Web (Persistance)

Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 12

Programmation côté client en JavaScript

Plan

- Document Object Model
- Ajax
- Exemple d'encapsulation d'Ajax

Document Object Model (DOM)

- C'est-à-dire le modèle orienté objet d'une page Web
- Standard du W3C
  - Interface d'accès à la structure et au style de documents XML et HTML
  - indépendant du langage de programmation
  - Indépendant de la plate-forme
- Permet de lire ou mettre à jour le contenu de la page
- Interaction étroite avec le navigateur

Niveaux du DOM

- DOM 0. Version de Netscape Navigator 2.0
- DOM 1 (1998). Internet Explorer 5 / Netscape 6
  - Représentation d'un document sous forme d'un arbre.
  - Chaque élément correspond à un nœud
  - Méthodes permettant de manipuler cet arbre
- DOM 2 (2000). Constitué de six parties :
  - Core, HTML, Events, Style, View, Traversal and Range
- DOM 3 (2004). Version actuelle et définitive

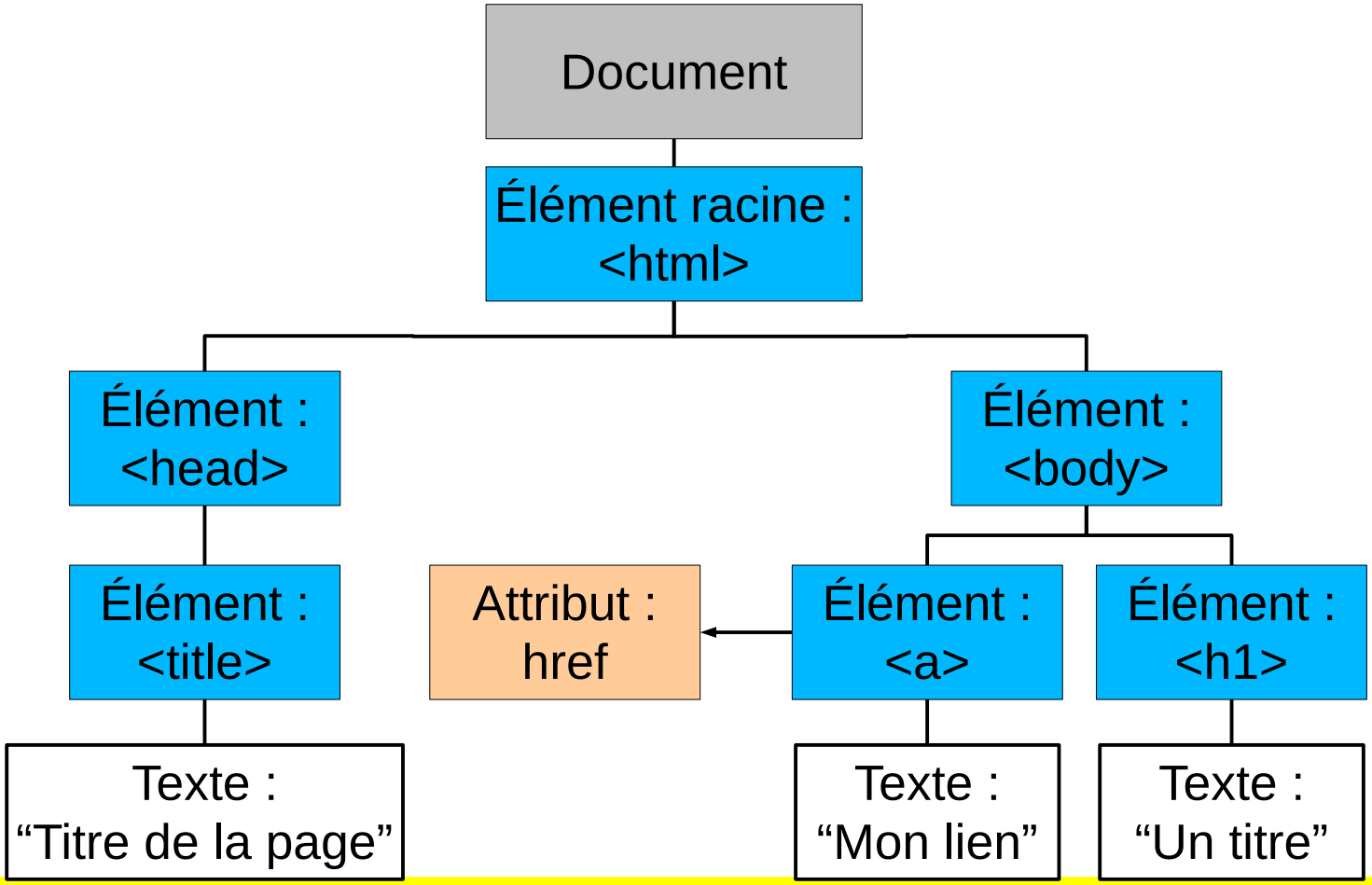
Structure du standard DOM

- Noyau (Core DOM)
  - Modèle standard pour n'importe quel document structuré
- Modèle XML (XML DOM)
  - Objets et propriétés de tous les éléments XML
  - Méthodes pour y accéder et les manipuler
- Modele HTML (HTML DOM)
  - API standard pour manipuler des pages HTML
  - Objets et propriétés de tous les éléments HTML
  - Méthodes pour y accéder et les manipuler

Nœuds DOM

- Tout est un nœud dans un document HTML :
  - le document dans son complexe est un **nœud document**
  - chaque élément HTML est un **nœud élément**
  - le texte dans les éléments HTML est constitué par des **nœuds texte**
  - chaque attribut HTML est un **nœud attribut**
  - même les commentaires sont des **nœuds commentaire**.
- Le document lui-même est un arbre, dont la racine est le nœud document
- On peut accéder à tous les nœuds de l'arbre avec JavaScript

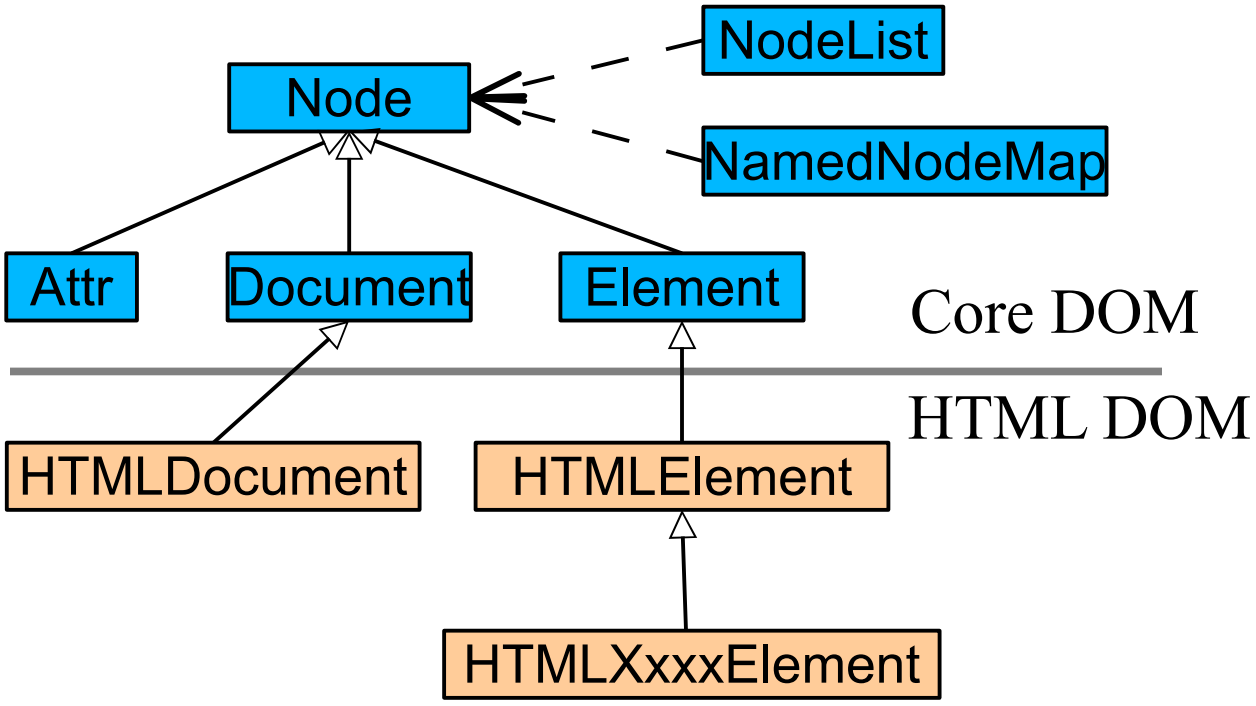
Example d'arbre DOM



Interface de programmation

- L'objet « document » encapsule le document HTML
  - Méthode getElementById(id)
  - getElementsByTagName(n) et getElementsByClassName(n)
- Objets nœuds éléments
  - Méthode appendChild(nœud) : ajoute un nouveau nœud fils
  - Méthode removeChild(nœud) : élimine un nœud fils
  - Propriétés :
    - innerHTML : le texte HTML contenu dans un nœud
    - parentNode : le nœud parent d'un nœud
    - childNodes : la collection des nœuds fils d'un nœud
    - attributes : les nœuds attributs d'un nœud

Les objets DOM et leurs relations



Objets du noyau

- Node, représente un nœud d'un document HTML
- NodeList, une collection ordonnée de nœuds, dont les éléments peuvent être récupérés par leur indice à l'aide de la méthode item(i), où i = 0, 1, . . . , length
- NamedNodeMap, une collection non ordonnée de nœuds, dont les éléments peuvent être récupérés par leur nom
- Document, qui représente un document abstrait et contient des méthodes pour créer, modifier et récupérer des nœuds
- Element, un élément HTML (donc un cas particulier d'un Node)
- Attr, un attribut d'un élément HTML (cas particulier d'un Node)

Objets HTML

- HTMLDocument (hérite de Document)
- HTMLElement (hérite de Element)
- Pour chaque balise HTML, l'objet correspondant s'appelle HTMLXxxxElement, par example :
  - <a> → HTMLAnchorElement
  - <p> → HTMLParagraphElement
  - <input> → HTMLInputElement
  - → HTMLUListElement
  - Etc.

Objets « navigateur »

- Window
- Navigator
- Screen
- History
- Location

- Ces objets permettent d'accéder à des propriétés et à des méthodes qui ne sont pas liées au document, mais à l' « agent utilisateur » qui le visualise

AJAX

- Abréviation de « asynchronous JavaScript and XML »
- Principe : faire une requête asynchrone
  - Pour vérifier
  - Pour obtenir
- Requête asynchrone HTTP (uniquement sur le serveur)
- On obtient en retour du XML
  - À exploiter... (c.f. getElementById + innerHTML)
- Le tout sans recharger...

AJAX : XMLHttpRequest

- Constructeur JavaScript qui fournit une API pour échanger des données entre un client et un serveur :
  - http://www.w3.org/TR/XMLHttpRequest/
- Le nom a des raisons historiques, mais
  - Il permet d'utiliser n'importe quel format textuel, pas que XML
  - Il permet d'utiliser autant HTTP que SHTTP (et d'autres)
  - « Requêtes » dans un sens très ample (tout msg HTTP)
- Principe de fonctionnement
  - On crée un objet avec ce constructeur
  - On affecte une méthode gestionnaire à la propriété « onreadystatechange » (callback)
  - Méthodes « open() » et « send() »

Méthode gestionnaire

```
function mon_gestionnaire( ) {
  // test de l'etat d'avancement de la requête
  if ((this.readyState==4) && (this.status==200))
  {
    // recuperation de la réponse
    // au format XML ou Text
    var myXML = this.responseXML; // c'est du DOM
    var myText = this.responseText;
  // … traiter la réponse
  }
}
```

Utilisation de XMLHttpRequest

```
var client = new XMLHttpRequest();
client.onreadystatechange = mon_gestionnaire;

client.open("GET", url);
client.send();

// alternativement :
client.open("POST", url);
client.setRequestHeader("Content-Type",
  "text/plain;charset=UTF-8");
client.send("var1=va1&var2=val2&…");
```

Génération de la réponse

- Fonction « header() » avec Content-type
  header('Content-type: text/xml;');
- Écriture du XML à la « main » (façon HTML)
- Ou avec une api XML...

- ... ou JSON

JavaScript et PHP

- Le PHP peut générer du JavaScript
- Javascript et Php peuvent communiquer par requête
  - La génération du XML s'intègre alors à l'architecture logiciel...
  - Exemple : un getView qui retourne du HTML encapsulé dans du PHP

    <![CDATA[....]]>
  - On récupère avec « responseText() » et on applique avec getElementById et innerHTML...

Exemple d'encapsulation d'AJAX

- Fichiers JavaScript disponible dans l'archive (TP sur JSON)
- À inclure dans les pages les utilisant
- Principe des librairies...

- Fonctionnement :
  - Le script réalise l'appelle
  - Il faut définir une « page » PHP recevant l'appel
  - Cette page retourne du code HTML à insérer dans le document initial
  - Le script remplace le contenu d'une balise HTML identifiée par son *id*

Objet par l'exemple : « requete »

```
function requete(id, url, params, concatenation, fretour)
{
  // [...]
  // définition d'un champ de classe
  this.id = id;
  // définition d'une méthode
  this.retour = fretour;
  // [...]
  // définition de la requête ajax
  this.xmlreq = new XMLHttpRequest();
  this.xmlreq.onreadystatechange =
    new Function("requestStateChange("+indice+")");
  // [...]
  // enregistrement objet (pour plusieurs requêtes simultanées)
  var indice = requestManager.current++;
  requestManager.request[indice] = this;
}
```

Objet par l'exemple : une collection

```
requestManager = new Object();

requestManager.request = new Array();

requestManager.current = 0 ;
```

Retour de la requête...

```
function requestStateChange(indice) {
  var req = requestManager.request[indice]; // objet REQUETE
  // test etat d'avancement du telechargement (http request)
  if((req.xmlreq.readyState==4) && (req.xmlreq.status==200)) {
    /** * @type Document */
    var myXML = req.xmlreq.responseXML; // la réponse XML reçue
    var r = myXML.getElementsByTagName("reponse").item(0);
    /** * @type HTMLElement */
    var htmlElem = document.getElementById(req.id);
    if(htmlElem)  // l'élément HTML à modifier
    {
      var htmltxt = "";
      if(r.textContent) htmltxt = r.textContent;
      if(req.concatenation == 0) htmlElem.innerHTML = htmltxt;
      else if(req.concatenation == 1)
        htmlElem.innerHTML += htmltxt;  // concaténation
      else htmlElem.innerHTML = htmltxt + htmlElem.innerHTML;
    }
    if((req.retour) && (typeof(req.retour) == 'function'))
      req.retour();  // appel à la fonction de retour
  }
}
```

Utilisation d'AJAX : client

HTML :

Événement déclencheur

Renvoie false : le formulaire ne sera pas soumis !

```
<form method="post" onsubmit="return ajax(this);">
  <!-- … →
</form>
```

JavaScript :

Id de l'élément HTML de retour, là où seront mises les informations

Page PHP

```
function ajax(form) {
  new requete("contenu", "json-02.php",
    new Array(new Array("adresse", form.adresse.value)),
    0, retourajax);
  return false;
}
```

Fonction de retour

Paramètres sous forme de tableau de tableau à 2 dimension (nom du param, valeur)

Précisions sur le constructeur « requete »

- new requete(...) peut avoir jusqu'à 5 paramètres
  - Le 1er : obligatoire : id de la balise de retour
  - Le 2ème : obligatoire : l'url où obtenir l'information
  - Le 3ème : optionnel : les paramètres de la requête
    tableau de tableau à 2 dimension (nom du param, valeur)
  - Le 4ème : optionnel : (un int qui vaut 0, 1 ou 2) pour dire si on va
    - 0 : remplacer le contenu de la balise identifiée par id,
    - 1 : concaténer
    - 2 ou autre valeur : insérer avant
  - Le 5ème : optionnel : fretour : fonction à rappeler après avoir reçu le HTML à intégrer dans la page

Utilisation d'AJAX : serveur

```php
<?php
  // calcul de $corps

  // génération de XML
  header("Content-type: text/xml");
  echo "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n";

  // la DTD
  echo '<!DOCTYPE reponse [
    <!ELEMENT reponse (#PCDATA)>
  ]>';

  // la réponse elle-même en CDATA
  // pour contenir des balises HTML
  echo "\n<reponse><![CDATA[$corps]]></reponse>";
?>
```

Exemple d'XML généré

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE reponse [
  <!ELEMENT reponse (#PCDATA)>
]>
<reponse><![CDATA[
<p style='clear:both;border-top: black thin solid;margin:
2em;'></p><iframe style='border: none;box-shadow: 1px 1px 3px
black;float: left; margin: 0 2em 2em 0;width:600px;
height:480px;'
src='http://www.openstreetmap.org/export/embed.html?
bbox=7.071304798126 22%2C43.615215301513 7%2C7.072373867034 91%2C4
3.615867614746 1&amp;layer=mapnik' ></iframe><br/><small><a
href='http://www.openstreetmap.org/#map=17/47.32851/6.01093'>Vi
ew Larger Map</a></small>'<article>Le temps à Sophia
Antipolis : <img alt=''
src='http://openweathermap.org/img/w/04n' /> Température de
12.31°C, nuageux. </article>]]></reponse>
```

Merci de votre attention