Apprendre JavaScript - Les chaînes de caractères

Table des matières

Introduction

Les chaînes de caractères sont des objets qui représentent une séquence de caractères. Elles sont utilisées pour stocker et manipuler du texte.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

Création d'une chaîne de caractères

Pour créer une chaîne de caractères, vous pouvez utiliser des guillemets simples, doubles ou les `backticks` (`).

```
let guillemetsSimples = "Hello, World!";

let guillemetsDoubles = "Hello, World!";

let backticks = `Hello, World!`;
```

Concaténation de chaînes de caractères

La concaténation est le processus de mise bout à bout de deux chaînes de caractères.

```
let prenom = "John";
let nom = "Doe";

let nomComplet = prenom + " " + nom;

console.log(nomComplet); // John Doe
```

Aujourd'hui, il est plus courant d'utiliser les `backticks` pour concaténer des chaînes de caractères. Pour cela, vous pouvez utiliser la syntaxe `${}`. À l'intérieur des accolades, vous pouvez mettre des variables, des expressions ou des fonctions.

```
let prenom = "John";
let nom = "Doe";

let nomComplet = `${prenom} ${nom}`;
```

Rechercher des éléments dans une chaîne de caractères

Longueur d'une chaîne de caractères

Pour obtenir la longueur d'une chaîne de caractères, vous pouvez utiliser la propriété `length`.

```
let prenom = "John";

console.log(prenom.length); // 4
```

Comparer des chaînes de caractères en incluant la casse et les accents

Pour comparer des chaînes de caractères en incluant la casse et les accents, vous pouvez utiliser la méthode `localeCompare`avec l'option `sensitivity` pour spécifier la sensibilité à la casse et aux accents. Cela est utilisé

souvent dans la fonction `sort` pour trier des chaînes de caractères.

La fonction prend 3 paramètres : la chaîne de caractères à comparer, la locale de la langue (ex: fr, en, es) et un objet d'options pour la sensibilité à la casse et aux accents.

```
let prenom1 = "Élizabeth";
let prenom2 = "eliZabeth";

console.log(prenom1.localeCompare(prenom2)); // 1 donc prenom1 est après prenom2

console.log(prenom1.localeCompare(prenom2, "fr", { sensitivity: "base" })); // 0
donc prenom1 est égal à prenom2
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl/Collator/Collator#sensitivity

Rechercher un élément

Méthode `indexOf`

Pour rechercher un élément dans une chaîne de caractères, vous pouvez utiliser la méthode `indexOf`. Cette méthode renvoie l'index de la première occurrence de l'élément recherché. Si l'élément n'est pas trouvé, la méthode renvoie `-1`.

```
let prenom = "John";

console.log(prenom.indexOf("o")); // 1

console.log(prenom.indexOf("a")); // -1
```

Méthode `includes`

La méthode `includes` renvoie `true` si la chaîne de caractères contient l'élément recherché, sinon elle renvoie `false`. Il n'indique pas la position de l'élément.

```
let prenom = "John";

console.log(prenom.includes("o")); // true

console.log(prenom.includes("a")); // false
```

Méthode `search`

La méthode `search` renvoie l'index de la première occurrence de l'élément recherché. Si l'élément n'est pas trouvé, la méthode renvoie `-1`. La méthode `search` peut également rechercher des expressions régulières.

```
// Pour vérifier si l'élément est un code postal type H1H 1H1

let codePostal = "H1H 1H1";
console.log(codePostal.search(/[A-Z][0-9][A-Z] [0-9][A-Z][0-9]/)); // 0
```

Remplacer des éléments

Méthode `replace`

La méthode `replace` remplace un élément par un autre. Il faut réenregistrer la chaîne de caractères dans une variable pour conserver les modifications. La méthode `replace` ne modifie pas la chaîne de caractères d'origine et peut prendre en paramètre une expression régulière.

```
let prenom = "John";

let prenomModifie = prenom.replace("John", "Jane");

console.log(prenomModifie); // Jane
```

Pour remplacer toutes les occurrences d'un élément, la fonction replaceAll peut être utilisée.

```
let prenom = "John John";
let prenomModifie = prenom.replaceAll("John", "Jane");

console.log(prenomModifie); // Jane Jane
```

Extraire des éléments

Méthode `slice`

La méthode `slice` extrait une partie d'une chaîne de caractères et renvoie une nouvelle chaîne de caractères. Vous pouvez spécifier un index de début et un index de fin.

Si vous ne spécifiez que l'index de début, la méthode renverra la partie de la chaîne de caractères à partir de l'index de début jusqu'à la fin de la chaîne de caractères.

Il est possible de stocker la nouvelle chaîne de caractères dans une variable pour conserver les modifications.

```
```

```
let prenom = "John";

console.log(prenom.slice(1, 3, 5)); // ohn

console.log(prenom.slice(1)); // ohn
```

Transformer des éléments

Méthode `toUpperCase`, `toLowerCase`

La méthode `toUpperCase` transforme tous les caractères d'une chaîne de caractères en majuscules.

La méthode `toLowerCase` transforme tous les caractères d'une chaîne de caractères en minuscules.

```
let prenom = "John";

console.log(prenom.toUpperCase()); // JOHN

console.log(prenom.toLowerCase()); // john
```

Pour transformer uniquement la première lettre d'une chaîne de caractères en majuscule, vous pouvez extraire la première lettre, la méthode `toUpperCase` pour la transformer en majuscule et la méthode `slice` pour extraire le reste de la chaîne de caractères en concaténant les deux.

```
let prenom = "john";

let prenomModifie = prenom[0].toUpperCase() + prenom.slice(1);
```

Méthode `split`

La méthode `split` divise une chaîne de caractères en un tableau de sous-chaînes. Vous pouvez spécifier un séparateur pour diviser la chaîne de caractères.

```
let prenom = "John Doe";

let prenomTableau = prenom.split(" ");

console.log(prenomTableau); // ["John", "Doe"]
```

Méthode `trim`

La méthode `trim` supprime les espaces en début et en fin de chaîne de caractères. Cette méthode est souvent utilisée pour nettoyer les données saisies par l'utilisateur qui pourraient contenir des espaces inutiles. Trim nettoie uniquement les espaces en début et en fin de chaîne de caractères.

```
let prenom = "  John      ";

console.log(prenom.trim()); // John
```

Méthode `padStart`, `padEnd`

La méthode `padStart` ajoute des caractères au début d'une chaîne de caractères pour atteindre une longueur spécifiée. Très pratique lors de l'affichage de numéros de produits d'une longueur fixe.

La méthode `padEnd` ajoute des caractères à la fin d'une chaîne de caractères pour atteindre une longueur spécifiée.

```
let numeroProduit = "123";
console.log(numeroProduit.padStart(5, "0")); // 00123
```