JavaScript 2

Matěj Cajthaml

Distanční výuka - 45. - 46. hod
Smíchovská SPŠ | WBA

Abstrakt

Tento dokument vznikl jako podpůrný materiál při **n-té** vlně pandemie CoVID-19 a spojených nemocí. Je určen pouze pro studijní účely. Text byl vytvořen ve školním roce 2020 / 2021 na škole Smíchovská SPŠ. Autorem textu je Matěj Cajthaml jakékoliv připomínky a dotazy buď odesílejte na e-mail matej.cajthaml@ssps.cz nebo na MS Teams.

Obsah

1 Úvod

Stále pokračujeme v tématu JavaScriptu. Tentokrát se vrhneme již na pokročilejší věci jako pole, cykly, funkce a další věci. Znovu připomínám že je důležité si věci zkoušet sami.

2 Řidící konstrukce

V minulých týdnech jsme se naučili používat řídící konstrukci if - podmínku, větvení. Nyní si do řídích podmínek přidáme taktéž switch - neboli výběr z možností. Z jazyku C# víte že switch je taková jednodušší možnost porovnávacích (spíše rovných se) podmínek. Do zápisu switch se vkládá hodnota - proměnná a ta se porovnává se všemi možnostmi (case). Až se bude hodnota v nějakém bloku rovnat, zavolají se příkazy z daného bloku. Jestli na konci bloku je break, celý switch se ukončí. Pokud break není přítomný, pokračuje se další možnosti.

Konstrukce switch může mít taktéž možnost “default”, která určuje co se stane, když žádná možnost nebude sedět.

Stránku si můžete zobrazit ZDE.

3 Porovnávání & logické operátory

V JavaScriptu, díky velmi omezeným možnostem nastavení datových typů, existuje několik variant porovnání. Jednomu se říká striktní (===), které kontroluje jak hodnotu tak datový typ (např. tedy “491” === 491 není pravda) a druhému obyčejné (==), které kontroluje pouze hodnotu (např. tedy “491” == 491 je pravda). Samozřejmě existují negace (!== a !=), porovnávání, zda je větší/menší je stejné (>, <, >=, <=). Výše zmíněný switch používá striktní porovnávání.

Existují samozřejmě logické operátory, díky kterým můžete různé podmínky vrstvit. Existuje “and” (a), který se zapisuje pomocí &&, “or” (nebo), který se zapisuje pomocí “||” a samozřejmě kulaté závorky, které usměrňují pořadí operací. Negaci, která se zapisuje pomocí “!”, jsme si již ukázali.

Podmínky se taktéž dají zmenšit na jednořádkovou konstrukci, která lze například přiřadit do proměnné. Skládá se z podmínky, znaku “?”, po kterém následuje hodnota pro kladnou reprezentaci, znak “:”, po kterém následuje hodnota pro negativní reprezentaci.

Stránku si můžete zobrazit ZDE.

```
<script>
    let fruit = "banana";

    if(fruit === "banana") {
        console.log("Ovoce je banán.")
    }

    let pseudonumber = "4200";

    if(pseudonumber == 4200) {
        console.log("String se rovná číslu 🙄.")
    }

    if(pseudonumber === 4200) {
        console.log("A tohle již nikdy neuvidíte.")
    }

    if(pseudonumber == "4200" && !(fruit === "apple")) {
        console.log("Pseudočíslo je 4200 a ovoce není jablko.")
    }

    let bananaOrApple = (fruit === "banana" || fruit === "apple");

    console.log(bananaOrApple);

    let cooked = fruit === "apple" ? "koláč" : "upečené ovoce";
    // je to stejné jako
    if(fruit === "apple") {
        cooked = "koláč";
    } else {
        cooked = "upečené ovoce";
    }

    console.log(cooked);
</script>
```

4 Pole

Pole již z jiných programovacích jazyků znáte: jedná se o seskupení dat do jedné položky, kterou můžeme pomocí indexů procházet. Oproti jiným jazykům však pole v JS má následující vlastnosti:

- indexuje se od 0
- má neomezenou velikosti (teoreticky lze přidávat do nekonečna)
- data - items mohou být jakéhokoliv datového typu (číslo, text, funkce, další pole, ...)

Pole v JS zakládáme pomocí hranatých složených závorek. Při zakládání můžeme přímo přidat prvky. K předmětům v poli přistupujeme pomocí hranatých závorek za danou proměnou.

Prvky do pole přidáváme pomocí metody .push(), kterou voláme na poli. Vlastností .length získáme velikost pole (počet prvků). Pomocí metody .includes() zjistíme, zda daný prvek existuje v poli. Pomocí metody .indexOf zistíme index určitého prvku (není-li v poli, metoda vrátí -1). Pomocí metody .splice() můžeme odstranit část pole: první parametr určuje začátek pole a druhý parametr konec pole.

Na poli existují další metody (.findIndex, .find, .map, .unshift, ...) které prozatím nepotřebujeme a zbytečně by nám zaplnili hlavu.

Stránku si můžete zobrazit ZDE.

5 Cykly

V JS existují stejné cykly, které již znáte z C#: for & while (existuje i do while, ten ale budeme ignorovat). Prvně se vrhneme na for.

Základní cyklus for umí projíždět od určitého číslo pomocí podmínky. Cyklus for umí projíždět například pole (i objekty, ale o tom později). Cyklus typu for... in místo prvků v poli ale vypisuje index daného prvku - pro získání musíme zavolat na poli daný index. To nemusíme dělat v cyklu typu for... of, který již přímo čte prvky z pole. Pomocí cyklu for můžeme projíždět i další objekty - například text.

Stránku si můžete zobrazit ZDE.

Cyklus typu while se používá tam, kde není potřeba interace. While provádí vnitřní příkazy dokud je podmínka pravdivá.

Stránku si můžete zobrazit ZDE.

```
<script>
    let i = 0;

    let stop = false;

    while (stop === false) {
        i += 2;

        if(i > 100) {
            stop = true;
        }
    }

    console.log("Konec s i =", i);

</script>
```

6 Objekty

Nyní už začíná pravá zábava JavaScriptu: objekty, věc, která se až přespříliš používá. Objekty v JS jsou jakési data. Samotný objekt může obsahovat nespočet věcí. Každá věc - další data - jsou pojmenované nějakým jménem. V objektu můžeme mít další objekty.

K přístupu proměnných v objektu používáme tečku. Do objektu můžeme přidávat další data. Z objektu můžeme mazat data tím, že na jejich jméno nastavíme hodnotu "undefined". Jednotlivé jména dat (a poté jejich data) v objektu můžeme projíždět cyklem typu for... in. Objekty můžeme mít v polích.

Stránku si můžete zobrazit ZDE.

7 Funkce

Funkce jsou jakési "černé krabičky", do kterých přivádíme vstup a oni vrací výstup. V JS můžeme funkce definovat do proměnných a potom je podle jména volat. Funkce se vždy značí klíčovým slovem function a následně parametry. Klíčové slovo return říká, co se při zavolání funkce má odeslat zpět.

Stránku si můžete zobrazit ZDE.

```
<script>
    let add = function(a, b) {
        return a + b;
    }

    console.log(add(12, 1.2));

    let fancyPrint = function(data) {
        console.log("~" + (data) + "~");
    }

    fancyPrint(add(12, add(13, 1)));
</script>
```

8 Úkol

Vaším úkolem je implementovat čtyři funkce, které provádějí různé akce nad polem objektů film. Objekt filmu je vždy tvořen z této struktury:

```
{
    name: string,
    category: string,
    actors: array (pole stringů obsahující jméno herců)
    rating: float (číslo od 0.0 do 10.0),
    releaseYear: int
}
```

Za každou správnou implementaci funkci můžete získat až dva body, bodový základ je pět bodů. Každá funkce má jeden parametr, což je zmíněné pole objektů. **Pro vytvoření úkolu použijte předpřipravený soubor ZDE.**. Neměňte jména proměnných ani funkcí.

8.1 1. funkce

Vypsání filmů které byly vydány po roce 2000 a mají hodnocení přesně nebo více jak 5.7. Filmy vkládejte do jednoho stringu a každý film ukončete novým řádkem (\n), string poté vraťte jako výstupní hodnotu funkce. Filmy musí být vypsány ve formátu: "{ name, releaseYear, rating / 10 }", například tedy:

```
{ Shrek, 2001, 10 / 10 }
{ Shrek 2, 2002, 9.3 / 10 }
{ Harry Potter and the Philosopher's Stone, 2001, 7.6 / 10 }
```

8.2 2. funkce

Vypočítejte průměrné hodnocení filmů, nejstarší a nejnovější film podle roku. Jako výstupní hodnotu vraťte string, ve kterém budou čísla rozdělena následovně: "průměr, nejstarší, nejnovější". Pokud není vypočítání možné, uveďte za údaj číslo 0. Výstup by mohl vypadat následovně:

```
4.1, 1992, 2021
```

8.3 3. funkce

Nalezněte herce, kteří hrají ve více filmech. Do výstupní hodnoty funkce - jako pole - vložte všechny herce, které jsou alespoň ve dvou filmech. Pole (NENÍ STRING) by mohlo vypadat následovně:

```
[
    "Daniel Radcliffe",
    "Rupert Grint",
    "Emma Watson"
]
```

8.4 4. funkce

Spočítejte počty filmů dle různých kategorií. Jako výstupní hodnotu funkce vložte objekt, ve kterém budou klíče jména kategorií a hodnoty počty filmů. Výstup jako objekt (NENÍ STRING) by mohl vypadat následovně:

```
{
    Adventure: 2,
    Fantasy: 1
}
```