1. Programovací jazyky

Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy. Pojem algoritmus se nejčastěji objevuje při programování, kdy se jím myslí teoretický princip řešení problému (oproti přesnému zápisu v konkrétním programovacím jazyce). Obecně se ale algoritmus může objevit v jakémkoli jiném vědeckém odvětví. Jako jistý druh algoritmu lze chápat jakýkoliv popis pracovního postupu nebo i např. kuchařský recept.

Programovací jazyk je prostředek pro zápis algoritmů, jež mohou být provedeny na počítači. Zápis algoritmu ve zvoleném programovacím jazyce se nazývá program. Programovací jazyk je komunikačním nástrojem mezi programátorem, který v programovacím jazyce formuluje postup řešení daného problému, a počítačem, který program interpretuje technickými prostředky. Programovací jazyk je vlastně soubor pravidel pro zápis algoritmu.

Existuje několik možností kritérií, podle kterých lze programovací jazyky dělit:

- podle míry abstrakce na **nižší** a **vyšší**

  Nižší programovací jazyk (strojový kód, assembler) poskytuje malou nebo žádnou abstrakci od toho, jak funguje procesor počítače. Označení „nižší“ odkazuje na velmi malý nebo žádný rozdíl mezi daným programovacím jazykem a strojovými instrukcemi procesoru. Nižší programovací jazyk může být snadno převeden do strojového kódu procesoru. Jsou obtížně osvojitelné pro programátora, protože je pro jejich efektivní používání nutné seznámit se s technickými detaily fungování hardware. Programy napsané v nižším programovacím jazyce mohou být rychlé s malými nároky na paměť.
  Vyšší programovací jazyk je jazyk s větší mírou abstrakce. Vyšší abstrakcí je míněno přiblížení zápisu zdrojového kódu programu k tomu, jak myslí člověk. Ve vyšších programovacích jazycích je možné používat prvky přirozeného jazyka. Struktura zdrojového kódu je logická. Další výhodou vyšších programovacích jazyků je jejich přenositelnost.

- podle **způsobu překladu a spuštění**

  Kompilační jazyky se nejprve přeloží a až poté spustí. Jsou rychlejší, ale náročnější na přesnost zápisu.
  Interpretační se překládají za běhu programu.

Vyšší jazyky se dále dělí**:**

- Procedurální
  - Strukturované
  - Objektově orientované
- Neprocedurální
  - Funkcionální
  - Logické

Strukturované programování je technika, kdy se algoritmus rozděluje na dílčí úlohy (t.j. procedury či funkce), které se spojují v jeden celek. Při strukturovaném programování se používá vybraných řídicích struktur (podmínka, větvení, cyklus), ostatní struktury nejsou povoleny, u strukturovaného programování se např. nepoužívá řídicí příkaz skoku. Strukturovaný program je zápis instrukcí (příkazů) tak, jak jdou za sebou.

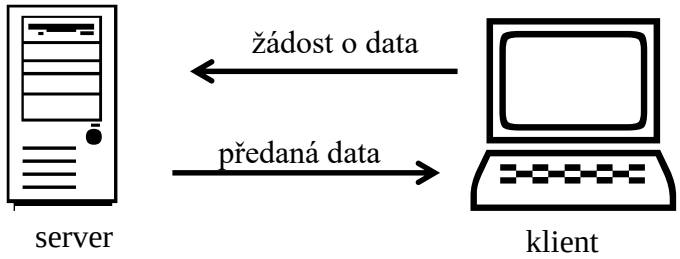
Objektově orientované programování (OOP). Při OOP se **odpoutáváme od toho, jak program vidí počítač (stroj) a píšeme program spíše z pohledu programátora (člověka).**
Základním prvkem je objekt, který má vlastnosti a metody, které se vtahují k určitým událostem. Např. Objekt člověk má vlastnosti věk, pohlaví, výšku, národnost,… a metody – dovednosti. Metoda se provede, pokud dojde k dané události. Objekt je instancí určité třídy. Např. třída člověk, objekt Pepa nebo Jarda. O programech v OOP říkáme, že jsou událostmi řízené

Životní cyklus programu

- zadání
- analýza
- návrh (algoritmizace)
- kódování (přepis do programovacího jazyka)
- testování
- dokumentace
- údržba

2. Programování webových stránek

Služba **WWW** je realizována na principu **klient / server.** To znamená, že při prohlížení nějaké webové stránky na internetu se zapojují do hry dvě strany. **Server** je zdrojem dat a z něj si danou stránku stáhneme. **Klientem** je prohlížeč, ve kterém uživatel stránky prohlíží.



Pokud vytvoříme webové stránky pouze pomocí jazyků HTML (nebo XHTML) a CSS, pak získáme **statické stránky**. To znamená stránky, které jsou neměnné a při opakovaném zobrazování jsou stále stejné.

Stránky, které mohou nějakým způsobem měnit svůj obsah (například reagovat na vstup od uživatele nebo pracovat třeba s datem a časem apod.) jsou **stránky dynamické**. Pro tvorbu dynamických stránek potřebujeme navíc ještě **skriptovací jazyk**. To je programovací jazyk, který se podílí na vytvoření stránky nebo mění její obsah.

Podle toho, kdo se stará o interpretaci skriptu rozlišujeme **skriptovací jazyky** na **klientské a serverové.**
Skripty psané klientskými jazyky se provádějí přímo v prohlížeči uživatele. Takové stránky pak můžeme prohlížet i na svém počítači před jejich publikováním. Klientské skripty umějí spíše vizuální efekty.
O provádění serverových skriptů se stará server. Serverové skripty umožňují programovat aplikace spolupracující s uživatelem, databázemi, poštou apod.

3. Formuláře v HTML

Formuláře se na stránkách uplatní tam, kde potřebujeme získat nějaké informace od uživatele. Jazyk HTML umí formuláře vytvářet, ale neumí je zpracovat.
Nejprve se budeme zabývat vytvoření formuláře pomocí HTML značek a až v dalších kapitolách se naučíme získaná data zpracovat pomocí PHP.

Formulář vkládáme do stránky pomocí značek **<form>…</form>**.
Na jedné stránce může být víc formulářů, ale nesmí se křížit ani vnořovat.
Značka **<form>** má dva povinné parametry – **action** a **method**.

Hodnotou parametru **action** je adresa stránky, kam mají být data po odeslání formuláře předána ke zpracování. Často data zpracovává stejná stránka, pak zadáme jen název dané stránky. Pokud parametr **action** chybí, pak si s tím většina prohlížečů poradí tak, že odešle data do stejné stránky, ale chybějící parametr odporuje specifikaci a stránka není validní.

Parametr **method** určuje, jakým způsobem budou data z formuláře přenesena.
Hodnotou **get** se data přenáší pomocí adresy – názvy polí a hodnoty, které uživatel vyplní, se stanou součástí webové adresy. Pokud jsou data důležitá pro zobrazení konkrétní stránky s konkrétním obsahem, pak je vhodné

použít právě metodu **get.** Cílovou stránku můžeme uložit k oblíbeným a tak ji vyvolat znovu i s daným obsahem.
Metodou **post** se data přenáší v dokumentu a nejsou tedy vidět v adrese cílové stránky. Metodu **post** použijeme, když se jedná o citlivá nebo příliš velká data, jako např. přihlašovací jméno a heslo nebo text mailu.

4. Formulářové prvky

Textové pole

Textové pole slouží k zadávání krátkých textů. Textové pole vložíme značkou **<input type="text">**.
Značkou **<input…>** lze vkládat i jiné formulářové prvky, proto je v ní povinný parametr **type**, kterým určíme, o jaký prvek se jedná.
Dalším povinným parametrem je u většiny formulářových prvků parametr **name**, který určuje jméno tohoto prvku a je tedy jedinečný (tj. každý prvek se v daném formuláři musí jmenovat jinak). Názvu prvku bude po odeslání formuláře přiřazena hodnota zadaná uživatelem. Např. pokud textovému poli s názvem *jmeno* zadáme hodnotu *Alois,* vznikne dvojice *jmeno=Alois*.

Skryté pole

Skryté pole umožní odeslat s formulářem data, o kterých uživatel nemusí vědět (např. IP adresa, operační systém, rozlišení, prohlížeč atd.). Používá se také při posílání dříve zadaných hodnot.

Vkládáme **<input type="hidden">**. Povinné parametry jsou **name** a **value**.

Odesílací tlačítko

Vkládáme **<input type="submit">**. Vzhled tlačítka nelze ovlivnit (záleží na prohlížeči). Pokud chceme ovlivnit text na tlačítku použijeme tento text jako hodnotu parametru **value**. Nemusí mít **name** (pokud to situace nevyžaduje). Pokud má tlačítko parametr **name**, pak vznikne dvojice „*jméno tlačítka*" = *„text na tlačítku"*.

Pro začátky programování v PHP nám to bude zatím stačit. Více v poznámkách pro VT – webová prezentace

5. Úvod do PHP

PHP je serverový skriptovací jazyk navržený pro potřeby webových stránek.
Vše, co PHP provádí, se interpretuje na straně serveru a generuje HTML (či jiný) výstup, který vidí uživatel.

Zdrojový kód PHP skriptu se vkládá přímo do zdrojového kódu HTML např.
<?php
…
?>

Syntaxe je velmi podobná syntaxi programovacího jazyka C. Každý příkaz musí být oddělen středníkem.

Příkaz **echo** slouží k vypisování textu. V příkazu **echo** můžeme zapisovat i příkazy HTML.

Komentář v PHP skriptech se píše buď jednořádkový
Př. //*text komentáře*
nebo víceřádkový
Př. /**text komentáře*
i ve více řádcích/

U příkazů PHP nerozlišuje mezi malými a velkými písmeny ovšem u názvů proměnných ano, proto je lepší vyvarovat se experimentů a používat pouze malá písmena. Zároveň u názvů proměnných i prvků ve formulářích raději zapomeneme na háčky a čárky.

Pokud je ve stránce použit kód PHP, musí mít soubor příponu **.php**

K zobrazení stránky včetně PHP skriptu musíme mít nainstalovaný překladač jazyka PHP a webserver nebo musíme publikovat na Internetu.

6. Získání dat z formuláře

Značka **<form>** má povinné parametry **action** (určuje stránku, kam jsou data odeslána) a **method** (určuje způsob přenesení dat - metody **get** a **post**). Každý formulářový prvek má parametrem **name** zadáno jedinečné jméno, které se v cílovém skriptu stane proměnnou s hodnotou zadanou ve formuláři.

Všechny proměnné předané prostřednictvím formuláře jsou uloženy v poli:

a) **$ _POST**, pokud byla použita metoda **post**
b) **$ _GET**, pokud byla použita metoda **get**

Př. Textové pole ve formuláři má name="jmeno" a method="post".
$jmeno=$ _POST[jmeno];

Pro návrat k formuláři a zadání jiných hodnot můžeme použít např. formulář pouze s odesílacím tlačítkem nasměrovaný na původní stránku s formulářem.
Př. <form action="…" method="post">
<input type="submit" value="zadat nové hodnoty">
</form>

Př. formulář

```
<form action="formular.php" method="get">
  <p>
   Jméno:    <input type="text" name="jmeno"><br>
   Věk:    <input type="text" name="vek"><br>
    <input type="submit" value="Pošli!" name="odeslano">
   </p>
  </form>
```

zpracování formuláře na stejné stránce

```
 <?php
  if (isset($_GET["odeslano"]))
  echo "<p>Jmenujete se ".$_GET["jmeno"]."a je Vám ".$_GET["vek"]."let.</p>";
?>
```

Funkce **isset()**, jejímž parametrem je jakákoliv hodnota z pole $_GET (nebo $_POST), vrací hodnotu **true**, jestliže daná hodnota (tedy vlastně prvek ve formuláři) existuje. V opačném případě vrací **false**. Při zpracování dat z formuláře na té samé stránce používáme vždy podmínku s touto funkcí a celý skript do podmínky uzavřeme (je-li více příkazů, tak zavíráme do {}). Dosáhneme tím toho, že skript se provede pouze po odeslání formuláře. Pokud bychom podmínku nepoužili, dočkáme se nejspíš chybových hlášek a také výstupu bez zadaných hodnot. Vyhnout se použití této funkce můžeme jedině vytvořením vlastní stránky pro zpracování formuláře.
Podrobnějšího vysvětlení podmínky se dočkáme později.

7. Proměnné

Proměnné v PHP poznáme podle znaku **$**, kterým začíná název každé proměnné. Znak $ napíšeme např. kombinací kláves pravý alt+ů. V programovacích jazycích se obvykle nepoužívají jednopísmenné proměnné, ale raději názvy, které napovídají o obsahu proměnné (ne $x, $y, $z, ale raději $cislo1, $cislo2, $soucet, …).
Proměnné jsou různých datových typů, podle toho jakou hodnotu obsahují.
Nejčastěji používané typy:

Datový typ	**Hodnota proměnné**
integer	celé číslo v rozmezí od -2^{31} do 2^{31} (2^{31}= 2 147 483 647)
float	reálné číslo
array	Pole
string	textový řetězec

boolean	logický typ (hodnoty TRUE, FALSE – pravda, nepravda)

V PHP na rozdíl od jiných programovacích jazyků není třeba proměnné deklarovat. Proměnná se zavádí při prvním použití zadáním hodnoty, tím také určíme jakého je typu.
Př.

```
<?php
   $cislo=12;                proměnná $cislo je typu integer
   $text="ahoj";             proměnná $text je typu string
?>
```

Do proměnné zadáváme pomocí znaku = buď přímo konkrétní hodnotu nebo výsledek nějaké operace s jinými proměnnými nebo hodnotami.

Operace	Způsob zápisu
sčítání	$soucet=12+4; nebo $soucet=$cislo1+$cislo2;
odčítání	$rozdil=12-4; nebo $rozdil=$cislo1-$cislo2;
násobení	$soucin=12*4; nebo $soucin=$cislo1*$cislo2;
dělení	$podil=12/4; nebo $podil=$cislo1/$cislo2;
zbytek po dělení	$zbytek=12%4; nebo $zbytek=$cislo1%$cislo2;

Násobení a dělení má přednost před sčítáním a odčítáním. Pokud chceme změnit prioritu operací, použijeme kulaté závorky.

Mezi operace s proměnnými patří i spojování řetězců pomocí tečky, které jsme už používali v příkazu **echo.**
Př. $text="Dvacet děleno deseti je ".(20/10);

```
   $text="začátek a ";
   $text=$text."konec";
```

Některé z výše uvedených operací lze zapisovat zkráceným (optimalizovaným) způsobem.

Klasický zápis	Optimalizovaný zápis
$text=$text."abc"	$text.="abc"
$cislo=$cislo+5;	$cislo+=5;
$cislo=$cislo-5;	$cislo-=5;
$cislo=$cislo*5;	$cislo*=5;
$cislo=$cislo/5;	$cislo/=5;
$cislo=$cislo+1;	$cislo++;
$cislo=$cislo-1;	$cislo--;

Jednotlivé proměnné můžeme porovnávat jak mezi sebou, tak s konkrétní hodnotou:

Operace	Způsob zápisu
test rovnosti	$a= =$b; $a= =12;
test nerovnosti	$a!=$b; $a!=12;
je větší	$a>$b; $a>12;
je menší	$a<$b; $a<12;
je větší nebo rovno	$a>=$b; $a>=12;
je menší nebo rovno	$a<=$b; $<=12;

Výsledkem porovnávání je hodnota TRUE nebo FALSE. Podrobněji si porovnávání vysvětlíme později, v souvislosti s podmínkami.

8. Switch – větvení

Příkaz větvení – **switch** můžeme chápat jako přepínač, který vykoná určitou část skriptu v závislosti na hodnotě dané proměnné.
Př.

```
<?php
  $i=…;
  switch ($i)
  {
   case hodnota1 : příkaz1; break;
```

```
    case hodnota2:příkaz2; break;
    ...
    case hodnotaN:příkazN; break;
    default: příkaz;
    }
?>
```

Příkazem **switch ($i)** začne skript testovat hodnotu proměnné **$i**. Za příkazem **case** jsou vypisovány přípustné hodnoty proměnné **i** a přízaz(y), které se vykonají pokud **$i** je rovno této hodnotě. Příkaz **break** ukončí testování dalších hodnot. Pokud není podmínka splněna v žádném z vyjmenovaných případů, provede se příkaz uvedený za **default**. Pokud **default** není uvedeno, nestane se nic.

9. If – podmínka

Jiným typem větvení programu je podmínka – **if.**
Podmínka může být:

- neúplná: **if** (*podmínka*) *příkaz*;
  Pokud je splněna podmínka, provede se příkaz. Není-li splněna, nestane se nic.
- úplná: **if** (*podmínka*) *příkaz1*; **else** *příkaz2*;
  Pokud je splněna podmínka, provede se příkaz1. Není-li splněna, provede se příkaz2.
- vnořená **if** (*podmínka1*) *příkaz1*; **elseif** (*podmínka2*) *příkaz2*;
  Pokud je splněna podmínka1, provede se příkaz1. Není-li splněna, testuje se podmínka2 atd.
  Za podmínkou2 může být **else** (nemusí) nebo další **elseif**.

Př. Do proměnné *$diskriminant* spočítáme diskriminant kvadratické rovnice.

```
if ($diskriminant<0) echo "Rovnice nemá v R řešení.";
elseif ($diskriminant==0) echo "Rovnice má v R 1 řešení.";
else echo "Rovnice má v R 2 řešení.";
```

10. Cykly

Pokud potřebujeme nějaký příkaz(y) opakovat vícekrát, použijeme cyklus.
V PHP máme k dispozici tři typy cyklů:

- **while**
- **do … while**
- **for**

Cyklus „while"

Cyklus **while** je cyklus s podmínkou na začátku a provádí se, dokud je splněna podmínka.
Zapisujeme: **while** (*podmínka*) *příkaz;*
Na začátku se zkontroluje podmínka - je-li splněna, provede se příkaz a pak se znovu testuje podmínka. Cyklus končí, pokud podmínka splněna není.
Chceme-li provádět více příkazů, pak použijeme složené závorky.
Př.

```
<?php
    $pocet=1;
    while ($pocet<=6) {
        echo "<h$pocet>Toto je nadpis úrovně $pocet </h$pocet>";
        $pocet++;
    }
    ?>
```

V příkazech cyklu musí být takový, aby podmínka měla šanci se změnit, jinak bude cyklus nekonečný.
Př. Tento cyklus bude donekonečna vypisovat text

```
<?php
$pocet=1;
while ($pocet>0) echo "Dobrý den!";
?>
```

Cyklus „do … while“

Tento cyklus se liší od předchozího tím, že nejdříve je příkaz(y) a pak teprve podmínka.

Zapisujeme: **do** *příkaz* **while** (*podmínka*);

Rozdíl mezi oběma cykly je v tom, že cyklus s podmínkou na začátku nemusí proběhnout vůbec, zatímco cyklus s podmínkou na konci proběhne vždy alespoň jedenkrát.

Př. <?php
```
$pocet=1;
do {
    echo “<h$pocet>Toto je nadpis úrovně $pocet </h$pocet>“;
    $pocet++;
}
while ($pocet<=6)
?>
```

Cyklus “for“

Zapisujeme: **for** (*výraz1; výraz2; výraz3*) {*příkazy*};

- *výraz1* Nastavuje počáteční stav počítadla, např. $pocet=1;
- *výraz2* určuje podmínku, která má být pro výkon cyklu splněna, např. $pocet<=6;
- *výraz3* aktualizuje počítadlo, obvykle zvyšuje hodnotu o 1, např. $pocet++;

Př. <?php
```
for ($pocet=1; $pocet<=6; $pocet++) {
    echo “<h$pocet>Toto je nadpis úrovně $pocet </h$pocet>“;
}
?>
```

Cyklus **for** má ještě speciální variantu – cyklus **foreach**. Tento typ cyklu je určený pro procházení kolekcí, např. polí. V PHP se jím zabývat nebudeme. Užití tohoto cyklu si necháme do vyšších ročníků, kde se s ním setkáme při programování v jazyce C#.

11. Datum a čas

V PHP je k dispozici rozsáhlá knihovna funkcí. Většina funkcí vyžaduje další parametry, které se předávají v kulatých závorkách za názvem funkce. Pokud je funkce bez parametrů, uvádíme prázdné závorky ().
Pro práci s časem používáme funkce **time()** a **date()**.

Funkce **time()** vrací počet sekund, které uplynuly od půlnoci 1.1.1970. Tato funkce je bez parametrů.
Výsledek funkce zobrazíme pomocí příkazu **echo**.
Př. <?php
```
    echo time();
?>
```
Do příkazu **echo** můžeme vkládat značky jazyka HTML, ale nemůžeme v nich používat uvozovky (považovaly by se za ukončení textového řetězce). Místo uvozovek používáme apostrofy (zapsat možno např. pravý alt+39).
Textové řetězce v příkazu **echo** spojujeme pomocí tečky.
Př. <?php
```
    echo “<p class=“text“>Od půlnoci 1. ledna 1970 uplynulo právě“.time().“ sekund“;
?>
```

Funkce **date()** vrací aktuální datum a čas. Pokud tuto funkci zavoláme bez parametrů, dojde k vypsání chybového hlášení.

Funkce **date()** očekává parametry, kterými sdělíme, co se má v oblasti data a času zobrazit

Parametr	**Návratová hodnota**
a	„am“ nebo „pm “ – dopoledne nebo odpoledne
A	Jako **a**, ale výsledek je zapsán velkými písmeny
g	aktuální hodina bez počáteční nuly – 0 až 12
G	aktuální hodina bez počáteční nuly – 0 až 23
h	aktuální hodina s počáteční nulou – 0 až 12
H	aktuální hodina s počáteční nulou – 0 až 23
i	aktuální minuta – 00 až 59
d	aktuální číslo dne v měsíci – 01 až 31
j	aktuální číslo dne v měsíci bez počáteční nuly – 1 až 31
l	název dne v týdnu - anglicky
D	název dne v týdnu (anglicky) vyjádřený třemi počátečními písmeny
w	den v týdnu numericky – 0 až 6 (0 je neděle)
F	název měsíce – anglicky
M	název měsíce (anglicky) vyjádřený třemi počátečními písmeny
m	číslo měsíce - 01 až 12
n	číslo měsíce bez počáteční nuly - 1 až 12
s	aktuální počet sekund – 00 až 59
t	počet dní v aktuálním měsíci – 28 až 31
Y	aktuální rok prezentovaný čtyřmi číslicemi
y	aktuální rok prezentovaný dvěma číslicemi
z	pořadí dne v roce – 0 až 365
L	přestupný rok – vrací 1 je-li přestupný, jinak vrací 0
I	letní čas - vrací 1 je-li platný letní čas, jinak vrací 0
U	počet sekund od 00:00:00 dne 1.1.1970 - stejné jako time()

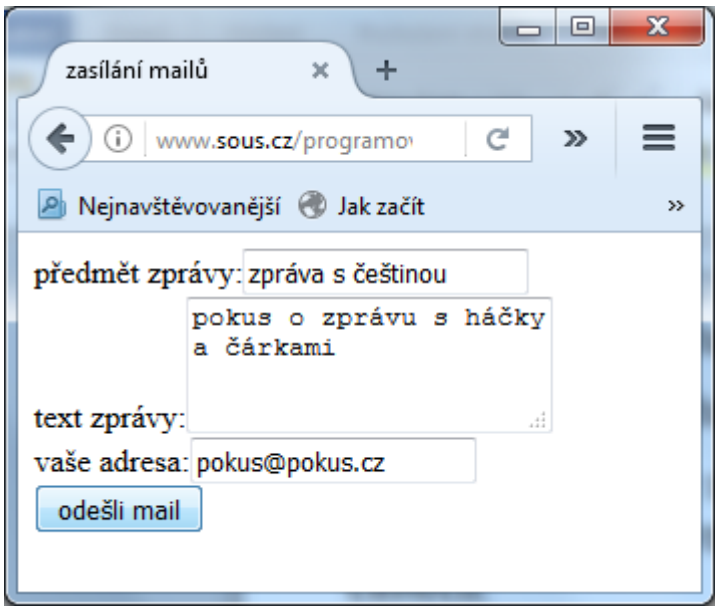
Aktuální datum ve tvaru *dd.mm.rrrr* tedy vypíšeme:
echo date(“d. m. Y“);
Uvozovky v závorkách jsou proto, aby se nám vypsaly i tečky a mezery.

12. Funce mail()

Funkce **mail()** umožňuje zpracovat formulář pro zaslání zprávy na zadanou mailovou adresu, aniž bychom adresu buď přímo zveřejnili, nebo uvedli v HTML kódu např. napište nám.

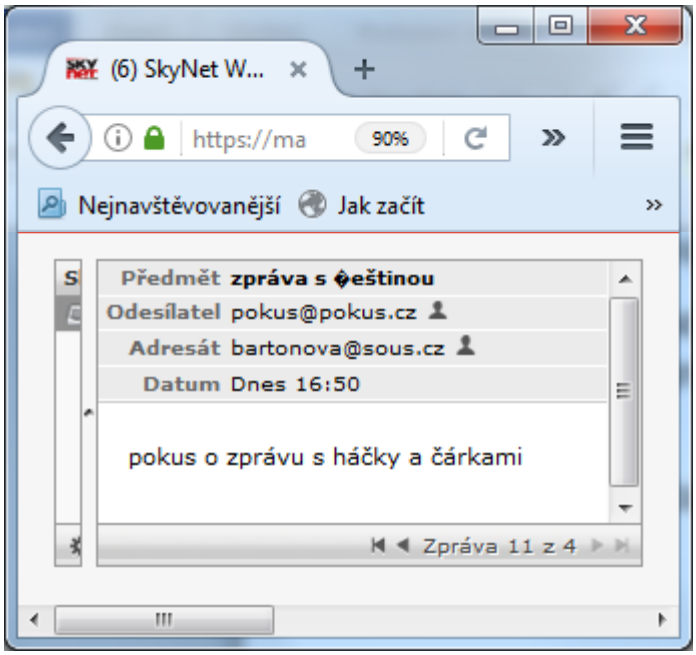
Funkce **mail()** má tři povinné parametry uvedené v daném pořadí – adresa, předmět a text zprávy. Je dobré vytvořit si na ně proměnné, se kterými pak budeme funkci volat. Jako čtvrtý nepovinný parametr může být proměnná, která nese hlavičky. Hlaviček je celá řada. Mezi sebou se oddělují znakem pro konec řádku „\n“ Je vhodné uvést alespoň adresu odesílatele a také řešit kódování češtiny ve zprávě.

Právě kódování je jediná věc, kterou se mi zatím nepodařilo uspokojivě vyřešit. V následujícím příkladu je použito stejné kódování jako v HTML dokumentu. Funguje to ovšem jen na text zprávy, ale ne na předmět a odesílatele. Pokud u odesílatele předpokládáme uvedení funkční mailové adresy, tak tam na češtinu nenarazíme, ale u předmětu ano. Pokud někdo ví, jak tento problém řešit, nechám se ráda poučit.



V příkladu je vytvořen formulář jako na levém obrázku. Na pravém obrázku je zobrazena došlá zpráva. Po stisku tlačítka se mail odešle na mou adresu. Adresu, i když je uvnitř skriptu, uvádím raději po částech.

Zdrojový kód těla stránky *email.php*:




```
<body>
  <form action="email.php" method="post">
  předmět zprávy:<input type="text"
name="predmet"> <br>
  text zprávy:<textarea name="zprava" rows="3"
cols="20"></textarea>
  <br>
  vaše adresa:<input type="text" name="odesilatel"
value="@"> <br>
  <input type="submit" name="tl" value="odešli
mail">
 </form>

<?php
  if (isset($_POST["tl"]))
  {
     $adresa="bartonova";
     $adresa.="@";
     $adresa.="sous.cz";
     $predmet=$_POST["predmet"];
     $zprava=$_POST["zprava"];
     $hlavicky="content-type:text/html; charset=utf-8\n";
     $hlavicky.="from:<".$_POST['odesilatel'].">\n";
     $vysledek=mail($adresa,$predmet,$zprava,$hlavicky);
     if ($vysledek)
     echo "zpráva odeslána";
     else echo "chyba";
  }
 ?>
</body>
```

13. Úvod do JavaScriptu

JavaScript je klientský skriptovací jazyk. Zapisuje se přímo do HTML kódu. Se zdrojovým kódem stránky se odešle do prohlížeče uživatele a tam se provede.

JavaScript je jazyk **interpretační** (vykonává se příkaz po příkazu, nekompiluje se), **objektový** (využívá objektů prohlížeče a zabudovaných objektů), velmi rozšířený a snadno dostupný (je součástí prohlížeče).

Naposledy jmenovaná vlastnost, tedy dostupnost v prohlížeči, může být i nevýhodou. Vzhledem k různým verzím jazyka a prohlížečů může snadno docházet k chybám. Uživatel navíc může JS skripty zakázat. Dalším omezením je fakt, že JavaScript nemůže pracovat se soubory a ukládat data (kromě cookies).

Začlenění skriptu do stránky

Zapisujeme třemi způsoby:

- **přímo do dokumentu** pomocí značky <script></script>
  Př. <script> *vyskočí informační okno s pozdravem*
  alert("Dobrý den!");
  </script>
- zápisem do **externího dokumentu** s příponou **.js**, který ke stránce připojíme pomocí <script src="*cesta k dokumentu*"></script>. Obvykle umísťujeme na konec těla stránky.
- **in-line zápisem**, kdy se nevyužívá značka <script>, ale celý skript je atributem jiné značky.
  Př. Spoje
  při kliknutí na odkaz se objeví informační okno

Všechny tři druhy zápisu jsou použity v archivu v příkladech *zapis1.html* a *zapis2.html* + *pozdrav.js*

V zápisech skriptů můžeme používat **komentáře**. Zápis je stejný jako u php skriptů.

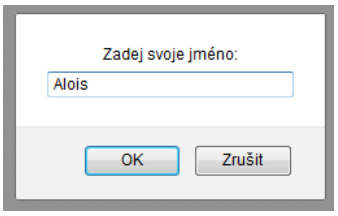
```
// jednořádkový komentář        /*víceřádkový
                                  komentář*/
```

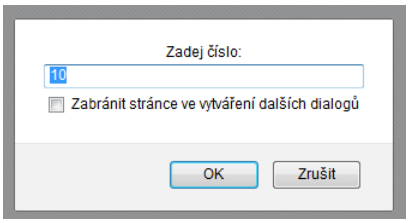
Vstup a výstup

prompt

Prompt() vyvolá dialogové okno, které vyžaduje vstup od uživatele.

Př.
```
<sript>
    var cislo, jmeno;
    jmeno=prompt("Zadej svoje jméno:", "");
    cislo=prompt("Zadej číslo:", "10");
    ... další zpracování zadaných dat ...
</script>
```



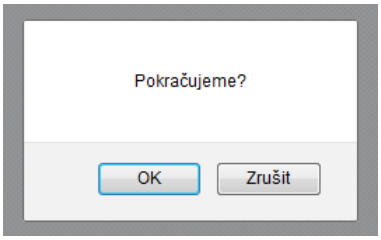


Hodnota zadaná v okně se přiřadí proměnné. V závorkách jsou dva vstupní parametry – text v okně a výchozí hodnota. Může být i bez výchozí hodnoty.

confirm

Confirm() vyvolá potvrzovací dialog. Parametrem je text v okně. Návratová hodnota je true/false.

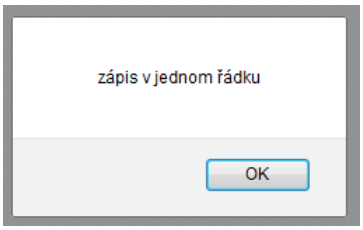
Př.
```
<sript>
    var pokracovat, zprava;
    pokracovat=confirm("Pokračujeme?");
    if (pokracovat) zprava="Pokračujeme!";
    else zprava="Končíme";
    alert(zprava);
</script>
```

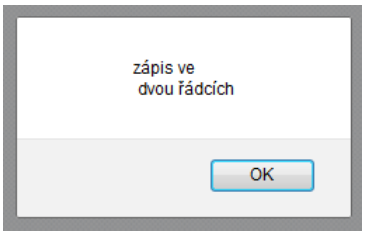


alert

Alert() vyvolá informační okno. Parametrem je text v okně. Pro odřádkování použijeme \n.

Př.
```
<sript>
    alert("zápis v jednom řádku");
    alert("zápis ve \n dvou řádcích");
</script>
```

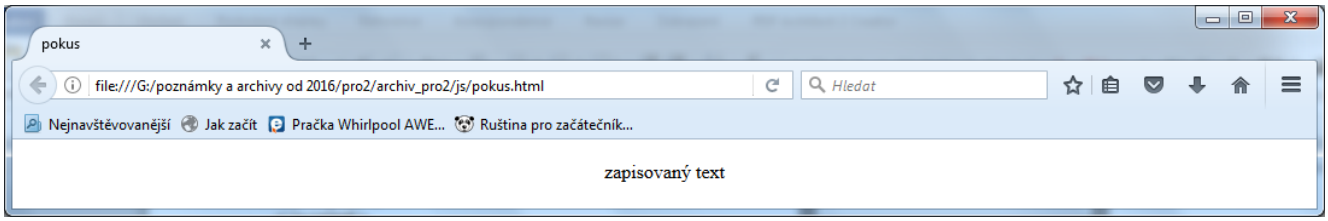




Zápis přímo do stránky

Pro výpis přímo do prohlížené stránky použijeme document.write(); Parametrem je text zápisu včetně HTML značek. Je to jako echo v PHP. Stejně jako v PHP musíme uvnitř textového řetězce v "*uvozovkách*" používat *'jednoduché uvozovky* '.

Př.
```
<sript>
    document.write("<p align='center'>zapisovaný text</p>");
</script>
```



Všechny výše uvedené funkce jsou použity v příkladu *vstup_vystup.html*.

14. Proměnné a datové typy

Proměnná se deklaruje klíčovým slovem **var**, za kterým následuje buď jen jméno proměnné, nebo jméno s přiřazenou hodnotou. Deklarovat lze více proměnných najednou. Deklarace oddělujeme čárkou.
Př. var cislo=5, text="abc", pokus;
Proměnná *cislo* je typu *number*, *text* je typu *string*, *pokus* je *undefined*.
Datový typ proměnné je určen hodnotou, kterou do ní přiřadíme. I v průběhu existence proměnné můžeme změnit její typ pouze tím, že jí přiřadíme hodnotu jiného typu.
JavaScript rozlišuje 6 datových typů:

Number - číslo

Jediný číselný typ v JS. Patří do něj všechny číselné hodnoty a speciální hodnota **NaN** (Not-a-Number). Tato hodnota se vrací jako výsledek při selhání matematické funkce nebo při načítání hodnoty, která má být číslo ale není (třeba na pokyn zadej číslo napíšeme „abc“).
Hodnota **NaN** je zajímavá tím, že **se nerovná sama sobě** – podmínky (NaN==NaN) a (NaN===NaN) vrací *false*. Někdy potřebujeme ověřit, zda vrácená hodnota je nebo není NaN. Na to je funkce*isNaN()*, ale má jisté záludnosti při použití. Jedinou spolehlivou cestou, jak zjistit jestli je proměnná hodnoty NaN, je porovnat ji samu se sebou. Žádná jiná hodnota nemá tu vlastnost, že by se nerovnala sama sobě.

String – textový řetězec

Všechno uzavřené v uvozovkách. Jsou-li uvozovky součástí řetězce, je nutné je nahradit jednoduchými uvozovkami.

Boolean – logický typ

Má pouze hodnoty *true* a *false*.

Undefined

Pouze jedna hodnota – undefined. Odpovídá stavu proměnné po deklaraci bez přiřazení hodnoty.

Null

Pouze jedna hodnota – null. Prázdná hodnota.

Object - objekt

Objekt je kolekce vlastností. Vlastnosti mohou proměnné být jiných datových typů, jiné objekty nebo funkce.

Undefined

Pouze jedna hodnota – undefined. Odpovídá stavu proměnné po deklaraci bez přiřazení hodnoty.

Operace

Matematické operace

Sčítání (+), odčítání (-), násobení (*), dělení (/) a zbytek po dělení (%). Stejné jako známe z PHP. Stejně tak funguje složené přiřazení (+=, -=, *=, /=, ++, --).
Pozor na operátor +. U řetězců + spojuje řetězce. Vzhledem ke slabé typovosti jazyka může být problém s převody mezi typy Number a String nebo Number a Boolean. Při použití + tak může dojít k tomu, že se čísla interpretují jako řetězce a místo sčítání se spojuje (10+5=105). Při sčítání čísla a logické hodnoty se zase *true* bere jako 1 a *false* jako 0.
Pokud potřebujeme sečíst dvě číselné proměnné, které jsme načetli od uživatele, tak je musíme nejprve vynásobit 1. Tím z nich uděláme čísla.

```
Př. var x=prompt("Zadej číslo x:", "");          zadáme  5
    var y=prompt("Zadej číslo y:", "");          zadáme  7
    var soucet1=x+y;                             soucet1 bude  57
    var soucet2=1*x+1*y;                         soucet2 bude  12
```

Porovnávání

Rovnost (==) – vrací hodnotu *true*, když se hodnoty rovnají, jinak *false*. Jako shodné vyhodnotí i hodnoty různých typů, takže pravda je např. "1"==1, 1==true, ""==0 nebo " "==0.

Striktní rovnost (===) – je jako == a navíc se musí shodovat i typy, takže např. "1"==1 nebo 1==true pravda není.
Nerovnost (!=) – k rovnosti.
Striktní nerovnost (!==) – opak striktní rovnosti.
Větší než (>)
Větší nebo rovno (>=)
Menší než (<)
Menší nebo rovno (<=)

15. Řízení toku - podmínka

Význam podmínky i syntaxe jsou téměř stejné jako v PHP.
Užití podmínky v příkladech *vstup_vystup.html* a *datum.html*.

16. Cykly

Význam cyklů i syntaxe jsou téměř stejné jako v PHP. I zde máme cyklus s daným počteme opakování (for), cyklus s podmínkou na začátku (while) a cyklus s podmínkou na konci (do-while).
Užití for cyklu v příkladu *nasobky.html*.

17. Události

Využití událostí pomocí in-line zápisu si ukážeme pouze na příkladu *udalosti.html*.