TP 1 – Variables et fonctions

Dans la suite du document les annotations [ES6] indiquent les éléments introduits dans la norme ECMAScript 2015 (6eme édition).

1. Introduction

Javascript est un langage de programmation principalement utilisé pour la conception de pages web interactives. Récemment il est également devenu un langage utilisé côté serveur (voirNode.js par exemple).Le langage javascript utilise les paradigmes impératif(comme le C), objet (comme le Java ou le C++) et fonctionnel(comme certains aspects de python, de Ocaml et autres).

Il est très facile de mal coder en javascript qui est un langage très (trop?) tolérant. Dans ces séances nous allons essayer de coder le plus proprement possible. En particulier:
-Nous allons utiliser le mode strict (cf bloc 2)
-nous allons valider le code avec jslint https://www.jslint.com/

Pour le développement en JS, vous pouvez écrire vos programmes :

- Soit dans la console de votre navigateur dans le cas de tests de programmes
- Soit utiliser un editeur de texte (ide) de préférence Visual Studio Code en créant un fichier html et un fichier JS contenant les différents exercices.

2. Variables

La portée des variables en javascript peut être soit globale (var) soit limitée à une fonction (var) ou à un bloc (let) ou encore constante (const).
Les déclarations de variables de type var et les fonctions sont remontées (hissage ou hoisting) en début de bloc mais pas les let ou les var. On peut ainsi utiliser une fonction avant de la définir. Attention cependant, les affectations ne sont pas remontées et tant qu'une variable n'est pas affectée son type est non défini (undefined) !
Une variable non déclarée devient toujours globale sauf si on se place en mode strict auquel cas cela lève une exception ReferenceError.

```
"use strict" ; // passage en mode strict

/* définition de variable */
var ma_variable ;   // variable globale ou locale à une fonction
let ma_variable2 ;  // variable locale à un bloc [ES6]
const c = 12 ;      // variable non modifiable et devant être initialisée [ES6]

// affectation de variable
ma_variable = "hello" ;

// définition, affectation et concaténation (+) de chaine
var mon_autre_variable = ma_variable + " world" ;

// affichage dans la console
console.log(mon_autre_variable) ;

// affichage du type d'une variable (opérateur typeof)
console.log(typeof "toto");
```

Exercice 1. Qu'affichent les codes suivants en mode strict et non strict ?

```
t = 2;
console.log(t);
```

```
t = 2;
console.log(t);
var t=4;
console.log(t);
```

```
t = 2;
console.log(t);
let t=4;
console.log(t);
```

```
console.log(y);
console.log(x);
let x = 2;
var y = 1;
```

Exercice 2. Qu'affichent console.log("12"+"34"), console.log(12+34), console.log(12+ "34") ?

3. Fonctions

```
// definition de function
function ma_function(argument1, argument2, ...) {
  ...
  return ...; // optionnel
}

// appel de fonction
ma_function(2, 3, "toto") ;

// expression de fonction ( !, non remontée au début du code)
const addTwo = function (x) {return x+2 ;}

// function avec paramètre par défaut    [ES6]
function une_function(param1,param2=20){
…
    return ...; // optionnel
}

// appel de fonction
une_function (2) ;        //param2 égal à 20
une_function(2,30) ;      //param2 égal à 30
```

Exercice 3. Ecrire les fonctions suivantes :

- **hello_world** : affiche "Hello World !" dans la console.
- **hello_prenom** : prend en argument un prénom et affiche « bonjour prénom » dans la console.
  hello("marc") -> "bonjour marc"
- **calcul_prixTTC** : prend en argument un prix hors taxe et la TVA ayant une valeur par défaut de 20% et qui renvoie le prix TTC du produit. Utiliser cette fonction avec une TVA par défaut ainsi qu'une TVA à 5.5%.
- Un paramètre par défaut peut-il être passé comme premier paramètre d'une fonction ?

4. Les fonctions fléchées (arrow function) [ES6]

```
// fonction fléchée ( ! pas de this) [ES6]
const addOne = x => x+1;

// presque équivalent à
function addOne(x) {return x+1 ;}


//différentes écritures selon le nombre de parametres et bloc de code

const arrow_fct1 = () => 'hello' ;
const arrow_fct2 = x => x+1 ;
const arrow_fct3 = (x) => x+1 ;
const arrow_fct4 = (x,y) => x+y  ;
const arrow_fct5 = x => { if(x>20 || (x<0))
                            return 'erreur'
                          else
                            return 'dans la plage'
                        };
```

Remarques :

- Nous utiliserons ses fonctions lorsque nous travaillerons sur les tableaux avec les methods reduce/map/filter
- Les fonctions fléchées non pas de 'this', elle partage le même 'this' lexical que leur scope parent. Nous travaillerons dessus lors de l'étude des objets.

Exercice 4. Ecrire les fonctions fléchées suivantes suivantes :

- **test_arrow** : qui affiche le texte 'voici le texte de la fonction'
- **test_arrow 2** : qui affiche le texte 'voici le texte de la fonction' et le contenu de this (Qu'en déduisez vous ?)
- **multi_x** : prend un paramètre x et renvoi la multiplication de x par 10
- **multi_x_y** : qui prend les 2 paramètres x et y est renvoie la multiplication de x par y

5. Test if/else

```
if (condition) {
  instructions;
} else {
  instructions;
}

// opérateur conditionnel
let x = (condition)?expression_si_vrai:expression_si_false;

switch(expression) {
  case a:
    instructions;
    break;
  case b:
    instruction;
    break;
  default:
    instructions;
}
```

Exercice 5. Ecrire les fonctions suivantes avec if ou switch/case selon la pertinence

- **max** : prend deux nombres en argument et retourne le plus grand. Ecrire avec if et opérateur conditionnel.
  max(1,3) -> 3
- **max3** : comme max mais avec 3 arguments
- **jour** : prend un numéro de jour entre 1 et 7 et retourne le nom du jour (ne pas utiliser de tableau)
  jour(3) -> "mercredi"

6. Boule for/while

```
for (initialization; condition d'arrêt; incrément) {
  instructions;
}
while (condition) {
  instructions ;
}
do {
  instructions ;
} while (condition);

// l'instruction break permet de sortir complètement d'une boucle
// l'instruction continue permet de terminer l'itération courante de la boucle
```

Exercice 6. Ecrire les fonctions suivantes :

- **factorielle_it_for** : calcule et retourne la factorielle d'un nombre avec une boucle for (n ! = n*(n-1)*…*2*1)
- **factorielle_it_while** : idem avec une boucle while
- **factorielle_rec** : idem en récursif
- **exposant_rec** : calcule et retourne l'exposant x d'un nombre y en récursif.
  exposant(2,3) -> 8
  exposant(5,2) -> 25

7. les paramètres rest d'une fonction [ES6]

Recherchez à quoi correspond les paramètres « rest » en JS

Exercice 7. Ecrire la fonction suivante :

- **calcul** : qui attend un paramètre rest et qui calcul la moyenne de tous les éléments passés via ce paramètre

8. Le linter eslint

Nous avons utilisé lors des premiers exercices le jslint (le premier linter crée et le plus simple). Nous souhaitons maintenant installer sur visual studio code le linter eslint.

Documentation : https://eslint.org/
Plugin visual studio code : ESLint de Dirk Baeumer

Vous aurez besoin dans votre de dossier d'exécuter les commandes suivantes :

- npm init
- npm install eslint
- ./node_modules/.bin/eslint –init
  Puis suivre la création en comprenant les différentes demandes
  Utiliser les règles de code js recommandées puis celles de airbnb Quelles sont les différences ? Comment pouvez-vous définir vos problèmes règles supplémentaires en par exemple autorisant le console.log() qui est tout de même très pratique.