DÉVELOPPEMENT D'APPLICATIONS WEB

COURS 6: JAVASCRIPT ET MODÈLE OBJET DE DOCUMENT

Enseignante: NGUYEN Thi Minh Tuyen

Plan du cours

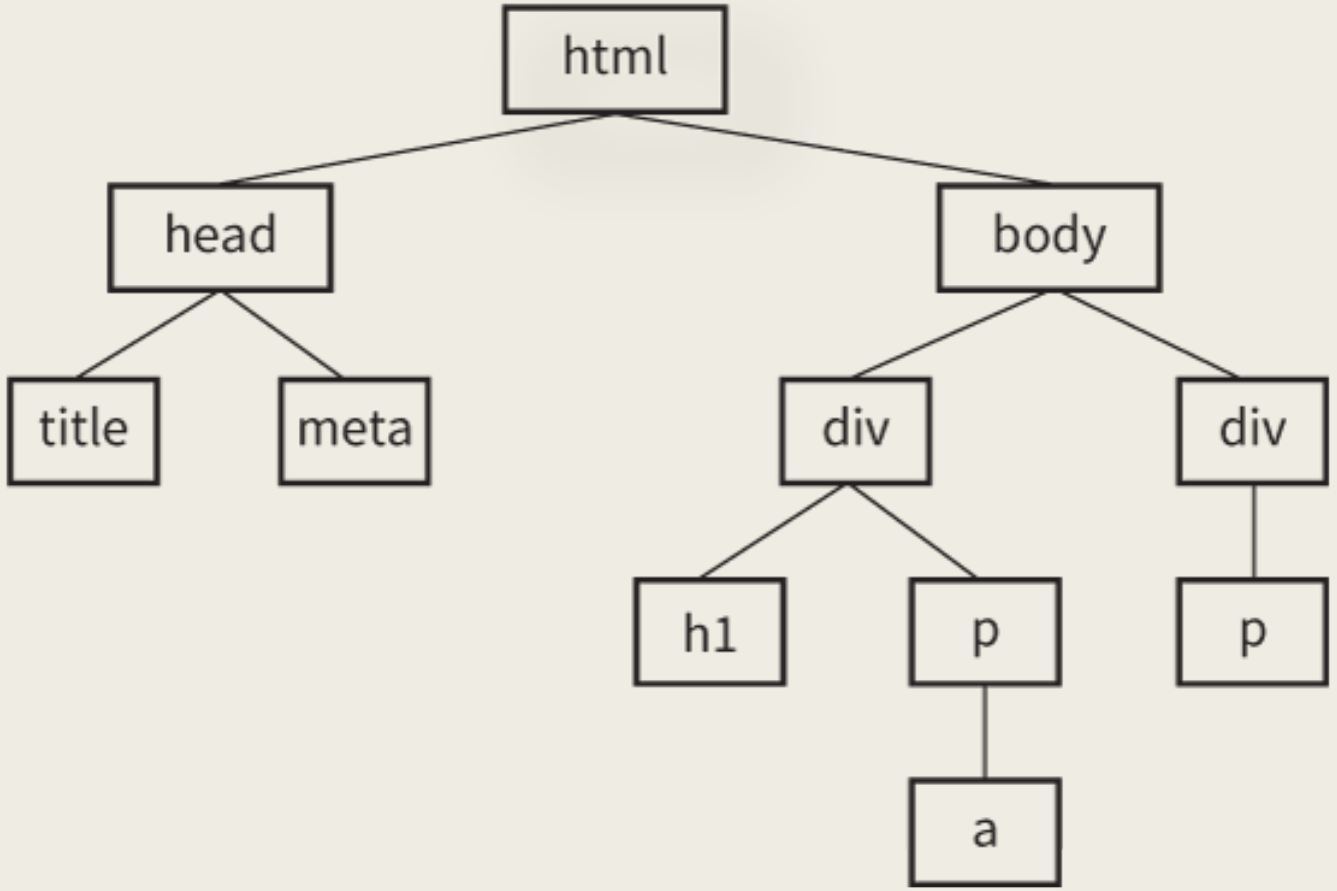
1. Qu'est-ce que le DOM?
2. Accès aux éléments
3. Accès aux attributs
4. Accès aux contenus
5. Modification de la structure HTML

Introduction au DOM

- DOM = **D**ocument **O**bject **M**odel
- Est une interface de programmation qui permet d'accéder au contenu d'un document et de le manipuler.
- Fournit une carte structurée du document et un ensemble de méthodes pour interagir avec eux.
- Peut être utilisé avec d'autres langages XML et d'autres langages de programmation (tels que PHP, Ruby, etc.) peuvent y accéder.

Nœud d’arbre [1]

- Le DOM traite la structure d'un document comme un arbre avec des branches:



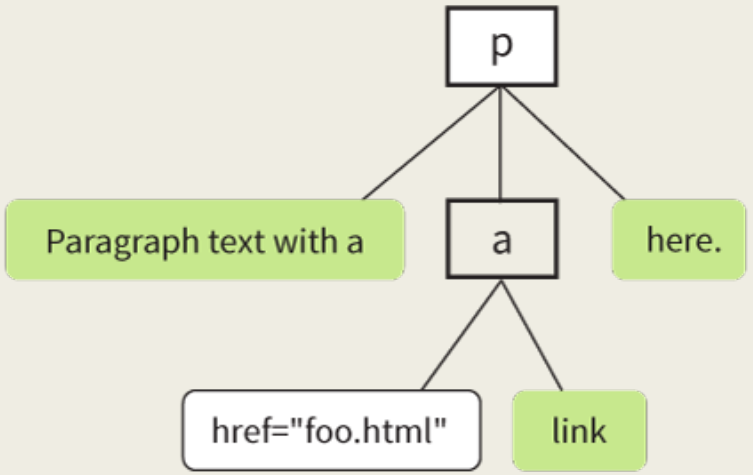
Nœud d’arbre [2]

- Chaque élément, attribut et élément de contenu est un nœud de l'arborescence et est accessible pour les scripts:

Paragraph text with a link here.

Les nœuds dans

un élément p



Accéder aux nœuds [1]

- Pour pointer vers des nœuds, répertoriez-les séparés par des points (.).
- Exemple:

```
var foo = document.getElementById("beginner").innerHTML;
```

L'objet document pointe sur la page elle-même.

`getElementById` spécifie un élément avec l'id `"beginner"`.

`innerHTML` représente le contenu HTML dans cet élément.

Accéder aux nœuds [2]

Méthode	Description
`document.getElementById(`*`id`*`)`	Trouver un élément par son **id**
`document.getElementsByTagName(`*`name`*`)`	Trouver les éléments par son nom de la balise **name**
`document.getElementsByClassName(`*`name`*`)`	Trouver les éléments par son nom de la class **name**

Accéder aux nœuds [2]

Méthodes pour accéder aux nœuds du document:

`getElementsByTagName()`

- Accède à tous les éléments avec le nom de la balise donnée
- Exemple: `document.getElementsByTagName("p");`

`getElementById()`

- Accède à un seul élément par la valeur de son attribut **id**
- Exemple: `document.getElementById("special");`

`getElementsByClassName()`

- Accéder aux éléments par la valeur d'un attribut **class**
- Exemple: `document.getElementsByClassName("product");`

Accéder aux nœuds [4]

`querySelectorAll()`

- Accède aux nœuds basés sur un sélecteur CSS
- Exemple: `document.querySelectorAll(".sidebar p");`

`getAttribute()`

- Accède à la valeur d'un attribut donné
- Exemple: `getAttribute("src")`

Accéder aux nœuds [5]

- Trouver des éléments HTML par id

```
var myElement = document.getElementById("intro");
```

- Trouver des éléments HTML par nom de la balise

```
var x = document.getElementsByTagName("p");
```

- Trouver des éléments HTML par nom de classe

```
var x = document.getElementsByClassName("intro");
```

- Trouver des éléments HTML par sélecteurs CSS

```
var x = document.querySelectorAll("p.intro");
```

Accéder aux nœuds [6]

- Trouver des éléments HTML par des collections d'objets HTML

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Manipulation de nœuds [1]

Propriété	Description
element.innerHTML = *new html content*	Récupère ou définit la syntaxe HTML décrivant les descendants de l'élément.
element.attribute = new value	Change la valeur d'attribut d'un élément HTML
element.style.*property = new style*	Change le style d'un élément HTML
Méthode	**Description**
element.setAttribute*(attribute, value)*	Change la valeur d'attribut d'un élément HTML

Manipulation de nœuds [2]

Plusieurs méthodes intégrées pour manipuler les nœuds:

`setAttribute()`

- Définit la valeur d'un attribut donné:

```
bigImage.setAttribute("src", "newimage.jpg");
```

`innerHTML`

- Spécifie le contenu à l'intérieur d'un élément (y compris le balisage si nécessaire):

```
introDiv.innerHTML = "<p>This is the intro
text.</p>"
```

`style`

- Applique un style en utilisant les propriétés CSS:

```
document.getElementById("intro").style.backgroundCo
lor = "#000;"
```

Modification du flux de sortie HTML

- En JavaScript, `document.write()` écrit directement dans le flux de sortie HTML.
- Exemple:

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      document.write(Date());
    </script>
  </body>
</html>
```

Modification de contenu HTML

- `document.getElementById(id).innerHTML = new HTML`
- Exemple:

```
<html>
  <body>
    <p id="p1">Hello World!</p>
    <script>
      document.getElementById("p1").innerHTML =
"New text!";
    </script>
  </body>
</html>
```

Modification de la valeur d'un attribut

- Syntaxe:

```
document.getElementById(id).attribute = new value
```

- Exemple:

```
<!DOCTYPE html>
<html>
  <body>
    <img id="myImage" src="smiley.gif">
    <script>
      document.getElementById("myImage").src =
"landscape.jpg";
    </script>
  </body>
</html>
```

Modification du style HTML

- Syntaxe:

```
document.getElementById(id).style.property = new style
```

- Exemple:

```
<html>
  <body>
    <p id="p2">Hello World!</p>
    <script>
      document.getElementById("p2").style.color =
"blue";
    </script>
    <p>The paragraph above was changed by a
script.</p>
  </body>
</html>
```

Ajout et suppression des éléments

Méthode	Description
document.createElement(*element*)	Créer un élément HTML
document.removeChild(*element*)	Enlever un élélent HTML
document.appendChild(*element*)	Ajouter un élément HTML
document.replaceChild(*new, old*)	Remplacer un élément HTML
document.write(*text*)	Écrire dans le flux de sortie HTML

Création de nouveaux éléments HTML

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
  var para = document.createElement("p");
  var node = document.createTextNode("This is new.");
  para.appendChild(node);

  var element = document.getElementById("div1");
  element.appendChild(para);
</script>
```

Suppression d'éléments HTML existants

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
  var parent = document.getElementById("div1");
  var child = document.getElementById("p1");
  parent.removeChild(child);
</script>
```

Remplacement des éléments HTML

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
  var para = document.createElement("p");
  var node = document.createTextNode("This is new.");
  para.appendChild(node);

  var parent = document.getElementById("div1");
  var child = document.getElementById("p1");
  parent.replaceChild(para, child);
</script>
```

L'objet DOM HTML NodeList

- Une liste (collecte) de noeuds extraites d'un document.
- Est presque identique à un HTMLCollection objet.
- Certains (anciens) navigateurs retournent un objet NodeList au lieu d'une HTMLCollection pour des méthodes telles que getElementsByClassName().
- Tous les navigateurs retournent un objet NodeList pour la propriété childNodes.
- La plupart des navigateurs retournent un objet NodeList pour la méthode querySelectorAll().

Exemple

```
var myNodeList = document.querySelectorAll("p");
```

- Les éléments du NodeList sont accessibles par un numéro d'index.
- Pour accéder au deuxième noeud <p> vous pouvez écrire:

```
y = myNodeList[1];
```