Université Lille1 Sciences et Technologies

Séances 4a/4b – Technologies du Web

FORMATIONS EN INFORMATIQUE DE LILLE 1

février 2013

Javascript

Exercice 1 : Préalable : mise en place des outils

Pour ce premier exercice, nous allons utiliser des outils de développement Web intégrés au navigateur **firefox**. Avant de commencer, il s'agit donc d'abord d'activer ces outils.

1. Ouvrez le navigateur firefox. Pour commencer avec une page blanche, entrez dans la barre d'URL : `about:blank`.
2. **La « Console Web ».** Pour activer la console web, utilisez le raccourci clavier `Ctrl+Maj+K`) ou passez par les menus (*Outils/Développeur Web/Console Web*). La console web occupe maintenant une partie de la fenêtre du navigateur (haut ou bas, selon les versions) et vous pouvez régler à la souris l'espace qui lui est dévolu.

   

   La plus grande partie de la console permet d'afficher des messages, répartis en 4 grandes catégories :
   - Réseau : messages concernant les accès aux différents fichiers (html, css, js, images ...) nécessaires à la composition de la page.
   - CSS : messages concernant la prise en charge des CSS, comme par exemple les erreurs de syntaxe CSS éventuelles.
   - JS : idem pour le javascript
   - Journal : messages divers, en particulier ceux émis par la fonction `console.log` (voir ci-dessous).

   Les 4 boutons (avec menus déroulants) en haut de la console permettent d'afficher ou de masquer les messages de ces 4 catégories.

   Désactivez les messages de la catégorie «Réseau» mais activez ceux des 3 autres catégories (à la fois les erreurs et les avertissements)
3. **L'ardoise Javascript (ou «ScratchPad»).** Ouvrez-la par le raccourci `Shift+F4` ou par le menu (*Outils/Développeur Web/Ardoise*)

   Une fenêtre s'ouvre avec un petit éditeur dédié au javascript. Il est fortement conseillé de redimensionner la fenêtre principale de firefox et celle de l'ardoise afin de pouvoir disposer ces deux fenêtres côte à côte.

   Le code javascript que vous écrivez sur l'ardoise peut-être exécuté par la commande `Ctrl+R`. Par défaut l'ensemble du code de l'ardoise est exécuté, mais si vous sélectionnez une partie du code avant de faire `Ctrl+R`, seule celle-ci le sera.

   Attention, chaque exécution « prolonge » la précédente. Si vous voulez retrouver l'environnement initial, il vous faut réinitialiser (utilisez le menu, pas de raccourci).

   Vous disposez d'une fonction javascript `console.log(...);` qui permet d'afficher quelque chose dans la console web. À titre d'exemple, écrivez dans l'ardoise le commande `console.log("Hello");`,

exécutez-la avec `Ctrl+R` et vérifiez que vous en voyez le résultat dans la « console web ». (NB important : la commande console.log est, bien entendu, liée à l'outil console et dédiée uniquement aux tests et debuggage. Ce n'est pas un standard JS et elle ne doit surtout pas être utilisée dans les programmes liés à une page web).

Attention : Le programme saisi dans le scratchpad peut être sauvegardé facilement dans un fichier (voir menu). Il vous est conseillé de le faire très régulièrement afin de conserver une trace écrite de votre travail et afin de vous prémunir des conséquences d'une sortie accidentelle et inopinée du scratchpad.

Question 1.1 : *Les types*

1. On peut connaître le type d'une valeur javascript par la fonction `typeof( )`. Affichez dans la console les types de `1, true, 'abcde', "abcde", 0.0, '0.0', [1,2,3]`
2. Affichez le résultat du test `1=="1"` . Vous constaterez ainsi que ces valeurs sont considérées comme «égales» alors qu'elles sont de types différents. Examinez également les résultats de `1==[1]`. et de `1==[1,1]`
3. Javascript possède un opérateur `===` qui ne considère comme égaux que des valeurs identiques **ET** de même type. Examinez les résultats des comparaisons précédentes en utilisant, cette fois, l'opérateur `===`
4. Vérifiez que les valeurs `'abcde'` et `"abcde"` sont strictement égales.

Question 1.2 : *Les tableaux*

1. Définissez une variable t qui désignera un tableau contenant les valeurs 1, 2 , 3 , 4, 5. puis affichez la valeur de t. Affichez la valeur de t[0].
2. En utilisant les méthodes disponibles sur les tableaux (voir notes de cours), ajoutez à t un 0 au début et un 6 à la fin. Quel est maintenant l'indice de la valeur 1 dans t ?
3. Écrivez une fonction `afficherTab` acceptant un paramètre (supposé être un tableau). Cette fonction doit renvoyer une chaîne comportant, entre accolades, les éléments du tableaux séparés par des `|`. (exemple `{23|25|10}`)
4. Écrivez une fonction `occurrences` acceptant deux paramètres (un tableau et une valeur quelconque). La fonction doit renvoyer le nombre d'occurrences de cette valeur dans le tableau (valeur ET type identiques) et testez-la.

Question 1.3 : *Les chaînes*

Référez-vous aux notes de cours pour connaître les principales méthodes disponibles sur les chaînes.

1. Écrivez une fonction `capitalize` acceptant un paramètre s (supposé être une chaîne). Cette fonction doit renvoyer la chaîne obtenue en mettant en majuscule la première lettre de s. Si s ne commence par par une lettre (et a fortiori si s est vide) le résultat de la fonction est une chaîne identique à s.
2. Écrivez une fonction `decoupe` acceptant un paramètre s (supposé être une chaîne). Le résultat doit être un tableau obtenu en découpeant la chaîne selon les espaces. Par exemple pour `"abcd efg h"` on obtient `["abcd","efg","h"]`

Question 1.4 : Ajout d'attributs aux objets

Il est possible d'ajouter des attributs à tout objet javascript. Nous illustrerons cette faculté sur un tableau (qui est un objet).

1. Définissez une variable t désignant un tableau, puis ajoutez-lui un attribut nommé `nom` qui contiendra une chaîne de caractère de votre choix.

2. Modifiez la fonction `afficherTab` de façon à ce qu'elle incorpore la valeur de l'attribut `nom` de son paramètre, **lorsque celui-ci est défini**, de la manière indiquée par cet exemple : si t est égal à `[23,25,10]` et si son attribut `nom` vaut `"Mon_tableau"` : le résultat de l'affichage sera `Mon_tableau:{23|25|10}` (NB : quand un attribut `x.att` n'est pas défini, alors `typeof(x.att)` vaut `'undefined'`).

Question 1.5 : *Ajout de méthodes*
Une méthode est un attribut qui désigne une fonction. Dans cette fonction, le mot `this` désigne l'objet dans le contexte duquel elle s'exécute.

1. Définissez une nouvelle fonction `oAfficherTab` qui reprend l'essentiel du code de `afficherTab` mais
   – en supprimant le paramètre
   – dans le corps de la fonction : en remplaçant par le mot *this*, l'identificateur du paramètre supprimé.
2. Vous allez maintenant créer une méthode `image()` pour le tableau t en exécutant l'instruction suivante : `t.image = oAfficherTab;`. (remarquez bien l'absence de parenthèses)
   Testez la méthode créée (`t.image();`).
3. Faites de même pour ajouter au tableau t une méthode `compter(val)` qui réalise la même action que la fonction `occurrences`
4. Ajoutez au tableau une méthode `rotationGauche()` qui décale les éléments en déplaçant le premier élément à la fin du tableau.
5. Si l'on veut qu'une méthode soit définie pour **l'ensemble** des tableaux, il faut la définir pour le **prototype** des tableaux. Ce prototype s'appelle `Array.prototype`. De manière analogue à ce que nous avions fait sur UN objet, on définit une méthode sur le prototype de la façons suivante
   ```
   Array.prototype.methodeADefinir = nomDeLaFonction;
   ```

   Par exemple

   ```
   Array.prototype.image=oAfficherTab;
   ```

   Faites en sorte que les 3 méthodes précédentes soient définies pour l'ensemble des tableaux. Vérifiez en définissant un nouveau tableau de votre choix et en invoquant les méthodes créées.

Question 1.6 : *Accès aux éléments du DOM*
Dans la fenêtre du navigateur, chargez le document `demojs.html` (ce document est associé à la feuille de style `demojs.css`). Consultez **sans les modifier** ces 2 fichiers pour en comprendre la structure et le contenu.
Le programme javascript exécuté dans l'ardoise peut consulter et même modifier l'arbre du document affiché dans la fenêtre du navigateur.
Écrivez le code javascript permettant d'afficher dans la console

1. le "tagName" de l'élément d'identifiant `section2`
2. le nombre d'éléments `h2` du document.
3. l'url de chaque image du document (rappel : l'url figure dans l'attribut `src` de l'élément `img`)
4. l'attribut `class` de chaque élément `div`
5. le nombre d'éléments `p` situés dans l'élément d'identifiant `messages`
6. le contenu du premier des ces éléments `p`.

Question 1.7 : *Recherche par **querySelectorAll***
Les versions raisonnablement récentes de tous les navigateurs implémentent une nouvelle API :

`document.querySelectorAll(selecteur)`. L'argument `selecteur` est une chaîne représentant un sélecteur selon la syntaxe CSS. Le résultat est la liste de tous les nœuds correspondants.
Par exemple `document.querySelectorAll('ol.toto>li')` renvoie tous les `li` d'une liste `ol` appartenant à la classe `toto`.
Écrivez le code javascript permettant d'afficher dans la console

1. le nombre d'éléments `p` situés dans l'élément d'identifiant `messages` (en utilisant cette fois querySelectorAll)
2. le nombre d'éléments `h2` qui sont "premier enfant"
3. le nombre d'éléments `p` qui sont "premier enfant" et, pour chacun d'eux, l'identifiant de l'élément dans lequel il est inclus.

Question 1.8 : *Modification d'attributs des éléments du DOM*

1. Modifiez l'attribut src de la deuxième image en y rangeant l'url `http://www.fil.univ-lille1.fr/technoweb/images/technoweb-gris.png`
2. Modifiez l'attribut `class` de la 3ème section pour qu'elle appartienne aux classes `section` et `important` et observez le résultat.

Question 1.9 : *Création et suppression d'éléments*

1. Créez un nouveau nœud de tag `p` puis un nœud de texte (à votre convenance) que vous y insèrerez. Insérez ensuite dans l'arbre du document le paragraphe créé : vous le placerez à l'intérieur de l'élément `messages`, en dernière position. Observez la mise à jour du document dans la fenêtre du navigateur.
2. Faites de même mais en insérant, cette fois, le paragraphe au début de l'élément `messages`
3. Construire une fonction `printMessage` à un argument représentant un texte à insérer dans un nouveau paragraphe qui sera inclus en tête de l'élément `messages` et testez-la.
4. Construire une fonction `deleteMessage` qui supprime le premier élément inclus dans l'élément `messages` (lorsqu'il existe).
5. Construire une fonction `clearMessages` qui supprime l'ensemble du contenu de l'élément `messages` .

Question 1.10 : *Gestion des évènements*

– Créez un élément de tag `div` auquel vous attribuerez l'identifiant (`id`) `boutonclear` et la classe `bouton` . Dans ce nouvel élément, vous mettrez le texte « Effacer les messages ». Vous insèrerez ce nouveau nœud juste **avant** l'élément `messages`. Vous constaterez que le nouvel élément prend l'apparence définie par des règles CSS déjà prévues pour la classe `bouton` dans le fichier `demojs.css`.
– On souhaite maintenant rendre ce nouvel élément « cliquable » avec pour effet de supprimer l'ensemble des messages. En utilisant la fonction `addEventListener` faites en sorte que la fonction `clearMessages` soit invoquée lors de tout évènement **click** sur cet élément. Testez le fonctionnement en ajoutant, depuis le scratchpad, des messages grâce à la fonction `printMessages` et cliquez sur le bouton créé pour les effacer.

Question 1.11 : *Création de « boutons »*
Vous allez maintenant structurer le code que vous avez testé à la question précédente et l'insérer dans des fonctions.

- Écrivez une fonction `creerBouton(ident, texte, action)`. Son rôle est de créer un bouton possédant l'identifiant `ident` et contenant le texte indiqué en paramètre. Le paramètre `action` désigne une fonction à déclencher lorsque l'on clique sur le bouton.
- Écrivez une fonction `creerBoutonEffacement()` qui crée le bouton `clearMessage` de la question précédente.
- le code de la question précédente est maintenant inutile. Mettez-le en commentaire, puis rechargez la page «Demo JS» dans le navigateur. Testez le fonctionnement de `creerBoutonEffacement()`

Question 1.12 : *Lancer une action javascript au chargement de la page : l'évènement* ***load*** *.*
Quand la totalité d'une page est chargée par un navigateur, celui-ci déclenche un évènement nommé **load** sur l'objet javascript prédéfini `window`.
Pour déclencher une action lors du chargement de la page, il suffit donc de faire de cette action un gestionnaire de l'évènement `'load'` sur l'objet `window`. Ce qui se réalise ainsi :
`window.addEventListener('load',action,false);` où action est le nom de la fonction à déclencher.

Ceci ne peut pas être testé à partir du scratchpad. Vous allez donc maintenant créer un fichier `demarrage.js` dans le même répertoire que les fichiers `demojs`.

Examinez le code de `demojs.html` : vous constaterez la présence d'une balise `<script>` . Celle-ci indique que le fichier `demarrage.js` doit être exécuté à chaque fois que le navigateur accède à la page. Comme ce fichier n'existait pas jusque là, rien ne se passait.
- copiez dans le fichier `demarrage.js` le code des fonctions `creerBouton` et `creerBoutonEffacement`
- ajoutez, après ces fonctions, la ligne `window.addEventListener('load',creerBoutonEffacement,false);`
- sauvez le fichier `demarrage.js` et rechargez la page. Vous devez constater que le bouton a été créé, signe que la fonction `creerBouton` a été exécutée une fois le document chargé. (si tel n'était pas le cas recherchez votre erreur en vous aidant des éventuels messages apparaissant dans la console).
- Vous constaterez que, s'il est cliquable, le clic n'a cependant aucun effet. La raison est simple : les fonctions `clearMessages` (etc) définies dans le scratchpad ne sont pas connues du document. Copiez donc dans le fichier `demarrage.js` le code des fonctions `clearMessages` et `deleteMessage`, rechargez la page dans le navigateur et testez le fonctionnement du bouton.

Question 1.13 : *Autre bouton*

1. Créez une fonction `printDate()` qui ajoute un message contenant la date et l'heure courante.
2. Écrivez une fonction `creerBoutonDate()` qui crée un bouton d'identifiant `boutondate`. Un clic sur ce bouton provoque l'exécution de printDate.
3. Créez un fichier `boutonDate.js` contenant les fonctions `printMessage printDate creerBoutonDate` ainsi que la ligne indiquant que la fonction `creerBoutonDate` doit être exécutée lors du chargement de la page (voir question précédente).
4. Éditez `demojs.html` et ajoutez une balise `<script>` analogue à celle qui s'y trouve déjà. Cette fois, c'est le script `boutonDate.js` que vous lierez à la page. Rechargez la page : les 2 boutons doivent être apparents et fonctionnels.

Exercice 2 : Dans cet exercice, il vous sera demandé d'associer un ou plusieurs fichiers javascript au fichier `recettes.html` (voir TD CSS). La **seule** modification que vous êtes autorisé à faire dans le fichier HTML est d'ajouter les balises `<script>` nécessaires.

On remarquera que le document HTML contient un unique titre principal (élément `h1`) et que chaque élément `h3` correspond à une nouvelle recette. Les titres `h2`, eux, définissent des "catégories de recettes".
Lors du chargement du document, le programme javascript va venir apporter quelques modifications au document. Ces modifications ne sont pas indispensables à la consultation du document mais elles vont la faciliter.

Question 2.1 : Faites en sorte que le titre principal comporte le nombre de recettes présentes dans la page, entre parenthèses (exemple : «Recettes (3)»).

Question 2.2 : Ajoutez à chaque titre `h3` un attribut `id` dont la valeur sera composée du mot `recette` suivi du numéro d'ordre de la recette dans la page (`recette0 recette1 recette2 ...`). Cette opération n'a pas d'effet visible sur le document mais sera utile pour la question suivante.

Question 2.3 : Faites apparaître dans le document un sommaire comportant la liste des noms de recettes c'est à dire la liste des textes figurant dans les titres `h3` (les titres `h2` sont ignorés).

- le sommaire sera composé d'un élément `div` possédant l'indentifiant `sommaire` .
- dans ce `div` on trouvera d'abord un titre `h5` contenant par exemple le mot "Sommaire".
- puis une liste (ul ou ol) des noms de recettes. Chaque nom de recette sera inclus dans un lien (élément `a`) renvoyant à la recette correspondante (utiliser ici dans les URL les identifiants `recette0` ... etc)
- le sommaire sera inséré **immédiatement après** le titre `h1` du document.

On supposera que chaque élément `h3` comporte un unique nœud de texte et pas d'autres éléments imbriqués.

Question 2.4 : Dans un fichier `styleSommaire.css` vous mettrez les règles CSS permettant d'améliorer la présentation du sommaire. Pour lier ce fichier à la feuille de style principale vous ajouterez au début du fichier `recettes.css`, la ligne : `@import url("styleSommaire.css");`

Exemple :