



# Ajax à la loupe

*En vue de son intégration à CartoWeb*



## T R A V A I L D E S E M E S T R E 2005

Lausanne, le 16 septembre 2005

Damien Corpataux  
Eivd département comem<sup>+</sup>  
Classe 31IT

---

**Expert interne**  
Claude Philipona  
Professeur

**Expert externe**  
Yves Bolognini  
Chef de projet, développeur  
Camptocamp SA

# Ajax à la loupe



## Table des matières

<b>1. Introduction.....</b>	<b>2</b>
But du document.....	2
Contenu du document.....	2
Aperçu de CartoWeb.....	2
<b>2. AJAX en général.....</b>	<b>3</b>
Fonctionnement.....	3
AJAX et les SOAs.....	4
Historique de l'acronyme AJAX.....	4
Inconvénients d'AJAX dans les applications web en général.....	4
<b>3. Utilisation d'AJAX dans le monde cartographique.....</b>	<b>6</b>
Apports d'AJAX dans la cartographie web.....	6
Exemples d'utilisation d'AJAX.....	6
Applications cartographiques web existantes.....	7
Googlemaps.....	7
MSN VirtualEarth.....	7
Map.search.ch.....	8
Frameworks cartographiques existants.....	8
Ka-Map.....	8
Mapbuilder.....	9
Enjeux de l'intégration d'AJAX dans CartoWeb.....	10
<b>4. Utilisation d'AJAX dans le monde open source.....</b>	<b>11</b>
Frameworks open source.....	11
API de services documentés.....	11
<b>5. Frameworks AJAX open source.....</b>	<b>12</b>
Catégories de frameworks.....	12
Difficultés rencontrées.....	12
Caractéristiques recherchées.....	14
Frameworks analysés.....	15
Frameworks retenus.....	15
<b>6. Résumé comparatif des frameworks AJAX.....</b>	<b>16</b>
<b>7. Conclusions.....</b>	<b>18</b>
Frameworks retenus.....	18
Problèmes à considérer.....	18
Objectifs pour le prototype CartoWeb.....	18
<b>8. Sources.....</b>	<b>20</b>
Liens internet.....	20
Bibliographie.....	20
<b>9. Déclaration d'honnêteté.....</b>	<b>21</b>
<b>10. Annexes.....</b>	<b>22</b>
Implications pour CartoWeb.....	22
Fonctionnement de Googlemaps.....	22
Exemple de fonctionnement de Ka-Map.....	23
Exemple d'appel asynchrone POST avec prototype.js.....	25

# 1. Introduction

La technologie AJAX promet des applications web plus réactives en permettant un comportement de l'interface utilisateur se rapprochant des applications desktop.

Bien que CartoWeb intègre des fonctionnalités plus étendues et plus complexes que la majorité des frameworks cartographiques web, il a besoin d'intégrer les innovations récentes pour rester compétitif et séduire de nouveaux utilisateurs<sup>1</sup>.

Pour le framework CartoWeb, il s'agit d'implémenter la technologie AJAX de la manière la plus élégante possible - c'est à dire simple, extensible et maintenable. En outre, CartoWeb étant en production chez des clients commerciaux, l'implémentation d'AJAX ne doit pas provoquer de cassure majeure dans son architecture et sa philosophie.

## But du document

Une étape importante dans le processus d'intégration d'AJAX à CartoWeb est de choisir un framework AJAX dont la multitude et la disparité sont les caractéristiques principales. En outre, il existe plusieurs manières d'implémenter AJAX. Pour répondre à ces questions, les frameworks AJAX et cartographiques existants doivent être analysés pour s'inspirer de la meilleure manière de faire.

Ce document a pour but d'apporter les éléments de réponse pour réaliser le prototype fonctionnel défini par les objectifs (voir *Annexes, Objectifs TD pour le prototype*).

De plus, ce document résume les discussions avec le client Yves Bolognini, chef de projet et développeur chez Camptocamp SA - principalement dans le chapitre *Conclusions*.

## Contenu du document

Le présent document fournit une vue d'ensemble d'AJAX, un aperçu de son fonctionnement – particulièrement dans le domaine de la cartographie – et un résumé comparatif des frameworks, complété par les problèmes et opportunités de son utilisation.

Certaines parties de ce document sont en langue anglaise (p.ex. le tableau comparatif des frameworks AJAX) pour faciliter leur réutilisation - si besoin est.

**En caractères distincts, ce document présente les aspects relatifs à CartoWeb, au fil des éléments analysés.**

## Aperçu de CartoWeb

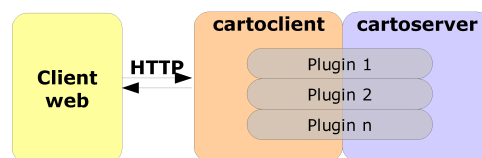
CartoWeb est un framework permettant de créer des applications cartographiques client/serveur, accessibles avec un client web (explorateur web). Il se divise en deux sous-parties: le *cartoclient* et le *cartoserver*.

Le client web effectue des requêtes par le protocole HTTP sur le cartoclient, qui communique avec le cartoserver – facultativement par le protocole SOAP, reçoit et traite les données pour les renvoyer au client web par HTTP.

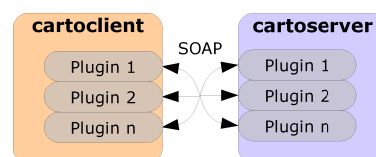
CartoWeb utilise le système de templates *Smarty* et renvoie à chaque requête du client web une page HTML complète.

L'implémentation d'AJAX permettra des échanges de messages asynchrones entre le client web et le cartoclient.

Le *dessin 2* montre que la partie interne de CartoWeb, c'est à dire le cartoclient et le cartoserver, peut communiquer de manière directe ou par des messages SOAP.



*Dessin 1* CartoWeb communique avec le client web via HTTP. Il traite ses requêtes, fait appel au cartoserver et renvoie les données au client web via HTTP.



*Dessin 2* CartoWeb en mode SOAP

<sup>1</sup> L'utilisateur lambda est plus sensible à l'aspect ergonomique qu'à l'aspect technique d'un produit.

## 2. AJAX en général

### Fonctionnement

Une application web client/serveur classique se compose d'une partie client (l'explorateur) et d'une partie serveur (le serveur web). La seule manière pour le client de communiquer avec le serveur est d'utiliser le protocole de transport HTTP, ce qui implique un rechargement de page à chaque communication.

En effet, à chaque clic de l'utilisateur, le client web effectue une requête HTTP vers le serveur et reçoit une page complète (headers HTTP et contenu HTML) que le client affiche en remplacement de la dernière.

Dans ce cas de figure, lorsque seul un élément de la page doit être modifié, cette dernière est entièrement recalculée, élément par élément. Dans le cas d'une application web compliquée – typiquement une application cartographique – cette manière de faire consomme du temps processeur côté serveur, de la bande passante et réduit les possibilités d'interactions utilisateur.

L'illustration 1 montre que AJAX se place comme couche sur la partie client. C'est donc le client qui devient responsable des appels asynchrones - entre rechargements de pages – et de la mise à jour de l'interface utilisateur – c'est à dire le DOM.

L'illustration 2 montre le fonctionnement séquentiel d'une application classique par rapport à une application AJAX – asynchrone. On remarque que le fonctionnement asynchrone permet aux actions utilisateurs d'obtenir un retour immédiat. Ceci enrichit le comportement de l'interface.

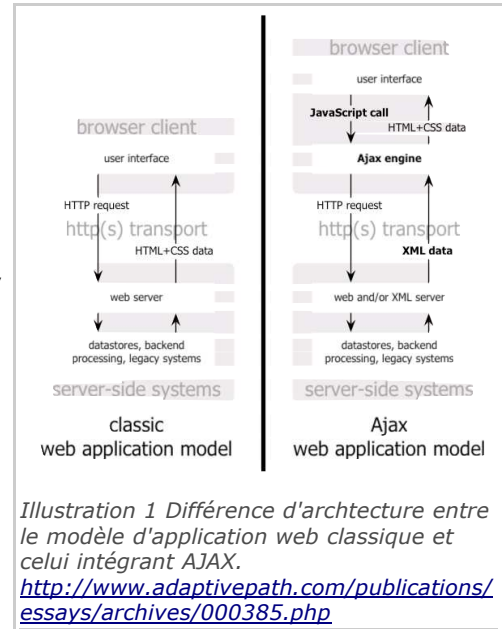


Illustration 1 Différence d'architecture entre le modèle d'application web classique et celui intégrant AJAX. <http://www.adaptivepath.com/publications/essays/archives/000385.php>

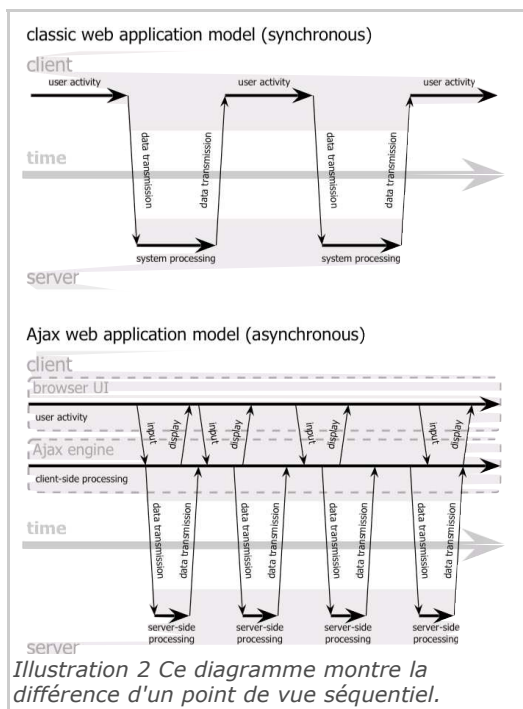


Illustration 2 Ce diagramme montre la différence d'un point de vue séquentiel.

#### Pour aller plus loin

Exemple en Java, structuré, avec diagramme de séquence: <http://www.javarss.com/ajax/j2ee-ajax.html>.

L'objet XMLHttpRequest vu par openweb.eu.org : [http://www.openweb.eu.org/articles/objet\\_xmlhttprequest/](http://www.openweb.eu.org/articles/objet_xmlhttprequest/).

## AJAX et les SOAs

Concernant CartoWeb, les messages asynchrones échangés entre le serveur et le client ne seront vraisemblablement pas de type SOAP. Selon de team de développement CartoWeb,

cette alternative est peu intéressante. En effet, la communication SOAP de CartoWeb est implémentée pour être utilisée de manière interne au serveur (entre le cartoclient et le cartoserver); le front-end de CartoWeb ne permet donc pas de communiquer les informations finales, utiles au client web. De plus, la jeunesse d'AJAX fait que son support SOAP n'est pas encore fiable.

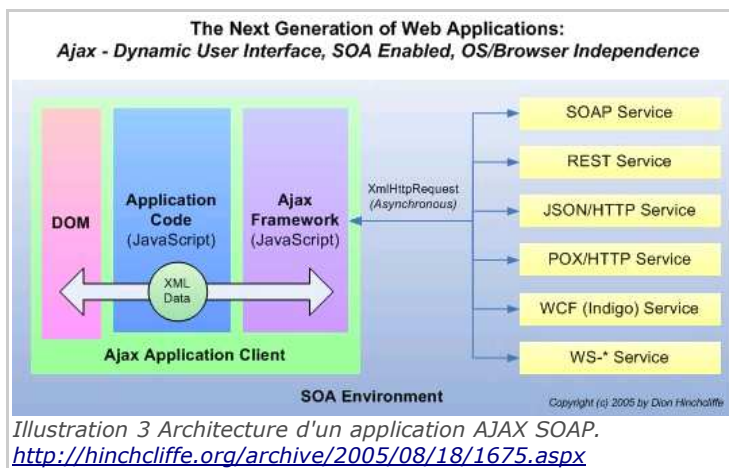
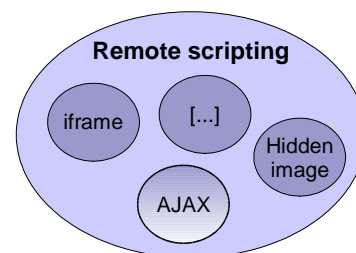


Illustration 3 Architecture d'un application AJAX SOAP.  
<http://hinchcliffe.org/archive/2005/08/18/1675.aspx>

Le schéma de l'illustration 3 montre l'architecture idéale pour l'utilisation asynchrone des WebServices dans une application web AJAX: le code Javascript de l'application s'appuie sur une framework AJAX capable d'échanger des messages SOAP (WebServices) et autres protocoles répandus (REST, JSON, etc).

## Historique de l'acronyme AJAX

L'ancêtre d'AJAX s'appelle Remote Scripting. Ce dernier utilise un élément HTML *iframe* pour recevoir des données asynchrones: en modifiant via Javascript la valeur de l'attribut *src* du *iframe*, les données se chargent dans le *iframe* – on peut alors les récupérer via Javascript, en accédant à la propriété DOM *innerHTML* de l'*iframe*. Il existe d'autres moyens de récupérer des données de manière asynchrone, par exemple la technique de l'image cachée. En ce sens, AJAX est un sous-groupe de la technique du remote scripting.



Dessin 3 AJAX est un sous-groupe de la technique du Remote scripting

Avec l'implémentation de l'objet XMLHttpRequest dans les principaux clients web et la propagation d'XML, cette technique s'est vue nommée AJAX, pour Asynchronous Javascript and XML.

Il existe une polémique sur cette appellation: certains soutiennent qu'il s'agit d'une technique marketing (un buzzword permet de vendre), d'autres pensent qu'il est bon de nommer ce regroupement technologique pour simplifier la communication.

## Inconvénients d'AJAX dans les applications web en général

Concernant l'implémentation prototype dans CartoWeb, le code Javascript sera réduit au maximum. Idéalement, il sera principalement responsable d'effectuer les requêtes asynchrones, puis recevoir et injecter du code HTML directement dans les éléments du DOM. Il pourra ainsi viser à être exempt de bugs, compatible avec les clients web principaux et le plus facilement maintenable possible. Les problèmes techniques et d'ergonomie cités ci-dessous devront dans la mesure du possible être pris en compte.

Dans son article *Ajax Mistakes*, Alex Bosworth énumère les problèmes posés par l'utilisation d'AJAX dans une application web<sup>2</sup>.

2 <http://alexbosworth.backpackit.com/pub/67688>

### Problèmes d'ergonomie

Une implémentation basique d'AJAX court-circuite plusieurs fonctionnalités largement utilisées par les internautes:

- Le bouton "Précédent" du client web ne fonctionne plus
- L'URL du client web ne reflète plus l'état de l'application
- Problèmes d'accessibilité pour les clients web anciens ou exotiques

De plus, un recours systématique à AJAX peut poser problème:

- Les utilisateurs peuvent être déroutés en ne retrouvant pas les interactions classiques attendues

### Problèmes techniques et d'implémentation

- Problèmes causés aux robots (référencement) et aux clients web anciens ou peu répandus
- La cohérence, la lisibilité et la maintenabilité du code javascript sont difficiles à garantir

### Problèmes liés au manque d'uniformité des clients web

L'implémentation Javascript des clients web n'est pas uniforme, ce qui favorise les astuces difficilement documentables et le code-forking<sup>3</sup>.

Parmi les clients web, Internet Explorer est particulièrement source de workarounds:

- Bug: Internet Explorer makes *responseXML.url* unavailable
  - Workaround: <http://support.microsoft.com/default.aspx?scid=kb;EN-US;234460>
- Bug: Internet Explorer memory leak – and solution
  - Book: SitePoint DHTML Utopia: Modern Web Design Using Javascript & DOM, Stuart Langridge, p. 182, *Fixing IE Memory Leak*.
  - MSDN Workaround: [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ietechcol/dnwebgen/ie\\_leak\\_patterns.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ietechcol/dnwebgen/ie_leak_patterns.asp)
  - Solution: <http://novemberborn.net/javascript/event-cache>
- Bug: IE implementation of *attachEvent* makes *this* point the wrong element
  - The heavily used *addEvent* function – meant to be cross-browser – is therefore broken.
  - Workaround: [http://www.quirksmode.org/blog/archives/2005/08/addevent\\_consider.html](http://www.quirksmode.org/blog/archives/2005/08/addevent_consider.html)
  - Recoding request: [http://www.quirksmode.org/blog/archives/2005/09/addevent\\_recode.html](http://www.quirksmode.org/blog/archives/2005/09/addevent_recode.html)

### Problèmes posés aux standards

Les organismes de standardisation s'efforcent d'imposer des standards afin de simplifier le développement, résoudre les problèmes d'accessibilité et rendre le monde meilleur. On se souvient des années 90 ou les sites inaccessibles, le code-forking et la mode *the bug is a feature*<sup>4</sup> étaient monnaie courante.

Ces dernières années, les standards se sont imposés, les pages codées en XHTML se sont répandues et l'usage du Javascript s'est raréfié.

Cependant, avec l'avènement d'AJAX, les standards se voient mis de côté – et l'accessibilité et la maintenabilité avec.

Un développeur AJAX a alors plusieurs alternatives. Parmi elles: restreindre l'accessibilité à certains clients web (trop exclusif mais parfois justifiable), coder afin de garantir l'accessibilité (complexe) ou fournir un fall-back pour les clients incompatibles (code-forking).

<sup>3</sup> Le code-forking désigne le fait de coder plusieurs fois une même fonction, afin de prendre en compte les différences d'interprétation de la part des clients-web. On essaye d'éviter ceci pour une meilleure productivité et maintenabilité du code.

<sup>4</sup> *Designing With Web Standards*, by Jeffrey Zeldman, New Riders Press, 2003.

## 3. Utilisation d'AJAX dans le monde cartographique

### Apports d'AJAX dans la cartographie web

Les apports d'AJAX dans la cartographie web sont idéalement doubles: ils promettent de meilleures performances et des interactions utilisateur enrichies.

#### Meilleures performances

Traditionnellement, chaque action utilisateur donne lieu à un rafraichissement de la page entière. Le rafraichissement d'une partie de la page oblige donc la partie serveur à régénérer cartes, requetes, légendes, etc...

Grâce à AJAX, seule une partie de la page peut être réactualisée. Par conséquent, la partie serveur traite un volume restreint de données à chaque action utilisateur. Cependant, ceci peut être faux en fonction du type d'application. A partir d'un certain degré de complexité, il faut donc penser performances dès la conception – c'est-à-dire du coté serveur *et* du côté client, qui se voit effectuer d'une partie des traitements de l'application.

#### Interactions utilisateur enrichies

L'utilisation de la technologie AJAX permet principalement:

- Recentrage (panning) continu: navigation fluide sur la carte
- La validation et l'autocomplétion des entrées utilisateur dans les formulaires
- L'insertion ou la récupération d'enregistrements dans une base de données
- L'affichage de popups HTML contenant des informations dynamiques (issues d'une BD, d'un Webservice ou autre), par exemple lors du survol d'une zone active par la souris de l'utilisateur
- Une bonne expérience utilisateur grâce aux effets visuels produits par la manipulation du DOM permettent (p.ex. le déplacement des barres d'outils).

### Exemples d'utilisation d'AJAX

#### Recentrage (panning) continu

L'image générée par le serveur cartographique est scindée en pièces et reconstituée dans la page web sous forme de mosaïque (tableau ou div html). Lorsque l'utilisateur recentre la carte, seules les pièces manquantes de l'images sont chargées.

Pour ce faire, la technique de la mosaïque est utilisée<sup>5</sup>.

Ceci permet une meilleure interactivité (la carte est déplacée en temps-réel sur la page) et de meilleures performances (seules les pièces manquantes de la carte sont générées par le serveur cartographique).

#### Affichage de données attributaires à la volée

Lorsque l'utilisateur place le curseur sur un point de la carte relatif à une donnée attributaire, les informations peuvent être affichées à la volée, sans rechargement des données déjà présentes.

#### Amélioration des performances

En ne chargeant qu'une partie de la page (par exemple des données attributaires à afficher), on économise du temps processeur et de la bande passante côté serveur en plus d'offrir à l'utilisateur une meilleure réactivité de l'application.

5 [http://markpasc.org/weblog/2005/05/28/landmarker\\_or\\_so\\_you\\_d\\_like\\_to\\_make\\_an\\_ajax\\_map](http://markpasc.org/weblog/2005/05/28/landmarker_or_so_you_d_like_to_make_an_ajax_map)

## Applications cartographiques web existantes

Pour le prototype AJAX CartoWeb, il me semble évident que créer un API orienté objet est une bonne idée: cela offre un jeu de méthodes simples à utiliser en plus des avantages d'une couche d'abstraction et de l'orienté objet<sup>6</sup>. L'API Javascript de CartoWeb s'appuierait sur d'autres frameworks de bas niveau: abstraction d'HttpRequest et éventuellement modification DOM (par exemple avec le framework Behaviour<sup>7</sup>).

### Googlemaps<sup>8</sup>

Googlemaps est l'application AJAX phare de google (avec google suggest et gmail). Ses forces sont sa convivialité et son look. Sa faiblesse est son incapacité à gérer une hiérarchie de couches complexe de manière performante.

La rapidité du chargement des cartes, leur lisibilité et la consistance de leur API Javascript font de Googlemaps une référence en terme de développement bleeding-edge des technologies web.

Sa convivialité est due à:

- Un recentrage continu de la carte (création d'une mosaïque d'images chargées à la volée)
- Un affichage des données attributaires dans des popups HTML
- Un outil de routing pour itinéraire (basique)
- Une mise à disposition d'un URL permettant de retrouver l'état de l'application (*link this page*)
- Une interface utilisateur bien conçue: grande taille de carte, recherche d'adresses full-text, vision d'ensemble (données attributaires dans des info-bulles placés sur la carte)

Aussi, google maps a-t-il fait figure d'innovateur dans sa conception d'interface utilisateur:

- Début du changement sur la manière de manipuler une application cartographique web  
Ceci pose plusieurs problèmes d'ergonomie, dont:
  - Equilibre entre le fonctionnement traditionnel et innovant – l'utilisateur peut être confus face à des interaction inhabituelles car l'internaute s'attend à une uniformité dans l'utilisation d'une application web
  - La mise à jour d'un fragment de la page risque d'échapper à l'utilisateur

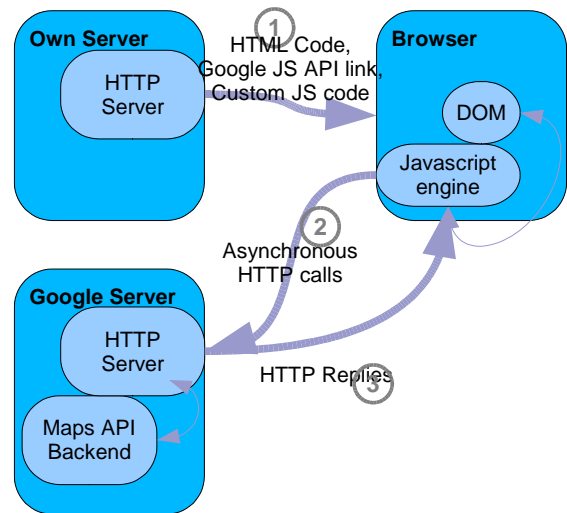
#### Fonctionnement

Voir: Annexes, Fonctionnement de Googlemaps.

### MSN VirtualEarth<sup>9</sup>

VirtualEarth est le concurrent direct de Googlemaps. Ses fonctionnalités sont similaires mais implémentées différemment. Googlemaps reste supérieur.

A noter que VirtualEarth permet d'utiliser la roulette de la souris pour contrôler le niveau de zoom – également dans Mozilla, bien vu.



Dessin 4 Fonctionnement de Googlemaps, point de vue messages HTTP.

6 Bien que Javascript ne soit pas réellement orienté objet, il est possible d'encapsuler des fonctions et des variables dans des pseudo-objets. Le typage étant faible, les seuls avantages de cette pratique est d'encapsulation: cela évite les conflits de nom de fonctions et de variables et favorisent la lisibilité du code.

7 <http://bennolan.com/behaviour/>

8 <http://map.google.com>

9 <http://virtualearth.msn.com/>

## Map.search.ch<sup>10</sup>

Map.search.ch est très similaire à Googlemaps: il permet la navigation continue, l'affichage de popups pour les données attributaires et la recherche d'adresses – mais il se limite à la Suisse.

Contrairement à Googlemaps, map.search.ch ne publie pas son API et reste ainsi une application 100% blackbox. De plus, le code Javascript semble être moins structuré. Il est compressé<sup>11</sup> et donc moins lisible (code source de la page web). Son aspect graphique est moins polishé.

### Fonctionnalités manquantes – par rapport à Googlemaps

- Full-text search
- Outil de routing pour les itinéraires

### Fonctionnalités supplémentaires – par rapport à Googlemaps

- Zoom continu (les pixels s'agrandissent en temps réel)
- En plus des données attributaires, affichage de données externes - comme les horaires pour les transports publics – utilisation probable de WebServices
- A la vue aérienne raster s'ajoute une couche pour les routes - certainement vectorielle. Son calage et l'uniformité du raster sont impressionnants

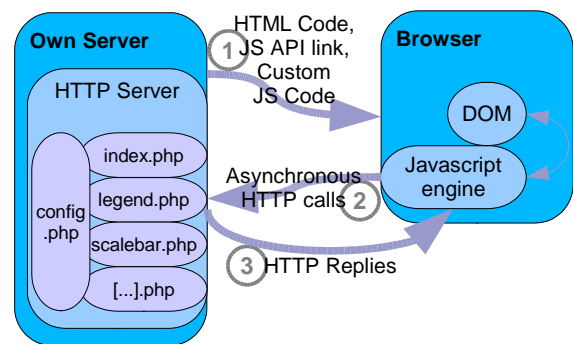
## Frameworks cartographiques existants

### Ka-Map<sup>12</sup>

Ka-Map vise à fournir un API Javascript pour développer des applications cartographiques réactives en utilisant la technologie AJAX. La partie server-side est écrite en PHP.

Ka-Map se configure via un fichier PHP (chemin du . map file, taille des pièces de la mosaïque, chemin du module MapScript).

Bien que le but de Ka-Map soit de fournir un API Javascript, les applications Ka-Map se construisent à la main: on code la structure HTML de la page et on écrit le code javascript décrivant le comportement.



Dessin 5 Fonctionnement de KaMap, point de vue messages HTTP.

Le code javascript de l'exemple fourni avec Ka-Map est peu documenté, peu structuré, mais lisible.

On attache à la main les gestionnaire d'événements aux widgets HTML les appels asynchrones (AJAX) appellent un URL (généralement un script PHP) et récupèrent généralement un URL au format texte. Le contenu du DOM est ensuite mis-à-jour, également à la main.

En ce sens, Ka-Map est plutôt centré sur la fonctionnalité de panning continu (technique de la mosaïque).

### Exemple d'actualisation de la barre d'échelle

Voir: *Annexes, Exemple de fonctionnement de Ka-Map.*

<sup>10</sup> <http://map.search.ch>

<sup>11</sup> Epuration des espaces successifs, des retours de ligne, etc. Ceci permet d'économiser du trafic et de la bande passante pour un site très fréquenté.

<sup>12</sup> <http://ka-map.maptools.org/>

**Avantages**

- Implémentation asynchrone des opérations cartographiques de base: recentrage continu (gestion de la mosaïque).
- Le code proposé dans l'exemple est assez simple et peut servir de piste pour l'implémentation d'AJAX dans CartoWeb.

**Inconvénients**

- Version beta (v0.1.1): instabilité.
- Manque de documentation, de tutoriaux et d'exemples – un exemple est fourni avec l'archive.
- Architecture et implémentation peu consistante: une grande partie des fonctionnalités est à implémenter à la main.

**Articles**

Build AJAX-Based Web Maps Using ka-Map:

<http://www.xml.com/pub/a/2005/08/10/ka-map.html> (tutorial)

**Mapbuilder<sup>13</sup>**

Mapbuilder est une librairie Javascript avec une partie server-side PHP et Java. Il est implémenté selon l'architecture Model-View-Controller<sup>14</sup> (utilisée par Ruby on Rails).

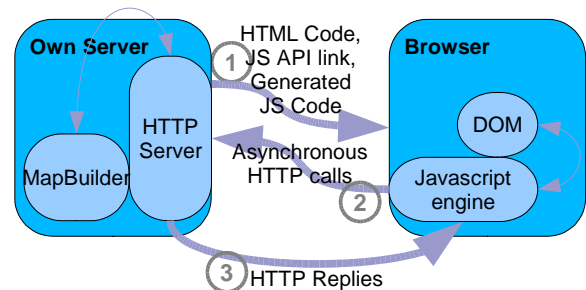
Pour créer une application Mapbuilder basique, il faut modifier le fichier de configuration et insérer des snippets javascript dans la page HTML<sup>15</sup>.

L'application est principalement décrite par les fichiers de configuration. La présentation se fait par CSS et modification des fichiers HTML. Le lien entre les widgets de Mapbuilder et les éléments HTML se fait par spécification de leur id HTML dans le fichier de configuration.

La partie serveur de Mapbuilder est écrite en PHP et en Java.

Exemple de modèle: <http://geoservices.cgdi.ca/mapbuilder/demo/data/context/atlasWorld.xml>

Exemple de config: <http://geoservices.cgdi.ca/mapbuilder/demo/simple/simpleConfig.xml>



Dessin 6 Fonctionnement de MapBuilder, point de vue messages HTTP.

**Avantages**

- Implémentation asynchrone des opérations cartographiques de base: recentrage semi-continu (PAS de gestion de la mosaïque), gestion automatique des couches et labels.
- Bonne activité du projet: 93.38% sur sourceforge.net / 9 développeurs

**Inconvénients**

- Version beta (v0.4): instabilité
- Manque de documentation

**Articles**

Build AJAX-Based Web Maps Using ka-Map (exemple):

<http://geoservices.cgdi.ca/mapbuilder/userGuide?page=start/start>

Mapbuilder demo:

<http://geoservices.cgdi.ca/mapbuilder/demo/complete/>

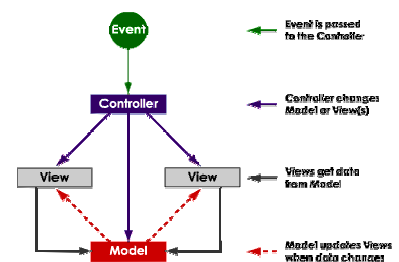


Illustration 4 Le modèle MVC (model, controller, view) est également promu par Ruby On Rails.

13 <http://geoservices.cgdi.ca/mapbuilder/>

14 <http://en.wikipedia.org/wiki/MVC>  
<http://java.sun.com/blueprints/patterns/MVC-detailed.html>

15 <http://geoservices.cgdi.ca/mapbuilder/userGuide?page=start/start>

## Enjeux de l'intégration d'AJAX dans CartoWeb

CartoWeb est une application open source implémentée chez de nombreux clients. Parmi eux, le *Département des infrastructures du canton de Vaud* (DINF<sup>16</sup>), l'*Inventaire Forrestier National* français (INF<sup>17</sup>) et l'*Ecole Polytechnique Fédérale de Lausanne* (EPFL<sup>18</sup>).

Ces clients ont opté pour CartoWeb en raison de ses fonctionnalités avancées, modulaires et extensibles. Ils peuvent par exemple modifier le fonctionnement de l'application, créer leurs propres plugins, modifier ceux existants, etc. Les implémentations en production de CartoWeb sont donc souvent complexes et comportent des fonctionnalités avancées.

Pour cette raison, chaque version de CartoWeb se doit d'être en continuité avec les précédentes, afin d'éviter aux clients des adaptations du code ou de l'architecture de leur version existante – et par là même éviter des coûts malvenus.

**Idéalement, la version AJAX de CartoWeb doit suivre cette règle et ne doit impliquer aucune modification du noyau du framework ou des plugins. Pratiquement, elle visera à les minimiser.**

---

16 <http://www.geoplanet.vd.ch/>

17 <http://www.ifn.fr>

18 <Http://map.epfl.ch>

## 4. Utilisation d'AJAX dans le monde open source

### Frameworks open source

L'intérêt pour AJAX a gagné le monde commercial comme celui de l'open source. Une multitude de frameworks ont ainsi vu le jour depuis 2003, mais surtout depuis février 2005 – en raison du lancement de *Gmail*, *Googlemaps* et *Googlesuggest*.

L'utilisation de frameworks est inévitable afin de permettre une couche d'abstraction pour résoudre les problèmes de compatibilité des clients web et automatiser le processus d'appels asynchrones et du traitement des informations retrouvées.

Le monde open source a déjà développé une multitude de frameworks:

- [ajaxmatters.com](http://www.ajaxmatters.com/r/resources?id=17): liste plus de 35 frameworks
- [sourceforge.net](http://sourceforge.net/search/?type_of_search=soft&forum_id=0&group_id=0&atid=0&words=ajax+AND+framework): liste environ 20 projets, dont 5 ayant une activité supérieure à 15%
- [ajaxpatterns.org](http://www.ajaxpatterns.org/Ajax_Frameworks): liste 38 projets tous azimuts

Certains frameworks ajoutent des fonctionnalités d'effets visuels aux fonctionnalités d'abstraction de l'objet XMLHttpRequest (modification du DOM, drag'n'drop, yellowfade, shake/shade/fade effects, etc). Ceci est justifiable, car AJAX implique également la modification du DOM et des effets visuels – ces derniers attirent l'attention de l'utilisateur sur les changements dans la page.

### AJAX et Ruby On Rails

Le team Ruby On Rails intègre directement la technologie AJAX à son framework. Il se base sur le framework libre *Prototype.js* – également utilisé par les renommés *Script.aculo.us* et *Open-Rico*.

### API de services documentés

Dans son fonctionnement, AJAX est similaire aux WebServices (p.ex. SOAP): les données transitent sur le port HTTP (80) et l'appel unitaire à des méthodes est possible. On peut donc utiliser AJAX comme un Webservice.

[Ajaxmatters.com](http://www.ajaxmatters.com) liste 19 API publics<sup>19</sup>. Parmi eux, certains utilisent le protocole SOAP – comme *Google AdWords*, d'autres utilisent une convention de format d'url pour l'appel de méthodes.

*43things.com* utilise cette technique. On peut rechercher des personnes en appelant l'url suivant: [http://www.43things.com/service/search\\_people?api\\_key=1234&q=erik](http://www.43things.com/service/search_people?api_key=1234&q=erik). Cet url retourne du XML contenant les informations sur les personnes correspondantes au mot-clé *erik*. Si on utilise Javascript pour effectuer la requête, parser et afficher les données en retour sur la page web, alors on pratique du Webservice à la mode AJAX!

Dans le monde commercial, la mise à disposition d'un API se répand de plus en plus. Parmi les grands, on retrouve *Amazon*, *Weatherbug* (USA), *UPS*, *eBay*, *Google*, *Yahoo* et *Flickr*. Certains n'utilisent que SOAP, d'autres proposent des protocoles liés aux WebServices (XMLRPC, REST), le reste propose de télécharger l'API pour différents langages. Peu utilisent une technique triviale de *43things.com*.

<sup>19</sup> <http://www.ajaxmatters.com/r/resources?id=28>

## 5. Frameworks AJAX open source

Les répertoires de frameworks principaux sont [http://www.ajaxpatterns.org/Ajax\\_Frameworks](http://www.ajaxpatterns.org/Ajax_Frameworks) et <http://wiki.osafoundation.org/bin/view/Projects/AjaxLibraries>.

### Catégories de frameworks

D'un point de vue architecture, ajaxpatterns.org classe les frameworks en deux grandes catégories:

- Pur javascript  
Fournissent uniquement des fonctions cross-browser côté client web.
- Server-side framework  
Fournissent l'export de méthodes côté serveur et généralement la génération des stubs Javascript coté client. Ce fonctionnement est similaire aux *proxys* de WebServices SOAP.

### Difficultés rencontrées

A ce stade d'évolution, les frameworks AJAX ne sont pas encore arrivés à leur stade de maturité. La technologie elle-même est sans cesse source de découvertes de bugs, de workarounds et de nouvelles techniques. On trouve alors une multitude de frameworks ayant des approches différentes. Il est donc difficile d'évaluer la viabilité de chacun, puisque tous les critères ne sont pas encore connus.

Il est néanmoins possible de choisir un framework en fonction de l'étendue de sa base utilisateur, du nombre de développeurs et de son activité. Par exemple, Sarissa est un de premiers frameworks a avoir vu le jour (février 2003 selon sourceforge.net<sup>20</sup>). Mais, il est grandement abandonnée aujourd'hui car il n'est pas orienté objet et son code n'est plus au goût du jour.

### Approches multiples

Les approches adoptées par les concepteurs des frameworks sont très variables. On distingue cependant quatre approches principales.

- Simple couche d'abstraction de l'objet ActiveX d'appels asynchrones ou XMLHttpRequest: résoud simplement les problèmes de compatibilité inter-clients web.
  - Les appels asynchrones retournent du text uniquement, celui-ci étant directement injecté dans un élément HTML, voire transformé sommairement - pas de parsing XML.  
Article: <http://www.onlamp.com/pub/a/onlamp/2005/05/19/xmlhttprequest.html>  
Exemple: <http://www.mien.ch/ajaxmusic/>
  - Les appels asynchrones retournent du XML, parsé par le Javascript côté client.
- Export de méthodes server-side:  
des méthodes d'un langage de script côté serveur (PHP/ASP/Java) peuvent être exposées et être appelées de manière simple via Javascript, qui recoit ce que retourne la méthode.
  - Le retour de la méthode est sérialisé en XML – souvent selon un format propre au framework – et désérialisé dans une variable Javascript côté client. Les types supportés pour la sérialisation dépend des frameworks:
    - Support des types primitifs, des tableaux uni et multidimensionnels, des objets
    - Certains frameworks ne retournent pas de valeur: la modification du DOM est décrite par du code PHP via l'API du framework (p.ex. XAJAX)
- Certains frameworks clament supporter SOAP
- Certains frameworks ajoutent des fonctionnalités d'effets visuels aux fonctionnalités d'abstraction de l'objet XMLHttpRequest

---

<sup>20</sup> Avant l'arrivée de de l'objet XMLHttpRequest, les appels asynchrone de Sarissa devaient utiliser la technique de l'iframe ou de l'image cachée .

### Implémentation SOAP

Les frameworks AJAX supportant SOAP sont sujets aux bugs ou sont incomplets: le traitement des réponses par javascript n'est pas optimal et réduit la compatibilité entre browsers. En effet, l'implémentation de SOAP dans les frameworks AJAX est à ses débuts.

Glm-AJAX propose le support SOAP mais la documentation ne spécifie rien concernant les différents standards, par exemple le type d'encodage. De plus, il ne propose pas la création d'un objet proxy pour faciliter son utilisation.

D'après mes tests, ce framework plante à la réception d'un message SOAP généré par NuSOAP/PHP, encodé selon XML-RPC.

A priori, l'interfaçage avec CartoWeb peut se faire de trois manières:

- Utiliser une librairie JS pur supportant SOAP et appeler les methodes via SOAP (Ajax framework project)
- Utiliser un framework server-side et exporter les méthodes de l'API CartoWeb. (SAJAX, XAJAX)
- Utiliser un framework pur Javascript, recevant au format HTML le contenu des éléments du DOM à mettre à jour

La troisième solution semble la plus viable car elle évite l'utilisation trop poussée d'AJAX. Le parsing XML intensif, les transformations XSL et les appels SOAP introduisent des incompatibilités, des effets de bord et rendent débbugage difficile - en raison du code Javascript trop complexe et conséquent.

**Suite aux discussions avec le team de développement, cette solution est la plus viable. Il est envisagé de créer une couche weapper sur le cartoclient pour retourner les données à mettre à jour.**

## Caractéristiques recherchées

Concernant CartoWeb, il semble judicieux d'utiliser un framework dédié uniquement à l'abstraction de l'objet XMLHttpRequest pour minimiser les bugs et maximiser la compatibilité.

Il me semble important de rester simple dans cette phase d'implémentation prototype pour éviter les effets de bord dus à des API trop complexes. De plus, une base utilisateur et documentation suffisante promettent de bonnes sources d'information et la viabilité du framework.

Cependant, si l'architecture de CartoWeb permet de récupérer dans une variable le résultat du parsing d'un seul template – et ceci de manière simple, il peut être intéressant de se tourner vers un framework server-side permettant l'export de méthodes PHP.

Le framework recherché doit:

- Se concentrer sur les requêtes HTTP asynchrones et leur traitement:  
certains frameworks incluent une API d'effets visuels sur le DOM (déplacement d'éléments, drop-downs, shake/blink/fade effects, etc.)
- Être capable de faire des requetes HTTP POST en plus de GET
- Être orienté objet:  
encapsuler la logique dans un objet permet une meilleure lisibilité du code et évite les conflits (noms de variables et de fonctions)
- Être cross-browser:  
compatibilité minimum: IE6, Firefox 1.0 (Mozilla 5.0), Safari 1.3
- Avoir un team de développement et une base utilisateurs suffisamment grande et active:  
signes d'évolutivité et de viabilité du framework
- Avoir une documentation suffisante – idéalement avec exemples, tutoriaux, forums et workarounds
- Si une partie server-side est proposée, elle doit être écrite en PHP 5

## Frameworks analysés

Catégorie	Nom	URL
Pur Javascript	Prototype	<a href="http://prototype.conio.net/">http://prototype.conio.net/</a> Documentation: <a href="http://www.sergiopereira.com/articles/prototype.js.html">http://www.sergiopereira.com/articles/prototype.js.html</a>
	OpenRico	<a href="http://openrico.org/rico/home.page">http://openrico.org/rico/home.page</a>
	Ajax framework project	<a href="http://glm-ajax.sourceforge.net/">http://glm-ajax.sourceforge.net/</a> <a href="http://sourceforge.net/projects/glm-ajax">http://sourceforge.net/projects/glm-ajax</a>
	Sarissa	<a href="http://sourceforge.net/projects/sarissa/">http://sourceforge.net/projects/sarissa/</a> <a href="http://sarissa.sourceforge.net/doc/">http://sarissa.sourceforge.net/doc/</a>
	DOJO	<a href="http://dojotoolkit.org/">http://dojotoolkit.org/</a>
	HTMLHttpRequest	<a href="http://www.twinhelix.com/javascript/htmlhttprequest/">http://www.twinhelix.com/javascript/htmlhttprequest/</a>
	Script.aculo.us	<a href="Http://script.aculo.us/">Http://script.aculo.us/</a>
Server-side PHP	AjaxAC	<a href="http://ajax.zervaas.com.au/">http://ajax.zervaas.com.au/</a>
	JPSpan	<a href="http://sourceforge.net/projects/jpspan">http://sourceforge.net/projects/jpspan</a> Examples: <a href="http://blog.joshuaeichorn.com/archives/2005/04/19/ajax-hello-world-with-jpspan/">http://blog.joshuaeichorn.com/archives/2005/04/19/ajax-hello-world-with-jpspan/</a>
	NAJAX	<a href="http://najax.sourceforge.net/">http://najax.sourceforge.net/</a>
	XAJAX	<a href="http://xajax.sourceforge.net/">http://xajax.sourceforge.net/</a>
	PEAR::HTML::Ajax	<a href="http://pear.php.net/package/HTML_AJAX">http://pear.php.net/package/HTML_AJAX</a>
Server-side multi-langages	SAJAX	<a href="http://www.modernmethod.com/sajax/">http://www.modernmethod.com/sajax/</a>
	CPAINT	<a href="http://cpaint.sourceforge.net/">http://cpaint.sourceforge.net/</a> Documentation: <a href="http://cpaint.sourceforge.net/doc/">http://cpaint.sourceforge.net/doc/</a>

## Frameworks retenus

Les frameworks retenus satisfont à ces exigences: est orienté objet, supporte le requêtes HTTP POST et fournit une documentation suffisante. De plus, la conception doit être intelligente, et la communauté importante.

### Prototype (pur JS)

Prototype.js est un framework pur Javascript. Il est le favori pour l'implémentation d'AJAX dans CartoWeb. En effet, il est bien conçu, bien documenté et très utilisé. Son code est 100% orienté objet et implémente l'héritage.

D'autres projets comme Open-Rico, Script.aculo.us et Ruby on Rails intègrent ce framework. Ces éléments favorisent sa compatibilité et sa longévité.

De plus, il se concentre uniquement sur la fonction d'appels asynchrone côté client, ce qui évite une complexité due à des fonctionnalités inutiles dans notre cas de figure.

### CPAINT (server-side)

CPAINT comporte un server-side permettant de coder une partie de la logique en PHP et y faire appel par Javascript de manière simple. Au cas où l'export de méthode s'avérait une bonne alternative, ce framework semble le plus approprié. En effet, il satisfait aux conditions de sélection.

### Script.aculous.us<sup>21</sup> (comporte des fonctionnalités d'effets visuels)

Au cas où des effets visuels et autres manipulations avancées du DOM étaient requis, script.aculo.us est parmi les plus répandus, les mieux conçus et documentés.

<sup>21</sup> <http://script.aculo.us/>

## 6. Résumé comparatif des frameworks AJAX

Voir: page suivante.

### Explicatif du résumé

Le résumé comparatif des frameworks est sous forme de tableau. Les données proviennent de la littérature propre aux frameworks (sites officiels), de la littérature connexe (portails ajax, etc) et des tests effectués personnellement.

Le but de ce tableau est de donner une vision d'ensemble des frameworks, regroupant les points comparables. Certaines informations, comme l'activité du développement, le nombre de développeurs et la compatibilité envers les clients web sont difficiles à évaluer par manque sources sûres (p.ex. les données projet de sourceforge.org). Ce manque d'information se traduit dans le résumé par un point d'interrogation (?).

Ce tableau fait état de la situation au 13 septembre 2005.

### Note d'évaluation

Pour chaque framework, une note d'évaluation est attribuée dans la colonne *Nom*. Celle-ci est calculée en fonction des caractéristiques analysées. Pour les caractéristiques importantes, un nombre de points est attribué. Les framework totalisent les points des caractéristiques qu'ils satisfont:

- *Object oriented*: 20 points.
- *HTTP Post Support*: 10 points.
- *Documentation*: 5 points.
- *Developers / activity*: 5 points.

### Sémantique

- *Major browsers* relates to IE6, Mozilla/5.0 et Safari 1.3.
- This underlining style shows an advantage in our integration context.  
This underlining style shows a drawback in out integration context.

## AJAX frameworks overview

	Nom	Version	Browser compatibility	Object oriented [20]	Server-side language support	JS Proxy for backend objects	JS auto-generation	SOAP support	HTTP POST Support [10]	DOM manipulation suport	Visual effects	Documentation [5]	Developers / activity [5]	Miscellaneous	Licence
Pur JS	Prototype [40]	<a href="#">1.3.1</a>	<a href="#">Major browsers</a>	<a href="#">Yes</a>	-	-	-	No	<a href="#">Yes</a>	<a href="#">Minimal (innerHTML updater)</a>	<a href="#">No</a>	<a href="#">Unofficial contributions: API reference, examples.</a>	<a href="#">Team Ruby On Rails.</a>	<a href="#">Contributed and used by Ruby On Rails team.</a> <a href="#">Used by visual effects scripts (Rico, script.aculo.us, Behavior).</a> <a href="#">Renowned for it's well thought and well written standard-compliant code.</a> <a href="#">Unit-tested.</a>	MIT
	Script.aculo.us [40]	<a href="#">1.5 pre4</a>	<a href="#">Major browser</a>	<a href="#">Yes</a>	-	-	-	No	<a href="#">Yes</a>	Yes	<a href="#">Yes</a>	<a href="#">WikiDoc, examples, forum.</a>	<a href="#">Much activity</a>	<a href="#">Uses the prototype.js framework with visual effets addition.</a> <a href="#">Unit tested.</a>	MIT
	Open-Rico [30]	<a href="#">1.1 beta2</a>	<a href="#">IE5.5+</a> <a href="#">Firefox 1.0x/Win</a> <a href="#">Camino/Mac</a> <a href="#">Firefox 1.0x/Mac</a>	<a href="#">Yes</a>	-	-	-	No	<a href="#">Yes</a>	Yes	<a href="#">Yes</a>	<a href="#">Limited: 2 pdf and 1 blog post.</a> <a href="#">Contributed tutorial.</a>	<a href="#">Unknown</a>	<a href="#">Uses the prototype.js framework with visual effets addition.</a> <a href="#">Based on an earlier proprietary framework: Rico.</a>	Apache 2.0
	Ajax framework project [25]	<a href="#">1.2</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	-	-	-	Yes (Not working yet)	?	<a href="#">No</a>	<a href="#">No</a>	<a href="#">Tiny</a>	<a href="#">1 / 99.17%</a>	<a href="#">Error handing through the exception propagation mechanism.</a>	GPL
	DOJO [25]	<a href="#">0.1.0</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	-	-	-	No	<a href="#">No</a>	Yes	<a href="#">Yes</a>	<a href="#">API Documentation</a>	<a href="#">Unknown</a>	<a href="#">Browser URL manipulation support.</a>	Academic Free License v2.1
	Sarissa [15]	<a href="#">0.9.6.1</a>	<a href="#">Major browsers</a> <a href="#">IE&amp;MSXML3.0+</a> <a href="#">Konqueror (KDE 3.3+ for sure</a> <a href="#">Opera</a>	<a href="#">No</a>	-	-	-	No	<a href="#">Yes</a>	Yes	<a href="#">No</a>	<a href="#">JSDoc documentation</a>	<a href="#">2 / 99.67%</a>	<a href="#">Active since February 2003.</a> <a href="#">Unit-tested.</a> <a href="#">Developed with Apache Ant.</a>	GPL
HTMLHTTP Request [20]	<a href="#">1.0 beta2</a>	<a href="#">Major browsers</a> <a href="#">IE5.5/Win</a> <a href="#">IE5/Win+Mac</a> <a href="#">IE4/Win</a> <a href="#">Opera7/Win</a>	<a href="#">Yes</a>	-	-	-	No	<a href="#">No</a>	Yes	<a href="#">No</a>	<a href="#">None</a>	<a href="#">Unknown</a>	<a href="#">IFRAME-based transport layer for excellent cross-browser compatibility.</a>	LGPL 2.1	
Server-side	CPAINT [40]	<a href="#">2.0.1</a>	<a href="#">Major browsers</a>	<a href="#">Yes</a>	PHP, ASP	<a href="#">Yes</a>	No	No	<a href="#">Yes</a>	Yes	<a href="#">No</a>	<a href="#">Complete</a>	<a href="#">2 / 99.94%</a>	<a href="#">Supporte tous les encodages texte internationaux (UTF8 par défaut).</a> <a href="#">Supports internationalization</a>	GPL & LGPL
	PEAR:HTML:Ajax [25]	<a href="#">0.1.4 alpha</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	PHP	<a href="#">Yes</a>	?	No	?	<a href="#">No</a>	<a href="#">No</a>	<a href="#">PEAR Documentation</a>	<a href="#">2 / ?</a>	<a href="#">JSON and NULL serialization models.</a> <a href="#">Real object oriented proxy.</a>	PHP
	AjaxAC [25]	<a href="#">0.4.4</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	PHP	<a href="#">Yes</a>	Yes	No	?	Yes	<a href="#">No</a>	<a href="#">Tutorial</a>	<a href="#">1 / 98.26%</a>	<a href="#">Not easy to implement. The server-side logic to be exported has to be encapsulated in a class that has to extend an AjaxAC API class.</a> <a href="#">HTML widgets have to be attached inside the PHP code.</a>	Apache License v2.0
	JPSpan [25]	<a href="#">0.4.3 beta</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	PHP	<a href="#">Yes</a>	Yes	No	<a href="#">No</a>	Yes	<a href="#">No</a>	<a href="#">None</a>	<a href="#">2 / 98.45%</a>	<a href="#">Unit-tested.</a> <a href="#">Developed by a SitePoint coworker.</a>	PHP License
	NAJAX [30]	<a href="#">0.3.0.0 rc1</a>	<a href="#">Unknown</a>	<a href="#">Yes</a>	PHP	<a href="#">Yes</a>	Yes	No	?	<a href="#">No</a>	<a href="#">No</a>	<a href="#">PHPDocumentor</a>	<a href="#">1 / 99.12%</a>	<a href="#">Real object oriented proxy.</a>	PHP License
	XAJAX [25]	<a href="#">0.1 beta4.</a>	<a href="#">IE 6</a> <a href="#">Firefox</a> <a href="#">Safari</a>	<a href="#">Yes</a>	PHP	<a href="#">Yes</a>	Yes	No	<a href="#">No</a>	Yes	<a href="#">No</a>	<a href="#">Tutorial, examples</a>	<a href="#">1 / 13.24%</a>	<a href="#">DOM behavior described in PHP Code.</a> <a href="#">Smarty templates enabled.</a>	LGPL
SAJAX [10]	<a href="#">0.1</a>	<a href="#">Unknown</a>	<a href="#">No</a>	ASP, ColdFusion, Io, Lua, Perl, PHP, Python, Ruby	<a href="#">Yes</a>	Yes	No	<a href="#">Yes</a>	<a href="#">No</a>	<a href="#">No</a>	<a href="#">Examples</a>	?	<a href="#">Développé par un collaborateur de SitePoint.</a>	BSD	

## 7. Conclusions

### Frameworks retenus

Dans l'ordre, les frameworks retenus sont:

1. **Prototype.js**: abstraction de l'objet XMLHttpRequest coté client web
2. **CPAINT**: abstraction de l'objet XMLHttpRequest et export de méthodes server-side (PHP)
3. **Script.aculo.us**: abstraction de l'objet XMLHttpRequest, manipulation du DOM et effets visuels

Ces frameworks sont orientés objet, supportent les requêtes HTTP POST et fournissent une documentation suffisante. De plus, leur conception est bonne et leur communauté importante.

### Problèmes à considérer

#### La jeunesse des frameworks

Les frameworks étant jeunes, il se peut que celui choisi soit abandonné. Pour éviter le recodage Javascript de chaque appel asynchrone, créer une *pseudo-couche d'abstraction* supplémentaire est peut-être une solution.

Dans ce cas de figure, on effectue alors les appels asynchrones Javascript via une méthode de la *pseudo-couche d'abstraction*, qui elle s'occupe d'appeler les bonnes méthodes du framework choisi. On peut alors idéalement changer de framework en modifiant un minimum de code, c'est à dire les méthodes wrapper de notre pseudo-couche d'abstraction uniquement.

Suite aux discussions avec le client Yves Bolognini, cette solution est bonne et viable pour autant qu'elle n'apporte pas plus de lourdeur que de simplicité. La pseudo-couche d'abstraction – ou wrapper de framework – devra donc être la plus simple possible.

#### Gestionnaire d'événements Javascript

L'utilisation du modèle de gestion des événements permet d'attacher un événement à un élément HTML sans utiliser les attributs onClick, onChange, onMouseOver, etc – ces attributs empêchent la validation de la page HTML.

Cependant, le modèle de gestion des événements n'est pas identique entre les différents clients web: [http://www.quirksmode.org/blog/archives/2005/08/addevent\\_consider.html](http://www.quirksmode.org/blog/archives/2005/08/addevent_consider.html)

Il faut donc être attentif lors de l'utilisation du gestionnaire d'événements.

Le framework *Behaviour*<sup>22</sup> semble résoudre le problème en prenant en charge l'attachement des événements. Behaviour se concentre sur la fonction d'attachement d'événements et n'intègre pas les fonctions d'effets visuels Javascript – comme le drag, drop, le fading, etc. – que nous voulons éviter.

### Objectifs pour le prototype CartoWeb

Selon les discussions avec le client Yves Bolognini, chef de projet et développeur, les objectifs pour le prototype AJAX de CartoWeb sont les suivants.

#### Minimiser les impacts sur l'existant

Idéalement, les impacts de l'implémentation d'AJAX seront nuls. C'est à dire que le code développé jusqu'à aujourd'hui (p.ex. les plugins) devra pouvoir fonctionner sans modification – éventuellement avec ajout de wrappers.

Les clients commerciaux devront pouvoir intégrer la version AJAX de CartoWeb avec un minimum de modifications dans leur environnement (plugins, développements annexes, etc).

#### Abandon d'échanges SOAP asynchrone

L'architecture CartoWeb actuelle ne permet pas l'échange de messages SOAP asynchrones entre le client web et le cartoclient sans modification majeure du cartoclient. De plus, les frameworks AJAX supportent encore mal l'échange de messages SOAP et l'utilisation d'XML solutionne le problème.

La communication SOAP entre le client web et le cartoclient est donc abandonnée au profit de la création d'un wrapper sur le cartoclient et d'un échange de messages au format XML.

<sup>22</sup> <http://bennolan.com/behaviour/>, licence BSD.

### Création de wrapper sur le cartoclient

CartoWeb est conçu pour fournir des pages HTML monolithiques, contenant carte, tableaux de résultats pour recherches et données attributaires, panneaux d'outils, barre d'échelle, etc.

Pour permettre l'utilisation d'AJAX, la partie cartoclient doit être modifiée pour fournir des fragments de page HTML à actualiser de manière asynchrone.

Pour ce faire, un wrapper doit être ajouté sur le cartoclient pour générer ces fragments de page.

Ce wrapper doit être conçu pour être extensible, autant du côté cartoclient que Javascript, pour permettre l'ajout de fonctionnalités et plugins dans CartoWeb de manière aisée.

### Approche en blocs et dépendances

Dans un souci de modularité, chaque élément de l'UI CartoWeb sera considérée comme un bloc (carte, panneau de sélection des thèmes, résultats de requêtes sur la carte, etc).

Cette modularité nécessite l'introduction du concept de blocs et de dépendance. Un bloc est un élément de l'UI CartoWeb. La dépendance désigne le fait que l'actualisation d'un bloc entraîne l'actualisation si nécessaire d'un ou plusieurs autres blocs<sup>23</sup>.

L'objectif de cette approche est de permettre une hiérarchie arbitraire de dépendances entre les blocs, pour maximiser la modularité de la couche AJAX.

### Blocs de CartoWeb à actualiser de manière asynchrone

Pour le prototype, les blocs et dépendances suivantes devront être actualisables de manière asynchrone. Dans l'ordre de priorité:

- Carte et barre d'échelle
- Sélection des couches (ou thèmes)
- Affichage des résultats de requêtes

### Blocs facultatifs pour le prototype

Tous les *GUI providers*, c'est à dire les plugins générant de l'HTML – actuellement insérés dans le template *Smarty* principal.

### Compatibilité des clients web

La version AJAX de CartoWeb doit être supportée par les clients web suivants:

- IE6
- Mozilla 5.0 (Firefox au minimum)
- Safari 1.3

### Autres

CartoWeb est actuellement à sa version 3. C'est celle-ci qui sera utilisée pour la réalisation du prototype.

---

<sup>23</sup> Par exemple, la modification du niveau de zoom entraîne l'actualisation du bloc carte, qui elle-même entraîne une actualisation du bloc barre d'échelle et éventuellement du bloc panneau de sélection des couches (ou thèmes). Typiquement, un zoom supérieur à un niveau donné ne permet plus l'affichage de la couche raster vue aérienne: un clic sur la carte avec l'outil de zoom provoque sa réactualisation et entraîne celle de la barre d'échelle et du panneau de sélection des couches si nécessaire. Le bloc sélection des couches est donc – entre autres – dépendant du bloc carte.

## 8. Sources

### Liens internet

<http://www.adaptivepath.com/publications/essays/archives/000385.php>

[http://www.openweb.eu.org/articles/objet\\_xmlhttprequest/.](http://www.openweb.eu.org/articles/objet_xmlhttprequest/)

<http://www.javarss.com/ajax/j2ee-ajax.html>

<http://hinchcliffe.org/archive/2005/08/18/1675.aspx>

<http://alexbosworth.backpackit.com/pub/67688>

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ietechcol/dnwebgen/ie\\_leak\\_patterns.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ietechcol/dnwebgen/ie_leak_patterns.asp)

[http://www.quirksmode.org/blog/archives/2005/08/addevent\\_consider.html](http://www.quirksmode.org/blog/archives/2005/08/addevent_consider.html)

[http://markpasc.org/weblog/2005/05/28/landmarker\\_or\\_so\\_you\\_d\\_like\\_to\\_make\\_an\\_ajax\\_map](http://markpasc.org/weblog/2005/05/28/landmarker_or_so_you_d_like_to_make_an_ajax_map)

<http://map.google.com>

<http://virtualearth.msn.com/>

<http://map.search.ch>

<http://ka-map.maptools.org/>

<http://www.xml.com/pub/a/2005/08/10/ka-map.html>

<http://geoservices.cgdi.ca/mapbuilder/demo/data/context/atlasWorld.xml>

<http://geoservices.cgdi.ca/mapbuilder/demo/simple/simpleConfig.xml>

<http://geoservices.cgdi.ca/mapbuilder/userGuide?page=start/start>

<http://geoservices.cgdi.ca/mapbuilder/demo/complete/>

<http://geoservices.cgdi.ca/mapbuilder/>

<http://en.wikipedia.org/wiki/MVC>

<http://java.sun.com/blueprints/patterns/MVC-detailed.html>

<http://geoservices.cgdi.ca/mapbuilder/userGuide?page=start/start>

<http://www.ajaxmatters.com/r/resources?id=17>

[http://sourceforge.net/search/?type\\_of\\_search=soft&forum\\_id=0&group\\_id=0&atid=0&words=ajax+AND+framework](http://sourceforge.net/search/?type_of_search=soft&forum_id=0&group_id=0&atid=0&words=ajax+AND+framework)

[http://www.ajaxpatterns.org/Ajax\\_Frameworks](http://www.ajaxpatterns.org/Ajax_Frameworks)

<http://wiki.osafoundation.org/bin/view/Projects/AjaxLibraries>

<http://www.onlamp.com/pub/a/onlamp/2005/05/19/xmlhttprequest.html>

<http://www.eweek.com/article2/0,1895,1842333,00.asp>

### Bibliographie

*SitePoint DHTML Utopia: Modern Web Design Using Javascript & DOM*, Stuart Langridge, 2005.

*Designing With Web Standards*, by Jeffrey Zeldman, New Riders Press, 2003.

## 9. Déclaration d'honnêteté

Je soussigné déclare avoir écrit seul le présent rapport et ne pas avoir utilisé d'autres sources que celles indiquées.

**Lieu, date**

**Signature**

---

## 10. Annexes

### Implications pour CartoWeb

Les paragraphes concernant CartoWeb énoncés au fil du document sont récapitulés ci-dessous. Chacun est suivi du numéro de page y relatif.

En caractères distincts, ce document présente les aspects relatifs à CartoWeb, au fil des éléments analysés. 2

Concernant CartoWeb, les messages asynchrones échangés entre le serveur et le client ne seront vraisemblablement pas de type SOAP. Selon de team de développement CartoWeb, cette alternative est peu intéressante. En effet, la communication SOAP de CartoWeb est implémentée pour être utilisée de manière interne au serveur (entre le cartoclient et le cartoserver); le front-end de CartoWeb ne permet donc pas de communiquer les informations finales, utiles au client web. De plus, la jeunesse d'AJAX fait que son support SOAP n'est pas encore fiable. 4

Concernant l'implémentation prototype dans CartoWeb, le code Javascript sera réduit au maximum. Idéalement, il sera principalement responsable d'effectuer les requêtes asynchrones, puis recevoir et injecter du code HTML directement dans les éléments du DOM. Il pourra ainsi viser à être exempt de bugs, compatible avec les clients web principaux et le plus facilement maintenable possible. Les problèmes techniques et d'ergonomie cités ci-dessous devront dans la mesure du possible être pris en compte. 4

Pour le prototype AJAX CartoWeb, il me semble évident que créer un API orienté objet est une bonne idée: cela offre un jeu de méthodes simples à utiliser en plus des avantages d'une couche d'abstraction et de l'orienté objet. L'API Javascript de CartoWeb s'appuierait sur d'autres frameworks de bas niveau: abstraction d'HttpRequest et éventuellement modification DOM (par exemple avec le framework Behaviour). 7

Idéalement, la version AJAX de CartoWeb doit suivre cette règle et ne doit impliquer aucune modification du noyau du framework ou des plugins. Pratiquement, elle visera à les minimiser. 10

Suite aux discussions avec le team de développement, cette solution est la plus viable. Il est envisagé de créer une couche wrapper sur le cartoclient pour retourner les données à mettre à jour. 13

Concernant CartoWeb, il semble judicieux d'utiliser un framework dédié uniquement à l'abstraction de l'objet HttpRequest pour minimiser les bugs et maximiser la compatibilité. 14

Il me semble important de rester simple dans cette phase d'implémentation prototype pour éviter les effets de bord dus à des API trop complexes. De plus, une base utilisateur et documentation suffisante promettent de bonnes sources d'information et la viabilité du framework. 14

Cependant, si l'architecture de CartoWeb permet de récupérer dans une variable le résultat du parsing d'un seul template – et ceci de manière simple, il peut être intéressant de se tourner vers un framework server-side permettant l'export de méthodes PHP. 14

### Fonctionnement de Googlemaps

#### Fonctionnement

Googlemaps a développé son propre API Javascript: <http://www.google.com/apis/maps/documentation/>.

Une application s'écrit donc en Javascript et en HTML - en utilisant l'API qui popule le contenu des éléments HTML.

Cet API est complet, orienté objet et comprend un jeu de fonctions de haut niveau – comme le constructeur *GMap()*. Il se charge des appels asynchrones et de modifier le DOM. Il intègre donc une couche d'abstraction de l'objet HttpRequest et les fonctions de modification du DOM.

On peut alors générer facilement une carte en 3 lignes de code Javascript:

```
<script type="text/javascript">
//
  if (GBrowserIsCompatible()) {
    var map = new GMap(document.getElementById("map"));
    map.centerAndZoom(new GPoint(-122.141944, 37.441944), 4);
  }
//]]&gt;
&lt;/script&gt;</pre>
</div>
<div data-bbox="128 187 932 219" data-label="Text">
<p>Ceci sous-entend que la page HTML contient un <code>&lt;div&gt;</code> dont l'attribut <code>id</code> vaut <code>map</code>. Une carte draggable appartient donc dans l'élément HTML désigné, centrée en -122.141944, 37.441944 (long, lat).</p>
</div>
<div data-bbox="128 224 922 271" data-label="Text">
<p>En utilisant les fonctions de l'API, On peut ensuite ajouter les outils de navigation (zoom, recentrage), des POI, des pop-ups, attacher des <i>event listeners</i> à n'importe quel élément – on peut ainsi faire exécuter son propre code Javascript, ce qui ouvre une multitude de possibilités.</p>
</div>
<div data-bbox="128 275 919 323" data-label="Text">
<p>On peut créer ainsi une application cartographique web en utilisant l'API Javascript pour décrire un fonctionnement personnalisé. Il est facile de créer des POI et leur associer des pop-ups contenant des informations provenant d'une autre source.</p>
</div>
<div data-bbox="128 327 915 375" data-label="Text">
<p>L'inconvénient est qu'une grande quantité de POI peuvent ralentir l'application, puisqu'ils sont gérés par le client web et consomment ainsi de la puissance de calcul coté client. Googlemaps est donc fait pour gérer un nombre limité de couches.</p>
</div>
<div data-bbox="128 380 932 427" data-label="Text">
<p>L'astucieuse fonction <code>GBrowserIsCompatible()</code> permet un contrôle des fonctionnalités du browser – elle effectue un test de l'existence des objets et méthodes nécessaires. Il est ainsi facile d'afficher un message d'avertissement en cas d'incompatibilité du client<sup>24</sup>.</p>
</div>
<div data-bbox="128 432 888 463" data-label="Text">
<p>On remarque également l'utilisation du tag <code>&lt;![CDATA[ ... ]]&gt;</code>, en plus de la mise en commentaire commentaire.</p>
</div>
<div data-bbox="128 469 925 486" data-label="Section-Header">
<h3>Googlemaps utilise VML (Vector language markup) pour dessiner certains overlays overlays</h3>
</div>
<div data-bbox="128 485 927 563" data-label="Text">
<p>« We recommend that you use standards-compliant XHTML on pages that contain maps. When browsers see the XHTML DOCTYPE at the top of the page, they execute the page in "standards compliant mode," which makes layout and behaviors much more predictable across browsers. Likewise, if you include Polylines on your map, you need to include the VML namespace in your XHTML document for IE browsers. Your HTML document should begin like this: »<sup>25</sup></p>
</div>
<div data-bbox="93 568 886 725" data-label="Text">
<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"&gt;
&lt;html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml"&gt;
  &lt;head&gt;
    &lt;style type="text/css"&gt;
      v\:* {
        behavior:url(#default#VML);
      }
    &lt;/style&gt;
    &lt;script src="http://maps.google.com/maps?file=api&amp;v=1&amp;key=abcdefg" type="text/javascript"&gt;
    &lt;/script&gt;
  &lt;/head&gt;</pre>
</div>
<div data-bbox="91 734 554 756" data-label="Section-Header">
<h3>Exemple de fonctionnement de Ka-Map<sup>26</sup></h3>
</div>
<div data-bbox="128 756 919 787" data-label="Text">
<p>Cet exemple illustre le rafraîchissement de la barre d'échelle dans le framework Ka-Map. Il est tiré de l'exemple inclu dans l'archive Ka-Map.</p>
</div>
<div data-bbox="91 849 935 888" data-label="Footnote">
<p>24 A noter qu'il est conseillé d'effectuer un test des fonctionnalités (existence d'un objet) du client plutôt que de tester le type client (mozilla, ie, etc) – Source: <i>SitePoint DHTML Utopia: Modern Web Design Using Javascript &amp; DOM</i>, Stuart Langridge, p. 75, Detectung browser features.</p>
</div>
<div data-bbox="91 889 693 915" data-label="Footnote">
<p>25 Source: <a href="http://www.google.com/apis/maps/documentation/#Browser_Compatibility">http://www.google.com/apis/maps/documentation/#Browser_Compatibility</a> Workshop: <a href="http://msdn.microsoft.com/workshop/author/VML/ref/appendix.asp">http://msdn.microsoft.com/workshop/author/VML/ref/appendix.asp</a></p>
</div>
<div data-bbox="91 915 413 932" data-label="Footnote">
<p>26 Tiré de l'exemple fourni avec le framework</p>
</div>
<div data-bbox="91 945 282 962" data-label="Page-Footer">Damien Corpataux, 31IT</div>
<div data-bbox="821 945 939 962" data-label="Page-Footer">Page 23 / 26</div>
```

Lorsqu'on clique sur le bouton de zoom (*in* ou *out*), la barre d'échelle est rafraîchie. Pour ce faire, un script PHP est appelé de manière asynchrone par Javascript. Ce script renvoie l'URL de la barre d'échelle rafraîchie. Cet URL est récupéré sous forme de texte simple par Javascript qui l'injecte dans la valeur de l'attribut `src` du tag HTML `img`.

### Code HTML

```
<div id="scalebar"></div>
```

L'identifiant de l'élément `img` – affichant la barre d'échelle – vaut `scalebarImg`. Ceci permet à la fonction `setScalebar()` d'accéder simplement à l'élément pour modifier le contenu de son attribut `src`.

### Event handler sur le bouton de zoom (l'échelle change lors d'un zoom in/out)

```
<input type="button" name="zoomin" value="Zoom In" onclick="zoomIn()">
```

L'événement `onClick` est attaché à l'élément `input` du DOM de manière non conforme aux standards (la page ne validera pas). Le bouton de zoom exécute la fonction Javascript `zoomIn()` lors d'un click.

### Code Javascript pour le zoom in (similaire au zoom out)

```
function zoomIn()
{
    if (nCurrentScale < aScales.length - 1)
    {
        var nZoomFactor = aScales[nCurrentScale]/aScales[nCurrentScale+1];
        nCurrentScale = nCurrentScale + 1;
        nScale = aScales[nCurrentScale];
        initializeLayers(nZoomFactor);
        var s = getRawObject('scale');
        s.innerHTML = nScale;
        checkLegend();
        checkScalebar();
    }
    updateButtons();
}
```

La fonction `zoomIn()` appelle la fonction `checkScalebar()`.

### Code Javascript de la fonction checkScalebar()

```
function checkScalebar()
{
    call('scalebar.php?map='+currentMap+'&scale='+nScale, setScalebar);
}
```

Cette fonction fait un appel asynchrone à un URL – correspondant à un script PHP. Celui-ci retourne du texte simple (l'URL de la nouvelle barre d'échelle). Cette valeur de retour est passée en paramètre à la fonction `setScalebar()`.

### Code PHP de scalebar.php, appelé de manière asynchrone

```
<?php
include_once('config.php');
if (!extension_loaded('MapScript'))
{
    dl( $szPHPMapScriptModule );
}

$oMap = ms_newMapObj($szMapFile);
$oPoint = ms_newPointObj( );
$oPoint->setXY($oMap->width/2, $oMap->height/2 );
$oMap->zoomScale( $_REQUEST['scale'], $oPoint, $oMap->width, $oMap->height, $oMap->extent );
$oImg = $oMap->drawScalebar();

$szURL = $oImg->saveWebImage();
echo $szURL;
?>
```

Ce script sert de « service » : il est appelé de manière asynchrone par le client web, génère une nouvelle barre d'échelle (via `MapScript`) et retourne son URL sous forme de texte simple.

### Code Javascript de la fonction setScalebar(string)

```
function setScalebar(szScalebarURL)
{
    var oImg = getRawObject( 'scalebarImg' );
    oImg.src = szScalebarURL;
}
```

Cette fonction Javascript est exécutée lors de la réception de la réponse – de l'appel asynchrone – et actualise l'attribut src de l'élément img.

## Exemple d'appel asynchrone POST avec prototype.js

### Qu'est-ce qu'une requête HTTP POST?

W3C HTTP Method Definition: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>.

HTTP Made really easy: <http://www.jmarshall.com/easy/http/>.

Une requête HTTP utilisant la méthode POST permet un nombre illimité de caractères pour la partie données. Ainsi, on peut passer un grand nombre de couples paramètre-valeur.

CartoWeb utilise cette méthode. Le framework choisi doit donc la supporter.

Un serveur web recevant une requête HTTP utilisant la méthode POST reçoit la chaîne de caractères suivante:

```
POST /path/script.cgi HTTP/1.0
From: frog@jmarshall.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

home=Cosby&favorite+flavor=flies
```

les couples paramètre-valeur sont encodée à la mode URL-encoded<sup>27</sup>.

### Voici le code Javascript pour effectuer un appel asynchrone utilisant la méthode HTTP POST

```
function call() {
    var url = 'http://127.0.0.1/Ajax_Test/Prototype/postrequest/ajaxService.php'
    var postBody = 's=My string&i=My int';
    var myAjax = new Ajax.Request( url, {method: 'post', postBody: postBody,
        onComplete: showResponse} );
}
```

Prototype.js s'occupe de placer le query string dans le corps de la requête POST. Les caractères spéciaux sont automatiquement convertis en entités HTML (ici, l'espace devient un +).

```
function showResponse(originalRequest)
{
    // Show the returned data in an alert box
    alert.innerHTML = originalRequest.responseText;
}
```

Le contenu texte de la réponse (c'est à dire le contenu de la page [http://127.0.0.1/Ajax\\_Test/Prototype/postrequest/ajaxService.php?s=My+string&i=My+int](http://127.0.0.1/Ajax_Test/Prototype/postrequest/ajaxService.php?s=My+string&i=My+int)) est affiché dans une boîte de dialogue.

<sup>27</sup> [http://www.jmarshall.com/easy/http/http\\_footnotes.html#urlencoding](http://www.jmarshall.com/easy/http/http_footnotes.html#urlencoding)

**Pour retourner du contenu XML...**

Lorsqu'un fichier .xml est requis par les méthodes GET ou POST, la réponse de la part du serveur est la suivante:

```
HTTP/1.1 200 OK
Date: Tue, 13 Sep 2005 13:41:03 GMT
Server: Apache/2.0.54 (Win32) PHP/5.0.4
Last-Modified: Tue, 13 Sep 2005 12:57:28 GMT
ETag: "a773140-1d7-c9429e00"
Accept-Ranges: bytes
Content-Length: 471
Connection: close
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8" ?>
<root>
  <parent />
</root>
```

On remarque que le serveur renvoie le *Content-Type* (le type contenu) dans les en-têtes HTTP. Ici, *application/xml* pour un fichier XML.

Lorsqu'un fichier d'un autre type est renvoyé, le *Content-Type* change. Pour un fichier .txt:

```
HTTP/1.1 200 OK
[... ]
Content-Type: text/plain
[... ]
```

Pour un fichier .txt, le type de contenu vaut *text/plain*. Prototype ne pourra utiliser cette réponse que sous forme de texte et ne pourra pas être parsée.

Pour que Prototype fonctionne correctement, le *Content-Type* des réponses aux appels asynchrones doit valoir *application/xml*. Il faut donc forcer cette valeur<sup>28</sup>.

---

28 En PHP, il faut ajouter la commande `header("Content-Type: application/xml");` avant toute sortie texte.