# Rozvrh hodin Richard Hývl, 3.D WA 24/25

## Dokumentace pojmů/dojmů

## Typování

PHP je jako příběh „Jak pejsek s kočikou dělali dort“. Zkombinovali typování z JavovýhoScriptu se syntaxí C jazyků a vzniklo z toho „(ne)chutné“ PHP.

#### Typování

PHP je známé jako dynamický jazyk (typy proměnných se určí během běhu program) a slabě typovaný jazyk (dovoluje automatickou konverzi mezi datovými typy bez striktní kontroly), stejně jako JavůvScript. V praxi to znamená, že můžeme sečíst „1“ a 1 a vyjde nám 2 (pokud nejste plochozemci). Stringy se spojují tečkou - $string1 . $string2.

Zápis proměnné může vypadat takto:   
String - $variable = ”Ahoj světe!“  
Integer - $variable =69  
Float - $variable = 59.99  
Boolean - $variable = true  
Array - $variable = [”jablko”, ”banán”, “třešeň”]

### Důležité funkce pro práci v poli

Aby to bylo přehlednější, rozdělil jsem to do základních skupin. A taky jsem si vytvořil imaginární pole a JSON, aby byla jednodušší demonstrace. Pole $array = [1, 2, 3], $array2 = [4, 5, 6], $array3 = [‘a’, ‘b’, ‘c’]’, array4(3, 1, 2) (takhle se i pole deklaruje) a JSON $data = [‘id’ => 1, ‘name’ => ‘Libor Klempíř’], [‘id’ => 2, ‘name’ => ‘Pavel Bajer’]

#### Úprava pole

array\_push($array, 4, 5) – přidá hodnoty na konec vybraného pole ($array = [1, 2, 3, 4, 5])  
array\_unshift($array, 0) – přidá hodnoty na začátek vybraného pole ($array = [0, 1, 2, 3])  
$last = array\_pop($array) – odebere posledí prvek z pole a uloží ho ($array = [1, 2], $last = 3)  
$first = array\_shift($array) – odebere první prvek z pole a uloží ho ($array = [2, 3], $first = 1)  
$merged = array\_merge($array, $array2) – spojí obě pole a uloží je ($merged = [1, 2, 3, 4, 5, 6])  
$squared = array\_map(function($n) {  
 return $n \* $n;  
}, $array) – aplikuje funkci na každý prvek ($squared = [1, 4, 9])  
sort($array4) – seřadí pole vzestupně ($array = [1, 2, 3])   
$combined = array\_combine($array3, $array) – vytvoří pole z 2 jiných polí, přičemž jedno obsahuje klíče a druhé hodnoty ($combined = ['a' => 1, 'b' => 2, 'c' => 3])

#### Čtení z pole

$exists = in\_array(2, $array) – vrací boolean, pokud je prvek v poli ($exists = true)  
$index = array\_search(2, $array) – vrací index prvku v poli podle klíče ($index = 1)  
$sliced = array\_slice($array, 1, 2) – vrací počet prvků od indexu ($sliced = [2, 3])  
$filtered = array\_filter($array, function($n){  
 return $n % 2 = 1;  
}) – filtruje každý prvek na základě podmínky ($filtered = [1, 3])  
$names = array\_column($data, ‘name’) – vrátí hodnoty daného sloupce pole ($names = ‘Libor Klempíř’, ’Pavel Bajer’)  
$reversed = array\_reverse($array) – vrátí opačné pořadí prvků ($reversed = [3, 2, 1])  
$count = count($array) – vrátí počet prvků = .length ($count = 3)

## Objektový model

#### OOP

PHP podporuje Objektově Orientované Programování stejně jako Java. To znamená, že obsahuje Třídy, Objekty a jejich vlastnosti, Metody, Dědičnost, Zapouzdření, Polymorfismus. Jediné, co je potřeba zmínit, je rozdílná syntaxe PHP a Javy.

Odkazování na Vlastnost - $this->variable  
Odkazování na Metody - $object->method(){}  
Odkazování na Statické Metody a Vlastnosti – Class::staticMethod();  
Magické metody – unikátní v PHP, začínají \_\_, automaticky se zavolá v určitých situacích  
Gettery (obecný) – public function \_\_get($name){return $this->name;}  
Settery (obecný) – public function \_\_set($name, $value){$this->name = $value;}  
Konstruktor – public function \_\_construct()

## Dokumentace řešení/kódu

### Řešení

Cílem bylo udělat rozvrh hodin s tím, že hodiny jsou dosazovány dynamicky (nikoliv staticky). Moje řešení bylo udělat objekty hodin, které obsahují den, kolikátá hodina to je v ten den, název předmětu, učitele, třídu, skupinu. Ty jsem pak vložil do objektu rozvrh, což je defakto pole hodin s metodami. Potom jsem v indexu přes foreach vypsal všechny hodiny do tabulky (podobně jako na bakalářích).

### Kód

#### Hour.class.php

* Objekt
* Vlastnosti: day, rank, group, teacher, name, classroom
* Konstruktor: Při vytvoření objektu je potřeba zadat všechny hodnoty
* Metody: obecný getter, obecný setter (v řešení k ničemu)
* Require: neobsahuje

#### Schedule.class.php

* Objekt
* Vlastnosti: schedule (pole Hour)
* Konstruktor: neobsahuje
* Metody:
  + addHour(Hour $hour) - přidá hodinu, return void
  + getClass(int $day, int $rank) - při zadávání hledá, pokud je v ten den a tu hodinu nějaký (může jich být více, pokud je to rozdělené na skupiny), return Hour[]/null
* Require: Hour.class.php

#### index.php

* Main
* Metody: neobsahuje
* Require: podle potřeby

V první části kódu se vytváří objekty Hour, které se pak následně uloží do Schedule

V druhé části kódu se vytváří celá html struktura (aby se dalo lehčeji stylovat). Hlavní částí je zde foreach v tabulce, který vypíše do buňky předmět, pokud mu ve vybraný den ve vybranou hodinu pasuje. Pokud ne, nechá buňku prázdnou.  
  
//Poznámka pro mě – není v A5