

JAVASCRIPT

Les opérateurs en JavaScript

<i>Opération</i>	<i>Opérateur</i>	<i>Exemple</i>	<i>Description</i>
<i>Multiplification</i>	*	A * B	Multiplier A par B
<i>Division</i>	/	A / B	Diviser A par B
<i>Modulo</i>	%	A % B	Retourne le reste de la division entière de A par B
<i>Élevé à la puissance</i>	**	A ** B	Élève A à la puissance de B
<i>Addition</i>	+	A + B	Additionner B à A
<i>Soustraction</i>	-	A - B	Soustraire B de A

JAVASCRIPT

Exercice

Donner les étapes de l'opération suivante :

$$((25 - 3) \% 4) * (4 + 10) \% 5$$

```
<script>  
  var resultat = ((25 - 3) % 4) * (4 + 10) % 5;  
  alert(resultat);  
</script>
```

Pourquoi

?

JAVASCRIPT

Les opérateurs d'incrémentation et de décrémentation


```
<script>  
    // Opérateur post-incrémentation.  
    var x = 5;  
    var y = x++;  
    alert(y);  
    alert(x);  
</script>
```




JAVASCRIPT

L'opérateur de *pré-incrémentation* de son côté effectue l'opération contraire soit, l'incrément de la variable avant l'affectation de sa valeur à une autre variable. Donc, le script suivant affichera le résultat de la dernière figure à l'écran, à deux reprises.

```
<script>
  // Opérateur pré-incrémentation.
  var x = 5;
  var y = ++x;
  alert(y);
  alert(x);
</script>
```



```
<script>
  // Opérateur post-décrémentation.
  var x = 5;
  var y = x--;
  alert(y);
  alert(x);
</script>
```



JAVASCRIPT

Dans le prochain script, nous obtiendrons le même comportement qu'en *pré-incrémentation* en utilisant la *pré-décrémentation* c'est-à-dire que l'opération de décrémentation sera effectuée avant que la valeur ne soit récupérée par notre variable *y*.

```
<script>
  // Opérateur pré-décrémentation.
  var x = 5;
  var y = --x;
  alert(y);
  alert(x);
</script>
```



JAVASCRIPT

Les opérateurs d'assignation

X = 5

<i>Opérateur</i>	<i>Nom</i>	<i>Description</i>	<i>Exemple</i>
<code>+=</code>	Addition et assignation	Ajoute la valeur de droite à la valeur de la variable et assigne cette nouvelle valeur à notre variable.	<code>x += 4;</code> x sera égal à 9.
<code>-=</code>	Soustraction et assignation	Soustrait la valeur de droite à la valeur de la variable et assigne cette nouvelle valeur à notre variable.	<code>x -= 3;</code> x sera égal à 2.
<code>*=</code>	Multiplication et assignation	Multiplie la valeur de la variable par la valeur de droite et assigne cette nouvelle valeur à notre variable.	<code>x *= 2;</code> x sera égal à 10.
<code>/=</code>	Division et assignation	Divise la valeur de la variable par la valeur de droite et assigne cette nouvelle valeur à notre variable.	<code>x /= 2;</code> x sera égal à 2,5.
<code>%=</code>	Modulo et assignation	Récupère le reste de la division de notre variable par la valeur de droite et assigne cette valeur à notre variable.	<code>x %= 3;</code> x sera égal à 2.
<code>**=</code>	Élevé à la puissance et assignation	Élève à la puissance notre variable par la valeur de droite et assigne cette valeur à notre variable.	<code>x **= 3;</code> x sera égal à 125.

JAVASCRIPT

Les opérateurs de comparaison

Opération	Opérateur	Exemple	Résultat
Égal à	==	2 == 3	Faux
Inférieur à	<	2 < 3	Vrai
Supérieur à	>	2 > 3	Faux
Inférieur ou égal à	<=	2 <= 3	Vrai
Supérieur ou égal à	>=	2 >= 3	Faux
Différent de	!=	2 != 3	Vrai
Égalité des valeurs et des types	===	10 === '10'	Faux
Différent en valeur ou en type	!==	10 !== '10'	Vrai

```
<script>  
| alert('10' == 10);  
</script>
```

```
<script>  
| alert('10' >= 9);  
</script>
```

Cette page indique

true

OK

JAVASCRIPT

Exercice

```
<script>  
|   alert("'Marc' <= 'Alain' = " + ('Marc' <= 'Alain'));  
</script>
```

?

Pourquoi

JAVASCRIPT

Exercise

```
<script>  
|   alert("'10' === 10 = " + ('10' === 10));  
</script>
```

?

JAVASCRIPT

Exercise

```
<script>  
|   alert("'10' !== 10 = " + ('10' !== 10));  
</script>
```

?

JAVASCRIPT

Les opérateurs logiques

<i>Opération</i>	<i>Opérateur</i>
NON logique	!
ET logique	&&
OU logique	

Faisons une petite récapitulation de ces trois opérateurs logiques.

- Pour qu'un opérateur **logique ET** retourne **Vrai**, il faut que tous les opérandes de l'expression retourne **Vrai**.
- Pour qu'un opérateur **logique OU** retourne **Vrai**, il faut que l'un des opérandes de l'expression retourne **Vrai**.
- L'opérateur **logique NON** inverse le résultat de l'opérande ou de l'expression. Si le résultat est **Vrai**, le **NON logique** retournera **Faux** et inversement, si le résultat est **Faux**, le **NON logique** retournera **Vrai**.

JAVASCRIPT

Exercise

```
<script>
  let a = false;
  let b = true;
  alert(a && b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!(a && b));
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!a && b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(a || b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!(a || b));
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(a || !b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(a & b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!a & b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!(a & b));
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(a | b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(a | !b);
</script>
```

```
<script>
  let a = false;
  let b = true;
  alert(!(a | b));
</script>
```

JAVASCRIPT

Exercice

$((A + B == 25) \mid (C - A == 10)) \& \! (A + A == B)$

```
<script>
  let a = 10;
  let b = 20;
  let c = 30;
  alert('((a + b == 25) || (c - a == 10)) && !(a + a == b) = ' +
        (((a + b == 25) || (c - a == 10)) && !(a + a == b)));
</script>
```

Pourquoi

?

JAVASCRIPT

Exercice

Créer un document html qui contient 2 boutons .

1.Ecrire un script qui affiche le message " Hello World" après le clic sur le premier bouton.

1.Lors du clic sur le deuxième bouton demander à l'utilisateur de saisir son nom. ensuite demander une confirmation du nom de l'utilisateur. Si le nom est confirmé, afficher ce dernier dans une boîte de dialogue.

JAVASCRIPT

***Méthodes sur les chaînes
de caractères, fonctions
mathématiques et de dates***

JAVASCRIPT

La concaténation

```
<script>
  let prenom = 'Yves';
  alert('Bienvenue ' + prenom);
</script>
```

```
<script>
  let prenom = 'Yves';
  let message = 'Bienvenue ';
  alert(message += prenom);
</script>
```


JAVASCRIPT

La concaténation

```
<script>  
|   alert("'Marc' <= 'Alain' = " + ('Marc' <= 'Alain'));  
</script>
```

Cette page indique

'Marc' <= 'Alain' = false

OK

JAVASCRIPT

La méthode indexOf()

Cette méthode renvoie l'indice de début d'une sous-chaine au sein d'une chaîne de caractères. La valeur **-1** est renvoyée si aucune correspondance n'est trouvée. Examinons un exemple d'utilisation de cette méthode. Nous avons une chaîne de caractères dans laquelle nous désirons savoir si une sous-chaîne existe et si oui, à quel endroit se trouve cette sous-chaîne. Examinez le prochain bloc de code.

```
<script>  
  let message = "Bienvenue dans le cours Développement Web II";  
  alert ('J'ai trouvé le mot \'cours\' à l'index ' + message.indexOf('cours'));  
</script>
```

Cette page indique

J'ai trouvé le mot 'cours' à l'index 18

OK

JAVASCRIPT

La méthode lastIndexOf()

Cette méthode renvoie l'indice de la dernière occurrence d'une sous-chaine au sein de la chaine de caractères spécifiée ou *-1* si aucune correspondance n'est trouvée. Cette méthode s'utilise de la même manière que la méthode *indexOf()* que nous venons tout juste d'étudier.

JAVASCRIPT

La méthode toLowerCase()

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert(message.toLowerCase());
</script>
```

La méthode toUpperCase()

La méthode charAt()

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert('J\'ai trouvé la lettre ' + message.charAt(5) + ' à l\'index 5.');
```

La méthode substring()

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert('J\'ai trouvé la chaîne "' + message.substring(5, 8) + '" entre les index 5 et 8.');
```

La méthode search()

Cette méthode retourne un résultat identique à la méthode *indexOf()* que nous avons étudiée plus tôt dans cette leçon.
Examinez le prochain bloc de code.

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert('J\'ai trouvé le mot "cours" à l\'index ' + message.search('cours'));
</script>
```

JAVASCRIPT

La méthode slice()

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert(message.slice(10, 14));
</script>
```

Cette page indique

dans

OK

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert(message.slice(10));
</script>
```

Cette page indique

dans le cours Développement Web II.

OK

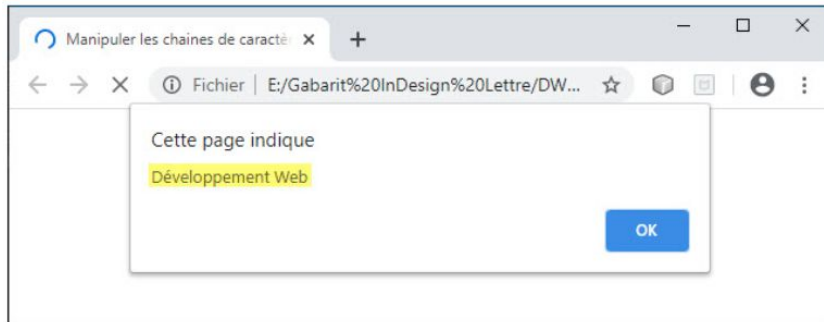
JAVASCRIPT

La méthode slice()

Il est aussi possible d'utiliser une valeur négative comme indice. Examinez le prochain exemple.

```
<script>
  let message = 'Bienvenue dans le cours Développement Web II.';
  alert(message.slice(24, -3));
</script>
```

Dans cet exemple, la chaîne de caractères retournée commencera avec la lettre se trouvant à l'indice 24 et sera complétée par tous les caractères jusqu'à la fin de notre chaîne moins les trois derniers caractères. L'exécution de ce script affichera le résultat de la prochaine figure à l'écran.



JAVASCRIPT

La méthode replace()

Cette méthode recherche une correspondance d'une sous-chaine dans une chaîne de caractères et remplace cette sous-chaine par une nouvelle. Rien de mieux qu'un bon exemple pour bien comprendre le fonctionnement de cette méthode.

```
<script>  
  let message = 'Le malheur des uns fait le bonheur des autres.';  
  alert(message.replace('malheur', 'bonheur'));  
</script>
```

JAVASCRIPT

La méthode endsWith()

Cette méthode retourne la valeur **true** si la chaîne de caractères se termine par la valeur qui lui est passée en paramètre et **false** dans le cas contraire. Examinez le prochain bloc de code.

```
<script>
  var message1 = "Ce texte est affiché avec JavaScript depuis l'élément head.";
  alert(message1.endsWith('head'));
</script>
```


JAVASCRIPT

La méthode includes()

Cette méthode vérifie si la valeur passée en paramètre existe à l'intérieur de la chaîne de caractères examinée par la méthode. Cette méthode retourne *true* si la valeur est trouvée dans la chaîne de caractères et *false* dans le cas contraire. Examinez le prochain bloc de code.

```
<script>  
  var message1 = "Ce texte est affiché avec JavaScript depuis l'élément head.";  
  alert(message1.includes('head'));  
</script>
```

JAVASCRIPT

La méthode match()

La méthode ***match()*** recherche la valeur passée en paramètre dans la chaîne de caractères visée. Si cette valeur est trouvée, elle retourne cette valeur et ***null*** dans le cas contraire. Examinez le prochain bloc de code.

```
<script>
  var message1 = "Le chien noir et le chien blanc s'amuse ensemble.";
  alert(message1.match('chien'));
</script>
```

JAVASCRIPT

La méthode repeat()

Cette méthode crée une nouvelle chaîne de caractères avec le nombre de copies spécifié de la chaîne de caractères utilisée par la fonction. Examinez le prochain bloc de code.

```
<script>
  var message1 = "Bonjour le monde ! ";
  alert(message1.repeat(3));
</script>
```

JAVASCRIPT

La méthode startsWith()

Cette méthode effectue la même recherche et retourne les mêmes résultats que la fonction *endsWith()* à la différence qu'elle effectuera la recherche de la valeur qui lui est passée en paramètre au début de la chaîne de caractères. Examinez le prochain bloc de code.

```
<script>
  var message1 = "Bonjour le monde ! ";
  alert(message1.startsWith('Bon'));
</script>
```

JAVASCRIPT

La méthode substr()

Cette méthode extrait le nombre spécifié de caractères du texte sur lequel elle est appliquée à partir de la position spécifiée. Le premier paramètre spécifie la position de départ et le deuxième paramètre, le nombre de caractères. Si le deuxième paramètre est omis, la méthode retourne tous les caractères à partir de la position de départ jusqu'à la fin de la chaîne de caractères. Examinez le prochain bloc de code.

```
<script>  
  var message1 = "Bonjour le monde ! ";  
  alert(message1.substr(3));  
</script>
```

JAVASCRIPT

La méthode toString()

Cette méthode, aussi inutile qu'elle puisse paraître, existe en JavaScript. Elle retourne la valeur de l'objet string. Examinez le prochain bloc de code.

```
<script>
  var message1 = "Bonjour le monde ! ";
  alert(message1.toString());
</script>
```

JAVASCRIPT

La méthode trim()

La méthode *trim()* élimine tous les espaces blancs en début et en fin de chaîne. Examinez le prochain bloc de code.

```
<script>
  var message1 = "    Bonjour le monde !    ";
  alert("\"" + message1 + "\"");
  alert("\"" + message1.trim() + "\"");
</script>
```

JAVASCRIPT

La propriété length

Cette propriété retourne le nombre de caractères que contient une chaîne de caractères. Examinez le prochain bloc de code.

```
<script>
  let message = 'Le malheur des uns fait le bonheur des autres.';
  alert('La chaîne "' + message + '" contient ' + message.length + ' caractères.');
```