UNIVERSITÉ DE ROUEN

Langages Web 1 – TP 08 2016–2017

L'API D.O.M. en langage Javascript.

Introduction

— Vous trouverez une référence pour HTML DOM à cette adresse : http ://www.w3schools.com/jsref/, et pour XML DOM http ://www.w3schools.com/dom/dom_nodetype.asp

— Nous utiliserons les mêmes outils qu'au TP précédent, à savoir, **la console d'erreur**, **l'ardoise Javascript** et la **console Web** de **Firefox**.

Exercice 1 – *Premiers pas avec DOM*

Ouvrez le document `DOM.html`

Question 1.1 : Examinez le contenu et indiquez :

1. combien il y a d'éléments de type `<P>`,
2. combien il y a d'éléments de classe `content`,
3. quel sont les éléments avec les *id* `intro` et `components` ?

Vous réaliserez les questions suivantes dans l'ardoise JavaScript (`maj+F4`) :

Question 1.2 : Rappelez à quel objet appartient la méthode `getElementById()`. Utilisez cette méthode pour obtenir une référence à l'élément d'identifiant `intro`. Affichez le type de l'objet obtenu (simplement avec `alert()`).

Question 1.3 : Quelle méthode allez-vous utiliser pour obtenir tous les éléments de type `P` et quelle méthode allez-vous utiliser pour obtenir tous les éléments de classe `content`. Comme précédemment, précisez sur quel(s) objet(s) on peut les appeler. Utilisez pour obtenir les éléments en question et comme précédemment, précisez la nature de l'objet que ces méthodes retournent respectivement.

Question 1.4 : À partir de l'objet retourné pour la liste des éléments de classe `content` de la question précédente, écrivez une boucle qui affiche dans la console web le contenu de cet objet (attributs+valeurs). Que peut-on dire de cet objet ?

Question 1.5 : Écrivez maintenant une boucle affichant dans la console web le nombre d'éléments sélectionnés puis la suite des types et des classes des éléments sélectionnés.

Dans le fichier CSS accompagnant est fourni une classe `comment`.

Question 1.6 : Écrivez une boucle qui permet ce changer la classe de tous les éléments de classe `content` pour la classe `comment`. Si vous avez utilisez une autre méthode, faites tout de même un test avec une boucle de type `for(var i=0; i<...; ++i)` et décrivez le résultat obtenu et en l'expliquant.

Toute la puissance de DOM réside dans le fait de pourvoir modifier complètement le contenu d'un arbre, y compris en y ajoutant du contenu créé dynamiquement.

Question 1.7 : Rechargez la page pour annuler les altérations des question précédentes. Écrivez un script qui

1. Créé un nouveau nœud de type `span`,
2. de classe `comment` et
3. contenant le texte « Je suis un commentaire dans une span. »
4. ajoute cet élément à la fin du document.

Question 1.8 : Reprenez la question précédente (rechargez la page) mais cette fois en greffant le nœud créé à la fois à la fin du document et à la fin de la `div` d'identifiant `intro`. (Mettez une pause entre les deux greffe à l'aide d'un `alert()` afin de pouvoir observer le résultat). Qu'en déduisez vous ? Quel en est la raison ?

Créer des structures complexes en DOM peut être à la fois laborieux et inutile. Une technique fréquemment utilisée consiste à cloner un sous-arbre d'un composant caché, plus ou moins complexe, préparé à l'avance (souvent envoyé avec la sérialisation initiale de la page).

Question 1.9 : Écrivez un script qui sélectionne tous les paragraphes (éléments de type `P`) et leur ajoute comme premier fils un clone du sous-arbre dont l'élément d'identifiant `pmenu` est la racine. Que signifie le clonage profond (*deep*) ?

Exercice 2 – *Validation de formulaire*

Question 2.1 : Récupérez le contenu du répertoire `validation` et ajoutez un script au fichier `validation.html` afin que le formulaire ne soit envoyé que si les conditions suivantes sont respectées :

- Le nom et le prénom doivent chacun comporter au moins 3 caractères ;
- L'utilisateur doit avoir choisi un des deux formats ;
- L'adresse email doit être valide. Pour faire simple elle doit être de la forme au moins un mot (une ou plusieurs lettres) suivie du signe arobase puis un mot, un point et un mot. Pensez aux expressions régulières (regexp) de javascript.

Lorsqu'un champs sera en erreur, on lui appliquera le style `error` (voir le css) afin de le mettre en évidence. On lui appliquera le style "none" s'il est valide (en cas de correction).

Question 2.2 : Placez tout le code de la question précédente dans un fichier externe (disons `validation.js`) et retirez toute trace de code javascript du fichier HTML y compris dans les attributs de gestion d'événements. Ajoutez du code dans le fichier externe de sorte que la simple inclusion de ce fichier rende son contenu actif. (Pensez à l'événement `onload` de `window`).

Question 2.3 : Ajoutez au formulaire l'image `subscribe.jpg` et ajoutez un script qui, lorsque le curseur de la souris survole l'image provoquera l'envoi du formulaire validé.

Exercice 3 – *DOM Only : un sommaire automatique*

Question 3.1 : Récupérez le contenu du répertoire `sommaire`. Ajoutez un script au fichier `sommaire.html` de telle sorte qu'un sommaire cliquable soit automatiquement généré au chargement du fichier. Pour simplifier, le fichier CSS fourni prend en charge la numérotation automatiquement, il suffit simplement d'ajouter des div de classes respectives `somH2`, `somH3` et `somH4`. On notera le nommage judicieux des classes. Pour cet exercice, on s'interdit l'emploi des attributs `textContent`, `innerHTML`, etc.. Les fonctions `document.createTextNode` et `document.createElement` devront être utilisées à la place. ATTENTION de bien ajouter le sommaire au document un fois le calcul terminé afin d'éviter que le sommaire ne s'indexe lui-même (récursion infinie) !

Question 3.2 : Ajoutez un titre de niveau 2 intitulé "Introduction" devant le premier paragraphe et constatez que le sommaire se met à jour.

Exercice 4 – *InnerHTML ... c'est mal !*

Question 4.1 : Récupérez le contenu du répertoire `bart`, et ouvrez le fichier `bart.html`. Créez le fichier `bart.js` déjà inclus afin que lorsque l'utilisateur clique sur le bouton, la phrase qu'il a saisie soit copiée 10 fois sur le tableau. Pour ce premier essai, nous allons utiliser la propriété `innerHTML` des éléments DOM.

Question 4.2 : Cette méthode est-elle robuste, testez des injections HTML (ex : `blabla<b>bold</b>blabla`) et Javascript (`blabla<script type="text/javascript">alert('Owned !... or not');</script>blabla` puis `<p onmouseover="alert('coucou')"> Gotcha ! !</p>`)

Question 4.3 : Remplacez la propriété `innerHTML` par la propriété `textContent` et testez à nouveau vos injections.

Question 4.4 : Expliquez pourquoi le code :

```
1 ...
2 for (var i=0; i<...: ++i) {
3   e.innerHTML += something[i];
4 }
5 ...
```

est moins performant que le code :

```
...
var s="";
for (var i=0; i<...: ++i) {
   s += something[i];
}
e.innerHTML+=s;
...
```

Enfonçons le clou pour montrer que l'usage de `innerHTML` est une mauvaise pratique et ne respecte pas l'esprit de DOM.

Question 4.5 : Rétablissez votre code afin qu'il utilise à nouveau `innerHTML`. Vérifiez qu'il fonctionne à nouveau. Ouvrez maintenant le fichier `bart.xhtml` et constatez qu'il inclus bien votre fichier `bart.js`. Que se passe-t-il lorsque l'on clique sur le bouton ? Quel en est l'explication ? (Vérifiez la console d'erreurs).

Question 4.6 : Expliquez pourquoi une approche DOM est plus robuste et résout le problème.

Question 4.7 : Réécrivez votre script en n'utilisant que DOM (sans `innerHTML` ni `textContent`) et vérifiez qu'il marche à la fois pour `bart.html` et `bart.xhtml`.

Exercice 5 – *Questions bonus.*

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 5.1 : Écrivez un script qui imprime l'arbre du document dans la div de sortie. On utilisera une simple indentation pour la profondeur et on utilisera la propriété `nodeName` pour imprimer le nom des nœuds. (Pensez à la récursivité)

Question 5.2 : Écrivez une instruction qui change l'étiquette du bouton1 en "Clear !", puis affectez lui une fonction anonyme qui effacera la div de sortie.

Question 5.3 : Écrivez une instruction qui change le style (en fait la classe) des modules (éléments de classe `module`) en `module_bis`. Notez que tous ces modules sont des `div` !