LES VARIABLES

Déclaration d'une variable

```
var nameVar = valeur
```

sert à stocker des données

let => variable à donnée modifiable

const => variable à donnée non modifiable

Les valeurs que peut contenir une variable	
'STRING'	Chaine de caractère
Number	Integer et float
Array	Tableau
Boolean	True ou False
BigInt	Big integer (grands chiffres ex: 25n***2n
Object	{contient des motsClés: _____ , };
Function	([param] , [param]) => { instructions }

LES STRINGS
‘chaine de caractère’

Comment écrire une string:

```
Entre apostrophe = 'string';
Entre guillemet = "string";
```

Syntaxe d’une string:

```
var name = "Julie";
var age = 28;
//Concatenation Classique
console.log("My name is " + name + " and I'm " + age);
//Template literal
console.log(`My name is ${name} and I'm ${age}`);
```

LES METHODES POUR ‘STRING’	
test.length	afficher la longueur de la string)
test.replace("mot1", "mot2")	(mot2 remplace mot1)
test.charAt(0)	accéder à un caractère dans une string
test.toUpperCase();	Convertir en majuscule
test.charAt(0).toLowerCase();	accès caractère + mettre en miniscule
test.split("délimiteur");	découpe une string et met en forme de tableau
test.indexOf("text");	afficher l’index qui précède le caractère demandé
test.indexOf("Z");	donnera -1 si caractère inexistant dans la string
test.substring(j, n);	donnera jklmn
test.substring(a);	donnera à partir de a jusqu’à la fin de la chaine

LES NOMBRES
Numbers

```
var integer = 10;
var float = 2.49;
```

`x += y` : raccourci pour `x = x + y`

`x -= y` : raccourci pour `x = x - y`

`x *= y` : raccourci pour `x = x * y`

`x /= y` : raccourci pour `x = x / y`

`x **= y` : raccourci pour `x = x ** y`

`x %= y` : raccourci pour `x = x % y`

Opérateur arithmétique

`+` : addition
`-` : soustraction
`*` : multiplication
`/` : division
`**`: puissance
`%` : reste (modulo)

```
var min = 1;
var max = 10;
Math.floor(Math.random() * (max - min + 1) + min)
Math.floor(Math.random() * (100 - 20 + 1) + 20);
```

Les méthodes pour les nombres:

```
let n = 215;
n = n.toString(); // convertir Number to String
```

Objet Math :

```
Math.min(12, 36, 240); // donnera 12
Math.max(12, 36, 240); // donnera 240
Math.floor(3.14); //donnera 3 (arrondi au plus bas)
Math.ceil(15.12); //donnera 16(arrondi au plus haut)
Math.round(2.5); // donnera 3 (arrondi au plus proche)
Math.random(); // donnera un nbr aléatoire entre 0 & 1
Math.sqrt(9); // donnera 3 (√racine carré)
```

LES CONDITIONS
if / else

CONDITION SIMPLE

```
if (condition === true) {
Instruction Console.log('exécution')
} else {
console.log('exécution par défaut)}
```

CONDITION TERNAIRE

```
var nombre=200;
var taille = nombre > 100 ? 'Très grand' : nombre > 10 ?
'Grand' : 'Petit';
```

CONDITIONS IMBRIQUEES

```
if (condition1)
  instruction1
else if (condition2)
  instruction2
else if (condition3)
  instruction3
...
else
  instruction
```

EXEMPLE CONDITION SIMPLE

```
if (isRaining === true) {
  console.log("Oh not again !");
        takeUmbrellaAndGo();
} else {
  console.log("Yesssss! :D ");
        takeSunglassesAndGo();
}
```

CONDITIONS SWITCH

```
var season = "Summer";

switch (season) {
        case "Spring":
                console.log("Flowers everywhere !");
                break;
        case "Summer":
                console.log("Let's go to the beach !");
                break;
        case "Autumn":
                console.log("Back to school !");
                break;
        case "Winter":
                console.log("Let it snow !");
                break;
        default:
                console.log("That's not a season...");
}
```

Ici, pour chaque scénario possible (annoncé avec le mot clé `case xx`, nous avons écrit le code à exécuter. On a même pris le soin d'écrire une option par défaut (grâce au mot clé `default`), au cas où rien ne corresponde !

On termine le code a exécuter pour chaque scénario prévu par le mot clé `break` pour que le switch ne regarde pas les conditions suivantes.

COMPARAISON

|| et/ou &&

OPERATEURS DE COMPARAISON

> : strictement supérieur

< : strictement inférieur

>= : supérieur ou égal

<= : inférieur ou égal

== : égal à

=== : strictement égal à

!= : différent de

!== : strictement différent de

&& = ET (les 2 conditions doivent être vraie)

```
var age = 15;
if (age > 6 && age < 11) {
console.log("you can play !");
} else { console.log("you can't play this game is only for kids between 6 and 11 years old!");
}
```

|| = OU (une des 2 conditions doit être vraie)

```
if (phoneBrand === "apple" || computerBrand === "apple") {
        console.log("you have at least one apple product !");
}
```

|| + && (combinaison)

```
var year = 2016;

if ((year % 4 === 0 && year % 100 > 0) || (year % 400 === 0)) {
        alert(year + " est bissextile");
} else {
        alert(year + " n'est pas bissextile");
}
```

LES TABLEAUX
[ARRAY]

```
var array = [“element1", “element2", “element3", “element3"];
```

LES METHODES => ARRAY	
array.length;	longueur tableau
array[index];	choisir et afficher un élément dans un tableau
array.indexOf(‘element’);	donne index de l’élément dans un tableau
array.push(‘element’);	ajoute élément à la fin du tableau
array.unshift(‘element’);	ajoute élément au début du tableau
array.pop();	supprime le dernier élément du tableau
array.shift();	supprime le premier élément du tableau
array.reverse();	inverser l’ordre des éléments dans le tableau
array.sort();	met dans l’ordre alphabétique croissant
array.sort((a, b) => a - b);	met les nombres dans l’ordre croissant
array.sort((a, b) => b - a);	met les nombres dans l’ordre décroissant
array.slice(a,b)	extraire une partie des éléments dans un tableau
array.slice(a)	extraire à partir de a jusqu’à la fin du tableau
Array.join(" ")	convertir tableau en String (" espace ")

LES BOUCLES

while for

For = soit combien de fois ce code devra être exécuté.
While = soit quelle est la condition pour que la boucle tourne.

```
while (condition) {
// … instruction à exécuter;
}
```

condition : tant qu'elle est vraie, la condition continue de tourner

```
for (start; condition; incrémentation) {
... // ce bloc s'exécute à chaque étape
}
```

Exemple:

```
var goal = 20;

for (var i = 0; i <= goal; i++) {
  console.log(i) // retournera tous les nombres de 0 à 20 !
}
```

Etape de la boucle: comment i évolue?

1) i = start, point de départ de la boucle
2) i <= goal = condition à vérifier à chaque itération
3) {instruction à exécuter}
4) i++ = incrémentation ou **i--** = décrémentation

LES BOUCLES

break; continue;

Le mot clé break sert à arrêter une boucle, tandis que le mot clé continue sert à passer à l'étape suivante sans exécuter tout le bloc de code de la boucle :

```
var menu = ["salad", "burger", "cheese",
"dessert", "coffee"];

// je créer une boucle pour manger, mais je
n'aime pas le fromage et je ne bois pas de
café... Comment faire ?

for (var i = 0; i < menu.length; i++) {
      if (menu[i] === "coffee") {
            break; // arrête avant le
console.log de "coffee"
      }

      if (menu[i] === "cheese") {
            continue; // saute le console.log
de "cheese"
      }

      console.log(menu[i])
}

// retournera :
// "salad"
// "burger"
// "dessert"
```

LES BOUCLES
Récursivité

La récursivité est une autre manière de répéter des étapes jusqu'à ce qu'une condition soit vérifiée. Une fonction récursive ressemblera beaucoup à une boucle, à un détail près : elle va s'appeler elle-même !

```
function countDown(number) {
	if (number === 0) {
	console.log(number + " finished !");
	return; //sert à stopper la condition
	}

	console.log(number);
	countDown(number - 1);
}

countDown(5);

// retournera :
// 5
// 4
// 3
// 2
// 1
// 0 finished !
```

LES OBJETS
Object = {};

Un objet est une variable dans laquelle on peut stocker plusieurs valeurs, associées à des clés :

```
var user = {
        name: "Jane",
        genre: "female",
        age: 25,
        isStudent: true,
        favorites: ["coffee", "nutella", "pizza"]
};
console.log(user.name); <<< afficher la valeur stocker dans name
```

Les méthodes pour OBJECT	
Object.assign(a, b)	ajoute le contenu d'un objet **b** dans un objet **a**. S'il y a la même propriété dans les deux objets, celle de l'objet de base sera remplacée
NomObject.hasOwnProperty(lol)	Ai-je le mot clé « lol » dans mon objet?
Object.keys(A)	renvoie un tableau contenant toutes les clés d'un objet A
Object.values(a)	renvoie un tableau contenant toutes les valeurs d'un objet **a**
Object.entries(a)	renvoie un tableau contenant des tableaux pour chaque paire "clé / valeur" d'un objet **a**

LES FONCTIONS

function functName(param) = {};

Les fonctions servent à effectuer des actions. On peut grouper du code à l'intérieur de fonctions pour découper la logique en morceaux identifiables et cohérents. Pour utiliser une fonction, il faut d'abord la déclarer, puis l'appeler à l'extérieur.

Function a son propre Scope càd a ses propres variables créés et serviront que à l'intérieur function.
PS: variables à l'extérieurs de la function pourront être appelées à l'intérieur de celle-ci

Console.log() et fonctions

Pour afficher le résultat d'une fonction :

Sans return; = un `console.log()` à **l'intérieur sert à afficher uniquement sans pouvoir utiliser le résultat.**

Avec return; = **Appeler la fonction** dans un `console.log()` pour afficher le résultat obtenu par la fonction : votre fonction est intacte ! Vous pourrez l'utiliser pour transmettre des données ailleurs dans le code.