De l'Internet au Web des objets

Lionel Médini

M2IA – Université Lyon 1

Laboratoire d'InfoRmatique en Image et Systèmes d'information

LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon

http://liris.cnrs.fr/

Plan de la présentation

- Introduction et historique
  - Internet, IoT, TCP/IP, les systèmes distribués, le Web
- Les technologies du Web
  - Technologies de base (URI, HTML, HTTP)
  - Programmation côté serveur (servlets)
  - Programmation côté client (JavaScript, AJAX)
  - Aperçu des bibliothèques / frameworks disponibles (jQuery)
  - WebSockets
  - Services Web
  - Web sémantique
  - Services Web sémantiques
- Conclusion
  - Le WoT

Rappel : Internet

- Un réseau de réseaux interconnectés (d'où le nom)
- Un ensemble de matériels, logiciels et protocoles (notamment IP)
- Un ensemble de services
  - Application qui utilise un protocole et un numéro de port
  - e-mail, transfert de fichiers, connexion à distance, WWW…
- Une somme « d'inventions » qui s'accumulent
  - Mécanismes réseau de base (TCP/IP)
  - Nommage et adressage des ressources (DNS, URL)
  - Outils et protocoles spécialisés
  - Langages d'échange d'informations standardisés (HTML, XML…)

IoT Internet of Things

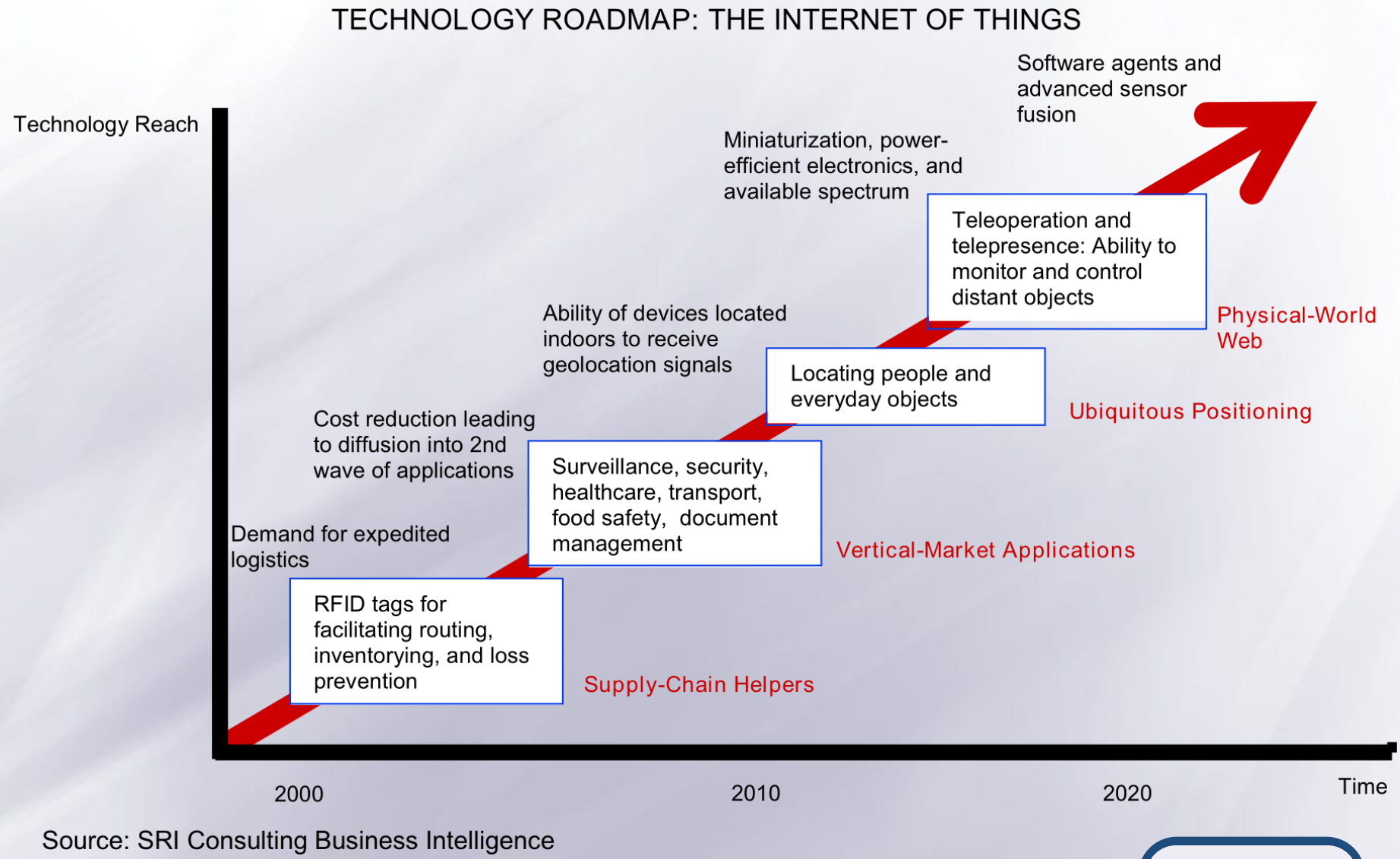
- **L'internet des choses**
  - Le réseau interconnectant les objets de tous les jours
  - Sous entendu : objet avec une pile TCP/IP ?
    - Pas forcément, si interactions limitées

(Sean Dodson, 2003) "IoT" can be expressed as the building of a global infrastructure for RFid tags. You could think of it as a wireless layer on top of the internet where millions of things from razor blades to euro banknotes to car tyres are constantly being tracked and accounted for. A network … is for computers to identify "any object anywhere in the world instantly". "Put a tag - a microchip with an antenna - on a can of Coke or a car axle, and suddenly a computer can 'see' it. Put tags on every can of Coke and every car axle, and suddenly the world changes.

IoT Internet of Things

- Notion de large échelle
  - Les définitions parlent d'intégrer « tous les objets »
  - Richesse : création d'interactions spontanées à partir de combinaisons d'objets non prévues
  - Dangers : contrôle sur le vivant et l'humain (tags RFID)
- Point de vue technologique
  - cf présentation 1
  - Limitation au niveau IP ? => IPv6

Feuille de route technologique



Source: SRI Consulting Business Intelligence

World Wide Web

- Principe original : accéder à des documents textuels
  - situés sur des machines accessibles par Internet
  - reliés entre eux par un mécanisme de lien « hypertexte »
- Actuellement : servir des ressources
  - De différentes natures : texte, image, son, vidéo, contenu applicatif…
  - Hypermédia
  - Interactives
  - Permettant à l'utilisateur d'accéder à un service donné : rechercher de l'information, acheter un objet, accéder à ses mails, consulter ses comptes en banque…
- ➔ Nombreuses évolutions techniques

Aspects techniques du Web

Les 3 mécanismes de base du Web

- **URL**
  - **Le Web permet d'accéder à un ensemble de ressources**
  - **Le mécanisme de localisation peut faire appel au protocole DNS**
- **HTML**
  - **Langage de description de « pages Web »**
    - *Texte, images et autres objets*
    - *Liens hypermédias entre les pages*
  - **Programmation déclarative**
- **HTTP**
  - **Protocole de niveau applicatif**
  - **Paradigme client-serveur**
  - **Protocole sans état (pas de « mémoire » des transactions précédentes)**

URI, URL et URN

URI : Uniform Resource Identifier

- **But : identifier de façon unique une ressource sur le web**
  - **En disant où elle se trouve**
    - *Donner son URL (Uniform Resource Locator)*
    - *Format : protocole ":" chemin "/" nom de fichier "/" requête*
    - `http://www.w3.org/2001/XMLSchema`
    - *Permet d'accéder réellement à la ressource (tant qu'elle existe)*
    - *Enregistrement des DNS auprès de l'entité concernée*

URI, URL et URN

URI : Uniform Resource Identifier

- **But : identifier de façon unique une ressource sur le web**
  - **En disant comment elle s'appelle**
    - *Donner son URN (Uniform Resource Name)*
    - *Format : "URN:" NID (namespace identifier) ":" NSS (namespace specific string)*
    - `URN:ISBN:0-395-36341-1`
    - *Choix plus « libre », et correspondant mieux à la définition d'un espace de noms*
    - *Enregistrement des NID à l'IANA (Internet Assigned Numbers Authority)*

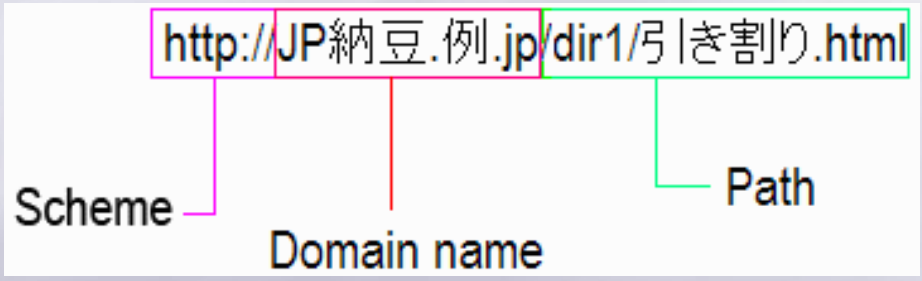
URI, URL et URN

- **URI : Uniform Resource Identifier**
  - But : identifier de façon unique une ressource sur le web
  - Syntaxe générique
    - « scheme » ":" autorité ":" chemin ":" requête ":" fragment
    - Avec le temps, on s'est mis à penser que « urn » peut aussi être un URI scheme

➔ **On n'utilise plus que la notion d'URI pour désigner les ressources**

URI, URL et URN

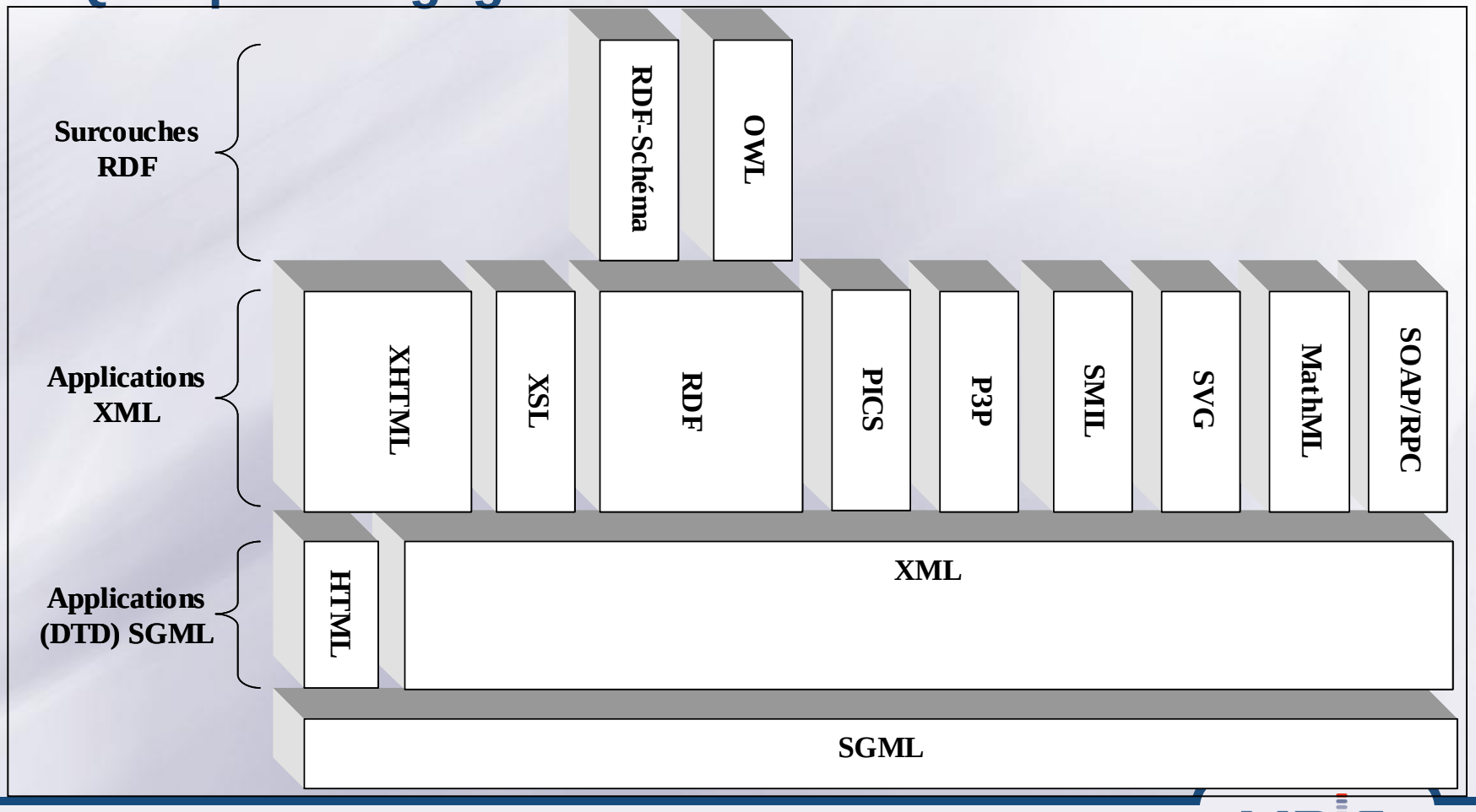
- On parle maintenant aussi d'IRI
  - Internationalized Resource Identifier
  - Une IRI est une URI acceptant des caractères Unicode
  - Exemple :



  - Remarque : nécessite un protocole similaire à DNS, mais gérant les noms de domaine Unicode : Internationalized Domain Name (IDN)
  - Référence : http://www.w3.org/International/articles/idn-and-iri/

Introduction aux langages à balises

Quelques langages dédiés au Web



HTML

- DTD SGML : ensemble (dé)fini d'éléments SGML
- Destiné à visualiser des hyperdocuments sur écran
- Interprété côté client par les navigateurs
- V. 1.0 : 1992 -> V. 4.01 (dernière…) : oct. 1997
- Souvent associé à d'autres langages
  - CSS : langage de feuilles de style
  - Langages de scripts : JavaScript, JScript, VBscript…
  - Inclusion d'objets définis dans d'autres langages : applets Java, contrôles ActiveX…
- Remarque : le langage D(H)TML n'existe pas

XHTML

- DTD XML : ensemble (dé)fini de balises XML
- But initial : réécrire HTML, mais en plus propre
- V. 1.0 : janv. 2000, révisée en 2002
  - Trois recommandations
    - XHTML frameset
    - XHTML transitionnel
    - XHTML strict
  - ⇒Séparation du fond et de la forme
  - ⇒Pris en charge par tous les navigateurs récents
  - ⇒Pris en charge par les outils de traitement XML

HTML

HTML 5 : http://www.w3.org/TR/html5/

- Historique
  - A l'origine (fin 2003) : initiative d'Opera pour harmoniser Xforms et HTML dans son navigateur
  - Par extension : un langage qui prendra en charge les différentes fonctionnalités des « Web applications »
  - Proposition de création d'un GdT du W3C rejetée (début 2004)
  - Finalement, création du Web Hypertext Application Technology WG (WHATWG) en 2006 sous la pression d'Apple, Mozilla et Opera
- Principes de développement
  - Don't Reinvent The Wheel
  - Pave The Cowpaths (!) : (X)HTML, CSS, DOM, EcmaScript…

HTML

- **HTML 5 : http://www.w3.org/TR/html5/**
  - **Détails des spécifications**
    - **Un langage de représentation abstrait**
      - *Description de documents et d'« applications Web »*
      - *API d'interaction avec les représentations mémoire de ces ressources (DOM)*
    - **Deux syntaxes concrètes**
      - *HTML5 : pour les contenus de type HTML, destinés à être traités par un navigateur*
      - *XHTML5 : pour les contenus de type XML, destinés à être parsés puis traités par une application particulière*
  - **Dernière version : Candidate Recommendation du 17 déc. 2012**
  - **Date initiale de sortie de la spec. : avant octobre 2010…**

Le protocole HTTP

Rappels

- HTTP : Hyper Text Transfer Protocol
- Dédié au Web (origine : CERN, 1990)
- RFC 2616 (HTTP 1.1)
- Fonctionne en mode client / serveur
- Port standard : 80
- Protocole sans état
  - Gestion légère des transactions
    - *aucune information conservée entre 2 connexions*
    - *permet au serveur HTTP de servir plus de clients*
  - Nécessite un mécanisme de gestion des **sessions**
    - *cookie, Id dans l'URL, champ caché de formulaire...*

Format des requêtes

- Commande HTTP
  - Méthode : `GET`, `POST`, `HEAD`, `PUT`, `DELETE`, `TRACE`, `OPTIONS`, `CONNECT`
  - URL à partir de la racine du serveur
  - Version HTTP
- En-têtes (ensemble de lignes)
  - Nom de l'en-tête
  - Deux-points
  - Valeur de l'en-tête
- Une ligne vide
- Contenu (éventuel)
  - Passage de paramètres à traiter par le serveur

La méthode GET

- Méthode standard de requête d'un document
  - récupérer un fichier, une image...
  - activer un programme en lui transmettant des données
- Le corps (contenu) de la requête est toujours vide
- Ajout de paramètres après le nom de la ressource
  - Transmission des données dans l'URL après un ?
  - Les champs sont séparés par un &

```
GET /cgi-bin/prog.cgi?email=toto@site.fr&pass=toto&s=login HTTP/1.1
```

- Remarques
  - Toutes les données sont transmises en clair et visibles dans l'URL
  - L'URL a une taille limitée (4Ko)

La méthode POST

- **Transmission des données dans le corps de la requête**

```
POST /cgi-bin/prog.cgi HTTP/1.1
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: localhost
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 36

email=toto@site.fr&pass=toto&s=login
```

- **Les données sont également transmises en clair**

Format des réponses

- Type de la réponse
  - Version HTTP
  - Code de la réponse
  - Description du code
- En-têtes (ensemble de lignes)
  - Nom de l'en-tête
  - Deux-points
  - Valeur de l'en-tête
- Une ligne vide
- Contenu éventuel
  - Ressource encodée en fonction du type MIME spécifié

Codes de réponses

- Les codes de réponse
  - Indiquent le résultat de la requête : succès ou échec
  - En cas d'échec, le contenu de la réponse en décrit la raison fichier non présent, problème de droit
- Classes de codes
  - 100-199 : information
  - 200-299 : succès
  - 300-399 : redirection
  - 400-499 : échec dû au client
  - 500-599 : échec dû au serveur
- Plus d'infos

  http://www.codeshttp.com/

  http://httpstatusdogs.com/

```
HTTP/1.1 200 OK
HTTP/1.1 304 Not Modified
HTTP/1.1 403 Forbidden
HTTP/1.1 404 Not Found
HTTP/1.1 500 Internal Server error
```

Une transaction typique (1)

- **Requête du client : client → serveur**

1. demande du document `test.html`

```
GET /~lmedini/MIF13/test.html HTTP/1.1
```

2. envoi des informations d'en-tête : informer le serveur
   - configuration
   - documents acceptés

```
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: www710.univ-lyon1.fr
Accept: image/gif, image/jpeg
```

3. envoi d'une ligne vide (fin de l'en-tête)
4. envoi du contenu (vide dans cet exemple)

Une transaction typique (2)

- **Réponse du serveur : serveur → client**
  5. code indiquant l'état de la requête

```
HTTP/1.1 200 OK
```

  6. envoi des informations d'en-tête : informer le client
     - configuration du serveur
     - document demandé

```
Date: Tue, 30 Sep 2008 06:11:28 GMT
Server: Apache/1.3.34 (Debian) PHP/5.2.1
Last-Modified: Tue, 30 Sep 2008 06:11:14 GMT
ETag: "600593b3-61-48e1c302"
Accept-Ranges: bytes
Content-Length: 97
Content-Type: text/html; charset=iso-8859-1
```

  3. envoi d'une ligne vide (fin de l'en-tête)
  4. envoi du contenu si la requête a réussi

Encodage des ressources

- Position du problème
  - Un serveur Web peut servir différents types de ressources : texte, pages Web, images, documents, fichiers exécutables…
  - Chaque type de ressource est codé de façon différente
  - Un client doit connaître le type de ressource pour pouvoir la traiter :
    visualisation dans un navigateur, utilisation d'un plugin, application externe
- Solution (HTTP et Internet en général)
  - MIME : Multi-purpose Internet Mail Extensions
  - Prise en charge de MIME dans HTTP : depuis V1.0

Encodage des ressources

- **Types MIME : composition**
  - Type général : `text`, `image`, `audio`, `video`, `application`...
  - Sous-type : dépend du type général
  - Exemples : `image/gif`, `image/jpeg`, `application/pdf`, `application/rtf`, `text/plain`, `text/html`
  - En perpétuelle évolution
- **Types MIME : utilisation**
  - Le serveur positionne le header `Content-type`
    `Content-Type: text/html; charset=UTF-8`
  - Le client associe chaque type MIME à un type de prise en charge

Encodage des caractères

- L'encodage des caractères utilisés dans une ressource
  - est considéré comme une sous-partie de l'encodage des ressources
    - lié au type MIME de la ressource
    - lié à la langue de la ressource
  - est indiqué dans les headers HTTP de la réponse

```
Content-Language: en, fr
Content-Type: text/html; charset=ISO-8859-1
```

Encodage des paramètres

- **Format : URL-encoded**
  - Permet de coder les données dans l'URL (méthode GET)
  - Encodage fait par le client
  - Syntaxe des URL (RFC 2396)
    - début des paramètres : `?`
    - séparation entre le nom du champ et sa valeur : `=`
    - séparateur de champ : `&`
    - espaces dans la valeur d'un champ : `+`
    - caractères réservés : `; / ? : @ & = + $ ,`
    - caractères non-alphanumériques remplacés par `%xx` (`xx` = code ASCII du caractère en hexadécimal)
  - Exemple

    ```
    nom_champ1=valeur1&nom_champ2=valeur2&...
    ```

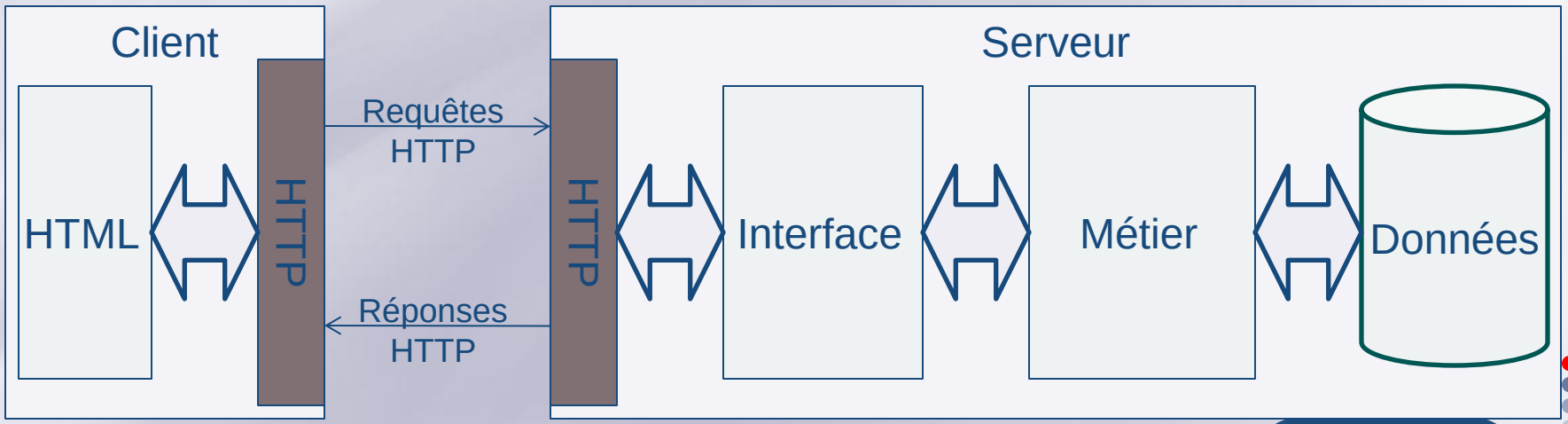
  - Cas des champs à valeurs multiples (listes de sélection)

    ```
    nom_liste=valeur1&nom_liste=valeur2&...
    ```

Programmation côté serveur

Application Web

- Application dont l'interface est visible dans un navigateur
  - Nécessairement des programmes côté serveur
  - Parfois une partie côté client
- Dépendent de l'infrastructure web choisie
- Exemple



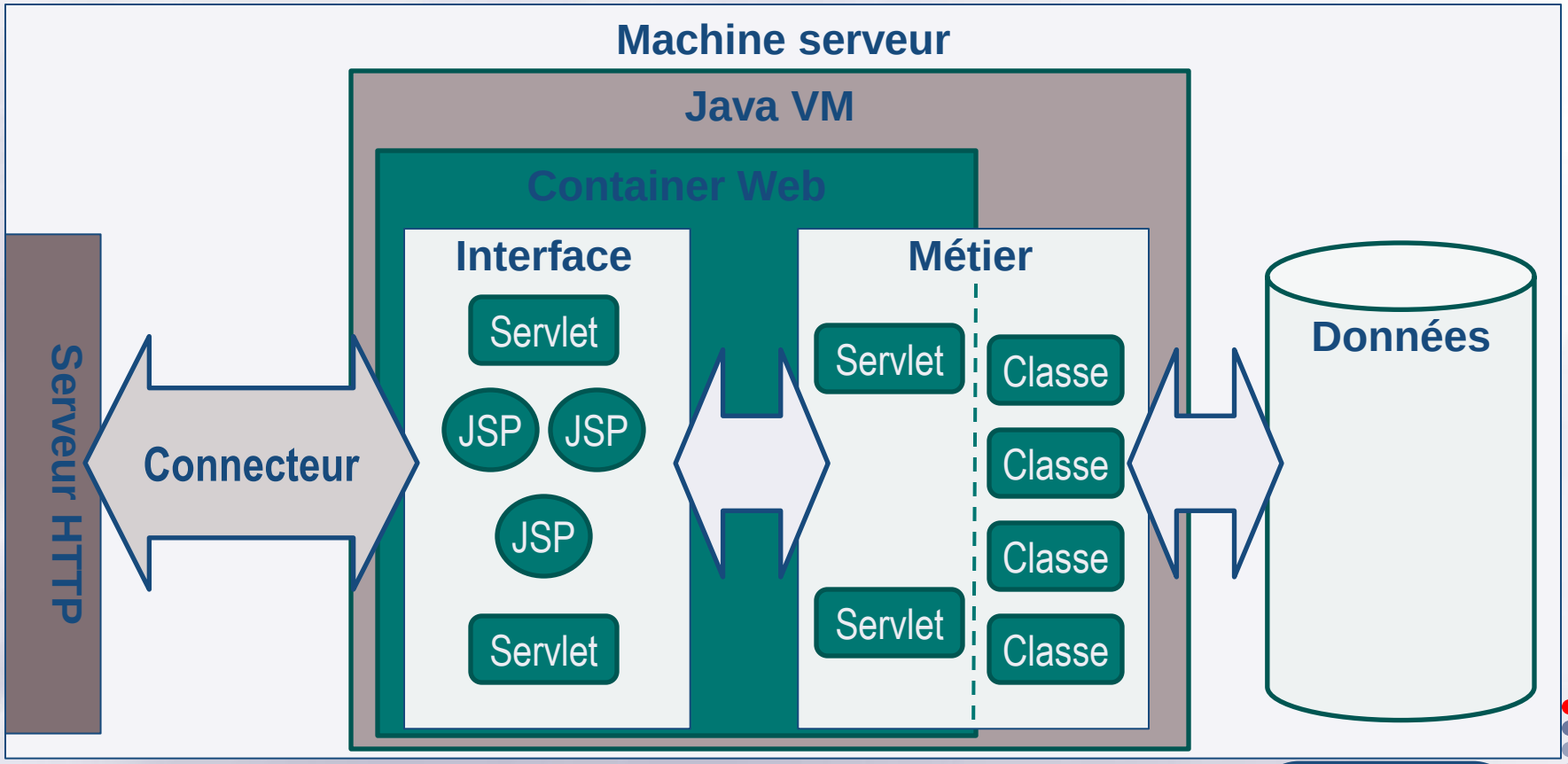
Applications Web

Exemples de technologies

- Php
  - Langage interprété
  - Type de programmation : scripts / fonctions / objets
  - Moteur : interpréteur existant sur la quasi-totalité des serveurs
- Java
  - Bytecode
  - Type de programmation : classes (servlets), scripts (JSP)…
  - Moteur : container de servlets Jakarta (+ Apache = Tomcat)
- Microsoft™ .Net Framework
  - Ensemble de technologies de développement
  - Type de programmation : dépend du langage VB, C#, J#, ASP…
  - Moteur : framework sur serveur IIS
- Python
  - Langage interprété
  - Type de programmation : scripts python, scriptlets, DTML…
  - Moteur : serveur d'applications Zope, Plone
- …

Programmation côté serveur en Java

Principes de la programmation côté serveur en Java



Programmation côté client

- **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

- Caractéristiques
  - Interprété
  - Fonctionnel
  - Orienté prototype
    - « object-based » plutôt qu'« object-oriented »
    - Typage dynamique : types associés aux instances et non aux classes
  - Événementiel
    - Mécanismes de « callback »
    - Pattern observer : eventListener
- Fonctionnalités
  - Reflection
  - E4X : ECMAScript for XML (ECMA-357)
  - JSON
  - …

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://en.wikipedia.org/wiki/Prototype-based_programming

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.epershand.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

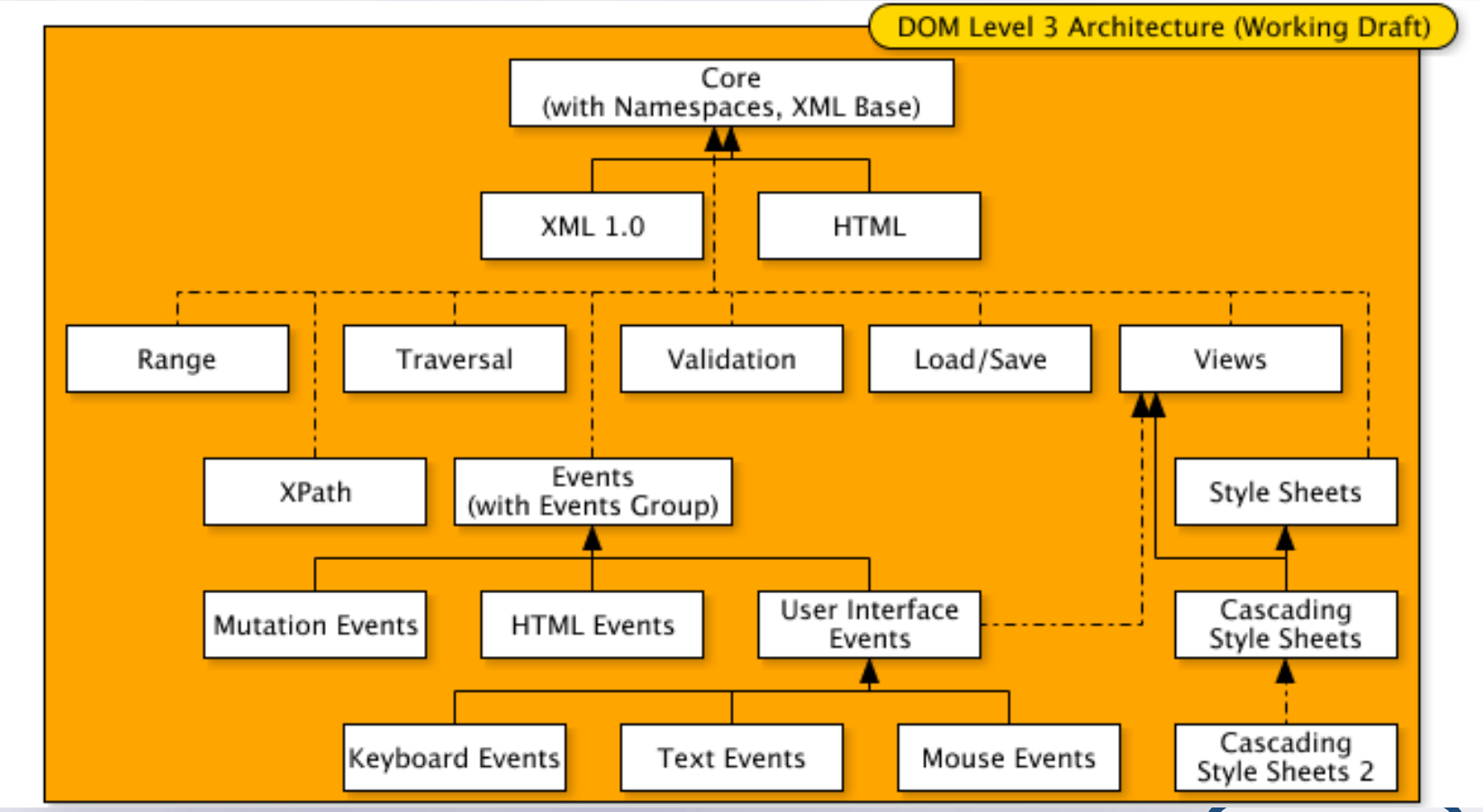
L'API DOM (Document Object Model)

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)

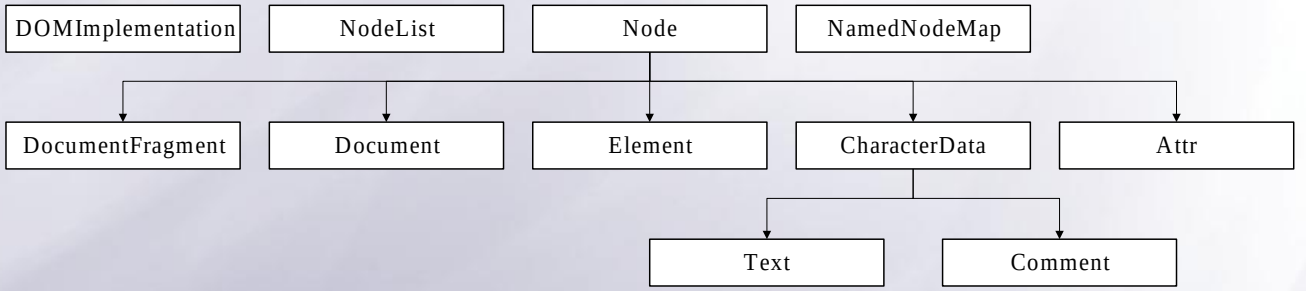
Fonctionnalités



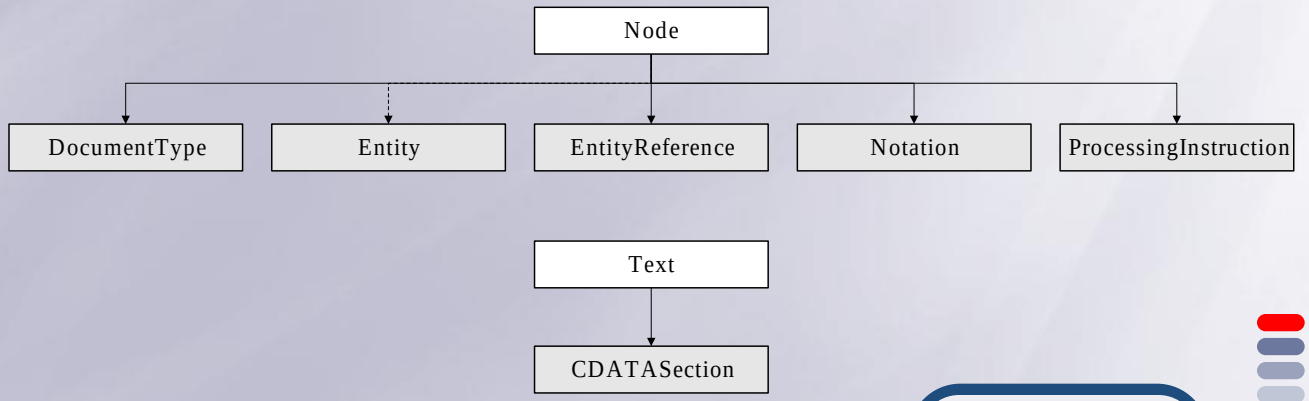
L'API DOM (Document Object Model)

Interfaces

- **DOM Core :**

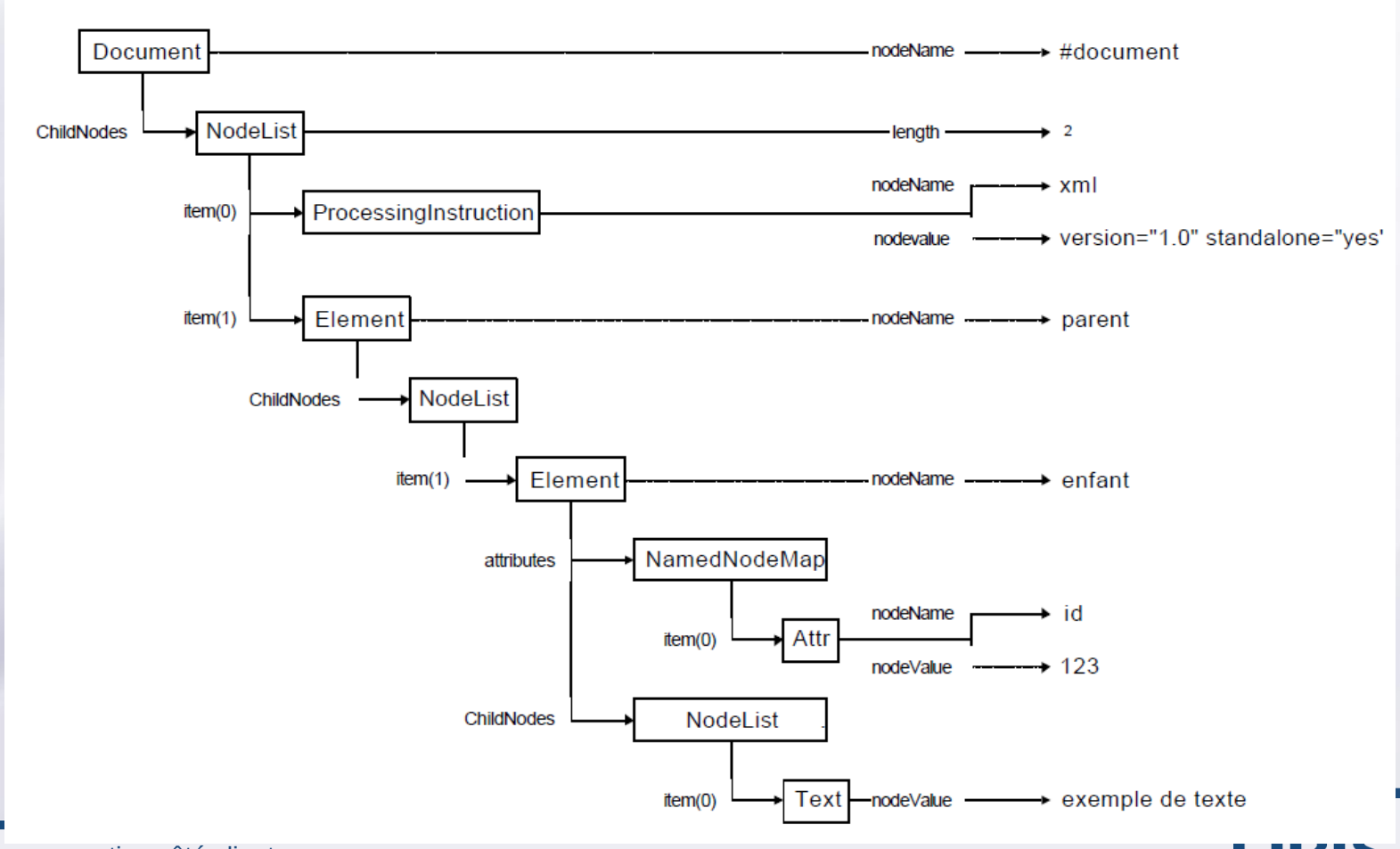


- **DOM XML :**



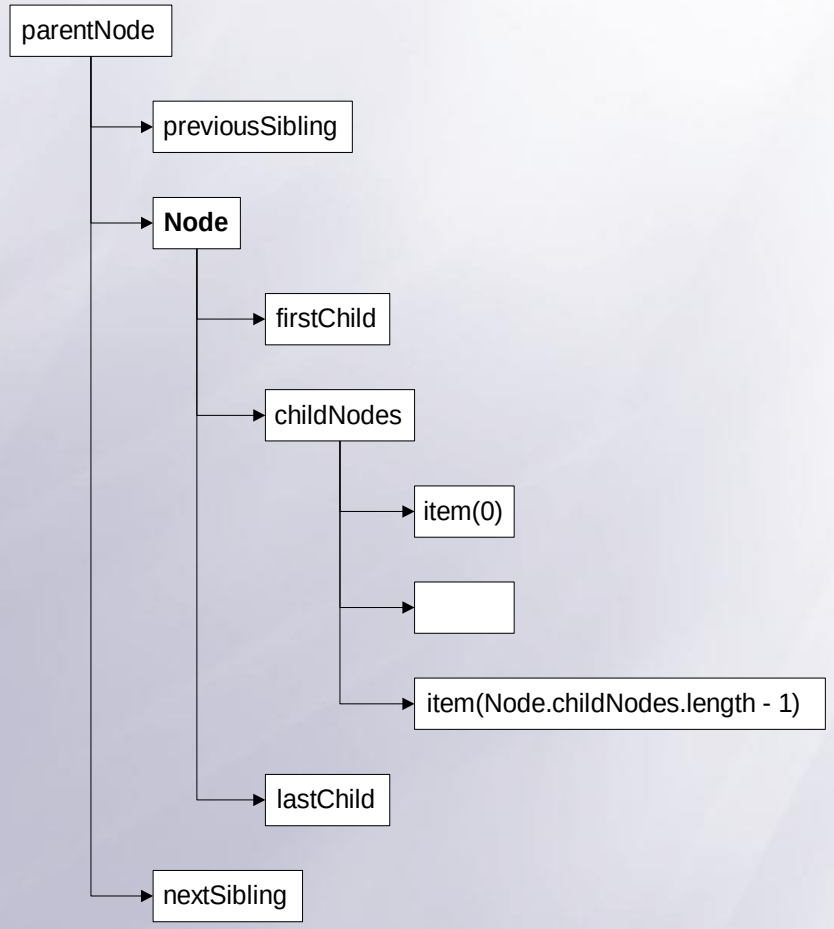
L'API DOM (Document Object Model)

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

- **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

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)

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/

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 XMLHTTP (IE)

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…

AJAX

Fonctionnement

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

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, …]

AJAX

Implémentation de la logique applicative

- Standardisation de la communication avec les langages de programmation côté serveur : JSON
  - Exemple de fichier au format JSON :

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

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

- Equivalence en XML :

```
<?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
```

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
```

AJAX - sécurité

Risques

- Déporter de la logique applicative sur le client présente des risques
  - Falsification du client
- L'envoi d'une requête asynchrone XHR à un autre serveur que celui ayant délivré le script est impossible (en principe)
  - Same Origin Policy

AJAX - sécurité

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)
- **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

AJAX - conception

- **Quelques règles pour développer une application Web riche**
  - **Outils de développement**
    - **Utilisez les ressources à votre disposition**
      - *Choisissez une bibliothèque aussi standard que possible*
    - **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**

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/

Bibliothèques / frameworks JS

- APIs d'applications Web externes
  - Principe : interfacer son application avec une plus connue
  - Nombreux exemples dans le Web 2.0 :
    Google (Calendar, Mail, Wave…), FaceBook, YouTube, Ebay…
  - ➔ Un moyen rapide d'améliorer vos applications
  - ➔ Permet d'attirer des utilisateurs
- Frameworks AJAX
  - Programmation dans un autre langage
  - Génération du code JavaScript
  - Mécanismes de communication standard entre client et serveur
- Références
  - http://softwareas.com/ajax-patterns/
  - Liste de près de 10000 API disponibles
    http://www.programmableweb.com/apis/directory

Bibliothèques Web

Bibliothèques CSS

- Exemples de feuilles de style CSS open source : http://www.oswd.org/

Bibliothèques AJAX

- Bibliothèques « directes »
  - Bibliothèques de fonctions JavaScript pour faciliter le codage
    - *Peu structurées, ne sont utilisables que pour de petites applications*
  - Éventuellement, des outils côté serveur facilitant la génération de pages liées à ces bibliothèques
    - *Nécessitent d'avoir une vue claire de l'application*
  - Exemples
    - *Génériques : jQuery, SAJAX, DHTMLX, Fleejix.js, JsHTTPRequest, My Library*
    - *Java : JSP Tags Library*
    - *PHP : XAJAX, PhpLiveX*
    - *.Net : DotNetRemoting Rich Web Client SDK for ASP.NET , ASP.Net AJAX*
    - *…*

Bibliothèques Web directes

- **jQuery**
  - **Présentation**
    - **Bibliothèque de fonctions d'aide à la génération d'applications Web**
      - *Navigation dans un document et sélection d'éléments (X)HTML*
      - *Gestion d'événements*
      - *AJAX*
      - *Animations…*
    - **Utilisation très répandue**
    - **Existence de plugins développés par la communauté**
    - **Remarque : 2 versions**
      - *Compressée (production) / Lisible (développement)*
  - **Site Web**

    http://jquery.com/
  - **Documentation**

    http://docs.jquery.com/

jQuery

L'objet jQuery

- **Équivalent : $**
- **Fonction membre de l'objet window**
- **Plusieurs utilisations**
  - **jQuery(selector [, context])**
    - *Renvoie tous les éléments DOM*
      - Correspondant au sélecteur selector
      - À partir de l'élément DOM donné en context
  - **jQuery(html [, ownerDocument])**
    - *Renvoie objet jQuery correspondant à un ou plusieurs élément(s) DOM*
      - Rajouté(s) au document ownerDocument
      - Correspondant à la chaîne de caractères html
  - **jQuery(callback)**
    - *Appelle une fonction de callback quand le DOM est chargé*
    - *Équivalent : jQuery(document).ready()*

jQuery

Sélecteurs

- Tous les sélecteurs CSS (versions 1 à 3)
- Des attributs et fonctions spécifiques
  - :checked, :empty, :even, :header…
  - :eq(), :lt(), :not(), :nth-child()…
- Des notations particulières
  - Sélecteurs multiples : ("selector1", "selector2 ", "selector3 ")
  - Next adjacent selector : ("previous + next ")
  - Next sibling selector : ("previous ~ sibling ")
- Référence : http://api.jquery.com/category/selectors/

jQuery

Objets jQuery

- Toutes les méthodes jQuery retournent un ou plusieurs (tableau) objets jQuery
- Chaque objet jQuery possède l'ensemble des méthodes définies par l'API jQuery

➔ On peut donc chaîner les méthodes entre elles :

```
$('h1#titre').html($('title').html()).before('Voici le titre :').click(mafonction);
```

- Le chaînage s'appliquera pour chacun des objets retournés par chaque fonction de la chaîne
- Exemples : http://www.siteduzero.com/tutoriel-3-160891-jquery-ecrivez-moins-pour-faire-plus.html?all=1

jQuery

Gestion des événements

- Fourniture de fonctions pour l'ajout d'EventHandlers d'événements standards…
  - click(), dblclick(), load()
- …Ou définis par la bibliothèque
  - ready()
- Permet d'attacher une callback à un événement quelconque
  - bind(), unbind()
- Référence : http://api.jquery.com/category/events/
- Remarque : l'objet Event est lui aussi surchargé par un objet jQuery spécifique http://api.jquery.com/category/events/event-object/

jQuery

Requêtes asynchrones

- AJAX
  - $.ajax({
    url: "test.html",
    context: document.body,
    success: function(){
      $(this).addClass("done");
    }
  });
- JSON
- Générale

jQuery

Requêtes asynchrones

- AJAX
- JSON
  - jQuery.getJSON(url [, data] [, success(data, textStatus, jqXHR)])
  - Équivalent à :
  - ```
    $.ajax({
      url: "test.html",
      datatype: 'json',
      context: document.body,
      success: success
    });
    ```
- Générale
    ```

jQuery

Requêtes asynchrones

- AJAX
- JSON
- Générale
  - jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])
  - Équivalent à :
  - ```
    $.ajax({
      url: "test.html",
      datatype: datatype,
      context: document.body,
      success: success
    });
    ```
- Référence : http://api.jquery.com/category/ajax/
    ```

jQuery UI

- Extension de jQuery
  - Bibliothèque d'éléments d'interface (thèmes, widgets, primitives d'interaction)
  - Permet de rajouter facilement des interactions complexes
  - Permet de rendre une application Web plus dynamique
  - Exemple
    - Drag'n drop : http://jqueryui.com/demos/draggable/
  - Utilisation
    1. Identifier les éléments dont on a besoin
    2. Construire et télécharger sa bibliothèque personnalisée
    3. L'utiliser dans son application
- Site Web
  http://jqueryui.com/

WebSockets

Position du problème

- **HTTP : protocole client-serveur**
- **Le serveur ne peut que répondre à une requête d'un client**
- **Manquent des technologies de**
  - **Push serveur**
    - *Requêtes répétées du client*
      - Gaspillage de bande passante
    - *Ne pas fermer la connexion persistante (COMET)*
      - Réponses "Chunked", Pushlet, Long polling, Flash XML sockets...
  - **Communication bidirectionnelle (P2P) entre clients**

La spécification WebSocket

- But : permettre la communication bidirectionnelle entre un client et un serveur
- Contenu de la spec
  - Un protocole de communication : RFC6455
    - Au-dessus de TCP
    - Headers en "HTTP-like"
    - "Compatible avec HTTP" : un même port peut être utilisé pour les deux protocoles
  - Une API en WebIDL : WebSocket API
    - Modèle d'implémentation pour les serveurs et les navigateurs
    - Description des échanges entre les deux parties

Techniquement

Handshake en HTTP (1.1)

- Ouverture d'un WebSocket à l'initiative du client

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

- Clé : chaîne de caractères aléatoire
- Options : sous-protocole, version...

Techniquement

Handshake en HTTP (1.1)

- Acceptation du serveur

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

- Statut : changement de protocole
- Clé : SHA de la concaténation de la chaîne envoyée + un Globally Unique Identifier (RFC 4122)
- Options : choix d'un sous-protocole...

La spécification WebSocket

Utilisation

- Échange de données bidirectionnel en TCP
  - Chaque côté peut envoyer des données quand bon lui semble
  - Pas de réponse à un envoi de données
  - Les ensembles de données envoyées simultanément sont appelées "**messages**"
- côté client :
  - Emission : méthode **send()** de l'interface Message
  - Réception : Abonnement à un événement **onMessage**
- côté serveur (nécessite la prise en charge des WebSocket) :
  - Réception : même principe
  - Emission : idem...

Utilisation côté client

```
var ws;
$(document).ready(
  function() {
    ws = new WebSocket("ws://localhost:8080/../WebSocketChat");
    ws.onopen = function(event) { }
    ws.onmessage = function(event) {
      var $textarea = $('#messages');
      $textarea.val($textarea.val() + event.data + "\n");
    }
    ws.onclose = function(event) { }
  });
function sendMessage() {
  var message = $('#username').val() + ":" +
$('#message').val();
  ws.send(message);
  $('#message').val('');
}
```

- Source : http://java.dzone.com/articles/creating-websocket-chat

Schémas d'échanges de données

- Types d'opérations
  - CRUD
  - Publish/Subscribe
- Méthodes HTTP
  - GET, PUT, POST, DELETE
  - Observe : PUT
  - Notify
    - Push serveur
    - Communication orientée-messages (WebSockets)

Autres protocoles du Web

- **Constrained Application Protocol (CoAP)**
  - **Protocole applicatif conçu pour l'Internet des objets**
  - **Destiné aux objets contraints**
    - **Microcontrôleurs, capteurs…**
  - **Pas de connexion persistante**
    - **Sur UDP**
    - **Fiabilisation (ack, retransmissions)**
  - **Schémas de communication**
    - **Couche HTTP-like légère (RESTful)**
      - *Mêmes méthodes que HTTP*
    - **Communication orientée-messages**
      - *Asynchrone*
      - *Extension OBSERVE*
  - **Spécification : RFC 7252**

Autres protocoles du Web

HTTP-2

- Optimisation de HTTP/1.1
  - Compression des headers
  - Multiplexage / pipelining des transactions
- Compatibilité ascendante
  - Méthodes, codes, headers…
- Nouveautés
  - Publish-subscribe
  - Push serveur
  - Streaming
- Conçu pour être compatible avec le Web des Objets
  - Mais complexe à utiliser
- Spécification : RFC 7540

Outils

- Côté client
  - Protocole et API JS intégrés aux navigateurs récents
  - Bibliothèques : plugin jQuery, jWebSocket
- Côté serveur
  - Nécessite un serveur qui prenne en charge le protocole et l'API
  - Exemples en Java :
    - jWebSocket (serveur standalone ou bundle pour Tomcat)
    - Jetty
    - GlassFish
  - Autres langages :
    - Socket.IO (node.js)
    - EventMachine (Ruby)
    - Twisted WebSocket Server (Python)

Références

- Le protocole RFC6455
- L'API WebSocket
  - Candidate Recommendation (septembre 2012)
  - Editor's Draft
- Documentation
  - Une page Wikipedia générale sur le push serveur
  - Une page Wikipedia générale sur les technologies Comet
  - La page Wikipedia sur les WebSockets