En JavaScript, les **tableaux** sont des objets très puissants avec plusieurs méthodes pratiques pour manipuler et travailler avec leurs éléments. Voici une liste des méthodes les plus courantes et utiles :

1. push()

Ajoute un ou plusieurs éléments à la fin d'un tableau et retourne la nouvelle longueur du tableau.

```
let arr = [1, 2];

arr.push(3); // arr devient [1, 2, 3]
```

2. pop()

Supprime le dernier élément d'un tableau et le retourne.

```
let arr = [1, 2, 3];

arr.pop(); // arr devient [1, 2] et retourne 3
```

3. shift()

Supprime le premier élément d'un tableau et le retourne.

```
let arr = [1, 2, 3];

arr.shift(); // arr devient [2, 3] et retourne 1
```

4. unshift()

Ajoute un ou plusieurs éléments au début d'un tableau et retourne la nouvelle longueur du tableau.

```
let arr = [2, 3];

arr.unshift(1); // arr devient [1, 2, 3]
```

5. concat()

Fusionne deux ou plusieurs tableaux et retourne un nouveau tableau.

```
let arr1 = [1, 2];

let arr2 = [3, 4];

let arr3 = arr1.concat(arr2); // arr3 devient [1, 2, 3, 4]
```

6. join()

Retourne une chaîne de caractères formée par la concaténation des éléments du tableau, séparés par un délimiteur spécifié.

let arr = ['a', 'b', 'c'];

let str = arr.join(', '); // "a, b, c"

7. slice()

Retourne une copie superficielle d'une portion du tableau dans un nouveau tableau.

let arr = [1, 2, 3, 4];

let sliced = arr.slice(1, 3); // [2, 3]

8. splice()

Permet de modifier un tableau en ajoutant, supprimant ou remplaçant des éléments à partir d'une position donnée.

let arr = [1, 2, 3, 4];

arr.splice(2, 1, 5); // arr devient [1, 2, 5, 4] (remplace 3 par 5)

9. forEach()

Exécute une fonction donnée sur chaque élément du tableau.

let arr = [1, 2, 3];

arr.forEach(num => console.log(num)); // Affiche 1, 2, 3

10. map()

Crée un nouveau tableau avec les résultats de l'appel d'une fonction sur chaque élément du tableau d'origine.

let arr = [1, 2, 3];

let doubled = arr.map(num => num * 2); // [2, 4, 6]

11. filter()

Crée un nouveau tableau contenant uniquement les éléments qui passent un test (défini par une fonction).

let arr = [1, 2, 3, 4];

let evenNumbers = arr.filter(num => num % 2 === 0); // [2, 4]

12. reduce()

Applique une fonction sur chaque élément du tableau (de gauche à droite) et retourne une seule valeur qui est le résultat de la réduction.

```
let arr = [1, 2, 3];

let sum = arr.reduce((acc, num) => acc + num, 0); // 6
```

13. find()

Retourne le premier élément du tableau qui satisfait une condition donnée.

```
let arr = [1, 2, 3, 4];

let found = arr.find(num => num > 2); // 3
```

14. indexOf()

Retourne l'index du premier élément trouvé qui correspond à la valeur donnée, ou -1 si l'élément n'existe pas.

```
let arr = [10, 20, 30];

let index = arr.indexOf(20); // 1
```

15. includes()

Vérifie si un élément est présent dans le tableau et retourne true ou false.

```
let arr = [1, 2, 3];

let isPresent = arr.includes(2); // true
```

16. sort()

Trie les éléments du tableau en place. (Peut être personnalisé avec une fonction de comparaison.)

```
let arr = [3, 1, 2];

arr.sort(); // [1, 2, 3]
```

17. reverse()

Inverse l'ordre des éléments dans le tableau.

```
let arr = [1, 2, 3];

arr.reverse(); // [3, 2, 1]
```

18. some()

Vérifie si au moins un élément du tableau passe un test (renvoie true ou false).

```
let arr = [1, 2, 3];

let hasEven = arr.some(num => num % 2 === 0); // true
```

19. every()

Vérifie si tous les éléments du tableau passent un test.

```
let arr = [2, 4, 6];

let allEven = arr.every(num => num % 2 === 0); // true
```

20. findIndex()

Retourne l'index du premier élément qui satisfait une condition.

```
let arr = [1, 2, 3, 4];

let index = arr.findIndex(num => num > 2); // 2
```

Ces méthodes couvrent un large éventail d'opérations courantes sur les tableaux. Elles sont utiles pour manipuler, filtrer, et transformer des données dans une application JavaScript.

En JavaScript, la classe **Math** offre plusieurs méthodes très utiles pour effectuer des calculs mathématiques. Voici une liste des méthodes les plus courantes et leur utilisation :

1. Math.abs(x)

Retourne la valeur absolue de x (c'est-à-dire la distance de x à zéro, indépendamment de son signe).

```
Math.abs(-5);  // 5
```

2. Math.round(x)

Retourne l'entier le plus proche de x. Si x est à égalité avec deux entiers, le plus proche vers zéro est retourné.

```
Math.round(4.7);  // 5

Math.round(4.4);  // 4
```

3. Math.floor(x)

Retourne l'entier inférieur à x (arrondi à l'entier le plus bas).

```
Math.floor(4.7);  // 4
```

4. Math.ceil(x)

Retourne l'entier supérieur à x (arrondi à l'entier le plus haut).

```
Math.ceil(4.3);  // 5
```

5. Math.max(x, y, ...)

Retourne le plus grand des nombres passés en arguments.

```
Math.max(1, 2, 3, 4);  // 4
```

6. Math.min(x, y, ...)

Retourne le plus petit des nombres passés en arguments.

```
Math.min(1, 2, 3, 4);  // 1
```

7. Math.random()

Retourne un nombre pseudo-aléatoire entre 0 (inclus) et 1 (exclus).

```
Math.random();  // Exemple: 0.3721782145111221
```

8. Math.pow(base, exponent)

Retourne base élevé à la puissance de exponent.

```
Math.pow(2, 3);  // 8
```

9. Math.sqrt(x)

Retourne la racine carrée de x.

```
Math.sqrt(16);  // 4
```

10. Math.sin(x) / Math.cos(x) / Math.tan(x)

Ces méthodes retournent respectivement le sinus, le cosinus et la tangente de l'angle x exprimé en radians.

```
Math.sin(Math.PI / 2);  // 1
```

11. Math.log(x)

Retourne le logarithme naturel (base e) de x. Si vous voulez un logarithme à une autre base, vous pouvez utiliser une formule comme Math.log(x) / Math.log(base) pour la base que vous souhaitez.

```
Math.log(Math.E);  // 1
```

12. Math.exp(x)

Retourne e élevé à la puissance de x (soit Math.E^x).

```
Math.exp(1);  // 2.718281828459045
```

13. Math.trunc(x)

Retourne la partie entière de x en retirant sa partie décimale.

```
Math.trunc(4.9);  // 4

Math.trunc(-4.9);  // -4
```

14. Math.PI

Représente la constante π (pi) en JavaScript.

```
Math.PI;  // 3.141592653589793
```

15. Math.E

Représente la constante e (la base des logarithmes naturels).

```
Math.E;  // 2.718281828459045
```

Ces méthodes couvrent une large gamme d'opérations mathématiques courantes en JavaScript. Si tu as une question spécifique sur une méthode ou sur la manière de les utiliser ensemble, n’hésite pas à me demander !