Javascript module 1

Les variables et les conditions

Les variables

Déclaration et nommage

Dans tous les langages de programmation, les variables permettent de stocker des données pendant l'exécution afin de les manipuler.

En javascript, les variables doivent être déclarées et nommées avant d'être utilisées, avec ***let*** ou ***const***.

Pour leur assigner une valeur on utilise le symbole =.

- ***const*** permet de déclarer une valeur qui ne change pas.

```
const myName = 'Kevin';
```

- ***let*** permet de déclarer une valeur qui peut changer au cours de l'exécution.

```
let myAge = 27;
```

Info: Vous rencontrerez parfois des déclarations de variable avec ***var*** qui a été supplanté par ***let*** et ***const***.

Le nom d'une variable doit impérativement commencer par une lettre et ne doit contenir que des lettres et des chiffres, sans caractères spéciaux.

> Info: Il y a des noms de variable interdits : break, case, catch, class, const, continue, debugger, default, delete, do, else, enum, export, extends, false, finally, for, function, of, import, in, instanceof, new, null, return, switch, super, this, throw, true, try, typeof, var, void, while, with

L'usage veut que l'on utilise le camelCase pour nommer les variables javascript : tout en minuscule, seule la première lettre de chaque mot est en majuscule pour les délimiter.

```
let gameModeList;

let playerName;

let isMyCodeWorking;
```

Les types de valeur de base

Une variable n'est pas typée en javascript. C'est-à-dire qu'une variable peut changer de type de valeur en cours d'exécution du script, contrairement à beaucoup d'autres langages.

Il est cependant recommandé de ne pas changer le type de valeur d'une variable.

Il est possible de connaître le type d'une variable ou d'une valeur en la faisant précéder du texte ***typeof***.

Les nombres

Les nombres peuvent être entiers ou réels, positifs ou négatifs.

```
let myNumber = 12;

myNumber = -12;

myNumber = 1.1345;

typeof myNumber; // "number"
```

Vous pouvez utiliser les 5 opérateurs :

- **+** addition
- **-** soustraction
- ***** multiplication
- **/** division
- **%** reste d'une division euclidienne

```
let a = 3 + 4; // 7

let b = a - 2; // 5

let c = b * 20; // 100

let d = c / 2; // 50

let e = d % 3 ; // 2     => 16 * 3 + 2 = 50
```

Vous pouvez également appliquer les opérateurs directement sur une variable comme ceci :

```
let a = 7;

a += 2; // 9
```

```
// équivaut à
a = a + 2;


let b = 5;
b -= 2; // 3
// équivaut à
b = b - 2;


let c = 100;
c *= 2; // 200
// équivaut à
c = c * 2;


let d = 50;
d /= 2; // 50
// equivalent to
d = d / 2;


let e = 50;
e %= 3 ; // 2
// equivalent to
e = e % 3;
```

Il est également possible d'incrémenter ou de décrémenter une variable comme ceci :

```
let a = 7;
a++; // 8
// equivalent to
a += 1;
// equivalent to
```

```
a = a + 1;


let b = 12;
b--; // 11;
// equivalent to
b -= 1;
// equivalent to
b = b - 1;
```

Les chaînes de caractère : les String

Une variable peut contenir une chaîne de caractères. La valeur doit être entouré de “ (guillemet), de ‘ (apostrophe), ou de ` (accen grave);

```
let myVar = "hello world";
// ou
let myVar = 'hello world';
// ou
let myVar = `hello world`;
typeof myVar; // "string"
```

Vous pouvez connaître la longueur de la chaîne avec la propriété ***.length***.

```
myVar.length; // 11
```

Le caractère *+* permet de concaténer des chaînes.

```
const firstWord = "hello";
const secondWord = " world";
const sentence = firstWord + secondWord; // "hello world"
```

Attention, un nombre entre guillemets est un texte.

```
let myText = "42"; // This is a text
let myNumber = 42;  // This is a number
```

Ce qui revient à avoir ce genre de comportement :

```
let a = "4";

let b = 2;

let theResponse = a + b; // “42”
```

Les méthodes de manipulations

Les chaînes de caractères peuvent êtres manipulés avec des méthodes spécifiques (remplacer du texte, extraire une partie d’un texte, passer en majuscule, ...) que vous trouverez ici :

- https://www.w3schools.com/js/js_string_methods.asp

Les gabarits de texte

Les gabarits permettent de construire des textes en les complétant avec des valeurs contenues dans des variables.

```
let sentence = `${name} drank ${nbOfCoffee} coffees today.`;

// equivalent to

let sentence = name + " drank " + nbOfCoffee + " coffees today.";
```

Les booléens

Il n’y a que 2 valeurs possibles pour un booléen :

- ***true***
- ***false***

Il existe 3 opérateurs logiques qui permettent de manipuler les booléens :

- ***&&*** c’est le ET (AND)
- || c’est le OU (OR)
- ***!*** c’est la Négation (NOT)

```
let a = true;

let b = false;

typeof a; // "boolean"


let c = !a; // false

let d = !b; // true
```

```
let e = a && b; // false
e = a && d; // true

let f = a || b; // true
f = c || b; // false
f= !(c || d); // false
```

Petit moyen pour s'en sortir avec les ET et les OU :

- Pour avoir ***true*** avec un ET, il faut que les 2 valeurs soient ***true***.
- Pour avoir ***true*** avec un OU, il faut qu'au moins l'une des 2 valeurs soit à ***true***.

Les conditions

Les opérateurs de comparaison

Il existe en javascript 8 opérateurs de comparaison :

- *==* égal à
- ***!=*** différent de
- *===* contenu et type de variable égal à
- ***!==*** contenu et type de variable différent de
- *>* supérieur à
- *>=* supérieur ou égal à
- *<* inférieur à
- *<=* inférieur ou égal à

Ces opérateurs permettent de comparer 2 valeurs et retournent le booléen correspondant.

```
let a = 12 < 15; // true;
let b = 10 >= 19; // false;
typeof a; // "boolean"

let theResponse1 = 42 == "42"; // true;
```

```
let theResponse2 = 42 === "42"; // false;


let me = "go" != "goes"; // true
```

Si vous avez plus de 2 valeurs à comparer, vous devrez certainement intégrer des opérateurs logiques, comme ceci :

```
let theUltimateQuestion = 42 === "42" && "World" != "world"; // false
```

Déclarer des conditions

if, else, else if

Les conditions permettent d'interpréter les booléens et d'exécuter un code en fonction de sa valeur.

```
if (condition) {
    // code to execute when condition is true
} else {
    // code to execute when condition is false
}
```

Dans le code précédent, ***"condition"*** peut-être une variable contenant un booléen, ou directement une expression faite d'opérateurs logiques et / ou de comparaison.

```
// Who's older ?
let whoIsOlder;
const yourAge = 45;
const myAge = 28;
if (yourAge > myAge) {
    whoIsOlder = "You're older than me.";
} else {
    whoIsOlder = "You're younger than me.";
}
```

Il est possible d'ajouter des conditions intermédiaires à la structure grâce à un "***else if ()***" pour améliorer notre code précédent :

```
// Who's older ?
let whoIsOlder;
const yourAge = 45;
const myAge = 28;

if (yourAge > myAge) {
    whoIsOlder = "You're older than me.";
} else if (yourAge == myAge) {
    whoIsOlder = "We are the same !";
} else {
    whoIsOlder = "You're younger than me.";
}
```

Les ternaires

Les ternaires permettent de simplifier la syntaxe quand le résultat de la condition a uniquement pour finalité l'attribution d'une valeur à une variable.

```
const yourAge = 45;
const myAge = 28;
const whoIsOlder = yourAge > myAge ? "you" : "me";    // you

// equivalent to
let whoIsOlder;
if (yourAge > myAge) {
    whoIsOlder = "you";
} else {
    whoIsOlder = "me";
}
```

La console et les boîtes de dialogue

Afficher dans la console

Dans les exercices qui suivent nous allons utiliser la déclaration suivante pour afficher des données dans la console du navigateur.

```
console.log(valueToDisplay);
```

valueToDisplay est ici une variable comportant la valeur qui soit s'afficher dans la console du navigateur.

Il est également possible de styliser les affichages dans la console en utilisant d'autres méthodes que ***log()***.

```
console.info("This is an information");

console.warn("This is a warning");

console.error("Houston, we've got a problem.");
```

Demander une valeur à l'utilisateur

Il existe une fonction Javascript permettant de demander une valeur à l'utilisateur via une boîte de dialogue. Dans le code, cette fonction retourne la valeur saisie par l'utilisateur et on peut ainsi l'assigner à une variable.

```
const aValue = prompt("Give me a value!");
```

Il existe également la fonction alert() qui affiche une boîte de dialogue, sans demander de saisie à l'utilisateur.

```
alert("For your information...");
```