

Programmation événementielle

JAVASCRIPT

Elisabeth Pecatte

elisabeth.pecatte@iut-tlse3.fr

Extraits des cours de Jean-Marie Pecatte et d'open-classroom

1 DOM (Document Object Model)

- modèle standardisé par le W3C (*World Wide Web Consortium*)
- représente un document sous la forme d'un arbre
- toutes les balises HTML sont donc des nœuds de l'arbre
- plusieurs types de nœud :
 1. Nœud élément (balise/tag HTML)
 2. Nœud attribut (attribut HTML)
 3. Nœud texte (contenu textuel d'un élément)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <title>Exemple DOM</title>
```

```
</head>
```

```
<body>
```

```
  <h1 class="titre">Titre</h1>
```

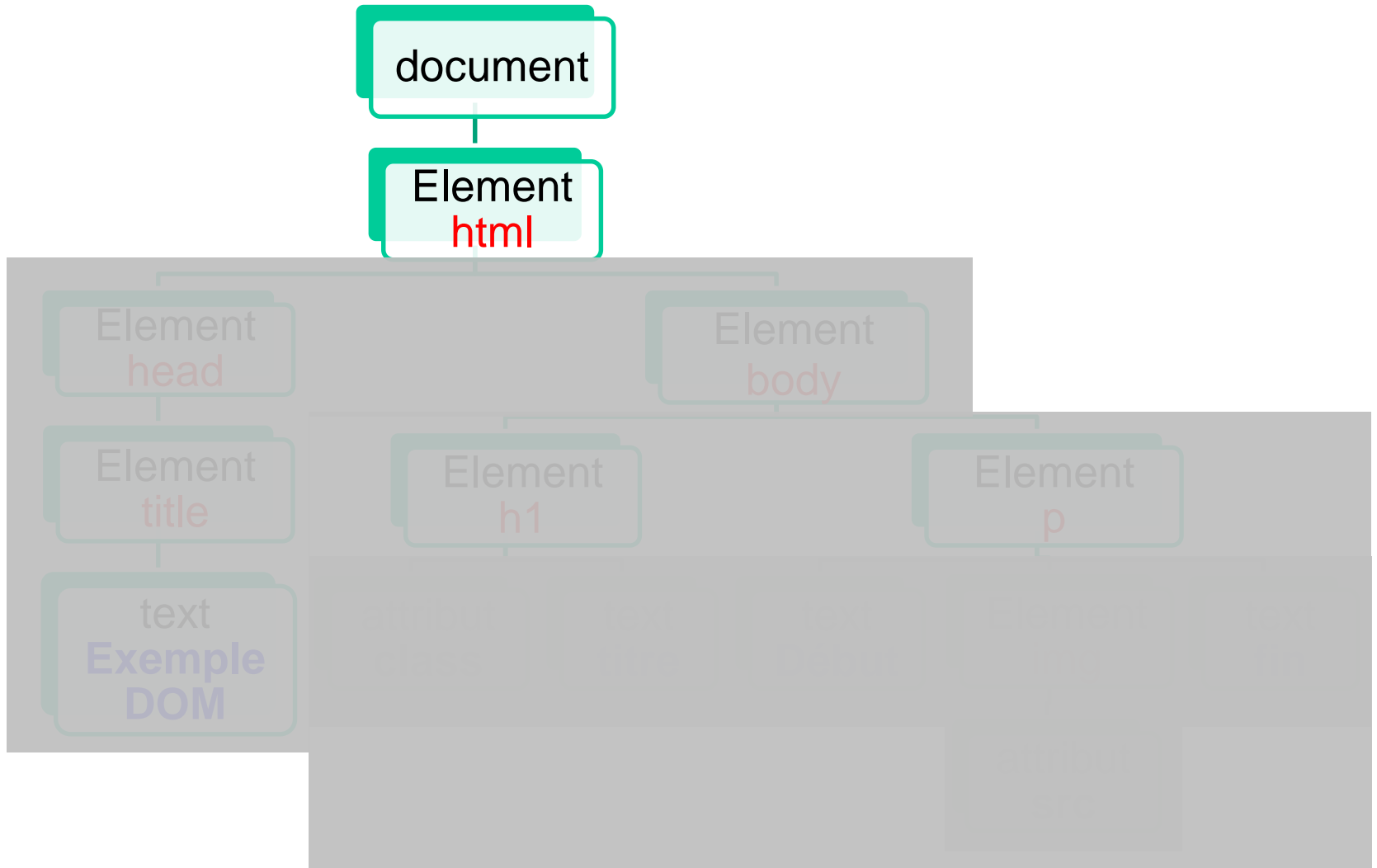
```
  <p>Débutfin</p>
```

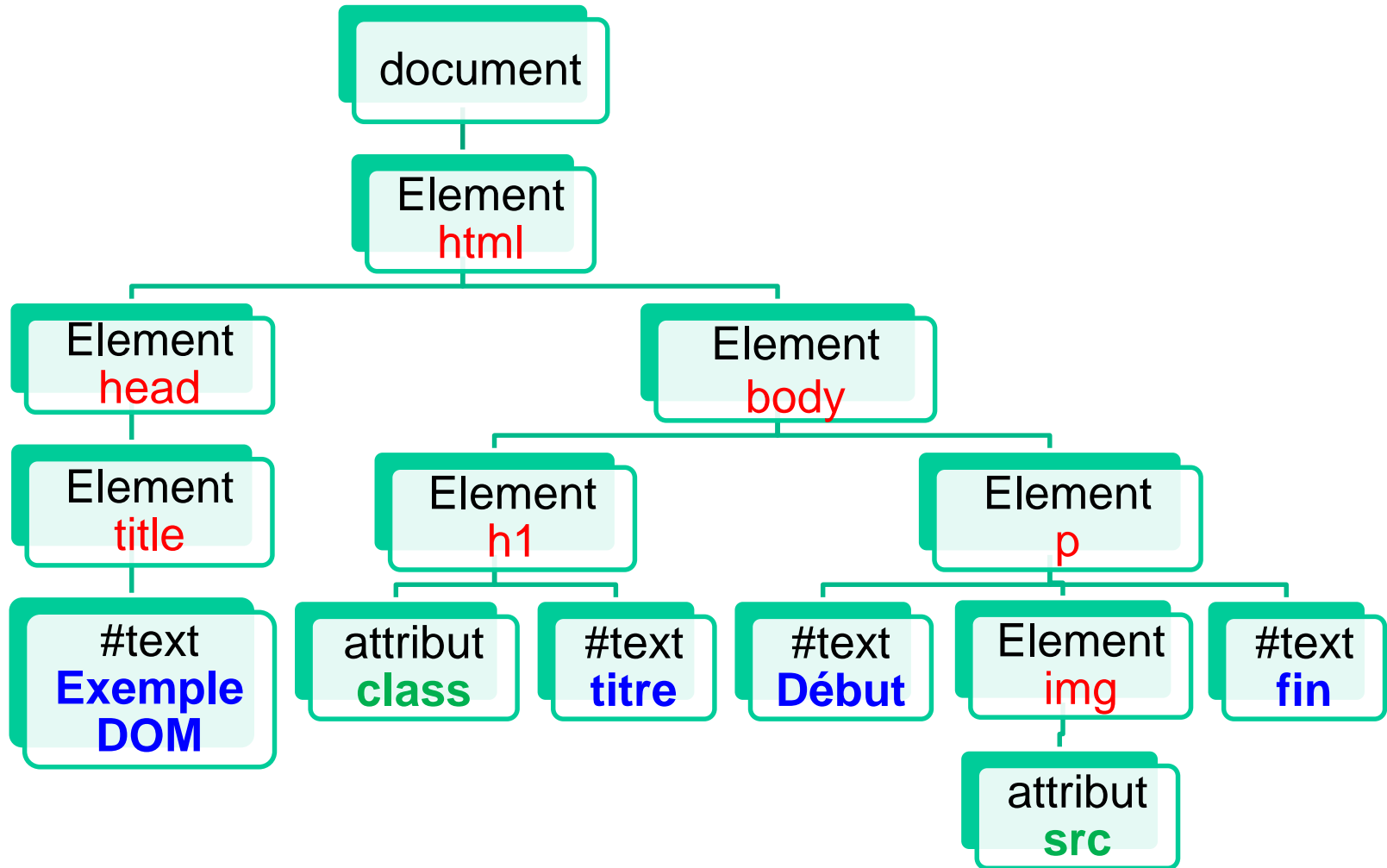
```
</body>
```

```
</html>
```



DOM





- **window :**

- Correspond à la fenêtre du navigateur
- C'est à partir de cet objet que le javascript est exécuté
- Objet implicite

alert('hello world') et non pas window.alert('hello word')

alert est une méthode de l'objet window/

- **document**

- Sous-objet de window qui présente la page web
(<html>)
- L'un des plus utilisé

Accéder au éléments

- L'objet document possède 5 méthodes principales:

1- getElementById()

- Permet d'accéder à un élément en connaissant son **id**

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

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

2 - getElement**s**ByTagName()

- Récupération dans un tableau, tous les éléments de la même famille

```
<script>  
  var tabDivs = document.getElementsByTagName('div');  
  
  for (var i = 0 ; i < tabDivs.length ; i++) {  
    alert('Element n° ' + (i + 1) + ' : ' + tabDivs[i]);  
  }  
</script>
```

Récupération de tous les éléments <div> dans le tableau tabDivs

Méthode accessible à tous les éléments HTML

3 - getElementByName()

Permet de récupérer les éléments qui ont l'attribut **name** spécifié.

Cet attribut est utilisé dans les balises de formulaire.

4 – querySelector() et querySelectorAll()

(ne fonctionne pas avec les navigateurs anciens.)

L'argument est // type sélecteur CSS

Ex : querySelector('#menu li') renvoie la première occurrence

querySelectorAll('#menu li') renvoie l'ensemble des éléments sous forme de tableau

```
<ul id="menu">  
  <li>  
  <li>  
</ul>
```


Rappel

- Les éléments HTML sont souvent composés d'attributs et d'un contenu, de type **text** ou un autre élément HTML.
- Un élément HTML est un objet qui appartient à plusieurs objets, et de ce fait, qui hérite des propriétés et méthodes de ses objets parents.

Attributs

- Méthodes **getAttribute()** pour récupérer l'attribut d'un élément et **setAttribute()** pour l'éditer

```
<a id="myLink" href="http://google.fr">Un lien modifié dynamiquement</a>


<script>
  var lien= document.getElementById('myLink');
  var href = lien.getAttribute('href');
  // On récupère l'attribut « href » de l'élément qui a pour id 'myLink'

  lien.setAttribute('href', 'http://www.univ-jfc.fr');
  // on édite un nouvel attribut « href »
</script>
```

Propriétés

- En fait, pour la plupart des éléments courants comme `<a>`, il est possible d'accéder à un attribut via une propriété.

```
<a id="myLink" href="http://google.fr">Un lien</a>
<script>
  document.getElementById('myLink').href = "http://univ-jfc.fr"
</script>
```



Éditer les éléments HTML

- **Propriété css de l'attribut style de l'élément**
- Il est possible d'atteindre l'attribut style des balises via le sous-objet **style** js, pour les modifier et les éditer :

```
<div id="encouleur">Test</div>
```

```
<script>
```

```
var divTest = document.getElementById("encouleur");  
  
divTest.className = "enrouge";  
divTest.style.backgroundColor = "green";  
divTest.className = "";
```

```
</script>
```

```
div{background-color: blue;}  
.enrouge{background-color: red;}
```



```
<div id="encouleur" class style="background-color: green;">Test</div> == $0
```

- **L'attribut classe**

- **class** est un mot réservé, aussi il n'est pas possible, pour modifier l'attribut class, d'utiliser la propriété `element.class`, il faut utiliser **className**
- La propriété ne retourne pas un tableau, mais une chaîne de caractère. Lorsque l'élément a plusieurs classes, il faut couper la chaîne obtenu à l'aide de la méthode `split()`.

```
<script>
    var classes = document.getElementById('myLink').className;
    classes = classes.split(' ');
    var classesNew = [];
    for (var i=0 ; i < classes.length; i++) {
        if (classes[i]) {
            classesNew.push(classes[i]);
        }
    }
</script>
```

Editer les éléments HTML

• innerHTML

- La propriété innerHTML permet de récupérer le code HTML d'un élément enfant sous forme de texte

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

```
<script>
  var div = document.getElementById('myDiv');
  alert(div.innerHTML);
</script>
```

navigateur

Cette page indique :

```
<<<p>Un peu de texte <a>et un lien</a></p>
□
```

• textContent

- La propriété textContent permet de récupérer le texte brut, sans balise

```
<script>
  var div = document.getElementById('myDiv');
  alert(div.textContent);
</script>
```

Cette page indique :

navigateur

```
<<<Un peu de texte et un lien
□
```

- Pour IE, il faut utiliser innerText – *non standardisé*

```
var div = document.getElementById('myDiv');
var txt = '';
if (div.textContent) { txt = div.textContent; }
else if (div.innerText) { txt = div.innerText; }
```

```
txt = div.textContent || div.innerText ;
```

Créer et supprimer un élément HTML

L'ajout d'un élément HTML se fait en 3 temps :

- 1 - On crée l'élément avec la création d'un élément se fait avec la méthode `createElement()`:

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

- 2 - On lui affecte des attributs soit avec `setAttribute()`, soit directement avec les propriétés adéquates.

```
newLink.id      = 'monLien';  
newLink.title   = 'Université Champollion !';  
newLink.setAttribute('href', 'http://www.univ-jfc.fr');
```

Créer et supprimer un élément HTML

3 - On l'insère dans le document, à l'aide de la méthode `appendChild()` et ce n'est qu'à ce moment-là qu'il sera « ajouté » à l'élément parent.

```
document.getElementById('encouleur').appendChild(newLink);
```

Parent d'id
encouleur

Ajout du lien

navigateur

```
<div id="encouleur" class style="background-color: green;"> == $0  
  "Test"  
  <a id="monLien" title="Université Champollion !" href="http://www.univ-jfc.fr"></a>  
</div>
```


Créer et supprimer un élément HTML

- **createTextNode()** sert à créer un nœud textuel (de type #text)

```
var newLinkText = document.createTextNode("Université Champollion");  
newLink.appendChild(newLinkText);
```

En résumé, pour la création d'un lien et du texte à cliquer

```
var newLink = document.createElement('a');  
var newLinkText = document.createTextNode("Université Champollion");  
  
newLink.id      = 'monLien';  
newLink.title   = 'Université Champollion !';  
newLink.setAttribute('href', 'http://www.univ-jfc.fr');  
  
document.getElementById('enCouleur').appendChild(newLink);  
newLink.appendChild(newLinkText);
```

Créer et supprimer un élément HTML

• Dupliquer un élément

Si on écrit :

```
var newDiv1 = document.createElement('div');  
var newDiv2 = newDiv1;
```

ne duplique pas, car les objets du DOM sont accessibles par référence => newDiv2 pointe vers la même 'div' que newDiv1 mais le contenu n'est pas dupliqué

Il faut utiliser **cloneNode()**

```
var newDiv2 = newDiv1.cloneNode(true);
```

- true : duplique également les enfants et les attributs
- false sinon

Créer et supprimer un élément HTML

- Pour remplacer un élément, il faut utiliser la méthode `replaceChild()`
- Et pour le supprimer, `removeChild()`

Naviguer entre les noeuds

- **parentNode**

- propriété permettant d'accéder à l'élément parent de l'élément

```
<blockquote>  
  <p id="myP">Ceci est un paragraphe !</p>  
</blockquote>
```

```
var paragraph = document.getElementById('myP');  
var blockquote = paragraph.parentNode;
```

- **firstChild** et **lastChild** servent respectivement à accéder au premier et au dernier enfant d'un nœud.

```
var first = paragraph.firstChild;  
var last = paragraph.lastChild;
```

- **nodeValue** ou **data** permettent de récupérer la valeur des nœud textuel

```
alert(first.nodeValue);  
alert(last.firstChild.data);
```

Naviguer entre les noeuds

- **childNodes** : propriété qui retourne un tableau contenant la liste des enfants d'un élément (children ne retourne que tous les éléments enfant, childNodes retourne tous les nœuds enfants)
- **nextSibling** et **previousSibling** : propriétés qui permettent d'accéder respectivement au nœud suivant et au nœud précédent