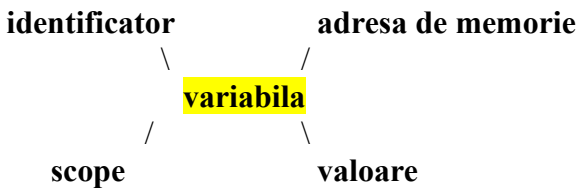
17.01.2018

Buna Teo,

Variabilele

Variabila = zona de stocare in memorie
- identificata printr-o **adresa de memorie**
- si care este asociata unui nume simbolic = **identificator**
- si care contine o **valoare**, de exemplu: 8, 'cartof', 45.3, true

Numele variabilei este modul uzual prin care facem referire la valoarea stocata
Valoarea variabilei se poate modifica (poate varia) pe parcursul rularii aplicatiei
=> de aici vine si numele de variabila (pentru ca variaza)



Scope-ul se refera la contextul in care exista variabila (variable context): unde este declarata, "unde traieste", cat timp "traieste", de exemplu:

- daca variabila este declarata la nivel global (nu intr-o functie) – atunci ea exista atata timp cat pagina de web este deschisa;
- daca variabila este declarata la nivel local (intr-o functie) – atunci ea exista doar atata timp cat ruleaza functia – adica cateva milisecunde, dupa care dispare. Daca functia este apelata de mai multe ori, se va crea mereu o alta variabila.

Valoarea variabilei este stocata intr-o zona de memorie. Identificatorul este stocat intr-o alta parte a memoriei, iar acesta va contine adresa la care este stocata valoarea variabilei. (Daca nu e clar, please tell me cand ne intalnim)

```
var catel = 'Pepi';   // declarare si asignare valoare (initializare)
      2    3    1
```

1. Mai intai este creata valoarea 'Pepi' sub forma de biti, ex: 1010100001010101 si este stocata undeva in memorie. Valoarea Pepi poarta numele de **literal**.
2. Este creata variabila **catel** in execution context.
3. In variabila catel este scrisa adresa de memorie unde a fost creat literalul.

Daca scriem doar:
```
var catel;   // doar declarare
```
atunci este creata doar variabila in execution context si aceasta va avea valoarea **`undefined`**.

Variabilele globale se vad peste tot in fisierul .js, precum si din alte fisiere .js (daca mai tii minte, am facut asta sambata asta).
Variabilele local se vad numai in functia respectiva si pot fi folosite doar acolo. - e si evident, pentru ca functia dispare in cateva milisecunde, la fel si variaibila, si de aceea, variabila nu poate fi apelata dintr-un alt context (adica dintr-o alta functie sau din contextul global – pentru ca ea nu mai exista)

Functiile

De ce, oare, s-au inventat functiile?
Pai, sa presupunem ca avem 10 elevi carora trebuie sa le calculam media generala. Avem doua varianta: ori scriem codul de calculare a mediei de 10 ori, ori il scriem o singura data - cream o functie careia sa ii pasam notele apeland-o de 10 ori si ea sa faca calculul. Care varianta e mai usoara?

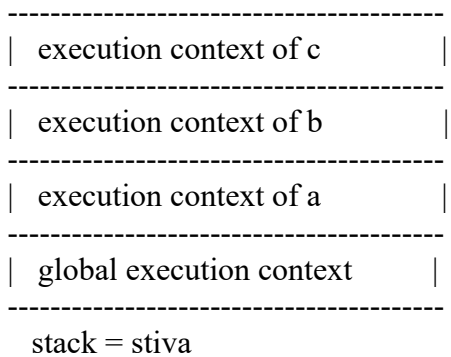
Execution stack

O functie poate sa apeleze o alta functie.

```
function a(){
      .........
      b();   // a o apeleaza pe b
      ........
}
function b(){
      ......
      c();  // b o apeleaza pe c
      ......
}
function c(){
      ......
}

....
a();    // o apelam pe a   // aici suntem in contextul global
...
```

Mai intai ruleaza linia **a();**
Apoi ruleaza functia a pana la linia unde o apelam pe b
Apoi ruleaza functia b pana la linia unde o apelam pe c
Apoi ruleaza functia c
Apoi programul revine unde a ramas in functia b si termina de rulat pe b
Apoi programul revine unde a ramas in functia a si termina de rulat pe a
Apoi programul revine la linia a(); si continua mai departe pana termina
Aici am facut exemplul cu piramida.



stack = stiva

returnare vs afisare
Daca mai ai neclaritati, te rog sa imi spui si revenim

argumente vs parametrii

```
function faCeva( x ){       // x se numeste parametru si ia valoarea
                                'frigider'

}
faCeva ('frigider');    // 'frigider' este un argument pe care il pasam
                           functiei faCeva() sa faca ceva cu el
```

Sa nu uitam sa vorbim de
- creation phase
- execution phase
- hoisting

Tipurile variabilelor

primitive	**referinte catre un obiect**
undefined – cand o variabila nu este initializata noi sa nu folosim valoarea asta niciodata **null** – daca vrem sa facem ca variabila nu mai aiba nici o valoare **boolean** **number** **string** **symbol**	**Referinta catre un obiect** vorbim despre asta mai tarziu

Javascript este un limbaj de tipul ***"Dynamic (weak) typing"***, adica chiar daca primitivele ocupa memorie diferita (de exemplu o variabila de tip number ocupa 8 bytes, iar o variabila de tip string ocupa 2 bytes * numarul de caractere) putem face asta:

```
var a = 45;
a = 'catel';
```

In limbaje de tipul ***"Static (strong) typing"*** cum e Java, acest lucru nu este posibil.

```
var a = 45;
console.log(typeof a);  // afiseaza 'number'
a = 'catel';
console.log(typeof a);  // afiseaza 'string'
```

1 byte ne mai numeste si 1 octet, pentru ca contine 8 biti (care pot fi 0 sau 1).

Sa nu uitam sa vorbim de hexazecimal

this si window

Am vazut ca in global environment avem cateva variabile predefinite:
window si this care se refera intr-o pagina web la acelasi obiect.
O sa vorbim pe indelete despre obiecte
Am vazut ca obiectul window are atribute si functii.

Daca in consola scriem: window si apoi Enter si iar Enter putem vedea toate atributele si functiile obiectului **window**
Jos de tot putem vedea atributul **console**.
Pe la inceput avem pe **document**.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Schimba culoarea</title>
  </head>
  <body>
    Introdu culoarea fundalului:
    <input type="text" id="culoare">
    <button onclick="schimbaCuloarea();">Schimba culoarea</button>
    <script>
             function schimbaCuloarea(){
                    var culoare = document.getElementById('culoare').value;
                    var element = document.getElementsByTagName('html')[0];
                    console.dir(element);
                    element.style.backgroundColor = culoare;
             }

    </script>
  </body>
</html>
```

Daca scriem intr-un fisier:
```
var a = 'palmier';
function b(){ ..... }
```
si ne uitam in consola la obiectul **window**, o sa vedem acolo si pe a si functia b.

Exercitiu:
In codul de mai sus cu schimba culoarea, am modificat pe console.**log**(element) in console.**dir**(element). Ruleaza aplicatia, apasa pe butonul de schimba culoarea si uita-te in consola. Ce a afisat? Deschide obiectul.

Taguri HTML

tagul <input>
```
<input type="text" id="culoare">
```
Type poate sa ia urmatoarele valori:
Exercitiu:
Incearca-le pe cele pe care le-am notat cu steluta.

button *
checkbox *
color *

```
date     *
datetime-local     *
email
file     *
hidden     *      - la ce foloseste oare asta?
image     - de facut la ora
month     *
number     *   incerca sa scrii ceva
password     *
radio     *
range     *
reset
search     *
submit     *
tel     *
text     *
time     *
url
week     *
```

Sambata o sa
verificam tema cu stapanul catelului
continuam tagurile html
facem niste programele

Scuze de intarziere