

Modification dynamique de pages HTML

Nom et prénom : _____

Introduction et objectifs du TP ¹

- Utiliser les fonctionnalités du DOM pour modifier le contenu d'une page WEB.

6.1 Petite séance de présentation

6.1.1 Qu'est-ce que le *Document Object Model* (DOM) ?

Le DOM est une recommandation du W3C qui définit une interface universelle (indépendante du langage de programmation et/ou du navigateur) pour accéder au contenu, à la structure et au style des pages HTML. D'ailleurs, bien qu'habituellement utilisé dans le cadre de la programmation Web, le DOM peut s'appliquer à n'importe quel type de document. Sa mise en place a démarré en 98 et venait en remplacement des autres modèles non standardisés (et peu pratiques).

6.1.2 Quel rapport entre DOM et XML ?

Le langage Extensible Markup Language (XML) est destiné à la production de document structurés. Dans ses grandes lignes, il se définit comme un ensemble de balises identifiées par les marqueurs < et >. La structure d'arbre du DOM est due à son utilisation.

6.1.3 Comment construire un arbre DOM ?

La construction de l'arbre se fait hiérarchiquement. Chaque ouverture de balise définit un nouveau sous-niveau. Les contenus de type texte constituent les feuilles de l'arbre. La figure 6.1 représente les relations de parentés entre les nœuds.

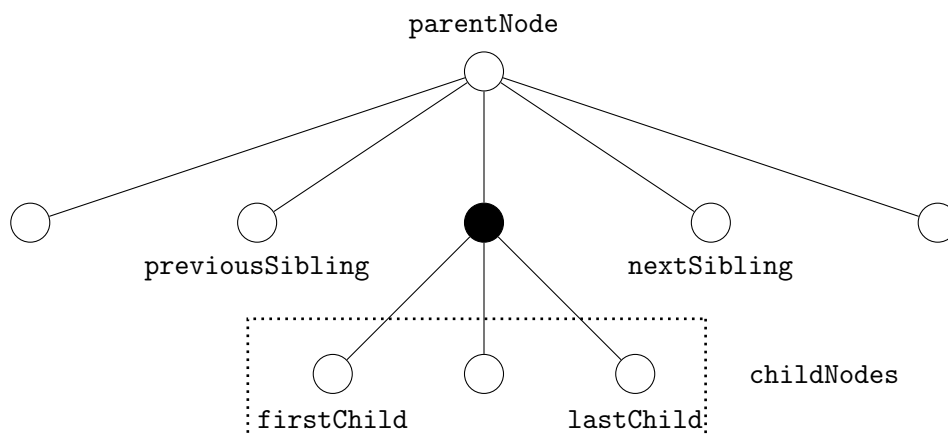


FIGURE 6.1 – Liens entre nœuds dans l'arbre

1. Inspirés des supports de cours de C Guéret

Question 1



En fonction de ce qui est décrit dans le schéma 6.1, indiquez à quoi correspondent les termes ci-dessous :

- parentNode : _____
- nextSibling : _____
- previousSibling : _____
- childNodes : _____
- firstChild : _____
- lastChild : _____

Ces relations sont utilisées comme étant des propriétés² des nœuds, comme « forms ». Dans Javascript, elles sont donc utilisées en les rajoutant à la suite du nœud.

Question 2

En utilisant l'attribut `length` et en partant du principe que le nœud en noir est associé à la variable de nom `node`, écrivez ci-dessous une ligne de code permettant d'afficher une boîte d'alerte indiquant le nombre de fils du nœud contenu dans la variable « `node` »



```
1 alert(node. _____)
```

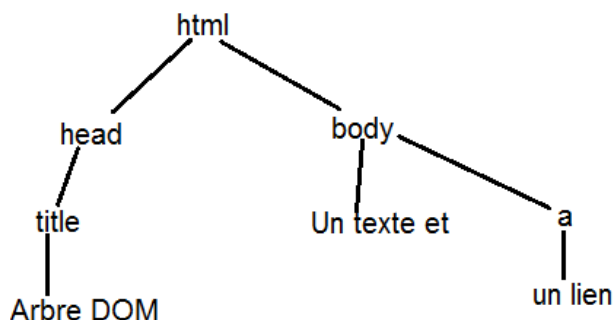
Selon le DOM, chaque page HTML est, dans sa globalité, décrite par un arbre de nœuds pouvant avoir d'autres nœuds fils. Il existe différents types de nœuds :

- “Element” (nom de la balise) pour les balises HTML
- “Texte” (`#text`) pour les textes contenus dans les éléments HTML
- “Racine” (`#document`) pour la racine du document
- “Commentaire” (`#comment`) pour les commentaires
- “Attribut” pour les attributs des balises

Les mots entre parenthèses indiquent la désignation employée dans les outils de parcours de DOM. Les attributs n'y sont habituellement pas représentés.

Question 3

Dans l'espace ci-dessous, construisez l'arbre DOM correspondant à la page HTML de l'exemple `lien.html`.



2. On utilise plus volontiers le terme attribut mais il y a ici risque de confusion entre les attributs des objets et les attributs des balises HTML

```

1 <html>
2   <head>
3     <title>Arbre DOM</title>
4   </head>
5   <body>
6     Un texte et <a href="page.html">un lien</a>
7   </body>
8 </html>

```

Listing 6.1 – Code source de lien.html

6.1.4 Comment parcourir et modifier un arbre DOM ?

Pour parcourir un arbre, on commence par chercher un nœud par lequel commencer puis on se sert des relations entre les nœuds pour se promener dans l'arbre. Voici, ci-après, un listing de quelques fonctions classées par catégories. Celles-ci vont nous être utiles par la suite. Les spécifications DOM en définissent beaucoup d'autres que vous pourrez trouver sur divers sites de documentation.

Rechercher un nœud

Les fonctions « getElement » permettent de retrouver un nœud dans l'arbre.

- `getElementById` recherche une balise selon son identifiant. Celle-ci est utilisée depuis l'objet « document ».

Question 4

Complétez le code ci-dessous pour mettre dans la variable « toto » la balise dont l'identité est « truc ».

```

1 var _____ = _____ ("truc");

```

- `getElementsByTagName` constitue un tableau contenant toutes les balises du type indiqué. Dans cette fonction, attention à la présence du « s » après « getElement » ! C'est ce qui fait la différence entre les fonctions retournant un tableau ou une valeur unique.

Question 5

Complétez le code ci-dessous pour obtenir dans la variable « toto » la troisième balise « select » de la page.

```

1 var _____ = _____ ("select") [ ____ ];

```

Ajout et suppression de nœud

- `appendChild` est utilisée pour ajouter un nouveau nœud dans l'arbre

Question 6

Ecrivez ci-dessous une ligne de code permettant d'ajouter au nœuds dont l'identifiant est « toto » le nœud contenu dans la variable « node » de l'arbre. Pour retrouver « toto », utilisez la fonction de recherche adaptée.

```

1 document. _____ toto _____

```

- `removeChild` est utilisé pour retirer de l'arbre un nœud fils

Question 7

Ecrivez ci-dessous une ligne de code permettant de retirer la variable « node » de l'arbre. Pour y arriver, pensez à tirer partie des relations de parenté entre les nœuds.

```

1 _____

```

Création et initialisation d'éléments

- `createElement` crée, comme on peut s'en douter, un nouveau nœud de type élément. La nature du nœud, c'est à dire le nom de la balise (`nodeName`) est indiqué en paramètre.

Question 8


Ecrivez ci-dessous une ligne de code permettant de créer un nouvel element de type « tr » dans la variable « ligne ».

```
1  var ligne _____
```

- `createTextNode` crée, un nouveau nœud de type texte (ceux utilisés pour faire les feuilles de l'arbre).

Question 9

Ecrivez ci-dessous une ligne de code permettant de créer un nœud contenant la chaine de caractères « test » dans la variable « ligne ».

```
1  var ligne _____
```

- `setAttribute` fixe un attribut (passé en premier paramètre) à une valeur donnée (passée en second paramètre).

Question 10

Ecrivez ci-dessous une ligne de code permettant de changer l'adresse vers laquelle pointe le lien de l'exemple 6.1. Pour l'URL cible mettez ce que vous voulez. Il y a plusieurs solutions possibles, utilisez la façon de faire qui vous est la plus intuitive.

```
1  _____
```

6.2 3 Exercices à réaliser

Question 11

Complétez le fichier *exo1.html* afin que la fonction `getAllParaElems()` affiche le nombre de balises `< p >` présentes dans le code HTML et que la fonction `div1ParaElems()` affiche le nombre de balises `< p >` présentes entre les deux balises `< div >`.

```
1
2 function getAllParaElems()
3 {
4   var allParas = _____ ;
5   var num = allParas. _____ ;
6   alert("Il y a " + num + " elements <p> dans ce document.");
7 }
```

```
1
2 function div1ParaElems()
3 {
4   var div1 = _____ ;
5   var div1Paras = div1. _____ ;
6   var num = _____ ;
7   alert("Il y a " + num + " elements <p> dans l'element div1.");
8 }
```

Question 12

Le fichier *exo2.html* se décompose en trois parties :

1. Une phrase dans une balise *div*
2. Un formulaire composé d'un champ texte et d'un bouton
3. Une division dont l'ID est "fin" qui ne comporte rien

Lorsqu'on appuie sur le bouton, on souhaite que les opérations suivantes se réalisent :

1. Le texte de la première division est modifié et affiche "merci d'avoir appuyé sur le bouton !"
2. Le formulaire est supprimé
3. Un Textnode est créé et inséré dans la division dont l'ID est "fin"

Complétez la fonction affiche du fichier *exo2.html*. Vous pouvez utiliser les fonctions du premier TP (remplacerTexte et effacerTexte).

```
1 function affiche()
2 {
3 //on recupere le formulaire
4 var val = _____ ("zone");
5 //On recupere le texte entre au clavier
6 var texte=val. _____ (" _____ ") [ ____ ].value;
7 //On recupere la division "debut"
8 var el= _____.getElementById("debut");
9 //on remplace l'ancien texte
10 remplacerTexte(el. _____ , "merci d'avoir...");
11 //suppression du formulaire
12
13 //creation d'un noeud contenant du texte
14 var node = _____ (texte);
15 //ajout du texte entre au clavier dans la div "fin"
16 document. _____ ('fin'). _____ ;
17 }
```

Question 13

Complétez le fichier *exo3.html* afin que le texte soit supprimé et que le drapeau anglais s'affiche quand la souris passe au dessus du lien "English" et que le drapeau français s'affiche quand la souris passe au dessus du lien "Français". Quand aucun des deux liens n'est survolé, le page doit être à son état initial (avec le texte).

```
1
2 function flag(type){
3 if( type == 1 ){//si la variable recue vaut 1
4     //Suppression de l'ancien contenu
5     var paragraphe = _____ );
6     var oldContenu = paragraphe. _____ ;
7     paragraphe. _____ ;
8     //Ajout du nouveau contenu
9     var img = _____ ("img");
10    img.setAttribute( _____ , " _____ ");
11    paragraphe. _____ ;
12
13 }
14
15 else if( type == 2 ){//si la variable recue vaut 2
16     //Suppression de l'ancien contenu
17     _____
18     _____
19     _____
20     //Ajout du nouveau contenu
```

```

21      _____
22      _____
23      _____
24  }
25  else{
26      //si la variable recue vaut 0
27      //Suppression de l'ancien contenu
28      _____
29      _____
30      _____
31      //Ajout du nouveau contenu
32      var texte = document. _____
33          ("Bienvenue. Passez la souris sur l'un des deux liens!");
34  }
35
36  }

```

Question 14

Ouvrez le fichier `exempleDOM.html`, étudiez le code puis visualisez ce code dans un navigateur.

Concluez sur ce à quoi permettent d'accéder (selon les cas) les attributs `nodeType`, `nodeName` et `nodeValue`.



Ce document est publié sous Licence Creative Commons « By-NonCommercial-ShareAlike ». Cette licence vous autorise une utilisation libre de ce document pour un usage non commercial et à condition d'en conserver la paternité. Toute version modifiée de ce document doit être placée sous la même licence pour pouvoir être diffusée.

<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>