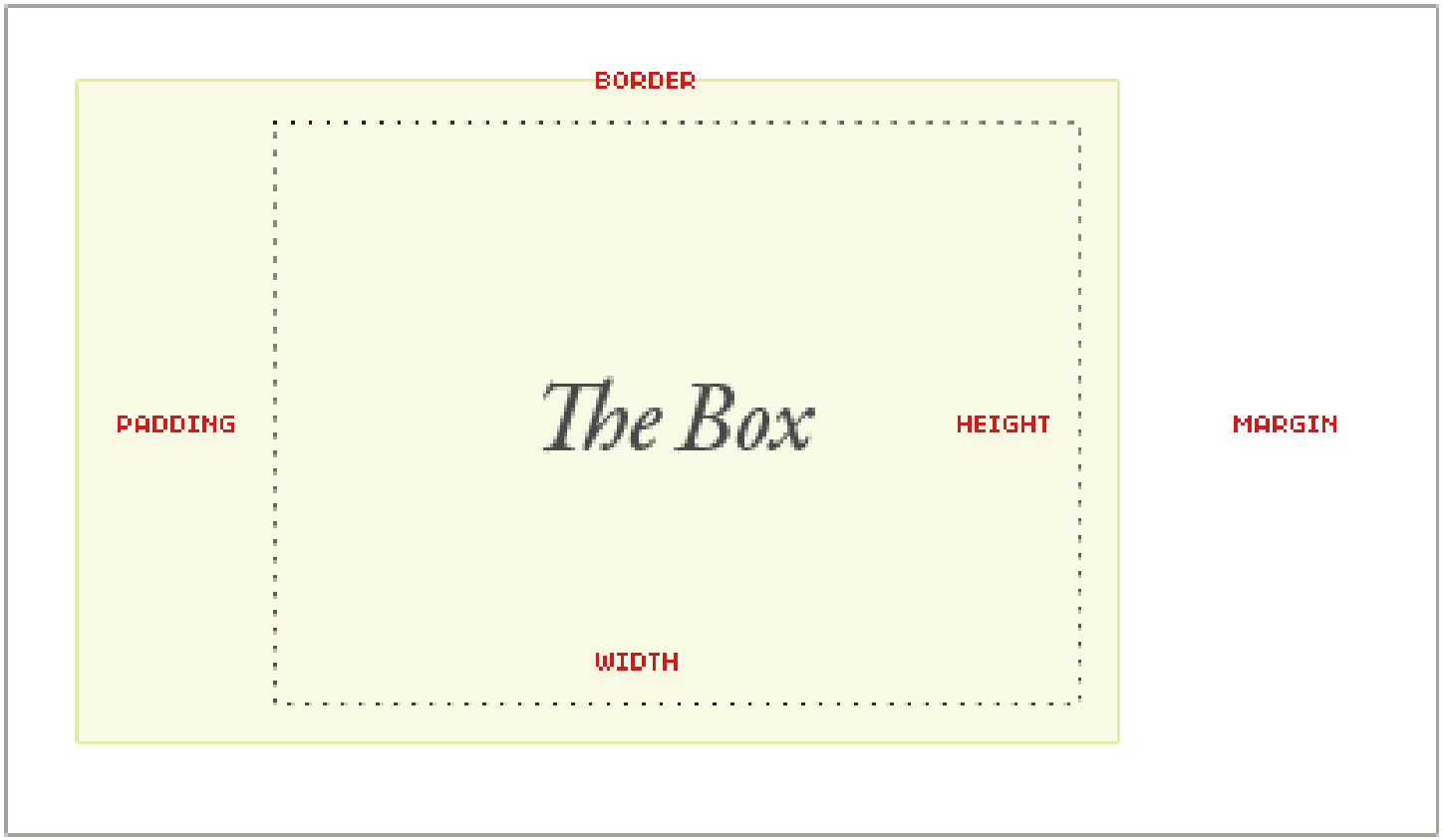
1. Nalezni všechny deklarace, které se vztahují k danému elementu.
2. Seřaď deklarace dle důležitosti:
   `UA < user < author < author !imp < user !imp`
3. V případě rovnosti, seřad deklarace dle specificity selektoru
4. V případě rovnosti, seřad deklarace dle pořadí, ve kterém byly specifikovány.
   pozdější má vyšší prioritu

Specificita

1. atribut `style` (1 nebo 0)
2. #id atribut (počet)
3. ostatní atributy a pseudo-třídy (počet)
4. elementy a pseudoelementy (počet)

```
          * 0, 0, 0, 0
         li 0, 0, 0, 1
li::first-line 0, 0, 0, 2
      ul li 0, 0, 0, 2
h1 + *[rel=up] 0, 0, 1, 1
ul.class li 0, 0, 1, 2
ul.class li:first-child 0, 0, 2, 2
        #id 0, 1, 0, 0
      h1#id 0, 1, 0, 1
h1[id="id"] 0, 0, 1, 1
style="..." 1, 0, 0, 0
```

Box model



Obrázek: Box model v CSS[11]

[11] Zdroj: https://css-tricks.com/the-css-box-model/

Box model

- `width` (při `position: relative, static`)
  - pokud je prázdný, pak se nastaví na 100% šířky rodiče
    pokud nastavíme padding, pak se ukrojí z vnitřní části
  - pokud ho nastavíme na 100% (nebo jinou velikost)
    pokud nastavíme padding, pak se rozšíří celý box do vnější části
- defaultně jsou hodnoty padding a margin (0 nebo co nastaví prohlížeč)
- proto se používá *CSS reset*
- při `position: absolute`, šířka (nedefinovaná) odpovídá minimální nutné šířce
- rozdíl mezi `box-sizing: content-box` a `box-sizing: border-box`

CSS 3

- media queries
- page
- nové selektory
- nové pozadí, okraje, stínování
- animace, transformace
- nové jednotky
- flexbox a `grid`
- *a mnoho dalšího...*

Flexbox

- `display: flex`
- strukturování lineárního flow
- definování zalamování

Zdroje:

- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://flexboxfroggy.com/

Grid

- `display: grid`
- strukturování do 2D
- definování regionů
- brzy podpora Chrome (zatím jen 50%) `subgrid`

Zdroje:

- https://css-tricks.com/snippets/css/complete-guide-grid/
- https://cssgridgarden.com/

Media queries

- více zařízení => jedna stránka (více vzhledů)
- specifikuje zařízení + další parametry

Ukázka 7: Media queries v CSS 3

```
@media screen and (min-width: 20rem) {
    /* aplikuje se pouze při splnění podmínek */
}
```

- adaptivní (přímo na konkrétní zařízení)
- responzivní (na konkrétní breakpointy)
- fluidní (využívá procenta)
- fixní (jasné využití pixelů)

Metodiky v CSS

- konvence pro pojmenování stylů
- většinou nerespektují sémantické CSS
- příklady
  - OOCSS *(object-oriented CSS)*
  - BEM *(block element modifier)*
  - SMACSS *(scalable and modular architecture for CSS)*
  - atomické CSS
- většinou zachovávají nízkou specificitu
- Case Study: Fakturoid CSS

OOCSS

hlavně struktura

komponenty `.button`

element `.button-icon`

modifikátor `.button-primary`, `.button-secondary`

```
<button class="button button-primary">
    <span class="button-icon"></span>
    Submit
</button>
```

- nezávislost na HTML elementech
- nezávislost na kontejneru
- nízká specificita
- komponenty
- prefixování

BEM

hlavně pojmenování

blok	`.btn`
element	`.btn__icon`
modifikátor	`.btn-primary`
modifikátor elementu	`.btn__icon-primary`

```
<button class="btn btn--primary">
    <span class="btn__icon btn__icon--secondary"></span>
    Submit
</button>
```

SMACSS

základní styly	HTML tagy
moduly	třídy bez prefixu
layout	`.l-content`, `.l-inline`
stav	`.is-opened`
téma	skiny jsou v samostatném souboru

Pro a proti

- přináší jednoduchý systém
- nízká/nulová specificita usnadňuje aplikování pravidel (z hlediska programátora)
- styly jsou více promítnuté do HTML, chybí sémantika
- přenáší se více dat, která se nedají jednoduše cachovat
- nevyužívají potenciál HTML5 atributů
  - `<input ... required>`
  - `input:invalid { ... }`
  - `<div ... data-opened="true»`

Preprocessory

- rozšiřují syntax CSS
- transpilují se do jednoho nebo více CSS
- příklady
  - SASS
  - LESS
  - Stylus
  - PostCSS *
- umožňují využívat některé vychytávky budoucího CSS
  - proměnné (už v CSS)
  - mixins
  - zanořené CSS (brzo v CSS)
  - práce s barvami (už v CSS)
  - *další*

SASS - ukázka zanořování

```
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

SASS - ukázka mixins

```
@mixin theme($theme: DarkGray)
  background: $theme
  box-shadow: 0 0 1px rgba($theme, .25)
  color: #fff

.info
  @include theme

.alert
  @include theme($theme: DarkRed)

.success
  @include theme($theme: DarkGreen)
```