JavaScript – základy jazyka

Základní věci jsou stejné jako v C, C++, Javě, C#

Velikost písmen se rozlišuje!
Na konci příkazu je středník. Další příkaz můžete psát hned za středníkem nebo na dalším řádku.
Rovnítko = přiřadí hodnotu do proměnné: a = b + c
Dvojité rovnítko == testuje rovnost, operátor != testuje nerovnost (hodí se pro příkaz if)
Násobení *, dělení /, zbytek po dělení %, AND &&, OR ||
Jednořádková poznámka začíná //, víceřádková je /* mezi tímto */
Funkce se volají pomocí kulatých závorek, tj. název(parametr);

Kam se program píše

JavaScript se musí psát do HTML souboru, čili do webové stránky. Není to ale složité.

```
<html>
<body>
<script>
  document.write("Hello World!");
</script>
</body>
</html>
```

Proměnné

Proměnná je pojmenovaná paměťová buňka. Je globální, nebo uvnitř funkce, nebo uvnitř objektu.
U proměnných se neuvádějí datové typy, všechna čísla jsou typu double (8bajtové FPU).
Novou proměnnou založíte pomocí var: **var a = 20**
Pozor! Napíšete-li jen **a = 20**, bude to většinou stačit, ale občas se to chová nečekaně! (Důvodem je, že JavaScript je dynamický funkcionální jazyk. Proto VŽDY zakládejte proměnné pomocí var.)

Text a psaní

Pro stringy můžete používat uvozovky **"** i apostrofy **'** (stejně jako v HTML).
document.write("html nebo text") pošle text do výstupu. Může to být i HTML zdroják. Ve skutečnosti to HTML zdroják je vždycky, takže například odřádkování uděláte pomocí **
, znak < napíšete pomocí **<, znak > pomocí **>**.
Víc řetězců můžete spojit pomocí **+**.

Opakování kódu

JavaScript má všechny běžné opakovací příkazy a ještě několik navíc. Dva základní **while** a **for** uveďme na příkladu vypsání čísel 0 až 9.

```
var a = 0;
while(a <= 9) {
  document.write(a);
  a++; //zvýší hodnotu proměnné o jedničku
}

for(var b = 0; b <= 9; b++) {
  document.write(b);
}
```

Podmíněné vykonání

Podmíněný příkaz **if** funguje podobně jako while, větev **else** je nepovinná.

```
if(a > 0) document.write("a je kladné");
if(b == 0) document.write("b je nula"); else document.write("b není nula");
```

Funkce

Na rozdíl od většiny jazyků, JavaScript umožňuje tzv. globální kód. Příkazy tedy můžeme psát přímo do HTML <script> tagu. Můžeme však také používat funkce. Funkce je pojmenovaná část programu. Funkce může mít parametry a vracet hodnotu. Jelikož JavaScript je dynamický jazyk, definice funkce má velmi jednoduchou syntaxi:

```
function jméno() {
  //sem píšeme příkazy
  //každý příkaz končí středníkem nebo je na novém řádku (a pak středník nepotřebuje)
}
```

Funkce obvykle má nějaké vstupní parametry. A příkazem **return** může vracet výsledek. (Výsledek vrátit může a nemusí, na rozdíl od jiných jazyků nikam nemusíme psát void apod.)

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

Když takto nadefinujeme funkci, tato část programu se nevykoná, dokud ji odněkud nezavoláme.

```
document.write(SoucetTri(2,3,5));
```

Cvičení – jednoduché algoritmy

1. Vypište vedle sebe čísla 1 až 10. Za každým číslem přidejte mezeru pro lepší čitelnost.

2. Vypište pod sebou sudá čísla 2 až 20.

3. Vypište trojúhelník z hvězd. Každý další řádek má o jednu hvězdu víc než předchozí.
   *
   **
   ***
   ****
   *****
4. Vypište zmenšující se trojúhelník.
   *****
   ****
   ***
   **
   *

5. Program pro trojúhelníky v předchozích příkladech upravte na funkci s parametrizovanou velikostí trojúhelníku (tj. abychom mohli vypisovat i jiné velikosti než 5).

Pole

Pole je řada více paměťových buněk, ke kterým můžeme přistupovat dle jejich indexu (pořadí v poli).
Nové pole založíme takto: **var p = new Array(velikost);**
Buňky pole mají indexy od nuly až do velikost–1.
Přístup k jednotlivým buňkám pole je pomocí hranatých závorek: p[0], p[1], atd.
Pole lze předat funkci celé jako jeden parametr. Předává se však odkazem, takže funkce mění přímo původní pole, ne svou lokální kopii!!
Kopie pole se vytvoř pomocí slice: **var kopie = p.slice(0);**

Strukturovaná data

JavaScript je dynamický jazyk a umožňuje velmi jednoduše skládat data do struktur.

osoba = { jmeno: "Jan", prijmeni: "Svoboda", rocnik: 1990 }

Vytvořili jsme proměnnou osoba jakožto strukturu skládající se ze tří položek. K vnitřním položkám přistupujeme stejně jako v jiných jazycích pomocí tečky, tedy osoba.jmeno, osoba.prijmeni, osoba.rocnik.

Objekty

Chceme-li v datových strukturách mít kromě samotných dat také nějakou funkcionalitu, použijeme objekty. Objekt je de facto datová struktura doplněná o nějaké funkce/metody, které s těmito daty pracují. JavaScript nepodporuje třídy, takže vytvoříme přímo jeden objekt jako vzor/šablonu/prototyp ten pak rozmnožujeme klonováním. (Toto beztřídové vytváření objektů se nazývá **prototypování**.)

Syntaxi objektů ukazuje následující příklad. Vytvoříme objekt Fronta. Ten obsahuje dvě proměnné prvni a posledni, to jsou reference na začátek a konec fronty. Inicializujeme je slovem **null**, což je označení pro referenci, která nikam neukazuje. Dále objekt obsahuje tři metody pridej, odeber a jeprazdna. (Pozn. Jak vidno na tomto příkladě, JavaScript je funkcionální jazyk, proto i objekty jsou zde vlastně funkce. Metody objektů jsou pak de facto proměnné odkazující na funkce.)

function Fronta() {

```
 this.prvni = null;
 this.posledni = null;

 this.pridej = function(hodnota) {
  //funkce pro přidání prvku do fronty
 }

 this.odeber = function() {
  //funkce pro odebrání prvku z fronty
}

 this.jeprazdna = function() {
  return this.prvni===null
 }
}
```

Takto vytvořená Fronta je prototypem pro vytváření dalších objektů pomocí **new**.

```
var f = new Fronta()
f.pridej(10)
f.pridej(20)
f.pridej("nazdar")

var g = new Fronta()
g.pridej(1)
g.pridej(2)
```

Cvičení – pole a objekty

1. Napište funkci, která vypíše obsah pole předaného jí jako parametr. Vytvořte pole čísel 1 až 10 a nechte ho vypsat pomocí vámi vytvořené funkce.

2. Dopište zde uvedenou kostru fronty na plně funkční frontu.

3. Analogicky dle fronty napište třídu pro zásobník.

4. Napište funkci Sort pro třídění pole. (JavaScript nerozlišuje datové typy, takže bude fungovat pro čísla i stringy).