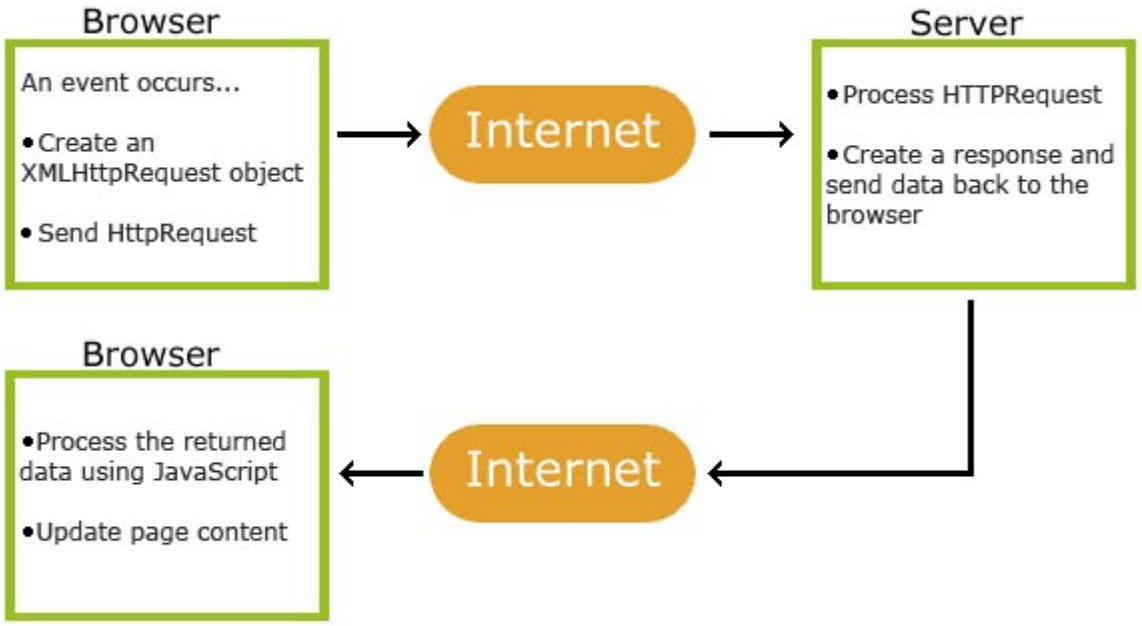
# AJAX

- Le terme AJAX a été introduit par Jesse James Garrett, le 18 février 2005
- AJAX n'est pas un langage de programmation
- AJAX est un concept de programmation Web reposant sur plusieurs technologies comme le JavaScript et le XML
- AJAX est une technique pour accéder aux serveurs Web depuis une page Web
- AJAX signifie Asynchronous JavaScript And XML.
- L'idée même d'AJAX est de faire communiquer une page Web avec un serveur Web sans occasionner le rechargement de la page.
- AJAX lit des données sur le serveur Web après qu'une page Web soit chargée
- AJAX met à jour une page Web sans avoir à la recharger
- AJAX envoie des données au serveur Web en arrière plan
- le principe d'Ajax est né en 2001 avec l'objet ActiveX **XMLHttp** de Internet Explorer. Les autres navigateurs n'étant pas compatible activeX, cet objet est resté longtemps inutilisé. Ensuite Mozilla créa **XMLHttpRequest** et fut suivis par les autres navigateurs (sauf IE évidement).

> *AJAX doit être utilisé pour charger/modifier de petites parties d'une page.*




- 1. un événement survient sur la page WEB (la page est chargée, un bouton est cliqué)
- 2. un objet **XMLHttpRequest** est créé par JavaScript

- 3. l'objet XMLHttpRequest envoie une requête au serveur WEB
- 4. Le serveur exécute la requête
- 5. Le serveur envoie la réponse à la page WEB
- 6. La réponse est lue par JavaScript dans le browser
- 7. L'action programmée (par exemple mise à jour d'une partie de la page WEB) est réalisée par JavaScript

> *Il existe plusieurs formats d'échange possible entre le serveur et la page Web:*

- Texte simple
- HTML
- XML
- JSON

> *limitations d'Ajax*

- l'objet **XMLHttpRequest** a des limites, la principale étant de ne pouvoir communiquer qu'avec le serveur courant
- On ne peut pas récupérer et lire du javascript se trouvant dans une page appelée par **XMLHttpRequest**. en fait c'est pas tout à fait juste, des astuces (déconseillées) vous permettent de lire le javascript en le plaçant dans des "onload" par exemple
- bien que l'objet **XMLHttpRequest** soit du javascript, il nécessite la réponse d'un serveur
- les données récupérées avec **XMLHttpRequest** reste en mémoire, donc **XMLHttpRequest** est un gros consommateur de RAM car une même page pourra plusieurs fois être chargée en mémoire.

Il faut attendre la disponibilité des données, et l'état est donné par l'attribut **readyState** de XMLHttpRequest.

> *Les états de readyState sont les suivants (seul le dernier est vraiment utile):*

- 0: non initialisé.

- 1: connexion établie.
- 2: requête reçue.
- 3: réponse en cours.
- 4: terminé.

L'objet XMLHttpRequest permet d'interagir avec le serveur, grâce à ses méthodes et ses attributs :

link:The XMLHttpRequest Object

> *Attributs :*

| | |
|---|---|
| **readyState** | le code d'état passe successivement de 0 à 4 qui signifie "prêt". |
| **status** | 200 est ok<br>404 si la page n'est pas trouvée. |
| **responseText** | contient les données chargées dans une chaîne de caractères. |
| **responseXml** | contient les données chargées sous forme XML, les méthodes de DOM servent à les extraire. |
| **onreadystatechange** | propriété activée par un évènement de changement d'état. On lui assigne une fonction. |

## *Méthodes :*

| | |
|---|---|
| **open**(mode, url, boolean) | mode: type de requête, GET ou POST<br>url: l'endroit ou trouver les données, un fichier avec son chemin sur le disque.<br>boolean: true (asynchrone) / false (synchrone).<br>en option on peut ajouter un login et un mot de passe. |
| **send**("chaine") | null pour une commande GET. |