Javascript
Partie 2

Javascript

Incrémentation et décrémentation

L'incrémentation permet d'ajouter une unité à un nombre au moyen d'une syntaxe courte. À l'inverse, la décrémentation permet de soustraire une unité.

```
<script>
var number = 0;
number++;
alert(number); // Affiche : « 1 »
number--;
alert(number); // Affiche : « 0 »
</script>
```

Javascript

La boucle while

Une boucle sert à répéter une série d'instructions. La répétition (ou itération) se fait jusqu'à ce qu'on dise à la boucle de s'arrêter.

Pour une boucle, on pose une condition, et la boucle se répète tant que la condition est vérifiée (true), selon la structure suivante :

```
<script>
  while (condition) {
        instruction_1; instruction_2; instruction_3;
  } </script>
```

Javascript

La boucle while

Quand la boucle s'arrête, les instructions qui suivent la boucle sont exécutées :

```
<script>
   var number = 1;
   while (number < 10) {
   number++; // Tant que le nombre est inférieur à 10, on l'incrémente de 1
   }
   alert(number); // Affiche : « 10 » </script>
```

Javascript

La boucle while

```
<script>
var prenoms = '', prenom; // On crée une variable prenoms pour mémoriser
while (true) {
prenom = prompt('Entrez un prénom :'); // L'utilisateur entre chaque prenom
if (prenom) {
prenoms += prenom + ' '; // Ajoute le nouveau prénom ainsi qu'une espace
} else {
break; // On quitte la boucle
}
} alert(prenoms); // Affiche les prénoms à la suite </script>
```

Javascript

La boucle for

```
<script>
  for (initialisation; condition; incrémentation) {
        instruction_1;
        instruction_2;
        instruction_3;
  } </script>
```

```
  <script>
  for (var iter = 1; iter <= 5; iter++) { // On initialise une variable, et tant qu'elle est inférieure ou égale à 5 on l'incrémente de 1.
  alert('Itération n°' + iter); // A chaque fois on affiche une boîte de dialogue (5 fois)
  } </script>
```

Javascript

Créer une fonction

```
<script>
function myFunction(arguments) { // Le terme "function" est obligatoire pour déclarer une fonction
// Le code que la fonction va devoir exécuter
} </script>
```

```
<script>
function byTwo() {
var result = parseInt(prompt('Donnez le nombre à multiplier par 2 :'));
alert(result * 2); }
byTwo(); // On appelle la fonction créée
alert('Vous en êtes à la moitié !'); // Puis un message intermédiaire
byTwo(); // Et appelle de nouveau la fonction </script>
```

Javascript

Les variables globales et locales

- toute variable déclarée dans une fonction n'est utilisable que dans cette même fonction.
- Ces variables spécifiques à une seule fonction ont un nom : les variables locales.
- Déclarées en dehors des fonction, on parle de variables globales.

Javascript

Les variables globales et locales

```
<script>
var message = 'Ici la variable globale !';
function showMsg() {
var message = 'Ici la variable locale !';
alert(message); }
showMsg(); // On affiche la variable locale
alert(message); // Puis la variable globale
</script>
```

Javascript

Les Arguments

- Pas obligatoire, l'argument peut être ainsi utilisé :

```
<script>
function myFunction(arg) { // Notre argument est la variable « arg »
alert('Votre argument : ' + arg); }
myFunction('En voilà un beau test !'); </script>
```

Javascript

Les Arguments

```
<script>
function myFunction(arg) {
alert('Votre argument : ' + arg); }
myFunction(prompt('Que souhaitez-vous passer en argument à la fonction ?'));
</script>
```

Javascript

Les Arguments

```
<script>
function moar(first, second) {
// On peut maintenant utiliser les variables « first » et « second » comme on le souhaite :
alert('Votre premier argument : ' + first);
alert('Votre deuxième argument : ' + second);
}
moar(
   prompt('Entrez votre premier argument :'),
   prompt('Entrez votre deuxième argument :')
); </script>
```

Javascript

Les valeurs de retour

- Une fonction peut retourner une seule valeur, stockée dans ue variable :

```
<script>
function sayHello() {
return 'Bonjour !'; // L'instruction « return » suivie d'une valeur, cette dernière est donc renvoyée par la fonction (il ne peut pas y en avoir d'autres)
}
alert(sayHello());
</script>
```

Les Objets

Les variables contiennent des objets, qui peuvent être des nombres, des chaînes de caractères ou des booléens. Mais le Javascript n'est pas un langage orienté objet (C++, C# ou Java), mais un langage orienté objet par prototype.

Les objets contiennent trois choses :

- un constructeur
- - des propriétés
- - des méthodes.

Javascript

Les Objets

```
<script>
var myString = 'Ceci est une chaîne de caractères'; // On crée un objet String
alert(myString.length); // On affiche le nombre de caractères, au moyen de la propriété « length »
alert(myString.toUpperCase()); // On récupère la chaîne en majuscules, avec la méthode toUpperCase(), l'inverse étant la méthode toLowerCase()
</script>
```

Javascript

Les Tableaux

- Après Number, String et Boolean, Array est un 4e objet natif de Javascript.
- Un tableau, ou plutôt un array en anglais, est une variable qui contient plusieurs valeurs, appelées items.
- Chaque item est accessible au moyen d'un indice (index en anglais) et dont la numérotation commence à partir de 0.

Javascript

Les Tableaux

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
// Le contenu se définit entre crochets, avec une virgule entre chaque valeur.
// La chaîne 'Rafael' correspond à l'indice 0, 'Mathilde' à l'indice 1...
alert(myArray[1]); // Affiche : « Laurence »
</script>
```

Javascript

Les Tableaux

Modifier une valeur :

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
myArray[1] = 'Paul';
alert(myArray[1]); // Affiche : « Paul »
</script>
```

Javascript

Les Tableaux

Ajouter une valeur :

```
<script>
var myArray = ['Rafael', 'Mathilde'];
myArray.push('Ahmed'); // Ajoute « Ahmed » à la fin du tableau
myArray.push('Jérôme', 'Guillaume'); // Ajoute « Jérôme » et « Guillaume » à
la fin du tableau
</script>
```

Javascript

Les Tableaux

Ajouter une valeur : (push)

```
<script>
var myArray = ['Rafael', 'Mathilde'];
myArray.push('Ahmed'); // Ajoute « Ahmed » à la fin du tableau
myArray.push('Jérôme', 'Guillaume'); // Ajoute « Jérôme » et « Guillaume » à
la fin du tableau
</script>
```

Javascript

Les Tableaux

Ajouter une valeur : (unshift)

La méthode unshift() fonctionne comme push(), excepté que les items sont ajoutés au début du tableau.

Retirer une valeur : (shift – pop)

Les méthodes shift() et pop() retirent respectivement le premier et le dernier élément du tableau.

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume'];
myArray.shift(); // Retire « Rafael »
myArray.pop(); // Retire « Guillaume »
alert(myArray); // Affiche « Mathilde,Ahmed,Jérôme »
</script>
```

Javascript

Les Tableaux

Découper une chaine en tableau (split)

```
<script>
var cousinsString = 'Jérôme Guillaume Paul',
cousinsArray = cousinsString.split(' '); // Avec les espaces, on a trois items
alert(cousinsString);
alert(cousinsArray);
</script>
```

Javascript

Les Tableaux

Découper un tableau en chaine (join)

```
<script>
var cousinsString_2 = cousinsArray.join('-');
alert(cousinsString_2); </script>
```

Javascript

Les Tableaux

Parcourir

```
<script>
var myArray = ['Rafael', 'Mathilde', 'Ahmed', 'Jérôme', 'Guillaume']; // La length est de 5
for (var i = 0, c = myArray.length; i < c; i++) { // On crée la variable c pour que la condition ne soit pas trop lourde en caractères
alert(myArray[i]); // On affiche chaque item, l'un après l'autre, jusqu'à la longueur totale du tableau
}
</script>
```

Javascript

Les Objets literaux

On peut remplacer l'indice par un identifiant. Dans ce cas on crée un objet

Javascript

Les Objets literaux

Les identifiants créés (self, sister...
sont des propriétés,
avec plusieurs possibilités
d'affichage
(ce qui convient à toutes
les propriétés,
également pour length
par exemple).

```
<script>
var family = {
self: 'Rafael',
sister: 'Mathilde',
brother: 'Ahmed',
cousin_1: 'Jérôme',
cousin_2: 'Guillaume'
};
var id = 'sister';
alert(family[id]); // Affiche : « Mathilde »
alert(family.brother); // Affiche : « Ahmed »
alert(family['self']); // Affiche : « Rafael »
family['uncle'] = 'Pauline'; // On ajoute une donnée, avec un identifiant.
family.aunt = 'Karim'; // On peut ajouter aussi de cette manière.
alert(family.uncle); </script>
```

Javascript

Parcourir un objet avec for in

- On ne peut pas parcourir l'objet avec for, parce for s'occupe d'incrémenter des variables numériques.
- on fournit une variable-clé pour le parcours

```
<script>
var family = {
self: 'Rafael',
sister: 'Mathilde',
brother: 'Ahmed',
cousin_1: 'Jérôme',
cousin_2: 'Guillaume'
};
for (var id in family) { // On stocke l'identifiant dans « id » pour parcourir l'objet « family »
alert(family[id]);
} </script>
```

Exercice

- Demandez les prénoms aux utilisateurs et stockez-les dans un tableau. Pensez à la méthode push().
- À la fin, il faudra afficher le contenu du tableau, avec alert(), seulement si le tableau contient des prénoms ; en effet, ça ne sert à rien de l'afficher s'il ne contient rien.
- Pour l'affichage, séparez chaque prénom par un espace. Si le tableau ne contient rien, faites-le savoir à l'utilisateur, toujours avec alert().

Javascript

Exercice : correction

Saisie des prénoms

```
<script>
var nicks = '', nick;
while (true) {
nick = prompt('Entrez un prénom :');
if (nick) {
nicks += nick + ' '; // Ajoute le nouveau prénom ainsi qu'un espace
} else {
break; // On quitte la boucle
}
}
alert(nicks); // Affiche les prénoms à la suite
</script>
```

Javascript

Fin de la présentation

Partie 2