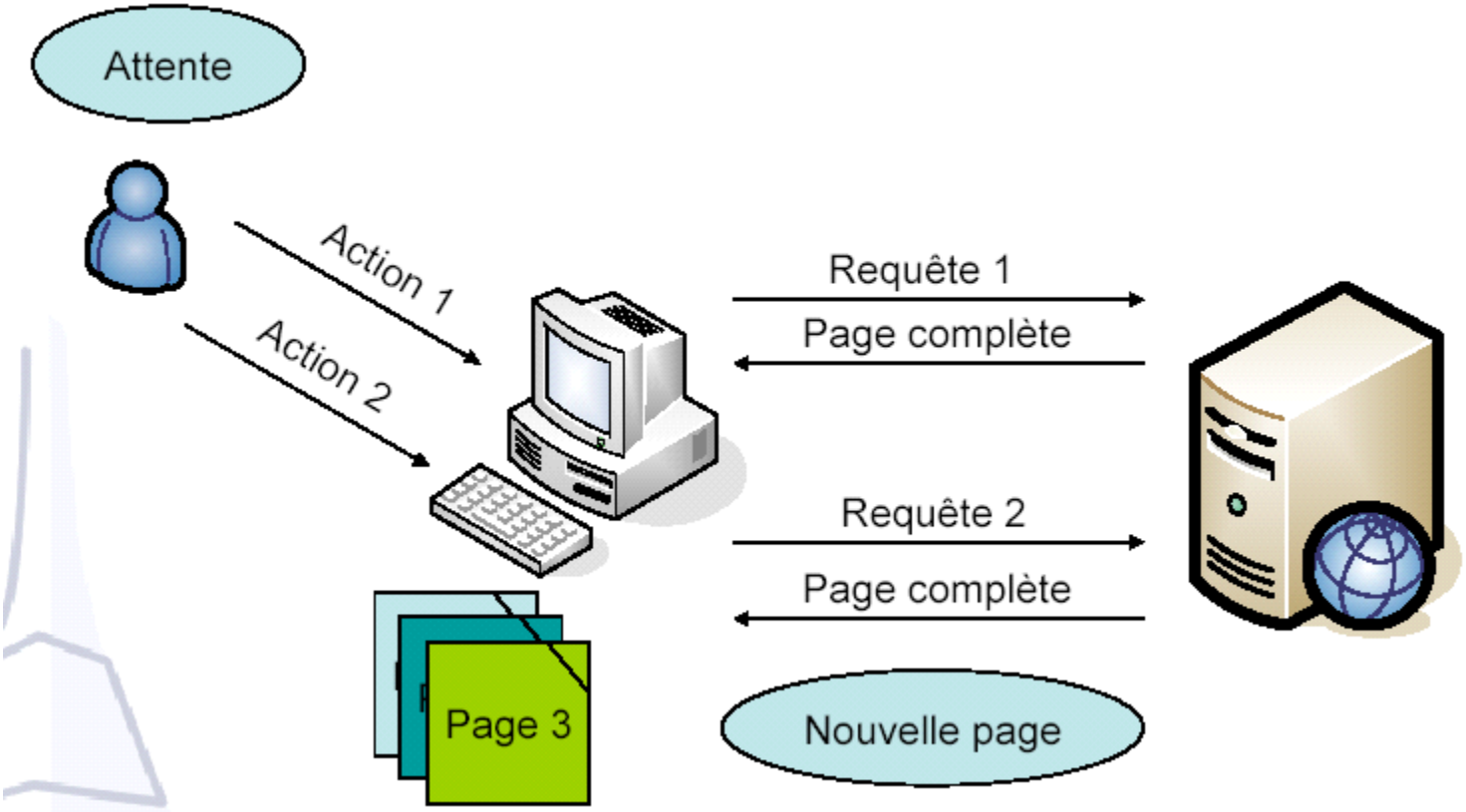

AJAX

AJAX:

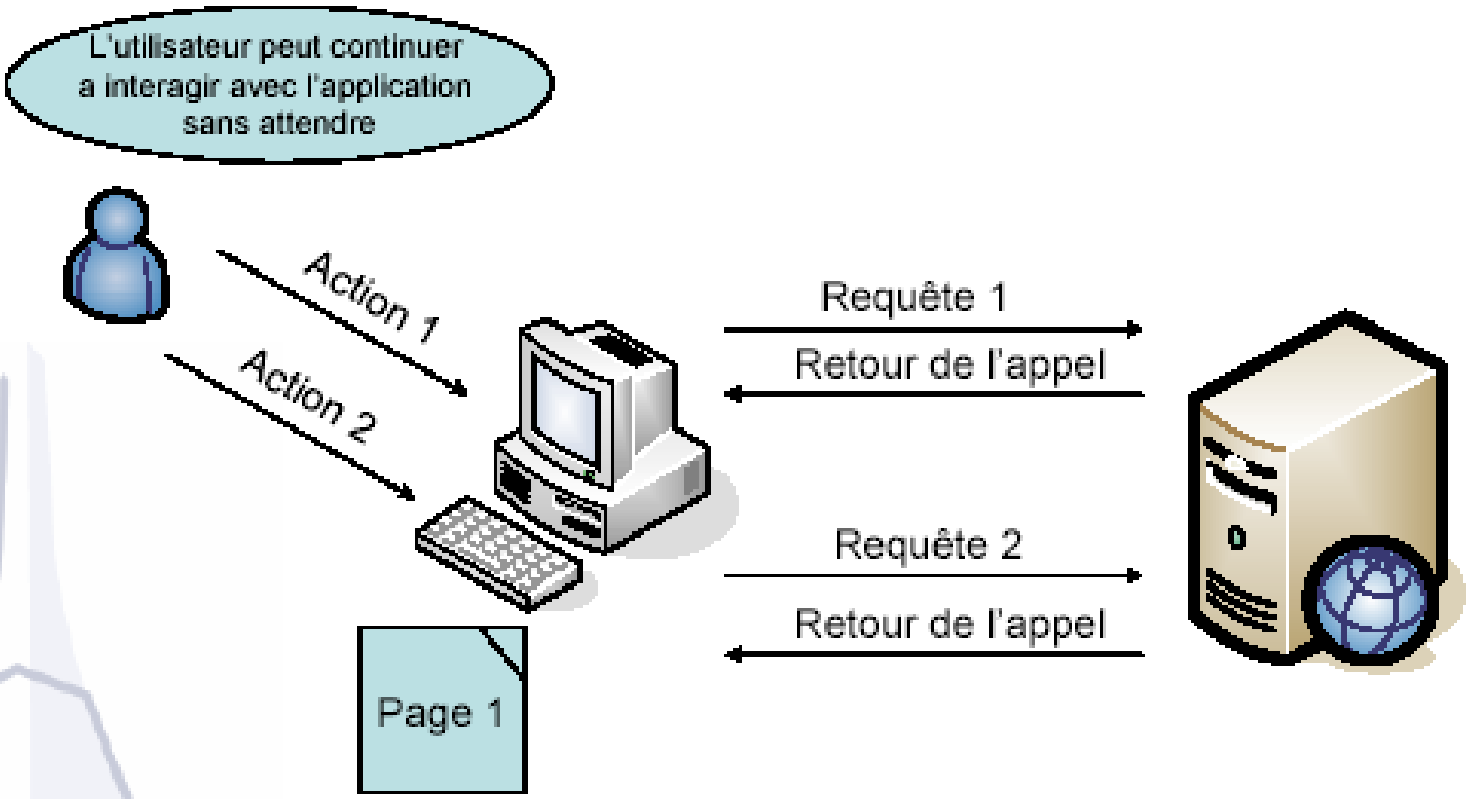
- **AJAX** (*Asynchronous Javascript And XML*) est une méthode de développement web basée sur l'utilisation d'un script Javascript pour effectuer des requêtes web à l'intérieur d'une page web sans recharger la page. AJAX rend plus interactifs les sites web et offre une meilleure ergonomie ainsi qu'une réactivité améliorée en permettant de modifier interactivement une partie de l'interface web seulement.

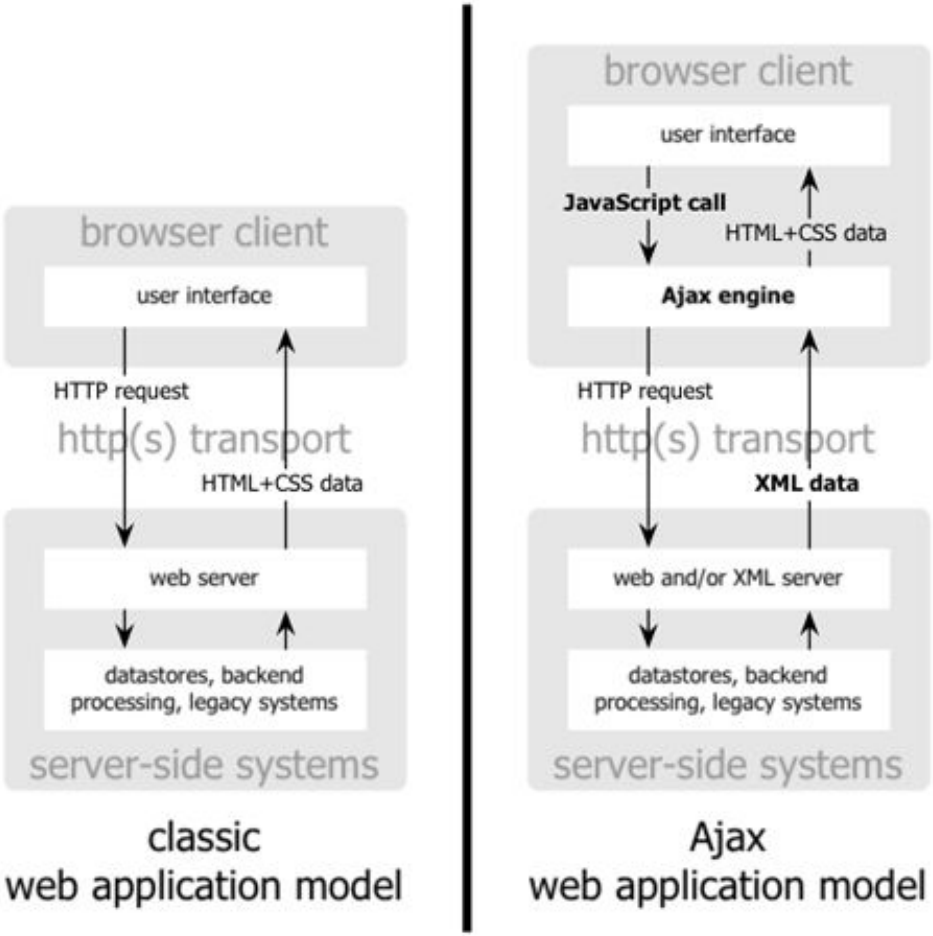
- En effet, le modèle web traditionnel est basé sur une suite de requêtes et de réponses successives, c'est-à-dire une navigation séquentielle de page web en page web. AJAX permet de ne modifier que la partie de la page web qui nécessite d'être mise à jour en créant une requête HTTP locale (XMLHttpRequest) et en modifiant tout ou partie de la page web en fonction du retour de la requête HTTP.

Web 1.0: Synchrone



Web 2.0: Asynchrone




browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model

- AJAX (Asynchronous Javascript XML) renvoit plus largement à une conception de site web reposant sur une architecture client/serveur qui consiste en effet à découper une application web de la façon suivante :

- une présentation utilisant XHTML et CSS ;
- la manipulation dynamique des pages à travers DOM ;
- la manipulation des données avec XML et XSLT ;
- l'échange des données (avec le serveur) de manière asynchrone avec XMLHttpRequest ;
- le tout étant coordonné avec du Javascript.

- Les éléments clés de cette architecture sont la séparation en couches distinctes des éléments du client à l'aide de technologies standardisées et les échanges asynchrones de données au format XML.

- l'objet non standard XMLHttpRequest permet de communiquer avec des scripts situés sur le serveur.
- L'objet permet d‘obtenir des informations sous différents formats (dont XML, HTML ou texte), tout cela sans recharger la page. C'est ce qui permet de mettre à jour certaines parties d'une page sur base d'évènements déclenchés par l'utilisateur.
- Les deux fonctionnalités combinées sont les possibilités de :
  - faire des requêtes vers le serveur sans avoir à recharger la page ;
  - analyser et travailler avec des documents réponse XML.

Mise en Œuvre

D'abord été introduite dans Internet Explorer sous la forme d'un objet ActiveX appelé XMLHTTP.

Par la suite, Mozilla, Safari et d'autres navigateurs ont suivi en implémentant une « classe » XMLHttpRequest qui fournit les mêmes méthodes et propriétés que l'objet ActiveX original de Microsoft.

1. Pour créer une instance (un objet) de la classe désirée fonctionnant sur plusieurs navigateurs, vous pouvez utiliser :

```
if (window.XMLHttpRequest)
    { // Mozilla, Safari, ...
      httpRequest = new XMLHttpRequest(); }
else
    if (window.ActiveXObject)
    { // IE
      httpRequest = new ActiveXObject("Microsoft.XMLHTTP"); }
```

2. Déterminer ce que vous ferez à la réception de la réponse du serveur.

- vous devez donc définir pour l'objet de requête HTTP la « méthode » (fonction JavaScript) qui effectuera le travail d'analyse de la réponse.
- => assignez à la propriété onreadystatechange de l'objet la fonction JavaScript :

  httpRequest.onreadystatechange = *nomDeLaFonction*;

- Par ailleurs, au lieu de donner un nom de fonction, vous pouvez utiliser la technique JavaScript de définition de fonctions au vol (une fonction anonyme), comme ceci :

```
httpRequest.onreadystatechange = function()
    { // instructions de traitement de la réponse };
```

3. lancer effectivement la requête. Il faut pour cela appeler les méthodes open() et send() de la classe de requête HTTP:

open(…,…,…)

- Le premier paramètre est le type de requête HTTP: GET, POST, HEAD ou toute autre méthode que vous voulez utiliser et qui soit gérée par le serveur.
- Le second paramètre est l'URL de la page que vous demandez. *Pour des raisons de sécurité, il n'est pas possible d'appeler des pages se situant sur un autre domaine.*
- Le troisième paramètre précise si la requête est asynchrone. Si TRUE, l'exécution de la fonction JavaScript se poursuivra en attendant l'arrivée de la réponse du serveur. C'est le A d'AJAX.

- Le paramètre de la méthode send() peut être n'importe quelle donnée que vous voulez envoyer au serveur en cas d'utilisation de la méthode POST. Les données doivent être sous la forme d'une chaîne de requête, comme :

  nom=valeur&autrenom=autrevaleur&ainsi=desuite

  (Notez que si vous voulez envoyer des données avec la méthode POST, vous devrez changer le type MIME de la requête à l'aide de la ligne suivante :

  httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

)

4. Gestion de la réponse du serveur :

Une fonction JavaScript « traite » la réponse du serveur:

```
httpRequest.onreadystatechange = nomDeLaFonction;
```

Que doit faire cette fonction ?

Tout d'abord, elle doit vérifier l'état de la requête. Si cet état a une valeur de 4, cela signifie que la réponse du serveur a été reçue dans son intégralité et qu'elle peut maintenant être traitée.

```
if (httpRequest.readyState == 4)
        { // tout va bien, la réponse a été reçue }
else { // pas encore prête }
```

Voici la liste complète des valeurs de readyState :

- 0 (non initialisée)
- 1 (en cours de chargement)
- 2 (chargée)
- 3 (en cours d'interaction)
- 4 (terminée)

- La seconde vérification concerne le code d'état de la réponse HTTP du serveur. Tous les codes possibles sont listés sur le site du W3C.

```
if (httpRequest.status == 200)
  { // parfait ! }
else
{ // il y a eu un problème avec la requête,
 // par exemple la réponse peut être un code 404
 // ou 500 (Erreur interne au serveur) }
```

5. Après avoir vérifié l'état de la requête et le code d'état HTTP de la réponse, on peut accéder aux données envoyées par le serveur:

- httpRequest.responseText — renvoie la réponse du serveur sous la forme d'une chaîne de texte
- httpRequest.responseXML — renvoie la réponse sous la forme d'un objet XMLDocument que vous pouvez parcourir à l'aide des fonctions DOM de JavaScript

```
<html>
  <head>
    <title>Enter the title of your HTML document here</title>
  </head>
  <body>
     <script type="text/javascript" language="javascript">
   function makeRequest(url) {
      var httpRequest = false;
      if (window.XMLHttpRequest) { // Mozilla, Safari,...
1.       httpRequest = new XMLHttpRequest();
         if (httpRequest.overrideMimeType) {
            httpRequest.overrideMimeType('text/xml');
            // Voir la note ci-dessous à propos de cette ligne
         }
      }
      else if (window.ActiveXObject) { // IE
         try { httpRequest = new ActiveXObject("Msxml2.XMLHTTP"); }
         catch (e) {
            try { httpRequest = new ActiveXObject("Microsoft.XMLHTTP");              }
            catch (e) {}
         }
      }
      if (!httpRequest) {
         alert('Abandon :( Impossible de créer une instance XMLHTTP');
         return false;
2.    }
3.    httpRequest.onreadystatechange = function() { alertContents(httpRequest); };
      httpRequest.open('GET', url, true);
      httpRequest.send(null);
   }
   function alertContents(httpRequest) {
4.
      if (httpRequest.readyState == 4) {
         if (httpRequest.status == 200) {
            alert(httpRequest.responseText);
         } else {
            alert('Un problème est survenu avec la requête.');   5.
         }
      }

   }
</script>
<span
   style="cursor: pointer; text-decoration: underline"
   onclick="makeRequest('testAj.html')">
      Effectuer une requête
</span>
  </body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
<pre id="msg"></pre>
<script type="text/javascript">
<!--
        requete = new XMLHttpRequest();
requete.onreadystatechange = function(){
            document.getElementById("msg").innerHTML += "Etat:"+ this.readyState ;
            }
requete.open("GET","DVD.xml",true);
requete.send(null);
        -->
</script>
</body>
</html>
```

Que va-t-il se passer au chargement de cette page par un navigateur ?

Etat:1Etat:2Etat:3Etat:4

Exemple 1: afficher dans une page web les 10 premières citations de citations.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
	<head>
	  <title>Enter the title of your XHTML document here</title>
	  <script type="text/javascript" language="javascript">
	  <!--
	    function Init(){
	        myXMLHTTPRequest = new XMLHttpRequest();
	        myXMLHTTPRequest.open("GET", "citations.xml", false);
	        myXMLHTTPRequest.send(null);
	        racine =  myXMLHTTPRequest.responseXML.documentElement ;
	        auteurs = racine.getElementsByTagName("texte");
	        d1 = document.getElementById('example');
	        for (i=0;i<10;i++) {
	                // attention au maj.min casse
	                 e = document.createElement("p");
	                 t = document.createTextNode(auteurs[i].firstChild.nodeValue);
	                 e.appendChild(t);
	                 d1.appendChild(e);
	        }
}
-->
</script>
	</head>
	<body>
	  <input type="button" value="go!" onclick="Init()" />
	  <div id="example" />
	</body>
</html>
```

go!

J'ai vu en mon temps cent artisans, cent laboureurs, plus sages et plus heureux que des recteurs de l'université.

L'essentiel est sans cesse menacé par l'insignifiant.

Quand les hommes ne peuvent changer les choses, ils changent les mots.

C'est en allant vers la mer que le fleuve reste fidèle à sa source.

Bien écouter, c'est presque répondre.

On fait la science avec des faits, comme on fait une maison avec des pierres : mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'es
maison.

Toutes choses sont dites déjà ; mais comme personne n'écoute, il faut toujours recommencer.

J'entends et j'oublie, Je vois et je me souviens, Je fais et je comprends.

Je ne cherche pas à connaître les réponses, je cherche à comprendre les questions.

Il n'y a rien de négatif dans le changement, si c'est dans la bonne direction.

```
<script type='text/JavaScript'>
        function getXhr(){}
        function go(namefile){
            var xhr = getXhr() ;
            xhr.onreadystatechange = function(){
            if(xhr.readyState == 4 && xhr.status == 200){
          objdiv = document.getElementById("objdiv");
           objul =objdiv.getElementsByTagName("ul");
          if (objul.length != 0) { objdiv.removeChild(objul[0]);}
              objXML = xhr.responseXML;
               libelles = objXML.getElementsByTagName("libelle");
               nbitems = libelles.length;
              valeurs = objXML.getElementsByTagName("valeur");
             objul = document.createElement("ul");
             for (i=0;i<nbitems;i++) {
              ex = document.createElement("li");
              texte = document.createTextNode(libelles[i].firstChild.nodeValue+":"+valeurs[i].firstChild.nodeValue) ;
              ex.appendChild(texte);
             objul.appendChild(ex);         }
            objdiv.appendChild(objul);   }}
         xhr.open("GET",namefile,true);
        xhr.send(null);
        }
    </script>
```

```
<body>
<form>
<select onchange="go(this.value)">
  <option value="obj1.xml">obj1</option>
  <option value="obj2.xml">obj2</option>
  <option value="obj3.xml">obj3</option>
  <option value="obj4.xml">obj4</option>
  <option value="obj5.xml">obj5</option>
</select>
</form>
<div id="objdiv"></div>
</body>
</html>
```

```
<itemize>
<item>
        <libelle>Nom</libelle>
        <valeur>Sinclair</valeur>
</item>
<item>
        <libelle>Prenom</libelle>
        <valeur>Brett</valeur>
</item>
<item>
        <libelle>email</libelle>
        <valeur>amicalement@votre.tv</valeur>
</item>
</itemize>
```

```
<itemize>
<item>
   <libelle>Marque</libelle>
   <valeur>Renault</valeur>
</item>
<item>
    <libelle>Modele</libelle>
    <valeur>Laguna</valeur>
</item>
<item>
     <libelle>Couleur</libelle>
     <valeur>Gris Metal</valeur>
</item>
</itemize>
```

JSON:Javascript Object Notation

Le XML est une **calamité** à parser en JavaScript
Le format JSON est beaucoup plus approprié
JSON (JavaScript Object Notation) est un format de données générique qui utilise la notation des objets JavaScript pour transmettre de l'information structurée

```
<?xml version="1.0" ?>
<playlist nom="MeshowRandom">
   <chanson>
      <titre>Best Improvisation Ever 2</titre>
      <auteur>David Meshow</auteur>
      <note>5</note>
   </chanson>
   <chanson>
      <titre>My Theory (Bonus)</titre>
      <auteur>David Meshow</auteur>
      <note>4</note>
   </chanson>
</playlist>
```

```
{
   "nom" : "MeshowRandom",

   "chansons" : [
      {
         "titre" : "Best Improvisation Ever 2",
         "auteur" : "David Meshow",
         "note" : 5
      },
      {
         "titre" : "My Theory",
         "auteur" : "David Meshow",
         "note" : 4
      }
   ]
}
```

```
var v = {
    "nom" : "MeshowRandom",

    "chansons" : [
        {
            "titre" : "Best Improvisation Ever 2",
            "auteur" : "David Meshow",
            "note" : 5
        },
        {
            "titre" : "My Theory",
            "auteur" : "David Meshow",
            "note" : 4
        }
    ]
};
alert(v.nom);
alert(v.chansons[1].titre);
```

La syntaxe de base

{ } exprime que le contenu englobé est un objet

ex : {objet json}

: séparation entre nom d'attribut et valeur

ex : {age:25}

“ ou ‘ délimiteur de valeur textuelle

ex : {name:"Idleman"}

, sépare un attribut d’un autre

ex : {name="Idleman",age:25}

[ ] exprime que le contenu englobé est un tableau

ex : {name="Idleman",age:25,hobby:
["informatique","musique","sport","sorties"]}

C’est tout pour la syntaxe! Il est maintenant possible de faire des combinaisons en englobant des objet dans d’autres

ex : {users:[{"name":"Idleman","age":25},{"name":"IdleGirl","age":20}]}

Côté Client:

JSON faisant partie de la norme JavaScript.
Le contenu d'un fichier JSON, ou la définition de données dans ce forat sont assignés à une variable, laquelle devient un objet du programme.

Côté serveur

Les fichiers au format JSON s'utilisent dans différents langages de programmation, notamment PHP et Java grâce à des parseurs qui permettent d'accéder au contenu, et éventuellement de le convertir en classes et attributs, dans ce langage.

Loading d'un flux Json

```
<html>
    <head>
      <title>Tutoriel Ajax (XHTML + JavaScript + XML)</title>
      <script type='text/JavaScript'>
            function getXhr(){}
            function go(){
                  var xhr = getXhr() ;
                  xhr.onreadystatechange = function(){
                  f(xhr.readyState == 4 && xhr.status == 200){
                        var lauteur = eval("("+xhr.responseText+")");
                        alert(lauteur.adresse.ville); }
                  }
                  xhr.open("GET","json.txt",true);
                  xhr.send(null);}
      </script>
    </head>
    <body>
      <input type='button' value='Dis quelque chose !' onclick='go()' />
    </body>
</html>
```

```
<SCRIPT language=javascript>
var a=1;
chaine="a = a + 10";
eval(chaine);
document.write("Valeur de a = "+a);
</SCRIPT>
```

```
JSON.parse(xhr.responseText);
```

```
{ "nom" :  "Giono",
  "prenom" : "Jean",
  "adresse" : {
            "numero" : 2,
            "rue" : "Lorraine",
            "ville" : "Hem",
             "code" : "59510"
          }
}
```