CSS

Les structures de répétition en JavaScript

- Une structure for
- Une structure for ... of
- Une structure for ... in
- Une structure while
- Une structure do ... while

Les structures for, for ... of et for ... in

```
<script>
    // Création de notre variable message.
    let message = "";
    for(i = 0; i < 5; i++)
    {
        message += 'Bienvenue dans le cours "Développement Web II"\n';
    }
    alert(message);
</script>
```

Les structures for, for ... of et for ... in

Ce type de boucle est utilisé afin de récupérer les valeurs d'un objet. La boucle ***for ... of*** est utilisable sur différents objets comme des tableaux, que nous verrons dans un prochain chapitre, et des chaines de caractères, pour ne citer que ces exemples. Examinez le prochain bloc de code.

```
<script>
    // Création de notre variable message.
    let message = "Bienvenue";
    // Création et initialisation de notre variable de contrôle.
    let x;
    // Création de notre variable pour affichage final.
    let affichage = "";
    for(x of message)
    {
        affichage += x + '\n';
    }
    alert(affichage);
</script>
```

JAVASCRIPT

Ce type de boucle un peu spéciale ne retourne pas chacun des éléments de notre chaine de caractères comme la boucle ***for ... of*** que nous venons d'étudier. Bien au contraire, elle retourne la valeur de la position de notre caractère dans notre chaine de caractères. Examinez le prochain bloc de code.

```
<script>
    // Création de notre variable message.
    let message = "Bienvenue";
    // Création et initialisation de notre variable de contrôle.
    let x;
    // Création de notre variable pour affichage final.
    let affichage = "";
    for(x in message)
    {
        affichage += x + '\n';
    }
    alert(affichage);
</script>
```

JAVASCRIPT

Récapitulation

Dans cette leçon, nous avons découvert comment utiliser trois différentes structures de répétition soit :

- La structure ***for***.
- La structure ***for ... of***.
- La structure ***for ... in***.

La structure ***for*** nécessite l'utilisation de trois paramètres. La syntaxe générale de cette structure de répétition est la suivante :

```
for (Argument 1; Argument 2; Argument 3)
{
    // Bloc de code à répéter.
}
```

- Argument 1 est exécuté qu'une seule fois, avant l'exécution du bloc de code.
- Argument 2 définie la condition pour l'exécution du bloc de code à répéter.
- Argument 3 est exécuté après chaque exécution du bloc de code à répéter.

Les arguments de la boucle for sont tous optionnels mais, il faut demeurer prudent lorsque nous utilisons cette boucle en omettant la déclaration de certains paramètres.

La boucle ***for ... of*** est, de son côté, un type de boucle utilisé afin de récupérer des valeurs contenues dans un objet comme un tableau ou une chaine de caractères pour ne citer que ces exemples. La structure de la boucle ***for ... of*** est la suivante :

```
for (variable of élément itérable)
{
    // Bloc de code à répéter.
}
```

La boucle ***for ... in*** est, de son côté, un type de boucle utilisé afin de récupérer la position des valeurs contenues dans un objet comme un tableau ou une chaine de caractères pour ne citer que ces exemples. La structure de la boucle ***for ... in*** est la suivante :

```
for (variable in élément itérable)
{
    // Bloc de code à répéter.
}
```

Exercice 4 : Somme des entiers de 1 à n

1. Demander à l'utilisateur un nombre n.
2. Calculer et afficher la somme des entiers de 1 à n.

Les structures while et do ... while

Dans ce type de boucle, les instructions comprises dans le bloc de code à être répété seront exécutées tant et aussi longtemps que condition retournera ***vrai***. Examinons ensemble un premier exemple d'utilisation de cette boucle.

```
<script>
    var texte = "";
    var i = 0;
    while (i < 10)
    {
        texte += "Le nombre est " + i + "\n";
        i++;
    }
    alert(texte += 'Fin de l\'exécution de la boucle "while".');
</script>
```

Dans ce type de boucle, les instructions comprises dans le bloc de code à être répété seront exécutées tant et aussi longtemps que condition retournera ***vrai***. Examinons ensemble un exemple d'utilisation de cette boucle.

```
<script>
    var texte = "";
    var i = 0;
    do
    {
        texte += "Le nombre est " + i + "\n";
        i++;
    }while (i < 10);
    alert(texte += 'Fin de l\'exécution de la boucle "do ... while".');
</script>
```

JAVASCRIPT

Exercice

En utilisant les trois types de boucles que nous venons d'étudier, la boucle ***for***, la boucle ***while*** et la boucle ***do ... while***, créez un message qui affichera à cinq reprises le texte ***J'aime faire de la programmation en JavaScript***.

JAVASCRIPT

Les instructions break et continue

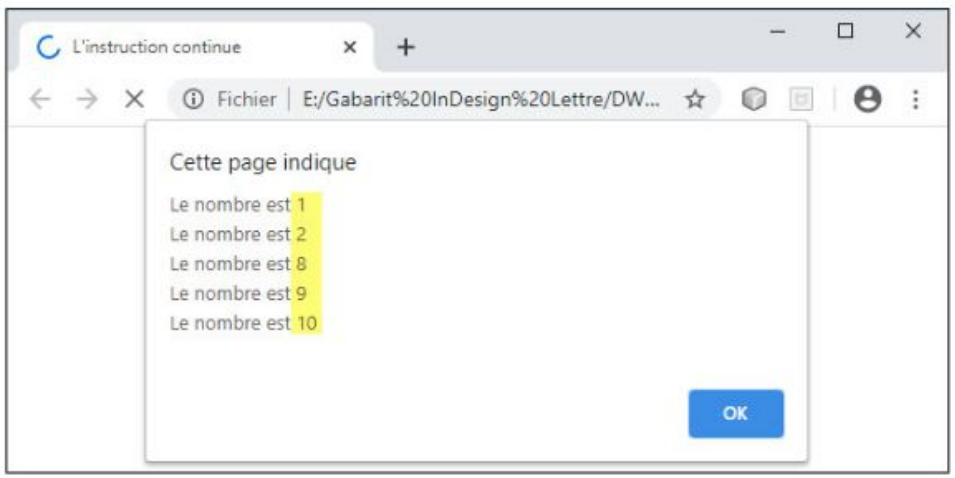
```
<script>
    // Création de notre variable message.
    let message = "";
    for(i = 0; i < 5; i++)
    {
        if(i == 2)
        {
            break;
        }
        message += 'Bienvenue dans le cours "Développement Web II"\n';
    }
    alert(message);
</script>
```

L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.

JAVASCRIPT

Les instructions break et continue

```
<script>
    var texte = "";
    var i = 0;
    while (i < 10)
    {
        i++;
        if(i >= 3 && i < 8)
        {
            continue;
        }
        texte += "Le nombre est " + i + "\n";
    }
    alert(texte);
</script>
```



Évaluez vos connaissances nouvellement acquises

Lire attentivement les différents énoncés puis, encercler la lettre correspondant à la bonne réponse.

1. Examinez attentivement le prochain script.

```
<script>
    for(var i = 0; i < 5; i++)
    {
        alert("Bonjour");
    }
</script>
```

Après l'exécution de ce script, combien de fois sera affiché le mot ***Bonjour*** ?

a. 0
b. 1
c. 4
d. 5

2. Vrai ou faux, une boucle while se comporte comme une boucle for dont les valeurs de ***Argument 1*** et ***Argument 3*** ont été omis.

a. Vrai
b. Faux

Évaluez vos connaissances nouvellement acquises

3. Examinez attentivement le prochain script.

```
<script>
    var a = 5;
    while( a < 10  || a > 10)
    {
        a++;
    }
</script>
```

Quelle sera la valeur de ***a*** suite à l'exécution de ce script ?

a. Il est impossible qu'une valeur soit plus petite et plus grande que 10 en même temps.
b. Nous faisons face à une boucle infinie.
c. 10
d. 11

4. Examinez attentivement le prochain script.

```
<script>
    let a = 5;
    do{
        a++;
    }while(a > 10);
</script>
```

Quelle sera la valeur de ***a*** suite à l'exécution de ce script ?

a. 5
b. 6
c. 7
d. Aucune de ces réponses

Évaluez vos connaissances nouvellement acquises

5. Vrai ou faux, il n'est pas nécessaire d'inclure les instructions du corps d'une boucle entre deux accolades même si le corps de notre boucle contient plus d'une instructions ?

   a. Vrai
   b. Faux

6. Quel est l'effet de l'instruction continue si elle est incluse dans le corps d'une boucle ?

   a. Elle met fin à l'exécution des instructions d'une boucle lorsque rencontrée.
   b. Elle poursuit l'exécution de la boucle sans exécuter les instructions qui se trouvent après.
   c. Cette instruction n'a aucun effet sur l'exécution d'une boucle.
   d. Cette instruction créera une boucle sans fin.

7. Examinez attentivement le prochain script.

```
<script>
    for(i = 0; i < 5; i++)
    {
        if( i > 2){
            continue;
        }
        alert("Bonjour");
    }
</script>
```

   Après l'exécution de ce script, combien de fois sera affiché le mot ***Bonjour*** ?

   a. 0
   b. 2
   c. 3
   d. 5

Exercices d’application

Exercice 1 : Compter de 1 à 10 avec une boucle `for`

Utiliser une boucle `for` pour afficher les nombres de 1 à 10.

Exercice 2 : Somme des entiers de 1 à n

1. Demander à l'utilisateur un nombre `n`.
2. Calculer et afficher la somme des entiers de 1 à `n`.

Exercice 3 : Compter les voyelles dans une chaîne de caractères

1. Demander une chaîne de caractères à l'utilisateur.
2. Compter et afficher le nombre de voyelles (`a`, `e`, `i`, `o`, `u`, `y`).

Déclaration et initialisation d'un tableau en JavaScript

```
<script>
    let voitures = ["Saab", "Audi", "BMW", "Mercedes"];
</script>
```

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
</script>
```

Accéder à un élément d'un tableau

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    alert(voitures[1]);
</script>
```

JAVASCRIPT

Ajouter un élément dans un tableau

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    let message = "";
    voitures.push("Cadillac");
    for(i = 0; i < voitures.length; i++)
    {
        message += voitures[i] + "\n";
    }
    alert(message);
</script>
```

JAVASCRIPT

Récapitulation

Dans cette leçon, nous avons découvert comment déclarer et initialiser un tableau. La déclaration et l'initialisation d'un tableau en JavaScript peut se faire de deux façons, soit :

- En utilisant la ***notation entre crochets***.
- En utilisant les mots clé ***new Array()***.

La propriété ***length*** d'un tableau retourne le nombre d'éléments que contient notre tableau.

Nous récupérons une valeur dans un tableau en utilisant la ***notation entre crochet***.

En JavaScript, contrairement aux autres langages de programmation, un tableau n'est pas un objet statique. Il est donc possible d'ajouter des éléments à la suite des éléments que contient notre tableau en utilisant les deux techniques suivantes :

- En utilisant la ***notation entre crochets*** et la ***propriété length*** de notre tableau.
- En utilisant la méthode ***push()*** appliquée sur notre objet tableau.

Il faut être prudent lorsque nous ajoutons des valeurs dans un tableau en utilisant la ***notation entre crochet*** afin de ne pas créer de ***trous*** dans notre tableau, c'est-à-dire l'ajout d'***éléments undefined*** à l'intérieur de notre tableau.

Il est possible d'utiliser tous les types de boucles que nous avons étudiés dans le chapitre précédent afin de récupérer l'ensemble des éléments contenus dans un tableau, les boucles ***for***, ***while***, ***do ... while***, ***for ... of*** et ***for ... in***.

La boucle ***for ... in***, de son côté, retourne les indices des éléments contenus dans un tableau. Il est donc possible d'utiliser la notation entre crochets en utilisant les valeurs des indices retournées afin de récupérer les valeurs associées à ces indices dans un tableau.

En JavaScript, un tableau a la capacité de contenir des valeurs de différents types.

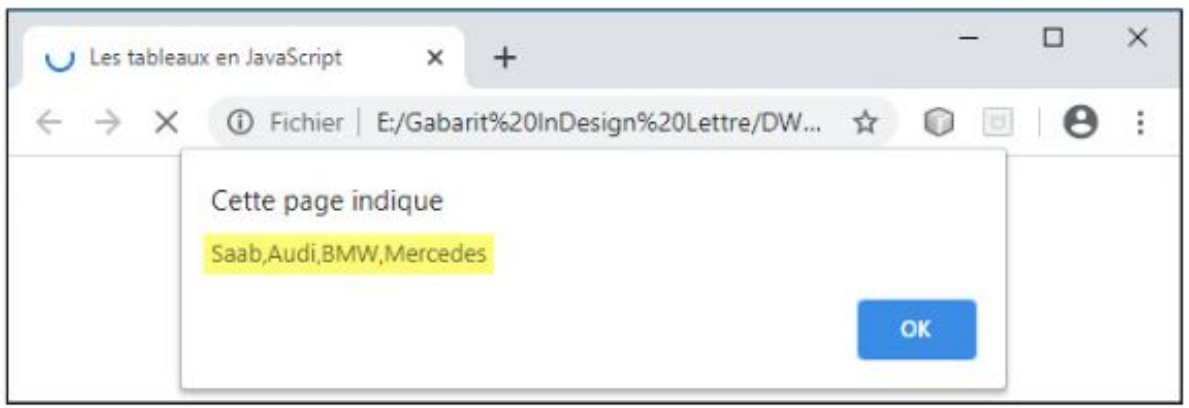
Autres méthodes utilisables avec les tableaux en JavaScript

La méthode toString()

La méthode ***toString()*** appliquée sur un tableau convertit le contenu d'un tableau en une chaine de caractères dont tous les éléments sont séparés par une virgule. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    alert(voitures.toString());
</script>
```

L'exécution de ce script affichera le résultat de la figure suivante à l'écran.

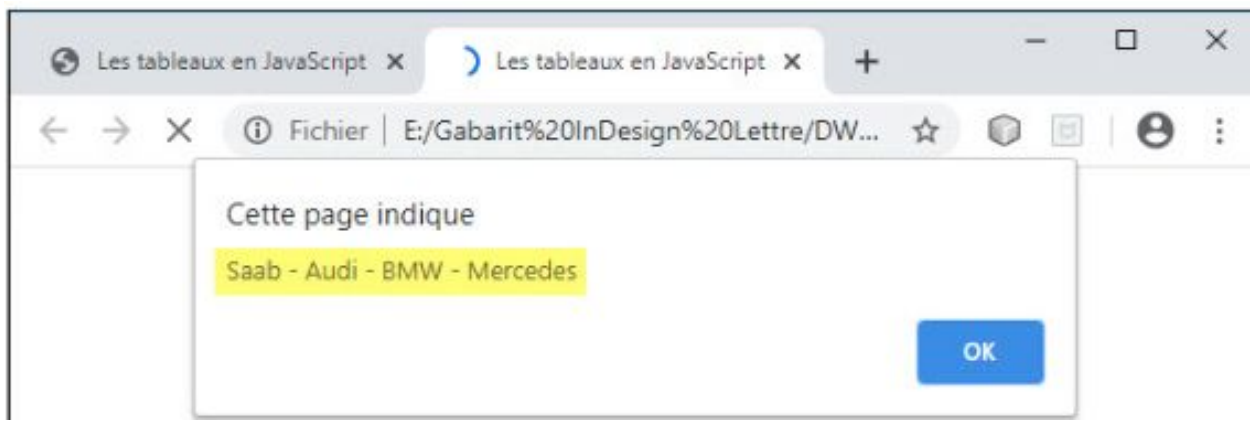


La méthode join()

Cette méthode se comporte comme la méthode ***toString()***. Un des avantage de cette méthode est que vous pouvez appliquer le caractère séparateur que vous désirez appliquer. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera la concaténation de tous les éléments que contient notre tableau en les séparant d'une espace suivi d'un trait d'union et d'un deuxième espace, comme le montre la prochaine figure.



JAVASCRIPT

La méthode pop()

La méthode ***pop()*** appliquée sur un tableau supprime le dernier élément de notre tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.pop();
    alert(voitures.join(" - "));
</script>
```

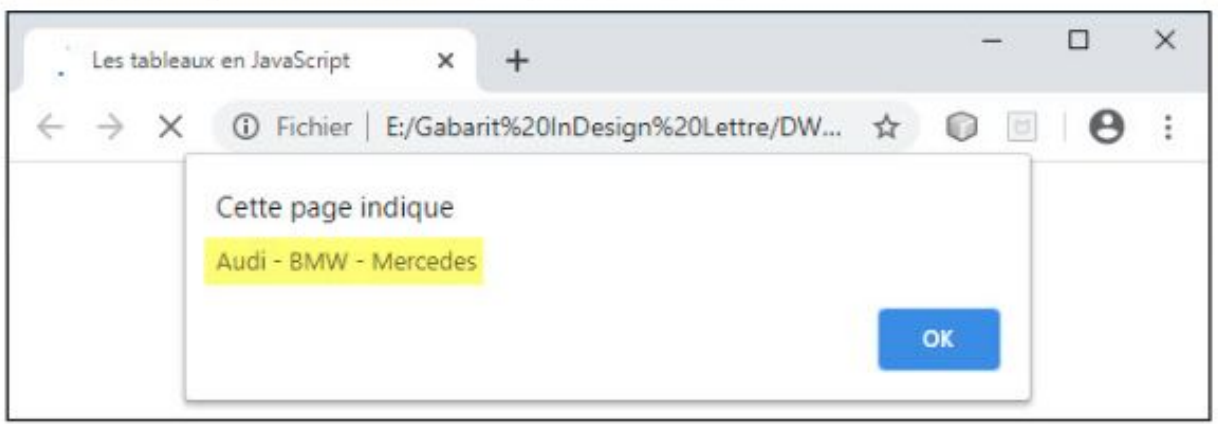
L'exécution de ce script affichera le résultat de la figure suivante à l'écran.

La méthode shift()

La méthode ***shift()*** est équivalente et produit le même résultat que la méthode ***pop()*** sauf qu'elle est appliquée sur le premier élément de notre tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.shift();
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera le résultat de la figure suivante à l'écran.

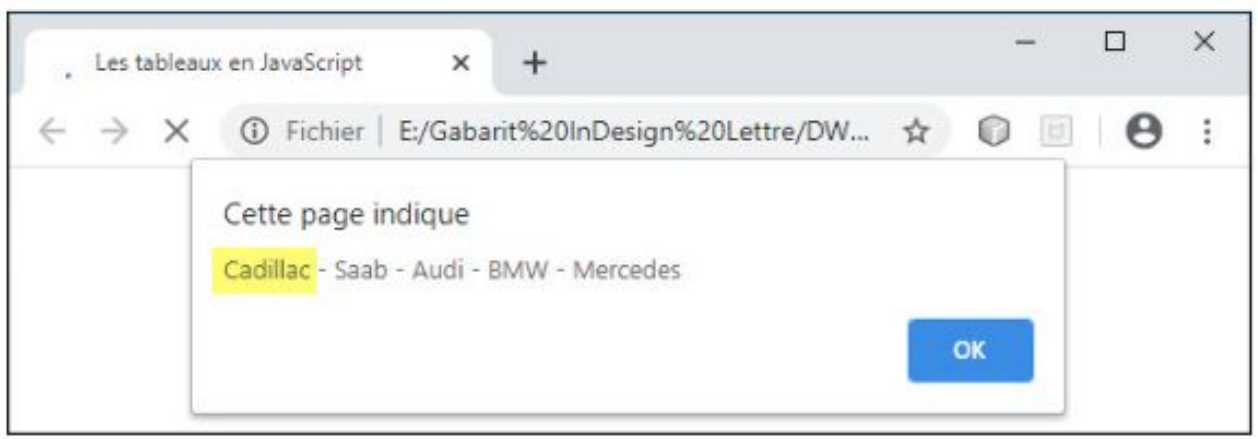


La méthode unshift()

La méthode ***unshift()*** est équivalente et produit le même résultat que la méthode ***push()*** sauf qu'elle est appliquée au début de notre tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.unshift("Cadillac");
    alert(voitures.join(" - "));
</script>
```

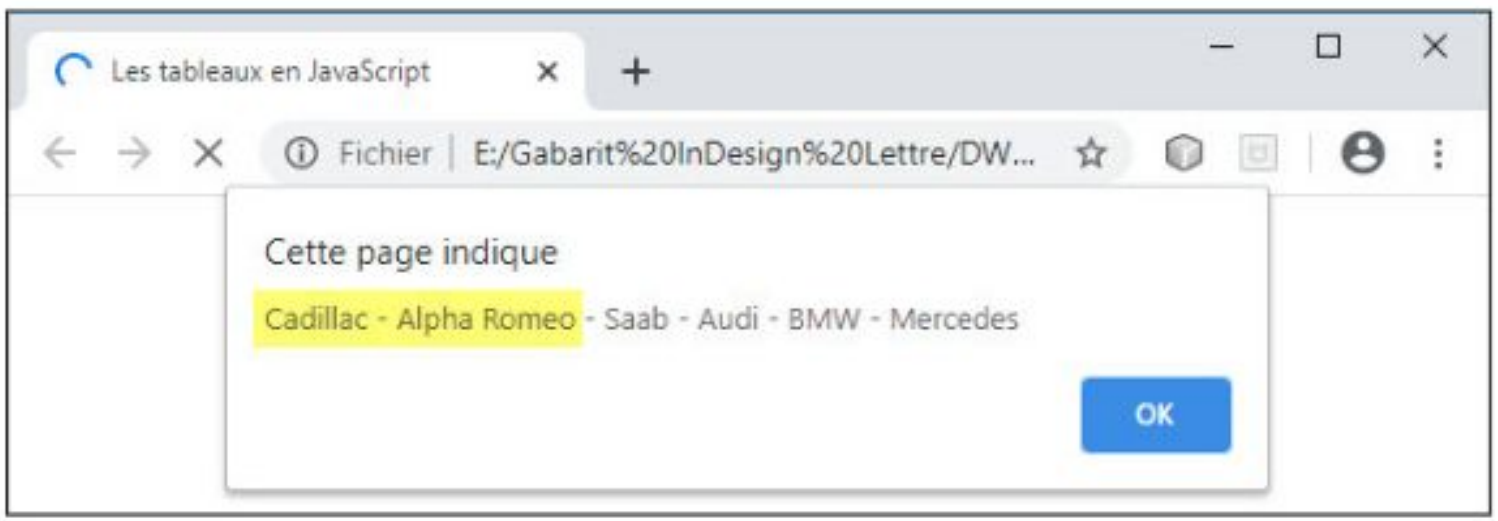
L'exécution de ce script affichera le résultat de la figure suivante à l'écran.



Tout comme nous l'avons vu avec la méthode ***push()***, la méthode ***unshift()*** nous permet aussi d'ajouter plus d'un élément au début de notre tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.unshift("Cadillac", "Alpha Romeo");
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera le résultat de la figure suivante à l'écran.



JAVASCRIPT

splice()

La méthode splice()

La méthode ***splice()*** peut être utilisée afin d'effectuer différentes opérations sur notre tableau.

La syntaxe générale de cette méthode est la suivante :

tableau.**splice**(*Argument 1, Argument 2, Argument 3, Argument n*);

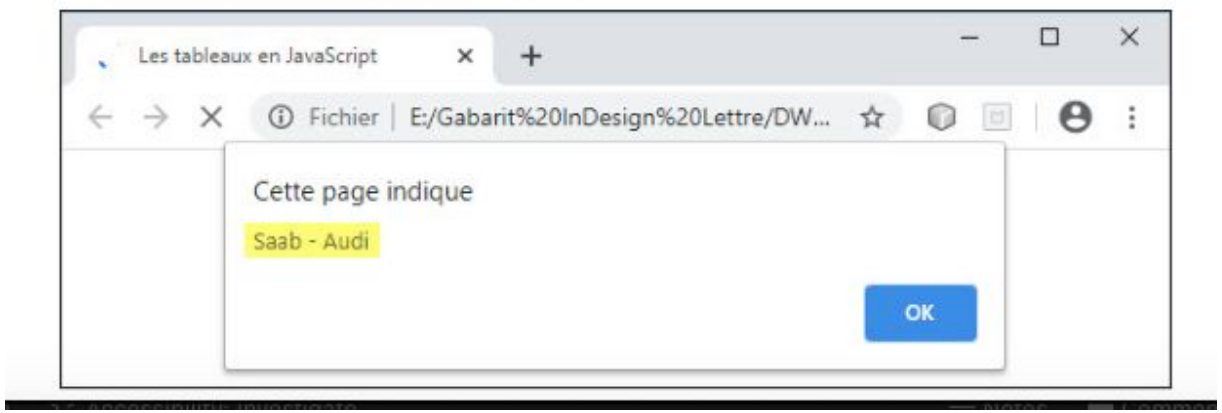
Cette méthode prend au minimum un argument afin de fonctionner. L'utilisation de cette méthode sans argument n'aura aucun effet sur le contenu de notre tableau.

- ***Argument 1*** est l'index de départ de notre méthode.
- ***Argument 2*** est le nombre d'éléments qui seront affectés par la méthode.
- ***Argument 3*** et les suivants sont les valeurs qui doivent être utilisées pour l'insertion ou le remplacement des valeurs visées par les deux premiers arguments.

Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(2);
    alert(voitures.join(" - "));
</script>
```

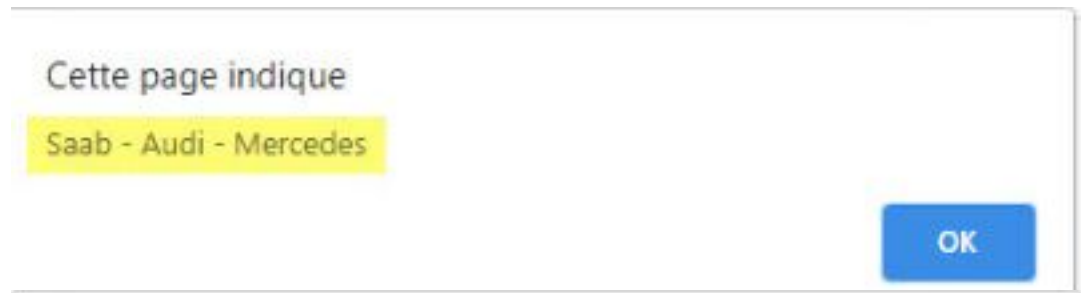
Dans ce script, tous les éléments, à partir de l'élément à l'indice 2, seront supprimés de notre tableau. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



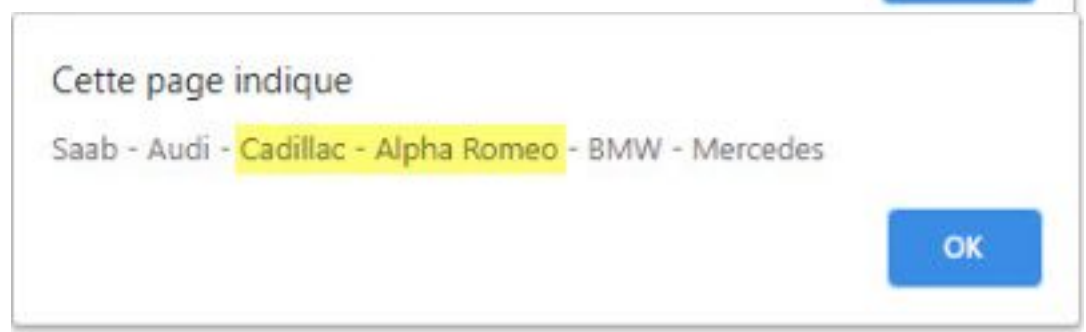
JAVASCRIPT

splice()

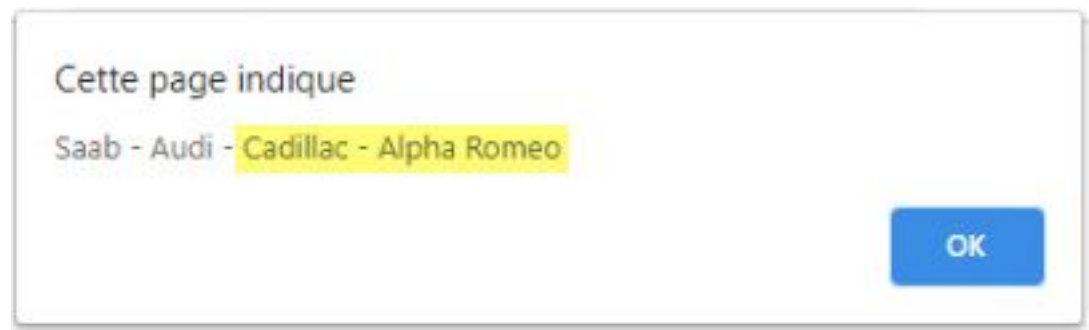
```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(2, 1);
    alert(voitures.join(" - "));
</script>
```




```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(2, 0, "Cadillac", "Alpha Romeo");
    alert(voitures.join(" - "));
</script>
```




```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(2, 2, "Cadillac", "Alpha Romeo");
    alert(voitures.join(" - "));
</script>
```

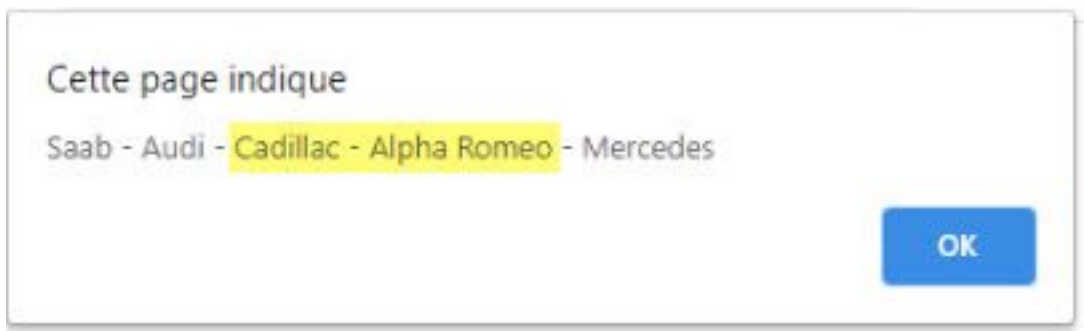


JAVASCRIPT

splice()

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(1, 3, "Cadillac", "Alpha Romeo");
    alert(voitures.join(" - "));
</script>
```


```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    voitures.splice(2, 1, "Cadillac", "Alpha Romeo");
    alert(voitures.join(" - "));
</script>
```



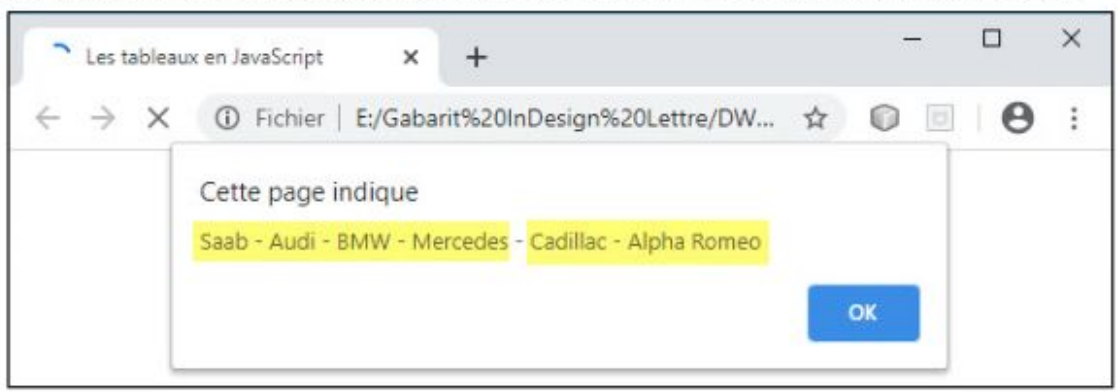
concat()

La méthode concat()

Vous l'aurez sûrement deviné, la méthode ***concat()*** est utilisée afin d'effectuer une concaténation de tableaux. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    let mesVoitures = new Array("Cadillac", "Alpha Romeo");
    let liste = voitures.concat(mesVoitures);
    alert(liste.join(" - "));
</script>
```

La méthode ***concat()*** retourne un nouveau tableau sans apporter de modification aux tableaux existants. Dans ce script, nous avons créé une nouvelle variable nommée liste afin de recevoir le tableau issu de la concaténation de nos tableaux ***voitures*** et ***mesVoitures***. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.

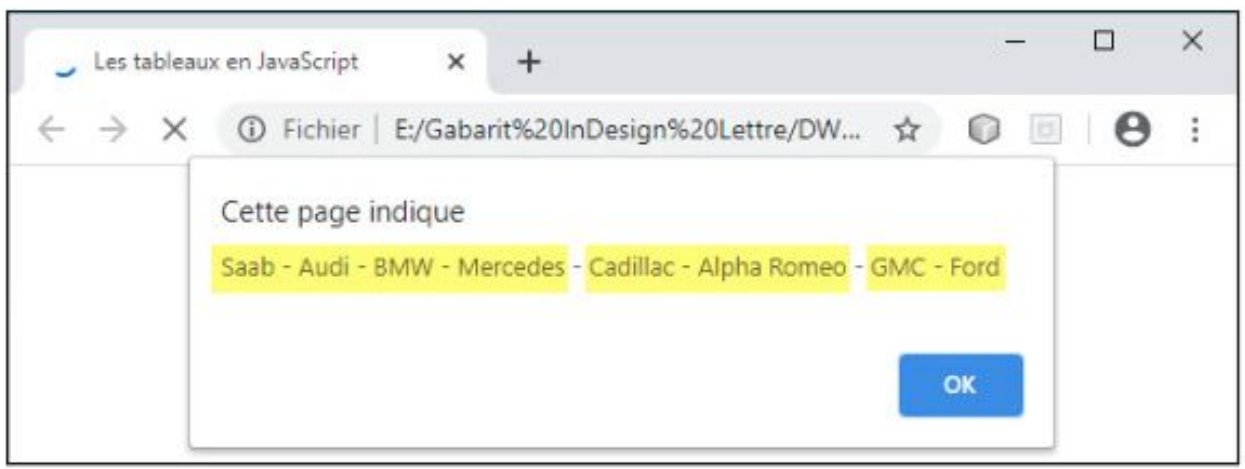


concat()

La méthode ***concat()*** peut prendre plus d'un argument. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    let mesVoitures = new Array("Cadillac", "Alpha Romeo");
    let americaines = new Array("GMC", "Ford");
    let liste = voitures.concat(mesVoitures, americaines);
    alert(liste.join(" - "));
</script>
```

Dans ce script, notre variable ***liste*** sera initialisée avec le résultat de la concaténation de nos trois tableaux, ***voitures***, ***mesVoitures*** et ***americaines***. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



concat()

La méthode ***concat()*** peut aussi prendre en paramètre un tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes");
    let liste = voitures.concat(["Cadillac", "Alpha Romeo", "GMC", "Ford"]);
    alert(liste.join(" - "));
</script>
```

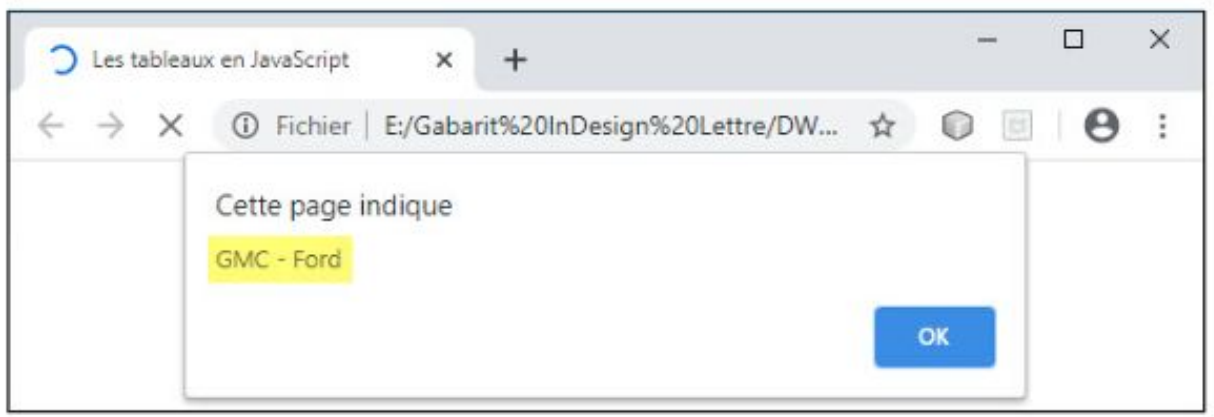
Dans ce script, notre variable ***liste*** sera initialisée avec le résultat de la concaténation de notre tableau, ***voitures*** et des valeurs qui lui sont passées en paramètre sous forme d'un nouveau tableau. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.

La méthode slice()

La méthode ***slice()*** est utilisée afin de récupérer une partie d'un tableau dans un nouveau tableau. Cette méthode peut prendre un ou deux arguments. Examinez le prochain bloc de code.

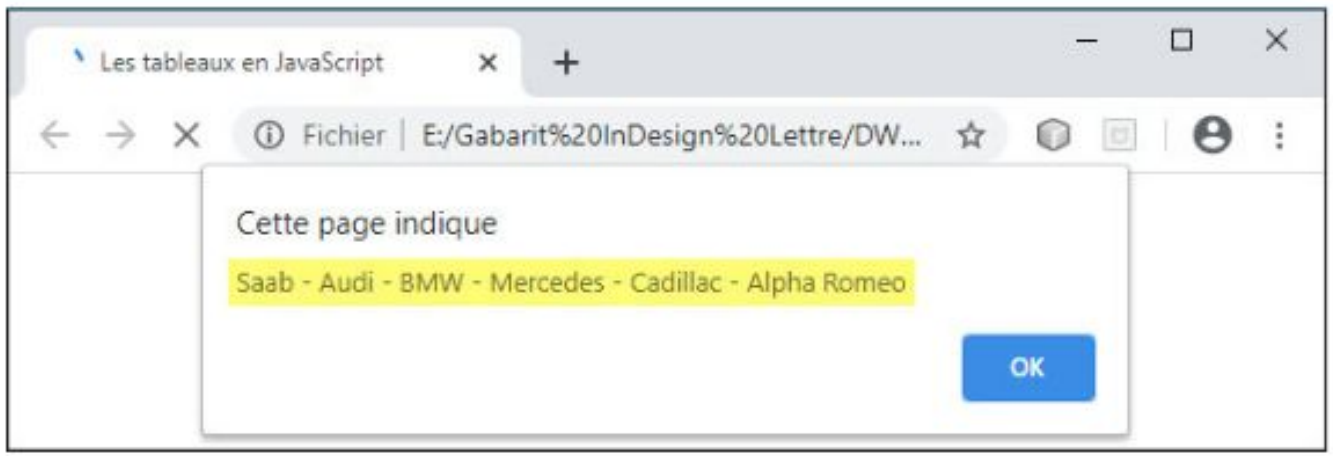
```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo", "GMC", "Ford");
    let liste = voitures.slice(6);
    alert(liste.join(" - "));
</script>
```

Dans ce premier exemple, la méthode ***slice()*** récupérera tous les éléments du tableau qui se trouve à partir de l'indice ***6*** jusqu'à la fin du contenu du tableau. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo", "GMC", "Ford");
    let liste = voitures.slice(0, 6);
    alert(liste.join(" - "));
</script>
```

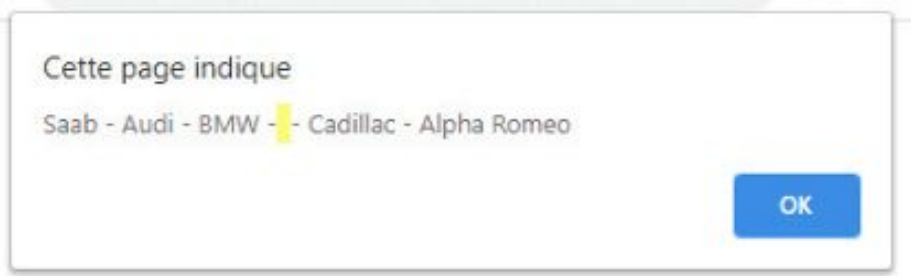
Dans ce script, la méthode ***slice()*** accepte deux paramètres. Le premier des deux représente l'***indice de début*** de la sélection et le deuxième paramètre indique l'***indice de fin*** de la sélection qui sera exclus dans la sélection. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



JAVASCRIPT

Étant donné qu'un tableau est un objet en JavaScript, il est possible d'utiliser l'opérateur ***delete*** de JavaScript afin de supprimer un élément dans un tableau. ***Attention***, cet opérateur supprime seulement que la valeur se trouvant à l'indice que l'on désire supprimer. La valeur sera remplacée par une valeur non définie. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo");
    delete voitures[3];
    alert(voitures.join(" - "));
</script>
```



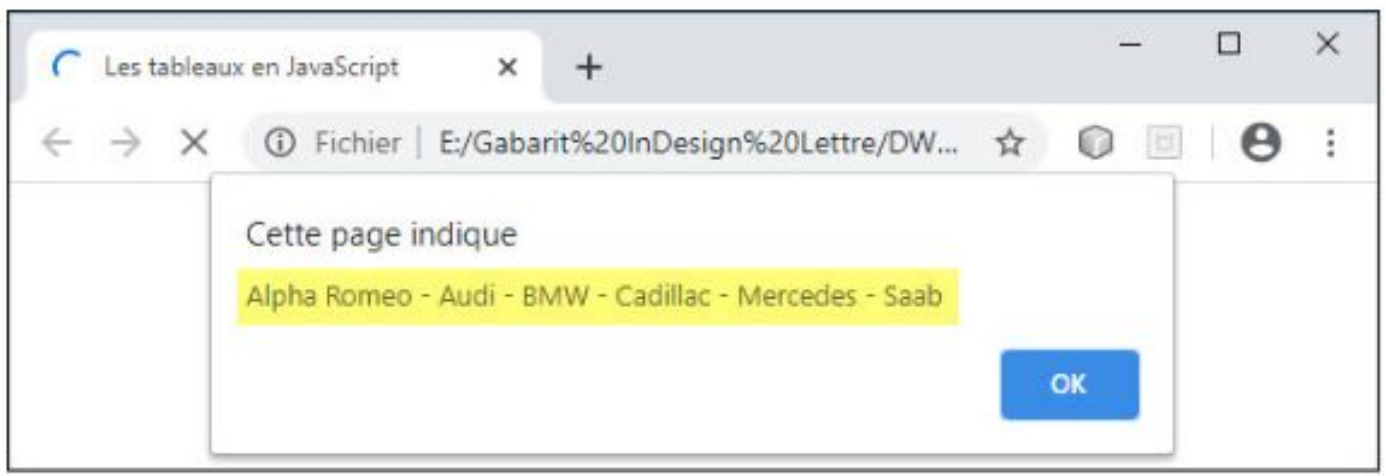
sort()

La méthode sort()

La méthode ***sort()*** est utilisée afin de mettre en ordre un tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo");
    voitures.sort();
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



sort()

Attention, cette méthode ne fonctionne que sur des éléments de type ***string***. Ainsi, les valeurs numérique ***25*** et ***100*** ne seront pas comparées correctement et la valeur ***«25»*** sera toujours supérieur à la valeur ***«100»*** sous forme de chaine de caractères, comme le montre le prochain exemple. Afin de pouvoir mettre en ordre des nombres, il faudra utiliser des méthodes en JavaScript. Le prochain bloc de code affichera le résultat de la prochaine figure à l'écran.

```
<script>
    let nombres = new Array(45, 75, 100, 25, 32, 10);
    nombres = nombres.sort();
    alert(nombres.join(" - "));
</script>
```

Javascript

You can fix this by providing a **compare function**:

Example

```
const points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

Try it Yourself »

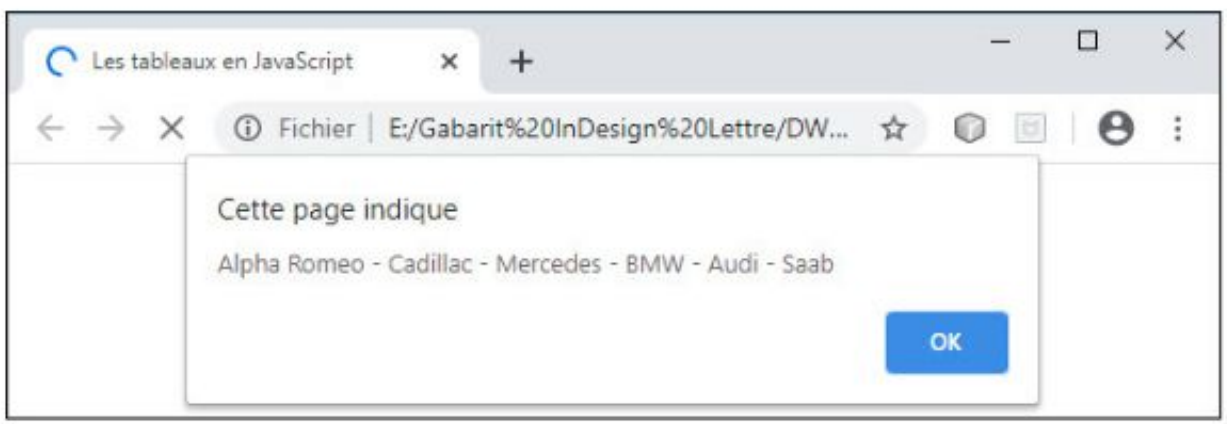
reverse()

La méthode reverse()

La méthode ***reverse()*** est utilisée afin d'inverser l'ordre des éléments d'un tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo");
    voitures.reverse();
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.

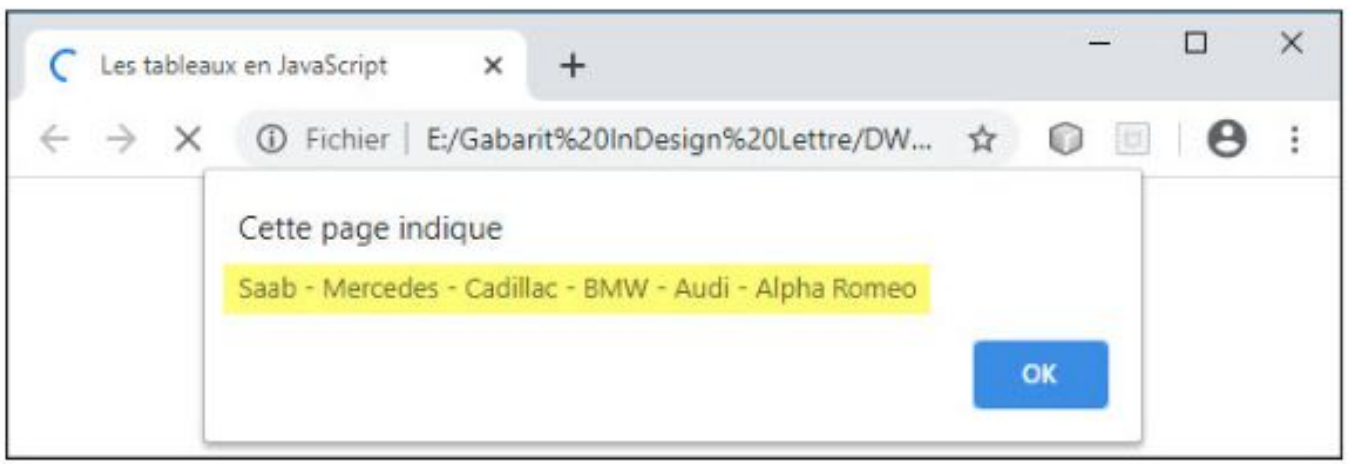


sort() et reverse()

En combinant la méthode ***sort()*** avec la méthode ***reverse()***, nous sommes donc en mesure d'obtenir un classement en ordre décroissant des éléments de notre tableau. Examinez le prochain bloc de code.

```
<script>
    let voitures = new Array("Saab", "Audi", "BMW", "Mercedes",
                             "Cadillac", "Alpha Romeo");
    voitures.sort();
    voitures.reverse();
    alert(voitures.join(" - "));
</script>
```

L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



Récapitulation

Dans cette leçon, nous avons découvert plusieurs méthodes applicables sur un objet tableau en JavaScript, soit :

- La méthode ***toString()*** qui convertit le contenu d'un tableau en une chaine de caractères dont tous les éléments sont séparés par une virgule.
- La méthode ***join()*** qui se comporte comme la méthode ***toString()***. Un des avantage de cette méthode est que vous pouvez appliquer le caractère séparateur que vous désirez appliquer.
- La méthode ***pop()*** qui supprime le dernier élément d'un tableau.
- La méthode ***shift()*** qui est équivalente et produit le même résultat que la méthode ***pop()*** sauf qu'elle est appliquée sur le premier élément d'un tableau
- La méthode ***unshift()*** qui est équivalente et produit le même résultat que la méthode ***push()*** sauf qu'elle est appliquée au début de notre tableau. Cette méthode peut accepter plus d'un paramètre à la fois.
- La méthode ***splice()*** qui peut être utilisée afin d'effectuer différentes opérations sur notre tableau.
  - Cette méthode peut prendre de un à plusieurs arguments.
  - Si on utilise qu'un seul argument, l'indice de départ, les éléments que contient notre tableau seront supprimés à partir de cet indice jusqu'au dernier élément.
  - Si on utilise deux arguments, l'indice de départ et le nombre d'éléments, les éléments se trouvant à partir de l'indice de départ seront supprimés selon le nombre qui est passée comme deuxième paramètre.
  - Tous les arguments à partir du troisième argument de la méthode sont des valeurs de remplacement pour les éléments identifiés par les deux premiers paramètres.
- La méthode ***concat()*** est utilisée afin d'effectuer une concaténation de tableaux. Cette méthode peut prendre plus d'un paramètre et retourne un nouveau tableau issu de l'opération de concaténation.
- La méthode ***slice()*** est utilisée afin de récupérer une partie d'un tableau dans un nouveau tableau. Cette méthode prend un ou deux paramètres afin de pouvoir effectuer une sélection dans le tableau d'origine. Le premier paramètre est l'indice de départ de la sélection et le deuxième est l'indice du dernier élément qui sera exclus de la sélection.
- La méthode ***sort()*** nous permet de placer en ordre croissant les éléments d'un tableau. ***Attention***, cette méthode ne fonctionne qu'avec des éléments de type ***string***. Les éléments de type ***numérique*** seront convertis en type ***string*** lors de l'exécution de cette méthode.
- La méthode ***reverse()*** est utilisée afin d'inverser l'ordre des éléments contenus dans un tableau.
- Si l'on combine les deux dernières méthodes, ***sort()*** et ***reverse()***, nous sommes en mesure de placer les éléments d'un tableau en ordre décroissant.
- Pour terminer nous avons découvert l'opérateur ***delete*** de JavaScript qui nous permet de supprimer un élément dans un tableau en le remplaçant par une valeur non définie.

Évaluez vos connaissances nouvellement acquises

Lire attentivement les différents énoncés puis, encercler la lettre correspondant à la bonne réponse.

1. Examinez attentivement le prochain bloc de code.

```
<script>
   let nombres = new Array(25, 10, 35, 12, 1);
   nombres = nombres.sort();
   nombres.reverse();
   alert(nombres.join(« - »));
</script>
```

Quel sera le résultat affiché suite à l'exécution de ce script ?

a. 35 - 25 - 12 - 10 - 1
b. 1 - 10 - 12 - 25 - 35
c. 12 - 10 - 1 - 35 - 25
d. 25 - 35 - 1 - 10 - 12

Évaluez vos connaissances nouvellement acquises

3. Examinez attentivement le prochain bloc de code.

```
<script>
   let tab = new Array(25, 50, 75, 100);
   for(i = 0; i < tab.length; i++)
   {
      tab[i] = tab[i] *2;
   }
   alert(tab.join(« - »));
</script>
```

Quel sera le résultat affiché à l'écran suite à l'exécution de ce script ?

a. Une page blanche
b. 25 - 50 - 75 - 100
c. 50 - 100 - 150 - 200
d. Aucune de ces réponses

4. Examinez attentivement le prochain bloc de code.

```
<script>
   let tab = new Array(25, 50, 75, 100);
   let total = 0;
   for(i = 0; i < tab.length - 2; i++)
   {
      total += tab[i];
   }
</script>
```

Quelle sera la valeur de ***total*** suite à l'exécution de ce script ?

a. 25
b. 50
c. 75
d. Aucune de ces réponses

Évaluez vos connaissances nouvellement acquises

6. Parmi les choix suivants, lequel est une méthode qui ajoute un élément au début d'un tableau sans supprimer aucune valeur de ce dernier ?

   a. unshift()
   b. push()
   c. insert()
   d. Aucune de ces méthodes

7. Examinez attentivement le prochain bloc de code.

```
<script>
   let tab = new Array(«Marc», «Sylvie», «Samuel», «Line»);
   let total = «»;
   for(i = 1; i < tab.length - 2; i++)
   {
      total += tab[i];
   }
</script>
```

   Après l'exécution de ce script, quel sera la valeur de ***total*** ?

   a. Marc
   b. Sylvie
   c. Samuel
   d. Line