ISEN ALL IS DIGITAL! OUEST | yncréa

JavaScript
- Bases du langage -

Groupe des étudiants : CIR1

Syntaxe

Généralités

- console.log("text") affiche un texte à la console
- La syntaxe de JavaScript est très similaire à celle du C
  - Les variables sont sensibles à la casse (mavariable et maVariable sont deux variables différentes)
  - Les instructions sont séparées par ;
  - Les commentaires s'écrivent avec // ou /* */

```
// ceci est un commentaire sur une ligne

/* ceci est un commentaire
sur plusieurs lignes */
```

  - Les blocs des boucles, fonctions, … sont délimités par { }

Variables

- Nom d'une variable
  - Commence par une lettre ou un underscore ;
  - Ne contient que des lettres, des chiffres et des underscore ;
  - Est sensible à la casse.
- Typage dynamique
  - Le type d'une variable n'est pas déclaré mais déterminé à l'exécution
  - Il peut changer suite à une opération
- Les variables se déclarent avec le mot clé let ⇒ la portée d'une variable est le bloc dans laquelle elle est déclarée (comme en C),
- Une variable non initialisée possède la valeur undefined
- Exemple :

```
let a;
console.log(a); // => undefined

a = 1;          // a est un nombre
console.log(a); // => 1

a = "hello";    // a est maintenant une chaîne de caractères
console.log(a); // => hello (sans ")
```

Types des variables : Boolean

- Boolean : True ou False
  - Déclaration

```
let var1 = true;
let var2 = false;
```

  - Le résultat de l'évaluation d'une expression logique est de type Boolean

```
let a = 42;

let b = ( a == 1 );
let c = ( a < 50 );
let d = ( a % 2 == 0 );

console.log(b); // => false
console.log(c); // => true
console.log(d); // => true
```

Types des variables : Number

- Number : nombres entiers ou réels.

- Number possède deux valeurs particulière :
  - Infinity : utilisé pour modéliser l'infini
  - NaN : utilisé pour modéliser le résultat d'une opération invalide

```
console.log(1/Infinity);          // => 0
console.log(0/0);                 // => NaN
console.log(Math.log(-1));        // => NaN
console.log(1/0);                 // => Infinity
console.log(-1/0);                // => -Infinity
```

```
let a = 0/0;
console.log(a == NaN);        // => false
console.log(isNaN(a));        // => true
```

Types des variables : chaîne de caractères

- Déclaration

```
1 let var1 = "Hello World!";
```

Les chaînes de caractères vont être traitées en détail dans le CM3

Types des variables

- L'opérateur typeof renvoie le type d'une variable sous la forme d'une chaîne de caractères

```
console.log( typeof "hello" );      // => string
console.log( typeof true );         // => boolean
console.log( typeof 42 );           // => number
```

undefined VS null

- undefined = la variable n’a pas de valeur, elle est non affectée

```
let a;
if (a == undefined) {
  echo 'La variable a est non définie.';
}
```

- null = la variable a une valeur qui vaut vide

```
let a = null;
if (a == null) {
  echo 'La variable a est vide.';
}
```

Opérateurs

Types d'opérateurs

- Les opérateurs arithmétiques :
  - **+** : addition ;
  - **-** : soustraction ;
  - ***** : multiplication ;
  - **/** : division ;
  - **%** : modulo ;
  - ****** : exponentiation.

Types d'opérateurs

- Les opérateurs arithmétiques :

- Les opérateurs d’affectation :
  - **=** : affectation ;
  - **+=** : addition combinée à une affectation ;
  - **-=** : soustraction combinée à une affectation ;
  - ***=** : multiplication combinée à une affectation ;
  - **/=** : division combinée à une affectation ;
  - **%=** : modulo combiné à une affectation;
  - ****=** : exponentiation combinée à une affectation.

Types d'opérateurs

- Les opérateurs arithmétiques :
- Les opérateurs d'affectation :
- Les opérateurs de comparaison :
  - **==** : égalité ;
  - **===** : identité (égalité et même type) ;
  - **!=** : non-égalité ;
  - **!==** : non-identité ;
  - **>** et **<** : strictement supérieur et inférieur ;
  - **>=** et **<=** : supérieur ou égal et inférieur ou égal ;

```
console.log( 42 == 42 );      // => true
console.log( 42 == 42.0 );    // => true
console.log( 42 == "42" );    // => true
console.log( 0 == false );    // => true
```

```
console.log( 42 === 42 );     // => true
console.log( 42 === 42.0 );   // => true
console.log( 42 === "42" );   // => false
console.log( 0 === false );   // => false
```

Types d'opérateurs

- Les opérateurs arithmétiques :
- Les opérateurs d’affectation :
- Les opérateurs de comparaison :
- Les opérateurs d’incrémentation / décrémentation :
  - **++var** : pré incrémentation (avant évaluation) ;
  - **var++** : post incrémentation (après évaluation) ;
  - **--var** : pré décrémentation (avant évaluation) ;
  - **var--** : post décrémentation (après évaluation).

Types d'opérateurs

- Les opérateurs arithmétiques :
- Les opérateurs d’affectation :
- Les opérateurs de comparaison :
- Les opérateurs d’incrémentation / décrémentation :
- Les opérateurs logiques :
  - **&&** : ET logique ;
  - **||** : OU logique ;
  - **!** : NON logique.

Types d'opérateurs

- Les opérateurs arithmétiques :
- Les opérateurs d’affectation :
- Les opérateurs de comparaison :
- Les opérateurs d’incrémentation / décrémentation :
- Les opérateurs logiques :
- Les opérateurs sur les chaînes de caractères (voir CM3)

Structures de contrôle

Structures conditionnelles : If Else ElseIf

- Syntaxe :

```
if (condition1) {
  // Code à exécuter si la condition 1 est vraie.
} else if (condition2) {
    // Code à exécuter si la condition 1 est fausse et que
    // la condition 2 est vraie.
} else {
  // Code à exécuter si les deux conditions sont fausses.
}
```

Structure conditionnelle : Opérateur ternaire

- Syntaxe :

```
condition ? exprSiVrai :
exprSiFaux
```

```
let a = 1
a>0?console.log("pos"):console.log("neg")
```

Structures conditionnelles : switch

- Syntaxe :

```
switch (n) {
  case label1:
    // Code à exécuter si n == label.
    break;
  case label2:
    ...
  default:
    // Code à exécuter si n différent de tous les labels.
}
```

Structures itératives : while & do while

- Syntaxe :

```
1 while (condition) {
2   // Code à exécuter tant que la condition est vraie.
3 }
```

```
1 do {
2   // Code à exécuter tant que la condition est vraie.
3 }
4 while (condition);
```

Structures itératives : For

- Syntaxe :

```
for (init counter; test counter; increment counter) {
    // Code à exécuter.
}
```

```
for (let i = 0; i < 10; i++) {
  console.log('The number is: ' + i);
}
```

Structures itératives : For/Of

- Parcours des éléments d'un objet itérable

```
1 let tab = ['Apple', 'Pear'];
2 for (element of tab) {
3   console.log(element);
4 } // Affiche : Apple Pear
```

Fonctions

Définition et appel

- Une fonction se déclare avec le mot clé function
- Pas de séparation prototype / fonction comme en C
- Pas de type de retour (logique compte tenu du typage dynamique)

- Exemple :
  - Définition :

```
function square(x){
    return x * x;
}
```

  - Appel :

```
let i = 3;
let squareI = square(i);
console.log(squareI);          // => 9
```

Fonctions mathématiques

- L'objet global Math contient des constantes et des fonctions mathématiques

- Exemples :

```
console.log( Math.PI );  // => 3.141592653589793
console.log( Math.E );   // => 2.718281828459045
console.log( Math.abs(-4) );          // => 4
console.log( Math.pow(2, 3) );        // => 8
console.log( Math.sin(Math.PI/2) );   // => 1
```

Bonnes pratiques

Test

- Les tests à effectuer doivent être encapsulés dans une fonction qui sera appelée en fin de script

```
function main(){

    let i = 2;
    console.log(typeof i);
}

main();
```

Blocs de code

- Toujours délimiter un bloc de code par des accolades, même s'il ne comporte qu'une seule ligne

```
if (x > 1)
    y = 2;
```

→

```
if (x > 1){
    y = 2;
}
```

- Indenter correctement votre code : ajouter une tabulation à chaque fois qu'un bloc est inclus dans un autre

```
for(let i = 1; i < 5; i++){
for(let j = 1; j < 5; j++){
k=i*j;
if(k%2==0){
console.log(k)
}}}
```

→

```
for(let i = 1; i < 5; i++){
    for(let j = 1; j < 5; j++){
        k=i*j;
        if(k%2==0){
          console.log(k)
        }
    }
}
```