Webové Aplikace – vyřešené otázky k maturitě

- **Autor:** Karel Čermák, info@k-cermak.com.
- **Ročník:** 2023.
- **Repozitář:** https://github.com/K-cermak/SPSE-Maturita.
- **Práva:** Materiály autor zveřejňuje bez záruk a pouze pro osobní nekomerční použití. Šíření těchto materiálů je povoleno pouze s původním (nezměněným) ponecháním této stránky či sdílením na oficiální repozitář uvedený výše.
- **Tip:** Pro rychlou orientaci v kapitolách lze použít klávesovou zkratku CTRL + F a přejít do funkce Nadpisy.

- **Donate:** Pokud ti mé materiály pomůžou a jsi ochoten ocenit moji snahu nějakou kačkou, můžeš tak učinit přes QR kódy níže: ❤️❤️❤️
  - **Účet:** 2262692018/3030

  - **Crypto:**
    - **BTC:** bc1qasgxc552wjqlpcm9vt7ucmw6p4zuz007dxh8n4
    - **ETH:** 0x29Ca9054B2241aB39010a1434fb50e504EE10871
    - **LTC:** ltc1qxpgp3jc5jyem6096n48w48qqrwsrnj5eq9j890
    - **ADA:** addr1q8c89cet02nyql4ygj96s0cz5ntusgzxfzuykfngmaf0zt2ftj7wrayqm7dx52et7k7tkjjl2edan0wykww6q4twn79shzx8vn
    - **DOGE:** DCYFq9hPcVJkYKAgttkRXNSAkfbjEmLGdo

  - A pokud jsi chudý student a všechny prachy prochlastáš, můžeš mi alespoň dát hvězdičku na GitHub repozitář...

1. HTML – struktura dokumentu, textové prvky stránky

- *Popište strukturu HTML dokumentu, informace v hlavičce dokumentu (znaková sada, externí a interní CSS styly, informace pro vyhledávače).*
- *Uveďte příklady párových a nepárových tagů.*
- *Popište HTML tagy pro nadpisy, odstavce, seznamy a jejich CSS vlastnosti.*
- *Popište vnořené seznamy a jejich použití pro vyjížděcí menu.*
- *Popište kontextuální definice stylů pro vnořené seznamy.*

- **Rozdělené tagů**
  - **Párovost**
    - Párové (h1-h6, div, nav, main, body)
    - Nepárové (img, br, hr, meta tags)
    - Nepovinně párový (p)
  - **Dle zabíraného místa**
    - Blokové (div, nadpisy, p)
    - Neblokové (strong, a, u)

- **Meta tagy**
  - **Title** - titulek v browseru <title>Stránka</title>.
  - **Author** - autor webu <meta name="author" content="Karel Čermák">
  - **Name** - popisek pro vyhledávače <meta name="description" content="Můj web">
  - **Keywords** - nepoužívá se <meta name="keywords" content="maturita, web">
  - **Theme-color** - nastavení motivu <meta name="theme-color" content="#ffffff">
  - **Charset** – UTF-8 – kódování <meta charset="UTF-8">

- **Nadpisy a texty**
  - **Nadpisy** – H1-H6
  - **Odstavce** – p

- **Seznamy**
  - **OL** – číslovaný seznam
  - **UL** – nečíslovaný seznam
  - **LI** – položka seznamu

  - **Vnořené seznamy**
    - Do LI vložím další OL / UL
    - CSS – ol li ul {..}

2. HTML – layout stránky

- *Popište vytvoření layoutu stránky pomocí bloků.*
- *Jmenujte všechny značky HTML5 pro layout, popište vlastnosti těchto bloků (neplovoucí, plovoucí, chování bloků při vnořování, popište výpočet celkové šířky bloků a způsoby jejich rozmístění vedle sebe či pod sebou).*
- *Popište principy tvorby responzivních webů včetně Media Queries.*

- **Hlavní bloky**
  - div
  - header
  - main
  - footer
  - nav (navigační menu)
  - aside (postranní panel)

- **Grid**
  - Spíše pro velké monitory, funguje ve dvou osách.
  - https://css-tricks.com/snippets/css/complete-guide-grid/
  - display: grid;
  - grid: 100px 300px / 3fr 1fr
    - splitne na dva řádky (100px a 300px) a 2 sloupce (3/4 a 1/4).

- **Flexbox**
  - Spíše pro mobily, funguje dobře v jedné ose.
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
  - flex-direction - tok elementů
  - flex-wrap - přetýkání prvků
  - justify-content - umístění elementů – začátek / konec / střed / obtékání
  - align-items - vertikální zarovnání elementů – nahoře / dole / ve středu
  - align-content - vertikální zarovnání celého bloku

- **Border-box**
  - Může změnit chování width v obsahu.
  - content-box - width bude 100px, border a padding se nepřičítá k obsahu (bude větší než 100px).
  - border-box - border a padding se přičte k obsahu (bude 100px).

- **Media Queries**
  - @media screen and (min/max-width: xxx px) { CSS code... }
  - Screen / print / all - zařízení, není nutno udávat.
  - Způsoby vývoje - **mobile first** / **desktop first**.

- **Plovoucí bloky**
  - float: left / right / none - nastaví to, kde má blok obtékat.
  - clear: left / right / both / none - zakáže obtékání.
  - Musí být nastavena šířka a musí být blokové (display: block).

3. HTML – obrázky, tabulky, odkazy

- *Popište atributy obrázku, způsoby umístění obrázku v textu. Uveďte příklady formátování pomocí CSS. Popište vytvoření plovoucích obrázků umístěných v textu.*
- *Popište strukturu tabulek, typy buněk, slučování buněk. Uveďte příklady formátování pomocí CSS. Uveďte pseudotřídy pro formátování prvního, posledního, n-tého, lichého či sudého řádku nebo sloupce.*
- *Vysvětlete absolutní a relativní URL obrázků a hypertextových odkazů, popište atributy odkazu. Uveďte příklady formátování pomocí CSS. Uveďte kontextuální definice stylů (např. odkazů v menu, odkazů v tabulce apod.)*

- **IMG**
  - Povinné src a alt.
  - CSS - border, height, width...
  - Float IMG - plovoucí obrázek, spíše se nepoužívá (float: right).

- **Relativní cesta**
  - Lepší, pokud se web víc přesouvá .
  - ../slozka/soubor.abc - ze složky zpět.
  - slozka/soubor.abc - ze současné lokace.
  - /slozka/soubor.abc - z rootu (neplést s rootem Linuxu!).

- **Absolutní cesta**
  - Lepší, že je přesně dané.
  - https://cdn.com/file.js - přímá URL.

- **<a>**
  - Href, target (_blank).
  - CSS - text-decoration.
  - download – stáhne soubor.

- **Tabulky**
  - HTML - table -> tbody / thead -> tr (table row) -> td (table data) / th (table head - bude tučné).
  - td / th – colspan / rowspan 2 - sloučí řádky / sloupce (<td colspan="2">..</td>).
  - CSS - border, border-collapse (zruší mezeru mezi buňkami), border-seperate.

- **Pseudotřída**
  - Aplikuje se, pokud je v prvek v daném stavu (například :hover).
  - last-child, first-child, nth-child(even – sudý / odd - lichý).
  - Předdefinovaná třída - nenastavuje se.

- **Oddělovače CSS**
  - **Čárka** (,) - více elementů bude mít stejné styly.
  - **Mezera** ( ) - všechny elementy následující po předchozím - KONTEXTOVÝ SELEKTOR.
  - **Šipka** (>) - element následující pouze po předchozím.
  - **Plus** (+) - první následující sourozenec (h1 + p).
  - **Vlnovka** (~) - ovlivní všechny sourozence (h1 ~ p) - ALT + 126.

4. HTML formuláře

- *Popište formulář, vysvětlete metody odeslání dat.*
- *Vyjmenujte tagy pro prvky formuláře (popisek, textové pole, datumové pole, číselník, rozsah, zaškrtávací pole, přepínače, pole se seznamem, víceřádkové textové pole, tlačítka).*
- *Uveďte příklady použití různých prvků formuláře.*
- *Popište důležité atributy formuláře a jeho prvků.*
- *Jmenujte atributy pro zaškrtnutí zaškrtávacího pole, pro označení jednoho ze skupiny přepínačů a pro výběr určité volby v seznamu.*
- *Uveďte značku pro vytvoření skupin polí.*
- *Popište možnosti formátování prvků pomocí CSS.*
- *Popište pseudoelementy.*

- **Form**
  - method - POST (v požadavku), GET (v URL).
  - action – přesměrování po dokončení (submitu).
  - enctype (multipart/form-data) - pro odesílání souborů v POST.

- **Input**
  - **Type**
    - text (text)
    - radio (jedna z více možností)
    - checbox (více z více)
    - submit (odeslání)
    - date (datum)
    - datetime-local (datum a čas)
    - range (rozsah)
    - number (číslo)
  - **Atributy**
    - required
    - value
    - name
    - placeholder

- **Textarea**
  - Dlouhý text
  - Párový tag

- **Select**
  - Výběr z možností (select - option).

- **Fieldset**
  - Skupina elementů.
  - Legend - název skupiny pro ohraničení.

- **Label**
  - Popisek, udává se ID.

- **Button**
  - Type submit (odeslání), reset (smazání dat).
  - Lze tam navíc použít obrázek.

- **Ovládání stavů**
  - Selected – select
  - Checked – checkbox / radio

- **CSS**
  - input[type="checkbox"] { ... }

- **Pseudoelement**
  - ::before
  - ::after

5. Kaskádové styly CSS

- *Vysvětlete způsob zápisu a důvody použití externích a interních stylů i přímého zápisu stylu u konkrétního tagu.*
- *Vysvětlete definici stylů pro různé selektory (elementy HTML, pseudoelementy, třídy a identifikátory, kontextové selektory), slučování definic, dědičnost kaskádových stylů.*

- **Typy stylů**
  - **Externí styly** - lze použít pro víc souborů, nemusí se ale načíst (`<link rel="stylesheet" href="css.css">`).
  - **Interní styly** - vždy se načte, ale méně přehledné.
  - **Přímý zápis (inline)** - jen pro jeden prvek (`style="color:red;"`), nelze použít Media Query.

- **Oddělovače CSS**
  - **Čárka** (`,`) - více elementů bude mít stejné styly.
  - **Mezera** ( ) - všechny elementy následující po předchozím - KONTEXTOVÝ SELEKTOR.
  - **Šipka** (`>`) - element následující pouze po předchozím.
  - **Plus** (`+`) - první následující sourozenec (`h1 + p`).
  - **Vlnovka** (`~`) - ovlivní všechny sourozence (`h1 ~ p`) - ALT + 126.

- **Kontextové selektory**
  - Odděluje se pouze mezerou.
  - Například: `div p`.

- **Informace k načítání CSS**
  - Přednost má to, co se načte poslední.
  - Dědění - dítě dědí od rodiče.

- **CSS značky**
  - ID - `#id`
  - Class - `.class`
  - Name - `input[name="text"]`
  - Tag – `div`

- **Pseudoelement**
  - `::before`
  - `::after`

6. Javascript – Document Object Model

- *Popište metody umožňující pracovat s jednotlivými elementy HTML dokumentu.*
- *Uveďte metody odkazující se na objekty dokumentu prostřednictvím id, name, třídy stylu či samotného názvu HTML značky.*
- *Popište objekt this.*
- *Vysvětlete způsob změny atributů prvků nebo získání hodnoty atributu, přidání či odebrání třídy stylu a změny CSS vlastností prvků pomocí Javascriptu (např. změna obrázku, změna barvy pozadí, změna textového obsahu, získání hodnoty textového pole apod.)*

- **DOM - strom - node (element, text, komentář):**
  - querySelector, querySelectorAll, getElementsBy(TagName, Name, ClassName), getElementById
  - querySelector - vrátí nodelist, nelze poslat do FOR OF, ale lze použít metodu foreach.
    - .setAttribute
    - .style
    - .innerHTML
  - children - vrátí všechny potomky.
  - firstChild, lastChild - může vrátit i text nebo comment.
  - firstElementChild, lastElementChild - bude vždy element, ne text.
  - nextSibling, previousSibling - další, předchozí sourozenec.
  - parentNode - rodič elementu.

- **Vytváření elementů:**
  - document.createElement(tag)
  - var.innerHTML = ...
  - var.setAttribute, var.getAttribute

- **Umístění elementů:**
  - node.append - konec node
  - node.prepend - začátek node
  - node.before - před node
  - node.after - after node
  - node.replaceWith - nahradí node

  - elem.insertAdjacentHTML(where, html)
    - beforebegin - před blok
    - afterbegin - na začátek
    - beforeend - na konec
    - afterend - po bloku

- **Odstranění Node:**
  - .remove

- **Classes:**
  - .classList
    - .add - přidá
    - .remove - odstraní
    - .toggle – přidá / odstraní
    - .contains - vrátí true / false
  - .className
    - Vrátí / nastaví danou class - nahrazuje již existují.

- **Defer:**
  - Vykoná script až po načtení.
  - <script src="script.js" defer></script>

- **Use strict:**
  - "use strict";
  - Vyvolá error pokud není např.: deklarována proměnná.

- **This**
  - Kontext vykonávané funkce, výše v hierarchii.
  - V objektu chceme jeho vlastnost.
  - Prakticky odvozuje na to, z čeho ho voláme – **TEDY KDE JE NABINDOVANÝ**.
  - Pokud nemá info, použije global var window.
  - Arrow functions nemají this.

- **Event.target**
  - Kontext události.
  - Odvozuje se na prvek, na kterém byla událost vyvolána.

- https://javascript.info/dom-navigation
- https://javascript.info/modifying-document
- https://javascript.info/styles-and-classes

7. Javascript – základy programování – proměnné, pole, podmíněné příkazy, cykly

- *Určete, z jakých znaků může být tvořen název proměnné, vyjmenujte operátory včetně inkrementace a dekrementace, popište syntaxi ternárního operátoru, podmíněných příkazů a cyklů.*
- *Popište proměnnou typu pole a základní funkce pro práci s poli.*

- **Proměnná**
  - Čísla (ne na začátku)
  - Podtržítko
  - Dolar
  - Velké a malá písmena

- **Const**
  - Nemění se.
- **Var**
  - Předzpracuje - kompilátor vytáhne na začátek var a function a udělá globální proměnnou.
- **Let**
  - Dostupné pouze v daném bloku a v něm vnořených.

- **Operátory (unární)**
  - x++
  - x--
  - ++x
  - --x
  - -x
  - +x (rychlá konverzace ze Stringu do INT).

- **Operátory (binární)**
  - = přiřadí hodnotu do proměnné
  - aritmetické operátory - + / * %
  - relační operátory < <= == >= > === !=
  - logické operátory & && | || (pro XOR !=)

- **Operátory (ternární)**
  - cond ? true : false

- **Priority operátorů**
  - Závorky
  - Násobení
  - Sčítání, odčítání
  - Relační (<, >...)

- `NaN` **- Not a Number**
- Dynamický jazyk - datový typ stanovený sám oproti statickému.

- `!!'true' => true`
- `+'1' => 1`

- **Podmínky**

```
if (cond) {
        //if command
} else {
        //else command
}
```

```
switch (prom) {
        case 1:
                //commands
        break;

        case 2:
                //commands
        break;

        default:
                //commands
        break;
}
```

  - `break` není povinný, pokud chceme přetečení.

- **Cykly**
  - while (cond) {...}
  - do {...} while (cond)

  - for - předem nastavujeme deklaraci, podmínku, inkrementaci.
    - for i - klasický for

```
for (let i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
}
```

    - for in - object

```
for (let x in person) {
    text += person[x];
}
```

    - for of - (foreach) - arrays, string, maps, nodelists...

```
for (let x of cars) {
    text += x;
}
```

- **Pole**
  - let arr = [1, 2, 3];
  - Pole v JS je vlastně objekt.
  - Kolekce hodnot s přístupem přes názvy.
  - Index - vždy nezáporné číslo, ne String!

- **Práce s polem**
  - .length - délka pole

  - .unshift(item) - přidání na začátek
  - .push(item) - přidání na konec
  - .shift() - odebrání ze začátku
  - .pop() - odebrání z konce

  - .splice(index, howManyDelete) - odebere od index howManyDelete itemů, lze použít k nahrazování či přidávání, pokud necháme nulu jako howMany, vrátí měněné hodnoty.
  - .slice(from, to) - vrátí dané items od from do to
  - .concat(values..) - přidává hodnoty na konec pole.

  - .indexOf - vrátí index nalezení, jinak -1
  - .includes - pokud obsahuje, vrátí true, jinak false
  - .find(function(item, index, array) {...});) - item je element, index je index, array je samotné pole, lze takhle hledat komplexně v poli.
  - .filter - funguje stejně jako find, ale projíždí a vrací všechny hodnoty.

8. Javascript – uživatelské funkce

- *Popište deklaraci funkce, parametry funkce a návratovou hodnotu funkce.*
- *Uveďte příklad funkce s parametrem a bez parametru.*
- *Vysvětlete rozdíl mezi globální a lokální proměnnou.*
- *Uveďte způsob spuštění funkce příkazem v jiné funkci, spuštění funkce událostí a spuštění funkce v časovém intervalu.*

- **Klíčové slovo function**

```
function name(vstupní hodnoty) {
        //commands
        return ..
}
```

- **Function expression (anonymní funkce)**
  - `const f = function() {}`
  - `this` se vztahuje k objektu.

- **Lambda expression (arrow function)**
  - `const f = (vstupní hodnoty) => {}`
  - `this` se vztahuje na DOM (nepřebírá kontext).

```
[...].filter(functionName);
[...].filter((c?) -> c%2 == 0);
```

- **Globální proměnná**
  - Definována kdekoliv v JS kódu mimo funkci či blok, dostupná odkudkoliv.
- **Lokální proměnná**
  - Definovaná v bloku, nejčastěji ve function, dostupná jen z bloku, při stejném názvu má přednost lokální.

Funkce se volá svým jménem, např.: `getDate();`

- **Spuštění událostí**
  - onclick v HTML
  - addEventListener("click", functionName) - nastavit event ve vstupní proměnné a pak nemusím používat this.
    - odebírání skrz proměnné – removeEventListener
    - click
    - contextmenu - right click
    - mouseove / mouseout – hover
    - mousedown / mouseup - mouse click
    - mousemove
    - keydown / keyup - zmáčknuto tlačítko
    - submit – form
    - focus - i když třeba pomocí tab
  - setTimetout - odložené puštění
  - setInterval - pravidelný interval
    - čas v ms
  - clearTimeout
  - clearInterval
    - udávat ID vytvořené události

- https://javascript.info (2.4 - 2.17)

9. Javascript – objekty Date, Math, String

- *Popište metody objektu Date pro práci s datumy, metody objektu Math pro zaokrouhlování a generování náhodných čísel, metody objektu String pro získání určité části textu, převedení textu na malá nebo velká písmena.*

- **Math**
  - Zaokrouhlování
    - round - podle pravidla
    - floor - vždy dolu
    - ceil - vždy nahoru
  - random - mezi 0 (včetně) a 1 (kromě)

- **String**
  - toUpperCase()
  - toLowerCase()
  - substring() - podřetězec mezi indexy
  - indexOf() - pozice prvnímu výskytu
  - slice() - podřetězec mezi zadanými indexy

- **Date**
  - objekt (new Date(rok, měsíc, den, hodiny, minuty, sekundy))
  - konstruktor bez parametru - současný čas
  - automatické opravování
  - Při kopírování – nutné let datum2 = new Date(datum1), jinak se vytvoří reference!

  - getDate() - vrací den v měsíci (1 až 31).
  - getDay() - vrací 0-6. den v týdnu, 0. je neděle.
  - getMonth() - vrací měsíc (0 až 11).
  - getFullYear() - vrací rok (4 číslice).
  - getHours() - vrací hodiny (0 až 23).
  - getMinutes() - vrací minuty (0 až 59).
  - getSeconds() - vrací sekundy (0 až 59).
  - getTime() - vrací počet milisekund od 1. ledna 1970.

  - setDate() - nastavuje den v měsíci.
  - setMonth() - nastavuje měsíc (0 až 11).
  - setFullYear() - nastavuje rok (4 číslice).
  - setHours() - nastavuje hodiny (0 až 23).
  - setMinutes() - nastavuje minuty (0 až 59).
  - setSeconds() - nastavuje sekundy (0 až 59).

10. Javascript – dynamické formuláře, kontrola dat, regulární výrazy

- *Popište formulářové prvky, uveďte různé metody Javascriptu umožňující získat a měnit hodnoty jejich atributů či CSS vlastností, popište různé události formuláře.*
- *Popište regulární výrazy a uveďte příklady použití.*
- *Uveďte příklady obrany proti odesílání formulářů roboty.*

- **Změny**
  - Prvky - change, input, focus (také vložení)
  - Form - submit, reset

- **Hodnoty**
  - value
  - checked

- **Zastavení provádění v eventu**
  - preventDefault - zastaví jen akci daného elementu
  - stopPropagation - zastaví i všechny akce které jsou nad elementem

- **Regex - funguje i příznak G (hledá všechny shody)**
  - match - závislé na příznaku G
  - matchAll (Nefunguje s příznakem G)

- **Dva typy**
  - **Objetkové**
    - exec (shody)
    - test (true, false)
  - **String**
    - match
    - matchAll
    - replace

11. – 16. Práce s databází v PHP

11. PHP – vyhledávání a filtr dat

- *Vysvětlete dotazy relační databáze s podmínkou pro filtr dat.*
- *Popište PHP funkce pro načtení a výpis dat z tabulky.*
- *Popište obě metody předání hodnoty prostřednictvím formuláře a její využití pro filtr dat.*

12. PHP – výpis a řazení dat

- *Vysvětlete pojem relační databáze a možnosti načtení dat z několika tabulek, které jsou ve vztahu.*
- *Popište PHP funkce pro načtení a výpis dat z tabulek.*
- *Popište metody předání hodnoty prostřednictvím hypertextového odkazu a její využití pro řazení dat.*

13. PHP – vložení dat do databáze

- *Popište vytvoření spojení s databází, nastavení znakové sady, princip kontroly dat a uložení zadaných dat jako nového záznamu do databáze.*
- *Vysvětlete souvislosti mezi metodou odeslání dat formuláře, pojmenováním polí formuláře a PHP proměnnými.*
- *Vysvětlete SQL dotaz pro vložení záznamu.*

14. PHP – načtení dat do formuláře

- *Popište způsob identifikace záznamu, načtení konkrétního záznamu a zobrazení hodnot ve formuláři (hodnoty textového pole, zaškrtnutí zaškrtávacího pole, označení správného přepínače či hodnoty výběrového pole).*

15. PHP – uložení editovaných dat

- *Popište způsob editace záznamu dat z databáze, načtení konkrétního záznamu, provedení změn údajů, kontroly odeslaných dat a provedení změny záznamu v databázi.*

16. PHP – smazání dat

- *Popište způsob výběru záznamu určeného ke smazání, zobrazení vybraného záznamu a výzvy k potvrzení smazání záznamu, provedení skutečného smazání záznamu.*

Jak to vlastně funguje?

- Udělám si instanci třídy.
- Připravím query a spustím prepare.
- Nabinduju parametry (pokud nějaké mám).
- Spustím.
    - Chci jen provést – execute.
    - Chci i získat data - fetchAll.

- **PDO**
    - **Instance třídy** - new PDO();
    - $pdo = new PDO('mysql:host=localhost;dbname=mojedatabaze', 'uzivatel', 'heslo');
    - $dotaz = $pdo->prepare("SELECT * FROM mojetabulka WHERE sloupec1 = :hodnota1 AND sloupec2 LIKE :hodnota2");
    - $dotaz->bindValue(':hodnota1', 'abc');
    - $dotaz->bindValue(':hodnota2', 'abc');
    - $dotaz->execute();
    - $vysledky = $dotaz->fetchAll();

- **bindValue** - přímo přijme hodnotu - umožňuje definovat typ, defaultně String.
- **bindParam** - přijme referenci, takže je možné ještě změnit - umožňuje definovat typ, defaultně String.

- **Do execute lze ještě dosadit hodnoty**

```
$dotaz->execute(array(
        ':filtr' => "%$filtr%"
));
```

- **Fetch** - vrátí pouze jeden řádek
- **setFetchMode**
    - FETCH_BOTH - vrací výsledky jako pole s oběma typy klíčů (číselné i názvy sloupců tabulky) – defaultní.
    - FETCH_NUM - vrací výsledky jako pole s číselnými klíči (klíče jsou pořadová čísla sloupců tabulky) .
    - FETCH_ASSOC - vrací výsledky jako asociativní pole (klíče jsou názvy sloupců tabulky) .
    - FETCHC_OBJ - vrací výsledky jako objekty s vlastnostmi odpovídajícími názvům sloupců tabulky - STD class.
    - FETCH_CLASS - vrátí do třídy - $dotaz->setFetchMode(PDO::FETCH_CLASS, 'MojeTrida');

- **Exec**
  - Pro insert, update, delete.
  - Pouze pokud nevrací výsledky.
  - Vrátí počet ovlivněných řádků.

- **LastInsertId**
  - Poslední vložené ID do databáze.
  - $pdo->lastInsertId()

- **Options**

```
$options = [
        PDO::MYSQL_ATTR_MULTI_STATEMENTS => false,
];
$pdo = new PDO($dsn, $username, $password, $options);
```

  - PDO::MYSQL_ATTR_MULTI_STATEMENTS - zakáže provádění více dotazů najednou.
  - PDO::ATTR_ERRMODE - nastavuje, jak se mají zpracovávat chyby.
  - PDO::ATTR_EMULATE_PREPARES - určuje, zda se mají dotazy emulovat.
  - PDO::ATTR_DEFAULT_FETCH_MODE - určuje výchozí způsob vracení výsledků.

- **Nastavení znakové sady**
  - V connection Stringu "mysql:host=$host;dbname=$db;charset=utf8mb4"
  - Nebo exec $dbh->exec("set names utf8mb4");

- **Odesílání dat a pojmenování proměnných**
  - GET / POST
  - Při GET dostupné v $_GET.
  - Při POST dostupné v $_POST.
  - Nastavuje se v atributu name u inputu.

17. PHP – regulární výrazy, datumové a řetězcové funkce

- *Uveďte příklady PHP funkcí pro práci s datumy a řetězci znaků.*
- *Popište jejich argumenty a výsledky.*
- *Uveďte příklady regulárních výrazů, popište způsob jejich využití.*

- **String funkce**
  - strlen() - délka řetězce.
  - strtolower() - na malé písmena.
  - strtoupper() - na velké písmena.
  - substr() - podřřetězec ze zadaného řetězce.
  - str_replace() - nahradí výskyty zadaného řetězce v daném řetězci.
  - strpos() - vrátí pozici zdaného řetězce nebo false.
  - trim() - odstaní mezerry na začátku a na konci.
  - htmlspecialchars() - převede zvláštní znaky na HTML entity.
  - explode() - rozdělí řetězec na pole podle oddělovače.
  - impode() - sloučí pole do jednoho řetězce s daným spojovačem.

- **Datumové funkce**
  - date("m/d/Y H:i:s"); - aktuální datum v daném formátu.
  - time() - vrátí Unix timestamp (od 1970) .
  - strtotime() - vrátí Unixový timestamp zadaného slovem (např.: next Monday).
  - getdate() - pole s aktuáním časem - [weekday], [hours]...
  - mktime() (vytvoří datum a vrátí unix timestamp) .
  - date_default_timezone_set() - nastaví časovou zónu na danou lokaci.

- **Datumové funkce – objektové**
  - $date = new DateTime();
  - echo $date->format("m/d/Y H:i:s");

  - $date = new DateTime();
  - $date->modify("+1 day");
  - echo $date->format("m/d/Y");

  - $date1 = new DateTime("2022-05-27");
  - $date2 = new DateTime("2022-06-03");
  - $interval = $date1->diff($date2);
  - echo "Rozdíl je " . $interval->format("%a dnů");

- **REGEX**
  - Funkce začínají vždy preg_xxx().
  - preg_match() - řetězec obsahuje jednu schodu - obvykle jestli odpovídá.
  - preg_replace() - nahrazení řetězce jiným řetězcem dle regexu.
  - preg_match_all() - vrátí v poli části řetězce, které odpovídají vzoru.
  - preg_split() - rozdělí řetězec dle regexu a vrátí jako pole (vylepšený explode) .

- **REGEX – psaní**
  - [] - přípustná množina znaků [a-z]; [abc]; [-abc] (zahrnuje pomlčku); [^a-z] (nemůže obsahovat a-z; alt + 94) .
  - () - skupina - např.: ^(demo|example)[0-9] - matchne demo1, example4, ale ne demoexample2.
  - {} - kvantitifátor - ? * + {1,}.
  - \ - escapuje.
  - . - jakýkoliv znak.
  - ? - může být opakováno 0 nebo 1.
  - ^ .. $ - začátek a konec.

  - **Přípustná množina + kvantifikátor.**

18. PHP – vlastní uživatelské funkce, třídy a objekty

- *Vysvětlete deklaraci vlastní třídy, jejích vlastností a metod a jejich použití.*
- *Vysvětlete deklaraci metody i funkce, předávání vstupních hodnot pomocí parametrů při jejím volání a výsledné hodnoty při jejím ukončení.*
- *Vysvětlete rozdíl mezi metodou a funkcí.*

Obsahuje: TŘÍDY, METODY, VLASTNOSTI (proměnné)

- **Třídy**
  - Klíčové slovo class
    - class name {...}
  - Obsahuje metody a vlastnosti.

- **Vlastnost - proměnná ve třídě.**
  - $obj->vlastnost = 5; - zavolá se metoda set, je pak na mě, co budu dělat.
  - Pokud je private a snažím se použít set a nemám set metodu – vyvolá chybu.
  - Pokud je public a snažím se použít set a nemám set metodu - vytvoří veřejnou proměnnou.

- **Metoda - funkce ve třídě**
  - Magická metoda - dvě podtržítka na začátku.
    - Nevolají se přímo.
    - __construct - spustí se při použití new.
    - __set ($propname, $value)
    - __get ($propname) - musí být deklarovaná a nesmí být private.
    - __clone() - duplikuje objekt, nutné nastavit.
  - Deklarace
    - function name($inputVars) {..}

- **METODA** - ve třídě.
- **FUNKCE** - mimo třídu.

- **Vlastnosti funkcí a vlastností**
  - public - dostupné v libovolného místa v kódu - (**METHODS**, **VARS**).
  - private - dostupné pouze v rámci dané třídy - (**METHODS**, **VARS**).
  - protected - dostupné v dané třídě a v odvozených třídách - (**METHODS**, **VARS**).
  - static - je dostupné i bez vytvoření instance třídy pomocí :: - (**METHODS**, **VARS**).

  - abstract - (**CLASSES**, **METHODS**)
    - Slouží ke stanovení metody, která bude dostupná ve všech odvozených třídách.
    - Například budu mít abstraktní třídu Shape, můžu pak odvozovat jednotlivé tvary a vždy musí mít metodu getArea().

```
abstract class Shape {
        abstract public function getArea();
}

class Circle extends Shape {
        private $radius;
        public function __construct($radius) {
                $this->radius = $radius;
        }
        public function getArea() {
                return 3.14 * $this->radius * $this->radius;
        }
}
```

  - final - (**CLASSES**, **METHODS**)
    - Nelze pak od tříd či metod dále dědit.

- **Přístup ke třídám a metodám**
  - $obj = new MyClass();
  - $obj->myMethod();

- **Přístup ke static metodám**
  - Trida::myMethod();

- **Vstupní data**
  - V závorce
  - function name($data, $data2) {...}

- **Výstupní data**
  - Na konci metody
  - return $data

- **POLE**
  - Spíše podobné mapě, spíše se orientuje dle klíčů než k indexu.
  - $myArray = array(1, 2, 3, 4, 5);
  - $myArray[] = 6; přidá na konec pole, zvýší poslední index o 1;

```
$myArray = array(
        'key1' => 'hodnota1',
        'key2' => 'hodnota2',
        'key3' => 'hodnota3'
);
```

- **new stdclass - vytvoří nový objekt, které nemá žádné vlastnosti**

```
$obj = new stdclass();
$obj->name = 'John';
$obj->age = 30;

echo $obj->name;
echo $obj->age;
```

- Při použití &$var předám odkaz na hodnotu. To znamená, že můžu upravovat v metodě například proměnné i mimo metodu.

```
$a = 5;
$b = &$a;
$b = 10;
echo $a; // ($a = 10)
```

- **Dynamické vyjádření názvu proměnné:**

```
$name = 'prom1';
$$name = 'hello'; // ekvivalentní zápis: $prom1 = 'hello';
```

19. PHP – session, cookies

- *Uveďte příklady využití session, způsob jejich vytvoření a načtení a jejich odstranění.*

- **Session - na straně serveru.**
- **Cookies - na straně uživatele.**

- **COOKIE**
  - setcookie - nastavení nové cookie / úprava staré
    - Jméno - povinné
    - Hodnota
    - Expirace - time() + seconds
    - Doména atd.
  - Smazání cookies
    - Nastavení času do minulosti od teď (např.: -1, 1...).
    - Čas 0 smaže ve chvíli zavření prohlížeče.
  - $_COOKIE se aktualizuje až po reloudu!

- **SESSION**
  - session_start() - obnoví proměnné session do $_SESSION / vygeneruje nové session ID a pošle klientovi.
  - unset($_SESSION["name"]) - zrušení hodnoty.
  - session_destroy() - zruší session včetně session id.

20. PHP – soubory

- *Uveďte náležitosti, které musí splňovat HTML formulář, aby mohl posílat soubory.*
- *Uveďte možnosti a funkce PHP pro upload souborů.*
- *Uveďte PHP funkce pro otevření složky a načtení názvů souborů.*
- *Uveďte možnosti rozeznání typů souborů a jejich vypsání buď jako odkazu (např. pro otevření PDF do nové záložky prohlížeče) nebo zobrazení obrázku (soubory jpg, png, gif).*

- **HTML formulář musí obsahovat:**
  - POST
  - enctype="mutlipart/form-data"
  - input type file

- **PHP - Pole $_FILES**
  - $_FILES['formInputName']['valueType']
    - [name] - jméno souboru.
    - [size] - velikost v bajtech.
    - [error] - typ chyby při uploudu.
    - [tmp_name] - dočasný název.
    - [type] - typ souboru (např.: image / gif) - MIME type.

- **Chyby uploadu:**
  - **0:** ok.
  - **1:** moc velké (v upload_max_filesize).
  - **2:** moc velké (v HTML).
  - **3:** pouze částečně nahráno.
  - **4:** nenahráno (chybí soubor).
  - **6:** chybí /tmp složka.
  - **7:** chyba zápisu na disk.
  - **8:** chyba extensionu - extension přerušil upload.

- **Uploud více souborů**

```
Array (
    [name] => Array (
        [0] => foo.txt
        [1] => bar.txt
    ),
    [type] => Array (
        [0] => text/plain
        [1] => text/plain
    ),
    [tmp_name] => Array (
        [0] => /tmp/phpYzdqkD
        [1] => /tmp/phpeEwEWG
    )
    ....
)
```

- **MIME TYPES**
  - Funkce mime_content_type();
  - image/gif, image/png, image/jpeg
  - text/css, text/html, text/javascript
  - video/mp4, video/mpeg
  - applikaction/pdf

- **move_uploaded_file() - přesun souboru do trvalého umístění**
  - $file_tmp = $_FILES['image']['tmp_name'];
  - $file_name = $_FILES['image']['name'];
  - move_uploaded_file($file_tmp, "dir/" . $file_name);

- **Otevření složky**
  - scandir(directory);
  - nevypisovat . a .. directory (buď podmínkou nebo array_diff)
  - glob("dir/*");
  - glob("dir/*.pdf");

21. Návrh databáze, základní databázové pojmy, datové typy

- *Popište princip relačních databází, základní pojmy z oboru databází (relace, entita, atribut, doména atributu) a datové typy.*
- *Návrh databáze – vytvoření databáze a tabulek (včetně syntaxe odpovídajících příkazů).*

- **Funguje na principu matematických množin.**
- **Relace** – tabulka.
- **Entita** – záznam.
- **Atribut** – sloupec.
- **Doména atributu** - hodnota atributu.
- Indexy se ukládají jako B+ strom.

- **Datové typy** - VARCHAR, TEXT, CHAR (doplňuje se mezerama), INT, TINYINT, ENUM, SET, DATETIME, TIMESTAMP
- **Relace** - propojení 1:1 - 1:n n:n

```
CREATE TABLE name (
    column1 datatype,
    column2 datatype,
    column3 datatype
);
```

22. Normalizace dat, klíče, indexy

- *Popište první, druhou a třetí normální formu dat.*
- *Co je to primární a cizí klíč, unikátní a neunikátní indexy.*
- *Popište jejich význam.*

- **NENÍ V ZEALU!!**

- **Primární klíč** - může být jednoduchý či složený, lze u něj nastavit auto increment, musí být jedinečný.
- **Cizí klíč** - odkaz na primární klíč do jiné tabulky, pokud je ve stejné, tak se nazývá jako unární vazba.

- **Normalizace databáze**
    - **1. Všechny hodnoty jsou atomické** - hodnota se v jednom záznamu neopakuje - například telefonní číslo.
    - **2. Každý neklíčový atribut musí být plně závislý na každém kandidátním klíči** - neopakování dat - například názvy (názvy kurzů, adresy zákazníků). Stačí tedy, pokud vše záleží na primárním klíči. Mám zákazníky - u nich jméno, adresu, město, stát.
    - **3. Neobsahuje tranzitivní závislost** - mezi neklíčovými atributy nesmí být závislost. - Nesmí už být ani vztah. Mám zákazníky - u nich jméno, ale adresa, město i stát už je v jiné tabulce

23. Spojování více tabulek, seskupování výsledků

- *Vysvětlete základy spojování tabulek v relačních databázích.*
- *Popište vlastnosti klauzule JOIN.*
- *Vysvětlete seskupování dat a agregační funkce.*

- Jedná se o spojení více tabulek pomocí klíče.

- Natural JOIN - spojuje na základě stejných názvů sloupců a stejných hodnot.
- Union - spojení více výsledků.

- **Druhy joinů:**
  - LEFT JOIN - Vybere vše v A včetně průniku.
  - INNER JOIN - Vybere pouze průnik.
  - RIGHT JOIN - Vybere vše v B včetně průniku.
  - LEFT JOIN WHERE B IS NULL - vybere pouze A bez průniku.
  - RIGHT JOIN WHERE A IS NULL - vybere pouze B bez průniku.
  - FULL OUTER JOIN - vybere vše.
  - FULL OUTER JOIN WHERE A IS NULL OR B IS NULL - vybere vše bez průniku.

- **Počet výsledků před JOINem** - můžeme poznat podle rozdílných hodnot v tabulkách.

- **Klasické funkce** - neupravují funkce dat.
  - CONCAT()
- **Agregační funkce** - upravují počet dat na výstupu.
  - COUNT(), MIN(), MAX(), AVG(), SUM(), GROUP_CONCAT()

- GROUP BY - shromáždění podle dané hodnoty
- HAVING - WHERE pro agregační funkce

24. Výpis dat, vestavěné funkce

- *Popište možnosti příkazu SELECT pro filtrování a řazení výsledků.*
- *Uveďte příklady matematických funkcí, funkcí pro práci s textovými řetězci, formátování datumu.*

- **Zeal 12.4 - 12.7**

```
SELECT data
FROM table
JOIN table2 ON table2.id = table.id
WHERE condition
GROUP BY id
HAVING condition
ORDER BY id
LIMIT 10
```

- WHERE - podmínka pro omezení výběru dat, aplikovaná na jednotlivé řádky - lze použít neagregační funkce, matematické operátory, LIKE, REGEXP (regulární výrazy) apod.
- ORDER BY - řazení dat ve sloupci vzestupně a sestupně.
- Aliasy - přejmenování sloupců - SELECT table.id AS myId.
- LIMIT - omezí výběr dat z výsledku.

25. Základy SQL příkazů, přidávání, úprava a odstranění dat

- *Popište syntaxi příkazů INSERT, UPDATE, DELETE, TRUNCATE a DROP.*

- **INSERT**
  - INSERT INTO (name1, name2) VALUES (value1, value2)
  - INSERT INTO SELECT - zkopírování z jiné tabulky
  - INSERT INTO SET (col) = (val)

- **REPLACE**
  - Stejné jako INSERT, ale nahradí při duplicitě.

- **UPDATE**
  - UPDATE name SET column = value
  - Lze použít WHERE, ORDER BY, LIMIT, JOIN.

- **DELETE**
  - DELETE FROM name
  - Stejné jako update.

- **TRUNCATE**
  - Smaže i auto increment, prakticky DROP a CREATE, nespouští triggery.
  - TRUNCATE name

- **DROP**
  - Smaže tabulku.
  - DROP tableName

- DELETE a UDPATE lze používtvat i u JOINu, ale je nutno určit, s jakou tabulkou se pracuje (před FROM).

```
UPDATE table1
JOIN table2 ON table1.id = table2.id
SET table1.atr = 1,
  table2.atr = 2
WHERE condition
```

Bonus – Jak psát REGEX?

1. Ubal si **pořádného jointa**.
2. Začni **na random mačkat klávesy**.
    - **Funguje to?** Super.
    - **Nefunguje to?** V září si snad vylosuješ lepší otázku.

- **Regex syntaxi najdeš v Zealu zde**: Javascript -> Guide -> Regular Expresions -> Regular Expresion Syntax Cheat Sheet.

- **Regex je vždy v těchto scuffed lomítkách**
    - /regex/

- **Kontrola znaků**
    - \d – jedno číslo od 0 do 9
    - \w – jeden znak – ASCII znak, číslo nebo podtržítko
    - \s – white space character – mezera, tab, new line
    - . – jakýkoliv znak (mimo new line)
    - \i – ignoruje velikost písmen

- **Kvantifikátory**
    - + - jeden nebo více
    - {n} – přesně nkrát ({3} - třikrát)
    - {n, k} – nkrát až kkrát ({2, 4} – dva až čtyřikrát)
    - {n, } – alespoň nkrát ({2,} – alespoň dvakrát)
    - * - nulakrát anebo více
    - ? - jednou anebo vůbec

- **Logika**
    - | - OR operand
    - () – capturing group (A(nt|pple) – Ant nebo Apple)

- **Skupiny znaků**
  - [...] – uvedené znaky ([AEIOU] – A nebo E nebo I nebo O nebo U)
  - [A-Z] – rozsah znaků A-Z
  - [^x] – až na znak x jakákoliv znak (alt + 94)

- **Kotevní znaky**
  - ^ - začátek Stringu (ne v hrané závorce, tam to znaky vylučuje)
  - $ - konec Stringu
  - \g – global search (pouze v JS)

- **Podmíněné hledání**
  - (?=...) – hledej, pokud **splnilo na začátku** – (?=\d{10})\d{5} označí 01234 v 0123456789
  - (?<=...) – hledej, pokud **splnilo na konci** - (?<=\d)cat označí cat v 1cat
  - (?!...) – hledej, pokud nesplnilo na začátku - (?!theatre)the\w+ označí, pokud hledáme v theme
  - (?<!...) – hledej, pokud nesplnilo na začátku - \w{3}(?<!mon)ster označí, pokud hledáme v Munster

- **Příklady (jednodušší)**
  - /hello/ - vyhledá všechny výskyty slova hello v textu
  - /hello|hi/ - vyhledá všechny výskyty slova hello nebo hi v textu
  - /hello|hi/i - vyhledá všechny výskyty slova hello nebo hi v textu, bez ohledu na velikost písmen
  - /h.llo/ - vyhledá všechny výskyty slova, které začínají na "h" a končí na "llo", přičemž kdekoliv mezi h a llo může být libovolný znak
  - /h.*llo/ - vyhledá všechny výskyty slova, které začínají na h a končí na llo, přičemž mezi h a llo může být libovolný počet znaků
  - /h.{3}llo/ - vyhledá všechny výskyty slova, které začínají na h a končí na llo, přičemž mezi h a llo musí být právě 3 znaky

- /^hello/ - vyhledá všechny výskyty slova hello, která se vyskytují na začátku textu
- /hello$/ - vyhledá všechny výskyty slova hello, která se vyskytují na konci textu
- /he{2}lo/ - vyhledá všechny výskyty slova hello, kde e se vyskytuje právě dvakrát
- /he{2,3}lo/ - vyhledá všechny výskyty slova hello, kde e se vyskytuje minimálně dvakrát a maximálně třikrát
- /he+lo/ - vyhledá všechny výskyty slova hello, kde e se vyskytuje alespoň jednou nebo vícekrát
- /he*lo/ - vyhledá všechny výskyty slova hello, kde e se může vyskytnout nulakrát nebo vícekrát
- /h[^aeiou]llo/ - vyhledá všechny výskyty slova hello, kde mezi h a llo může být libovolný znak, kromě samohlásek a, e, i, o, u
- /h[a-f]llo/ - vyhledá všechny výskyty slova hello, kde mezi h a llo může být libovolný znak z rozsahu a až f
- /h[abc]llo/ - vyhledá všechny výskyty slova hello, kde mezi h a llo může být libovolný znak z seznamu a, b, nebo c

- **Příklady (složitější)**
  - /^[a-zA-Z]+\s+\d+$/ - adresa s číslem popisným
  - /^[^@]+@[^@]+\.[^@]+$/ - jednoduchý email
  - /^([0-9]{1,2}|100)$/ - číslo v rozsahu 0 až 100
  - /^([1-9]|[1-2][0-9]|3[0-1])\.([1-9]|1[0-2])\.[2-9][0-9]{3}$/ - datum ve formátu d.m.rrrr
  - /^192\.168\.([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$/ - IP adresa třídy C
  - /^(https:\/\/|http:\/\/)?(www.)?([\w\d-]*\.){1,500}[\w\d]{1,10}$/i - URL včetně protokolu či subdomén

- **Zdroje**
  - https://www.rexegg.com/regex-quickstart.html