codecademy

introduction

`console.log()`

La `console.log()` méthode est utilisée pour enregistrer ou imprimer des messages sur la console. Il peut également être utilisé pour imprimer des objets et d'autres informations.

```
console.log('Hi there!');
// Prints: Hi there!
```

Javascript

JavaScript est un langage de programmation qui alimente le comportement dynamique de la plupart des sites Web. Avec HTML et CSS, c'est une technologie de base qui fait fonctionner le Web.

Méthodes

Les méthodes renvoient des informations sur un objet et sont appelées en ajoutant une instance avec un point `.`, le nom de la méthode et des parenthèses.

```
// Returns a number between 0 and 1
Math.random();
```

Bibliothèques

Les bibliothèques contiennent des méthodes qui peuvent être appelées en ajoutant au nom de la bibliothèque un point `.`, le nom de la méthode et un ensemble de parenthèses.

```
Math.random();
// ☝ Math is the library
```

Nombres

Les nombres sont un type de données primitif. Ils incluent l'ensemble de tous les entiers et nombres à virgule flottante.

```
let amount = 6;
let price = 4.99;
```

Chaîne de caractères `.length`

La `.length` propriété d'une chaîne renvoie le nombre de caractères qui composent la chaîne.

```
let message = 'good nite';
console.log(message.length);
// Prints: 9

console.log('howdy'.length);
// Prints: 5
```

Instances de données

Lorsqu'une nouvelle donnée est introduite dans un programme JavaScript, le programme en garde la trace dans une instance de ce type de données. Une instance est un cas individuel d'un type de données.

Booléens

Les booléens sont un type de données primitif. Ils peuvent être `true` ou `false` .

```
let lateToWork = true;
```

`Math.random()`

La `Math.random()` fonction renvoie un nombre aléatoire à virgule flottante compris entre 0 (inclus) et 1 non compris.

```
console.log(Math.random());
// Prints: 0 - 0.9
```

`Math.floor()`

La `Math.floor()` fonction renvoie le plus grand entier inférieur ou égal au nombre donné.

```
console.log(Math.floor(5.95));
// Prints: 5
```

Commentaires sur une seule ligne

En JavaScript, les commentaires sur une seule ligne sont créés avec deux barres obliques consécutives `//` .

```
// This line will denote a comment
```

Nul

Null est un type de données primitif. Il représente l'absence intentionnelle de valeur. Dans le code, il est représenté par `null` .

```
let x = null;
```

Cordes

Les chaînes sont un type de données primitif. Il s'agit de tout groupement de caractères (lettres, espaces, chiffres ou symboles) entourés de guillemets simples `'` ou de guillemets doubles `"` .

```
let single = 'Wheres my bandit hat?';
let double = "Wheres my bandit hat?";
```

Opérateurs arithmétiques

JavaScript prend en charge les opérateurs arithmétiques pour :

- `+` une addition
- `-` soustraction
- `*` multiplication
- `/` division
- `%` modulo

```
// Addition
5 + 5
// Subtraction
10 - 5
// Multiplication
5 * 10
// Division
10 / 5
// Modulo
10 % 5
```

Commentaires multi-lignes

En JavaScript, les commentaires multi-lignes sont créés en entourant les lignes avec `/*` au début et `*/` à la fin. Les commentaires sont de bons moyens pour diverses raisons, telles que l'explication d'un bloc de code ou l'indication de quelques conseils, etc.

```
/*
The below configuration must be
changed before deployment.
*/

let baseUrl
= 'localhost/taxwebapp/country';
```

Reste / Opérateur Modulo

L'opérateur de reste, parfois appelé modulo, renvoie le nombre qui reste après la division du nombre de droite par le nombre de gauche autant de fois que possible.

```
// calculates # of weeks in a year,
rounds down to nearest integer
const weeksInYear = Math.floor(365/7);

// calcuates the number of days left over
after 365 is divded by 7
const daysLeftOver = 365 % 7 ;

console.log("A year has " + weeksInYear
+ " weeks and " + daysLeftOver + "
days");
```

Opérateurs d'affectation

Un opérateur d'affectation attribue une valeur à son opérande gauche en fonction de la valeur de son opérande droit. En voici quelques uns:

- `+=` affectation supplémentaire
- `-=` devoir de soustraction
- `*=` devoir de multiplication
- `/=` affectation de division

```
let number = 100;

// Both statements will add 10
number = number + 10;
number += 10;

console.log(number);
// Prints: 120
```

Interpolation de chaîne

L'interpolation de chaîne est le processus d'évaluation des littéraux de chaîne contenant un ou plusieurs espaces réservés (expressions, variables, etc.).
Il peut être effectué à l'aide de littéraux de modèle : `text ${expression} text` .

```
let age = 7;

// String concatenation
'Tommy is ' + age + ' years old.';

// String interpolation
`Tommy is ${age} years old.`;
```

variables

Les variables sont utilisées chaque fois qu'il est nécessaire de stocker une donnée. Une variable contient des données qui peuvent être utilisées ailleurs dans le programme. L'utilisation de variables garantit également la réutilisation du code puisqu'il peut être utilisé pour remplacer la même valeur à plusieurs endroits.

```
const currency = '$';
let userIncome = 85000;

console.log(currency + userIncome + ' is more than the average income.');
// Prints: $85000 is more than the average income.
```

Indéfini

`undefined` est une valeur JavaScript primitive qui représente l'absence de valeur définie. Les variables déclarées mais non initialisées à une valeur auront la valeur `undefined` .

```
var a;

console.log(a);
// Prints: undefined
```

Apprendre Javascript : Variables

Une variable est un conteneur de données stocké dans la mémoire de l'ordinateur. Il est référencé par un nom descriptif qu'un programmeur peut appeler pour lui attribuer une valeur spécifique et la récupérer.

```
// examples of variables
let name = "Tammy";
const found = false;
var age = 3;
console.log(name, found, age);
// Tammy, false, 3
```

Déclaration de variables

Pour déclarer une variable en JavaScript, n'importe lequel de ces trois mots clés peut être utilisé avec un nom de variable :

- `var` est utilisé dans les versions antérieures à ES6 de JavaScript.
- `let` est la meilleure façon de déclarer une variable lorsqu'elle peut être réaffectée.
- `const` est la meilleure façon de déclarer une variable avec une valeur constante.

```
var age;
let weight;
const numberOfFingers = 20;
```

Modèles littéraux

Les littéraux de modèle sont des chaînes qui autorisent les expressions incorporées, `${expression}` . Alors que les chaînes normales utilisent des guillemets simples `'` ou doubles `"` , les modèles littéraux utilisent à la place des backticks.

```
let name = "Codecademy";
console.log(`Hello, ${name}`);
// Prints: Hello, Codecademy

console.log(`Billy is ${6+8} years
old.`);
// Prints: Billy is 14 years old.
```

`let` Mot-clé

`let` crée une variable locale en JavaScript et peut être réaffectée. L'initialisation lors de la déclaration d'une `let` variable est facultative. Une `let` variable contiendra `undefined` si rien ne lui est assigné.

```
let count;
console.log(count); // Prints: undefined
count = 10;
console.log(count); // Prints: 10
```

`const` Mot-clé

Une variable constante peut être déclarée à l'aide du mot-clé `const` . Il doit avoir une affectation. Toute tentative de réaffectation d'une `const` variable entraînera une erreur d'exécution JavaScript.

```
const numberOfColumns = 4;
numberOfColumns = 8;
// TypeError: Assignment to constant
variable.
```

Concaténation de chaînes

En JavaScript, plusieurs chaînes peuvent être concaténées à l'aide de l' `+` opérateur. Dans l'exemple, plusieurs chaînes et variables contenant des valeurs de chaîne ont été concaténées. Après l'exécution du bloc de code, la `displayText` variable contiendra la chaîne concaténée.

```
let service = 'credit card';
let month = 'May 30th';
let displayText = 'Your ' + service  + '
bill is due on ' +  month + '.';

console.log(displayText);
// Prints: Your credit card bill is due
on May 30th.
```