JavaScript :: Lectia 2

4.1 Variabile (var vs let) (continuare)

b. Tipuri de variabile: (continuare)

3. Siruri de elemente asociative (associative arrays)

```
var volunteer = {
   firstname: 'John',
   surname: 'Doe',
   dateOfBirth: '2000-01-01',
   destination: 'Ghana',
   project: 'Care'
};

console.log(volunteer.firstname); // afiseaza 'John' in consola
```

4. Obiecte

```
var person = {
   firstname: 'John',
   surname: 'Doe',
   age: 23,
   getFullname: function() {
         return this.firstname + ' ' + this.surname;
   }
}

console.log(person.firstname); // afiseaza 'John' in consola
console.log(person.getFullname()); // afiseaza 'John Doe' in consola
```

Best practices
- intotdeauna declaram variabilele! `var volunteerName;`

- intotdeauna folosim variabile locale! Variabilele locale se declara folosind cuvantul cheie `var`

```
var test = "I'm global";

function testScope() {
      var test = "I'm local";
      console.log (test);
}

testScope(); // afiseaza: I'm local

console.log(test); // afiseaza: I'm global
```

```
var test = "I'm global";

function testScope() {
      test = "I'm local";
      console.log(test);
}

console.log(test); // afiseaza: I'm global

testScope(); // afiseaza: I'm local

console.log(test); // afiseaza: I'm local (variabila globala este
modificata in interiorul functiei)
```

- variabilele se declara la inceput:
    - astfel codul este mai curat
    - toate variabilele se gasesc intr-un singur loc evitandu-se astfel folosirea de variabile globale si evitarea redeclararii de variabile
    - JS muta toate declararile de variabile la inceputul scriptului (vezi **JS Hoisting**)

- variabilele se initializeaza in momentul declararii:
    - codul este mai curat
    - toate initializarile se gasesc intr-un singur loc
    - se evita variabilele nedefinite

- initializand o variabila oferim o idee legata de tipul variabilei

- niciodata nu declaram obiecte de tip numeric, string sau boolean

- tot timpul numerele, string-urile si valorile boolean se trateaza ca valori primitive, nu ca obiecte

- nu folosim new Object(), ci folosim:

```
var x1 = {};           // new object
var x2 = "";           // new primitive string
var x3 = 0;            // new primitive number
var x4 = false;        // new primitive boolean
var x5 = [];           // new array object
var x6 = /()/;         // new regexp object
var x7 = function(){}; // new function object
```

- foarte mare atentie la conversiile automate:

- atentie ca numerele pot fii accidental convertite la string sau NaN (Not a Number)

- javascript nu tine cont de tip, iar o variabila isi poate schimba tipul pe perioada existentei ei

```
var x = "John"; // variabila de tip string
x = 3; // variabila a devenit de tip numeric
```

- cand facem operatii matematice, javascript poate converti numerele la stringuri

```
var x = 5 + 7;    // valoarea lui x este 12, iar tipul este numeric
var x = 5 + "7";  // valoarea lui x este 57, iar tipul este string
var x = "5" + 7;  // valoarea lui x este 57, iar tipul este string
var x = 5 - 7;    // valoarea lui x este -2, iar tipul este numeric
var x = 5 - "7";  // valoarea lui x este -2, iar tipul este numeric
var x = "5" - 7;  // valoarea lui x este -2, iar tipul este numeric
var x = 5 - "a";  // valoarea lui x este NaN, iar tipul este
numeric
```

4.2 Constante

```
const FILE_PATH = '/filestore/';
const ARGENTINA_ID = 13;
const PI = 3.14;
```

```
const MY_OBJECT = {'key': 'value'};
```

4.3 Comentarii

Pentru a introduce in script anumite informatii legate de acel cod folosim comentariile. Ele sunt ignorate de browser in momentul executarii scriptului JS. Comentariile sunt de doua tipuri:

```
// comentariul pe o singura linie

/* comentariu
pe mai multe
linii */

/*
 * comentariu
 * pe mai multe
 * linii
 */
```

4.4 Operatori

- Aritmetici:
    - + adunare
    - – scadere
    - / impartire
    - * inmultire
    - % rest
    - ** ridicare la putere
    - ++ incrementare
    - -- decrementare

- Relationali:
    - < mai mic
    - > mai mare
    - <= mai mic sau egal
    - >= mai mare sau egal

- De egalitate
    - == egal
    - != diferit (not equal)
    - === identic egal
    - !== identic diferit

- Logici:
    - && si
    - || ori

- Ternari:
    - `condition ? ifTrue : ifFalse`

- De atribuire:
    - =

```
var a = 5;
var b = 3;
var c = d = 7; (echivalent cu var c = 7; var d = 7;)
```

    - +=
    - -=
    - *=
    - /=
    - %=

- Comma ",": permite evaluarea mai multor expresii si returneaza rezultatul ultimei expresii

```
var i = 2;
var j;
console.log(j = i, j += i, j ** 2); // afiseaza 16 in consola
```

4.5 Structuri conditionale

a. if

```
if (conditie) {
      instructiuni care se vor executa doar daca
```

```
      evaluarea conditiei returneaza valoarea true
}

if (conditie) {
      instructiuni care se vor executa doar daca
      evaluarea conditiei returneaza valoarea true
} else {
      instructiuni care se vor executa doar daca
      evaluarea conditiei returneaza valoarea false
}

if (conditie1) {
      instructiuni care se vor executa doar daca
      evaluarea conditiei 1 returneaza valoarea true
} else if (conditie2) {
      instructiuni care se vor executa doar daca
      evaluarea conditiei 2 returneaza valoarea true
} else {
      instructiuni care se vor executa doar daca
      evaluarea conditiei 2 returneaza valoarea true
}
```

b. switch

```
switch (expresie) {
      case n:
            bloc de instructiuni
            break;

      case m:
            bloc de instructiuni
            break;

      default:
            bloc de instructiuni
}
```