JavaScript et HTML

Cours 2

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-js

Plan

- scalaires en JavaScript
- tableaux
- chaînes de caractères
- itérations
- tableaux multidimensionnels
- alias

Ecrire des programmes JavaScript

4 manières:

- utiliser un système intégré (Visual Studio ou autre)

- utiliser une simple fenêtre terminal et un éditeur de texte (Emacs, VI, TextEdit, ..)

exemple

• sur la fenêtre terminal, on tape:

```
mac$ node
Welcome to Node.js v14.18.0.
Type ".help" for more information.
>
```

• et on peut fonctionner en mode calculette:

```
> 23+42
65
> 438 * 234
102492
> (438 * 234) + 35
102527
> ((438 * 234) + 35) / 3)
102503.66666666667
```

- dans la console JavaScript d'un navigateur

- dans du code embarqué dans un fichier HTML (page web)

Scalaires

- on déclare des variables et des constantes

```
let x
```

undefined value

```
let y = 9
var z = 10
```

obsolete

```
const t = 3, u = 5, v = 10
```

déclaration multiple

```
x = 3
```

```
t = 11
```

interdit

Types de données

`String`	represents textual data	`'hello'`, `"hello world!"` etc
`Number`	an integer or a floating-point number	`3`, `3.234`, `3e-2` etc.
`BigInt`	an integer with arbitrary precision	`9007199254740999n`, `1n` etc.
`Boolean`	Any of two values: true or false	`true` and `false`
`undefined`	a data type whose variable is not initialized	`let a;`
`null`	denotes a `null` value	`let a = null;`
`Symbol`	data type whose instances are unique and immutable	`let value = Symbol('hello');`
`Object`	key-value pairs of collection of data	`let student = { };`

Types de données

- les types sont dynamiques en JavaScript
- vérifiés à l'exécution
- `typeof` donne le type d'une expression
- conversions de types implicites
- conversions de types explicites

Conversions de type implicites

```
let x;

x = '3' + 2;
x = '3' + true;
x = '3' + undefined;
x = '3' + null;

x = '4' - '2';
x = '4' - 2;
x = '4' * 2;
x = '4' / 2;

x = 'hello' - 'world';
x = '4' - 'hello';

x = '4' - true;
x = 4 + true;
x = 4 + false;

x = 4 + undefined;
x = 4 - undefined;
x = true + undefined;
x = null + undefined;
```

conversion en chaînes de caractères

conversion en nombres

conversion en nombres

conversion en nombres

conversion en NaN. (not a number)

Conversions de type explicites

```
let x;

x = Number('324');
x = Number('324e-1')
x = Number(true);
x = Number('hello');
x = Number(undefined);
x = Number(NaN);

x = String(2 + 4);

x = Boolean(0);
```

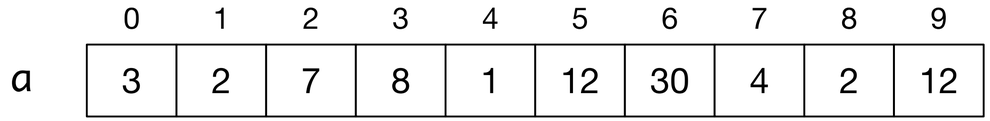
Value	String Conversion	Number Conversion	Boolean Conversion
1	"1"	1	`true`
0	"0"	0	`false`
"1"	"1"	1	`true`
"0"	`"0"`	0	`true`
"ten"	`"ten"`	`NaN`	`true`
`true`	"true"	1	`true`
`false`	"false"	0	`false`
`null`	"null"	0	`false`
`undefined`	"undefined"	`NaN`	`false`
''	""	0	`false`
' '	" "	0	`true`

Tableaux

- données structurées: tableaux

```
let a = [3, 2, 7, 8, 1, 12, 30, 4, 2, 12]

console.log (a, a[0], a[5], a[-1])
```



- les tableaux sont modifiables

```
a[7] = 28

console.log (a)
```

- les tableaux de JavaScript peuvent ne pas être homogènes

```
a[9] = 'aaa'

console.log (a)
```

```
[3, 2, 7, 8, 1, 12, 30, 28, 2, 'aaa']
```

- en JavaScript, les tableaux sont des objets, donc des paires (clé, valeur)

```
console.log (a[35])
```

```
undefined
```

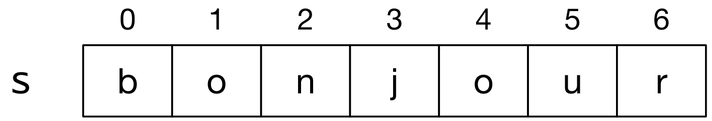
Chaînes de caractères

- les chaînes de caractères sont des tableaux non modifiables de caractères

```
let s = ‘bonjour'

console.log (s, s.length, s[0], s[1], s[2])
```

```
bonjour 7 b o n
```




```
s[3] = ‘Z'

console.log (s)
```

```
bonjour
```

- les chaînes de caractères ont 3 écritures possibles:

```
s = "coucou"
t = 'coucou'
u = `coucou`
```

```
c1 = 'AA', c2 = 'BB'
t = `abc${c1}defg${c2}hi`
console.log (t)
```

```
abcAAdefgBBhi
```

Tableaux

- itération sur un tableau

```
let r = 0
    a.forEach (m => {
        r = r + m
    })

console.log (r)
```

itération sur tous les éléments de a

- idem avec une fonction

```
function sum_of (x) {
    let r = -1
    x.forEach (m => {
        r = r + m
    })
    return r
}

console.log (sum_of (a))
```

x est l'argument de la fonction

Tableaux

- itération sur un tableau (autre écriture possible)

```
let r = 0
    a.forEach (function (m) {
        r = r + m
    })

console.log (r)
```

itération sur tous les éléments de a

- idem avec une fonction

```
function sum_of (x) {
    let r = -1
    x.forEach (function(m) {
        r = r + m
    })
    return r
}

console.log (sum_of (a))
```

x est l'argument de la fonction

Tableaux

- itération sur un tableau

```
let r = -1
    a.forEach (m => {
        if (m > r)
            r = m
    })

console.log (r)
```

a est un tableau de nombres positifs

- idem avec une fonction

```
function max_of (x) {
    let r = -1
    x.forEach (m => {
        if (m > r)
          r = m
    })
    return r
}

console.log (max_of (a))
```

x est l'argument de la fonction

Tableaux

- itération sur un tableau

```
let r = -1
    for (let i = 0; i < a.length; i++) {
        if (a[i] > r)
            r = a[i]
    }

console.log (r)
```

a est un tableau de nombres positifs

- idem avec une fonction

```
function max_of (x) {
    let r = -1;
    for (let i = 0; i < x.length; i++) {
        if (x[i] > r)
            r = x[i];
    }
    return r
}
```

x est l'argument de la fonction

Tableaux

- plus petite valeur entière sur 64 bits

```
let MIN_INT = Number.MIN_SAFE_INTEGER
```

- itération sur un tableau de nombres entiers arbitraires

```
let r = MIN_INT
    a.forEach (m => {
        if (m > r)
            r = m
    })

console.log (r)
```

- idem avec une fonction

```
function max_of (x) {
    let r = MIN_INT
    x.forEach (m => {
        if (m > r)
          r = m
    })
    return r
}

console.log (max_of (a))
```

x est l'argument de la fonction

Tableaux

Exercice Écrire la fonction qui trouve l'indice de la plus grande valeur

Exercice Écrire la fonction qui teste le nombre de 'a' dans une chaîne de caractères

Exercice Écrire la fonction qui retourne l'indice du premier 'abc' dans une chaîne de caractères

Exercice Écrire la fonction qui teste si une chaîne de caractères est un palindrome

Tableaux multi-dimensionnels

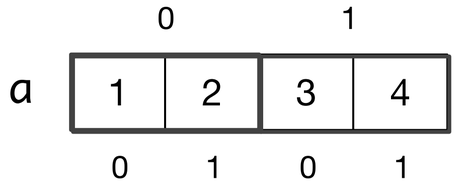
- une matrice est une liste de listes

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
let a = [[1,2],[3,4]]

console.log (a[0][0], a[0][1], a[1][0], a[1][1])
```

```
1 2 3 4
```



- impression de matrices

```
function print_matrix (a) {
   a.forEach (line => {
      s = ''
      line.forEach (elt => {s = s + ' ' + elt})
      console.log (s)
   })
}
```

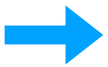
- addition de matrices

```
function add (a, b) {
    m = a.length, n = a[0].length
    mb = b.length, nb = b[0].length
    if (m != mb || n != nb) {
       console.log ("Erreur! "); return
    }
    r = new_matrix (m, n)
    for (let i=0; i < m; ++i)
      for (let j=0; j < n; ++j)
        r[i][j] = a[i][j] + b[i][j]
    return r
}
```

Tableaux multi-dimensionnels

- création de matrices

```
function new_matrix (m, n) {
    let r = [];
    for (let i = 0; i < m; ++i)
      for (let j = 0; j < n; ++j)
        r[i] = [];
    return r;
}
```



à comprendre plus tard !

Tableaux multi-dimensionnels

- création d'une matrice pleine de zéros

```
def new_matrix (m, n) :
    a = []; z = [0]*n
    for i in range(m): a.append (z.copy())
    return a
```

à comprendre plus tard !

- carré magique

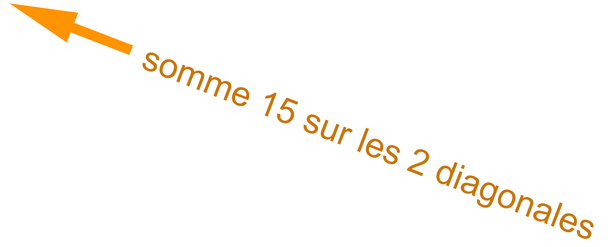
```
function magique (n) {
    a = new_matrix (n, n)
    i = n - 1
    j = Math.floor (n / 2)
    for (let k = 0; k < n*n; ++k) {
        a[i][j] = k + 1;
        if ((k + 1) % n == 0)
           i = (i - 1) % n;
        else {
            i = (i + 1) % n
            j = (j + 1 ) % n
        }
    }
    return a
}
```

n impair

```
print_matrix (magique(3))
```

```
4 9 2
3 5 7
8 1 6
```

somme 15 sur lignes et colonnes



somme 15 sur les 2 diagonales

```
print_matrix (magique(7))

 22  31  40  49   2  11  20
 21  23  32  41  43   3  12
 13  15  24  33  42  44   4
  5  14  16  25  34  36  45
 46   6   8  17  26  35  37
 38  47   7   9  18  27  29
 30  39  48   1  10  19  28
```

Tableaux

- valeur d'un tableau (*list*)

```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> b = a
>>> b
[1, 3, 4, 2, 3, 5, 9]

>>> b [3] = 42
>>> b
[1, 3, 4, 42, 3, 5, 9]

>>> a
[1, 3, 4, 42, 3, 5, 9]
```

a a été modifié ! !

- pourquoi ?

Tableaux

- alias

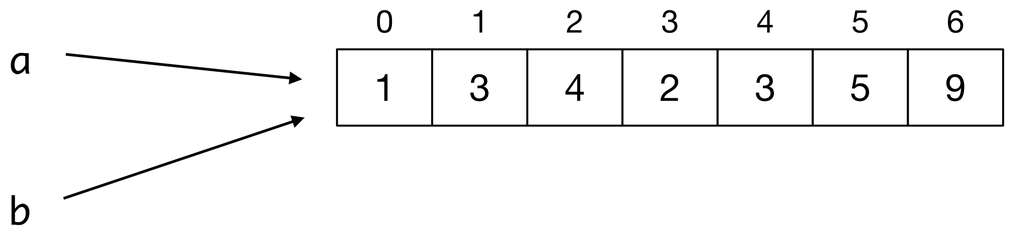
```
>>> a = [1, 3, 4, 2, 3, 5, 9]

>>> b = a
>>> b
[1, 3, 4, 2, 3, 5, 9]

>>> b [3] = 42
>>> b
[1, 3, 4, 42, 3, 5, 9]

>>> a
[1, 3, 4, 42, 3, 5, 9]
```

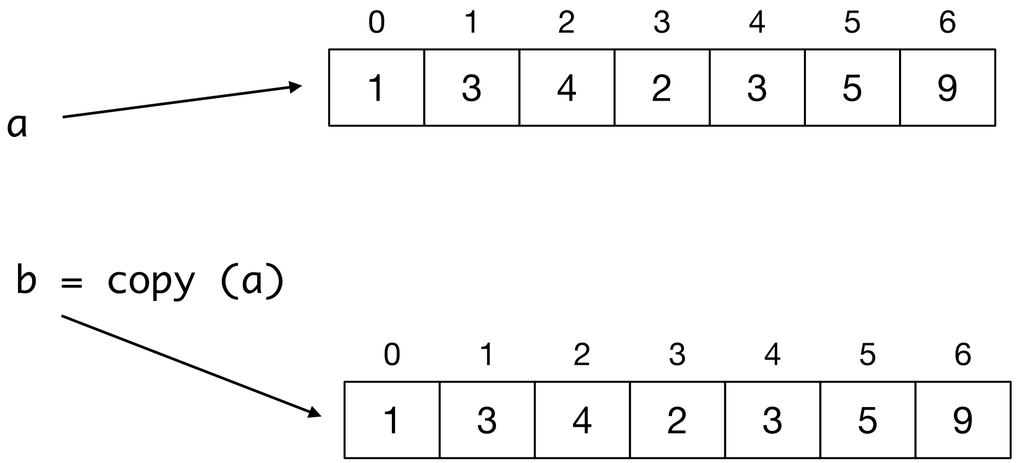
a et b sont des alias



- la valeur de a est son adresse mémoire

Tableaux

Exercice Écrire la fonction qui copie un tableau



Prochain cours

- un bon tutorial JavaScript: `http://www.programiz.com/javascript`
- objets en JavaScript