Les objets JavaScript String et Array

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : Visual Studio Code

Contexte

Chaque langage de programmation a sa propre structure de données, et c'est évidemment aussi le cas de JavaScript avec ses propres structures de données natives. Comme vous le savez, un des fondements de la programmation consiste en la définition de variables sur une donnée. Une donnée peut être de différents types tels que des chaînes de caractères (*string*) ou des tableaux (*arrays*).

Les développeurs utilisent ces deux types de données régulièrement lorsqu'ils écrivent des scripts. L'objet string est un objet permettant d'envelopper un type primitif string, et donc de l'utiliser comme un objet. Chaque objet possède ses propres propriétés et méthodes qui permettent de manipuler les chaînes de caractères et les tableaux, par exemple pour ajouter ou supprimer un élément. Pour illustrer, prenons le cas d'une chaîne de caractères : si nous définissons une variable avec une valeur de texte, nous pouvons obtenir le nombre de caractères dans ce texte en utilisant la propriété « *length* » de l'objet string.

Afin d'obtenir les résultats souhaités de manière efficace, il est crucial de savoir quel type de données utiliser et quelles propriétés et méthodes il faut associer pour les manipuler. Dans un premier temps, nous examinerons les concepts de chaînes de caractères et de tableaux. Ensuite, nous verrons comment utiliser différentes propriétés et méthodes pour effectuer diverses opérations sur ces types de données. L'objectif est de maîtriser suffisamment les deux concepts pour pouvoir effectuer diverses manipulations.

II. Les concepts string et array

A. Les concepts string et array

Le concept string en JavaScript

En langage de programmation, le terme « *string* » (qui signifie « *chaîne* » en anglais) désigne une séquence de caractères. Les chaînes de caractères sont considérées comme un type de données primitif en programmation. En JavaScript, les chaînes de caractères peuvent être représentées soit sous forme de types de données primitifs, soit sous forme d'objets string.

Toutes les valeurs littérales placées entre des guillemets simples ('), des guillemets doubles (") ou des backticks (`) sont considérées comme des chaînes de caractères primitives en JavaScript.

La distinction entre les chaînes de caractères primitives et les objets string est la suivante : en JavaScript, les chaînes de caractères primitives sont des valeurs simples qui ne possèdent pas de méthodes ou de propriétés associées. En revanche, les objets string sont créés en utilisant le constructeur string (avec ou sans le mot-clé « *new* »), et ils possèdent des méthodes et des propriétés spécifiques. Comme nous l'avons dit, un objet string permet d'envelopper (« *wrapper* ») une valeur primitive string. Il permet d'utiliser des méthodes et des propriétés spécifiques des objets sur la valeur primitive enveloppée.

Prenons un exemple pour mieux comprendre.

Exemple **La chaîne de caractères de type string passe au type objet**

```
let str_primi = "mandarine" ;
let str_obj = new String(str_primi) ;
console.log (typeof str_primi) ;
console.log (typeof str_obj) ;
```

L'affichage console sera :

String

Object

Explications :

La première ligne crée une variable **str_primi** contenant une chaîne de caractères primitive « *mandarine* ». La deuxième ligne crée une variable **str_obj** qui contient un objet string créé à partir de la variable **str_primi**.

Les deux dernières lignes utilisent la fonction **typeof** pour afficher le type de chaque variable dans la console. La première ligne affichera « *string* », ce qui indique que la variable **str_primi** contient une chaîne de caractères primitive. La deuxième ligne affichera « *object* », ce qui indique que la variable **str_obj** contient un objet string.

Ainsi, l'exemple montre que même si les deux variables contiennent la même chaîne de caractères, l'une est une chaîne de caractères primitive et l'autre est un objet string.

Comment utiliser la propriété « length » pour obtenir la longueur d'une chaîne de caractères en JavaScript ?

Les objets string en JavaScript sont des objets qui possèdent des propriétés et des méthodes qui leur sont propres. Il est important de les connaître pour pouvoir les utiliser efficacement. L'une des propriétés principales d'un objet string est « *length* ». Cette propriété permet de déterminer la longueur de la chaîne primitive enveloppée en retournant le nombre exact de caractères qu'elle contient. Mais cette propriété est aussi accessible via le type primitif string. En effet, lorsque nous cherchons à accéder à une méthode sur une chaîne de caractères primitive, JavaScript crée un objet temporaire qui enveloppe cette chaîne, afin d'appeler cette méthode. C'est ce principe qui permet d'appeler des méthodes comme length sur des valeurs primitives string.

Il existe deux façons courantes d'utiliser la propriété « *length* » sur une chaîne de caractères primitive. La première consiste à créer une variable qui contient la chaîne de caractères, par exemple :

```
1 let str = 'abcd';
```

Ensuite, pour obtenir la longueur de la chaîne, nous utilisons la propriété « *length* » de cette manière :

```
1 console.log(str.length);
```

La deuxième méthode consiste à créer une nouvelle chaîne de caractères directement dans l'argument du console.log, comme ceci :

```
1 console.log('efghi'.length);
```

Dans cette méthode, la nouvelle chaîne de caractères doit être placée entre guillemets simples.

Dans les deux cas, nous obtiendrons respectivement les valeurs quatre et cinq, qui correspondent à la longueur de la chaîne stockée dans la variable **str** et celle de la nouvelle chaîne créée directement dans l'argument du console.log.

Méthodes les plus couramment utilisées pour manipuler les chaînes de caractères

Une chaîne de caractères en JavaScript possède plusieurs méthodes qui permettent de la manipuler de différentes manières. Il est important de connaître les méthodes les plus couramment utilisées, telles que **CharAt()**, **indexOf()**, **lastIndexOf()**, **slice()**, **split()**, etc. Ces méthodes peuvent être utiles pour ajouter ou extraire des éléments d'une chaîne.

D'abord, nous allons voir comment échapper certains caractères. Si nous ne les échappons pas correctement, le navigateur peut confondre ces caractères et cela peut causer des erreurs.

Il vous faut savoir comment les caractères sont positionnés à l'intérieur d'une chaîne de caractères, car cela peut affecter le fonctionnement de certaines méthodes. Nous aborderons cela plus en détail lorsque nous parlerons de la manipulation de l'objet string à l'aide de méthodes.

Attention **L'échappement de certains caractères**

Des caractères sont reconnus comme une chaîne parce qu'ils sont placés entre des guillemets simples ou entre des guillemets doubles. Si une chaîne est entourée de guillemets simples, il peut y avoir une confusion lorsque nous insérons des apostrophes, par exemple. Pour l'éviter, il est nécessaire de placer le backslash avant cette apostrophe pour l'échapper.

Exemple **L'échappement de l'apostrophe, des doubles guillemets et des antislashs**

```
let str = 'C\'est un exemple d\'utilisation de l\'échappement de caractères';
```

Dans cet exemple, nous avons échappé les apostrophes en les précédant d'un caractère backslash. Cela permet à JavaScript de reconnaître que ces caractères font partie de la chaîne de caractères, et non pas de la syntaxe du langage.

Comprendre le positionnement des caractères

Une chaîne de caractères est constituée de n caractères où n représente le nombre de caractères. Chaque caractère dans une chaîne de caractères (string) est associé à une position numérique appelée « *index* ». L'index du premier caractère dans la chaîne est généralement 0, l'index du deuxième caractère est 1, l'index du troisième caractère est 2, et ainsi de suite. Dans une chaîne de caractères composée de n caractères, le dernier caractère est situé à la position n-1. Par exemple, si la chaîne de caractères contient vingt caractères, alors le dernier caractère est situé à la position 19. En effet, les chaînes de caractères sont construites comme des tableaux composées de différents caractères, avec un index pour chaque caractère.

Exemple **Comprendre le positionnement des caractères**

```
const str = "iuhzefpj";
console.log(str[0]); //i
console.log(str[3]); //z
```

En fin de compte, nous avons appris qu'une chaîne de caractères (string) est composée d'une séquence de caractères dont nous pouvons déterminer la longueur et la position individuelle. Il est important de veiller à bien échapper les caractères spéciaux afin d'éviter toute confusion pour le navigateur.

Distinction entre chaînes primitives et objets string en JavaScript

Il est essentiel de distinguer les objets string des valeurs primitives string. Les valeurs littérales entre guillemets simples, doubles ou backticks sont des chaînes primitives. Cependant, les objets string permettent d'envelopper des valeurs primitives de type string dans un objet. Comme nous l'avons vu précédemment, lorsque nous utilisons une propriété comme length avec une valeur primitive string, JavaScript crée automatiquement un objet temporaire string qui enveloppe la valeur primitive pour donner accès à la propriété appelée. Mais nous pouvons aussi directement créer un objet string qui enveloppe une chaîne de caractères primitive afin d'accéder constamment aux propriétés et méthodes de string. Pour cela, il va falloir appeler le constructeur string.

Exemple **Chaînes primitives et d'objets string**

```
let str1 = "Chaîne de caractères primitive entre doubles quotes"
let str2 = 'Chaîne de caractères primitive entre simples quotes';
let str3 = `Chaîne de caractères primitive entre backticks`;
let str4 = new String("Chaîne de caractères"); //objet String
```

Dans de nombreux cas, les valeurs primitives et les objets chaînes de caractères peuvent être utilisés de manière interchangeable. Les valeurs littérales pour les chaînes sont indiquées par des guillemets simples, doubles ou des backticks. Les backticks permettent de définir des littéraux de gabarits de chaînes pour interpoler des expressions à l'intérieur d'une chaîne.

Comprendre les tableaux en JavaScript

En JavaScript, les tableaux (array) sont une structure de données spéciale qui nous permet de stocker des collections ordonnées de valeurs. À la différence des chaînes, il n'existe pas de type primitif array. Les array sont des objets. Ils sont indispensables lorsque nous avons besoin de stocker une liste d'éléments, tels que des noms d'utilisateurs. Les tableaux nous permettent de conserver l'ordre des éléments et de les manipuler facilement.

Pour créer un tableau en JavaScript, nous pouvons utiliser des crochets et nous séparons les éléments par des virgules. Par exemple, si nous voulons stocker une liste de noms d'utilisateurs, nous pouvons écrire :

Exemple **Les éléments d'un tableau**

```
let utilisateurs = ['John', 'Peter', 'Chris'] ;
```

Dans cet exemple, la variable **utilisateurs** contient un tableau avec trois éléments : « *John* », « *Peter* » et « *Chris* ». Nous pouvons accéder aux éléments du tableau en utilisant leur position dans le tableau, également appelée « *index* ». Par exemple, pour accéder au deuxième élément (qui est « *Peter* »), nous pouvons utiliser l'index 1 :

```
console.log(utilisateurs[1]); // affiche "Peter"
```

Il est aussi possible d'appeler le constructeur array pour construire un tableau :

```
let utilisateurs = new Array('John', 'Peter', 'Chris');
```

Notre tableau est un objet qui contient plusieurs valeurs entre crochets séparées par des virgules. Les objets array peuvent être stockés via des variables. L'intérêt des array est que nous pouvons utiliser individuellement chacune de ces valeurs. Sans tableau, chaque valeur doit être attribuée à une variable, ce qui peut être fastidieux si nous en avons des centaines. De plus, nous pouvons utiliser une boucle pour parcourir plusieurs valeurs d'un tableau stockées dans une même variable.

Les tableaux seront donc utiles lorsque nous stockerons plusieurs données ordonnées via un même objet.

Propriétés et méthodes utiles pour manipuler des tableaux en JavaScript

Il est important de connaître ces méthodes de array, car elles nous permettent d'ajouter, de supprimer, de trier et de manipuler les éléments du tableau de manière efficace.

Parmi les méthodes les plus couramment utilisées, on peut citer **concat()**, **join()**, **pop()**, **push()**, **shift()**, **unshift()**, **slice()**, **splice()** et **reverse()**. Chacune de ces méthodes peut être utilisée pour résoudre des problèmes spécifiques en matière de manipulation de tableaux.

Pour avoir une idée de la syntaxe, voici comment procéder :

Méthode **La syntaxe pour utiliser une méthode**

```
let tableau = ['a', 'b', 'c'];
let autreTableau = tableau.push('d', 'e');
console.log(autreTableau);
console.log(tableau);
```

Ce code crée un tableau appelé « *tableau* » contenant les valeurs 'a', 'b' et 'c'. Ensuite, il crée un autre tableau appelé « *autreTableau* » en utilisant la méthode **push()** qui ajoute les valeurs 'd' et 'e' à la fin du tableau « *tableau* ». La méthode **push()** renvoie également la nouvelle longueur du tableau, qui est stockée dans la variable **autreTableau**.

Ensuite, le code utilise la fonction **console.log** pour afficher les deux tableaux dans la console du navigateur ou de l'éditeur de code. La première ligne de code affiche la nouvelle longueur du tableau « *autreTableau* » (qui est maintenant 5), tandis que la deuxième ligne affiche le tableau « *tableau* » mis à jour, qui contient maintenant les valeurs 'a', 'b', 'c', 'd' et 'e'.

Ce qu'il faut retenir

Nous avons examiné deux concepts de base en JavaScript : les chaînes de caractères (ou « *string* ») et les tableaux (ou « *array* »). Les chaînes de caractères et les tableaux ont des propriétés et des méthodes qui permettent de manipuler les données de manière efficace.

Les propriétés et méthodes associées aux chaînes de caractères et aux tableaux sont très utiles pour manipuler différents types de données. Les chaînes de caractères permettent de stocker des données simples, comme un mot ou un texte placé entre guillemets, tandis que les tableaux sont adaptés aux données plus complexes, telles qu'une collection ordonnée de chaînes de caractères. En utilisant les méthodes appropriées, nous pouvons facilement ajouter, supprimer, trier et manipuler ces données pour répondre à nos besoins en programmation.

B. Exercice : Quiz

[solution n°1 p.15]

Question 1

Quelle est la définition d'un string en JavaScript ?

- O Une chaîne de production
- O Une chaîne de caractères
- O Une instruction à exécuter

Question 2

À quoi sert la propriété length dans le contexte d'un objet String ?

- O Elle mesure la hauteur d'un objet de type string
- O Elle apporte de la valeur à une chaîne de caractères
- O Elle sert à retourner la longueur d'une chaîne de caractères

Question 3

Comment peut-on créer un array en JS ?

- O En utilisant le constructeur array
- O En utilisant des accolades
- O En utilisant uniquement des guillemets simples

Question 4

Qu'est-ce qu'un array ?

- O Un tableau
- O Un objet de type simple
- O Un fichier JavaScript

Question 5

Citez deux méthodes de array.

- O Length et input
- O Push et pop
- O List et object

III. Manipulation des chaînes de caractères (strings) et des tableaux (arrays)

A. Manipulation des chaînes de caractères (strings) et des tableaux (arrays)

Dans cette seconde partie, un accent particulier sera mis sur les manipulations des objets string et array. Pour cela, nous allons insister sur l'association variable et propriété ou méthode afin d'effectuer certaines actions avec nos données. Il est donc nécessaire d'en savoir davantage sur les propriétés et méthodes de deux types de données.

Les objets string sont utilisés pour représenter et manipuler des chaînes de caractères, qui sont utiles pour stocker des données sous forme de texte. Plusieurs opérations courantes sont disponibles pour manipuler les chaînes de caractères, telles que vérifier la longueur avec length, concaténer avec les opérateurs + et +=, rechercher des sous-chaînes avec les méthodes **includes()** ou **indexOf()**, et extraire des sous-chaînes avec la méthode **substring()**.

Nous allons examiner deux méthodes : la méthode **slice()** et la méthode **split()** pour respectivement extraire une sous-chaîne ou découper une chaîne en sous-chaînes.

Méthode **Extraction d'une sous-chaîne d'une chaîne et découpage d'une chaîne en sous-chaînes**

```
let historique = new String("(réf : 775533) commandé le 25/03/2020 10 sacs riz");
console.log(historique.slice(7, 13));
console.log(historique.split(""));
```

Nous stockons une chaîne dans un objet string. Puis, nous faisons un alert avec comme argument la méthode **slice(7, 13)** de notre objet historique. Il nous est renvoyé une sous-chaîne extraite entre les index 7 (compris) et 13 (non compris, soit 775533. Quant à la méthode **split("")**, rien qu'avec les simples quotes, elle va découper notre chaîne de caractères en sous-chaînes et tout mettre dans un tableau de 49 caractères. Elle considère le caractère passé comme caractère séparateur.

Méthode **D'autres tests de manipulation de l'objet string**

Nous allons nous entraîner à concaténer et à manipuler un string avec les méthodes courantes telles que **charAt()** ou **indexof()**.

```
let concat = 'Hello ' + ' World !';
console.log(concat) ;
let nom = "Paul";
let age = 44;
console.log('Je m\'appelle ' + nom + ', et j\'ai ' + age + ' ans.');
```

Dans ces deux exemples, nous avons utilisé l'opérateur « + » pour concaténer nos chaînes de caractères.

Utilisons maintenant la méthode **charAt()** pour comprendre son fonctionnement. La méthode **charAt()** permet de renvoyer le caractère d'une chaîne pour l'index passé comme argument.

```
let chaine = new String('Comment ça fonctionne ?');
let index = 4;
console.log(`Le caractère à l'index ${index} est ${chaine.charAt(index)}`);
```

Nous avons fait un console.log et entre parenthèses nous avons utilisé entre les backticks, une chaîne de caractères et l'interpolation pour insérer les variables. Nous écrivons le symbole dollar avec des accolades, **${}** à l'intérieur desquels nous plaçons nos expressions. La première fois, nous avons simplement interpolé la variable index. Mais la deuxième fois, nous interpolons la valeur de retour de la méthode **charAt()** de l'objet chaîne puis nous passons la variable index comme argument. Au final, nous est renvoyée la phrase : le caractère à l'index 4 est e. La manipulation a réussi. Parlons maintenant de la méthode **indexOf()**.

```
let chaine1 = 'Le renard brun et rapide saute par-dessus le chien endormi.';
let chaine2 = 'chien';
let chaine3 = chaine1.indexOf(chaine2) ;
console.log(`l'index du ${chaine2} à partir du commencement est ${chaine3}`);
```

L'**indexOf()** nous aide à trouver la position d'une sous-chaîne dans une chaîne de caractères. Dans cet exemple, nous ne créons pas d'objet string, mais c'est JavaScript qui créera un objet string temporaire lors de l'appel de la méthode **indexOf()**.

Notre première variable chaine1 contient une chaîne de caractères entre des guillemets. La deuxième variable chaine2 stocke une autre chaîne contenant « *chien* ». Quel est l'index du mot chien dans la première variable ? Nous avons créé une troisième variable chaine3 et l'avons défini sur la valeur de retour de chaine1.indexOf(chaine2). Pour terminer par un console.log, entre parenthèses nous avons mis le backticks du texte avec l'interpolation de chaine2 puis une autre interpolation avec chaine3. Voilà ce que cela a donné : l'index du 'chien' à partir du commencement est 45. La méthode **indexOf()** permet donc de connaître l'index d'une sous-chaîne dans une chaîne de caractères.

La manipulation d'un tableau par ses propriétés et méthodes

Pour connaître le nombre d'éléments d'un tableau, tout comme pour une chaîne de caractères, nous utilisons la propriété **length**. Cette propriété est de type entier et nous indique le nombre d'éléments présents dans un tableau.

Les développeurs ont souvent recours aux boucles pour parcourir les éléments d'un tableau. Par exemple, la boucle **for** est très courante. Ainsi, si la propriété **length** nous aide à déterminer la taille d'un tableau, la boucle **for** est un outil plus pratique pour récupérer la liste de tous les éléments du tableau.

L'exemple ci-dessus nous sert d'illustration :

Exemple **La propriété length est utilisée dans la boucle for**

```
let chiffre = ['un', 'deux', 'trois', 'quatre', 'quinze'];
for (let element of chiffre) {
  console.log(element);
}
```

Nous utilisons le mot clé « *of* » pour parcourir chaque valeur du tableau.

Méthode **L'ajout et la suppression d'un élément par les méthodes push() et pop()**

Pour ajouter ou supprimer un élément, nous pouvons utiliser **push()** et **pop()**. Pour cela, nous allons créer un tableau et lui appliquer ces deux méthodes.

```
let monTableau = ['Paris', 'Marseille', 'Lyon', 'Bordeaux', 'Toulouse'];
monTableau.push('Lille', 'Strasbourg');
monTableau.pop();
console.log (monTableau);
```

Nous pouvons voir que **push()** permet d'ajouter des éléments tandis que **pop()** supprime le dernier élément.

Méthode **Utiliser les méthodes shift() et unshift()**

```
let nombres = [1, 2, 3, 4];
nombres.shift();
console.log(nombres); //2,3,4
nombres.unshift(0);
console.log(nombres);//0,2,3,4
```

Le principe est assez simple, **unshift()** permet de rajouter un élément au début du tableau tandis que **shift()** supprime le premier élément du tableau.

Méthode **Autres manipulations possibles avec les méthodes split(), length, toString()**

Il existe de nombreuses autres méthodes pour manipuler les tableaux en JS. En voici quelques-unes :

```
// Crée une chaîne de caractères contenant les noms des villes séparées par des virgules
let mesDonnees = 'Paris, Londres, Berlin, Rome, Madrid';

// Utilise la méthode split() pour transformer la chaîne en un tableau en se basant sur le séparateur (la virgule suivie d'un espace)
let monTableau = mesDonnees.split(', ');

// Affiche la longueur du tableau (le nombre d'éléments)
console.log(monTableau.length);

// Affiche la chaîne de caractères initiale (mesDonnees)
console.log(mesDonnees);

// Affiche le tableau créé à partir de la chaîne de caractères (monTableau)
console.log(monTableau);

// Affiche le premier élément du tableau (l'index 0)
console.log(monTableau[0]);

// Affiche le dernier élément du tableau (l'index égal à la longueur du tableau moins 1)
console.log(monTableau[monTableau.length - 1]);

// Utilise la méthode join() pour fusionner les éléments du tableau en une nouvelle chaîne de caractères (myNewString)
let myNewString = monTableau.join(', ');
console.log(myNewString);

// Crée un tableau contenant les noms des joueurs
let nomsDesJoueurs = ["Neymar", "Kylian", "Cristiano", "Lionel"];

// Utilise la méthode toString() pour convertir le tableau des noms des joueurs en une chaîne de caractères
console.log(nomsDesJoueurs.toString());
```

Complément **Tableau multidimensionnel**

Un tableau multidimensionnel est un tableau qui, comme son nom l'indique, a plusieurs dimensions, par exemple, l'axe des colonnes et l'axe des lignes. Créons un tableau multidimensionnel.

```
let voitures = [["Peugeot", "208"], ["Renault", "Clio"]];
console.log(voitures[1][0]); //"Renault"
```

Dans cet exemple, nous créons un tableau à deux dimensions. Les crochets imbriqués représentent des lignes. Nous avons dans cet exemple deux lignes et deux colonnes. Nous cherchons à accéder à l'index 1 de notre tableau, qui correspond à ["Renault", "Clio"]. Puis nous cherchons à récupérer l'index 0 de ["Renault", "Clio"], qui est donc la valeur « *Renault* ».

B. Exercice : Quiz

[solution n°2 p.16]

Question 1

Comment savoir si une donnée est de type string ou de type objet ?

- Il faut utiliser des backticks
- Si sa valeur est une chaîne, c'est un type string. Si le constructeur string est appelé, c'est un objet
- Si sa valeur est entre simples guillemets et entre double guillemets

Question 2

Quels sont les deux cas pour utiliser length avec une valeur primitive string ?

- Soit l'utiliser directement avec une variable définie sur une chaîne, soit avec une valeur primitive directement
- Soit l'utiliser comme objet à stocker soit comme conteneur
- Pour le renvoi de la largeur et de la longueur d'une chaîne

Question 3

Pourquoi utilise-t-on la méthode slice() ?

- Elle est utilisée pour extraire une sous-chaîne d'une chaîne
- Elle est utilisée pour ajouter un caractère dans une chaîne
- Elle est utilisée pour supprimer un caractère à la fin de la chaîne

Question 4

Pourquoi utilise-t-on les méthodes push() et pop() ?

- push() est utilisé pour ajouter et pop() pour supprimer un élément en fin de tableau
- Parce qu'ils font partie d'un objet de type liste
- push() et pop() sont utilisés respectivement pour ajouter et supprimer un élément en début de tableau

Question 5

Quelle est la différence entre les méthodes shift() et unshift() d'une part et push() et pop() d'autre part ?

- shift() ajoute et unshift() supprime un élément en début de tableau. push() et pop() sont utilisés respectivement pour ajouter et supprimer un élément en fin de tableau
- La différence est que les unes agissent au début d'un tableau, et les autres à la fin
- Il n'y a pas de différence

IV. Essentiel

Nous avons abordé les notions de chaîne de caractères et de tableau, ainsi que leurs propriétés et méthodes qui offrent diverses possibilités de manipulation en JavaScript. String signifie « *chaîne* », il représente donc une chaîne de caractères. Array signifie tableau. Au-delà de ces simples définitions, nous avons cherché à savoir ce que veulent dire ces deux concepts, ce qu'ils impliquent, et surtout comment les manipuler. Nous avons par ailleurs compris la différence entre une valeur primitive string, et un objet string. Un objet string, créé via le constructeur string permet d'envelopper une chaîne de caractères primitives et de lui donner accès à des propriétés et méthodes. Array n'est pas un type primitif, c'est un objet.

Nous avons examiné la propriété length aussi bien pour le string que pour l'array. Sans les avoir toutes manipulées de façon exhaustive, nous avons vu par exemple les méthodes indexof(), slice(), split() pour les objets string. Et, pareillement, nous avons abordé aussi quelques méthodes des tableaux : split(), push(), pop(), shift(), unshift(), etc.

Les objets string et array sont donc fondamentaux en JavaScript. Vous pourrez les utiliser à de nombreuses reprises dans votre parcours de développeur.

V. Auto-évaluation

A. Exercice

Question 1

[solution n°3 p.17]

En quelques mots, expliquer les différentes façons de définir des chaînes de caractères et pourquoi les manipuler ?

Question 2

[solution n°4 p.17]

Pourquoi utiliser un tableau et expliquer comment le manipuler avec ses méthodes et propriétés ?

B. Test

Exercice 1 : Quiz

[solution n°5 p.17]

Question 1

Qu'est-ce qu'un objet string ?

- O Un objet qui enveloppe une valeur primitive string
- O Un objet qui donne accès à des outils de gestion de formulaire
- O Une valeur primitive de type chaîne de caractères

Question 2

Qu'est-ce que indexOf() ?

- O C'est une propriété de type « *valeur* » de l'objet string
- O C'est une méthode de l'objet string
- O Une fonction permettant d'indexer un élément de l'objet string

Question 3

Comment peut-on définir un tableau ?

- O En définissant une variable sur une liste de données comprises entre crochets et séparées par des virgules
- O En définissant une variable sur une liste de données comprises entre accolades et séparées par des virgules
- O En définissant une variable sur une liste de données comprises entre crochets et séparées par des points virgules

Question 4

Peut-on stocker n'importe quel élément dans un tableau ?

- O Non, ce n'est pas possible
- O On ne peut stocker que des number et des string
- O On peut stocker de nombreux types de données avec les tableaux

Question 5

Qu'est-ce que les développeurs en JavaScript utilisent pour passer en revue tous les éléments d’un tableau ?

- Ils comptent tous les éléments d’un tableau
- Les développeurs mettent la virgule après chaque élément d’un tableau
- Ils utilisent la boucle for

Solutions des exercices

Exercice p. 7 **Solution n°1**

Question 1

Quelle est la définition d'un string en JavaScript ?

- ○ Une chaîne de production
- ◉ Une chaîne de caractères
- ○ Une instruction à exécuter

Le mot clé « *string* » désigne les chaînes de caractères.

Question 2

À quoi sert la propriété length dans le contexte d'un objet String ?

- ○ Elle mesure la hauteur d'un objet de type string
- ○ Elle apporte de la valeur à une chaîne de caractères
- ◉ Elle sert à retourner la longueur d'une chaîne de caractères

Length est la propriété d'une chaîne de caractères qui permet de retourner sa longueur. Cette propriété est également utilisée pour les objets array.

Question 3

Comment peut-on créer un array en JS ?

- ◉ En utilisant le constructeur array
- ○ En utilisant des accolades
- ○ En utilisant uniquement des guillemets simples

Il est possible de créer un objet array en appelant le constructeur array suivi du mot clé « *new* ». Mais il est aussi possible de créer un array en définissant une variable sur un tableau entre crochets. Le constructeur est alors appelé en arrière-plan.

Question 4

Qu'est-ce qu'un array ?

- ◉ Un tableau
- ○ Un objet de type simple
- ○ Un fichier JavaScript

Les données composées avec plusieurs objets, une liste des valeurs comme les noms des clients, ne peuvent être décrites que par un type composé : un tableau (ou array). Un objet de type simple comme le string ne sera pas adapté pour une collection de valeurs.

Question 5

Citez deux méthodes de array.

- ○ Length et input
- ◉ Push et pop
- ○ List et object

Un array a plusieurs méthodes. Par exemple, pour ajouter ou supprimer un ou plusieurs éléments, nous pouvons utiliser respectivement push et pop.

Exercice p. 10 **Solution n°2**

Question 1

Comment savoir si une donnée est de type string ou de type objet ?

- ○ Il faut utiliser des backticks
- ◉ Si sa valeur est une chaîne, c'est un type string. Si le constructeur string est appelé, c'est un objet
- ○ Si sa valeur est entre simples guillemets et entre double guillemets

Une chaîne de caractères primitive est de type string. Par contre, en utilisant le constructeur string, elle est un objet. Nous pouvons aussi utiliser typeof de variable pour s'en apercevoir.

Question 2

Quels sont les deux cas pour utiliser length avec une valeur primitive string ?

- ◉ Soit l'utiliser directement avec une variable définie sur une chaîne, soit avec une valeur primitive directement
- ○ Soit l'utiliser comme objet à stocker soit comme conteneur
- ○ Pour le renvoi de la largeur et de la longueur d'une chaîne

Nous pouvons soit appeler length via une valeur définie sur une chaîne primitive, soit directement via une chaîne primitive. Dans tous les cas, JavaScript construira un objet string temporaire en arrière-plan pour accéder à la propriété length.

Question 3

Pourquoi utilise-t-on la méthode slice() ?

- ◉ Elle est utilisée pour extraire une sous-chaîne d'une chaîne
- ○ Elle est utilisée pour ajouter un caractère dans une chaîne
- ○ Elle est utilisée pour supprimer un caractère à la fin de la chaîne

Par exemple, dans une chaîne à sept éléments, nous pouvons en extraire le premier élément.

Question 4

Pourquoi utilise-t-on les méthodes push() et pop() ?

- ◉ push() est utilisé pour ajouter et pop() pour supprimer un élément en fin de tableau
- ○ Parce qu'ils font partie d'un objet de type liste
- ○ push() et pop() sont utilisés respectivement pour ajouter et supprimer un élément en début de tableau

Avec la méthode push(), nous pouvons ajouter un ou plusieurs éléments passés comme argument en fin de tableau. Et avec pop(), nous supprimons un ou plusieurs éléments en fin de tableau.

Question 5

Quelle est la différence entre les méthodes shift() et unshift() d'une part et push() et pop() d'autre part ?

- ○ shift() ajoute et unshift() supprime un élément en début de tableau. push() et pop() sont utilisés respectivement pour ajouter et supprimer un élément en fin de tableau
- ◉ La différence est que les unes agissent au début d'un tableau, et les autres à la fin
- ○ Il n'y a pas de différence

Nous avons le choix, si nous voulons ajouter et supprimer un ou plusieurs éléments en fin de tableau, nous utilisons push() et pop(), sinon si nous faisons pareil en début de tableau, nous utilisons shift() et unshift().

p. 12 Solution n°3

Il y a trois façons pour les variables de type string : la chaîne de caractères délimitée par des simples quotes ou par des doubles quotes ou enfin par des backticks si la chaîne est trop longue. Il faut utiliser l'antislash pour échapper certains caractères comme l'apostrophe sauf avec les backticks.

p. 12 Solution n°4

Vous stockez un nom dans une variable, donnée simple, vous utilisez un string. Ce string ne suffira plus si vous avez une liste de données à stocker, d'une manière ordonnée. Cette fois, vous aurez besoin d'un tableau. Combien de noms comporte votre liste ? Il faut utiliser la propriété length et si vous voulez par exemple supprimer certains noms ou en ajouter d'autres, il faut utiliser les méthodes comme push(), pop(), shift(), unshift(), etc.

Exercice p. 12 Solution n°5

Question 1

Qu'est-ce qu'un objet string ?

- ◉ Un objet qui enveloppe une valeur primitive string
- ○ Un objet qui donne accès à des outils de gestion de formulaire
- ○ Une valeur primitive de type chaîne de caractères

Un objet string est un objet qui enveloppe une valeur primitive string, afin de donner accès à des propriétés et méthodes.

Question 2

Qu'est-ce que indexOf() ?

- ○ C'est une propriété de type « *valeur* » de l'objet string
- ◉ C'est une méthode de l'objet string
- ○ Une fonction permettant d'indexer un élément de l'objet string

indexOf() est une méthode de l'objet string. Elle permet de rechercher la position de la première occurrence d'une sous-chaîne dans une chaîne.

Question 3

Comment peut-on définir un tableau ?

- [x] En définissant une variable sur une liste de données comprises entre crochets et séparées par des virgules
- [ ] En définissant une variable sur une liste de données comprises entre accolades et séparées par des virgules
- [ ] En définissant une variable sur une liste de données comprises entre crochets et séparées par des points virgules

Nous pouvons définir un Array en définissant une variable sur une liste de données comprises entre crochets et séparées par des virgules. Il est aussi possible d'appeler explicitement le constructeur array avec la syntaxe new Array(). Dans tous les cas, le constructeur est appelé, que ce soit en arrière-plan ou explicitement.

Question 4

Peut-on stocker n'importe quel élément dans un tableau ?

- [ ] Non, ce n'est pas possible
- [ ] On ne peut stocker que des number et des string
- [x] On peut stocker de nombreux types de données avec les tableaux

Les tableaux permettent de stocker différents types de données, que ce soient des données primitives ou des objets.

Question 5

Qu'est-ce que les développeurs en JavaScript utilisent pour passer en revue tous les éléments d'un tableau ?

- [ ] Ils comptent tous les éléments d'un tableau
- [ ] Les développeurs mettent la virgule après chaque élément d'un tableau
- [x] Ils utilisent la boucle for

Les développeurs utilisent souvent la boucle for/of pour parcourir les valeurs d'un tableau.