

Introduction au Document Object Model

Table des matières

I. Contexte	3
II. DOM, HTML et XML	3
A. Introduction au DOM	3
B. Structure d'un document	9
C. Exercice : Quiz	10
III. Introduction à la manipulation du DOM	11
A. Manipulation du DOM	11
B. Comprendre le DOM et l'importance du DOM Inspector pour les développeurs Front-end.....	11
C. Le DOM Forms	12
D. Exercice : Quiz.....	13
IV. Essentiel	14
V. Auto-évaluation	14
A. Exercice	14
B. Test.....	14
Solutions des exercices	15

I. Contexte

Durée : 1 h

Prérequis : connaître les bases du HTML/CSS/JavaScript

Environnement de travail : PC

Contexte

Lorsqu'une page est demandée *via* une URL, le navigateur commence à chercher un fichier contenant la page en format HTML ou PHP sur un serveur (ou localement). Une fois trouvé, ce fichier est chargé dans le navigateur pour s'afficher à l'écran. Le navigateur utilise ensuite un programme (une API) pour convertir les éléments de la page en un objet unique appelé « *document* » qui est formulé en langage JavaScript. Ce document est construit à partir d'un fichier central relié à des fichiers externes. Ce processus est connu sous le nom de Document Object Model (DOM), ou « *Modèle en Objet du Document* » en français. Dans la suite, nous allons examiner en détail ce qu'est le DOM.

II. DOM, HTML et XML

A. Introduction au DOM

La structure du code d'une page web

Lorsqu'un navigateur affiche une page web, celui-ci interprète le code qui lui est fourni. Il va ainsi interpréter le contenu avec le code HTML, interpréter la mise en page avec du CSS, et permettre la gestion de contenu dynamique à l'aide du PHP et du JavaScript. L'utilisateur ne doit voir en face de lui qu'une interface fonctionnelle.

Lors de la création de votre page web, vous travaillerez dans un premier temps sur des fichiers HTML et CSS afin de mettre en place toute la mise en forme de celle-ci. Il vous faudra utiliser les différentes balises HTML ou XML à bon escient. Dépendamment de la balise dans laquelle le contenu est placé, celui-ci sera visible ou non par l'utilisateur. Par exemple, du contenu écrit dans la balise `<head>` servira uniquement à donner des informations sur le document que vous êtes en train de rédiger. Du contenu peut également être rendu invisible d'autre façon, avec un attribut `hidden` par exemple.

Dans notre cas, le DOM (Document Object Model) est ce qu'on appelle une « *interface de programmation* » (API) créée pour le HTML et le XML.

Définition HTML

Le HTML est un langage de balisage (aussi appelé *mark up language*, soit un langage qui permet de structurer ou de mettre en forme des données en utilisant des balises) qui nécessite l'utilisation de chevrons (`<` `>`). Il ne s'agit pas d'un langage dynamique. En effet, à l'inverse de PHP ou de JavaScript qui vont nous permettre d'utiliser des boucles ou des conditions, le HTML va simplement permettre d'afficher du contenu comme des images, du texte, des liens, etc., par le biais de balises spécifiques.

Le HTML est une série de Balises ouvrantes et contenant. Chaque balise ouvrante `< . . . >` est jumelée avec une balise fermante `</ . . . >` qui établit la limite du contenu.

Exemple

```
1 <section></section>
```

Chaque balise peut contenir un ou plusieurs éléments, eux-mêmes balisés.

Exemple

```
1 <section>
2   <div></div>
3 </section>
```

Une balise ouvrante pourra inclure un ou plusieurs attributs (par exemple, class ou id) qui vont associer des informations, des liens, des spécifications au contenu du balisage. C'est là la tâche principale du html.

Exemple

```
1 <section class="comment">
2   <div class="name"></div>
3   <div id="important"></div>
4 </section>
```

Et c'est le langage JavaScript ou le css qui vont dynamiser, embellir ou animer les éléments html affichés.

Prenons l'exemple d'une section contenant deux div (le terme div étant l'abréviation de division). La première div contient un texte, la seconde une image. Le code HTML se composera alors de cette manière :

```
1 <section>
2   <div><p>Un texte qui accompagne l'image</p></div>
3   <div></div>
4 </section>
```

Définition

Le DOM

En informatique, le DOM (pour Document Object Model) est une interface de programmation permettant de représenter un document HTML ou XML sous la forme d'un arbre binaire, ou arbre de nœuds. Chaque nœud représente un élément du document (que ce soit une balise, un attribut, un texte, etc.). Le DOM permet aux programmeurs de manipuler des éléments d'un document en utilisant des langages de programmation comme le JavaScript par exemple. Lorsque des modifications sont effectuées sur le DOM, celles-ci sont automatiquement retransmises sur le document HTML ou XML. Ainsi, le DOM permet de créer des pages Web dynamiques et interactives.

La relation entre le HTML et le DOM

Le DOM (*Document Object Model*) en JavaScript sert à fournir une interface de programmation pour interagir avec les éléments d'une page HTML, tels que les éléments de formulaire, les images, les liens et les conteneurs de texte, ce qui permet aux développeurs de modifier la structure, le contenu et le style d'une page web dynamiquement, en réponse aux interactions de l'utilisateur ou à d'autres événements.

Grâce au DOM, il est possible de créer des pages web plus interactives et réactives en permettant la modification de la page en temps réel sans recharger la page. En outre, le DOM permet de créer des animations et des effets visuels sur la page, ce qui peut améliorer l'expérience utilisateur.

Le DOM peut être utilisé pour diverses tâches, telles que la validation de formulaire, l'ajout ou la suppression d'éléments HTML, la modification de texte et d'attributs, le positionnement et le dimensionnement d'éléments sur la page, ainsi que la gestion des événements de la souris et du clavier.

En outre, le DOM est utilisé par de nombreuses bibliothèques et frameworks JavaScript populaires, tels que jQuery, React, Vue.js et Angular pour créer des applications web plus complexes et plus avancées.

Définition

XML

Le XML est l'extension du format de balisage HTML, utilisant le même principe de balises ouvrantes et fermantes, mais où les noms des balises écrites dans le code sont plus libres.

Exemple

```
1 <espece>
2 <animale>
3 <chat>felix</chat>
4 <chien>médor</chien>
5 </animale>
6 <vegetale>
7 <arbre>chêne</arbre>
8 <fleur>paquerette</fleur>
9 </vegetale>
10 </espece>
```

Définition Le JSON

Un format a fait son apparition dans les années 2000 : le format JSON. Plus facile à lire, à élaborer, et plus rapide à analyser pour un ordinateur, il gère aussi certains types de données que le XML ne gère pas, par exemple les données en nombres ou les valeurs booléennes. Sa syntaxe est par ailleurs très proche du XML et le format en lui-même est compatible avec le langage JavaScript.

```
1 « 'espece' : {
2     « animale » : { « chat » : « felix »,
3                     « chien » : « 'médor »
4                     },
5     « 'vegetale' » : { « arbre » : « chêne »,
6                       « fleur » : « pâquerette »
7                       }
8 }
```

L'origine du DOM

D'un point de vue technique, le DOM est une API (interface de programmation) permettant la visualisation et la traduction des éléments composant la page web et son affichage contenu aujourd'hui dans tous les navigateurs. Il s'agit d'une standardisation et d'une optimisation qui se sont faites progressivement, à partir du milieu des années 1990, avec l'essor des nouvelles technologies de l'internet et du développement des langages associés. Anciennement, ECMAScript, aujourd'hui connu sous l'appellation de JavaScript, faisait partie de cet essor.

À partir du milieu des années 2000, une grande partie des DOM déployés étaient compatibles avec une norme standard définie par le W3C (World Wide Web Consortium), un organisme central créé à la fin des années 90 pour, justement, standardiser et compatibiliser les différents navigateurs et les technologies associées aux navigateurs. Ces évolutions et standardisations ont été portées jusqu'à récemment avec un DOM dit « level 4 » (niveau 4) datant de 2015. Par ailleurs, ce DOM est une API toujours en cours d'amélioration.

Le DOM a une histoire assez longue et tumultueuse. Les différentes sociétés Mozaic, Netscape, Microsoft, étaient en compétition dans les années 1990 en cherchant chacune à élaborer le navigateur le plus performant. On parle d'ailleurs de guerre des navigateurs. Chaque société développant ses propres standards, le problème de l'incompatibilité entre les différents navigateurs s'est très rapidement posé. Au début des années 2000, c'est Microsoft avec son Internet Explorer (IE) qui domine le marché. Aujourd'hui, le marché est bien plus concurrentiel, avec Google et son navigateur Chrome, Mozilla avec Firefox, Microsoft toujours présent avec son Internet Explorer devenu Edge et Apple avec Safari.

Aujourd'hui, le Web Browser est un produit normalisé, standardisé par le W3C, avec un DOM et des outils largement compatibles entre eux.

Définition Le DOM tree

Un DOM (*Document Object Model*) est une représentation hiérarchique d'une page web ou d'un document XML. Le *DOM tree* (ou arbre DOM) est un arbre de nœuds qui représente la structure d'un document HTML ou XML.

Chaque élément HTML est représenté par un nœud dans l'arbre DOM, qui contient des informations telles que le nom de l'élément, les attributs et le contenu. Les nœuds de texte, les commentaires et les instructions de traitement sont également représentés dans l'arbre DOM.

Le DOM tree permet aux développeurs web de manipuler dynamiquement le contenu et la présentation d'une page web à travers JavaScript ou d'autres langages de programmation. En utilisant des méthodes et des propriétés du DOM, les développeurs peuvent accéder, modifier, ajouter ou supprimer des éléments de la page web, et ainsi créer des applications web interactives et dynamiques. Chaque élément est alors représenté par un *node*, ou nœud.

Un node dans le DOM (*Document Object Model*) est un objet qui représente un élément dans un arbre de la structure d'un document HTML ou XML. Chaque élément, balise, texte, attribut, commentaire, etc. est représenté sous forme d'un nœud dans l'arbre DOM.

Le DOM tree est une structure hiérarchique où chaque élément est un nœud et est lié aux autres nœuds en fonction de sa position dans l'arborescence. Le nœud racine est généralement le document lui-même, suivi par les nœuds qui représentent les éléments HTML tels que les balises `<html>`, `<head>`, `<body>`, `<div>`, `<p>`, etc.

Chaque nœud a des propriétés qui le caractérisent : le nom de la balise, les attributs, les classes, l'ID, le contenu textuel, les événements, etc. Les nœuds peuvent également avoir des enfants (nœuds enfants) et des parents (nœuds parents) qui se trouvent respectivement en dessous et au-dessus d'eux dans l'arborescence.

En somme, le DOM tree est une représentation en arbre de la structure d'un document HTML ou XML, où chaque élément est représenté sous forme de nœud avec des propriétés et des liens avec les autres nœuds dans l'arbre. Il est constitué à sa base d'un node racine et de toute une série de nodes dits « *enfants* » (*child Nodes*) qui représentent les éléments, les attributs et les contenus de texte faisant partie de ce « *Document* ».

Les types de nodes

En programmation web, il existe plusieurs types de nœuds (ou nodes) dans le DOM (Document Object Model). Nous allons d'ailleurs voir encore les principaux types de nœuds avec un exemple pour chacun d'eux.

Élément (Element node)

Ce type de nœud est le plus courant dans le DOM, représentant un élément HTML. Par exemple, un élément `div` est représenté par un nœud de type `"Element"`.

Exemple

```
1 <div id="example">Ceci est un exemple</div>
```

Attribut (attribute node)

Un attribut permet de donner des informations supplémentaires concernant un élément HTML. Par exemple, une balise `` peut avoir un attribut `"alt"` permettant de donner une description pour les utilisateurs ne pouvant pas afficher l'image. En DOM, les attributs permettent au programmeur de manipuler les objets HTML en utilisant des langages de programmation tels que JavaScript. On peut par exemple utiliser des méthodes telles que `"getAttribute"` pour lire la valeur d'un attribut, `"setAttribute"` pour modifier la valeur d'un attribut, ou `"removeAttribute"` pour supprimer un attribut.

Exemple

```
1 
1 // Accéder à l'élément img
2 var img = document.getElementsByTagName("img")[0];
3
4 // Lire la valeur de l'attribut alt
5 var altValeur = img.getAttribute("alt");
```

Texte (text node)

Un nœud "texte" correspond tout simplement au texte contenu dans une balise. De la même façon que pour les attributs, un texte peut être modifié en utilisant la DOM.

Exemple

```
1 <p> Voici un court texte </p>
1 // Accéder à l'élément p
2 var p = document.getElementsByTagName("p")[0];
3
4 // Accéder au texte de l'élément p
5 var textNode = p.childNodes[0];
6
7 // Modifier le texte
8 textNode.nodeValue = "Voici un texte un peu plus long";
```

Commentaire (comment node)

Un *comment node* correspond à un commentaire HTML, qui n'est donc visible que dans le code par le programmeur, et qui permet de donner des informations sur le programme.

Exemple

```
1 <!-- Voici un comment node -->
```

Type de document (document type node)

Le document type node est tout simplement la balise spécifiant du type de document.

Exemple

```
1 <!DOCTYPE html>
```

Document (document node)

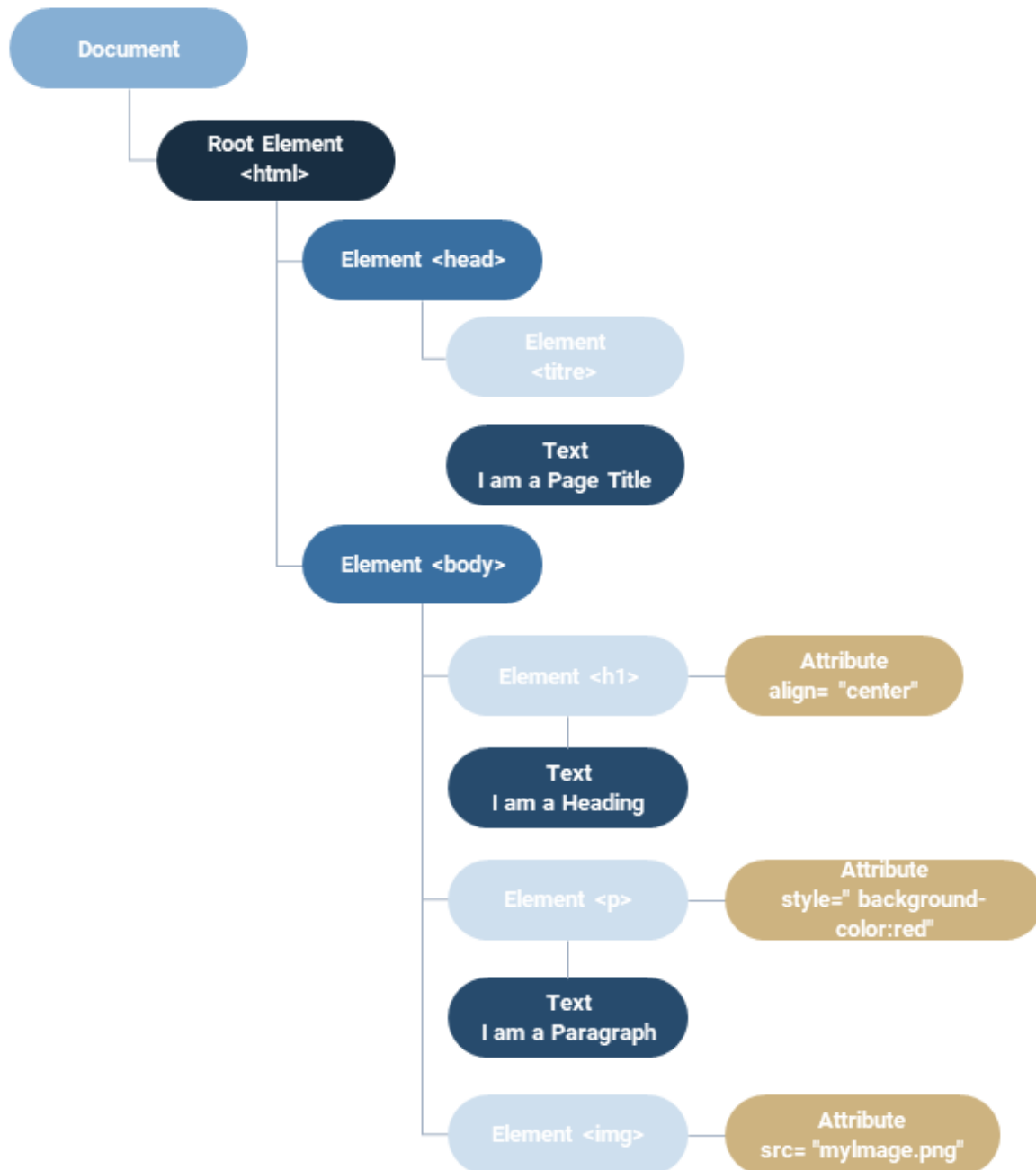
Le document node représente l'ensemble du document HTML ou XML, y compris tous les éléments, les attributs, les "text nodes", les "comment nodes" et autres nœuds le composant. Il est donc le nœud le plus haut dans la hiérarchie de la DOM.

Exemple

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Ceci est mon document node</title>
5   </head>
6   <body>
7     <h1>Titre de la page</h1>
8     <p>Ceci est un texte</p>
9   </body>
10 </html>
```


B. Structure d'un document

Ainsi, le DOM est structuré comme un arbre et, si on le compare à un arbre généalogique, le « *document* » est le Parent de tous les éléments et de tous les nodes enfants eux-mêmes contenant des éléments et des nodes qui sont leurs enfants et dont ils sont les parents.



Tous les éléments ont donc un node parent, excepté le node contenant, c'est-à-dire le « *document* ». Celui-ci est l'élément racine, il est parent de tous les autres. De même, tous les nodes ou éléments peuvent avoir zéro, un ou plusieurs nodes enfants. Des éléments se situant à un même niveau hiérarchique sont alors nommés des éléments frères (sibling).

Méthode Création de notre document

Nous allons créer notre premier document en suivant ces étapes :

- On commence par le choix du type de document. Par convention, on veut qu'une page web soit établie comme un document DOCTYPE. L'en-tête du document sera alors celle-ci : `<!DOCTYPE html>`.
- On pourra ensuite ajouter les balises html, nous permettant de spécifier la nature du document.
- Vient ensuite une partie `<head>` (tête du document) qui permettra de donner des informations sur le document, invisibles par l'utilisateur, au navigateur ou au moteur de recherche.
- Enfin, la balise `body`, correspondant au contenu de notre page.

Exemple

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <title>Hello World!</title>
6   </head>
7   <body></body>
8 </html>
```

C. Exercice : Quiz

[solution n°1 p.17]

Question 1

Le code source en HTML et le DOM sont la même chose.

- ☐ Vrai
- ☐ Faux

Question 2

Un nœud et un élément sont la même chose.

- ☐ Vrai
- ☐ Faux

Question 3

Le DOM génère une structure en arbre.

- ☐ Vrai
- ☐ Faux

Question 4

Dans le DOM, le nœud « *Document type* » est l'élément racine.

- ☐ Vrai
- ☐ Faux

Question 5

Les commentaires en HTML apparaissent dans le DOM.

- ☐ Vrai
- ☐ Faux

III. Introduction à la manipulation du DOM

A. Manipulation du DOM

L'objet document du DOM est fait pour être « *manipulé* ». Ce que l'on entend par là, c'est qu'il est possible de modifier ses différents composants par le biais du JavaScript.

On pourrait définir les grandes catégories du DOM de cette manière :

- Le DOM methods permet de manipuler le DOM tree,
- Le DOM events permet de créer des interactions client dans la page,
- Le DOM forms permet de préparer à la bascule de données côté serveur.

DOM contient une série de méthode et de propriétés qui sont le propre des objets, séries qui permettent d'interagir avec l'arborescence déployée.

On dit de ces méthodes et de ces propriétés qu'elles sont Natives parce qu'elles habitent déjà l'objet, elles sont déjà écrites dans l'objet, elles en font partie et sont à disposition du programmeur pour être utilisées. Elles sont comme des outils. Il est par exemple possible d'utiliser cette méthode : `document.getElementById()`.

Cette méthode existe déjà dans l'objet Dom et peut être invoquée par le programmeur dans son code en JavaScript. Et comme son nom l'indique, elle permet de pointer un élément précis du document en visant son identifiant (unique dans le document). C'est ce que nous verrons dans le chapitre « *manipuler le DOM* ».

Est désigné par le terme *DOM methods* (en français « méthodes DOM ») des fonctions au sein de l'Objet Document qui vont permettre aux développeurs d'interagir avec le DOM et de le manipuler. Les méthodes DOM peuvent avoir différentes fonctions comme accéder à chaque élément du document, accéder à des attributs, à des nœuds de textes, modifier le contenu du document, les attributs, le style, etc. En résumé, les méthodes sont un moyen de modifier le document de fond en comble.

Exemple Quelques exemples de DOM methods

`document.createElement()` : permet de créer un élément HTML dans le document.
`element.setAttribute(name, value)` : permet de donner un attribut à un élément déjà créé.

Rappel

La POO, ou « *Programmation Orientée Objet* », est un paradigme de programmation qui permet de modéliser des entités sous forme d'objets qui ont des propriétés et des méthodes. Elle repose sur des concepts clés tels que l'encapsulation, l'abstraction et l'héritage. La POO permet de structurer et d'organiser efficacement le code, offrant ainsi une meilleure maintenabilité et une plus grande flexibilité.

B. Comprendre le DOM et l'importance du DOM Inspector pour les développeurs Front-end

Les événements

Le DOM, en tant qu'objet, a au cœur de sa programmation une série de méthodes (des fonctions au sein d'un objet) et de propriétés (des valeurs variables ou constantes), agissant comme des outils qui pourront être utilisés pour modifier, retirer ou ajouter des éléments (elements) composant la page web.

L'un des aspects majeurs du développement Front-end est de fabriquer des interactions entre les actions de l'utilisateur qui parcourt la page, et le serveur de la société ou du propriétaire du site, en privilégiant l'optimisation de la performance et de l'expérience utilisateur.

C'est grâce au DOM et aux nombreuses améliorations qu'apporte celui-ci que les applications web et le web de manière générale ont explosé depuis les années 2000. Côté client, les conditions de la visite ont été grandement améliorées : la visite est plus facile, plus optimisée, et plus agréable. Le web s'est grandement développé grâce à son potentiel d'attractivité et son potentiel commercial.

Cette programmation autour de la page web va donc plus souvent consister à désigner certains éléments de la page et à les pointer pour les manipuler. Concrètement, le programmeur va associer à des éléments de notre document des événements précis (events), que cela soit un changement esthétique ou un recueil de données, et ainsi permettre aux utilisateurs d'interagir eux-mêmes avec les différents éléments du site.

Le DOM Inspector

Le DOM Inspector est un outil de développement web qui permet aux développeurs de visualiser et de modifier le DOM d'une page web. Grâce au DOM Inspector, les développeurs peuvent inspecter, naviguer et modifier la structure de la page web en temps réel, ce qui est particulièrement utile lors lorsque l'on est confronté à un problème de mise en page ou d'interactivité par exemple. Les développeurs peuvent également modifier les styles CSS en direct pour visualiser les effets immédiats sur la page, ce qui facilite grandement leur travail.

Le DOM Inspector est également utile pour identifier les éléments spécifiques d'une page web, comme les formulaires ou les tableaux, et pour comprendre la hiérarchie des éléments dans la page. Il permet également de comprendre comment les différents éléments interagissent entre eux (avec les DOM events).

En somme, le DOM Inspector est un outil essentiel pour les développeurs front-end qui leur permet d'inspecter et de modifier le code HTML, CSS et JavaScript d'une page web en temps réel pour détecter et régler des problèmes de mise en page et d'interactivité, et pour mieux comprendre la structure et le fonctionnement d'une page web.

C. Le DOM Forms

L'intérêt d'une page web dynamique, c'est bien sûr d'afficher des informations, mais c'est aussi de pouvoir en récolter, et de pouvoir communiquer des données d'un client à un serveur. De plus en plus de données sont transmises *via* les navigateurs notamment, de manière active (l'utilisateur donne ses informations directement) ou de manière passive ou transparente (à l'aide de cookies). Le développeur aura donc pour travail de mettre en place des méthodes afin de récupérer des données sur les visiteurs.

Une des meilleures façons de récupérer les données d'un visiteur de manière consentie est la mise en place d'un formulaire. Dans le DOM, on parle alors de DOM Forms. Le DOM Forms permettra, tout comme pour l'ensemble du site, de mettre en place des formulaires dynamiques. Il permet d'ajouter ou de supprimer des éléments de formulaire, de modifier les attributs des éléments existants, et d'ajouter des événements pour répondre aux interactions utilisateur.

Au niveau légal, la loi exige que le visiteur d'un site soit prévenu et informé du type d'information qui peut être récupéré de manière invisible et que le propriétaire du site pourra utiliser. C'est pour cela que, sur chaque site, une fenêtre demande à l'utilisateur d'accepter ou non les cookies utilisés par le site.

Le serveur

Un serveur, c'est une unité physique de stockage, comme un grand disque dur, qui permet de stocker des millions de données, et d'y accéder de manière sécurisée. C'est donc avec le serveur que le site doit faire un lien lorsqu'un utilisateur remplit un formulaire sur une page web.

Fonctionnement du DOM Forms

De nombreux sites proposent à leurs utilisateurs de remplir un formulaire. Ainsi, une catégorie des méthodes contenues dans le DOM est adaptée à la gestion de formulaires.

Lorsque des informations sont entrées dans un formulaire, elles sont ensuite basculées vers le serveur. Celui-ci va alors trier l'information, afin de la ranger dans la case qui lui correspond. Néanmoins, si une ligne entrée dans un formulaire se fait passer pour une instruction au serveur, cela peut avoir des conséquences néfastes (on parle alors d'injection).

Le DOM forms est donc équipé de méthodes, qui dès l'entrée des données côté client, trient le contenu de l'information, mais aussi le type de donnée, son format, sa syntaxe.

Le champ à remplir sera également « cerné » par un code qui filtre l'information entrée par le client. Aussi ces filtres qu'on appelle des « *expressions régulières* » (*Regular Expressions* ou RegEx en anglais) sont-ils contenus dans le DOM (par exemple, votre mot de passe doit contenir au moins 2 chiffres et 1 caractère spécial), mais peuvent-ils être encore des méthodes propres à la catégorie formulaire DOM forms.

Il est à noter qu'afin de pouvoir communiquer des données avec un serveur depuis une application web, le JavaScript n'est pas le langage adapté, et on privilégiera le PHP. Le JavaScript aura seulement l'objectif de rendre le formulaire dynamique pour les utilisateurs.

D. Exercice : Quiz

[solution n°2 p.17]

Question 1

Le HTML est un langage de typage.

- ☐ Vrai
- ☐ Faux

Question 2

Manipuler le DOM se fait principalement avec le langage JavaScript.

- ☐ Vrai
- ☐ Faux

Question 3

Le DOM contient des méthodes pour interagir avec les éléments.

- ☐ Vrai
- ☐ Faux

Question 4

Les formulaires d'une page web ne peuvent pas être modifiés grâce au DOM.

- ☐ Vrai
- ☐ Faux

Question 5

La programmation orientée objet consiste en l'écriture d'un programme de manière procédurale.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

En conclusion, la connaissance du *Document Object Model* est primordiale pour tout développeur d'applications web. Le DOM permet la manipulation de la structure, du style et du contenu des pages web de manière dynamique et interactive. Cela ouvre un large éventail de possibilités pour créer des applications web sophistiquées et modernes. En comprenant le fonctionnement du DOM, les développeurs peuvent améliorer l'expérience utilisateur et créer des interfaces utilisateur plus interactives.

V. Auto-évaluation

A. Exercice

Vous êtes embauché par l'équipe de sport de votre ville afin de moderniser leur site internet. Ce site sert de présentation à l'équipe, et leur permet de noter tous leurs événements. On vous demande de créer un site dynamique en utilisant la DOM.

Question 1

[solution n°3 p.18]

Décrivez une manipulation du DOM impliquant la création d'une liste de 5 joueurs (valeurs contenues dans un tableau) et la suppression ensuite d'un élément de cette liste.

Spécifiez comment créer les éléments, ensuite comment atteindre un des éléments, et finalement comment effacer un de ces éléments.

Question 2

[solution n°4 p.19]

Dans l'environnement `'repl.it'`, en utilisant des méthodes et propriétés du DOM, vous créez un élément contenant deux autres éléments.

Vous attribuez une classe identique à ces derniers qui leur définit une hauteur de 80 pixels et une largeur de 130 pixels.

Au premier de ces deux éléments vous attribuez un événement (clic de souris) qui déclenche l'affichage d'un texte dans son élément voisin.

B. Test

Exercice 1 : Quiz

[solution n°5 p.20]

Question 1

Aujourd'hui, le DOM est toujours différent selon le navigateur utilisé.

- ☐ Vrai
- ☐ Faux

Question 2

La méthode du DOM `.createElement()` crée un bloc de texte sur le document.

- ☐ Vrai
- ☐ Faux

Question 3

Le DOM permet de communiquer avec un serveur.

- ☐ Vrai
- ☐ Faux

Question 4

Dans l'arborescence du DOM, l'élément racine du document est la balise `<head>`.

- ☐ Vrai
- ☐ Faux

Question 5

Dans le DOM, les propriétés permettent de manipuler les Nodes et les méthodes permettent de manipuler les éléments.

- ☐ Vrai
- ☐ Faux


Solutions des exercices

Exercice p. 10 Solution n°1**Question 1**

Le code source en HTML et le DOM sont la même chose.

☐ Vrai

☒ Faux


 Le code source en HTML fournit une simple représentation de la page web sur une base de texte balisé. Le DOM lui, représente la page web par le biais d'un objet en JavaScript pour traduire le contenu en HTML du code source.

Question 2

Un nœud et un élément sont la même chose.

☐ Vrai

☒ Faux

 Les Nodes sont des parties génériques plus du DOM. Un élément est composé de Nodes. La balise de l'élément est un Node, l'attribut de ce même élément un autre Node.

Question 3

Le DOM génère une structure en arbre.

☒ Vrai

☐ Faux


 Le DOM déploie une structure hiérarchique en arborescence des éléments.

Question 4

Dans le DOM, le nœud « *Document type* » est l'élément racine.

☐ Vrai

☒ Faux


 L'élément racine est le « *Document* » du DOM, il est parent de tous les éléments qui composeront la page.

Question 5

Les commentaires en HTML apparaissent dans le DOM.

☒ Vrai

☐ Faux

 Dans le node, les commentaires sont des *comment nodes*.


Exercice p. 13 Solution n°2

Question 1

Le HTML est un langage de typage.

☐ Vrai

☒ Faux


 HTML est un langage informatique de marquage. Il sert à décrire le formatage et la structuration de documents en « *base-texte* », comme une page web ou encore un document informatique établi pour l'impression. Les langages de marquage font usage de balises et de codes qui indiquent de quelle manière un document doit être agencé, formaté ou établi de manière graphique.

Question 2

Manipuler le DOM se fait principalement avec le langage JavaScript.

☒ Vrai

☐ Faux


 Le DOM se manipule avec le JavaScript. JavaScript est un langage de script adapté à la manipulation des éléments déployés dans le DOM d'une page web pour une programmation basée sur l'interaction dans un environnement web. Le DOM est lui-même un objet composé en JavaScript.

Question 3

Le DOM contient des méthodes pour interagir avec les éléments.

☒ Vrai

☐ Faux


 Avec le DOM, il est possible d'interagir avec des éléments du document. Par exemple, il est possible d'en créer de nouveaux, ou d'en supprimer, ou d'en modifier.

Question 4

Les formulaires d'une page web ne peuvent pas être modifiés grâce au DOM.

☐ Vrai

☒ Faux


 Le DOM peut permettre d'apporter des modifications aux formulaires, de les rendre plus sécurisés et plus interactifs. On parle de « *DOM Forms* ».

Question 5

La programmation orientée objet consiste en l'écriture d'un programme de manière procédurale.

☐ Vrai

☒ Faux

 La programmation orientée objet est un paradigme de programmation qui consiste à organiser le code en objets interagissant. Les objets ont des attributs (variables) et des méthodes (fonctions) qui leur permettent de manipuler leurs données et de communiquer avec les autres objets.

```

1 <ul id="maListe">
2   <li>Tom</li>
3   <li>Stéphanie</li>
4   <li>Pierre</li>
5   <li>Paul</li>
6   <li>Maria</li>
7 </ul>

```

Ou

```

1 // creation du tableau contenant les 5 valeurs
2 var joueurs = ["Tom","Stéphanie","Pierre","Paul","Maria"] ;
3 // creation d'un élément ul > contenant de liste
4 maListe=document.createElement('ul');
5 //creation d'un identifiant pour ce contenant
6 maListe.id='liste' ;
7 // élément est attaché au corps du document
8 document.body.appendChild(maListe) ;
9 //creation des elements de la liste avec une boucle
10 for (var n in joueurs){
11   monItem=document.createElement('li');
12   monItem.innerHTML=joueurs[n];
13   maListe.appendChild(monItem) ;
14 }
15
16 // récupération de l'identifiant du contenant de la liste dans une variable
17 var monIdListe = document.getElementById('liste')
18 // pointage sur le dernier élément de la liste
19 monDernier=monIdListe.lastChild;
20 //enlèvement de l'élément pointé
21 monDernier.remove();

```

p. 14 Solution n°4

Voici la réponse attendue :

```

1 // Création d'une div contenante
2 var contenant=document.createElement('div');
3 // aspect du contenant brièvement visuellement défini
4 contenant.style="height:200px;width:300px;border:1px solid black;display:flex; justify-
   content:space-around"
5 //contenant est attaché au corps du document
6 document.body.appendChild(contenant);
7
8 // une div A contenue dans le contenant est créée
9 var contenu_A=document.createElement('div');
10 // son aspect visuel brièvement défini
11 contenu_A.style="height:190px;width:100px;border:1px solid blue"
12 // elle est attachée comme enfant au contenant
13 contenant.appendChild(contenu_A);
14 // une div B contenue dans le contenant est créée
15 var contenu_B=document.createElement('div');
16 // son aspect visuel brièvement défini
17 contenu_B.style="height:190px;width:100px;border:1px solid red"
18 // elle est attachée comme enfant au contenant
19 contenant.appendChild(contenu_B);
20
21 // ajout d'un événement sur la div A associé à une fonction
22 contenu_A.addEventListener('click', function(e) {

```

```
23 // une fonction qui ajoute du texte quand l'utilisateur clique
24 contenu_B.innerHTML="Le texte est ajouté dans l'élément";
25 });
```

Exercice p. 14 Solution n°5

Question 1

Aujourd'hui, le DOM est toujours différent selon le navigateur utilisé.

- ☐ Vrai
- ☒ Faux

Q Le DOM l'était certes à ses débuts quand l'outil s'est développé au sein d'entreprises différentes, mais aujourd'hui un consortium international (le W3C) a établi des normes et des standards qui font que les navigateurs de différentes marques déploient tous un DOM similaire permettant une compatibilité entre les différents navigateurs.

Question 2

La méthode du DOM `.createElement()` crée un bloc de texte sur le document.

- ☐ Vrai
- ☒ Faux

Q La méthode `.createElement()` permet de créer un élément nouveau dans le DOM. Il devra d'ailleurs être spécifié, entre les parenthèses qui demandent un paramètre, quel élément sera créé. Pour une division de la page `<div></div>`, on écrira `.createElement('div')`; pour un titre `<h2></h2>`, on écrira `.createElement('h2')`, etc. L'élément créé n'est par contre pas encore attaché à la page, donc n'apparaîtra pas. Une seconde instruction utilisant la méthode `.appendChild()` sera nécessaire pour spécifier à quel autre élément du document l'élément créé s'attachera.

Question 3

Le DOM permet de communiquer avec un serveur.

- ☐ Vrai
- ☒ Faux

Q Le DOM n'est pas un outil de communication avec le serveur. Un langage comme le php est plus approprié et permettra également un lien avec le DOM.

Question 4

Dans l'arborescence du DOM, l'élément racine du document est la balise `<head>`.

- ☐ Vrai
- ☒ Faux

Q La balise `<html>` est l'élément racine du document. La balise `<html>` est un élément, pas un node.

Le Node parent dans le document est le « Document lui-même, avec comme node enfant le DOCTYPE qui lui sert de préambule précisant quelles spécifications et quels standards le document va suivre ».


Il est à noter que le DOM peut aussi être défini entre les balises `<xml>` `</xml>` ouvrant alors sa rédaction aux balises propres au marquage XML.

Question 5

Dans le DOM, les propriétés permettent de manipuler les Nodes et les méthodes permettent de manipuler les éléments.

☐ Vrai

☒ Faux

 Le DOM contient des méthodes pour manipuler les éléments autant que les Nodes. Le DOM contient également des propriétés permettant d'agir sur les éléments autant que sur les Nodes. Une méthode est comparable à une fonction.

La propriété (*Property*) est comparable à un espace d'allocation de données (variables ou constantes).