Les objets et les tableaux en JavaScript

Les objets en JavaScript

Qu'est ce que un objet ?

Un objet est un ensemble de données et/ou de fonctionnalités liées entre elles

Objet = Propriétés + Méthodes

voiture

Propriétés	Méthodes
marque : "Ferrari", poids : "850kg", Couleur : "noir"	start : function(){...}, conduire : function(){...}, freiner : function(){...}

user

nom : "AMCF", password : "test"	connexion : function(){...}, publier : function(){...}

- ✓ Chaque objet contient des propriétés
- ✓ Chaque propriété a un nom et une valeur
- ✓ Les propriétés sont séparées par des virgules (,)
- ✓ Le pair nom et valeur est séparé par deux points (:)

Les propriétés d'un objet

Syntaxes pour créer un objet vide :

```
<script>
    let nomObjet = new Object(); //constructeur d'objet
    let nomObjet2 = {}; //objet littéral
</script>
```

Nous pouvons ajouter, supprimer et accéder des propriétés d'un objet;

AJOUTER

```
<script>
    let nomObjet = {
        nom: 'AMCF',
        age: 26
    };
</script>
<script>
    let nomObjet = new Object();
    nomObjet.nom = 'Yahya';
    nomObjet.age = 26;
</script>
```

ACCÉDER

```
nomObjet.nom = 'nouvelle valeur';
alert(nomObjet.nom);

nomObjet['nom'] = 'nouvelle valeur';
alert(nomObjet['nom']);
```

SUPPRIMER

```
delete nomObjet.age;

delete nomObjet['age'];
```


Les méthodes d'un objet

```
var voiture = {
    marque: "Ferrari",
    poids: "850kg",
    Couleur: "noir",
    start: function() {
        alert("La voiture " + this.marque + " est démaré.");
    },
    conduire: function() {
        alert("La fonction conduire est appelée");
    },
    freiner: function() {
        alert("Stop !");
    }
}
```

- ✓ Un objet peut contenir également certain fonction (des méthodes).
- ✓ Il existe deux façons pour accéder aux instances d'un objet (propriétés et méthodes)
  1. La notation du point **.** : (Ex. voiture.marque)
  2. La notation des crochets **[]:** (Ex. voiture["marque"])
- ✓ Le mot clé « this » se réfère à l'objet courant (ex. this.marque équivqlent à voiture.marque).

Les méthodes d'un objet

```
<script>
    const  user = {
        nom: "AMCF",
        profession: "dev",
        password: 'test',
        presentation: function() {
            return this.nom + " est un " + this.profession;
        }
    };
</script>
```

- Pour accéder aux méthodes d'un objet

nomObjet.nomMethode()

- Exemple:

user.presentation()

Une méthode est une fonction stockée en tant que propriété.

Le mot-clé this est l' objet courant

```
<script>
    const  mohammed = {
        nom: "mohammed",
        age: 26,
        incrementerAge: function(inc) {
            this.age += inc;
        },
        estVieux: function() {
            return (this.age > 30);
        }
    };
    mohammed.incrementerAge(4);
    console.log(mohammed.estVieux());
</script>
```


Amina et Yahya essayent de comparer leur IMC

(Indicateur de Masse Corporelle), qui se calcule à l'aide de la formule suivante :

$$IMC = \frac{masse}{taille^2}$$ (masse en kg et taille en mètre)

Réaliser un script JS dont vous :

1. Stockez la masse et la taille de « Amina » et de « Yahya » dans des variables.
2. Calculez leurs IMC respectifs à l'aide de la formule
3. Créez une variable booléenne "aminaPlusIMC" contenant l'information si « Amina » a un IMC plus élevé que « Yahya ».

Testez votre script en fournissant des données de votre choix

Les chaînes de caractères en JavaScript

- En JS, les chaînes de caractères sont des morceaux de texte, c'est-à-dire un ensemble de caractères entre guillemets.
- Puisque tout dans JS est constitué d'objets, lorsque vous créez une chaîne, celle-ci devient une instance d'objet String.

Les méthodes des chaînes de caractères

- `str.length` : Pour obtenir la longueur d'une chaîne
- `str[index]` : récupération d'un caractère dans une chaîne.
- `str.indexOf("sousChaine")` et `str.search("sousChaine")` : Pour trouver une sous-chaîne dans une chaîne
- `slice(start,end)` - `substring(start,end)` et `substr(start,length)` : extraction d'une chaîne de caractères
- `str.replace("exist","nouveau")` : replacer le premier occurance du mot exist dans str par nouveau. Pour remplacer tous les occurance d'exist dans str : `str.replace(/exist/g,"nouveau")`
- `str.trim()` : suppression de l'espace blanc supplémentaire aux deux extrémités d'une chaîne de caractères
- `str.concat("separator", "chaîne")` : concaténer str separator et chaîne.

Les nombres en JavaScript

Les méthodes des nombres

L'objet **Number** fournit des méthodes qui nous permettent de travailler avec des valeurs numériques

- `Number.isInteger()` : retourne vrai si l'argument passé (valeur) est un entier
- `isNaN()` : Renvoie vrai si l'argument passé (valeur) n'est pas un nombre légal
- `parseInt()` : Prend un argument de type chaîne de caractères (valeur) et renvoie un entier selon la valeur donnée
- `parseFloat()` : Prend un argument de type chaîne de caractères (valeur) et renvoie un nombre réel
- `toFixed()` : Formater un nombre en utilisant la notation du point fixe. Il renvoie une chaîne avec un nombre spécifié de décimales.
- `toString()` : Renvoie une chaîne de caractères qui représente le nombre donné

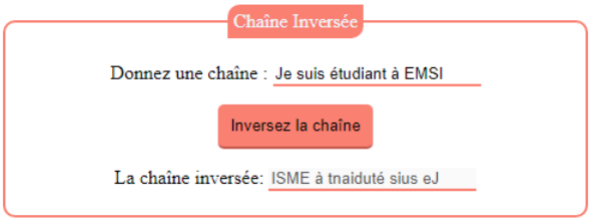
Les nombres en JavaScript

```
<script>
    console.log("=============== La méthode Number.isInteger() ===============");
    console.log("Number.isInteger(26) : " + Number.isInteger(26));
    console.log("Number.isInteger(5.6) : " + Number.isInteger(5.6));
    console.log("Number.isInteger('AMCF') : " + Number.isInteger('AMCF'));
    console.log("=================== La méthode isNaN() ======================");
    console.log("isNaN(35) : " + isNaN(35));
    console.log("isNaN(3.14) : " + isNaN(3.14));
    console.log("isNaN('Dev Info') : " + isNaN('Dev Info'));
    console.log("isNaN(2 / 'test') : " + isNaN(2 / 'test'));
    console.log("=================== La méthode parseInt() ===================");
    console.log("parseInt('26')" + parseInt('26'));
    console.log("parseInt('.25')" + parseInt('.25'));
    console.log("parseInt('10 octobre')" + parseInt('10 octobre'));
    console.log("parseInt('Octobre 10')" + parseInt('Octobre 10'));
    console.log("parseInt('que du text')" + parseInt('que du text'));
    console.log("================ La méthode parseFloat() ================");
    console.log("parseFloat('.25') : " + parseFloat('.25'));
    console.log("parseFloat('6.75 KG') : " + parseFloat('6.75 KG'));
    console.log("parseFloat('Poids 6.75 KG') : " + parseFloat('Poids 6.75 KG'));
    console.log("parseFloat('Que du texte') : " + parseFloat('Que du texte'));
    console.log("================ La méthode toFixed() ================");
    console.log("(3.14159265).toFixed(1) : " + (3.14159265).toFixed(1));
    console.log("(3.14159265).toFixed(2) : " + (3.14159265).toFixed(2));
    console.log("(3.14159265).toFixed(3) : " + (3.14159265).toFixed(3));
    console.log("================ La méthode toString() ================");
    console.log("(15).toString() : " + (15).toString());
    console.log("(2.5).toString() : " + (2.5).toString());
    console.log("(2022).toString() : " + (2022).toString());
</script>
```

```
=============== La méthode Number.isInteger() ===============
Number.isInteger(26) : true
Number.isInteger(5.6) : false
Number.isInteger('AMCF') : false
=================== La méthode isNaN() ======================
isNaN(35) : false
isNaN(3.14) : false
isNaN('Dev Info') : true
isNaN(2 / 'test') : true
=================== La méthode parseInt() ===================
parseInt('26')26
parseInt('.25')NaN
parseInt('10 octobre')10
parseInt('Octobre 10')NaN
parseInt('que du text')NaN
================ La méthode parseFloat() ================
parseFloat('.25') : 0.25
parseFloat('6.75 KG') : 6.75
parseFloat('Le poids est 6.75 KG') : NaN
parseFloat('Que du texte') : NaN
================ La méthode toFixed() ================
(3.14159265).toFixed(1) : 3.1
(3.14159265).toFixed(2) : 3.14
(3.14159265).toFixed(3) : 3.142
================ La méthode toString() ================
(15).toString() : 15
(2.5).toString() : 2.5
(2022).toString() : 2022
```


Manipulation des nombres et des strings en JavaScript

1. Créez une fonction qui prend deux chaînes str1 et str2 comme arguments et renvoie TRUE si le nombre total de caractères dans la première chaîne est le même dans la deuxième chaîne sinon renvoie FALSE.
2. Dans un script JavaScript, Créer une fonction qui renvoie la chaîne de caractères écrite dans la première zone de texte inversée et afficher le résultat dans la deuxième zone de texte comme c'est illustrée dans la figure.



Les tableaux en JavaScript

Les tableaux JavaScript sont des objets de type liste. Un seul tableau peut stocker plusieurs données .

Syntaxe de Création d'un tableau

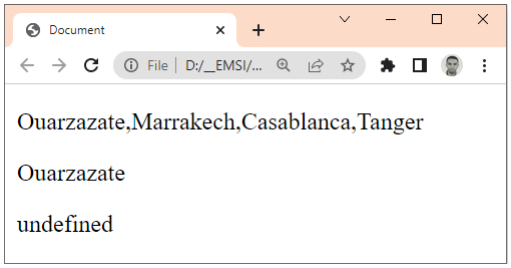
```
var nomTableau = [element1, element2, '...',  elementn];
var nomTableau2 = new Array(element1, element2, '...',  elementn);
```

il est recommandé d'utiliser des crochets, c'est beaucoup plus rapide en exécution et plus lisible.

accès à un élément du tableau

les éléments des tableaux sont accessibles à l'aide des entiers, les tableaux sont indexés de zéro (c'est-à-dire que le compteur commence à 0 et non à 1).

```
<p id="demo"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    document.getElementById("demo").innerHTML = villes;
    document.getElementById("demo2").innerHTML = villes[0];
    document.getElementById("demo3").innerHTML = villes[5];
</script>
```

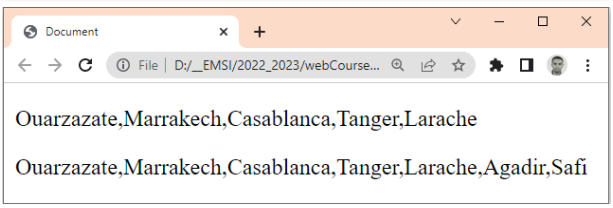


Les tableaux en JavaScript

Ajouter des éléments à un tableau

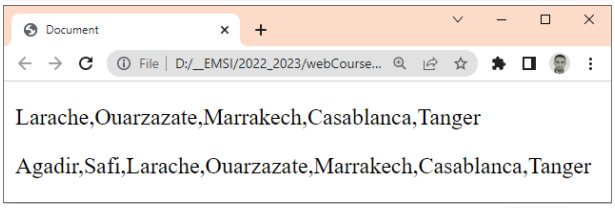
La méthode **push()** ajoute un ou plusieurs éléments à **la fin** d'un tableau.

```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    villes.push("Larache");
    document.getElementById("demo").innerHTML = villes;
    villes.push("Agadir", "Safi");
    document.getElementById("demo2").innerHTML = villes;
</script>
```



La méthode **unshift()** ajoute un ou plusieurs éléments au **début** d'un tableau.

```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    villes.unshift("Larache");
    document.getElementById("demo").innerHTML = villes;
    villes.unshift("Agadir", "Safi");
    document.getElementById("demo2").innerHTML = villes;
</script>
```



Supprimer des éléments dans un tableau

La méthode **pop()** supprime le dernier élément d'un tableau.

```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    document.getElementById("demo").innerHTML = villes;
    villes.pop();
    document.getElementById("demo2").innerHTML = villes;
</script>
```

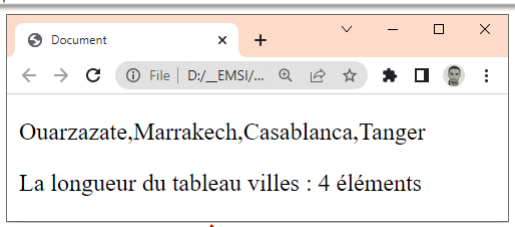
La méthode **shift()** supprime le premier élément d'un tableau.

```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    document.getElementById("demo").innerHTML = villes;
    villes.shift();
    document.getElementById("demo2").innerHTML = villes;
</script>
```

Obtenir la longueur d'un tableau

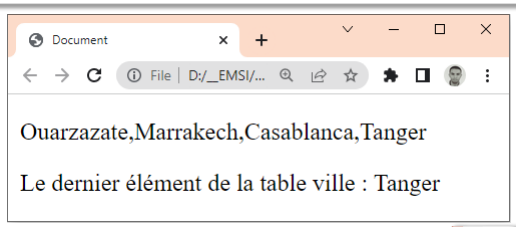
La longueur d'un tableau représente le nombre de ses éléments, pour cela utiliser la propriété **length.**

```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    document.getElementById("demo").innerHTML = villes;
    var longueur = villes.length;
    document.getElementById("demo2").innerHTML =
        "La longueur du tableau villes : " + longueur + " éléments";
</script>
```



Alors, pour accéder au dernier élément d'un tableau, il faut déduire 1 de sa longueur.

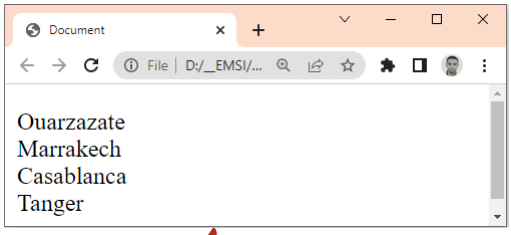
```
<p id="demo"></p>
<p id="demo2"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    document.getElementById("demo").innerHTML = villes;
    document.getElementById("demo2").innerHTML =
        "Le dernier élément de la table ville : " +
        villes[villes.length - 1];
</script>
```



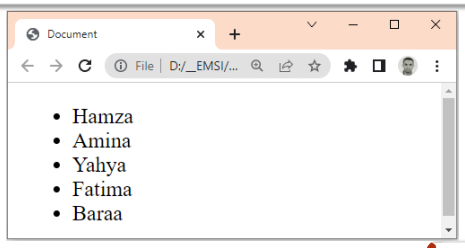
Boucle sur un tableau

Boucler sur un tableau signifie accéder à ses données individuellement un par un.

```
<p id="demo"></p>
<script>
    const villes = ["Ouarzazate", "Marrakech", "Casablanca", "Tanger"];
    for (var i = 0; i < villes.length; i++) {
        document.getElementById('demo').innerHTML +=
            villes[i] + "<br>";
    }
</script>
```

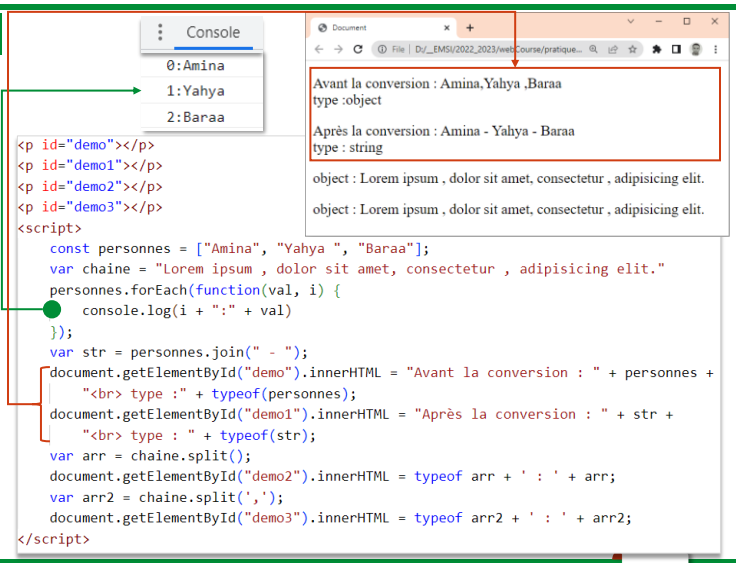



```
<dev id="demo"></dev>
<script>
    const personnes = ["Hamza", "Amina", "Yahya ", "Fatima", "Baraa"];
    list = '<ul>';
    for (var i = 0; i < personnes.length; i++) {
        list += '<li>' + personnes[i] + '</li>';
    }
    list += '</ul>'
    document.getElementById("demo").innerHTML =
        list;
</script>
```



Les méthodes d'un tableau

- `forEach()`: C'est un autre moyen de boucler sur un tableau, ou d'accéder aux données de chaque élément. Il permet d'exécuter une fonction de **callback** pour chaque élément du tableau.
- `join()`: Crée et renvoie une nouvelle chaîne de caractères en combinant tous les éléments d'un tableau; les éléments sont séparés par un caractère spécifié join("-"). Si aucun caractère n'est spécifié, les éléments sont séparés par des virgules.
- `split()`: Prend un séparateur comme argument ; et convertit une chaîne de caractères en un tableau



Les tableaux en JavaScript

Les méthodes d'un tableau

- `concat()`: fusionne deux ou plusieurs tableaux.
- `indexOf()` : recherche d'un élément, elle renvoie l'index de la première occurrence d'un élément dans le tableau.
- `lastIndexOf()` : elle renvoie l'index de la la dernière occurrence d'un élément dans un tableau.
- `slice()` : découpage d'un tableau, prend 2 arguments :
  1. l'indice où commence l'extraction
  2. l'indice où l'extraction devrait se terminer (il n'est pas inclus)
- `splice()` : modifie le contenu d'un tableau, prend 3 args.
  1. l'indice où le changement commence
  2. le numéro des éléments à enlever et à remplacer
  3. les éléments à ajouter au tableau

```
var t1 = ['elem1', 'elem2', 'elem3'];
var t2 = ['elem4', 'elem5', 'elem2'];
var t3 = ['elem7', 'elem2', 'elem5'];
var tab = t1.concat(t2);
var tab2 = t1.concat(t2, t3);
console.log(tab);
console.log(tab2);
console.log(tab2.indexOf("elem2"));
console.log(tab2.lastIndexOf("elem2"));
console.log(tab2.indexOf("ameksa"));
console.log(tab2.includes("ameksa"));
console.log(tab2.slice(0, 7));
tab2.splice(1, 0, "test", "test2");
console.log(tab2);
tab2.splice(1, 1, "val1", "val2");
console.log(tab2);
```

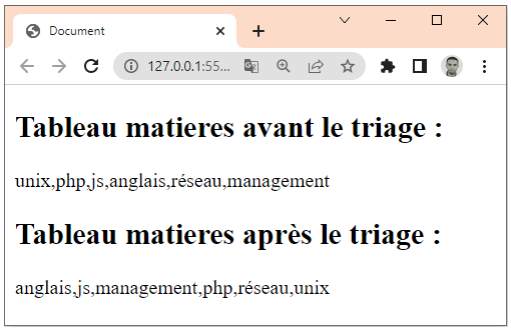
Les tableaux en JavaScript

Triage de tableaux

- **sort() :** trie les éléments d'un tableau, l'ordre de tri par défaut est ascendant.

```
<body>
    <h2>Tableau matieres avant le triage :</h2>
    <p id="demo"></p>
    <h2>Tableau matieres après le triage :</h2>
    <p id="demo2"></p>
    <script>
        var matieres = ['unix', 'php', 'js', 'anglais', 'réseau', 'management'];
        document.getElementById('demo').innerHTML = matieres;
        document.getElementById('demo2').innerHTML = matieres.sort();
    </script>

</body>
```



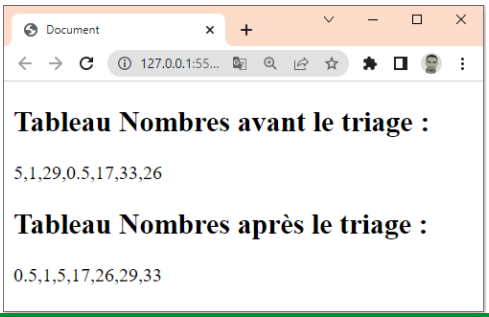
Triage de tableaux

- **Sort ()** peut prendre un argument, il s'agit d'une fonction de comparaison. Cette fonction définit l'ordre de tri.

fonctionDeComparaison(a, b)

- Si la function renvoie une **valeur < 0**, le **a** viens en **premier**.
- S'elle renvoie une **valeur = 0**, **a** et **b** restent **inchangés**.
- S'elle renvoie une **valeur > 0**, le **b** viens en **premier**.

```
<body>
    <h2>Tableau Nombres avant le triage :</h2>
    <p id="demo"></p>
    <h2>Tableau Nombres après le triage :</h2>
    <p id="demo2"></p>
    <script>
        var nombres = [5, 1, 29, 0.5, 17, 33, 26];
        document.getElementById('demo').innerHTML = nombres;
        nombres.sort(function(a, b) {
            return a - b;
        });
        document.getElementById('demo2').innerHTML = nombres;
    </script>
</body>
```



Triage d'un tableaux d'objet

- Pour trier un tableau d'objets, il suffit de comparer les propriétés numériques des objets.

```
<body>
    <h2>Tableau d'objets avant le triage :</h2>
    <p id="demo"></p>
    <h2>Tableau d'objets après le triage selon age :</h2>
    <p id="demo2"></p>
    <script>
        var etudiants = [{
            prenom: "Amina ",
            age: 12
        }, {
            prenom: "Yahya ",
            age: 10
        }, {
            prenom: "Baraa ",
            age: 6
        }]
        for (i = 0; i < etudiants.length; i++) {
            document.getElementById('demo').innerHTML +=
                etudiants[i].prenom + '(' + etudiants[i].age + ') ';
        }

        etudiants.sort(function(a, b) {
            return a.age - b.age;
        });
        for (i = 0; i < etudiants.length; i++) {
            document.getElementById('demo2').innerHTML +=
                etudiants[i].prenom + '(' + etudiants[i].age + ') ';
        }
    </script>
</body>
```

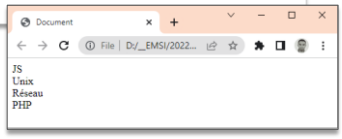
Les tableaux en JavaScript {Itération}

- l'itération sur un tableau signifie effectuer une opération pour chaque élément du tableau.
- Vous pouvez parcourir un tableau à l'aide de :
  - La boucle « **for** »
  - La méthode « **forEach()** » : exécute une fonction fournie pour chaque élément du tableau. Cette fonction peut prendre 1 à 3 arguments.
    - Syntaxe : `nomTableau.forEach( function(valeur, index, tableau){…});`
  - La méthode « **map()** » : crée un nouveau tableau dont chaque élément est modifié en utilisant la logique de la fonction fournie. De même elle peut prendre 1 à 3 arguments.
    - Syntaxe : `nouveauTableau = nomTableau.map( function(valeur, index, tableau) {return …});`
  - La méthode « **filter()** » : crée un nouveau tableau qui ne contient que les éléments ayant passé le test de la fonction fournie. De même elle peut prendre 1 à 3 arguments.
  - Syntaxe : `nouveauTableau = nomTableau.filter( function(valeur, index, tableau) {return condition});`

Les tableaux en JavaScript {Itération}

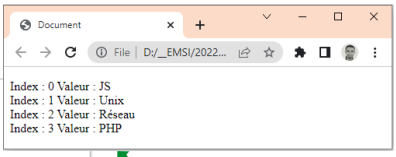
Avec la boucle "for"

```
<script>
    var cours = ["JS", "Unix", "Réseau", "PHP"];
    for (var i = 0; i < cours.length; i++) {
        document.write(cours[i] + '<br>');
    }
</script>
```



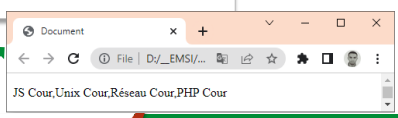
Avec la méthode "forEach()"

```
<p id="demo"></p>
<script>
    var cours = ["JS", "Unix", "Réseau", "PHP"];
    cours.forEach(function(valeur, index, tableau) {
        document.getElementById('demo').innerHTML +=
            "Index : " + index + " Valeur : " + valeur + " <br>";
    });
</script>
```



Avec la méthode "map()"

```
<p id="demo"></p>
<script>
    var cours = ["JS", "Unix", "Réseau", "PHP"];
    var cours2 = cours.map(function(valeur) {
        return valeur + ' Cours'
    });
    document.getElementById('demo').innerHTML += cours2
</script>
```



Avec la méthode "filter()"

```
<p id="demo"></p>
<script>
    var cours = ["JS", "Unix", "Réseau", "PHP"];
    var cours2 = cours.filter(function(valeur) {
        return valeur.length <= 3;
    });
    document.getElementById('demo').innerHTML += cours2
</script>
```