

Module 2 (laboratoire)

exercices introductifs

Introduction

Ces notes rassemblent divers exercices visant à mettre en pratique les notions vues lors des exposés théoriques. Il est fort probable que vous n'ayez pas le temps de réaliser tous les exercices au cours des séances prévues. C'est quasiment voulu : pour répartir la charge de travail et contrebalancer les cours qui vous demandent de réaliser des travaux de grande taille vers la fin du quadrimestre, nous avons choisi de vous donner plus de boulot au début du quadrimestre et moins par la suite.

Au cas où vous ne termineriez pas les exercices avant la fin de la séance, nous vous conseillons de les achever chez vous au plus tôt et ce, avant la séance suivante, de manière à pouvoir poser des questions au sujet des éventuelles difficultés rencontrées. Ces exercices ont été prévus pour pouvoir être réalisés quasiment n'importe où (du moment que vous avez accès au navigateur Firefox et, dans certains cas, à Internet).

Note au sujet du navigateur Firefox

Malgré les nombreuses tentatives d'uniformisation et de création de standards pour le web, certains navigateurs continuent de se démarquer... Dans le cadre de ce cours, nous ne nous intéresserons qu'au navigateur Firefox, principalement parce que son utilisation est assez répandue et qu'il possède une documentation en ligne facile à consulter.

Toutefois, il faut garder à l'esprit que, dans le cadre de travaux professionnels, il est généralement nécessaire d'assurer la compatibilité avec toute une série de navigateurs (dont Internet Explorer, sans doute à la fois le plus commun et celui qui se démarque le plus des autres). Cela peut compliquer considérablement les scripts et programmes mais des informations sur la manière de procéder sont disponibles un peu partout sur la toile.

Javascript : aspects pratiques

Dans la plupart des exercices qui suivent, vous allez devoir créer des fichiers html, css et js. Pour tester ces derniers, il vous suffit de les placer dans un répertoire quelconque (sur une clef USB, sur votre partition, sur le disque dur de votre portable...) puis d'ouvrir le fichier html à l'aide de Firefox.

Pour éditer ces fichiers, nous vous conseillons d'utiliser Notepad++ (<http://notepad-plus-plus.org/>), un éditeur de texte plutôt universel avec mise en évidence de la syntaxe.

Deux raccourcis claviers à connaître !

Sous Firefox, vous pouvez ouvrir la console d'erreurs avec le raccourci Ctrl+Shift+J (ça peut se révéler pratique pour voir pourquoi le comportement que vous attendez ne se produit pas) et la console de développement avec le raccourci Ctrl+Shift+K. Cette dernière vous permet de rentrer directement des lignes Javascript pour des tests rapides.

Exercice 1 : prise en main de la console

Lancez Firefox puis utilisez le raccourci Ctrl-Shift-K. En bas du browser apparaît la console interactive. Celle-ci permet entre autres d'examiner les éléments HTML de la page web affichée mais également d'exécuter diverses instructions Javascript à la volée.

Dans la ligne tout en bas de la console, tapez les commandes suivantes une à une et examinez leur effet. (Assurez-vous que la case « Console » est bien cochée pour rendre visibles les messages envoyées à la console).

Conseil : prenez le temps de retaper les lignes plutôt que d'utiliser le copier/coller afin de mieux vous familiariser avec le langage.

Variables simples

```
let nbJoursSem = 7;
```

La ligne précédente initialise une variable « nbJoursSem » à 7. La console répond « undefined » car vous ne lui avez pas demandé d'effectuer un calcul : vous lui avez juste ordonné de placer la valeur 7 dans une nouvelle variable appelée « nbJoursSem ».

```
alert(nbJoursSem);
```

Une fenêtre s'ouvre, mais ici encore, la console vous répond « undefined » car l'instruction en question est un appel de fonction (alert) qui ne renvoie aucune valeur.

Voici comment obtenir une autre réponse que « undefined » : la console vous permet également de connaître la valeur d'une variable. Entrez maintenant la ligne

```
nbJoursSem
```

pour afficher la valeur de cette variable.

Poursuivez avec les lignes suivantes.

```
console.log("Par semaine : " + (nbJoursSem * 24) + " heures.");  
let age = prompt("Quel est votre âge ?");  
alert("Âge = " + age);
```

Notez la différence entre alert et console.log !

Définition d'une fonction

```
function nbJours (nbAns) { return nbAns * 365; }
```

La console vous permet de connaître la valeur d'une fonction pour un argument donné en entrant

```
nbJours(1);  
nbJours(10);
```

Poursuivez avec les lignes suivantes.

```
let nb = 4;  
alert("Dans " + nb + " ans, il y a " + nbJours(nb) + " jours.");
```

Pour rentrer dans la console du texte qui tient sur plusieurs lignes, utilisez « Shift + Enter » pour passer à la ligne sans exécuter/évaluer le code tapé.

```
function entre (nbAns) {  
    var nbJ1 = nbJours(nbAns);  
    var nbJ2 = nbJours(nbAns + 1);  
    return "entre " + nbJ1 + " et " + nbJ2 + " jours";  
}  
  
entre(1);  
  
entre(10);
```

Quelques tests de plus.

Vous avez vu plus haut que la console vous permet de connaître la valeur d'une variable en entrant tout simplement son nom. Par exemple :


```
nbJoursSem;
```

pour afficher la valeur de cette variable. Observez le résultat lorsque vous entrez

```
entre;
```

Testez la fonction « entre » en utilisant l'âge que vous avez entré plus haut.

```
console.log("Vous avez " + entre(age) + " !");
```

Observez la dernière ligne affichée par la console : les nombres de jours sont-ils corrects ?
Pouvez-vous déterminer d'où vient « l'erreur » ? 

Exercice 2 : Première page HTML-dynamique

Le point de départ de cet exercice est le document suivant.

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8"/>  
  </head>  
  <body>  
    <p><strong>Javascript</strong> est un langage de scripts compris  
    par la plupart des navigateurs web. Ce sont ces derniers qui  
    exécutent le code Javascript.</p>  
    <p>Grâce à Javascript, les pages web deviennent</p>  
    <ul>  
      <li>interactives,</li>  
      <li>plus intéressantes,</li>  
      <li>plus <span id='dyna'>dynamiques</span> !</li>  
    </ul>  
    <p>Javascript est un standard géré par la  
    <a href=' http://www.ecma-international.org/'>ECMA</a>.</p>  
  </body>  
</html>
```

Étape 1 : création du fichier HTML

Créez le fichier HTML, observez son contenu pour vous rappeler la syntaxe du HTML et observez son apparence dans le navigateur.

Étape 2 : script pour ajouter un titre

Éditez le fichier HTML pour ajouter du code Javascript au sein de l'élément `<body>`, et plus précisément juste avant le dernier paragraphe. Dans ce code,

1. utilisez « `alert` » pour afficher le message « Cliquez sur Ok. » ;
2. utilisez « `document.write` » pour insérer dans le document HTML le code suivant (qui servira de titre placé juste avant le dernier paragraphe parlant de la ECMA).

```
|| <h2>Les origines de Javascript</h2>
```

Rechargez la page HTML pour voir les effets du changement (idem dans les étapes suivantes).

Étape 3 : script pour ajouter l'heure

Notez que, pour ajouter un titre, il vaut mieux simplement l'écrire en HTML. L'utilisation de Javascript est plutôt inutile ici. Par contre, Javascript est plus utile si l'idée est d'ajouter un message qui dépend de certaines circonstances.

Le but de cet étape est d'ajouter un paragraphe en début de document qui indiquera « Bonjour, il est 14h37. » Pour ce faire, ajoutez une nouvelle balise `<script>` tout au début de l'élément `<body>` et utilisez à nouveau `document.write`, mais avec un message dépendant de l'heure.


Pour obtenir l'heure, vous pouvez recopier le code suivant.

```
|| let maintenant = new Date ();  
|| let h = maintenant.getHours();  
|| let m = maintenant.getMinutes();
```

Étape 4 : fonction changeDyn

Dans l'élément `<head>` du fichier HTML, ajoutez une nouvelle balise `<script>` et du code Javascript pour déclarer une fonction nommée `changeDyn`, avec un argument appelé `txt` en recopiant le code suivant.

```
|| function changeDyn (txt) {  
||   let cible = document.getElementById("dyna");  
||   cible.innerHTML = txt;  
|| }
```

Rechargez la page HTML... rien n'a changé ? Pourquoi ? 

Ouvrez la console (Ctrl + Shift + K) et entrez la ligne suivante et observez son effet.

```
|| changeDyn("BELLES");
```

L'expression « `document.getElementById("dyna")` » permet de retrouver l'élément HTML dont l'identificateur (Id) est « `dyna` ». La syntaxe « `cible.innerHTML` » permet de modifier

le contenu d'un élément HTML. Ici, le but est de remplacer le contenu du « span » identifié par « dyna » par le texte passé en argument.

Notez que le code de la fonction changeDyn aurait pu être écrit en une seule ligne.

```
function changeDyn (txt) {  
    document.getElementById("dyna").innerHTML = txt;  
}
```

Étape 5 : programmation événementielle

Dans le fichier HTML, repérez l'élément identifié par « dyna ». Ajoutez un attribut à la balise , à savoir :

```
onclick = "changeDyn( 'DYNAMIQUE' );"
```

Rechargez la page HTML... et testez les effets de la dernière modification en cliquant sur l'élément HTML correspondant au .

Pour rappel, cette utilisation de Javascript est de la programmation événementielle : le code donné dans l'attribut onclick s'exécute lorsqu'un événement se produit. L'événement en question est un clic sur le .

Étape 6 : fonction toggleDyn

Revenez dans la balise <head> et déclarez-y (avant la définition de fonction) une variable booléenne appelée dynMinuscule et initialisée à true (ce qui signifie que le mot « dynamique » est actuellement écrit en minuscules).

```
let dynMinuscule = true;
```

Dans la balise , remplacez le code associé à l'événement onclick par le suivant.

```
onclick = "toggleDyn();"
```

Ensuite, dans la balise <head>, définissez une fonction appelée toggleDyn faisant en sorte qu'à chaque clic sur le mot « dynamiques », celui-ci change entre minuscules et majuscules. Utilisez la variable booléenne pour retenir l'état actuel du mot (minuscule ou majuscule) et savoir s'il faut remplacer le texte par « dynamiques » ou par « DYNAMIQUES ».

Deux notes « clean code » :

- Faites bien sûr appel à la fonction changeDyn !
- Si vous écrivez du code ressemblant à « if (variable == true) », considérez-vous comme privé de dessert et de guindailles pendant deux semaines !

Exercice 3 : révision des tables de multiplication

Dans cet exercice, vous devrez tout créer (le fichier HTML, éventuellement une feuille de styles CSS et le code Javascript) à partir de rien.

Étape 1 : création du fichier HTML

Créez un fichier appelé multiplication.html contenant le code suivant.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <p>Voici le nombre entré :</p>
  </body>
</html>

```

Étape 2 : demander un nombre

Ajoutez deux scripts Javascript à ce document HTML :

1. le premier, situé dans la balise <head> et demandant à l'utilisateur d'entrer un nombre (en utilisant prompt) ; le nombre sera stocké dans une variable judicieusement nommée ;
2. le second, après la balise de fin </p>, qui écrira ce nombre dans un nouveau paragraphe <p> (en utilisant document.write).

Par exemple, si le nombre entré est 7, le second script devra produire le code HTML

```

<p>7</p>

```

Étape 3 : le faire avec style

Afin de mettre en évidence le nombre choisi (qui servira de base pour une table de multiplications par la suite), on peut lui ajouter un style (c'est aussi l'occasion de revoir un peu de CSS). Créez un fichier multiplication.css contenant le code suivant.

```

.nombre {
  text-align: center;
  background-color: blue;
  color: white;
  font-weight: bold;
}

```

Ajoutez également la ligne suivante dans la partie « head » du document HTML afin de lier la feuille de style.

```

<link rel="stylesheet" type="text/css" href="multiplication.css"/>

```

Modifiez le script Javascript pour que le nombre entré soit affiché en utilisant ce style. Le code HTML produit par Javascript devra être similaire à

```

<p class="nombre">7</p>

```

Note. En Javascript, on peut utiliser indifféremment des guillemets ou des apostrophes pour encadrer les chaînes de caractères. C'est tout particulièrement intéressant quand on veut écrire dans le document du code HTML contenant des guillemets. Une autre option consiste à utiliser \" à l'intérieur d'une chaîne encadrée par des guillemets (ou \' à l'intérieur d'une chaîne encadrée par des apostrophes).

Étape 4 : boucle Javascript (version liste)

C'est parti pour la table de multiplication...

Complétez le script d’affichage (celui qui se trouve dans la partie « body ») pour qu’après avoir sorti le nombre choisi, il affiche la table de multiplication de celui-ci. Dans un premier temps, vous pouvez sortir la table de multiplication sous la forme d’une liste HTML.

Pour rappel, une liste HTML se compose avec les balises `` et ``. À l’intérieur de ces balises, chacun des éléments est encadré par les balises `` et `` (li = list item). Par exemple :

```
<ul>
  <li>Élément 1</li>
  <li>Élément 2</li>
  <li>Élément 3</li>
</ul>
```

Dans la situation qui nous préoccupe, le but est d’obtenir la sortie suivante (si le nombre entré est 7) :

- $7 \times 1 = 7$
- $7 \times 2 = 14$
- $7 \times 3 = 21$
- $7 \times 4 = 28$
- ...
- $7 \times 10 = 70$

Pour ce faire, vous devrez étoffer votre script Javascript en utilisant une boucle. Les boucles Javascript utilisent la même syntaxe que celles en C ou en Java ; on peut même déclarer la variable utilisée dans la boucle à l’intérieur de celle-ci. Voici un exemple dont vous pouvez vous inspirer.

```
for (let age = 18 ; age < 25 ; age++) {
  alert("age = " + age);
}
```

Pour voir l’effet de ce bout de code, vous pouvez tout simplement le copier/coller dans la partie interactive de la console de Firefox !

Étape 5 : boucle Javascript (version tableau)

Dans un second temps, sortez la table de multiplication sous la forme d’un tableau HTML. Pour rappel, un tel tableau a la forme suivante.

```
<table>
  <tr><th>Nombre</th><th>Facteur</th><th>Résultat</th></tr>
  <tr><td>7      </td><td>1      </td><td>7      </td></tr>
  <tr><td>7      </td><td>2      </td><td>14     </td></tr>
  <tr><td>7      </td><td>3      </td><td>21     </td></tr>
  <tr><td>7      </td><td>4      </td><td>28     </td></tr>
  ...
  <tr><td>7      </td><td>10     </td><td>70     </td></tr>
</table>
```

Étape 6 : révisions CSS

Retour au CSS. Définissez, dans la feuille de styles, des règles de style correspondant aux sélecteurs suivants :

- `th` : pour les titres des colonnes (par exemple : fond grisâtre, écriture bleue, gras)
- `tr` : pour les lignes (par exemple : fond blanc)
- `table tr:nth-child(odd)` : pour les lignes impaires (par exemple : fond jaune)
- `tr td:nth-child(3)` : pour les résultats (les 3^e valeurs) de chaque ligne (par exemple : gras)

Assurez-vous que vous comprenez bien la signification de chacun de ces quatre sélecteurs !

Étape 7 : un code en ordre

Mettez un peu d'ordre dans votre code.

Généralement, on place toutes les définitions de fonctions Javascript dans la balise `<head>` du document HTML. À l'intérieur de la balise `<body>`, on ne trouve donc plus que des appels de fonctions.

Transformez le code d'affichage du nombre choisi et le code d'affichage de la table de multiplication en deux fonctions qui seront définies (mais pas appelées) dans `<head>`. Placez les appels adéquats au bon endroit dans `<body>`.

Étape 8 : valeurs inattendues

Testez votre script Javascript lorsqu'on lui donne des données inattendues, comme par exemple :

- rien (on appuie directement sur Enter sans entrer de nombre) ;
- un mot (« pomme ») au lieu d'un nombre ;
- « 12+4 »
- « 5doigts »

Étape 9 : validation du nombre entré

La commande `prompt` renvoie la réponse donnée par l'utilisateur sous la forme d'une chaîne de caractères. Rien ne dit qu'il s'agit d'un nombre valable !

Dans de nombreuses applications sur le web, il est nécessaire de « valider » les informations entrées par l'utilisateur. Ce mot peut recouvrir toute une série de vérifications plus ou moins complexes (validité d'une adresse mail, validité d'un mot de passe...).

Dans un premier temps, convertissez la réponse de l'utilisateur en nombre. Pour ce faire, vous pouvez utiliser la fonction `Number` : `Number(x)` est la conversion de « x » (quel que soit son type) en un nombre.

Javascript prévoit trois « nombres » particuliers :

- `NaN`, qui signifie « not a number » : c'est la réponse donnée quand on tente de transformer une chaîne comme "bonjour" en un nombre ;
- `Infinity`, qui est le résultat de `5/0` par exemple ; et
- `-Infinity`, qui est le résultat de `-3/0`.

Pour observer ces valeurs, vous pouvez par exemple rentrer les trois lignes suivantes dans la console interactive de Firefox.

```
"pomme" * 4  
71 / 0  
-12 / 0
```

La fonction booléenne `isFinite(x)` permet de vérifier que `x` est un nombre « fini », c'est-à-dire une valeur numérique autre que `NaN`, `Infinity` et `-Infinity`. Modifiez le script de `<head>` pour qu'il demande à l'utilisateur de rentrer un nombre jusqu'à ce que ce dernier le fasse (si l'utilisateur entre autre chose, on lui pose la question à nouveau).

Dans votre script, vous pourrez utiliser une boucle `while` (là aussi, la syntaxe est identique à celle qu'on emploie en C ou en Java).

Étape 10 : Javascript en mode externe

Pour terminer, passez à une version externe (fichier séparé) du code Javascript.

Extrayez tout le code Javascript de `<head>` sous la forme d'un fichier séparé (`multiplication.js`) puis remplacez-le par le code HTML permettant d'intégrer ce fichier.

Exercice 4 : compteur, boutons et liens

Cet exercice est moins détaillé que le précédent : ici, c'est à vous à voir où placer les scripts et comment les organiser.

Étape 1

Créez un fichier html comportant le code suivant. Observez son affichage.

```
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8"/>  
  </head>  
  <body>  
    <p>Combien de fois aurez-vous le courage de cliquer sur le  
bouton ?</p>  
    <button>Click me!</button>  
    <p>Cliquez <a>ici</a> pour remettre le compteur à zéro !</p>  
  </body>  
</html>
```

Étape 2

Le document comporte deux éléments potentiellement réactifs mais ils ne font rien pour l'instant. Il s'agit du bouton « Click me ! » et du lien sur le mot « ici » (qui, vu qu'il ne mène nulle part, n'est sans doute même pas représenté comme un lien par Firefox).

Dans un premier temps, pour tester l'association de scripts Javascript à ces éléments, modifiez le fichier html pour que

- cliquer sur le bouton ouvre une fenêtre indiquant « Vous avez cliqué sur le bouton ! » (vous pouvez utiliser la fonction « alert ») ;
- cliquer sur le lien fasse apparaître une fenêtre indiquant « Vous avez cliqué sur le lien ! ».

Comme ces deux opérations sont assez courtes, vous pouvez incorporer le code directement dans les balises html <button> et <a> (dans un attribut onclick pour <button> et dans un attribut href pour <a>).

Étape 3

Pour faciliter les éditions qui vont suivre, mettez le code de ces actions dans des fonctions séparées (appelées par exemple clicBouton() et clicLien()) et, dans les balises, appelez simplement la fonction adéquate.

Étape 4

Si vous avez observé le contenu du fichier html, vous aurez sans doute compris que le but final de cet exercice est de réaliser un compteur de clics sur le bouton. Chaque fois que vous cliquerez sur le bouton, il devra afficher (via « alert ») le nombre de clics déjà effectués. D'autre part, en cliquant sur le lien, ce nombre devra être remis à zéro.

Modifiez les fonctions clicBouton() et clicLien() pour que tout se passe de la sorte ! N'oubliez pas de déclarer la variable compteur en-dehors des fonctions !

Exercice 5

Construisez un document HTML (avec un ou plusieurs scripts Javascript) qui permettra à l'utilisateur de rentrer des nombres (les uns après les autres, une chaîne vide ou autre chose qu'un nombre pour terminer). Le document devra produire un tableau à 5 colonnes comportant une ligne de titres (voir ci-dessous) et une ligne de plus pour chaque nombre entré.

Chaque fois que l'utilisateur aura entré un nombre, le script écrira une nouvelle ligne (à l'aide de document.write) reprenant (a) le numéro du nombre entré, (b) le nombre entré lui-même, (c) le minimum des nombres entrés jusque-là, (d) le maximum des nombres entrés jusque-là et (e) la somme des nombres entrés jusque-là.

Par exemple, si l'utilisateur entre les nombres -3, 17, -5 et 20 (avant d'appuyer simplement sur Enter), le document devrait afficher le tableau suivant.

No	Nombre	Min	Max	Somme
1	-3	-3	-3	-3
2	17	-3	17	14
3	-5	-5	17	9
4	20	-5	20	29

Le script principal sera donc une boucle qui demandera à l'utilisateur une entrée puis qui, en fonction de cette entrée, ajoutera une ligne au tableau (via document.write) ou se terminera (n'oubliez pas de fermer les balises laissées ouvertes).

Vous pouvez vous aider des fonctions `Math.min()` et `Math.max()` qui acceptent un nombre quelconque d'arguments et donnent (respectivement) le minimum et le maximum de ces valeurs.

Dans le tableau, les nombres positifs (et nuls) seront affichés en bleu et les nombres négatifs en gras rouge. Pour réaliser cet affichage de manière « organisée », il sera sans doute préférable de vous baser sur les deux styles CSS suivants (à placer dans une balise `<style>` située dans l'en-tête du document).

```
.negatif { color : red ; font-weight : bold ; }  
.positif { color : blue ; }
```

Pour vous faciliter la tâche, créez une fonction `afficheValeur()` prenant comme argument le nombre à afficher et renvoyant le code HTML pour une cellule de tableau affichant ce nombre (y compris le style à utiliser : classe `negatif` ou classe `positif`).