Syntaxe Javascript

Les variables, les types (Number, String, Boolean, etc…)

Les variables

Déclaration de variables

```
var nom_de_la_variable;
var mavariable1, mavariable2, mavariable3;

let ma_let;
const MA_CONST;
```

Dans ce cas de figure, les variables n'ayant pas reçu de valeurs, elles sont dites `undefined` .

JavaScript est un langage particulièrement souple et il autorise, aussi, la déclaration des variables sans utiliser le mot clé var, cela s'appelle une **déclaration implicite.**

Affectation de variables

```
var mavariable = 10;
var somme = mavariable1 + mavariable2;
var reponse = prompt(" Quelle est la valeur de cette variable ? ");
```

Transfert de valeurs entre variables et conversion de type

Pour **donner la valeur d'une variable à une autre**, il suffit d'utiliser le signe égal pour transférer la valeur de la variable à droite du signe égal dans celle située à gauche du signe égal.

```
var mavariable2 = mavariable1;
```

Il est possible de **convertir une variable d'un type à un autre** (généralement de texte vers du numérique) grâce aux méthodes parseInt() (conversion de texte vers un nombre entier) et parseFloat() (conversion de texte vers un nombre décimal). La syntaxe est la suivante :

```
var variablenumerique = parseInt(variabletexte);
var variablenumerique = parseFloat(variabletexte);
```

À l'inverse, il est possible de **convertir une variable numérique en texte** par l'intermédiaire de la méthode toString() dont la syntaxe est la suivante :

```
Var variabletexte = variablenumerique.toString();
```

Règles de nommage et mots réservés

Il est préférable de nommer les variables **explicitement**, c'est-à-dire pour ce qu'elles représentent plutôt que de prendre des noms de variables totalement abstraites. Ainsi, vous pourrez les retrouver plus facilement dans le code lors du débogage. Il existe, cependant, une liste de mots qu'il est totalement interdit d'utiliser lors de la déclaration de vos variables, car réservés par JavaScript.

Les types

Type	primitif	special	exemple

Type	primitif	special	exemple
Number	x	-	12
String	x	-	“bonjour”
Boolean	x	-	true / false
Null	x	x	null
Undefined	x	x	-
Object	-	-	

```
var nombre = 12;
var chaine = "bonjour";
var booleen = true;
var x = null;
var u;
var objet = {};
```

Tableaux, boucles et tests

Tableaux

MDN - Array (https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array)

```
var fruits = ['Apple', 'Banana'];

var pomme = fruits[0];

fruits.forEach(function(item, index, array) {
  console.log(item, index);
});

var length;
length = fruits.unshift('Strawberry');
length = fruits.push('Orange');

var last = fruits.pop();
var first = fruits.shift();

var position = fruits.indexOf('Banana');
var removedItem = fruits.splice(position, 1);

var copy = fruits.slice();
```

Les instructions de répétitions (boucles)

- for

```
for (i = 0; i < 5; i++) {
    console.log("i est égal à " + i);
}
```

- for…in

```
for (var propriete in document) {
    console.log(propriete, document[propriete]);
}
```

- while / do while

```
while (i < 10) {
    console.log("i est égal à " + i);
    i++;
}
```

```
do {
    console.log("i est égal à " + i);
    i++;
}
while (i < 10);
```

Les instructions conditionnelles (tests)

- if / else / else if

```
var n = 3;
if(n === 1){
    alert("if");
} else if (n === 2) {
    alert("else if");
} else {
    alert("else");
}
```

- switch

```
var n = 3;
switch(n){
    case 1:
        alert("1");
        break;
    case 2:
        alert("2");
        break;
    default:
        alert("default");
        break;
}
```

- opérateur conditionnel ternaire

```
var n = 3;
alert(n === 1 ? "true" : "false");
```

Interrompre et quitter les boucles

- `break`
- `continue`

Les opérateurs arithmétiques, de comparaison et logiques

MDN - Expressions et Opérateurs (https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Expressions_et_Op%C3%A9rateurs) :

- Les opérateurs d'affectation
- Les opérateurs de comparaison
- Les opérateurs arithmétiques

- Les opérateurs binaires
- Les opérateurs logiques
- Les opérateurs de chaînes
- L'opérateur (ternaire) conditionnel
- L'opérateur virgule
- Les opérateurs unaires
- Les opérateurs relationnels

Travaux pratiques : réalisation d'exemples simples

Gestion des erreurs et des exceptions, exemples de mise en oeuvre des instructions 'try', 'catch', 'throw', 'finally'

```
try {
    alert("try");
    // throw 4245;
    throw new Error("ça ne marche pas");
}
catch(exception){
    alert("catch");
}
finally {
    alert("finally");
}
```

Utilisation de la console

openclassrooms - Déboguer votre code (https://openclassrooms.com/fr/courses/1916641-dynamisez-vos-sites-web-avec-javascript/2724891-deboguer-votre-code-partie-1-3)

Pour vérifier les valeurs de vos variables, calculs, etc., la fonction `alert()` vous viendra probablement à l'esprit, mais sachez qu'il existe une méthode spécialement adaptée à ce cas de figure, il s'agit de `console.log()`.

Méthodes et outils de debugging

openclassrooms - Déboguer votre code (https://openclassrooms.com/fr/courses/1916641-dynamisez-vos-sites-web-avec-javascript/2724891-deboguer-votre-code-partie-1-3)

Créer des scripts paraît facile au premier abord, mais on finit toujours par tomber sur le même problème : notre code ne fonctionne pas ! On peut alors dire qu'il y a un bug, en clair il y a une erreur dans le code qui fait qu'il s'exécute mal ou ne s'exécute tout simplement pas.

- Les kits de développement et leur console
- Utiliser les points d'arrêt

Présentation des fonctions globales et des classes natives

MDN : Objets globaux (https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux)

Définition des fonctions

MDN : Function (https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Function)

Gestion des arguments

MDN : Arguments (https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions/arguments)