Développement WEB
Introduction

HTML, CSS, XML,
PHP, JavaScript, AJAX

jean-pierre.gerval@isen-ouest.yncrea.fr

Généralités

- Le **World Wide Web**, littéralement la «toile (d'araignée) mondiale», communément appelé le **Web**, parfois la **Toile** ou le **WWW**, est un système hypertexte public fonctionnant sur Internet et qui permet de consulter, avec un navigateur (*browser*), des pages mises en ligne dans des sites.
- Trois technologies sont à la base du World Wide Web :
  - les **URL** (*Uniform Resource Locator*), ou « adresse web », pour pouvoir identifier toute ressource dans un hyperlien
  - le langage **HTML** (*Hypertext Markup Language*), pour écrire des pages contenant des hyperliens
  - le protocole de communication client/serveur **HTTP** (*Hypertext Transfer Protocole*), utilisé entre les navigateurs et les serveurs web. HTTPS est la variante sécurisée de ce protocole.

Historique (1/2)

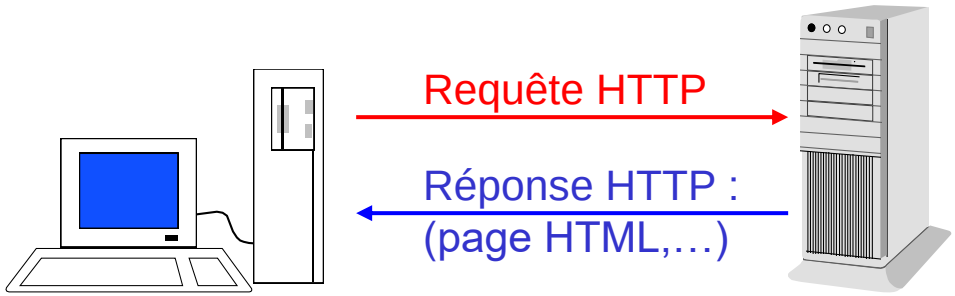
- Le **World Wide Web**, a été inventé par Tim Berners-Lee, informaticien au CERN à Genève, en **1990**.
- **1993** Le CERN met les logiciels du WWW dans le domaine public. En décembre 1993, il existe 623 sites web dans le monde.
- **1994** Création du **World Wide Web Consortium** (**W3C)**, organisme de normalisation qui fixe les standards du Web. Son président est toujours Tim Berners-Lee.
- **1995** Naissance des langages *Java*, *JavaScript*, *PHP*, des navigateurs *Netscape* et *Internet Explorer*, du serveur *Apache*.
- **1996** Création par *Microsoft*, de la technologie *ASP* (*Active Server Page*), concurrente de *PHP ;* Création par *Macromedia*, de la technologie *Flash*.

Historique (2/2)

- **1998 à 2000** Publication par le W3C des grands standards de base : CSS level 2, DOM level 1, ECMAScript ed. 3, HTML4, XHTML1.0, XML1.0, XSLT1.0.
- **2004** Naissance du navigateur *Mozilla Firefox*.
- **2005** Apparition du concept AJAX.
- **2008** Premier « *working draft* » de HTML5, proposition de standard en 2012, normalisé en 2014. HTML5 remplace non seulement HTML4, mais aussi XHTML1 et une partie de DOM 2 (Document Object Model). HTML 5 introduit une API complète pour le développement d'applications Web.
- **2011** le WHATWG (groupe de travail ± lié au W3C) décide de faire évoluer HTML en permanence (« *HTML living standard* »).
- **septembre 2013 :** 1 milliard de sites web dans le monde (source : wikipedia).

Modèle de base client/serveur Web

Modèle d'architecture *REST* (*Representational State Transfer)*



Navigateurs les plus répandus :

Internet Explorer, Google Chrome, Mozilla Firefox.

Serveur HTTP les plus répandus :

Apache (60%), Nginx (22%), Microsoft IIS (17%)

Requête HTTP :

Une URL (adresse Web) suivie éventuellement de paramètres : seulement sous forme GET ou POST en HTTP/1.0 (+PUT, DELETE en en HTTP/1.1)

Réponse HTTP : une page Web, généralement une page HTML, mais aussi XML...

Evolutions du modèle (1/3)

- Dans les premières années du Web, la réponse du serveur HTTP était une page HTML statique.
- **Après 1995**, apparaît le concept *DHTML* (HTML dynamique) :
  - côté serveur, la page HTML est crée dynamiquement à l'aide de langages de programmation : PHP, Perl, *servlets* Java, ASP… et de lecture dans des bases de données (notamment MySQL). Toutefois chaque mise à jour demande le rechargement complet d'une nouvelle page HTML.
  - côté client, la page devient interactive grâce au langage JavaScript (associé au DOM), à des *applets* Java, ou des technologies comme Flash (Macromedia).

Evolutions du modèle (2/3)

- **Après 2000**, les navigateurs commencent à accepter les préconisations du W3C, notamment les CSS (*Cascading Style Sheet*) et XHTML (*eXtensible Hypertext Markup Language*) dont un des buts est de séparer le contenu du document et sa présentation (couleurs, fontes, positionnement,…). La visualisation de pages XML devient aussi possible grâce à différentes techniques (notamment les CSS ou XSLT).

- **En 2005**, apparaît le concept AJAX (*Asynchronous JavaScript and XML*), qui a pour but de pouvoir mettre à jour une page en allant chercher des informations sur le serveur, mais sans recharger la totalité de la page HTML. AJAX n'est que l'utilisation conjointe de technologies préexistantes (JavaScript muni de l'objet *XMLHttpRequest*, XML).

Evolutions du modèle (3/3)

- **Vers 2006** apparaît le concept Comet, ou Ajax Inverse (ou *Server push*, *Ajax push*, *HTTP streaming*, *HTTP long polling...*). Dans celui-ci c'est le serveur Web qui est à l'initiative de la communication, et qui envoie des messages asynchrones vers le client.
- **A partir de 2008**, le développement des spécifications de HTML5 révolutionne l'architecture simple (REST) qui caractérisait les premières époques du Web. Les spécifications de HTML5 se sont assez vite divisées en plusieurs spécifications annexes séparées : *Web Workers*, *Web Storage*, *Web Databases*, *local Filesystem*, *Web Socket*,… Ces différents apports permettent maintenant de développer des applications Web « côté client » (« *thin client* ») aussi riches que des applications « non Web » (« *thick* ou *fat client* »).

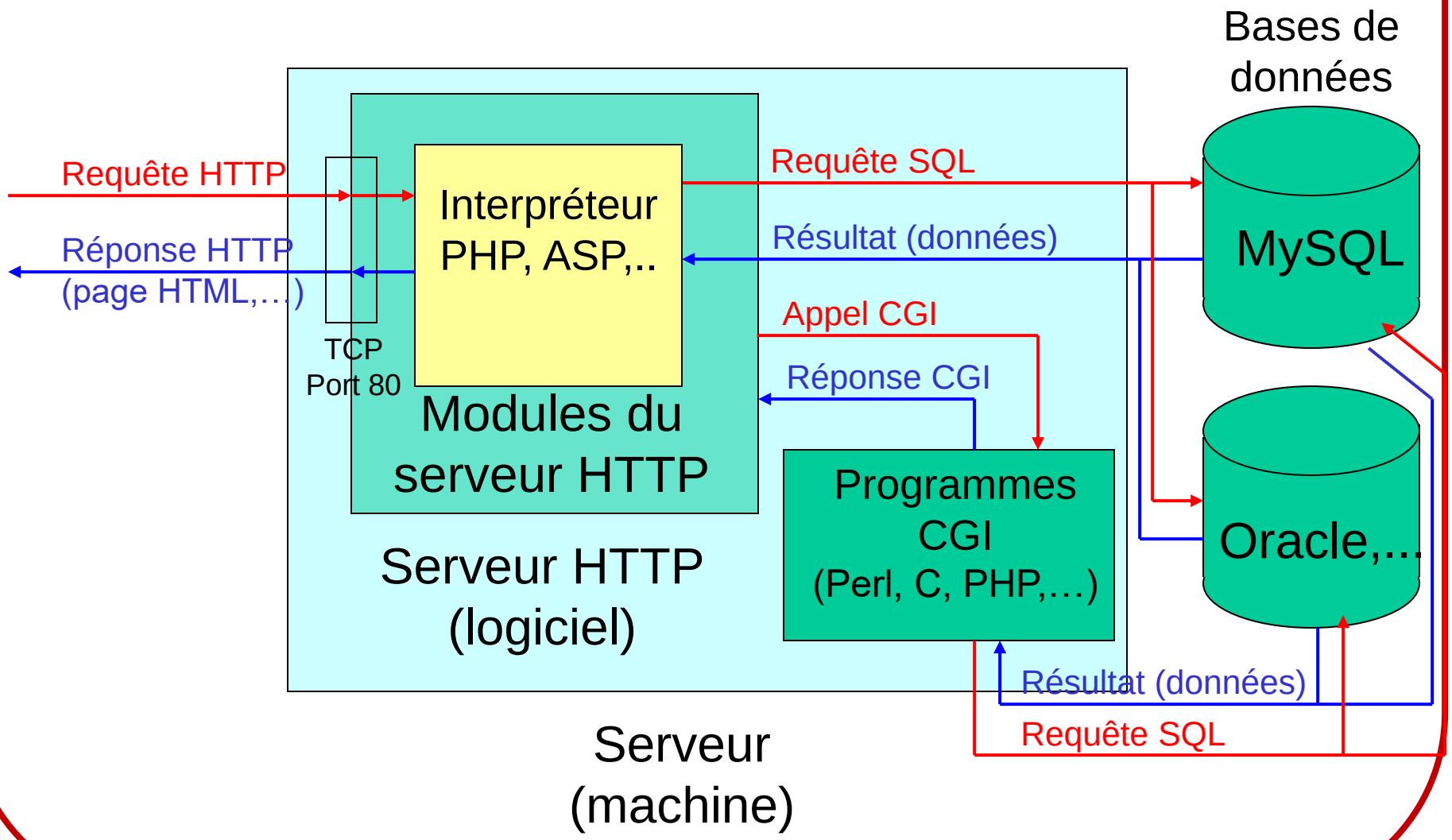
  On les nomme parfois « *rich Internet applications* » (RIA).

Les techniques côté serveur (1/4)

On peut globalement séparer deux techniques :

- soit la requête est traitée par un programme indépendant du serveur HTTP :
  - les programmes dits « CGI » ou « fastCGI » dialoguent avec le serveur HTTP selon le protocole normalisé CGI (*Common Gateway Interface*). Ces programmes sont souvent écrits en Perl, mais il n'y a aucun langage imposé. C'était anciennement la seule technique qui existait.
  - les moteurs de *servlets* Java assurent l'interface avec le serveur HTTP dans le cas des JSP (*Java Server Pages*).
- soit la requête est traitée par le serveur HTTP lui-même (le module nécessaire doit être installé en tant qu'élément du serveur) :
  - les programmes PHP (l'interpréteur est installé en tant que module Apache) fonctionnent généralement de cette façon avec le serveur Apache.
  - les programmes ASP fonctionnent de cette façon avec le serveur IIS.

Les techniques côté serveur (2/4)



Les techniques côté serveur (3/4)

- Quelle que soit la technique, il s'agit toujours de générer dynamiquement une page web (généralement une page HTML).

  Ceci est nécessaire :

  – pour présenter des données à jour dans la page, typiquement par lecture dans une base de données (ex: MySQL),
  – pour présenter une page tenant compte d'informations envoyées par l'utilisateur (ex: saisie dans un formulaire),
  – pour construire dynamiquement des pages basées sur des modèles (*templates*). Le contenu d'une page particulière est inséré dynamiquement au sein d'une page servant de modèle;
  – ou, tout simplement, pour présenter une page dont le contenu dépend de la date ou de l'heure.

Les techniques côté serveur (4/4)

- Les techniques « côté serveur » les plus populaires dans le développement web actuel sont :
  - **PHP**, souvent associé à une base de données MySQL et à un serveur HTTP Apache. Ensemble de technologies « *open source* », appelé LAMP (Linux Apache MySQL PHP). PHP est le langage utilisé par 82% des serveurs HTTP, selon *w3techs.com* (octobre 2014).
  - **ASP** (*Active Server Page*) qui est une technologie Microsoft, souvent associée au langage VBScript, à une base de donnée SQLServer, au serveur HTTP IIS (*Internet Information System*) et au *framework* .NET, toutes technologies « propriétaire » Microsoft.
  - **JSP** (*Java Server Pages*) qui est une technologie d'origine *Sun Microsytems*, associée au langage Java et qui requière un *moteur de servlet :* Tomcat (Apache), GlassFish (Oracle)…

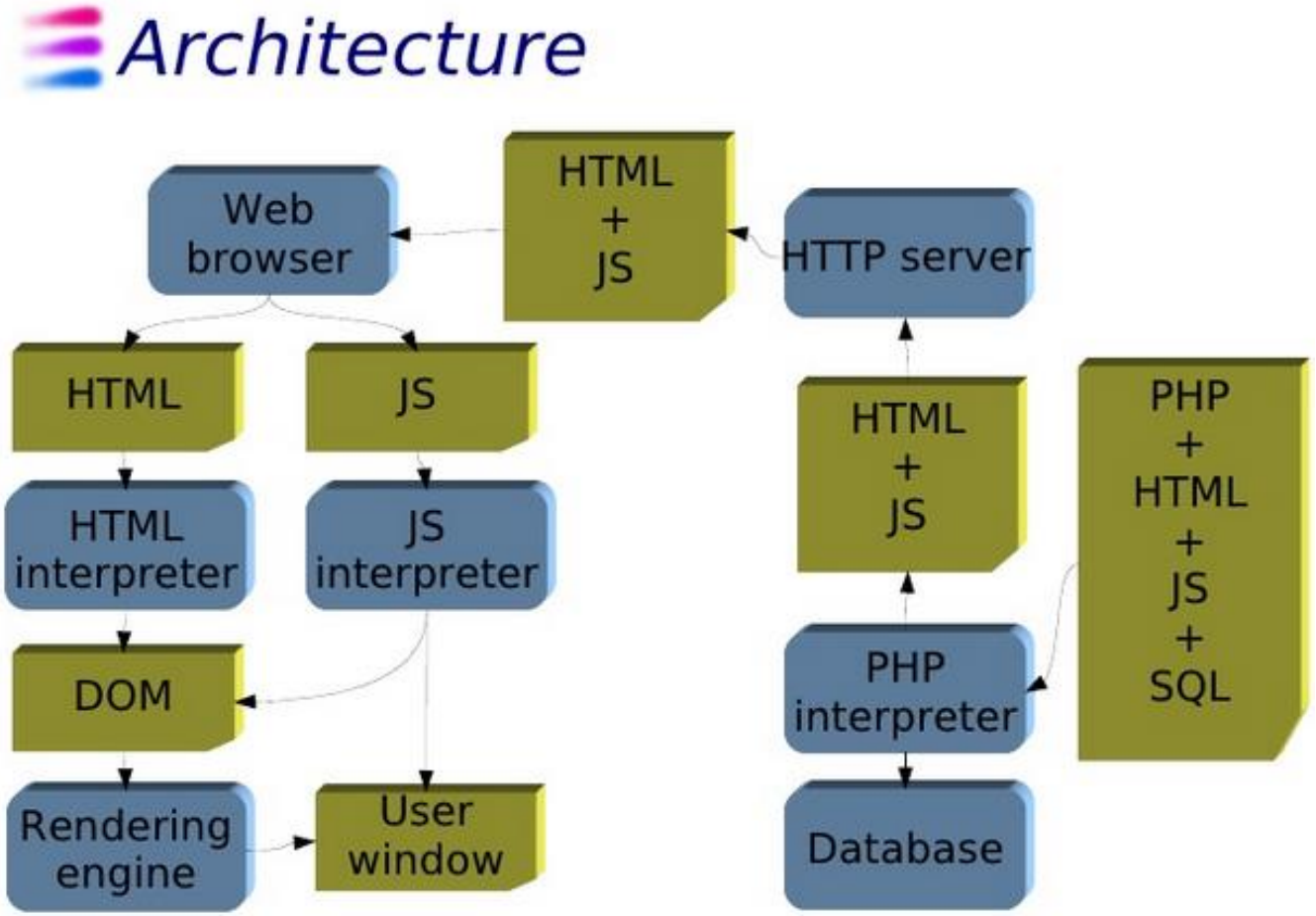
Les techniques côté client (1/2)

- Il existe essentiellement 3 techniques pour rendre interactive une page web :
  - **JavaScript**, langage de script, associé au DOM (*Document Object Model*), qui est une description normalisée (W3C) d'une page HTML ou XML. Tous les navigateurs modernes interprètent le JavaScript sans *plug-in*.
  - **Flash**, technologie d'origine *Macromedia* (*Adobe* depuis 2005), permet d'ajouter des animations et des objets interactifs à une page web. Nécessite l'installation d'un *plug-in* dans le navigateur.
  - ***Applet* Java**, code Java précompilé, permet d'introduire dans la page des éléments d'interface graphique Java (*awt* et/ou *swing*). Nécessite un *plug-in* dans le navigateur et une Machine Virtuelle Java (JVM).

Les techniques côté client (2/2)

- Les normes XHTML 1.0 strict, puis XHTML 1.1 ont modifié la façon de concevoir une page HTML :
  - La syntaxe HTML, qui est assez permissive, est rendue plus contraignante (par ex : balises fermantes obligatoires, attributs entre guillemets, balises écrites en minuscules,…) pour se rapprocher de XML. (XHTML 1.1 est en fait un dialecte XML)
  - La présentation est obligatoirement séparée du contenu : interdiction des balises de présentation (<font>, , <u>,…) et des attributs de présentation (background, bgcolor, align,…). La présentation doit être réalisée à l'aide des feuilles de style CSS.
  - Abandon des <frame> ou des <table> pour la mise en page, remplacés par une structure de blocs positionnés et dimensionnés à l'aide de classes CSS.

Une architecture courante



Source : http://fr.slideshare.net/fulvio.corno/introduction-to-javascript-programming

jean-pierre.gerval@isen-ouest.yncrea.fr

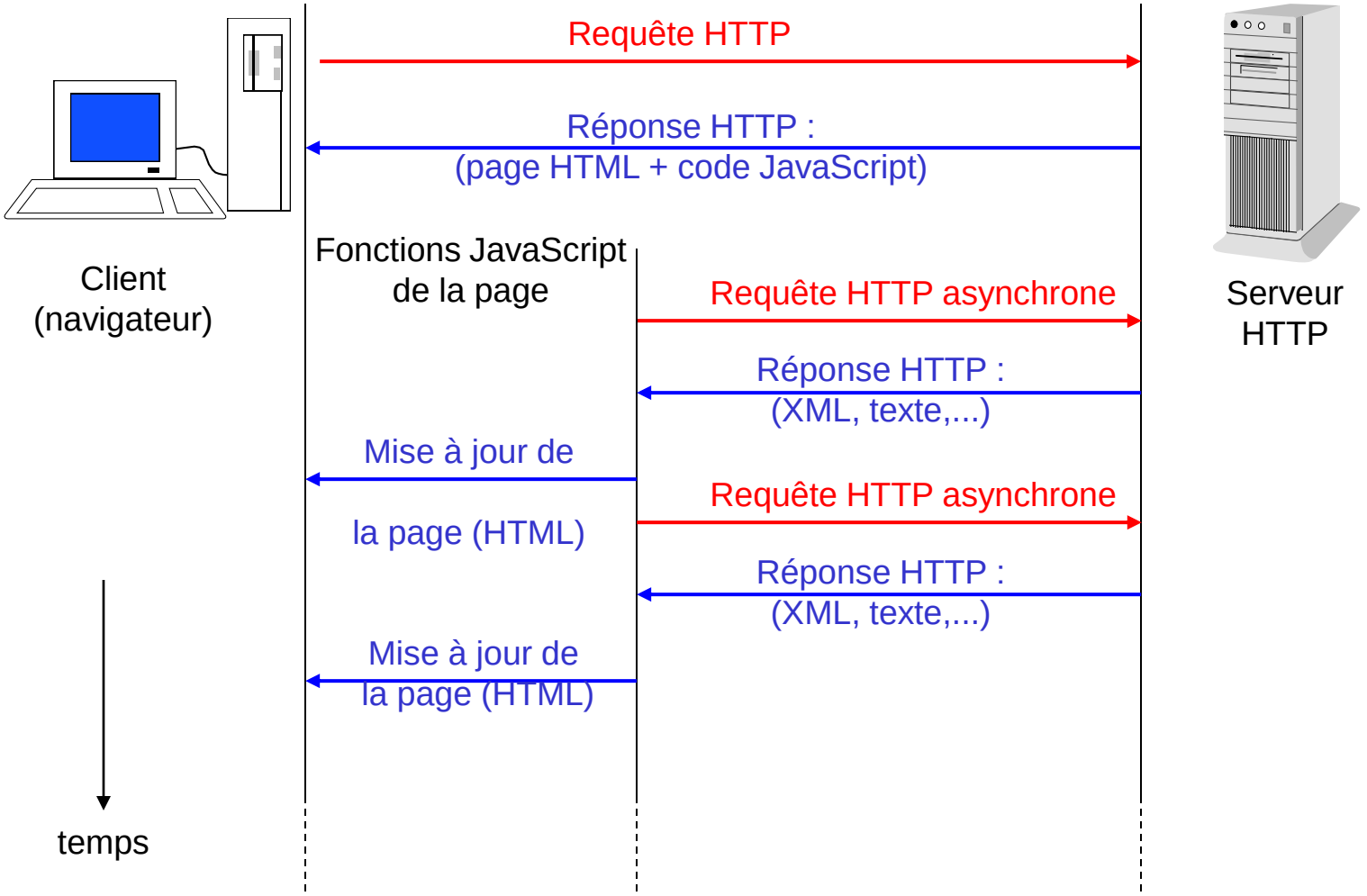
La technique « AJAX » (1/7)

- Le but de la technique AJAX (*Asynchronous JavaScript and XML*) est de pouvoir réaliser des appels « asynchrones » depuis un client (navigateur) vers un serveur web. « Asynchrone » signifie que l'appel ne suspend pas le travail en cours dans le navigateur : il est (+ ou -) transparent pour l'utilisateur, il n'y a pas de rechargement de page.
- Il s'agit en général de mettre à jour des éléments dans la page affichée, par lecture de données (qui peuvent être complexes) sur le serveur (sous forme XML ou autre forme « texte », car XML n'est pas obligatoire, contrairement à ce que laisse supposer le nom de la technique).

La technique « AJAX » (2/7)

- Le terme AJAX a été forgé en 2005 dans un article de *J. J. Garrett*, mais les technologies concernées existaient bien avant :
  - l'objet *XMLHttpRequest* a été créé en 1999 par Microsoft en tant qu'objet *ActiveX* pour Internet Explorer 5. A partir de 2002 il a été implémenté par Mozilla puis par d'autres navigateurs. Le W3C tente de standardiser l'objet (dernières spécifications publiées en 2009).
  - L'appel à *XMLHttpRequest* se fait « côté client », en langage JavaScript. C'est le script JavaScript qui se charge d'inclure les données reçues dans le document affiché, à l'aide de l'arborescence et des méthodes du DOM, et des CSS pour la mise en forme.
  - Les données chargées depuis le serveur peuvent être en XML, en HTML ou dans d'autres formats « riches » comme JSON (*JavaScript Object Notation*), ou tout simplement en texte brut.
  - Aucune technologie spécifique n'est requise « côté serveur », mais on peut employer des programmes PHP ou ASP pour générer les données ou transformer leur format (par exemple avec XSLT).

La technique « AJAX » (3/7)

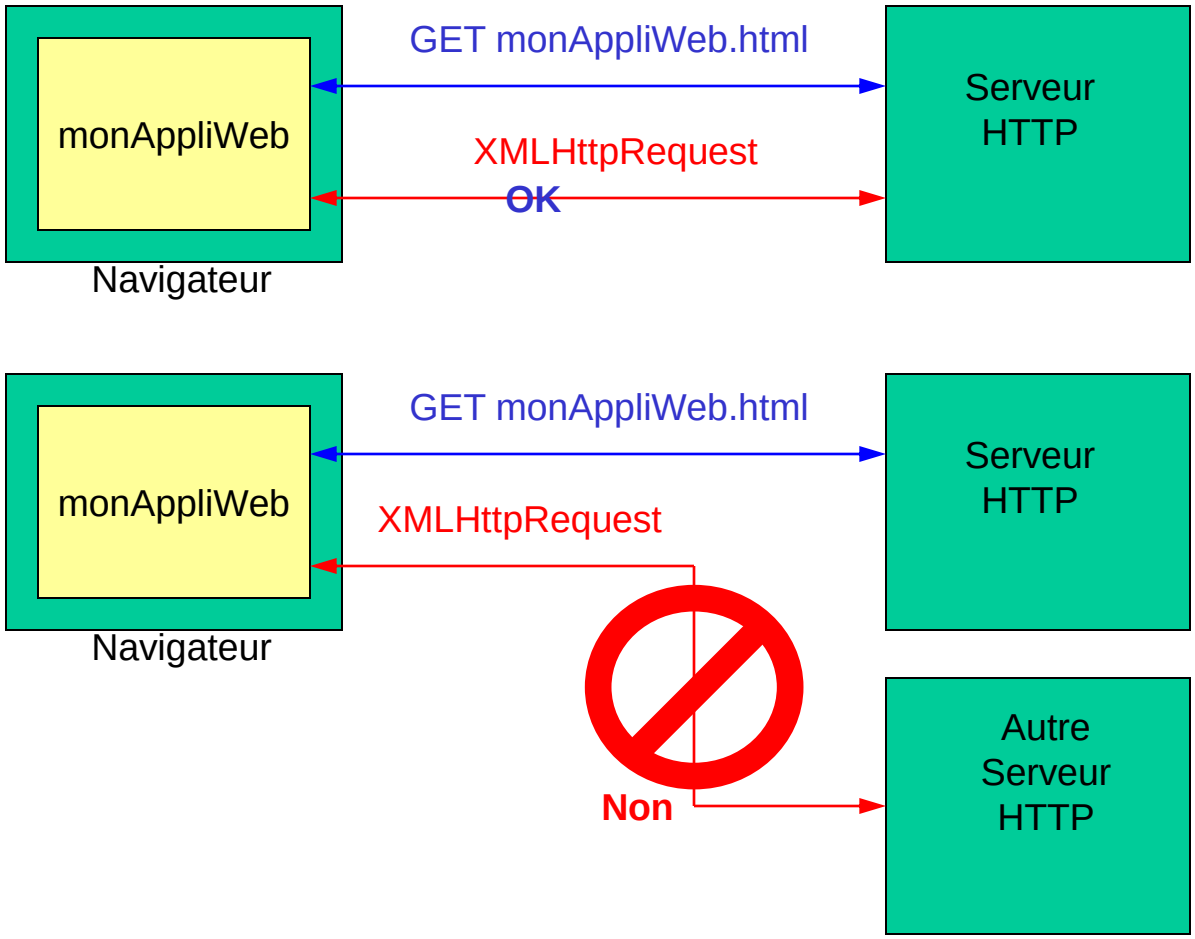


La technique « AJAX » (4/7)

- Avantages d'Ajax :
  - Réponse rapide aux actions de l'utilisateur : la plus grande partie du travail se fait en local, avec JavaScript, les requêtes au serveur sont réduites au minimum,
  - Réduction du trafic réseau: pas d'utilisation intensive de la bande passante,
  - Pas de *plugin* à installer : tous les technologies nécessaires sont *a priori* présentes en standard dans les navigateurs,
  - Portabilité : toutes les technologies nécessaires sont conformes aux normes W3C de base, respectées par les navigateurs modernes.
- Inconvénients d'Ajax :
  - L'utilisateur doit autoriser l'exécution de JavaScript sur son navigateur : il peut refuser pour des problèmes de sécurité,
  - Difficultés pour le « marque-pages » et l'historique (« *deep linking* ») : on ne peut pas revenir à un état précis d'une application,
  - Difficultés pour le référencement de pages par les moteurs de recherche,
  - Problème du « *Cross Site Request* » : la requête HTTP asynchrone n'est autorisée que sur le serveur d'origine de la page (« *Same Domain Policy* »),
  - Complexité de programmation (asynchrone) « *callback-style programming* ».

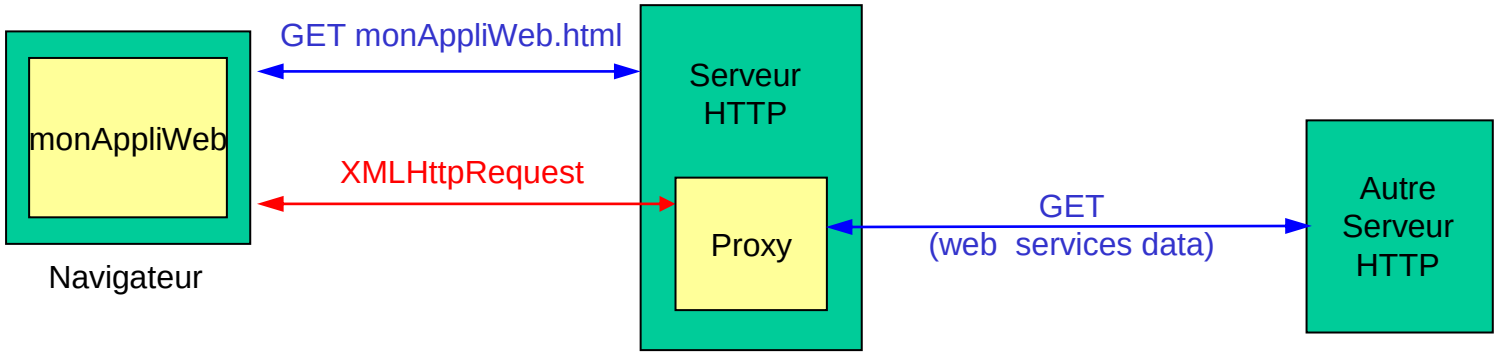
La technique « AJAX » (5/7)

- Le problème du « *Cross Site Request* » :



La technique « AJAX » (6/7)

- Des solutions au problème du « *Cross Site Request* » :
  - Solution ancienne, par contournement : un script (par ex PHP) sur le serveur (script « *Proxy Server* ») qui accédera au serveur distant à la place du client.



  - Solution récente : le W3C vient de normaliser (janvier 2014) un champ d'entête HTTP nommé *Access-Control-Allow-Origin*. Ce « *header* » (dans la réponse HTTP du second serveur) doit avoir une valeur qui correspond à l'URL d'origine (ou bien "*") pour que le navigateur accepte la réponse à la requête. Tous les navigateurs importants récents acceptent ce « *header* ».

La technique « AJAX » (7/7)

- Des solutions au problème du « *deep linking* » avec Ajax:
  - Solution ancienne, à l'aide des # (« *hash* ») dans l'URL : elle nécessite de la programmation Javascript . Cette solution est basée sur le détournement de la notion classique de « *fragment identifier* » en fin d'URL: par ex. dans l'URL *http://www.exemple.org/mapage.html#**truc***, « *truc* » est l'identifiant de fragment. Classiquement c'est souvent l'« id » d'un élément de la page et le navigateur doit faire le nécessaire pour que cet élément soit visible.

    Ici, on va créer une fonction Javascript qui ajoute un « #identifiant » à l'URL à chaque fois que l'on estime que l'application est, après une mise à jour Ajax, dans un état (« *state* ») stable, qui mérite d'être mémorisé. Il faut aussi ajouter une fonction Javascript qui détecte toute modification du « fragment » dans l'adresse et réaffiche la page correspondant à cet état.

  - Solution récente : la nouvelle spécification HTML5 du W3C a complété la notion d'historique par la notion de « *state object* », objet qui est stocké dans l'attribut « *history.state* » du navigateur. Tous les navigateurs n'acceptent pas encore cet attribut.

Les techniques « HTML5 » (1/2)

HTML5 (ou plutôt l'ensemble des technologies liées à l'apparition de HTML5) révolutionne l'architecture simple (REST) qui caractérisait les premières époques du Web. Ces techniques permettent :

- Stockage d'informations sur le client : API *Filesystem*, *WebStorage,* bases de données coté client (*WebSQL*, *IndexedDB*).
- Interactions complexes dans les deux sens : à la suite de *AJAX* (client→serveur) et *Comet* (serveur→client), apparaît l'API *WebSocket* (client↔serveur).
- *Multi-threading* sur le client : *WebWorkers*.

jean-pierre.gerval@isen-ouest.yncrea.fr

Les techniques « HTML5 » (2/2)