

Javascript

partie 1

1) Instructions importantes :

- `alert("Hello world");` - déclanche une popup
- `let reponse = prompt("Comment tu t'appelle ?");` - stock la réponse dans la variable « reponse »
- `console.log(reponse);` - affiche dans la console la valeur de « reponse »

```
► Object { brand: "Alfa Roméo", buyDate: Date Sat Nov 20 2010 01:00:00 GMT+0100 (heure normale d'Europe centrale), passengers: (2) [...], year: 2012 }
```

- `console.table(myArray);` - affiche les valeurs d'une variable du type array en format tableau dans la console

console.table()		debugger eval code:13:9	
(index)	0	1	Valeurs
brand			Alfa Roméo
buyDate			
passengers	Nicolas	Charlotte	
year			2012

2) Variables :

Lexique

```
var prenom;  
var clientEmail;  
var dateActuelle;  
var bar;  
var c;  
var 22etudiant; ✗  
var etudiant22; ✓
```

lettres
chiffres
—
\$

- **var toto** ; - façon générique de déclarer une variable (à éviter)
- **let toto** ; - avec le let, nous pouvons d'abord le déclarer et ensuite assigner une valeur
- **const toto = "Toto"** - Avec le const, nous devons le déclarer et l'assigner une valeur immédiatement

```
// Ne plus utiliser le var  
// Toujours utiliser le const  
// Cependant, si on a besoin d'assigner une nouvelle valeur à la  
variable, on utilise le let
```

Types de variables

Primitifs :

- **Numbers** – numéro entier ou flottant (avec les décimaux)

```
console.log(5);  
console.log(2.5);
```

- **Booleans** -

L'objet Boolean est un objet permettant de représenter une valeur booléenne (true ou false)

```
new Boolean([valeur])
```

- **String** - chaînes de caractères

```
console.log("Hello world");  
console.log('Hello world');  
console.log("12"); //différence de 12, égal à 1 suivi de 2
```

Concaténation de strings

```
var MomPrenom = "Vanessa";  
var MomNomdeFamille = "Sant'André";  
var MonNomComplet = MomPrenom + " " + MomNomdeFamille  
console.log(MonNomComplet);
```

Conversion de Strings en Numbers

```
let result = parseInt(nomString) - entiers  
let result = parseFloat(nomString) - flottants
```

Opérations sur les strings

```
var maChaine = "C'est ma chaine, laissez-la tranquille !";
```

```
maChaine.length;
```

La propriété length représente la longueur d'une chaîne de caractères. Donc le résultat ici est: 40 (les espaces blancs sont comptés).

```
maChaine.toUpperCase();
```

```
maChaine.toLowerCase();
```

Transforme le string en majuscule ou minuscule

```
maChaine.indexOf("laissez-la");
```

"indexOf" renvoie l'indice de la première occurrence de la valeur cherchée. Elle renvoie -1 si la valeur cherchée n'est pas trouvée. Donc le résultat ici est: 17

```
let phrase = "La méthode slice() extrait une section de la chaîne de caractères."
```

```
phrase.slice(3, 10);
```

La méthode slice() extrait une section du string et renvoie un nouveau string. Par exemple, phrase.slice(3, 10) extrait le quatrième caractère jusqu'au dixième caractère. Donc le résultat ici est le string: "méthode"

```
phrase.substr(3, 7);
```

La méthode substr() renvoie les caractères d'un string commençant à un endroit donné et sur une longueur donnée. Autrement dit, cette méthode renvoie la sous-chaîne du string à partir d'un indice. Donc le résultat ici est le string: "méthode"

- **Undefined, Null**

`console.log(undefined);` - pas de valeur, pas encore de valeur assignée

`// exemple : var toto; retourne undefined`

`console.log(null);` - absence de valeur

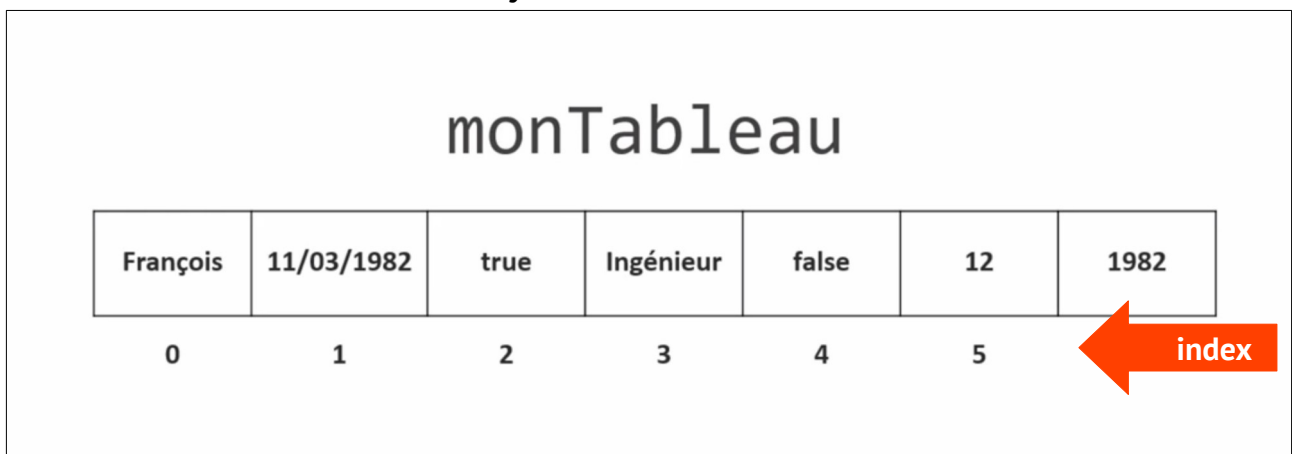
`// exemple : var toto = 0`

Par référence :

- **Array**

L'objet Array est utilisé pour créer des tableaux. Les éléments d'un tableau sont numérotés (index ou indice) à partir de zéro

Array à une dimension :



```
let monTableau = [12, 25, "François", "Frémont", true, false]
let monTableau = new Array(12, 25, "François", "Frémont", true, false)
```

```
let monTableau = Array(12,
    "François",
    "Frémont",
    ["a", "b", "c"],
    true,
    false)
```

Array dans une array

- Méthode pour accéder à un élément spécifique d'un tableau

```
let monTableau = ["element0", "element1", "element2"]
console.log(monTableau[2]); // sort l'élément element2
console.log(monTableau[4][0]); // sort l'élément1 dans l'array à
l'intérieur de l'array (index 4)
```

- Méthode pour modifier un élément spécifique d'un tableau

```
let monTableau = [12, 25, "François", "Frémont", true, false]
monTableau[2] = "Vanessa" // remplace "François" par "Vanessa" (index
2)
console.table(monTableau);
```

console.table()

debugger eval code:3:9

(index)	Valeurs
0	12
1	25
2	Vanessa
3	Frémont
4	true
5	false

- Autres méthodes sur les tableaux :

```
let mesClients = ["Robert", "Alice", "Francis", "Elina", "Alpha"];
```

```
mesClients.push("Sara");  
mesClients.push("François", "Frémont");
```

Cette méthode ajoute un ou plusieurs éléments à la fin d'un tableau

```
var retire = mesClients.pop();
```

Cette méthode supprime le dernier élément d'un tableau et retourne cet élément.

L'opérateur delete permet de supprimer une valeur donnée d'un tableau.


```
delete mesClients[3];
```

Lorsqu'on supprime cet valeur du tableau, la longueur du tableau n'est pas modifiée. Cela vaut également lorsqu'on supprime la dernière valeur du tableau avec .pop().

```
console.table()
```

```
debugger eval code:3:9
```

(index)	Valeurs
0	Robert
1	Alice
2	Francis
3	
4	Alpha



```
let retire = mesClients.splice(3, 1);  
console.table(mesClients);  
console.log(retire);
```

La méthode splice est utilisé souvent pour retirer une ou plusieurs valeurs d'un tableau à partir d'une position donnée. Splice retourne un tableau contenant les valeurs supprimés. Si une seule valeur est supprimée, cette est retournée (ici retire = « Alpha »).

console.table()

debugger eval code:2:9

(index)	Valeurs
0	Robert
1	Alice
2	Francis

► Array ["Alpha"]

debugger eval code:3:9

```
mesClients.sort();  
console.table(mesClients);
```

Cette méthode trie en place les éléments d'un tableau par ordre alphabétique et retourne le tableau.

console.table()

debugger eval code:2:9

(index)	Valeurs
0	Alice
1	Francis
2	Robert

```
mesClients.reverse();  
console.table(mesClients);
```

Cette méthode renverse l'ordre des éléments d'un tableau par ordre alphabétique

console.table()

debugger eval code:2:9

(index)	Valeurs
0	Robert
1	Francis
2	Alice

```
let maNouvelleChaine = "Robert, Alice, Francis, Elina, Alpha";  
let monNouveauTableau = maNouvelleChaine.split(", ");  
console.table(monNouveauTableau);
```


La méthode `split()` permet de diviser une chaîne de caractères à partir d'un séparateur pour fournir un tableau de sous-chaînes. Le séparateur définit le ou les caractères à utiliser pour scinder la chaîne.

Avant :

```
>> var maNouvelleChaine = "Robert, Alice, Francis, Elina, Alpha";
```

Après :

```
► Array(5) [ "Robert", "Alice", "Francis", "Elina", "Alpha" ]
```

Si le séparateur est une chaîne vide, la chaîne `maNouvelleChaine` sera convertie en un tableau de caractères.

```
var maNouvelleChaine = "Robert, Alice, Francis, Elina, Alpha";  
maNouvelleChaine.split("");
```

```
► Array(36) [ "R", "o", "b", "e", "r", "t", ",", " ", "A", "l", "i", "c", "e", " ", "F", "r", "a", "n", "c", "i", "s", " ", "E", "l", "i", "n", "a", " ", "A", "l", "p", "h", "a" ]
```

```
resultat = mesClients.includes("Robert");  
console.log(resultat);
```

```
resultat = mesClients.includes("Jasmin");  
console.log(resultat);
```

Cette méthode détermine si le tableau contient ou non un certain élément. Elle renvoie `true` ou `false` selon le cas de figure.

```
>> resultat = mesClients.includes("Robert");  
console.log(resultat);
```

```
resultat = mesClients.includes("Jasmin");  
console.log(resultat);
```

```
true
```

```
false
```

Array à deux dimensions :

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

Pour construire des tableaux à deux dimensions en javascript, l'astuce consiste à stocker une liste de tableaux dans un autre tableau.

```
let notesEtudiants = [  
  [65, 70, 80, 90],  
  [100, 95, 100, 70],  
  [60, 100, 90, 70]  
];
```

65	[0][0]	70	[0][1]	80	[0][2]	90	[0][3]
100	[1][0]	95	[1][1]	100	[1][2]	70	[1][3]
60	[2][0]	10	[2][1]	90	[2][2]	70	[2][3]

```
console.log(notesEtudiants[0][1]);
```

70

```
console.log(notesEtudiants[1]);
```

► **Array(4)** [100, 95, 100, 70]

```
console.log(notesEtudiants[2][0]);
```

60

Array.length

Retourne le nombre d'éléments d'un tableau

Exemple :

```
let fruits=Array("Fraise", "Pomme", "Poire", "Abricot");  
document.write("Il y a "+fruits.length+" éléments dans le tableau  
fruits");
```

Résultat : Il y a 4 éléments dans le tableau fruits

Autres méthodes :

- **concat()** Retourne la concaténation de 2 tableaux
- **flat()** Retourne un tableau multidimensionnel aplati en un tableau simple
- **forEach()** Parcourt un à un les éléments du tableau pour y exécuter une fonction
- **from()** Construit un tableau à partir d'un autre objet
- **includes()** Indique si l'élément passé en paramètre appartient bien au tableau
- **indexOf()** Retourne l'index du tableau contenant l'élément recherché
- **isArray()** Retourne true si l'objet passé en paramètre est un tableau
- **join()** Retourne la chaîne composée de tous les éléments du tableau séparés par un séparateur
- **reduce()** Réduit un tableau grâce à une fonction accumulatrice
- **shift()** Retourne et supprime le premier élément du tableau
- **unshift()** Insère des éléments en début de tableau

- **Objets**

Différence entre Array et Objet :

Un Array est un Object, cependant :

- Dans un Objet les propriétés sont nommées par des clés
- Dans un Array nous avons que les valeurs / nous pouvons accéder à sa taille par la méthode length

Array

```
let myArray = ["William Fontaine", 32, false]
```

Objet

```
let personne = {  
  nom_complet: "William Fontaine",  
  age: 32,  
  administrateur: false  
};
```

Les objets permettent de stocker des données par clé/valeur

Objet littéral

```
let personne = {  
  nom_complet: "William Fontaine",  
  age: 32,  
  administrateur: false  
};
```

Par constructeur

La méthode constructor est une méthode qui est utilisée pour créer et initialiser un objet lorsqu'on utilise le mot clé **class**. Une classe est un modèle pour un objet. Elle permet de construire plusieurs objets du même type (appelés **instances**) plus facilement, rapidement et en toute fiabilité (comme une moule :-)). Regarde les exemples ci-dessous :

1) On crée une class et on la nomme (*N'oublier pas de mettre en majuscule le nom des classes pour les faire distinguer des fonctions.*) :

```
class Book {  
  
}
```

Pour cette classe, nous souhaitons que chaque **Book** ait un titre, un auteur et un nombre de pages. On définit la class Pour cela, on utilise ce qu'on appelle un **constructor**.

2) La class :

2.1

```
class Book {  
    constructor(title, author, pages) {  
        }  
}
```

title, author, pages sont les clés auxquelles nous souhaitons remplir ensuite avec des valeurs

2.2

Pour attribuer les clés titre, l'auteur et le nombre de pages qui seront ensuite remplis par l'instance, utilise le mot clé `this` et la notation dot.

```
class Book {  
    constructor(title, author, pages) {  
        this.title = title;  
        this.author = author;  
        this.pages = pages;  
    }  
}
```

Et ensuite on crée l'instance qui se basera dans cette class

Le **constructor** d'une classe est la fonction qui est appelée quand on crée une nouvelle instance en utilisant le mot clé **new**.

Création de l'instance et appel des clés de la class en utilisant **new** :

```
let myBook = new Book ( "L'Histoire de Tao" , "Will Alexander" ,  
250 );
```

A partir de l'instance, nous remplissons les clés préalablement créées avec les valeurs "L'Histoire de Tao", "Will Alexander", 250

```
{  
    title: "L'Histoire de Tao",  
    author: "Will Alexander",  
    pages: 250  
}
```

```
console.table(myBook) ;
```

console.table() debugger eval code:2:9	
(index)	Valeurs
title	L'Histoire de Tao
author	Will Alexander
pages	250

Au autre exemple :

```
class Episode {  
  constructor(title, duration, hasBeenWatched) {  
    this.title = title;  
    this.duration = duration;  
    this.hasBeenWatched = hasBeenWatched;  
  }  
}  
  
let firstEpisode = new Episode('Dark Beginnings', 45, true);  
let secondEpisode = new Episode('The Mystery Continues', 45, false);  
let thirdEpisode = new Episode('The Unexpected Climax', 60, false);  
  
console.table(firstEpisode) ;  
console.table(secondEpisode) ;  
console.table(thirdEpisode) ;
```

console.table() debugger eval code:13:9	
(index)	Valeurs
title	Dark Beginnings
duration	45
hasBeenWatched	true

console.table() debugger eval code:14:9	
(index)	Valeurs
title	The Mystery Continues
duration	45
hasBeenWatched	false

console.table() debugger eval code:15:9	
(index)	Valeurs
title	The Unexpected Climax
duration	60
hasBeenWatched	false