


Le langage Javascript

Introduction

- Pour ce TP, nous nous assisterons de trois outils de **Firefox** accessibles sous le menu *Outils* et d'un document préparé :
 1. la page **jslab.html**, (depuis le répertoire **jslab** de l'archive) nous servira d'environnement de test. Nous utiliserons la machine virtuelle associée pour y exécuter nos scripts. Dans un premier temps, nous n'utiliserons pas son contenu.
 2. la **console d'erreur**, (**ctrl+maj+J**) : permet de lire les messages d'erreurs et d'avertissement.
 Lorsqu'une erreur survient en Javascript, aucune information n'est portée à la connaissance de l'utilisateur, le programme s'arrête simplement sans notification. Pour connaître la nature de l'anomalie, il faut consulter la console d'erreurs. La console d'erreur est commune à toutes les instances du navigateur, pour une meilleure lisibilité, il convient de l'effacer puis de recharger la page à tester,
 3. l'**ardoise Javascript**, (**maj+F4**) : permet de saisir du code et de l'exécuter (**ctrl+r**) dans le contexte de la page chargée (dans l'onglet actif de la fenêtre active). Cette ardoise vous permettra de faire quelques expérimentations sans avoir à écrire le code à tester dans le fichier. **Attention** : tout ce qui a été fait dans l'ardoise est perdu lors du rechargement de la page. Veillez donc à copier-coller vos résultats dans un fichier pour les consigner au fur et à mesure.
 4. la **console web**, (**ctrl+maj+K**) : permet de consulter les messages que nous imprimerons grâce à l'instruction `console.log(expression à imprimer)`;
- Une référence concise du langage JavaScript mise à disposition par la fondation Mozilla est disponible à cette adresse : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>
- Les aspects D.O.M. et Ajax feront l'objet des TP suivants.

Exercice 1 – Faisons connaissance avec l'environnement

Question 1.1 : Ouvrez une nouvelle fenêtre de navigateur, puis ouvrez l'ardoise Javascript (**maj+F4**).

- Tapez le code suivant dans l'ardoise et exécutez le (**ctrl+r**) :

```
1 function coucou() {  
2     alert('Coucou!');  
3 }  
4  
5 coucou();
```

Que se passe-t-il ?

- Effacez le contenu de l'ardoise et exécutez le code suivant :

```
1 coucou();
```

Que se passe-t-il ?

- Exécutez le code suivant :

```
1 alert(coucou);
```

Quel est le résultat de l'affichage ?

- Sans fermer l'ardoise Javascript, ouvrez une nouvelle fenêtre de navigateur, retournez dans l'ardoise Javascript et ré-exécutez les deux dernières instructions. Que se passe-t-il ?
- Sans fermer l'ardoise Javascript, fermez la fenêtre que vous venez d'ouvrir et retournez dans la fenêtre originale et exécutez à nouveau les deux instructions.
- Rechargez la page complètement dans le navigateur (F5), puis exécutez à nouveau les deux instructions. Expliquez ce que toutes ces observations mettent en évidence ?

Exercice 2 – *Un peu de syntaxe*

Ouvrez le document `jslab.html` dans votre navigateur, ouvrez la console web ainsi que l'ardoise JavaScript.

Question 2.1 : Copiez le contenu du fichier `fibonacci.js` dans votre ardoise. Selon-vous, que doit faire ce programme ? Exécutez-le. Est-il conforme ?

Question 2.2 : Corrigez le programme afin qu'il produise le résultat attendu. (Indication, il n'est nécessaire de renommer aucun symbole.)

Question 2.3 : Soit le tableau suivant :

```
1 var tab = ['ichi', 'ni', 'san', 'shi', 'go', 'roku', 'shichi', 'hachi', 'kyū', 'jū'];
```

Écrivez un programme qui parcourt le tableau à l'aide d'une boucle `for` (avec la syntaxe à la *foreach*) et d'une variable `i` et affiche chacune des valeurs du tableau dans la console web.

Question 2.4 : Modifiez ce programme afin qu'il affiche les indices des valeurs affichées en commençant à 1 (au lieu de 0). Avant cela, affichez le nom du type associé à la variable `i`.

Exercice 3 – *Les objets.*

Récupérez le contenu du répertoire `jslab`, et ouvrez le fichier `jslab.html`. celui-ci contient seulement un squelette de document avec quelques éléments pré-définis pour nos expériences. Pour toutes les questions de cet exercice, nous travaillerons dans l'ardoise JavaScript (maj+F4).

Question 3.1 : Écrivez une fonction `printObject()` qui affiche la structure d'un objet passé en paramètre avec par exemple un affichage de cette forme :

```
1 Object [nom objet] : [type objet] =
2   attribut1 : type1 = valeur1
3   attribut2 : type2 = valeur2
4   ...
5   attributn : typen = valeurn
```

Appliquez la fonction sur les objets prédéfinis `document`, `window` et sur une fonction.

Question 3.2 : En utilisant la notation JSON, créez et initialisez (avec les valeurs de votre choix) un objet contenant : une chaîne de caractères `nom`, un entier `age`, un booléen `majeur`, un objet `adresse` contenant un entier `numéro` et une chaîne `rue`, et enfin une fonction `travail` qui renvoie (toujours) `false`.

Affichez cet objet avec votre fonction `printObject()`.

Question 3.3 : Bonus : Écrivez une fonction `merge(a, b)` qui renvoie un nouvel objet issue de la fusion de deux objets `a` et `b`.

Exercice 4 – *Prototypes*

Question 4.1 : Nous allons étudier le comportement des prototypes :

- Écrivez un constructeur `rect(lon, lar)` qui initialise un objet avec une longueur et une largeur.
- Appelez `printObject()` sur ce constructeur puis sur `rect.prototype`.
- Instanciez un objet `r` et affichez le.
- Ajoutez un attribut `color` avec la valeur `blue` au prototype de `rect`.
- Instanciez un objet `r2` et affichez le.
- Afficher à nouveau `r`.

Question 4.2 : Modifier la fonction `printObject` afin qu'elle fasse préfixer par `PROTOTYPE`: les attributs appartenant au prototype de l'objet. (Recherchez `hasOwnProperty()`).

Question 4.3 : Ajoutez une méthode `repeat(x)` aux chaînes de caractères du langage afin de répéter leur contenu :

```
1 // Affiche "coucou" :  
2 console.log("cou".repeat(2));
```

Question 4.4 : Bonus :Ajoutez une méthode `shuffle()` aux tableaux du langage permettant de mélanger leur éléments :

```
1 var tab = [1, 2, 3, 4, 5, 6, 7, 8, 9];  
2 tab.shuffle();  
3 console.log(tab); // [2, 4, 1, 8, 9, 6, 5, 3, 7]
```

Question 4.5 : Bonus : création d'une structure complexe à base de prototypes :

1. Créez un constructeur
2. Créez un objet appelé `Shape` ayant une propriété appelée `Type` ainsi qu'une méthode `getType()`.
3. Définissez un constructeur `Triangle()` ayant pour prototype `Shape`. Les objets produits