

# MIF13

## Programmation Web



### ASYNCHRONOUS JAVASCRIPT AND XML (CONCEPTION ET DÉVELOPPEMENT D'APPLICATIONS WEB RICHES)

LIONEL MÉDINI  
SEPTEMBRE-DÉCEMBRE 2014

# Plan du cours



- Introduction et rappels JavaScript
- Programmation XML en JavaScript
  - Retour sur l'API DOM (Document Object Model)
  - Utilisation du DOM XML en JavaScript
- Asynchronous Javascript And XML (AJAX)
  - Mécanismes de requêtes asynchrones
  - Composants d'une application
  - Quelques patterns de conception en AJAX
  - Outils de conception et de développement
- JavaScript avancé
  - Quelques exemples de fonctionnalités HTML5
- Conclusion

# Introduction



- Objectif : concevoir des applications Web « riches »
  - Web-based
    - ✦ Paradigme client-serveur, HTTP
      - ➔ Programmation côté serveur et côté client
  - Expérience utilisateur proche des applications natives
    - ✦ Interface utilisateur fluide, ergonomique, dynamique
      - ➔ Traitement de l'interface côté client (JavaScript, CSS , DOM)
      - ➔ Échanges client-serveur asynchrones (AJAX)
  - Logique métier complexe
    - ✦ Outils « évolués » de modélisation, conception, développement
      - ➔ IDE, POO, UML, design patterns, méthodes agiles, XP...
- ➔ Où placer la logique métier ? La couche données ?

# Rappels JavaScript / ECMAScript



- L'offre côté client (rappel)
  - Moteur de rendu : cf. cours 1
  - Moteur de scripting :
    - ✦ Version de base : ECMAScript-262 (1999)
    - ✦ Plusieurs versions découlant de cette spécification
      - 3 (1999), 4 (abandonné), 5 (2009), Harmony (work in progress)
      - Chaque moteur implémente une version différente du langage :  
[http://en.wikipedia.org/wiki/ECMAScript#Version\\_correspondence](http://en.wikipedia.org/wiki/ECMAScript#Version_correspondence)
      - JavaScript 2.0 sera totalement compatible avec ECMAScript Harmony
    - ✦ Moteurs JavaScript (~1.8)
      - SpiderMonkey (Firefox), V8 (Chrome), Nitro (Safari) : ECMAScript 5
    - ✦ Moteur JScript (9.0) implémenté dans :
      - Trident (Microsoft IE) : ECMAScript 5

# Rappels JavaScript / ECMAScript



- **Caractéristiques**
  - Interprété
    - ✦ Typage dynamique
  - Fonctionnel
    - ✦ Fonctions anonymes
    - ✦ First-class functions
  - Orienté prototype
    - ✦ « object-based » plutôt qu'« object-oriented »
    - ✦ Typage dynamique : types associés aux instances et non aux classes
  - Événementiel
    - ✦ Mécanismes de « callback »
    - ✦ Pattern observer : `addEventListener`
- **Fonctionnalités**
  - Reflection
  - E4X : ECMAScript for XML (ECMA-357)
  - JSON
  - ...

# Rappels JavaScript / ECMAScript



- Rappels de programmation fonctionnelle
  - S'appuie sur une pile d'appels imbriqués (contexte)
  - Fermeture (closure)
    - ✦ Permet de capturer l'environnement d'une fonction

```
function makeFunc() {  
    var name = "Mozilla";  
    function displayName() {  
        alert(name);  
    }  
    return displayName;  
}  
  
var myFunc = makeFunc();  
myFunc();
```

## Sources

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures>

<https://developer.mozilla.org/fr/docs/JavaScript/Guide/Closures>

# Rappels JavaScript / ECMAScript



- Rappels de programmation fonctionnelle
  - S'appuie sur une pile d'appels imbriqués (contexte)
  - Function factory
    - ✦ Permet de passer des paramètres au moment de la création d'une fonction

```
function makeAdder(x) {  
  return function(y) {  
    return x + y;  
  };  
}  
  
var add5 = makeAdder(5);  
var add10 = makeAdder(10);  
  
console.log(add5(2)); // 7  
console.log(add10(2)); // 12
```

## Sources

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures>

<https://developer.mozilla.org/fr/docs/JavaScript/Guide/Closures>

# Rappels JavaScript / ECMAScript



- **Programmation orientée prototype**
  - POO sans classe : on ne manipule que des objets
  - Objets représentés sous forme de dictionnaires (tableaux associatifs)
  - Propriétés
    - ✦ Pas de distinction entre les propriétés (attributs/méthodes) d'un objet
    - ✦ On peut remplacer le contenu des propriétés et en ajouter d'autres
  - Réutilisation des comportements (héritage)
    - ✦ se fait en clonant les objets existants, qui servent de prototypes
  - Sources
    - ✦ [http://fr.wikipedia.org/wiki/Programmation\\_orient%C3%A9e\\_prototype](http://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_prototype)
    - ✦ [http://en.wikipedia.org/wiki/Prototype-based\\_programming](http://en.wikipedia.org/wiki/Prototype-based_programming)



# Rappels JavaScript / ECMAScript

```
// Example of true prototypal inheritance style in JavaScript.
// "ex nihilo" object creation using the literal object notation {}.

var foo = {name: "foo", one: 1, two: 2};
var bar = {three: 3};

// For the sake of simplicity, let us pretend that the following line works
// regardless of the engine used:
// bar.__proto__ = foo;
// bar.[[ prototype ]] = foo
bar = Object.create( foo ); // JS 1.8.5 } // foo is now the prototype of bar.

// If we try to access foo's properties from bar from now on, we'll succeed.
bar.one // Resolves to 1.

// The child object's properties are also accessible.
bar.three // Resolves to 3.

// Own properties shadow prototype properties
bar.name = "bar";
foo.name // unaffected, resolves to "foo"
bar.name // Resolves to "bar"
```

Source : [http://en.wikipedia.org/wiki/Prototype-based\\_programming](http://en.wikipedia.org/wiki/Prototype-based_programming)

# Rappels JavaScript / ECMAScript



- Comment programmer « proprement » en orienté-objet
  - Plutôt « object-based » qu'« object-oriented »
  - ➔ Pour programmer en objet, il faut simuler des objets
    - ✦ Créer des constructeurs
    - ✦ Encapsuler les données (avec « `this` »)
    - ✦ Utiliser des « inner functions » à l'intérieur du constructeur
  - Exemple
    - <http://www.sitepoint.com/article/oriented-programming-1/>

# Rappels JavaScript / ECMAScript



- Fonctions de rappel (callback)

- Définition

- ✦ Fonction qui est passée en paramètre à une autre fonction afin que cette dernière puisse en faire usage

- Exemple : soient une fonction A et une fonction B

- ✦ Lors de l'appel de A, on lui passe en paramètre la fonction B : A(B)
    - ✦ Lorsque A s'exécutera, elle pourra exécuter la fonction B

- Intérêt : faire exécuter du code

- ✦ Sans savoir ce qu'il va faire (défini par un autre programmeur)
    - ✦ En suivant une interface de programmation qui définit
      - Le nombre et le type des paramètres en entrée
      - Le type de la valeur en sortie

- Source :

<http://www.eppershand.net/developpement/algorithmie/explication-utilite-fonctions-callback>

# Rappels JavaScript / ECMAScript



- Fonctions de rappel (callback)
  - La fonction qui reçoit une callback en paramètre doit respecter son interface

```
fonctionNormale(fonctionCallBack) {... fonctionCallback(argument); ...}
```
  - 2 syntaxes pour le passage d'une fonction callback en argument d'une autre fonction
    - ✦ Sans paramètre : directement le nom de la fonction

```
fonctionNormale(fonctionCallback);
```
    - ✦ Avec paramètre : encapsulation dans une fonction anonyme

```
fonctionNormale(function() { fonctionCallback(arg1); });
```

# Rappels JavaScript / ECMAScript



- Programmation événementielle
  - ✧ Deux processus en parallèle
    - Principale : déroulement des traitements et association des événements à des fonctions de callback
    - Callbacks : récupèrent et traitent les événements
  - ✧ Deux syntaxes
    - DOM 0 : attributs HTML / propriétés JavaScript spécifiques onclick, onload...  
(<http://www.w3.org/TR/html4/interact/scripts.html#h-18.2.3>)
    - DOM 2 : ajout d'eventListeners en JavaScript  
monElement.addEventListener("click", maFonctionCallback, false);
      - Remarques :
        - Le troisième paramètre indique le type de propagation dans l'arbre DOM
        - Internet Explorer utilise la méthode attachEvent() au lieu de addEventListener()

Source : <http://www.alsacreations.com/article/lire/578-La-gestion-des-evenements-en-JavaScript.html>

# Rappels JavaScript / ECMAScript



- Programmation événementielle

- ✦ L'objet Event

- Dénote un changement d'état de l'environnement
  - Peut être provoqué par l'utilisateur ou par l'application
- Peut être intercepté à l'aide de code JavaScript
- Possède un **flux d'événement** : propagation dans l'arbre DOM
  - Capture : du nœud Document au nœud visé par l'événement
  - Cible : sur le nœud visé
  - Bouillonnement (bubling) : remontée jusqu'au nœud document
- Principales propriétés
  - **type** : type de l'événement ("click", "load", "mouseover"...)
  - **target** : élément cible (élément a pour un lien cliqué)
  - **stopPropagation** : arrête le flux d'un événement
  - **preventDefault** : empêche le comportement par défaut (navigation quand un lien est cliqué)

Source : <http://www.alsacreations.com/article/lire/578-La-gestion-des-evenements-en-JavaScript.html>

# Outils de programmation avec XML



- Définitions

- Qu'est-ce qu'un parser ?

- ✦ « Un module logiciel [...] utilisé pour lire les documents XML et pour accéder à leur contenu et à leur structure. »

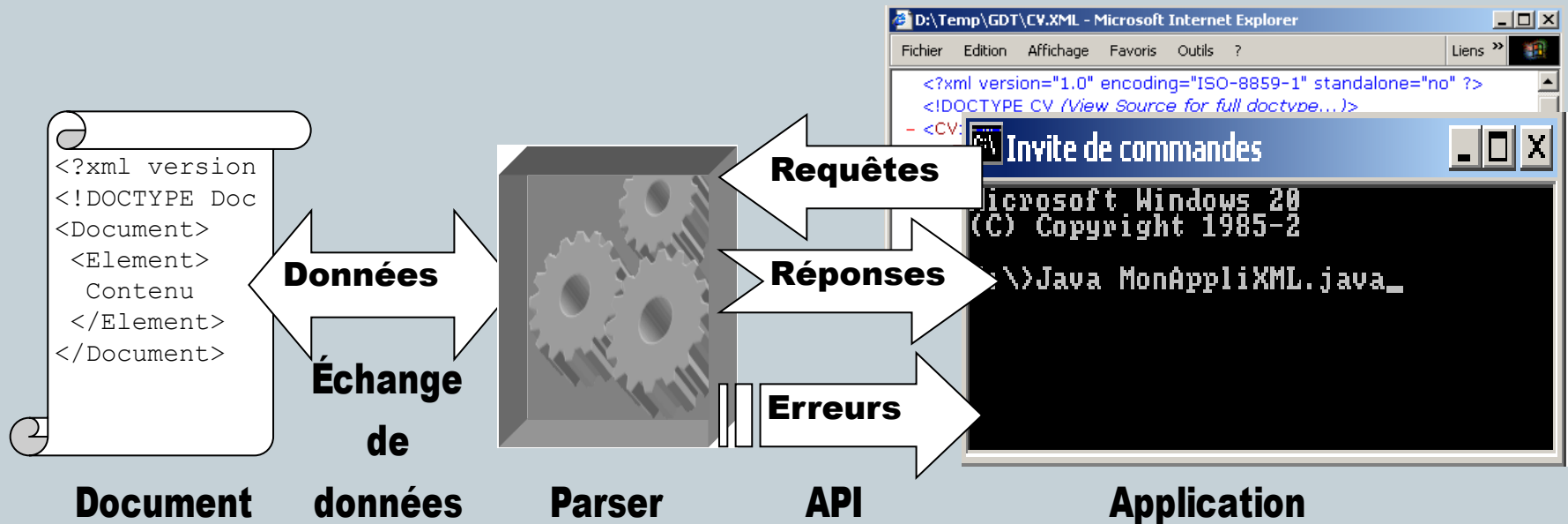
- Qu'est-ce qu'une application ?

- ✦ « On suppose qu'un processeur XML effectue son travail pour le compte d'un autre module, appelé l'application. »

[http://babel.alis.com/web\\_ml/xml/REC-xml.fr.html#dt-xml-proc](http://babel.alis.com/web_ml/xml/REC-xml.fr.html#dt-xml-proc)

# Outils de programmation avec XML

- Communications entre parsers et applications
  - Rappel : Application Programming Interface
    - ✦ Outils
    - ✦ Protocole de communication
  - Schéma des échanges de données





# L'API DOM (Document Object Model)



- Généralités
  - Modèle objet de document
  - Motivations
    - ✦ Rendre les applications W3 dynamiques
    - ✦ Accéder aux documents HTML et XML depuis un langage de programmation
  - Utilisations courantes
    - ✦ Intégré aux navigateurs
    - ✦ Utilisé en programmation comme API XML
  - Origine : DOM working group (W3C)
    - ✦ Début : 1997 ; fin : ...
    - ✦ But : standardiser les tentatives existantes

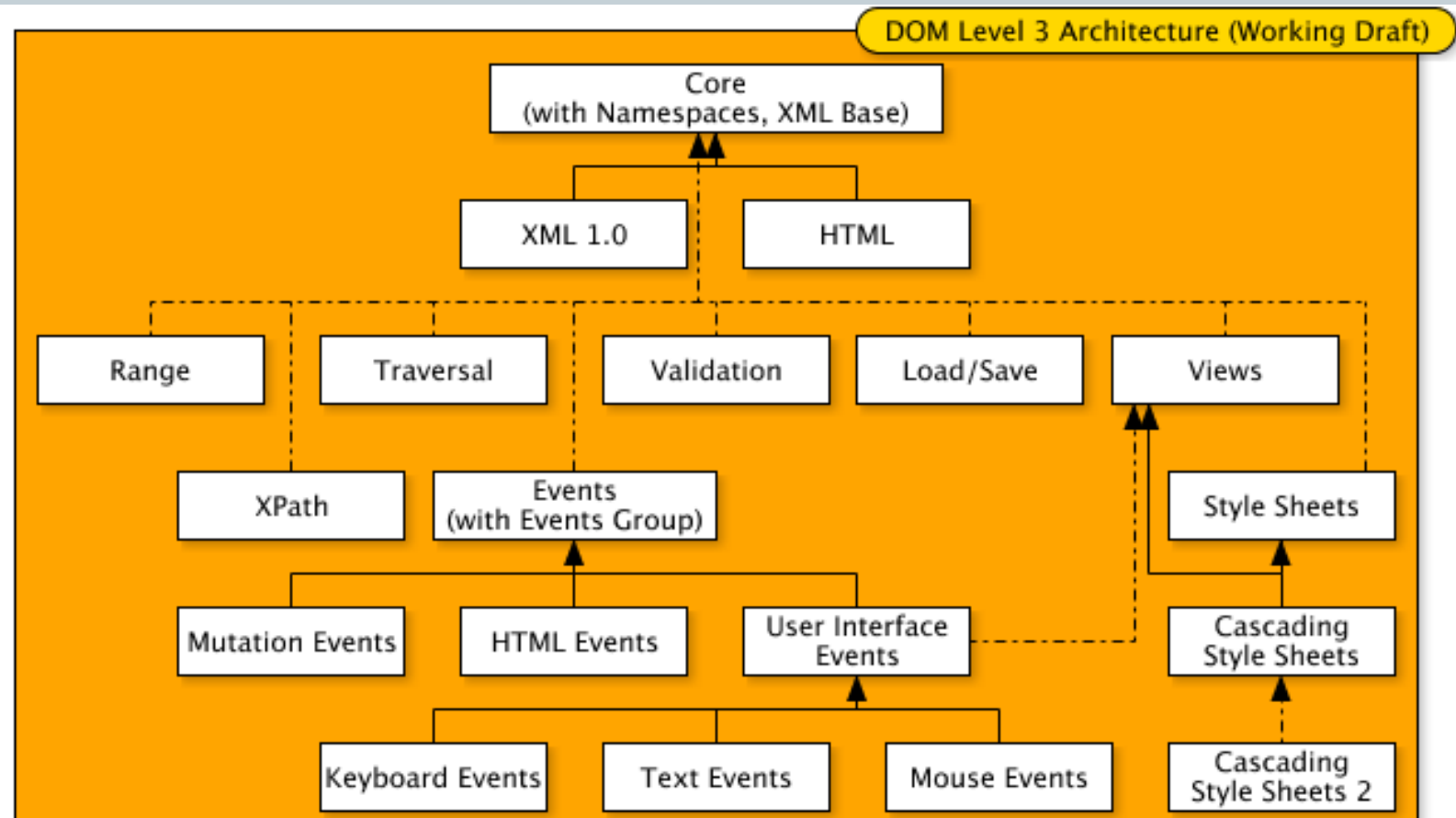
# L'API DOM (Document Object Model)



- Principes fondamentaux
  - Représentation arborescente d'un document
    - ✦ Tout le document est chargé en mémoire
    - ✦ Navigation dans la structure arborescente
    - ✦ Représentation des nœuds par des *interfaces*
      - Propriétés
      - Méthodes
  - Recommandations sous forme de niveaux
    - ✦ Niveau 0 : avant...
    - ✦ Niveau 1 : octobre 1998
    - ✦ Niveau 2 : depuis novembre 2000
    - ✦ Niveau 3 : depuis janvier 2004

# L'API DOM (Document Object Model)

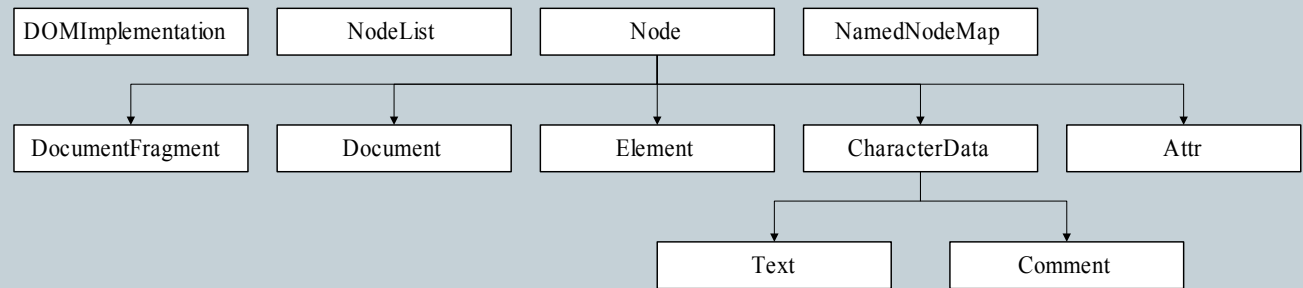
- Fonctionnalités



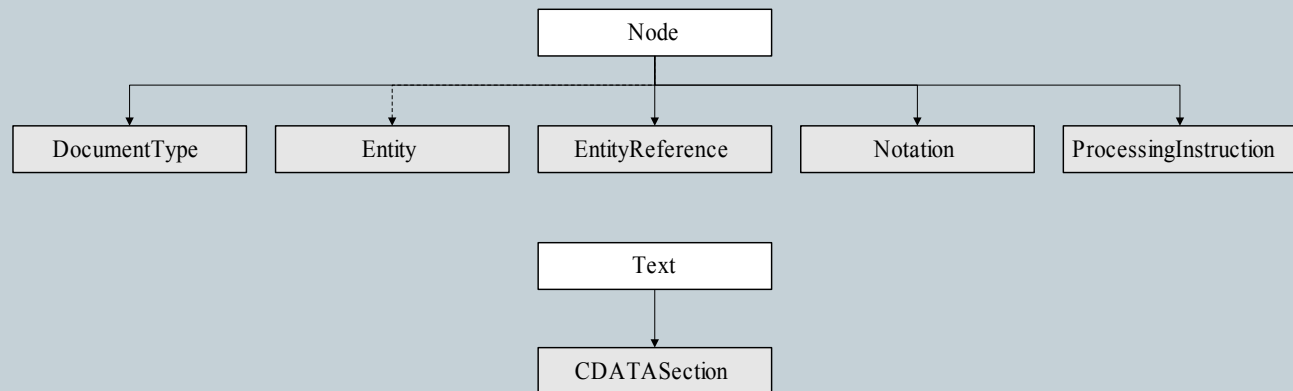
# L'API DOM (Document Object Model)

- Interfaces

- DOM Core :



- DOM XML :



# L'API DOM (Document Object Model)



- Interfaces DOM (Core et XML) les plus utilisées
  - Node : tout type de nœud de l'arbre DOM
    - ✦ Constantes  
Tous les types de nœuds définis (exemple : `node.ELEMENT_NODE`)
    - ✦ Propriétés  
`nodeName`, `nodeType`, `nodeValue`, `childNodes`, `textContent`
    - ✦ Méthodes  
`insertBefore()`, `replaceChild()`, `removeChild()`, `appendChild()`, `cloneNode()`
  - NodeList : comme son nom l'indique...
    - ✦ Propriétés : `length`, `item(i)`

# L'API DOM (Document Object Model)



- Interfaces DOM (Core et XML) les plus utilisées
  - Document : le nœud racine de l'arbre DOM
    - ✦ Dérive de Node
    - ✦ Propriétés  
doctype, documentElement, encoding
    - ✦ Méthodes  
createElement(name), createTextNode(), createAttribute(name),  
getElementById(id), getElementsByTagName(name)
  - DocumentFragment : partie d'un document ; cf. Node

# L'API DOM (Document Object Model)



- Interfaces DOM (Core et XML) les plus utilisées
  - Element : un élément, au sens HTML ou XML
    - ✦ Propriété : tagName
    - ✦ Méthodes
      - getAttribute(name), setAttribute(name, value), hasAttribute(name),
      - getAttributeNode(name), setAttributeNode(node),
      - removeAttribute(name), removeAttributeNode(node),
  - Attr : un attribut...
    - ✦ Propriétés : name, value, ownerElement

# L'API DOM (Document Object Model)



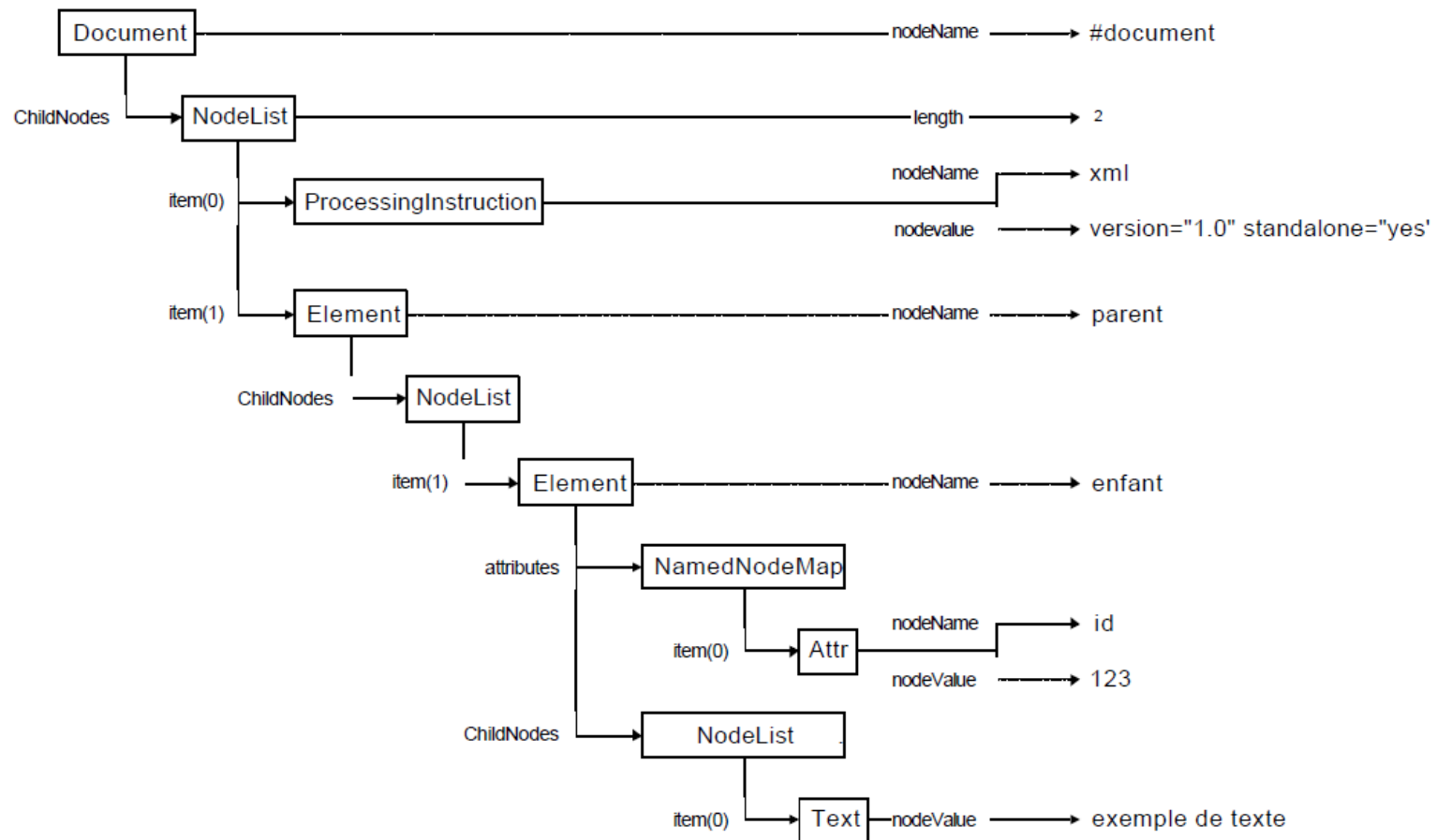
- Interfaces DOM (Core et XML) les plus utilisées
  - Text : nœud textuel (sous-nœud d'un élément)
    - ✦ Propriétés  
data, length (héritées de CharacterData)
    - ✦ Méthodes  
appendData(), insertData(), deleteData(), replaceData(),  
substringData() (héritées de CharacterData)  
replaceWholeText()



# L'API DOM (Document Object Model)

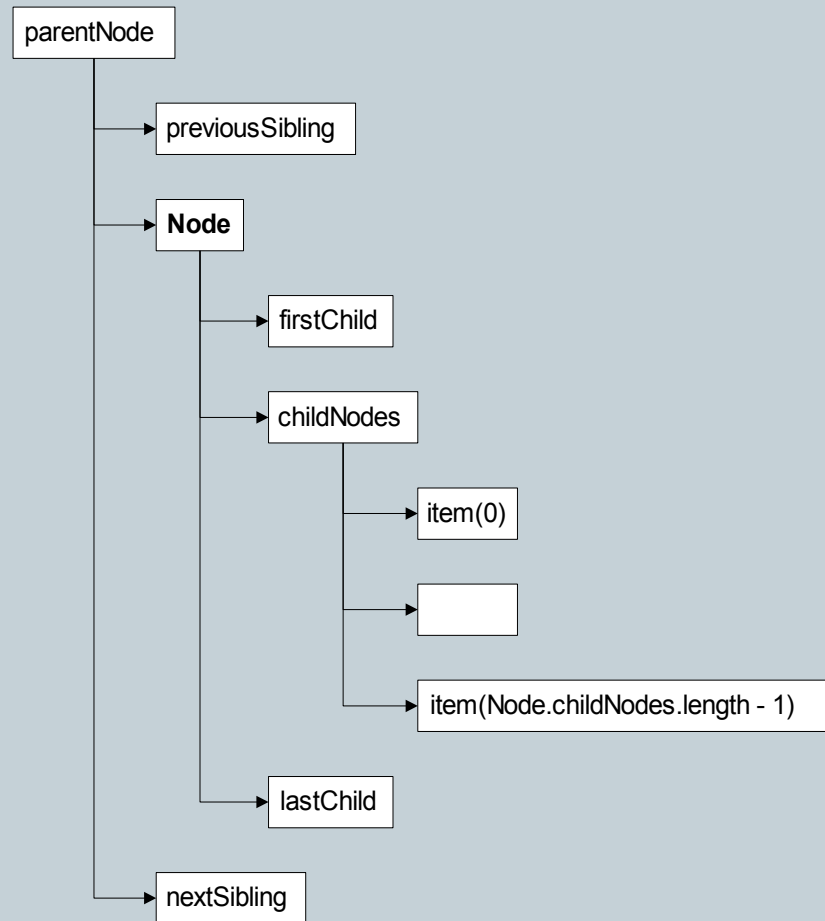


- Hiérarchisation des interfaces d'un document XML



# L'API DOM (Document Object Model)

- Déplacement dans une arborescence DOM (interfaces du module Core)



# L'API DOM (Document Object Model)

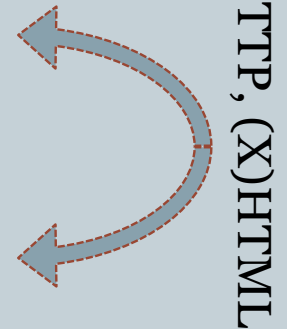


- Conclusion sur le DOM
  - Utilisation du DOM XML en JavaScript
    - ✦ Utilisation directe des propriétés
    - ✦ DOM XML relativement standardisé sur les navigateurs récents
      - Exemple : `document.getElementById()`
    - ✦ En revanche, DOM HTML plus dépendant du navigateur
      - Exemple : `monElement.innerHTML += ...;`  
n'interprétait pas le nouveau code HTML sous IE 6 et 7
- Références sur le DOM
  - <http://www.w3.org/DOM/>
  - <http://www.w3schools.com/dom/>

# Asynchronous Javascript And XML (AJAX)



- Composants d'une application Web « classique »
  - Côté serveur
    - ✦ Contrôleur général de l'application (index.jsp)
    - ✦ Ressources statiques
      - Modèle de document, bibliothèques de scripts, feuilles de style
    - ✦ Traitements dynamiques des données (couche métier)
    - ✦ Composition dynamique de l'interface (couche vue)
  - Côté client
    - ✦ Gestion des événements utilisateur
    - ✦ Composition dynamique de l'interface (couche vue)



# Asynchronous Javascript And XML (AJAX)



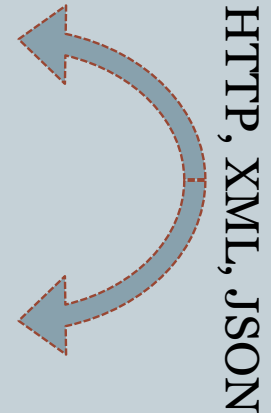
- Composants d'une application Web AJAX

- Côté serveur

- ✦ Contrôleur général de l'application (index.php)
- ✦ Ressources statiques
  - Modèle de document, bibliothèques de scripts, feuilles de style
- ✦ Traitements dynamiques des données (couche métier)

- Côté client

- ✦ Contrôleurs délégués relatifs à un type de vue
- ✦ Gestion des événements utilisateur
- ✦ Traitement des données reçues (couche métier)
- ✦ Composition dynamique de l'interface (couche vue)



# Asynchronous Javascript And XML (AJAX)



- Généralités sur AJAX
  - Applications web avec interface utilisateur
  - Déporter un maximum de code sur le client
    - ✦ Réduction des ressources consommées côté serveur
    - ✦ Réduction de la bande passante réseau
  - Applications Web AJAX les plus connues
    - ✦ Google (Mail, Map, Earth...)
    - ✦ Suggestions automatiques
    - ✦ Traitement de texte
    - ✦ ...
  - Exemple
    - ✦ <http://www.standards-schmandards.com/exhibits/ajax/>

# Asynchronous Javascript And XML (AJAX)



- **Fonctionnement**
  - Requête asynchrone au serveur dans une fonction JavaScript (déclenchée par un événement quelconque)
  - Transfert asynchrone de données en XML
  - Traitement dynamique côté client
    - ✦ Affichage (inclusion au document HTML, transformation XSLT...)
    - ✦ Logique applicative (fonctions JavaScript dédiées)
- **Spécificité de la technologie AJAX**
  - Requête asynchrone sur un document XML *via* un
    - ✦ Objet XMLHttpRequest (Mozilla)
    - ✦ Contrôle ActiveX XMLHttpRequest (IE)

# Asynchronous Javascript And XML (AJAX)



- **Fonctionnement**

- Étapes d'une communication AJAX côté client

- ✦ Envoi de la requête

- Créer un objet requête

- Spécifier les éléments de la requête

- URL, méthode , headers HTTP, paramètres

- Lui associer un gestionnaire d'événement

- L'envoyer

- ✦ Réception de la réponse

- À chaque changement d'état de la requête, tester si l'état est « ready »

- Traiter les données reçues

- Ajout à l'interface, transformation XSL...



# Asynchronous Javascript And XML (AJAX)



- **Fonctionnement**
  - Étapes d'une communication AJAX côté serveur
    - ✦ Que doit faire un serveur Web à la réception d'une requête asynchrone AJAX ?

# Asynchronous Javascript And XML (AJAX)



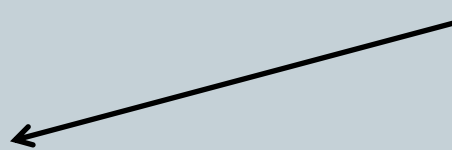
- Exemple de code : création d'un objet requête

```
var req = null;
```

```
function getRequest()
```

```
{  
  if (window.XMLHttpRequest)  
  {  
    req = new XMLHttpRequest();  
  }  
  else if (typeof ActiveXObject != "undefined")  
  {  
    req=new ActiveXObject("Microsoft.XMLHTTP");  
  }  
  return req;  
}
```

**Safari / Mozilla**



**Internet Explorer**



# Asynchronous Javascript And XML (AJAX)



- Exemple de code : chargement asynchrone

```
function GetDataUsingAJAX (HttpMethod, url, params, elt)
{
    if(req != null)
    {
        // méthode avec paramètres
        req.onreadystatechange = function() {stateChange(elt)};
        // méthode sans paramètre
        // req.onreadystatechange = stateChange;

        req.open(HttpMethod, url, true);
        req.setRequestHeader("Accept", "application/xml");
        req.send(params);
    }
}
```

**Association  
d'une fonction  
de callback  
aux  
changements  
d'état de la  
réponse**



# Asynchronous Javascript And XML (AJAX)



- Exemple de code : gestion de l'état

```
function stateChange (elt)
{
    if(req.readyState == 4) {
        if (req.responseXML != null) {
            var docXML= req.responseXML;
        } else {
            var docXML= req.responseText;
            docXML=parseFromString(docXML);
        }
        var docXMLresult = traiteXML(docXML);
        var str = (new XMLSerializer()).serializeToString(docXMLresult);
        document.getElementById(elt).innerHTML += str;
    }
}
```

**READY\_STATE\_COMPLETE** ←

# Asynchronous Javascript And XML (AJAX)



- Exemple de code : transformation XSLT

//Après chargement asynchrone des documents XML et XSLT

function transform XSLT (XMLDoc, XSLDoc, id)

```
{  
  if(XMLDoc == null || XSLDoc == null) {return;}
```

```
  try {
```

```
    if (window.ActiveXObject)
```

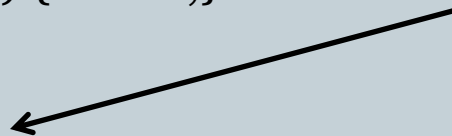
```
    {
```

```
      var target = document.getElementById(id);
```

```
      target.innerHTML = xml.transformNode(xsl);
```

```
    }
```

**Internet Explorer**



# Asynchronous Javascript And XML (AJAX)



- Exemple de code : transformation XSLT

```
} else if (window.XSLTProcessor) {  
    var fragment;  
    var xsltProcessor = new XSLTProcessor();  
    xsltProcessor.importStylesheet(xsl);  
    fragment = xsltProcessor.transformToFragment(xml, document);  
    var target = document.getElementById(id);  
  
    target.appendChild(fragment);  
}  
} catch (e) {  
    return e;  
}  
}
```

← **Safari / Mozilla**

# Asynchronous Javascript And XML (AJAX)



- Implémentation de la logique applicative
  - Programmation d'un ensemble de fonctions JavaScript
    - ✦ Réécriture de fonctionnalités existantes
    - ✦ Mélange de la logique métier et des fonctionnalités techniques
    - ✦ Pas forcément à l'épreuve des changements technologiques
    - ✦ Réutilisabilité moyenne
    - ✦ Code parfois un peu « fouillis »
  - ➔ Utiliser / s'approprier des outils existants
    - ✦ Langages / EDI spécifiques (ou plugins de votre EDI préféré)
    - ✦ Lirairies / frameworks open source

# Asynchronous Javascript And XML (AJAX)



- Implémentation de la logique applicative
  - Standardisation de la communication avec les langages de programmation côté serveur : JSON
    - ✦ Spécification liée à ECMAScript – RFC 4627
    - ✦ Implémentée par tous les navigateurs
    - ✦ Permet de sérialiser des types de données (alternative à XML)
    - ✦ Définit des types de données de façon simple
    - ✦ Indépendant du langage de programmation utilisé
      - ➔ Permet les échanges de données entre serveur et client
    - ✦ Syntaxe : des inclusions
      - d'objets sous forme d'une liste de membres  
{ nommembre1 : valmembre1, nommembre2: valmembre2, ... }
      - de tableaux sous forme d'une liste de valeurs  
[ valeur1, valeur2, valeur3, ...]



# Asynchronous Javascript And XML (AJAX)

- Implémentation de la logique
  - Standardisation de la communication programmation côté serveur :
    - ✦ Exemple de fichier au format JSON :

```
{ "menu": "Fichier", "commandes":  
[ { "title": "Nouveau",  
  "action": "CreateDoc" }, {  
  "title": "Ouvrir", "action":  
  "OpenDoc" }, { "title": "Fermer",  
  "action": "CloseDoc" } ] }
```

- ✦ Equivalence en XML :

- Source :

<http://www.xul.fr/ajax-format-json.html>

```
<?xml version="1.0" ?>  
<root>  
  <menu>Fichier</menu>  
  <commands>  
    <item>  
      <title>Nouveau</value>  
      <action>CreateDoc</action>  
    </item>  
    <item>  
      <title>Ouvrir</value>  
      <action>OpenDoc</action>  
    </item>  
    <item>  
      <title>Fermer</value>  
      <action>CloseDoc</action>  
    </item>  
  </commands>  
</root>2
```

# Asynchronous Javascript And XML (AJAX)



- Implémentation de la logique applicative
  - Standardisation de la communication avec les langages de programmation côté serveur : JSON
    - ✦ Utilisation côté client :

```
req.open("GET", "fichier.json", true); // requête
...
var doc = eval('(' + req.responseText + ')'); // récupération
...
var nomMenu = document.getElementById('jsmenu'); // recherche
nomMenu.value = doc.menu.value; // assignation
...
doc.commands[0].title // lire la valeur "title" dans le tableau
doc.commands[0].action // lire la valeur "action" dans le tableau
```

- ✦ Utilisation côté serveur : librairies *ad hoc*

# Asynchronous Javascript And XML (AJAX)



- Quelques règles de conception en AJAX
  - Utiliser des design patterns
    - ✦ Adaptateur
      - Le plus utilisé
      - Testez la fonctionnalité à utiliser, pas le navigateur...
    - ✦ MVC
      - De préférence type 2 (avec contrôleurs délégués)
      - Isoler les parties du modèle
      - Répartir les traitements de chaque partie entre serveur et client
      - Indiquer à la vue comment restituer les objets du modèle
    - ✦ Observateur
      - Permet de définir un modèle événementiel
      - Si celui de JavaScript est insuffisant
      - Il en existe plusieurs dans des librairies open source (W3C)
    - ✦ ...

# Asynchronous Javascript And XML (AJAX)



- Outils de conception et de développement
  - Bibliothèques
    - ✦ Ensembles de fonctions JavaScript réalisant des traitements spécifiques
    - ✦ Peuvent être réutilisées dans des applications
  - Frameworks AJAX
    - ✦ Programmation dans un autre langage
    - ✦ Génération du code JavaScript
    - ✦ Mécanismes de communication standard entre client et serveur
- Référence : <http://www.ajaxpatterns.org/>

# Asynchronous Javascript And XML (AJAX)



- **AJAX a aussi ses inconvénients**
  - Toute une application dans la même page
    - ✦ Bouton « Back » inutilisable
    - ✦ Définition de bookmarks sur une vue particulière impossible
  - Génération dynamique des contenus
    - ✦ Indexation par des moteurs de recherche impossible
  - Téléchargement du code applicatif sur le client
    - ✦ Temps de latence importants au lancement de l'application
  - Nécessite d'avoir activé JavaScript
    - ✦ Prévoir une solution de repli « acceptable » lorsqu'il est désactivé
  - Complexité des développements
    - ✦ Appropriation et utilisation des différentes technos parfois coûteuse

Source : <http://dico.developpez.com/html/1710-Internet-Ajax-Javascript-Asynchrone-et-XML.php>

# Asynchronous Javascript And XML (AJAX)



- **Sécurité**

- Déporter de la logique applicative sur le client présente des risques
- Remarque
  - ✦ L'envoi d'une requête asynchrone XHR à un autre serveur que celui ayant délivré le script est impossible (en principe)
- Types d'attaques
  - ✦ Usurpation de session/d'identité :
    - on ne peut jamais être sûr que le client est celui qu'il prétend être
    - la partie applicative tournant sur le client est-elle réellement celle envoyée par le serveur ?
  - ➔ Double validation (mots de passe)

# Asynchronous Javascript And XML (AJAX)



- Sécurité

- Types d'attaques

- ✦ Cross-site scripting (XSS)

- <http://cwe.mitre.org/top25/index.html#CWE-79>

- <https://www.owasp.org/index.php/XSS>

- violation de la *same-origin policy*

- exécution de scripts malicieux dans le contexte d'un site « trusté »

- exemple: injection de scripts dans les commentaires des forums

- ➔ Revenir au HTML de base pour les données sensibles

- ➔ Vérifier le contenu saisi par les utilisateurs

- ✦ Cross-site request forgery (CSRF)

- <http://cwe.mitre.org/top25/index.html#CWE-352>

- <https://www.owasp.org/index.php/CSRF>

- utiliser l'authentification d'un utilisateur pour réaliser des actions à son insu

- souvent permise par l'authentification par cookies

- ➔ Utiliser des champs hidden ou l'en-tête HTTP Referer

# JavaScript avancé



- Fonctionnalités en lien avec la spécification HTML5
- Philosophie
  - Rapprocher les fonctionnements des navigateurs de ceux des OS
- Exemples de fonctionnalités
  - Sélecteurs CSS : accès standardisé aux contenus de la page
  - Workers : threads
  - WebSockets : streaming, server push, connexion avec d'autres clients (P2P)
  - WebStorage : émulation BD pour stockage des données de session (sessionStorage) ou d'une application (localStorage)
  - GeoLocation
  - Device APIs...
- Implémentations variables selon les moteurs/navigateurs
- Utilisation simplifiée par de nombreuses bibliothèques
- plus de détails : <http://blog.xebia.fr/2010/03/18/html5-les-api-javascript/> ; <http://html5demos.com/>



# Conclusion



- Quelques règles pour développer une application Web riche
  - Outils de développement
    - ✦ Utilisez les ressources à votre disposition
      - Choisissez une bibliothèque aussi standard que possible
      - Il existe aussi des feuilles de style CSS open source  
Exemple : <http://www.oswd.org/>
    - ✦ Vérifiez la compatibilité avec les navigateurs visés
  - Compatibilité avec les navigateurs
    - ✦ Testez la fonctionnalité à utiliser, pas le navigateur...
    - ✦ Utilisez des façades aussi souvent que possible
    - ✦ Restez calmes : <http://webilus.com/tableau/repartition-du-temps-passe-pour-un-webdesign-moderne>

# Conclusion



- De plus en plus d'applications Web « riches »
  - Charge répartie entre client et serveur
  - Outils de conception et de développement matures
  - Bonne ergonomie grâce aux technologies CSS, JavaScript
  - Standardisation
    - ✦ Indépendance vis-à-vis de l'OS
    - ✦ Ne correspond pas aux stratégies des vendeurs d'OS ou de logiciels
  - Disponibles sur Internet
    - ✦ Indépendance vis-à-vis de la machine utilisée

# Perspectives



- Quelles applications Web pour demain ?
  - Deux types d'applications Web
    - ✦ RIA : Rich Internet Application
      - S'exécute dans un navigateur
        - ➔ Doit être compatible avec une majorité de navigateurs
    - ✦ RDA : Rich Desktop Application
      - S'exécute dans un complément installé sur le poste de travail
        - Microsoft SilverLight
        - Adobe AIR
      - ➔ Moins de restrictions de sécurité
  - Ne pas perdre de vue l'arrivée de l'informatique ubiquitaire
    - ✦ PDA, Smartphones, connexions par réseaux GSM
    - ✦ Ressources client réduites : matérielles et logicielles
    - ✦ Fonctionnalités et usages spécifiques : contextualisation, géolocalisation...

# Quelques références



- **Spécifications**

- En règle générale, la vérité est ici : <http://www.w3.org>
- ...Sauf quand elle est ailleurs :
  - ✧ <http://www.ecmascript.org/>
  - ✧ <https://developer.mozilla.org/fr>, <http://xulplanet.com/references/objref/>
  - ✧ [http://msdn.microsoft.com/en-us/library/hbxc2t98\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/hbxc2t98(VS.85).aspx)

- **Tutoriels**

- XPath : [http://www.zvon.org/xxl/XPathTutorial/General\\_fre/examples.html](http://www.zvon.org/xxl/XPathTutorial/General_fre/examples.html)
- XSL : <http://www.w3schools.com/xsl/>
- DOM : <http://www.w3schools.com/dom/>
- AJAX : <https://developer.mozilla.org/fr/AJAX> (sur Mozilla)
- JSON : <http://www.xul.fr/ajax-format-json.html>

- **Règles ergonomiques :**

<http://webilus.com/illustration/10-astuces-pour-ameliorer-les-contenus-de-vos-sites-internet>

- **Tendances et enjeux :**

[http://nauges.typepad.com/my\\_weblog/2008/09/firefox-chrome-safari-et-internet-explorer-quatre-strat%C3%A9gies.html](http://nauges.typepad.com/my_weblog/2008/09/firefox-chrome-safari-et-internet-explorer-quatre-strat%C3%A9gies.html)

# Quelques références



- **Ressources**

- Une liste de frameworks : <http://www.ajaxprojects.com/>
- Une liste de plein de choses : <http://ajaxpatterns.org/>
- En particulier, quelques outils de conception et de développement
  - ✦ **Frameworks**
    - [openAjax](#) (IBM) : Dojo
    - Ruby / Ruby on Rails (RoR)
    - Plugins Eclipse : [Rich Ajax Platform](#), Direct Web Remoting
    - PHP : [http://ajaxpatterns.org/PHP\\_Ajax\\_Frameworks](http://ajaxpatterns.org/PHP_Ajax_Frameworks)
  - ✦ **Librairies**
    - JQuery : <http://jquery.com/>
    - Google Web Toolkit (AJAXSLT...)