www

World
Wide
Web

www

HTTP	PHP	Webkit	Vyhledávač
HTML	Python	Blink	Katalog
CSS	Ruby	Gecko	Portál
JavaScript	Node.js		Sociální síť
			Blog
URL			CMS
Cookie	AJAX	Apache	Wiki
Hyperlink	jQuery	Lighttpd	Fórum

World Wide Web

- WWW je informační prostor v němž jeho položky interpretované jako zdroje jsou identifikovány globálními identifikátory URI
  - Základy a principy WWW
  - Lokace zdrojů (URL a URI)
  - Přenos informací (HTTP protokol)
  - Reprezentace informací (HTML jazyk)

Prohlížeč → URI → DNS → TCP →

- **URL** zadané do prohlížeče:
  `https://www.czso.cz/csu/czso/domov`
- **DNS** dotaz na A/AAAA záznam pro doménu
  `www.czso.cz`
- **DNS** odpověď:
  `www.czso.cz. IN A 194.48.241.132`
- dle protokolu v **URL** (**https**) se prohlížeč připojí k **TCP** portu **443** počítače **194.48.241.132**

→ TLS → HTTP dotaz →

- vytvoří se šifrované spojení (**TLS**), proběhne kontrola identity serveru/provozovatele dle certifikátu dodaného serverem (*Digicert, Inc.*)
- v rámci šifrovaného spojení a na základě daného **URL** `https://www.czso.cz/csu/czso/domov` položí prohlížeč (klient) **HTTP** dotaz:

```
GET /csu/czso/domov HTTP/1.1
Host: www.czso.cz
```

- server dotaz zpracuje, podívá se do databáze a vrátí požadované informace v jazyce **HTML**

HTML stránka

```
<!DOCTYPE html>
<html class="ltr" dir="ltr" lang="cs-CZ">
<head>
    <title>Český statistický úřad | ČSÚ</title>
    <meta charset="utf-8">
    ...
</head>
<body>
...
<h2>Nejnovější údaje</h2>
<table>
    ...
    <tr>
        <td> <a href="/csu/czso/hdp_narodni_ucty">Hrubý domácí produkt</a></td>
        <td> <span class="POKLES" title="3. čtvrtletí 2023">-0,6 %<span>
        </td>
    </tr>
    <tr>
        <td> <a href="/csu/czso/prumysl_energetika">Průmyslová produkce</a></td>
        <td> <span class="POKLES" title="září 2023">-5,0 %<span>
        </td>
    </tr>
    <tr>
        <td> <a href="/csu/czso/stavebnictvi">Stavební produkce</a></td>
        <td> <span class="RUST" title="září 2023">1,7 %<span>
        </td>
    </tr>
</table>
...
```

Zobrazení prohlížečem

- prohlížeč interpretuje získané informace a v případě potřeby postahuje dodatečné informace (komponenty, obrázky, styly, skripty, apod.)
- prohlížeč zobrazí informace včetně hypertextových odkazů na další zdroje

Hrubý domácí produkt	-0,6 % ↓
Průmyslová produkce	-5,0 % ↓
Stavební produkce	1,7 % ↑

Příklad ukazuje 3 pilíře architektury WWW

- **Identifikace**:
  - každý zdroj je identifikován jedinečným URI
- **Interakce**:
  - webobvý agent komunikuje standardním protokolem, který provádí interakci výměnou zpráv dané syntaxe a sémantiky
- **Interpretace** (reprezentace)
  - v protokolu přenášíme data a popřípadě metadata z určeného zdroje

Identifikace

- **Uniform resource identifier** (URI)
  - jedinečný identifikátor
  - vytváří *globální pojmový prostor* a tím globální síťový efekt.
  - Idea globálního hypertextového prostoru pochází vizionáře Douglase Engelbarta z roku 1990 (také vynalezl myš v roce 1961)

URI schéma HTTP

- URI schéma HTTP:

  http://počítač/adresare/soubor

  https://počítač/adresare/soubor

- Dále je možné připojit

  ?parametr=hodnota;parametr2=hodnota

  #vnitřní_odkaz

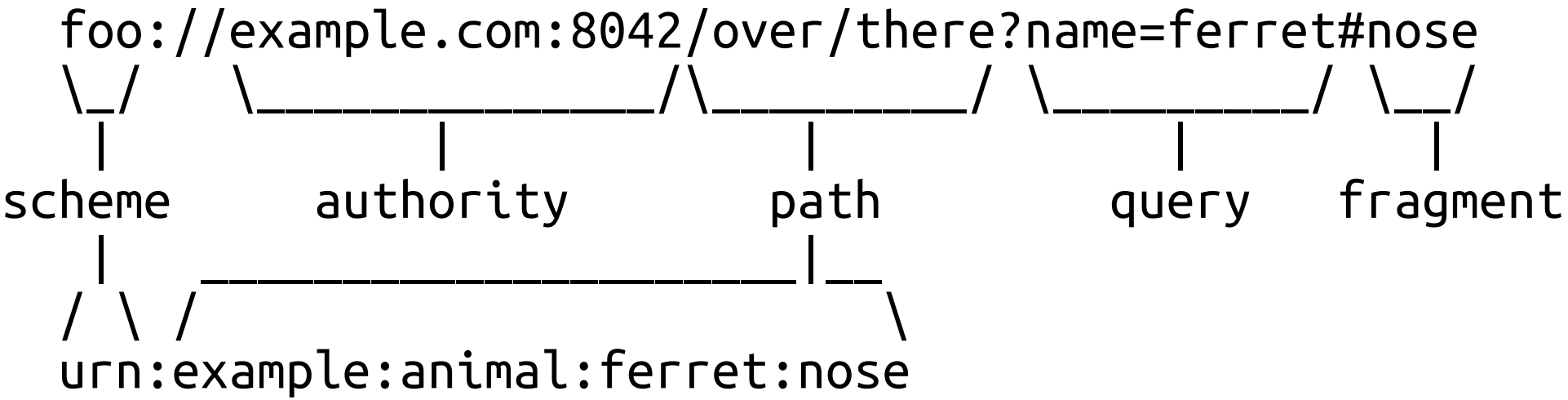
Uniform Resource Identifier

- **URI** se dělí na **URN** a **URL**
  - **URN**: **Uniform Resource Name** – jednotné jméno zdroje, neřeší cestu k němu

    `urn:isbn:0451450523`
  - **URL**: **Uniform Resource Locator** – specifikuje umístění zdroje

    `https://4iz110.vse.cz/docekal`

Struktura URI



RFC 3986

Uniform Resource Locator

`protokol://doménové_jméno:port/cesta?dotaz#kotva`

- `protokol` – protokol pro přenos dat (http, https, mailto, ftp, atd.)
- `doménové_jméno:port` – doménové jméno (*host name*) serveru, kde se zdroj nachází
  - lze nahradit IP adresou, popřípadě i s číslem portu
  - lze specifikovat uživatele: **`uzivatel@domena:port`**
- `cesta` – cesta ke zdroji na serveru
- `dotaz` (*query*)– parametry pro skript na serveru:

  `?nazev=hodnota&nazev2=hodnota2`
- `kotva` (*fragment, anchor*) – odkaz na „fragment", tedy část zdroje – odkaz na nějakou část stránky

Neprůhlednost URI

- z **URI** by agenti získávající zdroje **neměli** dělat žádné předpoklady o **typu** zdroje/obsahu
  - přípony nic neznamenají!
- informace o typu zdroje/obsahu se předává v položce **`Content-type`** protokolu **`HTTP`**

HTTP

HyperText Transfer Protocol

Michal Dobšovič

HyperText Transfer Protocol

- vývoj od 90. let
- původní protokol pro jednoduché (vědecké) dokumenty
- transakční protokol (dotaz → odpověď)
- verze
  - **`0.9`** rok 1991
  - **`1.0`** rok 1996
  - **`1.1`** rok 1997
  - **`2.0`** publikováno v roce 2015 v RFC 7540
  - **`3.0`** navrhovaný standard (2023), RFC 9114

HTTP 0.9

„jednořádkový protokol”

HTTP 0.9

- původní, první verze protokolu
- *jednořádkový protokol*
- dotaz:

  `GET SP Request-URI CRLF`

  - kde **`Request-URI`** může být celkové **URI** a nebo jen absolutní cesta
- odpověď:
  - data zdroje (obsah souboru)
- nevýhody:
  - **žádná** metadata, **žádné** hlavičky
  - neznalost klienta o **délce obsahu**

HTTP 0.9

- Připojení na **`TCP`** port **`80`** webového serveru
- Požadavek: **`GET /kde/index.html`**
- Odpověď: obsah souboru **`index.html`**

HTTP 0.9 transakce

```
GET /
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
  <head>
    <title>Titulek stránky</title>
  </head>
  <body>
    <h1>Důležitá stránka</h1>
    <p>Na které ovšem nic není.</p>
  </body>
</html>
```

HTTP 0.9 dotaz

```
GET /
```

HTTP 0.9 odpověď

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
  <head>
    <title>Titulek stránky</title>
  </head>
  <body>
    <h1>Důležitá stránka</h1>
    <p>Na které ovšem nic není.</p>
  </body>
</html>
```

HTTP 1.0

HTTP 1.0

- základ pro současnou podobu protokolu
  - metadata, hlavičky, stavové kódy, atd.
- protokol z roku 1996, specifikace
- dotaz:
  - *Řádek dotazu*
  - *hlavičky*
  - *(prázdný řádek) CRLF*
  - *Tělo (volitelné)*

```
GET / HTTP/1.0
Host: www.vse.cz
```

Struktura HTTP 1.0 dotazu a odpovědi

dotaz

- **request line**
  řádek dotazu
- **general header**
  obecná hlavička
- **request header**
  hlavička dotazu
- **entity header**
  entitní hlavička
- **CRLF**
- tělo dotazu

odpověď

- **status line**
  stavová řádka
- **general header**
  obecná hlavička
- **response header**
  hlavička odpovědi
- **entity header**
  entitní hlavička
- **CRLF**
- tělo odpovědi

HTTP 1.0: Odpověď

- **Stavová řádka**
- **Obecná hlavička**
- **Hlavička odpovědi**
- **Entitní hlavička**
- **CRLF**
- **Tělo**

```
HTTP/1.1 200 OK
Date: Wed, 06 Apr 2022 18:49:34 GMT
Server: Apache
Content-Type: text/html
Last-Modified: Sun, 07 Nov 2021 18:14:27 GMT

<!DOCTYPE html>
<html lang="cs">
<head>
  <meta charset="utf-8">
  <title>Titulek stránky</title>
  <link rel="stylesheet" href="mujuzasnystyl.css">
</head>
<body>
    <h1>Moje úžasná stránka</h1>
    <p>Můj ještě úžasnější odstavec.</p>
</body>
</html>
```

HTTP 1.0: Odpověď

- Stavová řádka
- Obecná hlavička
- Hlavička odpovědi
- Entitní hlavička
- CRLF
- Tělo

```
HTTP/1.1 200 OK
Date: Wed, 06 Apr 2022 18:49:34 GMT
Server: Apache
Content-Type: text/html
Last-Modified: Sun, 07 Nov 2021 18:14:27 GMT

<!DOCTYPE html>
<html lang="cs">
<head>
  <meta charset="utf-8">
  <title>Titulek stránky</title>
  <link rel="stylesheet" href="mujuzasnystyl.css">
</head>
<body>
    <h1>Moje úžasná stránka</h1>
    <p>Můj ještě úžasnější odstavec.</p>
</body>
</html>
```

Řádek dotazu

`Metoda SP Request-URI SP verze CRLF`

- **Metoda** požadavku: **GET**, **POST** nebo **HEAD**
- **SP** = space = mezera
- **Request-URI** – URI požadavku
- **verze** – verze protokolu HTTP
- **CRLF** = ukončení řádku znaky CR a LF

Řádek dotazu

Metoda SP Request-URI SP verze CRLF

- **Metoda** požadavku: **GET**, **POST** nebo **HEAD**
- **SP** = space = mezera
- **Request-URI** – URI požadavku
- **verze** – verze protokolu HTTP
- **CRLF** = ukončení řádku znaky CR a LF

Metody v HTTP 1.0

- **GET** – získá daný zdroj (stránku / soubor)
- **POST** – zasílání dat na server (anotace, odesílání zpráv, vložení záznamu do databáze, odeslání formuláře)
- **HEAD** – žádost pouze o hlavičku bez těla (a tedy bez obsahu zdroje)

Obecná hlavička dotazu

- Obecná část hlavičky může obsahovat:
- **Date:**
  - informace o čase, formát uveden v definici protokolu
- *Pragma*:
  - specifická hlavička pro implementačně-specifické direktivy
  - v HTTP 1.0 je definováno pouze `Pragma: no-cache`
  - v aktuálních specifikacích HTTP 1.1 se od ní upouští

Hlavička dotazu

- **Authorization:** – slouží k autentikaci vůči serveru
- **From:** – e-mailová adresa klienta (moc se nepoužívá)
- **If-Modified-Since:** – podmíněné stažení obsahu v případě, že na serveru je novější obsah, jinak 304
- **Referer:** – odkud jsme byli na tuto stránku odkázáni
- **User-Agent:** – identifikace prohlížeče

Entitní hlavička

- **Allow:** – které metody jsou podporované daným serverem
  - `Allow: GET, POST`
- **Content-Encoding:** – kódování obsahu
  - `Content-Encoding: x-gzip`
- **Content-Length:** – délka obsahu
  - `Content-Length: 3495`

Entitní hlavička

- **Content-Type:** – typ obsahu
  - `Content-Type: text/html`
- **Expires:** – kdy obsah vyprší a měl by se znovu stáhnout
  - `Expires: Thu, 01 Dec 1994 16:00:00 GMT`
- **Last-Modified:** – kdy byl obsah naposledy změněn
  - `Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT`

HTTP 1.1

HTTP 1.1

- rozšíření metod
- rozšíření hlaviček
  - hlavička **`Host:`** je nyní povinná!
- možnost trvalého spojení a pipelining
- RFC 2068 rok 1997 (RFC 2616, rok 1999)
  - specifikace
- doplnění v roce 2014 a 2022 (RFC 9110)
- současná specifikace opouští termíny jako entita, entitní hlavička či entitní tělo

Trvalé spojení

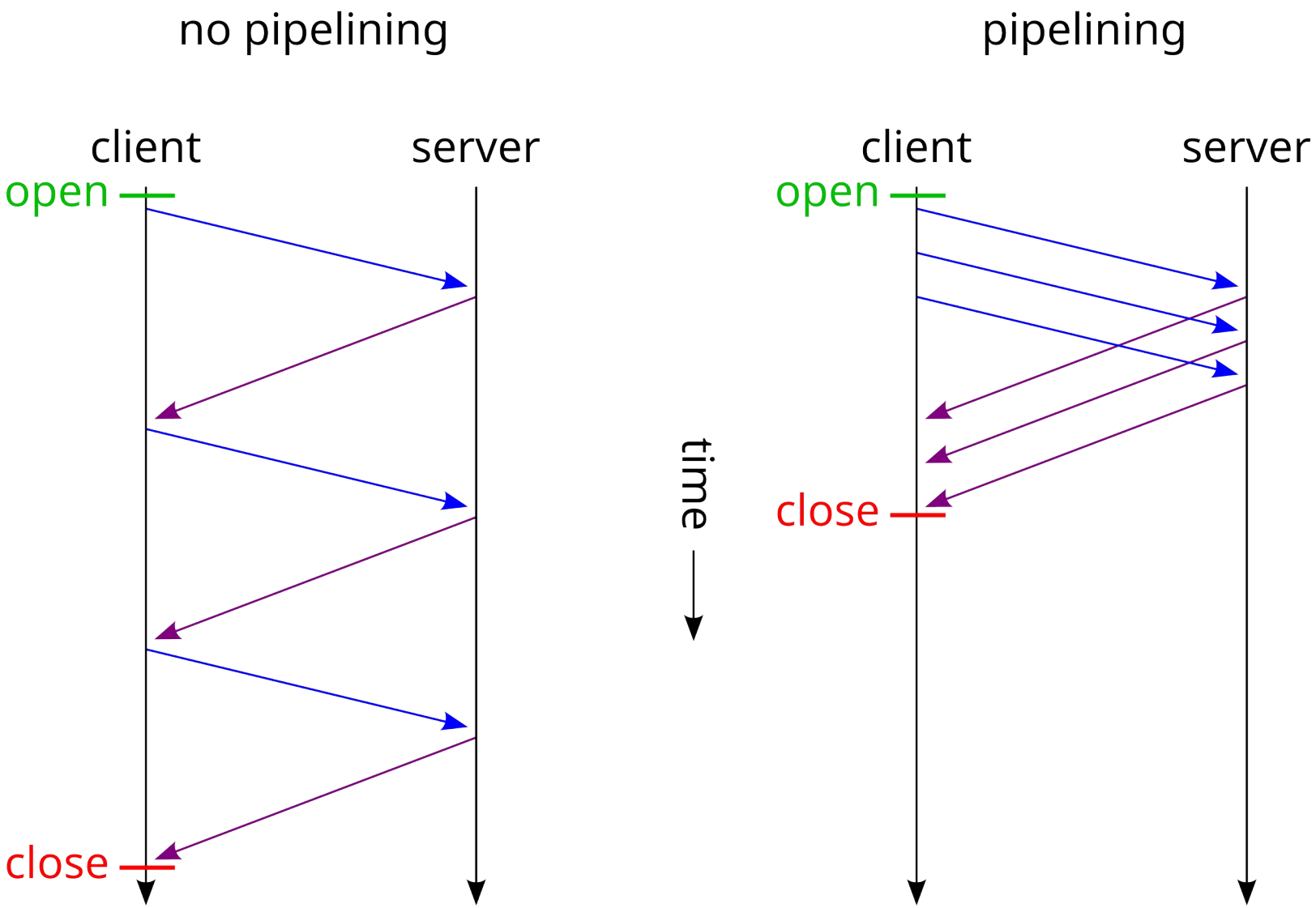
- předchozí verze protokolu HTTP vytvářely pro každý objekt nové spojení
- pokud tedy např. HTML stránka obsahovala čtyři obrázky, pro její stažení bylo potřeba navázat 5 spojení (1 stránka + 4 obrázky)
- HTTP/1.1 proto spojení mezi klientem a serverem po vyřízení požadavku neuzavírá, ale umožňuje jej použít pro přenos více objektů
- spojení může ukončit klient nebo server tím, že do požadavku/odpovědi zařadí hlavičku:

```
Connection: close
```

Pipelining

- místo odeslání dotazu a čekání na odpověď, než odešlu další dotaz, odešlu více dotazů naráz
- Pipelining tohle umí přes jedno konkrétní TCP spojení

Pipelining


no pipelining
client
server
open
close
time
pipelining
client
server
open
close

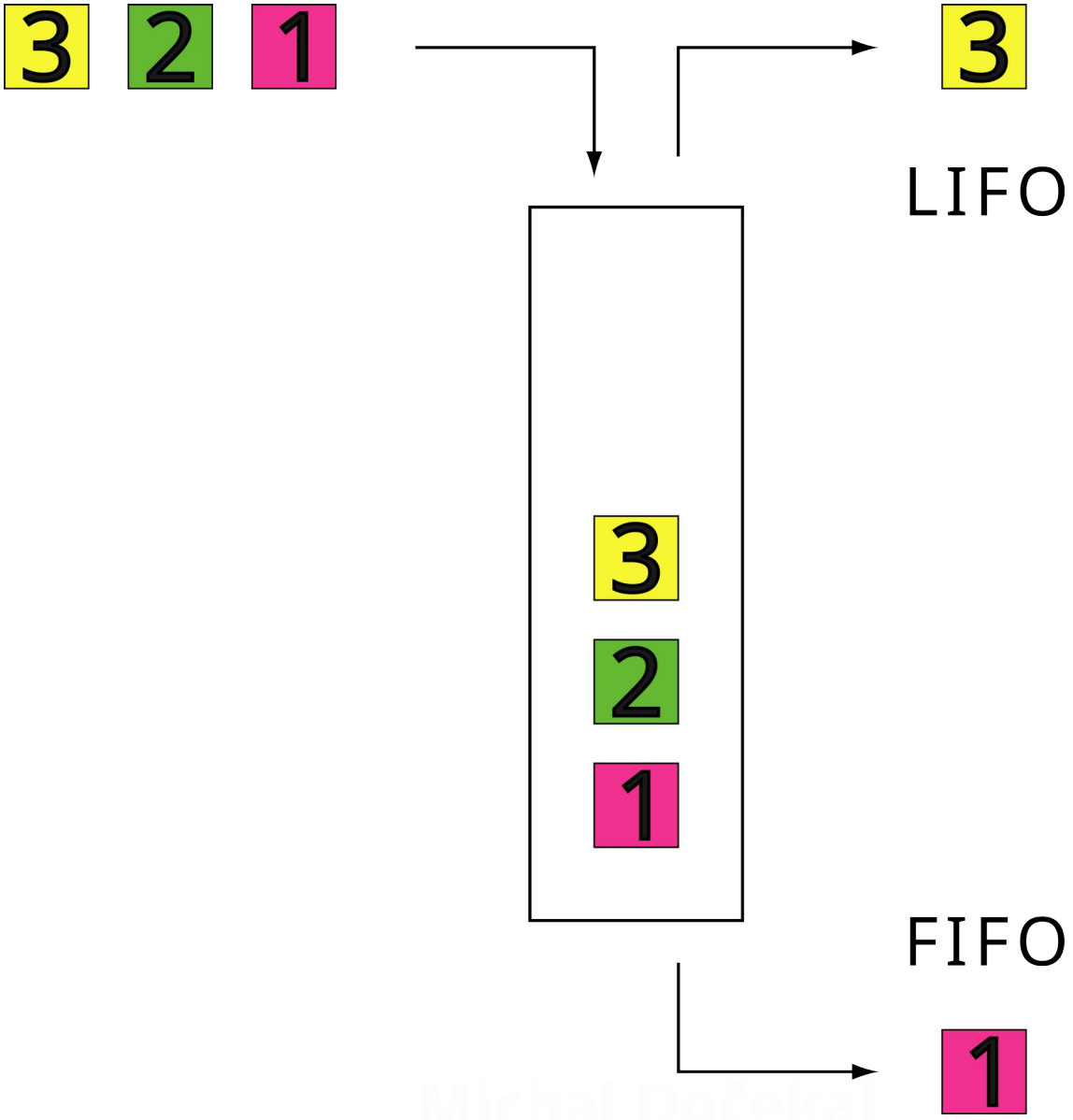
Blokování čela fronty

- odpovědi na dotazy u pipeliningu musí následovat ve stejném pořadí, v jakém došly požadavky: **`FIFO`** (First-In-First-Out) fronta
- **Head of Line Blocking** *(blokování čela fronty)*
  - pošleme 6 požadavků, ale první se zpozdí a server bude vyžadovat delší čas na jeho zpracování
  - ostatní požadavky jsou vyřízené, ale nemohou být odeslané klientovi, protože server musí vracet odpovědi v pořadí požadavků
  - první požadavek tedy celou frontu zablokuje a hotové odpovědi dorazí, až když se vyřídí první požadavek
- HOLB řeší HTTP/2 na úrovni aplikační vrstvy, zůstává HOLB na úrovni transportní vrstvy (TCP), což řeší HTTP/3

Odbočka: fronty

FIFO: first in, first out (jako fronta lidí)

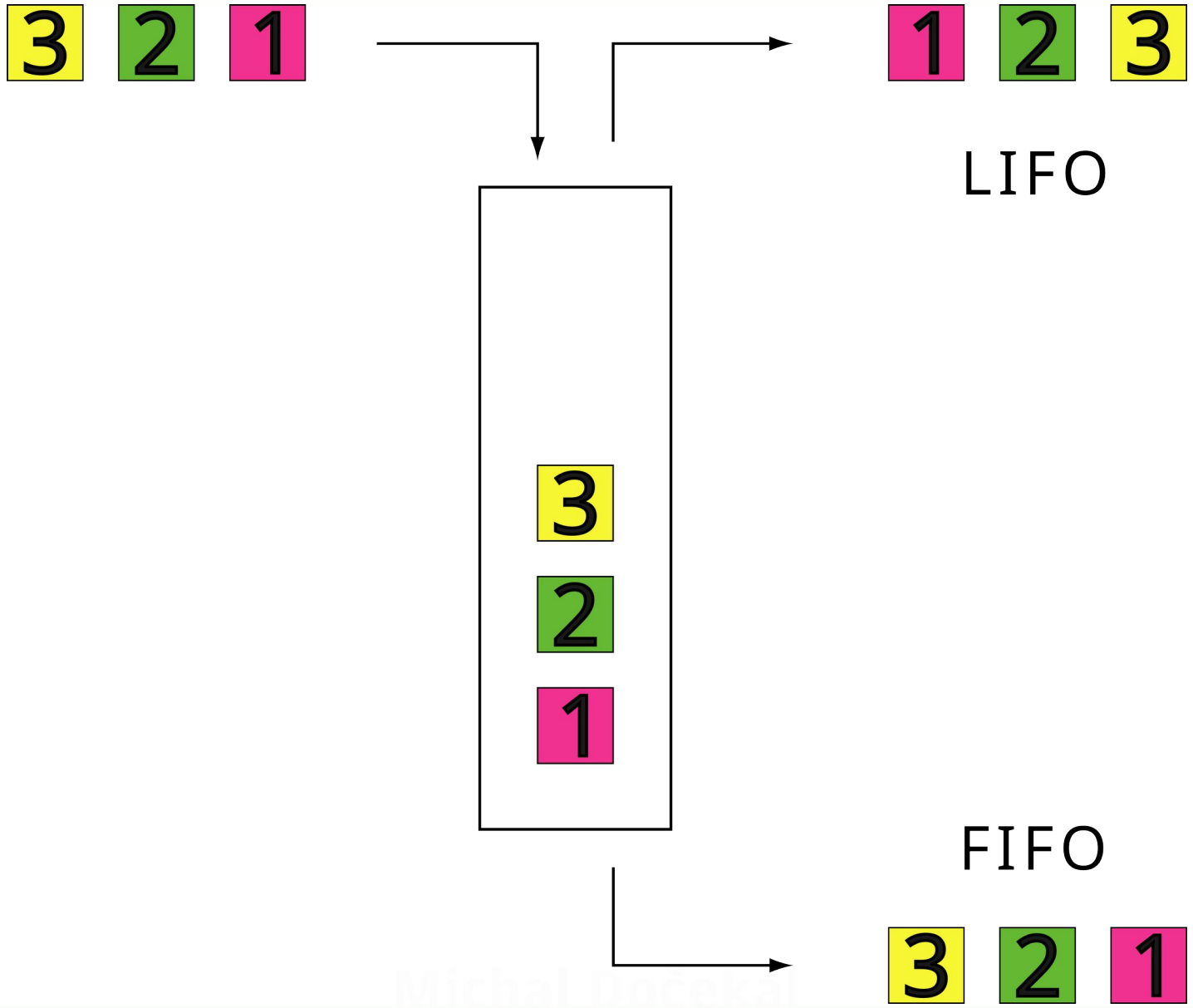
LIFO: last in, first out (zásobník)



Odbočka: fronty

FIFO: first in, first out (jako fronta lidí)

LIFO: last in, first out (zásobník)



Rozšíření metod

- *Zůstávají:*

  GET

  POST

  HEAD

- Nové:

  OPTIONS

  PUT

  DELETE

  TRACE

  CONNECT

Nové metody

- **OPTIONS** – klient může požádat server, ať mu sdělí možnosti komunikace (podporované metody požadavku), server vrací mj. hlavičku **Allow** (s podporovanými metodami)
- **PUT** – metoda pro vytvoření nebo přepsání zdroje na serveru (obvykle vypnuto)
- **DELETE** – metoda pro smazání zdroje ze serveru (také obvykle vypnuto)
- PUT a DELETE

Nové metody

- **CONNECT** – používá se spolu s proxy serverem, který se může dynamicky přepnout na tunel (SSL tunelování)
- **TRACE** – cílový příjemce požadavku (server) odesílá kopii požadavku zpět odesílateli (klientovi), aby zjistil, co na něm případné mezičlánky (servery na cestě) mění

HTTP 1.1 dotaz (GET)

```
GET / HTTP/1.1

Host: 4iz110.vse.cz

Connection: keep-alive

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74
Safari/537.36

Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,imag
e/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9
```

HTTP 1.1 dotaz (POST)

```
POST https://insis.vse.cz/system/login.pl
Host: insis.vse.cz
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:98.0) Gecko/20100101
Firefox/98.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Origin: https://insis.vse.cz
Connection: keep-alive
Referer: https://insis.vse.cz/auth/?lang=cz

lang=cz&username=xname00&password=mojetajneheslo
```

HTTP 1.1 hlavičky odpovědi

```
HTTP/1.1 200 OK
Server: Apache
Date: Wed, 06 Apr 2022 18:49:34 GMT
Content-Type: text/html
Last-Modified: Sun, 07 Nov 2021 18:14:27 GMT
Transfer-Encoding: chunked
Connection: keep-alive
ETag: W/"61881783-813"
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Security-Policy: default-src 'self' 'unsafe-inline';
frame-ancestors 'self'; block-all-mixed-content
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Encoding: gzip
```

Obecné hlavičky HTTP/1.1

- **Cache-Control** ovládání kešování
- **Connection** ovládání spojení
- **Date** čas dotazu/odpovědi
- ~~Pragma~~
- Trailer
- Transfer-Encoding kódování pro přenos těla
- Upgrade protokolu (např. HTTP 1.1 → 2.0)
- Via přidáváno proxy servery
- ~~Warning~~

Hlavičky dotazu HTTP/1.1

- **Accept**
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization
- Expect
- From
- **Host**
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Proxy-Authorization
- Range
- **Referer**
- TE
- **User-Agent**

Accept hlavičky – co klient podporuje

```
Accept: text/plain;q=0.5, text/html, text/x-dvi;q=0.8
Accept-Charset: iso-8859-2, unicode-1-1;q=0.8
Accept-Encoding: gzip, deflate, br
```

- **Accept**: - klientem podporované typy obsahu (content type)
- *Accept-Charset*: klientem podporované znakové sady – už se nepoužívá, preferuje se UTF-8, které je široce podporované
- **Accept-Encoding**: klientem podporované kódování obsahu (typicky komprimace)

Důležité hlavičky dotazu

- **Referer** - adresa, ze které byl požadován aktuálně požadovaný zdroj (odkud jsme sem přišli) - potenciální narušení soukromí

```
Referer: https://4iz110.vse.cz/docekal
```

- **User-Agent** - identifikace prohlížeče, včetně OS, apod. - další potenciální narušení soukromí

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64;
rv:47.0) Gecko/20100101 Firefox/47.0
```

- **Host** - specifikuje hostitele (resp. jeho doménové jméno), popřípadě číslo portu
  - **hlavička `Host:` je nyní povinná!**

```
Host: krkavec.net
```

Hlavička `Host` a virtual hosts

- jeden server může hostovat více webových prezentací – je ale třeba je od sebe odlišit
  - IP adresou (IPv4 adres málo, není dobrý nápad)
  - jménem
- **více domén** ukazuje A/AAAA záznamem na **jednu IP adresu** HTTP serveru
- v hlavičce `Host:` řekne klient serveru, k webu které domény chce přistupovat
- podle jména vrátí server příslušnou prezentaci

Problém: SSL

- pro šifrované spojení je třeba ověřit identitu serveru (certifikát)
- certifikát je vydávaný pro doménu
- je-li na serveru více virtual hostů, je třeba vrátit certifikát pro požadovanou doménu
- ale tu se z hlavičky `Host:` server dozví až PO vytvoření šifrovaného spojení
- a co teď?

Server Name Indication

- rozšíření protokolu TLS
- umožňuje specifikovat požadované jméno před uzavřením šifrovaného spojení
- server tak může vrátit odpovídající certifikát
- problém: jméno se předává nešifrovaně – information leak!
  - využíváno pro cenzurování obsahu internetu
  - Encrypted Client Hello (ECH) v TLS 1.3 problém řeší, ale dosud moc nerozšířeno, může být blokováno

(Entitní) hlavička

- **Allow**
- **Content-Encoding**
- Content-Language
- **Content-Length**
- Content-Location
- Content-MD5
- Content-Range
- **Content-Type**
- **Expires**
- **Last-Modified**
- extension-header

Důležité hlavičky

- **Allow** - vypisuje zdrojem podporované metody
- **Content-Encoding** - vypisuje jakákoliv kódování, která byla aplikována na obsah a v jakém pořadí (v HTTP to jsou kompresní metody)
- **Content-Length** - indikuje velikost těla zprávy v bytech
- **Content-Type** - typ obsahu, jako u MIME
- **Expires** - datum a čas, po kterém je odpověď považována za expirovanou
- **Last-Modified** - datum a čas, kdy původní server věří, že zdroj byl modifikovaný

Hlavička odpovědi

- Accept-Ranges
- Age
- **ETag** identifikátor verze zdroje
- Location
- Proxy-Authenticate
- Retry-After
- **Server** identifikace serveru (SW)
- Vary
- WWW-Authenticate

HTTP kešování (caching)

- obsah přenášený v rámci HTTP se může kešovat, aby nemusel být přenášený při každém načtení stránky
- typy keší (cache)
  - soukromé (private) - keš klienta (typicky prohlížeče)
  - sdílené (shared) - mezi serverem a klientem, proxy keš nebo managed keš (CDN, reverzní proxy, atd.)
- keš typicky řídí **server** pomocí hlavičky **Cache-Control**, kterou říká klientovi, jak a dokdy má odpověď kešovat

Expirační pravidla

- dobu platnosti, a nutnost revalidace řídí **server**
- používá hlavičku **`Expires`** nebo direktivu **`max-age`**, která je součástí hlavičky **`Cache-Control`**
- pokud server nic neuvádí, kešuje se heuristicky
  - např. uvádí-li hlavička **`Last-Modified`** rok staré datum, můžeme předpokládat, že se hned tak nezmění, takže můžeme kešovat
  - heuristické kešování se používalo před rozšířením podpory pro **`Cache-Control`**, dnes by všechny **`HTTP`** odpovědi měly mít hlavičku **`Cache-control`**

Řízení cache

- keš musí vracet nejčerstvější keší drženou odpověď, která odpovídá **`HTTP`** dotazu a zároveň platí jedna z podmínek:
  - **odpověď je dostatečně čerstvá**
    - lhůtu čerstvosti řídí direktiva **`max-age`** hlavičky **`Cache-Control`** nebo hlavička **`Expires`** (typické pro `HTTP 1.0`)
  - **odpověď byla revalidována** na serveru
    - pokud vypršela doba čerstvosti, provede se validace dotazovou hlavičkou **`If-Modified-Since`** nebo **`If-None-Match`**
    - tím se ověří, že se na serveru obsah nezměnil, takže se znovu nestahuje a obsah se vrátí z keše
  - server odpoví **stavovým kódem `303`**, **`304`** nebo **chybou**

Direktivy pro řízení keše

- Omezení, co je pro keš, nastolí **server**:
  - **public** - lze uložit do sdílené keše
  - **private** - pouze do "sourkromé" keše (v prohlížeči)
  - **no-cache** - lze kešovat, ale je třeba vždy provádět validaci
- Omezení, co může být vloženo do keše
  - klient i server: **no-store** - nelze kešovat

Direktivy pro řízení keše

- Modifikace expiračního mechanismu – klient i server:
  - **`max-age`** klient akceptuje odpověď, která není starší než tuto dobu
  - **`min-fresh`** klient chce odpověď, která zůstane čerstvá po danou dobu
  - **`max-stale`** klient akceptuje expirovanou odpověď, která není starší než daná doba
- Ovládání revalidace a aktualizace – klient
- Ovládání transformací entit (např. konverze obrázků pro redukci velikosti): `no-transform`

Direktivy pro požadavky

- "no-cache"
- "no-store"
- "max-age" "=" delta v sekundách
- "max-stale" ["=" delta v sekundách]
- "min-fresh" "=" delta v sekundách
- "no-transform"
- "only-if-cached"
- cache-extension

Direktivy pro odpovědi

- "public"
- "private" ["=" <"> 1#field-name <">]
- "no-cache" ["=" <"> 1#field-name <">
- "no-store"
- "no-transform"
- "must-revalidate"
- "proxy-revalidate"
- "max-age" "=" delta v sekundách
- "s-maxage" "=" delta v sekundách
- cache-extension

Stavové kódy 1xx

- Informational 1xx
  - `100` Continue
  - `101` Switching Protocols

Stavové kódy 2xx

- Successful 2xx
  - **200 OK** *HTTP požadavek úspěšný*
  - 201 Created
  - 202 Accepted
  - 203 Non-Authoritative Information
  - 204 No Content
  - 205 Reset Content
  - 206 Partial Content

Stavové kódy 3xx

- Redirection 3xx
  - 300 Multiple Choices — *víc možností, vyber si*
  - **301 Moved Permanently** — *zdroj přemístěn jinam*
  - 302 Found
  - 303 See Other
  - **304 Not Modified** — *zdroj je čerstvý (keš)*
  - 305 Use Proxy
  - 306 (Reserved)
  - **307 Temporary Redirect** — *dočasné přesměrování*

Stavové kódy 4xx

- Client Error 4xx
  - **400 Bad Request** *neplatný požadavek*
  - **401 Unauthorized** *klient neautorizován*
  - 402 Payment Required
  - **403 Forbidden** *server nemá přístup ke zdroji*
  - **404 Not Found** *zdroj nenalezen*
  - **405 Method Not Allowed** *nepovolená HTTP metoda*
  - 406 Not Acceptable
  - 407 Proxy Authentication Required
  - 408 Request Timeout

Stavové kódy 4xx

- Client Error 4xx
  - 409 Conflict
  - **410 Gone** — *dříve to tu bylo, teď už to tu* ***není a nebude***
  - 411 Length Required
  - 412 Precondition Failed
  - **413 Request Entity Too Large** — *příliš velký* ***HTTP*** *požadavek*
  - **414 Request-URI Too Long** — *příliš dlouhé požadované* ***URI***
  - 415 Unsupported Media Type
  - 416 Requested Range Not Satisfiable
  - 417 Expectation Failed

Stavové kódy 5

- Server Error 5xx
  - **500 Internal Server Error** — došlo k chybě na straně serveru
  - 501 Not Implemented
  - **502 Bad Gateway** — server se chová jako proxy, ale cílový server mu v časové lhůtě neodpověděl
  - 503 Service Unavailable
  - 504 Gateway Timeout
  - 505 HTTP Version Not Supported

Předávání parametrů

- webové aplikace potřebují nastavení určitých proměnných (např. jméno a heslo pro přihlášení, ID článku pro zobrazení, atd.)
- řeší se předáváním:
  - **GET**: v URL požadavku – information leak!

    `https://krkavec.net?login=petr&password=rot13vajler`
  - **POST**: v těle požadavku

    `https://insis.vse.cz/system/login.pl`

Předávání parametrů metodou GET

- potenciální information leak (nevhodné pro hesla)
- třeba nahradit některé znaky
  - mezeru nahradí **+**
  - znaky se spec. významem nahradí **%XX**
  - kde **XX** je hexadecimální hodnota znaku
  - např. **%2F** je lomítko
  - podobné kódování *quoted printable* ze **SMTP**, ale místo **=** se použije **%**

Předávání parametrů metodou POST

- parametry jsou součástí požadavku, neobjevují se v adresním řádku prohlížeče:

```
POST https://insis.vse.cz/system/login.pl
Host: insis.vse.cz
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Referer: https://insis.vse.cz/auth/?lang=cz

lang=cz&username=docm00&password=jaradjatra
```

Cookies RFC 2109

- cookie je informace **iniciovaná serverem** a **uložená na klientovi** k pozdějšímu použití
- je to **textová informace**
- zavedena do protokolu firmou Netscape.

Příklad cookie

- **server** pošle cookie v HTTP odpovědi a klient (prohlížeč) si ji uloží

```
Content-type: text/html
Set-Cookie: foo=bar; path=/; expires datum
```

- **klient** následně připojuje hlavičku Cookie se všemi cookies ke každému požadavku:

```
Content-type: text/html
Cookie: foo=bar
```

Příklad cookie

- **server** pošle každou z více cookies v samostatné hlavičce HTTP odpovědi:

```
Content-type: text/html
Set-Cookie: login=docm01; path=/; secure
Set-Cookie: id=4a5437a551; path=/; secure
```

- **klient** následně připojuje hlavičku Cookie se všemi cookies ke každému požadavku:

```
Content-type: text/html
Cookie: login=docm01; id=4a5437a551
```

Struktura cookie

- **Jméno** *název cookie*
- **Hodnota** *hodnota cookie*
- `Expires=datum` *maximální doba života cookie*
- `Path=cesta` *cesta, která musí být v URI, aby byla cookie vrácena*
- `Domain=doména` *doména, pro níž a pro jejíž subdomény bude cookie vrácena*
- `Secure` *cookie bude vrácena jen přes HTTPS spojení*
- `HttpOnly` *cookie nebude k dispozici pro klientské skriptování (JavaScript)*

→

Session cookies

- webové aplikace vytváří relace (session) pro dané klienty, zejména pro ty přihlášené
- to vyžaduje jednoznačnou identifikaci klienta (prohlížeče), aby nedošlo k záměně (stejná IP adresa, jiný klient – NAT, atd.)
- řeší se typicky přes session cookies (relační sušenky)
- bez session cookies nebude fungovat většina webových aplikací

Session cookies

- relační sušenka uloží do prohlížeče *identifikátor*, dle kterého přiřadí server klienta k dané session (relaci), která je vytvořená na serveru

  `PHPSESID=abbh0ib2mbjujph51evnr2n48p`

- do session na serveru lze ukládat data, která jsou pak neviditelná pro klienta (a klientem nemodifikovatelná)

  `admin=true`

- některá data je vhodné ukládat do klienta (prohlížeče), k tomu se pak použijí normální cookies

  `theme=dark`

- session cookies mají nastavenou platnost do zavření prohlížeče *(ale funkce obnovení relace prohlížeče obnoví po znovuotevření prohlížeče i session cookies!!!)*

3rd party cookies

- umožňují sledování uživatelů napříč webem
- jedna webová stránka může obsahovat komponenty načítané z jiných serverů
- stažení takové komponenty vyžaduje HTTP požadavek a odpověď, jejichž součástí může být vytvoření a uložení cookie
  - to je pak cookie třetí strany
- třetí strana si může vytvořit cookie s jednoznačným identifikátorem, dle kterého pozná váš prohlížeč

3rd party cookies

- z hlavičky **`Referer:`** pozná, ze které stránky vedl odkaz na komponentu této třetí strany
  - a tudíž zjistí, jakou stránku že jste si to načetli
- když pak navštívíte jinou stránku se stejnou komponentou, dojde k načtení cookie a třetí strana zjistí, že jste to vy, kdo si před chvílí prohlížel jinou stránku
- Google měl v plánu zrušit 3rd party cookies v Chromu ke konci roku 2023, posunuto na konec 2024, Firefox je již blokuje

3rd party cookies

- reklama platí mnoho webových služeb "zdarma" (platíte svým soukromím)
- jelikož za cílenou reklamu se platí mnohem více než za necílenou, je velký zájem o sledování uživatelů webu
- existují již různé alternativy a různé možnosti, jak 3rd party cookies nahradit
  - FLoC (Federated Learning of Cohorts), opuštěno
  - Topics API - prohlížeč zaznamenává zájmy uživatele, které pak sdílí s navštívenými stránkami
  - jiné možnosti - browser fingerprinting, identifikace uživatele přes ID sdílené se sledovacími službami, **evercookie**, apod.

HTTP/2

- stejné metody, stavové kódy, hlavičky, URI
- co je nového: jak jsou data předávána transportnímu protokolu
  - **komprese** hlaviček
  - **prioritizace** požadavků
  - **zmenšení** (minifying)
  - **pipelining** nahrazen **multiplexováním** požadavků (paralelní zpracování na úrovni aplikačního protokolu), ale HOLB u TCP
  - server „push“ – server může posílat klientovi zdroje před tím, než o ně požádá
- `RFC 7540`, rok 2015
- **`11/2023`** – **36%** ↓ serverů, **98%** ↑ klientů

HTTP/3

- `RFC 9114`, stejné metody, kódy, pole hlaviček
- místo TCP je použit protokol **`QUIC`** (RFC 9000) nad **`UDP`**
  - Řeší blokaci čela fronty (head-of-line blockling)
  - **`HTTP/1.1`** má *pipelining*, ale požadavky se zpracovávají v pořadí, tj. když se čeká na první, ty následující čekají také
  - **`HTTP/2`** řeší tento problém multiplexováním požadavků v rámci jednoho spojení, což eliminuje blokaci čela fronty na aplikační úrovni, ale ne na transportní (TCP) úrovni, kde ztráta 1 paketu zbrzdí celou frontu
  - **`HTTP/3`** řeší tento problém výměnou transportní vrstvy (TCP za QUIC), která tento problém nemá
- **`11/2023`**: navrhovaný standard, **`27%`** serverů, **`95%`** klientů

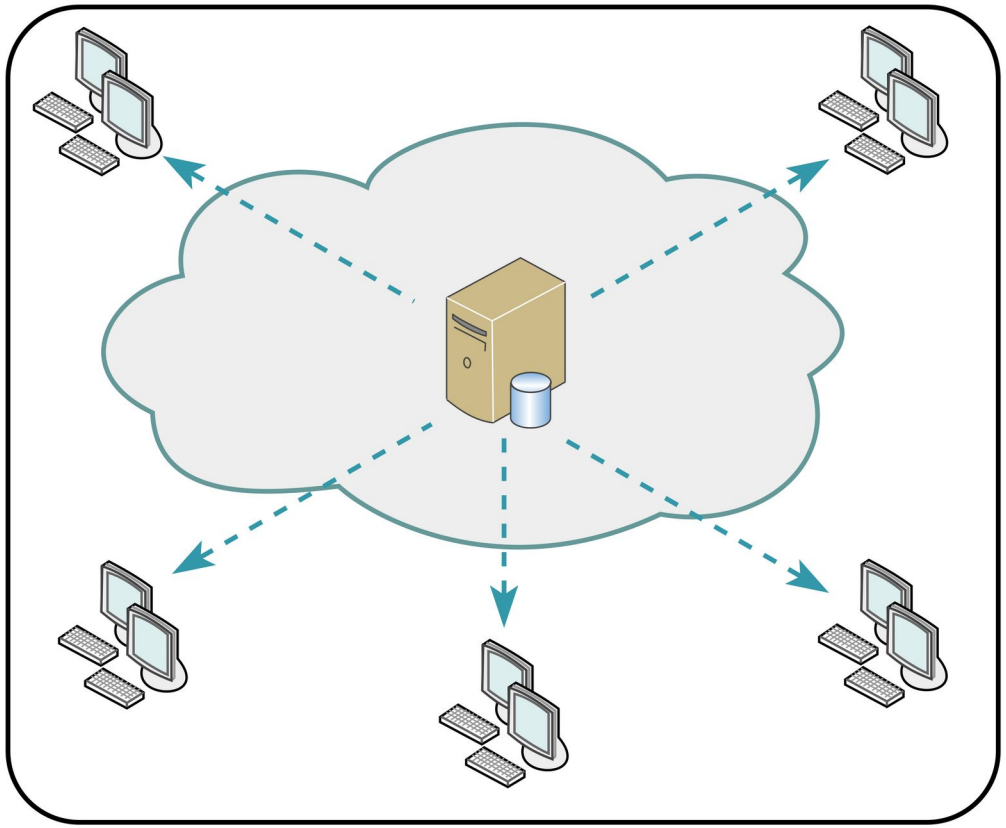
Řešení s vysokou dostupností

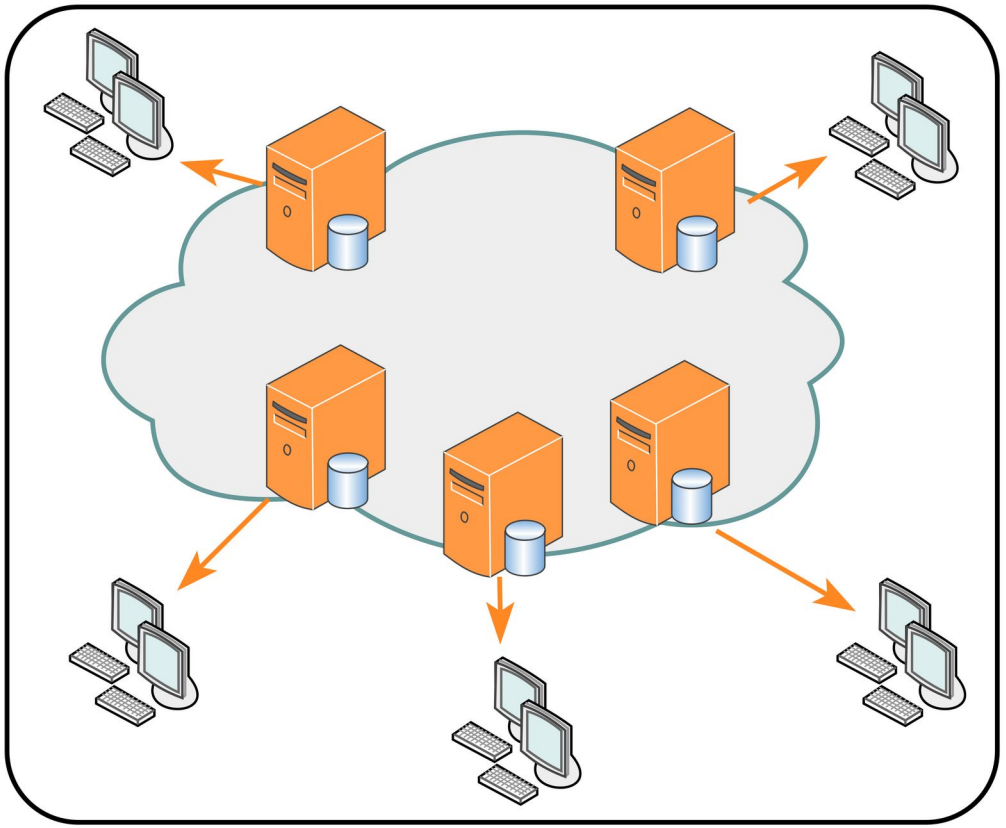
- **Load balancer** – distribuuje zatížení po celé (funkční) infrastruktuře
- **Serverová farma** – X webových serverů vyřizujících požadavky, 1+ databázových serverů s replikací
- **DNS záznam** vede na load balancer
- Balancer pouze **předává požadavky** funkčním serverům (proxy) a vrací odpovědi klientům
- Balancer může odkazovat klienty na cílové servery (třeba na úrovni DNS)

CDN

- CDN: Content Delivery/Distribution Network
  - velké služby využívají CDN pro distribuci obsahu
  - místo jednoho zdrojového systému, který poskytuje data různě rychle v závislosti na geografické odloučenosti klienta se obsah replikuje na servery rozmístěné po různých částech světa
  - klient se pak nějakým směrovacím mechanismem, který je součástí CDN, spojí vždy s geograficky nejbližší replikou, a ta mu data poskytne
  - různé typy CDN služeb pro různý obsah, různý princip
  - = distribuovaná keš: vysoká dostupnost, vysoký výkon

CDN





HTML

Hyper**T**ext
Markup
Language

HTML

- HTML je značkovací jazyk pro tvorbu webových stránek, a jejich propojení pomocí hypertextových odkazů
- dnes doplněno řadou dalších technologií (CSS, serverové skripty, klientské skripty, atd.)
- vznik kolem roku 1990

Základní struktura HTML

- značky v ostrých závorkách

  `<p> … </p>`

- Dvě základní části

```
<html>
<head></head>
<body></body>
</html>
```

Některé značky

- `<title></title>` - součást hlavičky, nadpis do horní lišty
- `<h1>`nadpis 1`</h1>`
- `<a href="http://www.w3.org">`odkaz na konsorcium W3C`</a>`
- `<ul></ul>` nečíslovaný seznam
- `<ol></ol>` číslovaný seznam
- `<li></li>` položka seznamu

Pozor na podvodné odkazy

```
<a href="https://krkavec.net">https://insis.vse.cz/</a>
```

https://insis.vse.cz/

Pozor na podvodné odkazy

```
<a href="https://krkavec.net">https://insis.vse.cz/</a>
```

https://insis.vse.cz/

Reálný příklad podvodu:

```
<a href="https://insis-vse.cz">https://insis.vse.cz</a>
```

Pozor na podvodné odkazy

```
<a href="https://krkavec.net">https://insis.vse.cz/</a>
```

https://insis.vse.cz/

Reálný příklad podvodu:

```
<a href="https://insis-vse.cz">https://insis.vse.cz</a>
```

- `insis-vse.cz:` doména 2. řádu zaregistrovaná podvodníkem
- `insis.vse.cz:` doména 3. řádu ve vlastnictví VŠE

Reprezentace informací

- HTML – Hypertext Markup Language
  - standardní jazyk popisu stránek
- XML – EXtensible Markup Language
  - nemá definované tagy a je určen více k popisu informací, často je používán k výměně dat mezi různými systémy
- XHTML – Extensible Hypertext Markup Language
  - na XML založený následovník HTML

Společné prvky HTML a XHTML

- značkovací jazyky ve WWW reprezentují informace o charakteru textu značkami
  - párové, nepárové
- otevírací značka `<slovo>`
- uzavírací značka `</slovo>`
- značky jsou vzájemně vnořené
  - ** **Tohle je potřeba **<u>**podtrhnout**</u> **

Rozdílné prvky XHTML a HTML

- XHTML oproti HTML
  - striktnější požadavky
  - správné zahloubení elementů (elementy se nesmějí překrývat):
    - ✔ `<x>..<y>..</y></x>`
    - ✘ `<x>..<y>..</x><x>..</y></x>`
    - ✔ `<x>..<y>..</y></x><x><y>..</y></x>`
  - case-sensitive, malá písmena
  - XHTML povinně vyžaduje ukončení tagů

Rozdílné prvky XHTML a HTML

- XHTML oproti HTML
  - striktnější požadavky
  - správné zahloubení elementů (elementy se nesmějí překrývat):

✔ <x>..<y>..</y></x>

✘ <x>..<y>..</x><x>..</y></x>

✔ <x>..<y>..</y></x><x><y>..</y></x>

  - case-sensitive, malá písmena
  - XHTML povinně vyžaduje ukončení tagů

Reprezentace informací

- HTML5 – HTML verze 5
  - nové tagy pro sémantické strukturování stránek (`<section>`, `<article>`, `<main>`, `<aside>`, `<header>`, `<footer>`, `<nav>`, `<figure>`, atd.)
  - větší podpora pro média (vestavěné přehrávače pro audio a video)
  - vestavěná podpora pro vektorovou grafiku
  - perzistentní úložiště pro data webových stránek
  - není tak striktní a case-sensitive jako XHTML
  - od HTML 5.1 je součástí standardu DRM

Příklad XHTML dokumentu

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cz">
  <head>
    <title>Já jsem titulek</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <style>
    body {
         background: #fff;
         color: #000;
         }
    h1,h2 {
         font: bold italic 1.2em "Liberation Sans", sans-serif;
         color: blue;
         }
    </style>
  </head>
  <body>
    <h1>Hlavička v tučná modrá italika</h1>
    <p>Normální text.</p>
    <h2 style="color: red; background: yellow;">
     Toto bude v tučné červené italice na žlutém pozadí;
     style pro h2 definované nahoře je částečně překryt.
     </h2>
    <p>Normální text.</p>
    <h2>Toto bude tučná modrá italika</h2>
    <p>Normální text.</p>
  </body>
</html>
```

Příklad HTML5 dokumentu

```
<!DOCTYPE html>
<html lang="cs">
  <head>
    <title>Já jsem titulek</title>
    <meta charset="UTF-8">
    <style>
    body {
         background: #fff;
         color: #000;
         }
    h1,h2 {
         font: bold italic 1.2em "Liberation Sans", sans-serif;
         color: blue;
         }
    </style>
  </head>
  <body>
    <h1>Hlavička v tučná modrá italika</h1>
    <p>Normální text.</p>
    <h2 style="color: red; background: yellow;">
     Toto bude v tučné červené italice na žlutém pozadí;
     style pro h2 definované nahoře je částečně překryt.
     </h2>
    <p>Normální text.</p>
    <h2>Toto bude tučná modrá italika</h2>
    <p>Normální text.</p>
  </body>
</html>
```

Příklad HTML5 renderovaný prohlížečem (engine Gecko)

Hlavička v tučná modrá italika

Normální text.

Toto bude v tučné červené italice na žlutém pozadí; style pro h2 definované nahoře je částečně překryt.

Normální text.

Toto bude tučná modrá italika

Normální text.

Cascading Style Sheets

- CSS umožňuje definovat vzhled i umístění jednotlivých HTML elementů (či elementů XML dokumentu)
- oddělení obsahu dokumentu od definice vzhledu
- většinou separátní CSS soubory, na které je odkaz v HTML hlavičce
- styl (style sheet) se skládá ze série pravidel, pravidla mají jeden či více selektorů (selector = na co se bude pravidlo vztahovat), kterým je pak přidělen deklarační blok (declaration block = opět sada pravidel [vlastnost (property), hodnota (value)] pro formátování obsahu)

Cascading Style Sheets (příklad)

```
body {                          /* selektor: HTML body */
    font-family: sans-serif;    /* typ písma: bezpatkové */
    background-color: #000000; /* barva pozadí: černá */
    color: #999999;             /* barva textu: světle šedá */
}
h1 a, h1 a:visited {           /* selektor: linky v nadpisech */
    text-decoration: none;      /* dekorace textu: bez podtržení*/
    color: #DDDDDD;             /* barva textu: světlejší */
}
h1 a:hover {                    /* selektor: linky při nájezdu myší */
    color: #DDDDDD;              /* barva textu: světlejší */
    font-weight: bold;           /* řez písma: tučně */
    text-decoration: underline; /* dekorace textu: s podtržením */
}
```

Webové aplikace

Michal Ročkai

Webové aplikace (skriptování na serveru a/nebo na klientovi)

- místo statického obsahu můžeme vracet obsah dynamický
- skriptovací jazyky na straně serveru
  - přijmou data z formuláře (požadavek POST) či z URL (požadavek GET)
  - zpracují je
  - na jejich základě se v případě potřeby obrátí na další informační systémy (např. databázový server)
  - výsledky zpracují do HTML/XHTML/XML stránky, kterou odešlou klientovi

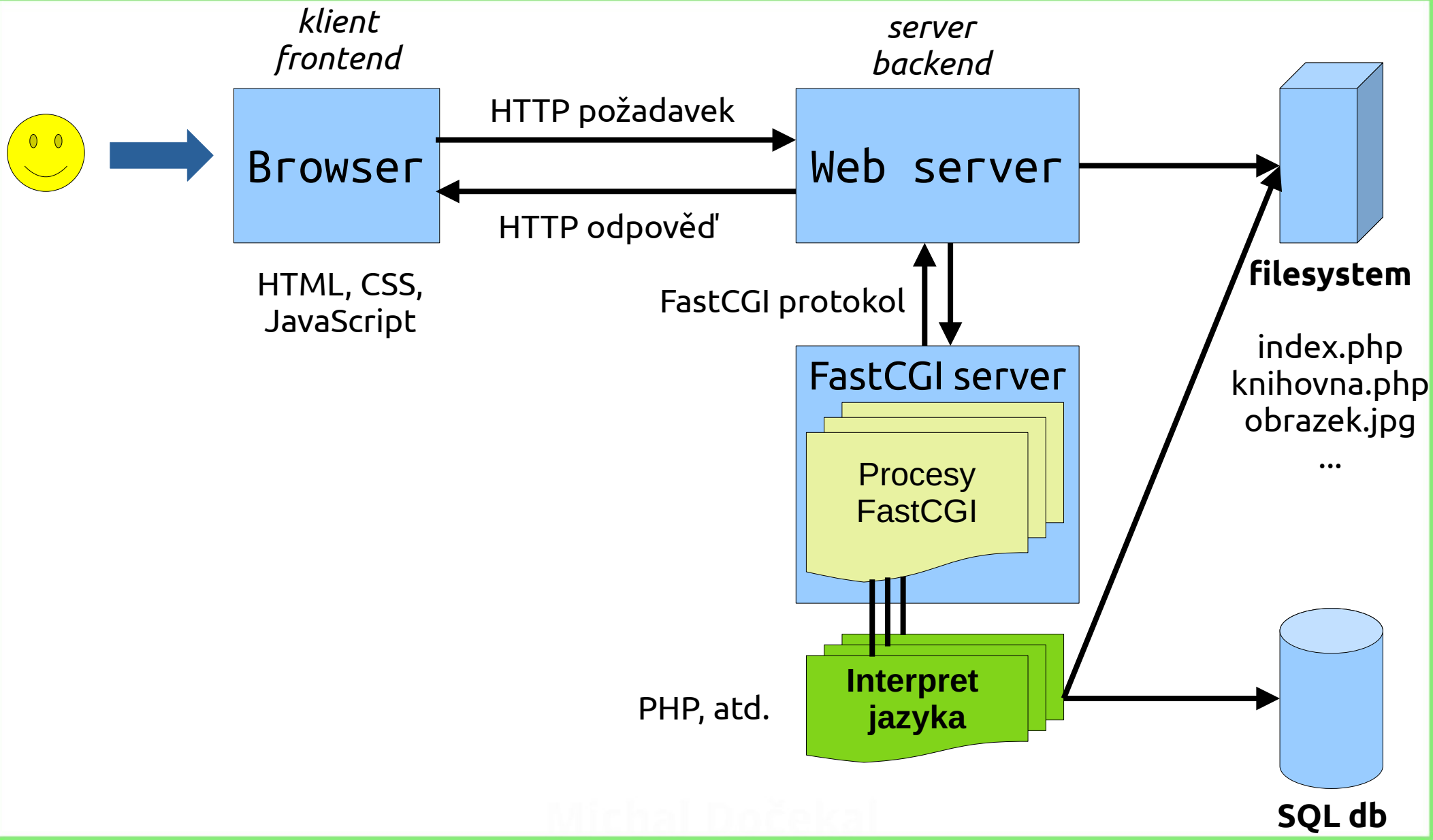
Webové aplikace (skriptování na serveru a/nebo na klientovi)

- skriptovací jazyky na straně serveru
  - PHP (PHP: Hypertext Preprocessor)
  - ASP (Active Server Pages)
  - JSP (Java Server Pages)
- skriptovací jazyky na straně klienta
  - **JavaScript**
  - VBScript

Interface mezi webovým serverem a skriptovacím jazykem

- **CGI** a jeho varianty
  - **C**ommon **G**ateway **I**nterface
    - původní model interakce s aplikací
    - server uloží parametry do proměnných prostředí (environment variables) a zavolá aplikaci
    - server vrátí zcela naformátovanou stránku, vygenerovanou s pomocí skriptovacího jazyka
  - **FastCGI**
    - místo nového procesu pro každý požadavek se vytvoří perzistetntní procesy, které zpracovávají více požadavků, binární protokol
- **Apache API**
  - MPM (Multi-Processing Module)
- atd.

Webové aplikace: obecné schéma



PHP

- vznik 1993, vydáno 1995
- stále nejrozšířenější skriptovací jazyk pro webové aplikace
- procedurální i objektově orientovaný
- v poslední době velký posun vpřed (PHP 8.x), i za cenu zpětně nekompatibilních změn
- ukázka →

datum.php

```
<!DOCTYPE html>
<html lang="cs">
  <head>
    <title>Datum</title>
    <style>
    body { background: #222; color: white; }
    </style>
  </head>
  <body>
    <h2>Datum</h2>
    <p>Dnešní datum: <?php echo date("j");?>. <?php echo
date("n");?>. <?php echo date("Y");?></p>
  </body>
</html>
```

datum.php

```
<!DOCTYPE html>
<html lang="cs">
  <head>
    <title>Datum</title>
    <style>
    body { background: #222; color: white; }
    </style>
  </head>
  <body>
    <h2>Datum</h2>
    <p>Dnešní datum: 22. 11. 2023</p>
  </body>
</html>
```

jmeno.php

```php
<?php

if (empty($_COOKIE['jmeno'])&&!empty($_POST['name'])) {
   $jmeno=trim($_POST['name']);
   if (strlen($jmeno)>64) {
      die("Prilis dlouhe jmeno.");
   }
   setcookie('jmeno', $jmeno, time()+600, '', '', true, true);
}

if (empty($jmeno)&&!empty($_COOKIE['jmeno'])&&strlen(trim($_COOKIE['jmeno']))<=64) {
   $jmeno=trim($_COOKIE['jmeno']);
}

?><!DOCTYPE html>
<html lang="cs">
  <head>
    <title>Jméno</title>
    <style>
    body { background: #222; color: white; }
    </style>
  </head>
  <body>
    <h2>Jméno</h2>
<?php if (!empty($jmeno)) { echo "    <p>Vaše jméno je: ".htmlspecialchars($jmeno)."</p>\n"; } else {?>
    <form action="<?php echo basename(__FILE__);?>" method="POST">
       <label for="name">Zadejte své jméno:</label><br>
       <input type="text" id="name" name="name" value="Kilroy was here">
       <input type="submit" value="Odeslat">
    </form>
<?php } ?>
  </body>
</html>
```

jmeno.php

```php
<?php

if (empty($_COOKIE['jmeno'])&&!empty($_POST['name'])) {
   $jmeno=trim($_POST['name']);
   if (strlen($jmeno)>64) {
      die("Prilis dlouhe jmeno.");
   }
   setcookie('jmeno', $jmeno, time()+600, '', '', true, true);
}

if (empty($jmeno)&&!
empty($_COOKIE['jmeno'])&&strlen(trim($_COOKIE['jmeno']))<=64) {
   $jmeno=trim($_COOKIE['jmeno']);
}

?><!DOCTYPE html>
...
```

jmeno.php

```
<html lang="cs">
  <head>
    <title>Jméno</title>
    <style>
    body { background: #222; color: white; }
    </style>
  </head>
  <body>
    <h2>Jméno</h2>
<?php if (!empty($jmeno)) { echo "    <p>Vaše jméno je:
".htmlspecialchars($jmeno)."</p>\n"; } else {?>
    <form action="<?php echo basename(__FILE__);?>" method="POST">
       <label for="name">Zadejte své jméno:</label><br>
       <input type="text" id="name" name="name" value="Kilroy was here">
       <input type="submit" value="Odeslat">
    </form>
<?php } ?>
  </body>
</html>
```

Další jazyky pro tvorbu webů

- Java Server Pages (JSP)
  - jazyk Java (neplést s JavaScriptem!)
  - objektový přístup
  - opět se vkládá do HTML stránky **<%! Java %>**
- Active Server Pages (ASP)
  - Microsoft, prováděno při volání uvnitř IIS (Internet Information Services) - webový server Microsoftu
  - implicitní programovací jazyk: VBScript
  - překvapivě se vkládá do HTML stránky **<% ASP %>**

JSP - příklad

```
<html>
<head>
<basefont face="Arial">
</head>
<body>
<center>Turning The Tables, JSP-Style!</center>
<br>
<%!
// define the number
int number = 7;
int x;

// use a for loop to calculate tables for that number
for (x=1; x<=15; x++) {
    out.println(number + " X " + x + " = " + (number*x) + "<br>");
}
%>
</body>
</html>
```

ASP - příklad

```
<html>
<body>
<%
response.write("<h2>Headline</h2>")
%>

<%
response.write("<p style='color:#0000ff'>Current time:"+Now()+"</p>")
%>
</body>
</html>
```

Skriptování na straně klienta

- moderní webové aplikace staví nejen na serverových skriptech, ale i na klientských
- primárně interpretovaný jazyk **JavaScript**
- všechny majoritní prohlížeče mají JavaScriptový engine, který vykonává skripty webových stránek
- pozor: **Java** a **JavaScript** jsou dvě naprosto odlišné záležitosti!

AJAX

- **A**synchronous **J**avaScript **A**nd **X**ML
  - skriptování na straně serveru: požadavek → odpověď, interakce se stránkou → nový požadavek → nová odpověď a znovunačtení stránky
  - AJAX umožňuje úprava obsahu stránky bez nutnosti jejího znovunačtení
    - asynchronní operace
    - načtení dat ze serveru pomocí objektu `XMLHttpRequest` a dynamická úprava části obsahu stránky
  - moderní implementace mnohdy pro výměnu dat využívají formát JSON místo XML

cas.php

```php
<?php

echo date('Y-m-d H:i:s', time());
```

ajax.php

```
<!DOCTYPE html>
<html>
<body>
  <h1>AJAX demo</h1>
     <p>Toto je demo asynchronního JavaScriptu.</p>
     <img src="spice.png"/>
     <h2>Přesný čas</h2>
        <p>Přesný čas <span id="demo">při načtení stránky je <?php require('cas.php');?>.</span></p>
        <button type="button" onclick="loadDoc()">Načti čas asynchronně</button>
  <script>
    function loadDoc() {
      const xhttp = new XMLHttpRequest();
      xhttp.onload = function() {
        document.getElementById("demo").innerHTML = 'načtený asynchronně je '+this.responseText+'.';
      }
      xhttp.open("GET", "cas.php");
      xhttp.send();
    }
  </script>
</body>
</html>
```

Webové frameworky

- „čisté" jazyky typu PHP paradoxně nemusí být nejvhodnější pro tvorbu webových aplikací
- frameworky jsou hotová řešení pro tvorbu webových aplikací často s řešením mnoha problémů, které se vyskytly v mateřském jazyce
  - mohou podporovat konkrétní návrhový vzor (např. MVC: model-view-controller)
- mnoho věcí, které programátorům usnadní práci (šablonovací systémy, abstrakce nad databází, automatické ošetření vstupů, apod.)

Webové frameworky (výběr)

- Skriptování na serveru
  - ASP - ASP.NET MVC
  - Java - Spring
  - PHP - CakePHP, CodeIgniter, Laravel, Nette, Symfony, Zend, Slim, atd.
  - Python - Django, Flask
  - Ruby - Ruby on Rails, Sinatra
  - JavaScript - node.js
  - Erlang - Yaws
- Skriptování na klientovi
  - JavaScript - jQuery, React, Angular, atd.

XML

- XML je jednoduchý, velmi flexibilní formát založený na jazyku SGML (ISO 8879)
- HTML je zaměřeno na zobrazení dat a soustřeďuje se na to, jak data vypadají
- XML je zaměřeno na popis dat a soustřeďuje se na to, jaká data to jsou
  - vhodné pro uložení či výměnu strukturovaných dat
  - mnoho datových formátů je založených na XML (např. OpenDocument, SVG, apod.)
  - některé komunikační protokoly jsou založené na XML (např. SOAP či XMPP, dále např. AJAX)

XML vlastnosti

- je **značkovací jazyk** podobně jako HTML
- je vytvořen pro **popis dat**
- **nemá definované tagy** – uživatel si je vytváří sám
- používá různé mechanismy pro **popis dat** (DTD, XML Schema, RELAX NG, atd.)

XML jako výměnný formát

- XML tvoří cestu k obecnému **výměnnému formátu**
- Vytváří:
  - **Hierarchickou strukturu informací**
  - **Struktury pro kontrolu validity dat**
  - **Struktury pro prezentaci dat**

Hierarchická struktura

- klasické relační databáze nabízejí strukturu tabulek, tj. jednoúrovňové rozdělení dat
- XML nabízí libovolně strukturovanou množinu
- v hierarchické struktuře lze zachytit jak tabulky tak i strukturu jako ISO 2709 (bibliografický popis)

Příklad

```
<book>
	<chapter>
		<title>Úvod</title>
	</chapter>
	<chapter>
		<title>Příběh</title>
		<subChapter>
			<title>Díl 1</title>
		</subChapter>
		<subChapter>
			<title>Díl 2</title>
		</subChapter>
	</chapter>
	<chapter>
		<title>Obsah</title>
	</chapter>
</book>
```

Kontrola a specifikace struktury

- volnost při tvorbě e současně anarchie
- pro výměnu dat potřebujeme dohodu o struktuře dat (dokumentu) a ne vyměňovat nahodilé sady značek
- definici značek vytvářejí jazyky pro popis struktury dokmentu - nazývají se schémata a je jich mnoho
- např. WML (wap), XHTML, MathML, DocBook

XML schéma

- základní funkce schématu je ve formální definici značkovacího jazyka
- značkovací jazyk je možné chápat i jako definici datového modelu
- schéma jednoznačně definuje, jak může dokument vypadat, lze jej tedy použít pro validaci
- DTD, XML Schema, RELAX NG

DTD

- Document Type Definition
- původní definice dat
- součást specifikace XML
- podporuje velké množství aplikací

Příklad DTD

```
<!ELEMENT zamestnanci (zamestnanec+)>
<!ELEMENT zamestnanec (jmeno, prijmeni, narozeni)>
<!ELEMENT jmeno       (#PCDATA)>
<!ELEMENT prijmeni    (#PCDATA)>
<!ELEMENT narozeni    (#PCDATA)>
```

Nevýhody DTD

- DTD je psáno naprosto jinou syntaxí než ostatní XML dokument.
  - definice shodnou syntaxí může používat formátovací jazyky jak XSLT na výstup a tisk.
- DTD nepodporuje jmenné prostory
  - pokud vytváříme dokument v XHTML a rádi bychom vnořili vzorec v MathML máme problém
- DTD nemá datovou kontrolu
  - SGML bylo vytvořeno pro textové dokumenty
  - v současnosti se XML používá i pro datovou výměnu
  - pro účinnou validaci je třeba nejen zanoření elementů ale i datové typy

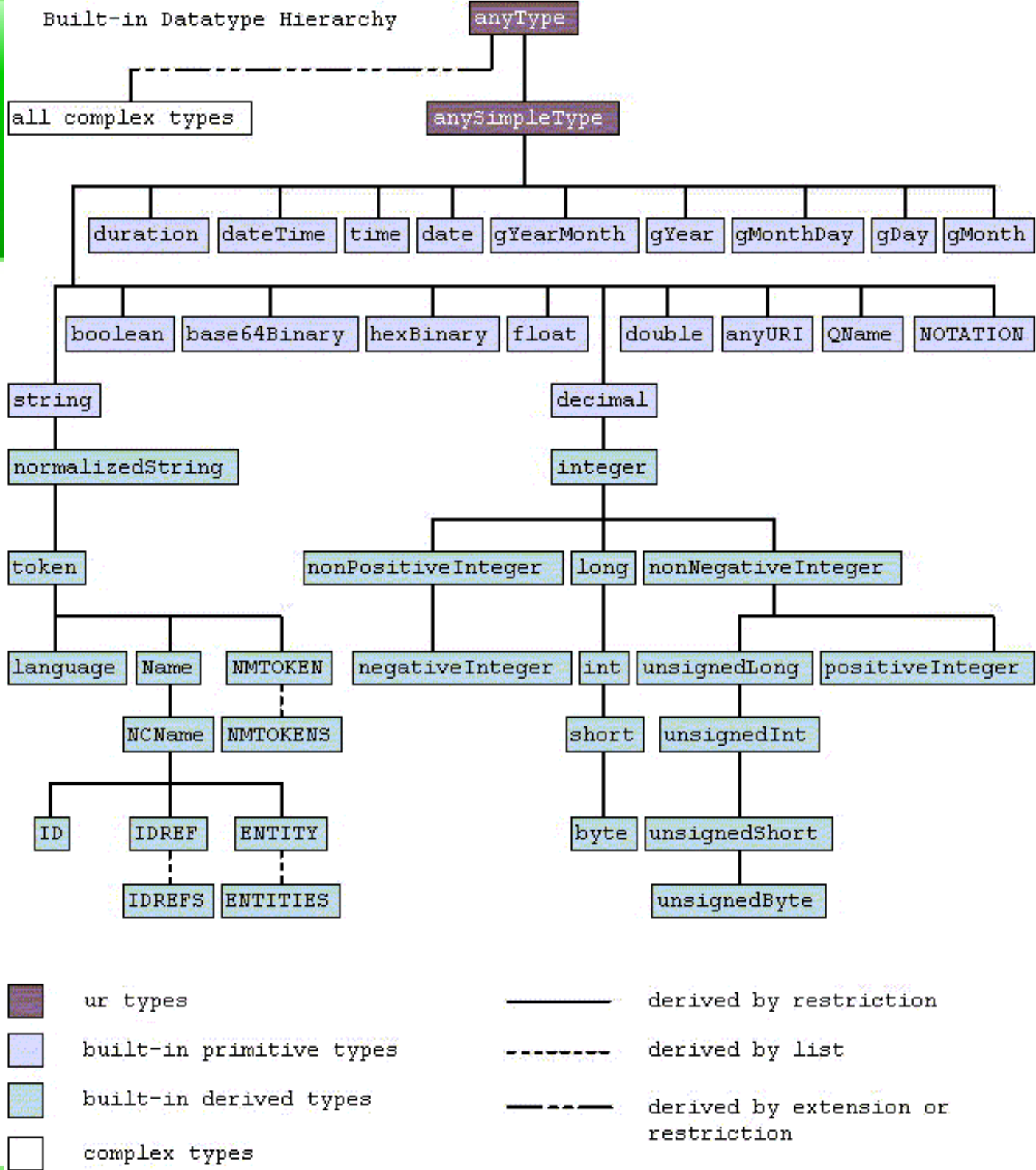
XML Schema

- vznikl postupným vývojem přes XML-Data a XDR (XML Data Reduced)
- přijat v roce 2001 jako doporučení konsorcia W3C
- po DTD nejpoužívanější schémový jazyk
- poměrně složitý

Příklad

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="zamestnanci">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="zamestnanec" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="jmeno" type="xs:string"/>
              <xs:element name="prijmeni" type="xs:string"/>
              <xs:element name="narozen" type="xs:date"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

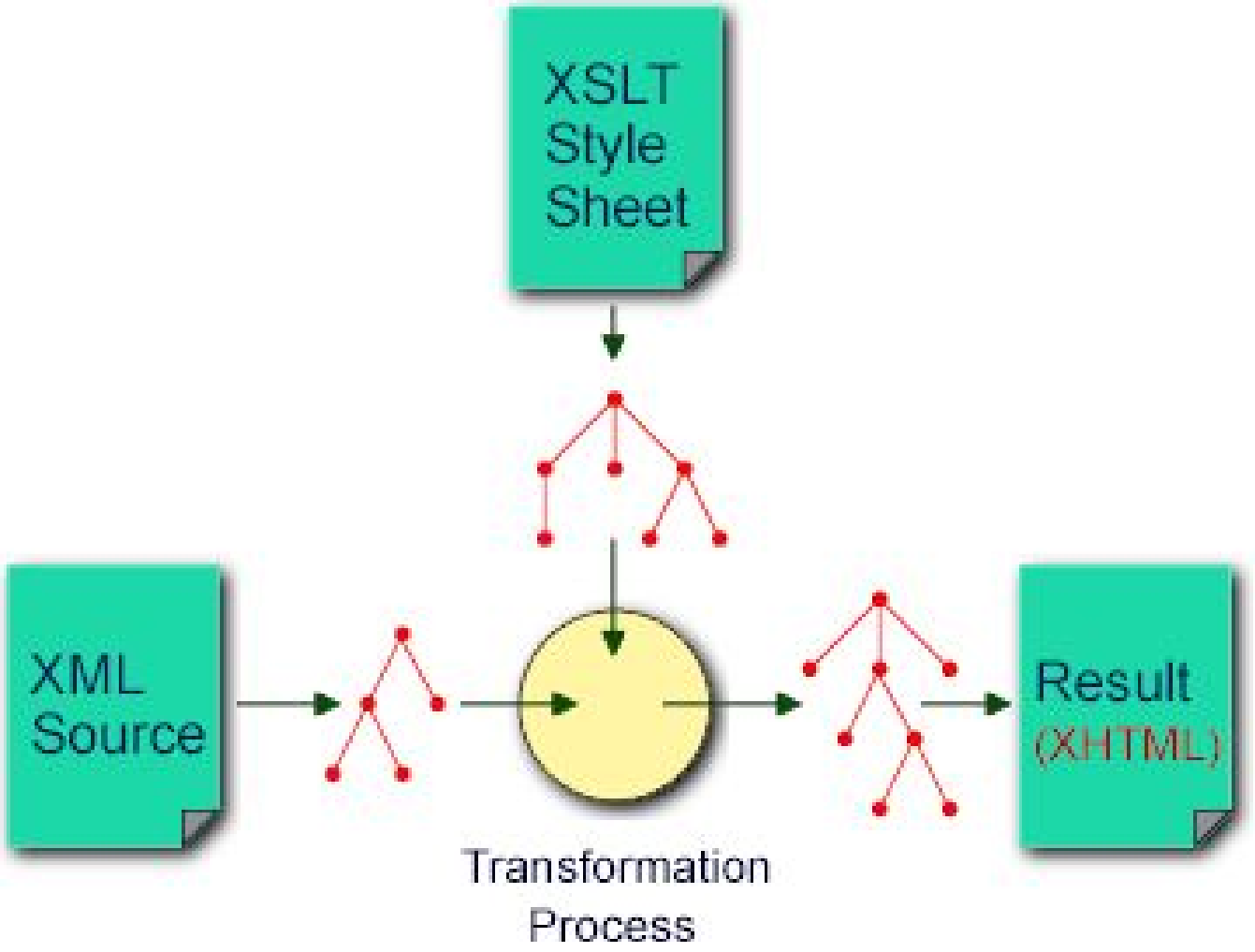
XML Schema datové typy



XSL stylesheet

- XML nedefinuje jak se mají objekty zobrazovat.
- Pro zobrazení se používají dva formátovací jazyky.
  - CSS (káskádové styly)
  - XSL (eXtensible Stylesheet Language)
- Druhý z nich se rozdělil na
  - XSLT (XSL Transformations) zodpovídá za transformaci do např HTML nebo jiného XML nebo textového dokumentu
  - XSL FO (formátovací objekty) provádí přímé formátování do výsledného tvaru

XSLT


XSLT
Style
Sheet
XML
Source
Result
(XHTML)
Transformation
Process

XSLT

Příklad: generování XML souboru

Původní XML dokument

```
<?xml version="1.0" ?>
<lidi>
  <clovek login="JN1">
    <jmeno>Jan</jmeno>
    <prijmeni>Novák</prijmeni>
  </clovek>
  <clovek login="MN1">
    <jmeno>Marta</jmeno>
    <prijmeni>Nollová</prijmeni>
  </clovek>
</lidi>
```

XSL stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="/lidi">
    <root>
      <xsl:apply-templates select="clovek"/>
    </root>
  </xsl:template>

  <xsl:template match="clovek">
    <jmeno login="{@login}">
      <xsl:value-of select="jmeno" />
    </jmeno>
  </xsl:template>

</xsl:stylesheet>
```

Vygenerované XML

```
<?xml version="1.0"?>
<root>
  <jmeno login="JN1">Jan</jmeno>
  <jmeno login="MN1">Marta</jmeno>
</root>
```

XSLT

Příklad: generování XHTML souboru

Původní XML dokument

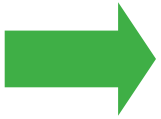
```
<?xml version="1.0" ?>
<lidi>
  <clovek login="JN1">
    <jmeno>Jan</jmeno>
    <prijmeni>Novák</prijmeni>
  </clovek>
  <clovek login="MN1">
    <jmeno>Marta</jmeno>
    <prijmeni>Nollová</prijmeni>
  </clovek>
</lidi>
```

XSL stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/1999/xhtml">

  <xsl:output method="xml" indent="yes" encoding="UTF-8"/>

  <xsl:template match="/lidi">
    <html>
      <head> <title>XML příklad</title> </head>
      <body>
        <h1>Lidé</h1>
        <ul>
          <xsl:apply-templates select="clovek">
            <xsl:sort select="prijmeni" />
          </xsl:apply-templates>
        </ul>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="clovek">
    <li>
      <xsl:value-of select="prijmeni"/><xsl:text>, </xsl:text><xsl:value-of select="jmeno"/>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

Vygenerované XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XML příklad</title>
  </head>
  <body>
    <h1>Lidé</h1>
    <ul>
      <li>Nollová, Marta</li>
      <li>Novák, Jan</li>
    </ul>
  </body>
</html>
```



Lidé

- Nollová, Marta
- Novák, Jan

Kam dál s XML?

- https://www.kosek.cz/xml/
- předmět 4IZ238: XML - Teorie a praxe značkovacích jazyků

<EOF>