JAVASCRIPT

Langage de script interprété

Les Variables

int -> 64bits

Javascript fait peur aux dev Java ou C++ :

```
console.log("5" - 1)
console.log("5" + 1)
```

Les opérateurs

opérateurs binaires : +, -, *, /, %
opérateurs de comparaison : ==, !=, ===, !==, <, >, <=, >=
opérateurs unaire: ! , typeof

Liaisons

Exemples:

```
var name = "Ayda," ;
const greeting = "Hello " ;
let autre = " ça va ?" ;
console.log( greeting + name + autre+ " bien" ); // → Hello Ayda, ça va ? bien

let one = 1, two = 2 ; // déclaration de plusieurs variables avec un seul (let, var, const)

console.log(one + two);
```

Fonctions

Par exemple, la fonction Math.max prend n'importe quelle quantité d'arguments numériques et rend le plus grand nombre.

```
console.log(Math.max(2, 4)) ;
```

Ici le paramètre de console.log est le résultat de la fonction qui a fait un retour de valeur **(return**).

Lorsqu'une fonction produit une valeur, on dit qu'elle renvoie cette valeur. Tout ce qui produit une valeur est une **expression**, ce qui signifie que les appels de fonction peuvent être utilisés dans des expressions plus larges. Ici, un appel à Math.min, qui est l'opposé de Math.max, est utilisé dans le cadre d'une expression :

```
console.log(Math.min(2, 4) + 100) ;
```

Conditions

```
let num = 5 ;
if (num < 10) {
     console.log("Small") ;
}
else if (num < 100) {
     console.log("Medium"); }
else {
     console.log("Large");
}
```

Boucles

```
let number = 0;
while (number <= 12) {
     console.log(number);
     number = number + 2; // number += 2
}
```

for (let number = 0; number <= 12; number = number + 2)
{ console.log(number); }

var tableau = [a, b , c….]
pour parcourir

avec le for
for (let number = 0; number <= 12; number = number++) {
console.log(tableau[number])
}

avec le foreach

tableau.forEach(***function(elt) { console.log(elt) }*** **)**

pour chaque élément du tableau il faudra executer cette fonction

function(data) { console.log(data) }

Switch à la place de plusieurs if

```
if (x == "value1") action1();
else if (x == "value2") action2();
else if (x == "value3") action3();
else defaultAction();
```

Relou…

```
switch (prompt("What is the weather like?")) {
case "rainy":
     console.log("Remember to bring an umbrella."); break;
case "sunny":
     console.log("Dress lightly."); break ;
case "cloudy":
     console.log("Go outside."); break;
default:
     console.log("Unknown weather type!"); break;
}
```

exercice.

modifier ces instructions en utilisant **switch**

```
let num = 5 ;

if (num < 10) {
     console.log("Small") ;
}
else if (num < 100) {
     console.log("Medium"); }
else {
     console.log("Large");
}
```

Fonctions (base)

```
const moncarre = function(x)
          { return x * x; }


console.log(moncarre(12));
```

autre exemple

```
const power = function(base, exponent) {
     let result = 1;
     for (let count = 0; count < exponent ; count++) {
     result *= base;
}

return result;
};

console.log(power(2, 10));
```

Le nombre d'arguments d'une fonction est optionnel

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

console.log(carre(4, true, "marmotte"));



console.log(power(4));
console.log(power(2, 6));
```

Closure.

```
function multiplier(factor)
          { return function(number)
                    { return number * factor ;  }
}



var twice = multiplier(2);


twice est une référence sur
          function(number) { return number * 2 } // twice est une
fonction

console.log( twice(5)); //-> 10

var tri = multiplier(3)
tri(2) // -> 6
```

Tableaux

Pour travailler avec un morceau de données numériques, nous devons d'abord trouver un moyen de le représenter dans la mémoire de notre machine. Disons, par exemple, que nous voulons représenter une collection des nombres 2, 3, 5, 7 et 11.

JavaScript fournit un type de données spécifique pour le stockage des séquences de valeurs. Il est appelé tableau et s'écrit sous la forme d'une liste de valeurs entre crochets, séparées par des virgules.

```
let listOfNumbers = [2, 3, 5, 7, 11];

console.log(listOfNumbers[2]);
console.log(listOfNumbers[0]);

let sequence = [1, 2, 3];
sequence.push(4);
sequence.push(5);
console.log(sequence);
```

Exercice

let listOfNumbers = [2, 3, 5, 7,4];
parcourir ce tableau (forEach) et afficher chaque valeur en affichant pair ou impair pour cette valeur

résultat attendu

```
2: pair
3: impair
5: impair
7: impair
4: pair
```

utiliser l'opérateur modulo: % (faire une fonction qui rend la chaine "pair" ou "impair")

Objets

Les valeurs du type objet sont des collections arbitraires de propriétés. Une façon de créer un objet est d'utiliser des accolades comme expression.

```
let day1 = { squirrel: false, events: ["work", "touched tree", "pizza", "running"] };
console.log(day1.squirrel);
console.log(day1.events[1]);
```

Un tableau d'objets avec un tableau dedans :

```
let journal = [
                {
                events: ["work", "touched tree", "pizza", "running", "television"],
                squirrel: false
                },
                {
                events: ["work", "ice cream", "cauliflower", "lasagna", "touched tree",
"brushed teeth"],
                squirrel: false},

                {
                events: ["weekend", "cycling", "break", "peanuts", "beer"],
                squirrel: true
                }
]
```

Quelques fonctions sur les chaines

```
console.log("coconuts".slice(4, 7));

console.log("coconut".indexOf("u"));

console.log("one two three".indexOf("ee"));

let sentence = "Secretary birds specialize in stomping";

let words = sentence.split(" ");

// words devient un tableau avec un mot par indice.
// words[0] -> "Secretary"
// words[1] -> "birds"
// words[2] -> "specialize"
// words[3] -> "in"
// words[4] -> "stomping"

console.log(words);

console.log(words.join(" "));

console.log("LA".repeat(3));

console.log(words.length) ;

console.log(sentence.length) ;

console.log(sentence.replace("birds","dogs")); // replaceAll

console.log(sentence.toUpperCase())

console.log(sentence.toLowerCase())

let txt1 = "Hello"
let txt2 = "World"
let txt3 = txt1.concat(" ",txt2) // txt3 = txt1 + " " + txt2
console.log(txt3)

let txt4 = "   Hello   ";
console.log(txt4.trim())  // enlève  les espaces des 2 côtés
// trimStart() au debut
// trimEnd() à la fin
```

JSON

Les objets et les tableaux sont stockés dans la mémoire de l'ordinateur sous forme de séquences de bits contenant les adresses - la place en mémoire - de leur contenu. Ainsi, un tableau avec un autre tableau à l'intérieur consiste en (au moins) une zone de mémoire pour le tableau intérieur, et une autre pour le tableau extérieur, contenant (entre autres) un nombre binaire qui représente la position du tableau intérieur.

Si vous souhaitez enregistrer des données dans un fichier pour plus tard ou les envoyer à un autre ordinateur via le réseau, vous devez d'une manière ou d'une autre convertir ces enchevêtrements d'adresses mémoire en une description qui peut être stockée ou envoyée.

Ce que nous pouvons faire, c'est **sérialiser** les données. Cela signifie qu'elles sont converties en une description lisible. Un format de **sérialisation** populaire est appelé JSON (prononcé "Jason"), qui signifie **JavaScript Object Notation**. Il est largement utilisé comme format de stockage et de communication de données sur le Web, même dans des langages autres que JavaScript.

```
{

        "squirrel": false,
        "events": ["work", "touched tree", "pizza", "running"]

}
```