Partie 2 - Modeler vos pages Web
Manipuler le code HTML

Le Document Object Model
Le Document Object Model (abrégé DOM) est une interface de programmation pour les documents XML et HTML.

L'objet window
L'objet window est ce qu'on appelle un objet global qui représente la fenêtre du navigateur. C'est à partir de cet objet que le Javascript est exécuté.

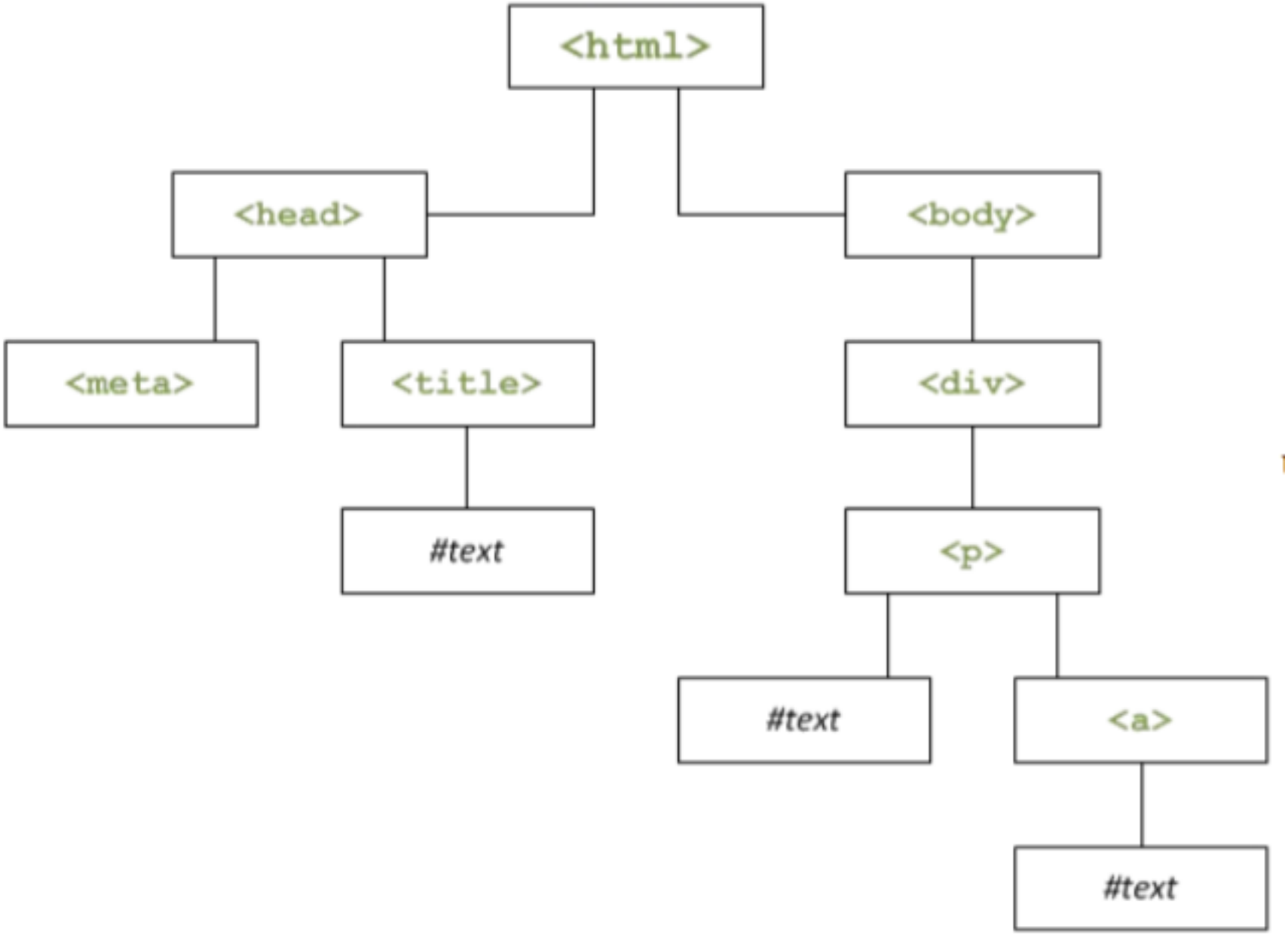
Code : HTML
```
<!DOCTYPE html>
<html>
	<head>
		<title>Hello World!</title>
	</head>
	<body>
		<script>
			alert('Hello world!');
		</script>
	</body>
</html>
```

Code : JavaScript
```
window.alert('Hello world!');
alert('Hello world!');
```

alert() est une méthode. Elle appartient à l'objet window. L'objet window est dit implicite, il n'a pas besoin d'être spécificier.

Le document
L'objet document est un sous-objet de window, l'un des plus utilisés.

Naviguer dans le document
La structure DOM



Code : HTML

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8" />
		<title>Le titre de la page</title>
	</head>
	<body>
		<div>
			<p>Un peu de texte et <a>un lien</a></p>
		</div>
	</body>
</html>
```

Le schéma est plutôt simple : l'élément <html> contient deux éléments, appelés enfants : <head> et <body>. Pour ces deux enfants, <html> est l'élément parent. Chaque élément est appelé nœud (node en anglais). L'élément <head> contient lui aussi deux enfants : <meta> et <title>. <meta>ne contient pas d'enfant tandis que <title>en contient un, qui s'appelle #text. Comme son nom l'indique, #text est un élément qui contient du texte.

Accéder aux éléments

L'objet document possède trois méthodes principales : getElementById(), getElementsByTagName() et getElementsByName().

getElementById()

Cette méthode permet d'accéder à un élément en connaissant son ID qui est simplement l'attribut id de l'élément.

Code : HTML

```
<div id="myDiv">
	<p>Un peu de texte et <a>un lien</a></p>
</div>
<script>
	var div = document.getElementById('myDiv');
	alert(div);
</script>
```

getElementsByName()
Cette méthode est semblable à getElementsByTagName() et permet de ne récupérer que les éléments qui possèdent un attribut name que vous spécifiez.

Accéder aux éléments grâce aux technologies récentes

On va étudier deux méthodes. La première, **querySelector()**, renvoie le premier élément trouvé correspondant au sélecteur CSS, tandis que **querySelectorAll()** va renvoyer tous les éléments (sous forme de tableau) correspondant au sélecteur CSS fourni.

Code : CSS
```
#menu .item span {
}
```

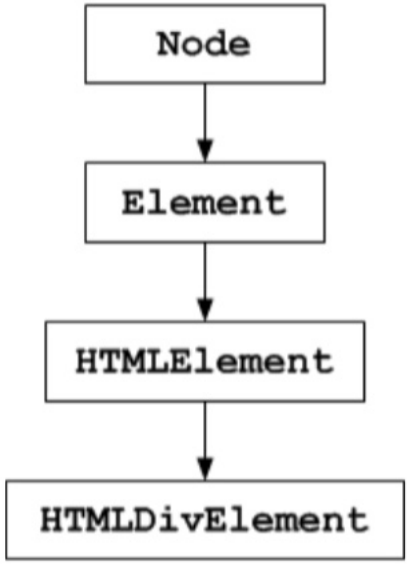
Code : HTML
```
<div id="menu">
	<div class="item">
		<span>Élément 1</span>
		<span>Élément 2</span>
	</div>
	<div class="publicite">
		<span>Élément 3</span>
		<span>Élément 4</span>
	</div>
</div>
<div id="contenu">
	<span>Introduction au contenu de la page...</span>
</div>
```

On va utiliser la propriété nommée **innerHTML** qui permet d'accéder au contenu d'un élément HTML.
Code : JavaScript

```
var query = document.querySelector('#menu .item span'),
    queryAll = document.querySelectorAll('#menu .item span');
alert(query.innerHTML); // Affiche : "Élément 1"
alert(queryAll.length); // Affiche : "2"
alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML); // Affiche : "Élément 1 - Élément 2"
```

L'héritage des propriétés et des méthodes

Notion d'héritage
En Javascript, un objet peut appartenir à plusieurs groupes.



L'objet Node apporte un certain nombre de propriétés et de méthodes qui pourront être utilisées depuis un de ses sous-objets. En clair, les sous-objets héritent des propriétés et méthodes de leurs objets parents.

Voilà donc ce que l'on appelle l'héritage.

Éditer les éléments HTML

Les attributs

Via l'objet Element
Code : HTML

```
<body>
	<a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié dynamiquement</a>
	<script>
		var link = document.getElementById('myLink');
		var href = link.getAttribute('href'); // On récupère l'attribut “href “
		alert(href);
		link.setAttribute('href', 'http://www.viaformation.com'); // On édite l'attribut “href”
	</script>
</body>
```

On commence par récupérer l'élément myLink, et on lit son attribut href via la methode **getAttribute()**.
Ensuite on modifie la valeur de l'attribut href avec la méthode **setAttribute()**. Le lien pointe maintenant vers http://www.viaformation.com.

Les attributs accessibles

Pour la plupart des éléments courants comme <a>, il est possible d'accéder à un attribut via une propriété. Dans le cas où on veut modifier la destination d'un lien, on peut utiliser la propriété **href**.

Code : HTML

```
<body>
	<a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié dynamiquement</a>
	<script>
		var link = document.getElementById('myLink');
		var href = link.href;
		alert(href);
		link.href = 'http://www.viaformation.com';
	</script>
</body>
```

La classe

Code : HTML

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8" />
		<title>Le titre de la page</title>
		<style type="text/css">
			.blue {
				background: blue;
				color: white; }
		</style>
	</head>
	<body>
		<div id="myColoredDiv">
			<p>Un peu de texte et <a>un lien</a></p>
		</div>
		<script>document.getElementById('myColoredDiv').className = 'blue'; </script>
	</body>
</html>
```

Dans cet exemple, on définit la classe CSS .blue à l'élément **myColoredDiv**, ce qui fait que cet élément sera affiché avec un arrière-plan bleu et un texte blanc.

Le contenu : innerHTML

Récupérer du HTML

innerHTML permet de récupérer le code HTML enfant d'un élément sous forme de texte.

Code : HTML

```
<body>
	<div id="myDiv">
		<p>Un peu de texte et <a>un lien</a></p>
	</div>
	<script>
		var div = document.getElementById('myDiv');
		alert(div.innerHTML);    //affiche <p>Un peu de texte et <a>un lien</a></p>
	</script>
</body>
```

Ajouter ou éditer du HTML

Code : JavaScript

```
document.getElementById('myDiv').innerHTML = '<blockquote>Je mets une citation à la place du paragraphe</blockquote>';
```

innerText

Le fonctionnement de la propriété innerText est le même que la propriété innerHTML, excepté le fait que seul le texte est récupéré, et non les balises.

Code : HTML

```
<body>
	<div id="myDiv">
		<p>Un peu de texte <a>et un lien</a></p>
	</div>
	<script>
		var div = document.getElementById('myDiv');
		alert(div.innerText);    //affiche Un peu de texte et un lien
	</script>
</body>
```

textContent

Tester le navigateur

Il est possible via une simple condition de tester si le navigateur prend en charge telle ou telle méthode ou propriété.

Code : HTML

```
<body>
	<div id="myDiv">
		<p>Un peu de texte <a>et un lien</a></p>
	</div>
	<script>
		var div = document.getElementById('myDiv');
		var txt = '';
		if (div.textContent) { // « textContent » existe ? Alors on s'en sert !
			txt = div.textContent;
		} else if(div.innerText) { // « innerText » existe ? Alors on doit être sous IE.
			txt = div.innerText + ' [via Internet Explorer]';
		} else { // Si aucun des deux n'existe, cela est sûrement dû au fait qu'il n'y a pas de texte
			txt = ''; // On met une chaîne de caractères vide }
		alert(txt);
	</script>
</body>
```

En résumé

- Le DOM va servir à accéder aux éléments HTML présents dans un document afin de les modifier et d'interagir avec eux.
- L'objet window est un objet global qui représente la fenêtre du navigateur. document, quant à lui, est un sous-objet de window et représente la page Web. C'est grâce à lui que l'on va pouvoir accéder aux éléments HTML de la page Web.
- Les éléments de la page sont structurés comme un arbre généalogique, avec l'élément <html> comme élément fondateur.
- Différentes méthodes, comme getElementById(), getElementsByTagName(), querySelector() ou querySelectorAll(), sont disponibles pour accéder aux éléments.
- Les attributs peuvent tous être modifiés grâce à setAttribute(). Certains éléments possèdent des propriétés qui permettent de modifier ces attributs.
- La propriété innerHTML permet de récupérer ou de définir le code HTML présent à l'intérieur d'un élément.
- De leur côté, textContent et innerText ne sont capables que de définir ou récupérer du texte brut, sans aucunes balises HTML.

Les événements

La théorie

Liste des événements

Voici la liste des événements principaux, ainsi que les actions à effectuer pour qu'ils se déclenchent :

Nom de l'événement	Action pour le déclencher
click	Cliquer (appuyer puis relâcher) sur l'élément
dblclick	Double-cliquer sur l'élément
mouseover	Faire entrer le curseur sur l'élément
mouseout	Faire sortir le curseur de l'élément
mousedown	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
mouseup	Relâcher le bouton gauche de la souris sur l'élément
mousemove	Faire déplacer le curseur sur l'élément
keydown	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
keyup	Relâcher une touche de clavier sur l'élément
keypress	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
focus	« Cibler » l'élément
blur	Annuler le « ciblage » de l'élément
change	Changer la valeur d'un élément spécifique auxformulaires (input,checkbox, etc.)
select	Sélectionner le contenu d'un champ de texte (input, textarea, etc.)

Il existe aussi deux événements spécifiques à l'élément <form>, que voici :

Nom de l'événement	Action pour le déclencher
submit	Envoyer le formulaire
reset	Réinitialiser le formulaire

La pratique

Utiliser les événements

Prenons l'événement click sur un simple :

Code : HTML

```
<span onclick="alert('Hello !');">Cliquez-moi !</span>
```

Les événements au travers du DOM

Toujours avec l'événement click :

Code : HTML

```
<span id="clickme">Cliquez-moi !</span>
<script>
var element = document.getElementById('clickme');
element.addEventListener('click', function() {
        alert("Vous m'avez cliqué !");
        }, false);
 </script>
```

Nous utilisons une méthode nommée **addEventListener()** qui prend trois paramètres :

- Le nom de l'événement (sans les lettres « on ») ;
- La fonction à exécuter ;
- Un booléen pour spécifier si l'on souhaite utiliser la phase de capture ou bien celle de bouillonnement.
  Nous expliquerons ce concept un peu plus tard dans ce chapitre. Sachez simplement que l'on utilise généralement la valeur false pour ce paramètre.

Les phases de capture et de bouillonnement

Ces deux phases sont deux étapes distinctes de l'exécution d'un événement. La première, la capture (capture en anglais), s'exécute avant le déclenchement de l'événement, tandis que la deuxième, le bouillonnement (bubbling en anglais), s'exécute après que l'événement a été déclenché.

L'objet Event

Code : JavaScript

```
element.addEventListener('click', function(e) { // L'argument « e » va récupérer une référence vers l'objet « Event »
        alert(e.type); // Ceci affiche le type de l'événement (click, mouseover, etc.)
        }, false);
```

Les fonctionnalités de l'objet Event

Récupérer la position du curseur
La position du curseur est une information très importante, beaucoup de monde s'en sert pour de nombreux scripts comme le drag & drop.

Code : HTML
```
<div id="position"></div>
<script>
	var position = document.getElementById('position');
	document.addEventListener('mousemove" function(e) {
	position.innerHTML = 'Position X : ' + e.clientX + 'px<br/>Position Y : ' + e.clientY + 'px';
	}, false);
</script>
```

Récupérer les touches frappées par l'utilisateur

Code : HTML
```
<p>
	<input id="field" type="text" />
</p>
<table>
	<tr>
		<td id="down"></td>
	</tr>
	<tr>
		<td>keypress</td>
		<td id="press"></td>
	</tr>
	<tr>
		<td>keyup</td>
		<td id="up"></td>
	</tr>
</table>
```

```
<script>
	var field = document.getElementById('field'),
	    down = document.getElementById('down'),
	    press = document.getElementById('press'),
	    up = document.getElementById('up');
	document.addEventListener('keydown" function(e) {
	    down.innerHTML = e.keyCode;
	}, false);
	document.addEventListener('keypress', function(e) {
	    press.innerHTML = e.keyCode;
	}, false);
	document.addEventListener('keyup', function(e) {
	    up.innerHTML = e.keyCode;
	}, false);
</script>
```

En résumé

- Les événements sont utilisés pour appeler une fonction à partir d'une action produite ou non par l'utilisateur.
- Différents événements existent pour détecter certaines actions comme le clic, le survol, la frappe au clavier et le contrôle des champs de formulaires.
- Le DOM introduit l'objet Event et la fameuse méthode addEventListener().
- L'objet Event permet de récolter toutes sortes d'informations se rapportant à l'événement déclenché : son type, depuis quel élément il a été déclenché, la position du curseur, les touches frappées...

Les formulaires

Les propriétés

Il est possible d'utiliser les propriétés spécifiques aux éléments d'un formulaire comme **value, disabled, checked, etc.**

Une propriété classique : value

Code : HTML

```
<input id="text" type="text" size="60" value="Vous n'avez pas le focus !" />
<script>
	var text = document.getElementById('text');
	text.addEventListener('focus', function(e) {
		e.target.value = "Vous avez le focus !";
	}, true);
	text.addEventListener('blur', function(e) {
		e.target.value = "Vous n'avez pas le focus !";
	}, true);
</script>
```

Les booléens avec disabled, checked et readonly

Code : HTML

```
<input id="text" type="text" />
<script>
	var text = document.getElementById('text');
	text.disabled = true;
</script>
```

Code : HTML

```
<label><input type="radio" name="check" value="1" /> Case n° 1</label><br />
<label><input type="radio" name="check" value="2" /> Case n° 2</label><br />
<label><input type="radio" name="check" value="3" /> Case n° 3</label><br />
<label><input type="radio" name="check" value="4" /> Case n° 4</label>
<br /><br />
<input type="button" value="Afficher la case cochée" onclick="check();" />
```

```
<script>
function check() {
	var inputs = document.getElementsByTagName('input'),
	    inputsLength = inputs.length;
	for (var i = 0 ; i < inputsLength ; i++) {
		if (inputs[i].type == 'radio' && inputs[i].checked) {
			alert('La case cochée est la n°'+ inputs[i].value);
		}
	}
}
</script>
```

Les listes déroulantes avec selectedIndex et options

Les listes déroulantes possèdent elles aussi leurs propres propriétés. Nous allons en retenir seulement deuxparmi toutes celles qui existent : **selectedIndex** , qui nous donne l'index (l'identifiant) de la valeur sélectionnée, et **options** qui liste dans un tableau les éléments <option> de notre liste déroulante.

Code : HTML

```
<select id="list">
	<option>Sélectionnez votre genre</option>
	<option>Homme</option>
	<option>Femme</option>
</select>
<script>
	var list = document.getElementById('list');
	list.addEventListener('change', function() {
	// On affiche le contenu de l'élément <option> ciblé par la propriété selectedIndex
	alert(list.options[list.selectedIndex].innerHTML);
	}, true);
</script>
```

En résumé

- La propriété value s'emploie sur la plupart des éléments de formulaire pour en récupérer la valeur.
- Les listes déroulantes fonctionnent différemment, puisqu'il faut d'abord récupérer l'index de l'élément sélectionné avec selectedIndex.

Manipuler le CSS
Quelques rappels sur le CSS

Code : HTML
```
<style type="text/css">
	div {
		color: red;
	 }
</style>
<div style="color:blue;">Le texte n'est pas rouge mais bleu.</div>
```

Éditer les styles CSS d'un élément

Code : HTML
```
<div id="myDiv" style="background-color: orange">
	Je possède un fond orange.
</div>
<script>
	var myDiv = document.getElementById('myDiv');
	alert('Selon le Javascript, la couleur de fond de ce DIV est : ' + myDiv.style.backgroundColor); // On affiche la couleur de fond orange
</script>
```

Récupérer les propriétés CSS
La fonction getComputedStyle()

Code : HTML
```
<style type="text/css">
#text {
	color: red;
}
</style>

<span id="text">La couleur du texte est rouge.</span>

<script>
	var text = document.getElementById('text'),
	    color = getComputedStyle(text).color;
	alert(color);
</script>
```

En résumé

- Pour modifier les styles CSS d'un élément, il suffit d'utiliser la propriété style. Il ne reste plus qu'à accéder à la bonne propriété CSS, par exemple : element.style.height = '300px'.
- Le nom des propriétés composées doit s'écrire sans tiret et avec une majuscule pour débuter chaque mot, à l'exception du premier. Ainsi, **border-radius** devient borderRadius.
- La fonction getComputedStyle() récupère la valeur de n'importe quelle propriété CSS. C'est utile, car la propriété style n'accède pas aux propriétés définies dans la feuille de style.