INTRODUCTION AU JAVASCRIPT

LES BASES DE JAVASCRIPT

Pr. Youssouf EL ALLIOUI
youssouf.elallioui@uhp.ac.ma

Plan

Partie 1

Les bases du JavaScript

Partie 2

Modeler vos pages Web

Partie 2 : Modeler vos pages Web

Objectif

- Nous allons étudier dans cette partie la modification de la structure d'une page web

Menu

1. Manipuler le code HTML
2. Les événements
3. Les formulaires
4. Manipuler le CSS
5. TP : un formulaire interactif

① Manipuler le code HTML

Objectif

Dans cette partie, nous allons voir :

- **L'API DOM ?**
- **Comment naviguer entre les différents nœuds qui composent une page HTML ?**
- **l'édition du contenu d'une page :**
  - **Ajouter** des nœuds
  - **Supprimer** des nœuds
  - **Modifier** des nœuds

① Manipuler le code HTML

❑ Plan

1. **Le Document Object Model** (abrégé **DOM**)
2. **Naviguer dans le document**
3. **Éditer les éléments HTML**
4. **innerText et textContent**
5. **Naviguer entre les nœuds**
6. **Créer et insérer des éléments**
7. **Notions sur les références**
8. **Cloner, remplacer, supprimer…**
9. **Autres actions**
10. **Mini-Projet : recréer une structure DOM**

① Manipuler le code HTML

Le Document Object Model (abrégé DOM)

- **Le DOM est une API (Interface de programmation) pour les documents XML et HTML**
- **Il permet, via le Javascript, d'accéder au code XML et/ou HTML d'un document**
- **Il permet également de modifier des éléments (balises) HTML (ajouter, déplacer ou même supprimer)**

① Manipuler le code HTML

❑ Le Document Object Model (abrégé DOM)

◆ Petite historique

Le DOM n'est pas unifié

- Il dépend de navigateur

W3C : DOM-1

- Définit ce qu'est le DOM
- Définit comment un document HTML ou XML est schématisé

W3C : DOM-2

- Introduction de la méthode getElementById()

① Manipuler le code HTML

❑ Le Document Object Model (abrégé DOM)

◆ L'objet window

- ❖ Il s'agit un objet global qui représente *la fenêtre du navigateur*
- ❖ C'est à partir de cet objet que le Javascript est exécuté
- ❖ Toute variable déclarée dans le contexte global de script est une variable de l'objet window
- ❖ Toute variable non déclarée (sans le mot-clé var) deviendra immédiatement une propriété de l'objet window

① Manipuler le code HTML

❑ Le Document Object Model (abrégé DOM)

◆ L'objet document

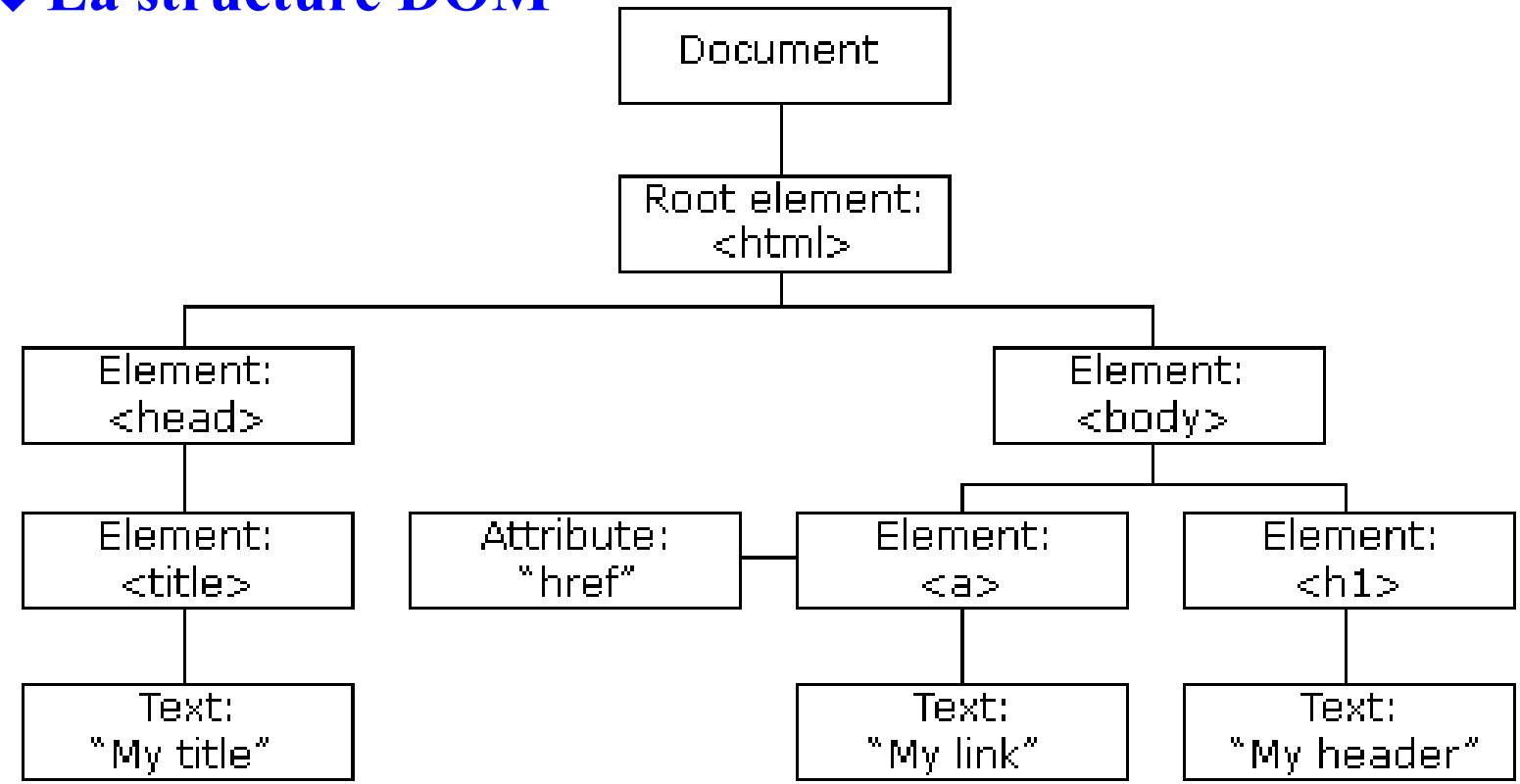
- ❖ l'objet **document** est un sous-objet de l'objet **window** l'un des plus utilisé
- ❖ **Il représente** la page web et plus précisément la balise **<html>**
- ❖ C'est grâce à cet élément-là que nous allons pouvoir **accéder aux éléments HTML** et les modifier

Voyons donc, dans la sous-partie suivante, comment naviguer dand le document

① Manipuler le code HTML

❑ Naviguer dans le document

◆ La structure DOM

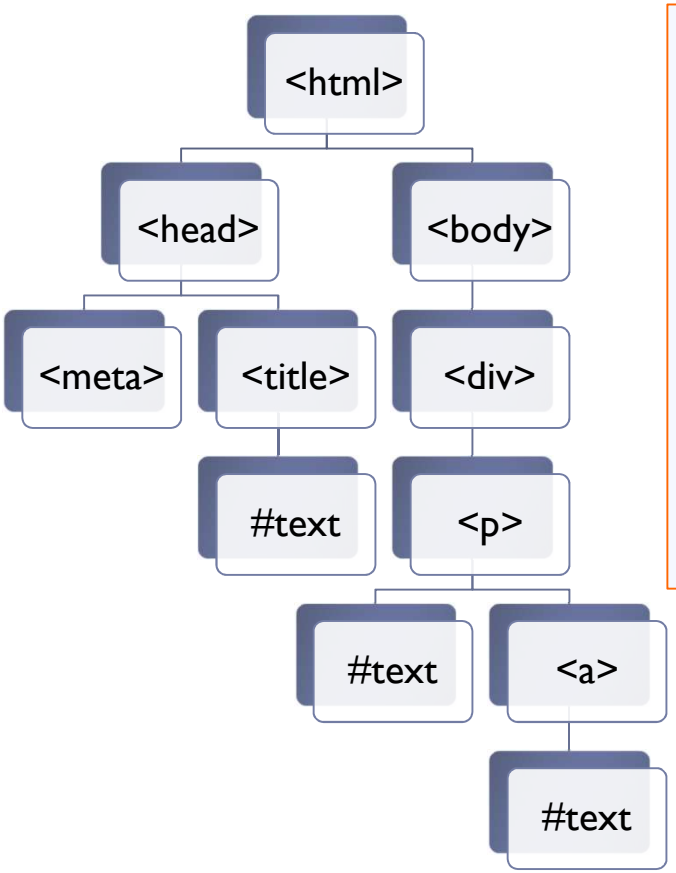


[Réf. Site de W3C]

① Manipuler le code HTML

❑ Naviguer dans le document

◆ La structure DOM




```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
  </head>

  <body>
    <div>
      <p>Un peu de texte <a>et un lien</a></p>
    </div>
  </body>
</html>
```

- Le DOM pose comme concept que la page Web est vue comme un arbre, comme une hiérarchie d'éléments
- le texte présent dans une page Web est vu par le DOM comme un nœud de type #text

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Accèder aux éléments

- L'accès aux éléments HTML via le **DOM** est assez simple mais demeure actuellement plutôt limité.
- L'objet **document** possède 5 méthodes principales :

Accès simple	
getElementByID ()	Permet d'accéder à un élément en connaissant son ID qui est simplement l'attribut id de l'élément.
getElementsByTagName ()	Permet de récupérer, sous la forme d'un tableau, tous les éléments de la famille (Exp :)
getElementsByName ()	Permet de récupérer les éléments qui possèdent un attribut name que vous spécifiez
Accès grâce aux selecteurs CSS (Nouveau)	
querySelector ()	❑ Méthodes récentes et ne sont pas supportées par les vieilles versions des navigateurs ❑ Elles prennent pour paramètre un seul argument : une chaîne de caractères ! ❑ Cette chaîne de caractères doit être un sélecteur CSS
querySelectorAll ()	

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Accèder aux éléments

□ Exemple : getElementById ()

```
<div id="myDiv">
  <p>Un peu de texte <a>et un lien</a></p>
</div>

<script>
  var div = document.getElementById('myDiv');

  alert(div);
</script>
```

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Accèder aux éléments

□ Exemple : getElementsTagName () / getElementsByName ()

```
var divs = document.getElementsByTagName('div');

for (var i = 0, c = divs.length ; i < c ; i++) {
    alert('Element n° ' + (i + 1) + ' : ' + divs[i]);
}
```

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Accèder aux éléments

□ Exemple : querySelector () / querySelectorAll ()

```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>
```

Code HTML

```css
#menu .item span
```

Code CSS

```js
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"
```

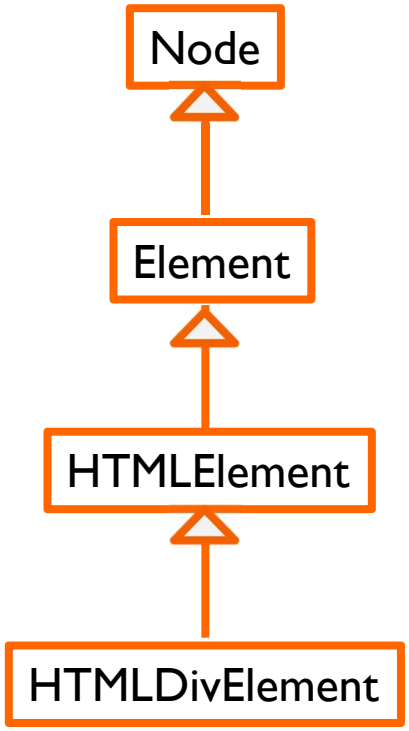
Code JS

① Manipuler le code HTML

❑ Naviguer dans le document

◆ L'héritage des propriétés et des méthodes

- Le Javascript voit les éléments HTML comme étant des objets
- Chaque élément HTML possède des propriétés et des méthodes
- Tous les éléments HTML sont d'un même type : le type **Node**
- Les sous-objets *héritent* des propriétés et méthodes de leurs objets parents



① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

- Maintenant que nous avons vu comment accéder à un élément, nous allons voir comment l'éditer.
- Les éléments HTML sont souvent composés d'**attributs** (Expl'attribut href d'un **<a>**), et d'un **contenu**, qui est de type #text
- Le contenu peut aussi être un autre élément HTML

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer les attributs

- l'objet **Element** nous fournit deux méthodes, **getAttribute ()** et **setAttribute ()** permettant respectivement de **récupérer** et d'**éditer** un attribut

```
8   <body>
9
10      <a id="myLink" href="http://www.usmba.ac.ma">Un lien modifié dynamiquement</a>
11      <script>
12          var link = document.getElementById('myLink');
13          // On récupère l'attribut « href »
14          var href = link.getAttribute('href');
15          alert(href);
16          // On édite l'attribut « href »
17          link.setAttribute('href', 'http://www.uh1.ac.ma');
18      </script>
19
20  </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer les attributs

- Pour la plupart des éléments courants comme **<a>, il est possible d'accéder à un attribut via une propriété**

```
 8  <body>
 9
10      <a id="myLink" href="http://www.usmba.ac.ma">Un lien modifie dynamiquement</a>
11      <script>
12          var link = document.getElementById('myLink');
13          // On récupére l'attribut « href »
14          var href = link.href;
15          alert(href);
16          // On édite l'attribut « href »
17          link.href = 'http://www.uh1.ac.ma';
18      </script>
19
20  </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer les attributs

- pour récupérer ou modifier la valeur de certain attribut (**class**, **for** pour les labels, …), **l'utilisation de la propriété est impossible**
- Les mots-clés **class**, **for**, **…** sont réservés en Javascript
- À la place de **class** et **for**, il faudra utiliser **className** et **htmlFor**.

```
<!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 <a>et un lien</a></p> </div>
        <script>
            document.getElementById('myColoredDiv').className = 'blue';
        </script>
    </body></html>
```

Essayer

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer le contenu

- **3 propriétés** pour modifier **le contenu** des éléments (balises) HTML :
  - **innerHTML** : Pour tous les navigateur (**normalisée au sein du HTML 5**).
  - **innerText** : Just pour Internet Explorer
  - **textContent** : Pour les navigateurs autres que Internet Explorer

- 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.

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer le contenu

□ innerHTML

```
8  <body>
9
10     <div id="myDiv">
11
12     <p>Un peu de texte <a>et un lien</a></p> </div>
13
14     <script>
15         var div = document.getElementById('myDiv');
16         alert(div.innerHTML);
17
18         document.getElementById('myDiv').innerHTML = '<blockquote>Je mets une citation ' +
19                                                      'à la place du paragraphe</blockquote>';
20     </script>
21
22 </body>
23 </html>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Naviguer dans le document

◆ Editer les éléments HTML

❖ Editer le contenu

- □ **innerText & textContent** (Avec test de navigateur)

```
8  <body>
9
10     <div id="myDiv">
11     <p>Un peu de texte <a>et un lien</a></p> </div>
12
13     <script>
14         var div = document.getElementById('myDiv');
15         var txt = '';
16         if (div.textContent) { // « textContent » existe ? Alors on s'en sert !
17             txt = div.textContent;
18         } else if(div.innerText) { // « innerText » existe ? Alors on doit être sous IE.
19             txt = div.innerText + ' [via Internet Explorer]';
20         } else { // Si aucun des deux n'existe, cela est suûrement du au fait qu'il n'y a pas de texte
21             txt = ''; // On met une chaîne de caractères vide
22         }
23         alert(txt);
24     </script>
25
26 </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Résumé

- Le DOM va servir à accéder aux éléments HTMLprésents 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.
- L'objet **document** 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()**, pour accéder aux différents 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.

① Manipuler le code HTML

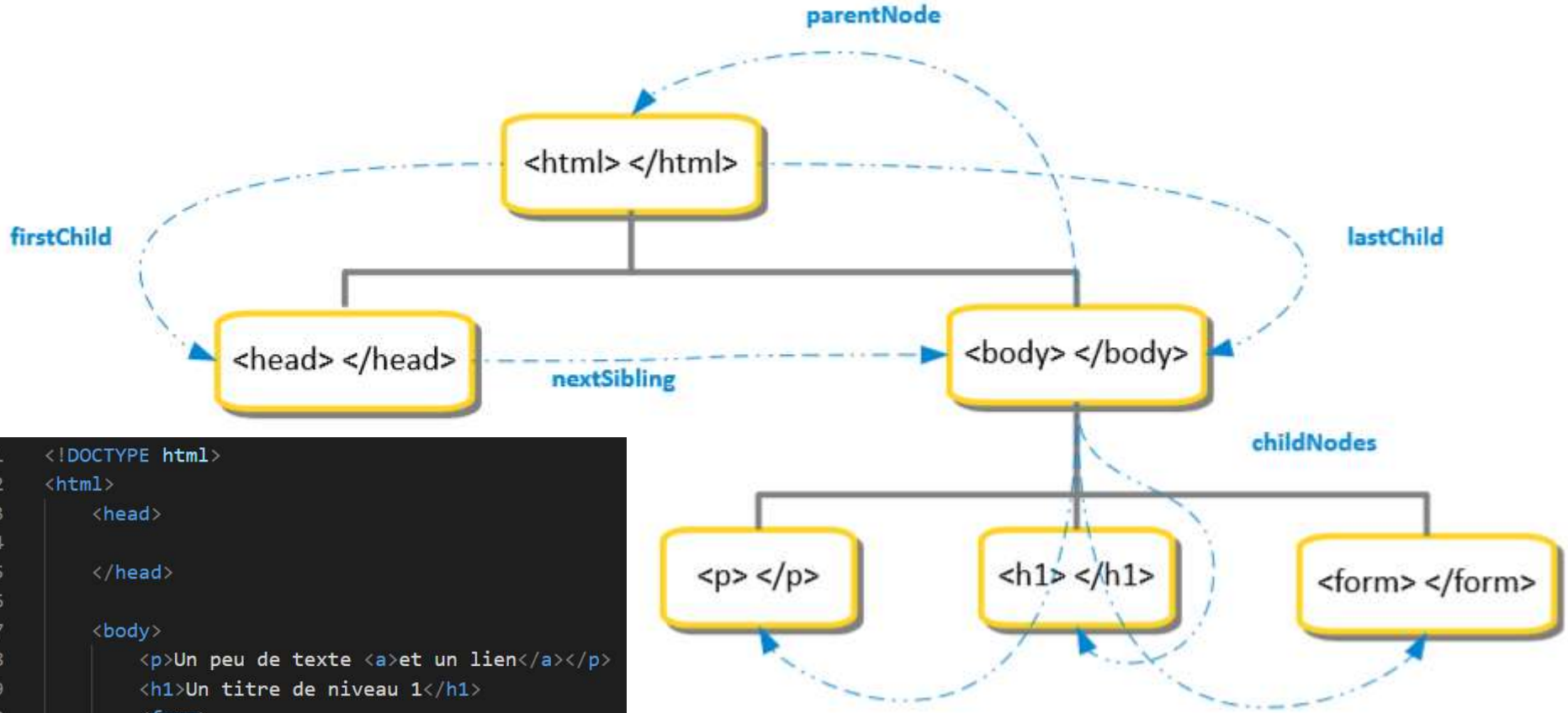
❑ Naviguer entre les noeuds (Plusieures propriétés)

parentNode	Permet d'accéder à **l'élément parent** d'un élément
nodeType & nodeName	Servent respectivement à vérifier le ***type*** et le ***nom*** d'un nœud
firstChild & lastChild	Servent respectivement à accéder au **premier** et au **dernier** enfant d'un nœud
firstElementChild & lastElementChild	Servent à ne récupérer que **les enfants qui sont considérés comme des éléments** HTML
nodeValue & data	Servent à **récupérer le texte**. Elles ne s'appliquent *que* sur des nœuds textuels
childNodes	Retourne un tableau contenant la **liste des enfants** d'un élément.
nextSibling & previousSibling	Permettent d'accéder respectivement au **nœud suivant** et au **nœud précédent**
nextElementSibling & previousElementSibling	Permettent de ne récupérer que les éléments HTML

Partie 2 : Modeler vos pages Web

① Manipuler le code HTML

❑ Naviguer entre les noeuds (Plusieures propriétés)




```
<!DOCTYPE html>
<html>
    <head>

    </head>

    <body>
        <p>Un peu de texte <a>et un lien</a></p>
        <h1>Un titre de niveau 1</h1>
        <form>

        </form>
    </body>
</html>
```

① Manipuler le code HTML

❑ Naviguer entre les noeuds (Plusieures propriétés)

◆ nodeType

- Retourne un nombre, qui correspond à un type de nœud.
- Voici un tableau qui liste les types les plus utilisés en HTML 5, ainsi que leurs numéros :

Numéro	Type de noeud
1	Noeud élément
2	Noeud attribut
3	Noeud texte
4	Noeud pour commentaire
…	…
8	Noeud pour commentaire

◆ nodeName

- Retourne simplement le nom de l'élément, en majuscule.

① Manipuler le code HTML

❑ Naviguer entre les noeuds (Plusieures propriétés)

◆ Exp 1 : firstChild, lastChild, nodeName, nodeValue, data

```
8  <body>
9
10     <div>
11         <p id="myP">Un peu de texte, <a>un lien</a> et <strong>une portion en emphase</strong></p>
12     </div>
13
14     <script>
15         var paragraph = document.getElementById('myP');
16         var first = paragraph.firstChild;    var firstElement = paragraph.firstElementChild;
17         var last = paragraph.lastChild;      var lastElement = paragraph.lastElementChild;
18
19         alert(first.nodeName.toLowerCase());
20         alert(last.nodeName.toLowerCase());
21
22         alert(firstElement.nodeName.toLowerCase());
23         alert(lastElement.nodeName.toLowerCase());
24
25         alert(first.nodeValue);
26         alert(last.firstChild.data);
27
28         alert(next.firstChild.data);
29     </script>
30
31 </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Naviguer entre les noeuds (Plusieures propriétés)

◆ Exemple 2 : childNodes

```
8   <body>
9
10      <div>
11          <p id="myP">Un peu de texte <a>et un lien</a></p>
12      </div>
13
14      <script>
15          var paragraph = document.getElementById('myP');
16          var children = paragraph.childNodes;
17
18          alert(children.length);
19
20          for (var i = 0, c = children.length; i < c; i++) {
21              if (children[i].nodeType === 1) { // C'est un élément HTML
22                  alert(children[i].firstChild.data);
23              }
24              else { // C'est certainement un nœud textuel
25                  alert(children[i].data);
26              }
27          }
28      </script>
29
30  </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Ajouter un élément

□ Avec le DOM, l'ajout d'un élément HTML se fait en **quatre temps** :

1. **Création de l'élément**

```
18      var newLink = document.createElement('a');
```

2. **Lui affecter des attributs**

```
20      newLink.id = 'VersUHP';
21      newLink.href = 'http://www.uhp.ac.ma';
22      newLink.title = 'Visiter le site de l''université Hassan Premier de Settat';
23      newLink.setAttribute('tabindex', '3');
```

3. **L'insèrer dans le document**

```
26      document.getElementById('myP').appendChild(newLink);
```

4. **Créer le contenu textuel**

```
27      var newLinkText = document.createTextNode("Université Hassan Premier");
28      newLink.appendChild(newLinkText);
```


① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Ajouter un élément (Bilan)

```
8  <body>
9      <div>
10         <p id="myP">Un peu de texte <a>et un lien</a></p>
11     </div>
12     <script>
13         var newLink = document.createElement('a');
14
15         newLink.id = 'VersUHP';
16         newLink.href = 'http://www.uh1.ac.ma';
17         newLink.title = 'Visiter le site de Université Hassan Premier de Settat';
18         newLink.setAttribute('tabindex', '3');
19
20
21         document.getElementById('myP').appendChild(newLink);
22
23         var newLinkText = document.createTextNode("Université Hassan Premier");
24
25         newLink.appendChild(newLinkText);
26     </script>
27 </body>
```

Essayer JavaScript

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Cloner un élément

- cloneNode (**true** ou **false**) :
  - **True :** cloner le nœud **avec** ses enfants et ses différents attributs
  - **False :** cloner le nœud **sans** ses enfants et ses différents attributs

```
8   <body>
9       <div>
10          <p id="myP">Un peu de texte <a>et un lien</a></p>
11      </div>
12      <script>
13          // Ici, on clone un e?le?ment existant :
14          var paragraph1 = document.getElementById('myP');
15          var paragraph2 = paragraph1.cloneNode(true);
16
17          // Et attention, l'élément est cloné, mais pas «inséré»
18          paragraph1.parentNode.appendChild(paragraph2);
19      </script>
20  </body>
```

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Remplacer un élément par un autre

- replaceChild (Nouvel_Elément, Elément_A_Remplacer) :

```
8  <body>
9      <div>
10         <p id="myP">Un peu de texte <a>et un lien</a></p>
11     </div>
12     <script>
13         var link = document.getElementsByTagName('a')[0];
14
15         var newLabel= document.createTextNode('et un hyperlien');
16         link.replaceChild(newLabel, link.firstChild);
17     </script>
18 </body>
```

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Supprimer un élément

- removeChild (Elément_A_Supprimer) :

```
8  <body>
9      <div>
10         <p id="myP">Un peu de texte <a>et un lien</a></p>
11     </div>
12     <script>
13         var link = document.getElementsByTagName('a')[0];
14
15         link.parentNode.removeChild(link);
16
17     </script>
18 </body>
```

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Vérifier la présence d'éléments enfants

- hasChildNode ()

```
8  <body>
9      <div>
10         <p id="myP">Un peu de texte <a>et un lien</a></p>
11     </div>
12     <script>
13
14         var paragraph = document.getElementsByTagName('p')[0];
15
16         alert(paragraph.hasChildNodes()); // Affiche true
17
18     </script>
19 </body>
```

① Manipuler le code HTML

❑ Manipuler les éléments HTML (les noeuds)

◆ Insérer à la bonne place

- **insertBefore(param1, param2)**, ~~**insertAfter(param1, param2)**~~
  - **Param1 :** l'élément à insérer
  - **Param2 :** l'élément avant lequel l'élément va être inséré

```
<body>
    <div>
        <p id="myP">Un peu de texte <a>et un lien</a></p>
    </div>
    <script>

        var paragraph = document.getElementsByTagName('p')[0];
        var emphasis = document.createElement('em');
        var emphasisText = document.createTextNode('Le texte inséré');

        emphasis.appendChild(emphasisText);

        paragraph.insertBefore(emphasis, paragraph.lastChild);

    </script>
</body>
```

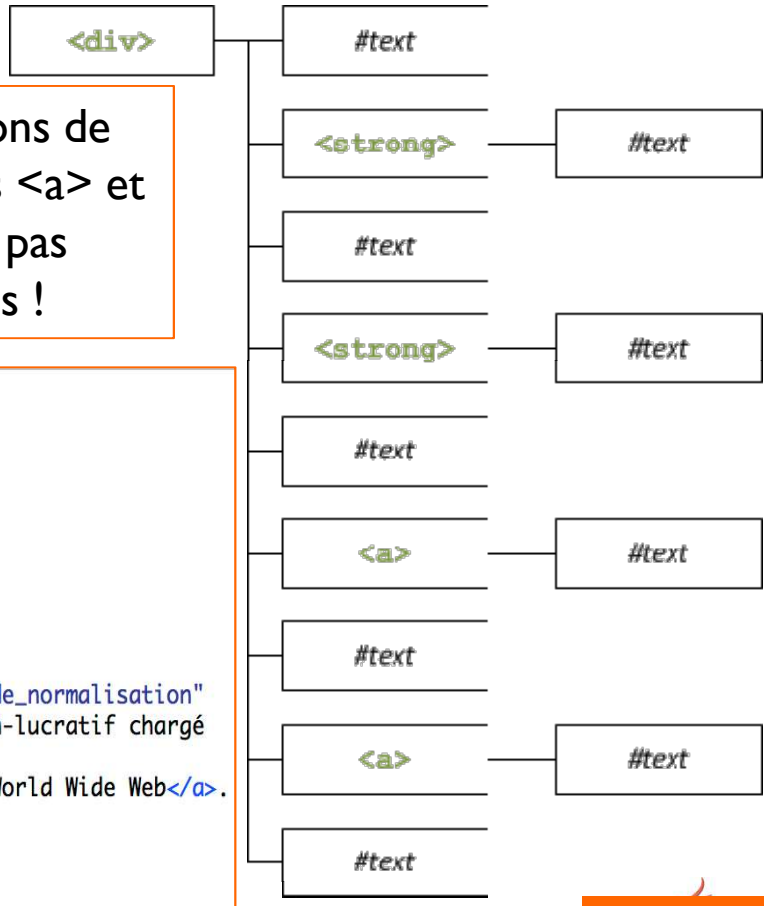
Essayer JavaScript

① Manipuler le code HTML

❑ TP – Recréer une structure DOM

◆ Exercice N°1

Énoncé : Pour ce premier exercice, nous vous proposons de recréer « du texte » mélangé à divers éléments tels des <a> et des . C'est assez simple, mais pensez bien à ne pas vous emmêler les pinceaux avec tous les nœuds textuels !




```
<!DOCTYPE html>

<html>
<head>
    <title>Page Title</title>
</head>

<body>
    <div id="divTP1">
    Le <strong>World Wide Web Consortium</strong>, abre?ge? par le sigle
    <strong>W3C</strong>, est un <a href="http://fr.wikipedia.org/wiki/Organisme_de_normalisation"
    title="Organisme de normalisation"> organisme de standardisation</a> à but non-lucratif chargé
    de promouvoir la compatibilite? des technologies du
    <a href="http://fr.wikipedia.org/wiki/World_Wide_Web" title="World Wide Web">World Wide Web</a>.
    </div>
</body>

</html>
```

Essayer JavaScript

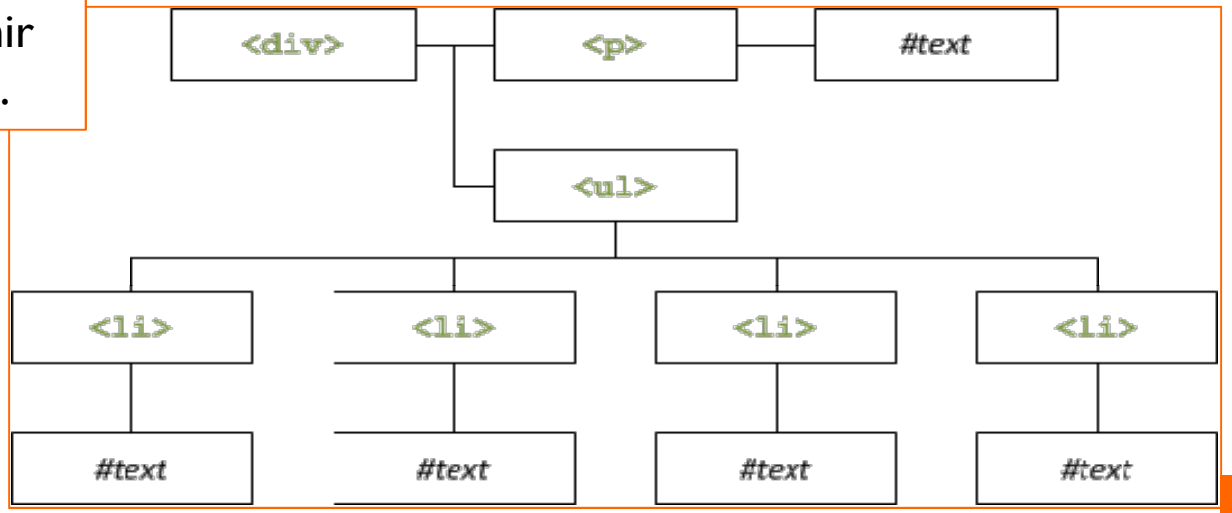
① Manipuler le code HTML

❑ TP – Recréer une structure DOM

◆ Exercice N°2

Énoncé : On ne va tout de même pas créer quatre éléments **<li>** « à la main »... Utilisez une boucle **for** ! Et souvenez-vous, utilisez un tableau pour définir les éléments textuels.

```
8  <body>
9      <div id="divTP2">
10         <p>Langages basé sur ECMAScript :</p>
11         <ul>
12             <li>JavaScript</li>
13             <li>JScript</li>
14             <li>ActionScript</li>
15             <li>EX4</li>
16         </ul>
17     </div>
18 </body>
```



Essayer JavaScript

```
<body>
    <script>
        // On crée l'élément conteneur
        var mainDiv = document.createElement('div');
            mainDiv.id = 'divTP2';

        // On crée tous les nœuds textuels, pour plus de facilité
        var languages = [
            document.createTextNode('JavaScript'),
            document.createTextNode('JScript'),
            document.createTextNode('ActionScript'),
            document.createTextNode('EX4')
        ];

        // On crée le paragraphe
        var paragraph     = document.createElement('p');
        var paragraphText = document.createTextNode('Langages basés sur ECMAScript :');
        paragraph.appendChild(paragraphText);


        // On crée la liste, et on boucle pour ajouter chaque item
        var uList = document.createElement('ul'),
            uItem;

        for (var i = 0, c=languages.length; i < c; i++) {
            uItem = document.createElement('li');

            uItem.appendChild(languages[i]);
            uList.appendChild(uItem);
        }

        // On insère le tout dans mainDiv
        mainDiv.appendChild(paragraph);
        mainDiv.appendChild(uList);

        // On insère mainDiv dans le <body>
        document.body.appendChild(mainDiv);
    </script>
```

① Manipuler le code HTML

Résumé

- Une fois qu'on a accédé à un élément, on peut **naviguer vers d'autres éléments** avec **parentNode**, **previousSibling** et **nextSibling**, ainsi que récupérer des informations sur le nom des éléments et leur contenu.
- **Pour ajouter un élément**, il faut d'abord le créer, puis lui adjoindre des attributs et enfin l'insérer à l'endroit voulu au sein du document.
- Outre le « passage par valeur », le Javascript possède un « passage par référence » qui est fréquent lorsqu'on manipule le DOM. C'est cette histoire de référence qui nous oblige à utiliser une méthode telle que **cloneNode()** pour dupliquer un élément. En effet, copier la variable qui pointe vers cet élément ne sert à rien.
- Si **appendChild()** est particulièrement pratique, **insertBefore()** l'est tout autant pour insérer un élément avant un autre. Créer une fonction **insertAfter()** est assez simple et peut faire gagner du temps.

Partie 2 : Modeler vos pages Web

Objectif

- Nous allons étudier dans cette partie la modification de la structure d'une page web

Menu

1. Manipuler le code HTML
2. Les événements
3. Les formulaires
4. Manipuler le CSS
5. TP : un formulaire interactif

② Les événements

Objectif

Dans cette partie, nous allons aborder les événements en Javascript :

- **L'utilisation des événements sans le DOM**
- **L'utilisation des événements avec le DOM-0**
- **L'utilisation des événements avec le DOM-2**

Nous verrons comment

- **mettre en place ces événements,**
- **les utiliser,**
- **modifier leur comportement, etc.**

② Les événements

❑ Plan

1. Que sont les événements ?
2. Les événements au travers du DOM
3. L'objet Event
4. Résoudre les problèmes d'héritage des événements

② Les événements

❑ Que sont les événements ?

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 aux formulaires (input, checkbox, etc.)
select	Sélectionner le contenu d'un champ de texte (input, textarea, etc.)
Evénements spécifiques à l'élément <form>	
submit	Envoyer le formulaire
reset	Réinitialiser le formulaire

② Les événements

❑ Que sont les événements ?

◆ Utiliser les événements

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

◆ Le mot-clé this

```
 9   <body>
10       <span onclick="alert('Voici le contenu de l\'élément span :\n\n' + this.innerHTML);">
11           Cliquez-moi !
12       </span>
13   </body>
```

◆ Retour sur le focus

```
 9   <body>
10       <input id="input" type="text" size="50" value="Cliquez ici !"
11              onfocus="this.value='Appuyez maintenant sur votre touche de tabulation.';"
12              onblur="this.value='Cliquez ici !';"/>  <br /><br/>
13       <a href="#" onfocus="document.getElementById('input').value = 'Focus sur le lien, bravo !';">Un lien bidon</a>
14   </body>
```

Essayer JavaScript

② Les événements

❑ Que sont les événements ?

◆ Bloquer l'action par défaut de certains événements

- Le click sur un lien fait une redirection vers le contenu de l'attribut **href** :

```
11    <a href="http://www.uh1.ac.ma" onclick="alert('Vous avez cliqué !');">
12        Cliquez-moi !
13    </a>
```

- Pour bloquer cette redirection, il suffit juste d'ajouter le code **return false;** dans notre événement **click** :

```
11    <a href="http://www.uh1.ac.ma" onclick="alert('Vous avez cliqué !'); return false;">
12        Cliquez-moi !
13    </a>
```

➔ **Ici, le return false; sert juste à bloquer l'action par défaut de l'événement qui le déclenche**

② Les événements

❑ Les événements au travers du DOM

◆ Le DOM-0

- Cette interface est vieille mais n'est pas forcément dénuée d'intérêt
- Elle reste très pratique pour créer des événements et peut parfois être préférée au DOM-2

```
9  <body>
10
11     <span id="clickme">Cliquez-moi !</span>
12
13     <script>
14
15       var element = document.getElementById('clickme');
16
17       element.onclick = function() {
18           alert("Vous m'avez cliqué !");
19       };
20
21     </script>
22
23 </body>
```

Voyons par étapes ce que nous avons fait dans ce code :

1. On récupère tout d'abord l'élément HTML dont l'ID est clickme ;
2. On accède ensuite à la propriété onclick à laquelle on assigne une fonction anonyme ;
3. Dans cette même fonction, on fait un appel à la fonction alert() avec un texte en paramètre.

➔ **Concernant la suppression d'un événement avec le DOM-0, il suffit tout simplement de lui attribuer une fonction anonyme vide :**

```
21        element.onclick = function() {};
```

② Les événements

❑ Les événements au travers du DOM

◆ Le DOM-2

➔ Pourquoi le DOM-2 et non pas le DOM-0 voire pas de DOM du tout ?

- **Sans le DOM**, c'est simple : on ne peut pas y utiliser l'objet **Event** qui est pourtant une mine d'informations sur l'événement déclenché
- **Avec le DOM**-0, il a deux problèmes majeurs : il est vieux, et il ne permet pas de créer plusieurs fois le même événement.
- **Avec le DOM**-2, il permet la création multiple d'un même événement et gère aussi l'objet Event

➔ Tout dépend de votre code ..

② Les événements

❑ Les événements au travers du DOM

◆ Le DOM-2

- Nous n'utilisons plus une propriété mais une méthode nommée **`addEventListener()`** (ne fonctionne pas sous IE8 et antérieur)

```
9   <body>
10      <span id="clickme">Cliquez-moi !</span>
11
12      <script>
13
14          var element = document.getElementById('clickme');
15
16          element.addEventListener('click', function() {
17              alert("Vous m'avez cliqué !");
18          }, false);
19
20      </script>
21  </body>
```

Cette méthode prend trois paramètres :

1. Le nom de l'événement (sans les lettres « on ») ;
2. La fonction à exécuter (une fonction anonyme) ;
3. Un booléen :
   - **True** : The event handler is executed in the capturing phase
   - **False** : Default. The event handler is executed in the bubbling phase

② Les événements

❑ Les événements au travers du DOM

◆ Le DOM-2

- **Le DOM-2** permet la **création multiple d'événements** identiques pour un même élément :

```
9   <body>
10      <span id="clickme">Cliquez-moi !</span>
11      <script>
12          var element = document.getElementById('clickme');
13
14          // Premier événement click
15          var fonctionPremierClick = function() {
16              alert("1er click !");;
17          };
18
19          // Deuxième événement click
20          var fonctionDeuxiemeClick = function() {
21              alert("2eme click !");;
22          };
23
24          element.addEventListener('click', fonctionPremierClick, false);
25          element.addEventListener('click', fonctionDeuxiemeClick, false);
26      </script>
27  </body>
```

Essayer JavaScript

② Les événements

❑ Les événements au travers du DOM

◆ Le DOM-2

- **la suppression des événements** s'opère avec la méthode `removeEventListener()` et se fait de manière très simple :

```
9  <body>
10     <span id="clickme">Cliquez-moi !</span>
11     <script>
12         var element = document.getElementById('clickme');
13
14         // On crée l'événement
15         var fonctionPremierClick = function() {
16             alert("1er click !");
17         };
18         element.addEventListener('click', fonctionPremierClick, false);
19
20         // On supprime l'événement en lui repassant les mêmes paramètres
21         element.removeEventListener('click', fonctionPremierClick, false);
22     </script>
23 </body>
```

② Les événements

❑ L'objet Event

- Fournir une multitude d'informations sur l'événement actuellement déclenché
  - élément qui a déclenché l'événement,
  - coordonnées du curseur, …
- Il n'est accessible que lorsqu'un événement est déclenché
- Son accès ne peut se faire que dans une fonction exécutée par un événement

② Les événements

❑ L'objet Event

◆ Récupérer l'élément de l'événement actuellement déclenché

```
8   <body>
9       <span id="clickme">Cliquez-moi !</span>
10
11      <script>
12          var clickme = document.getElementById('clickme');
13
14          clickme.addEventListener('click', function(e) {
15              e.target.innerHTML = 'Vous avez cliqué !';
16          });
17      </script>
18  </body>
```

`target` : propriétés de l'objet **e**, permet de récupérer une référence vers l'élément dont l'événement a été déclenché

② Les événements

❑ L'objet Event

◆ Récupérer l'élément à l'origine du déclenchement de l'événement

```
8   <body>
9       <p id="result"></p>
10
11      <div id="parent1">
12          Parent
13          <div id="child1">Enfant N°1</div>
14          <div id="child2">Enfant N°2</div>
15      </div>
16
17      <script>
18          var parent1 = document.getElementById('parent1'),
19              result = document.getElementById('result');
20
21          parent1.addEventListener('mouseover', function(e) {
22              result.innerHTML = "L'élément déclencheur de l'événement possède l'ID : " + e.target.id;
23          });
24      </script>
25  </body>
```

② Les événements

❑ L'objet Event

◆ Récupérer la position du curseur

```
8   <body>
9       <div id="position"></div>
10
11      <script>
12          var position = document.getElementById('position');
13
14          document.addEventListener('mousemove', function(e) {
15              position.innerHTML = 'Position X : ' + e.clientX +
16              'px<br />Position Y : ' + e.clientY + 'px';
17          });
18      </script>
19  </body>
```

② Les événements

❑ L'objet Event

◆ Récupérer les touches frappées par l'utilisateur

```
8   <body>
9       <p>
10          <input id="field" type="text" />
11      </p>
12
13      <table>
14          <tr>
15              <td>keydown</td>
16              <td id="down"></td>
17          </tr>
18          <tr>
19              <td>keypress</td>
20              <td id="press"></td>
21          </tr>
22          <tr>
23              <td>keyup</td>
24              <td id="up"></td>
25          </tr>
26      </table>
```

```
28      <script>
29          var field = document.getElementById('field'),
30              down = document.getElementById('down'),
31              press = document.getElementById('press'),
32              up = document.getElementById('up');
33
34          document.addEventListener('keydown', function(e) {
35              down.innerHTML = e.keyCode;
36          });
37
38          document.addEventListener('keypress', function(e) {
39              press.innerHTML = e.keyCode;
40          });
41
42          document.addEventListener('keyup', function(e) {
43              up.innerHTML = e.keyCode;
44          });
45      </script>
46  </body>
```

② Les événements

❑ L'objet Event

◆ Bloquer l'action par défaut de certains événements

```
8   <body>
9       <a id="link" href="http://www.fpk.ac.ma/">Cliquez-moi !</a>
10
11      <script>
12          var link = document.getElementById('link');
13
14          link.addEventListener('click', function(e) {
15              // On bloque l'action par défaut de cet événement
16              e.preventDefault();
17              alert('Vous avez cliqué !');
18          });
19      </script>
20  </body>
```

Partie 2 : Modeler vos pages Web

Objectif

- Nous allons étudier dans cette partie la modification de la structure d'une page web

Menu

1. Manipuler le code HTML
2. Les événements
3. Les formulaires
4. Manipuler le CSS
5. TP : un formulaire interactif

② Les événements

❑ Plan

1. Que sont les événements ?
2. Les événements au travers du DOM
3. L'objet Event
4. Résoudre les problèmes d'héritage des événements

② Les formulaires

❑ Les propriétés

value	permet de définir une valeur pour différents éléments d'un formulaire comme les<input>, les<button>, etc
disabled	
checked	
readonly	
selectedIndex	
options	

② Les formulaires

❑ Les méthodes et un retour sur quelques événements

◆ Les méthodes spécifiques à l'élément <form>

- **submit() :** permet d'effectuer l'envoi d'un formulaire sans l'intervention de l'utilisateur
- **reset() :** permet de réinitialiser tous les champs d'un formulaire

```
8   <body>
9       <form id="myForm">
10          <input type="text" value="Entrez un texte" />
11          <br /><br />
12          <input type="submit" value="Submit !" />
13          <input type="reset" value="Reset !" />
14      </form>
15
16      <script>
17          var myForm = document.getElementById('myForm');
18
19          myForm.addEventListener('submit', function(e) {
20              alert('Vous avez envoyé le formulaire !');
21              alert('Mais celui-ci a été bloqué pour que vous ne changiez pas de page.');
22              e.preventDefault();
23          });
24
25          myForm.addEventListener('reset', function(e) {
26              alert('Vous avez réinitialisé le formulaire !');
27          });
28      </script>
29  </body>
```

Essayer JavaScript

… Fin