JAVASCRIPT

HarmWeb

Introduction

- Le Javascript est un langage "de script" "orienté objet" :
  - Initialement élaboré par Netscape en association avec Sun Microsystem.
  - Standardisé par un comité spécialisé, l'ECMA (European Computer Manufactures Association).

- Javascript permet :
  - De rendre dynamique un site internet développé en HTML :
    - Validation de formulaires, calculs, messages,
    - Modification de la page web,
    - Communication avec un serveur directement (AJAX)
  - De développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet.

Caractéristiques principales

- Le Javascript est :
  - Ecrit directement dans le document HTML
  - Un script encadré par des balises HTML
  - Exécuté chez le client (pas d'appel réseau)
  - Interprété (compilation à la volée)
- Supporté par la plupart des navigateurs web
- Syntaxe proche du C

Plan du cours

- Syntaxe : tests, boucles, fonctions
- Événements / Manipulation de page
- AJAX
- JSON

JAVASCRIPT SYNTAXE

LI288 – web et développement web

HTML et JavaScript

- Deux types d'insertion (comme CSS)
  - Directement dans le fichier HTML
  - Dans un fichier externe et inclus en HTML

- Utilisation de balises spécifiques :
  - <script type="text/javascript">…</script>

Insertion dans une page HTML

- Dans l'entête ou le corps de la page HTML
  - Le code s'exécute lors du chargement de la page

```
<html>
<body>
<script type="text/javascript">
  alert('bonjour');
</script>
</body>
</html>
```

Insertion dans une page HTML

- Fichier en format texte contenant uniquement du JS
- Permet de réutiliser les scripts dans plusieurs pages
- Requête supplémentaire vers le serveur
  - Bloque le rendu de la page

```
<html>
<head>
  <script type="text/javascript" src="fichier.js"></script>
</head>

<body>
  <input type="button" onclick="popup()" value="Clic">
</body>
</html>
```

Syntaxe

- Similaire à Java ou C

- Règles générales
  - On peut mettre des espaces n'importe où
  - (On sépare les commandes par des point-virgule ";")
  - Les réels sont notés avec un "." et pas une virgule ","
  - Commentaires : //… ou /*…*/
    - // ceci est un commentaire
    - /* ceci est aussi
      un commentaire */

Les variables

- Déclaration et affectation
  - Déclaration avec le mot clé "var"
  - Affectation avec le signe d'égalité (=)
- Remarques :
  - La déclaration est faite par défaut (si affectation sans déclaration préalable)
  - La lecture d'une variable non déclarée provoque une erreur
  - Une variable déclarée non affectée est de type undefined (indéfinie)

```
//Déclaration de i, de j et de k.
var i, j, k;
 //Affectation de i.
i = 1;
//Déclaration et affectation de prix.
var prix = 0;

//Déclaration et affectation d'un tableau
var car = ["a", "b", "c"];
```

Les types de variables

- Principaux types :
  - Chaînes
  - Nombre (entier ou décimaux) : 10^-308 >nombre < 10^308
  - 3 valeurs spéciales :
    - Positive Infinity ou +Infinity (valeur infini positive)
    - Negative Infinity ou –Infinity (valeur infinie négative)
    - Nan (Not a Number) en général le résultat d'une opération incorrecte
  - Boolean
    - true (vrai) et false (faux)
  - …

- Le type d'une variable dépend de son contenu
  - var maVariable = "Philippe"; // type chaîne
    maVariable =10; // type nombre (entier)

Portée des variables

- Globale :
  - Variable déclarée en début de script
  - Accessible à n'importe quel endroit du programme
- Locale :
  - Variable déclarée à l'intérieur d'une fonction
  - Accessible uniquement dans la fonction
- Pas de portée de type bloc

Boucles et conditions

- Boucle for : for (i=0; i<5; i++) {…}
- Boucle while : while (test) {…}
  - do {…} while (test)
- Conditions : if (test) {} else {}
  - Egalité : ==, !=
  - Inférieur, supérieur : =<, >=, >, <
  - Identique à : ===, !== (teste valeur et type)
    - ('1' == 1) // true
    - ('1' === 1) // false
  - Opérations logiques : &&, ||

Les fonctions

- Définition :
  - function maFonction(arg1,arg2) {instr;}
  - Pas de type dans la signature de la fonction
- Appel :
  - maFonction("12",13);

- Exemple : calcul de la fonction factoriel
  - Calcul récursif + conditionnelle (if)
- Mais aussi (à venir) :
  - Appelé sur un événement
  - Utilisation de document.getElementById
  - Utilisation de this.value

Fonction factorielle

```
<html>
<head>
<script type="text/javascript">
function fac(n){
  if (n < 2) {
    return 1;
  } else {
    return n * fac(n-1);
  }
}
</script>
</head>

<body>
<form>
  <input id="facArg“ type="text"
    onchange="result = fac(this.value); document.getElementById('facResult').value = result;" />
  <input id="facResult" type="text" />
</form>
</body>
</html>
```

Nombre variable d'arguments

- On peut ensuite passer plus d'arguments que prévu
  - Et y accéder via la variable arguments
  - Variable locale accessible dans toutes les fonctions

```
function somme() {
  var result = 0;
  for (var i=0 ; i<arguments.length ; i++) {
    result += arguments[i];
  }
  return result;
}

console.log(somme());
console.log(somme(1,3));
console.log(somme(2,5,9));
```

Fonctions d'ordre supérieur

- Une fonction peut prendre une fonction en argument
- Permet plus de flexibilité

```
function carre(x) {
    return x*x;
}

function map (a,f) {
    for (var i=0 ; i<a.length ; i++) {
          a[i]=f(a[i]);
    }
    return a.toString();
}

map([1,2,3],carre);
```

Quelques fonctions de base

- Eval :
  - Prend une chaîne de caractère et l'interprete comme du JS
- Attention :
  - On n'a en général pas besoin de s'en servir
  - Risque si on ne sait pas ce qu'on execute (code venant d'ailleurs)

```
<html>
<body>
<script type="text/javascript">
  eval('function carre(n){ return n*n;};alert(carre(2));');
</script>
</body>
</html>
```

Quelques fonctions de base

- isNaN
  - Détermine si le paramètre n'est pas un nombre
  - NaN : Not a Number

- isFinite
  - Détermine si le paramètre est un nombre fini

```
isNaN("un nombre") //retourne true
isNaN(20) //retourne false

isFinite(240) //retourne true
isFinite("Un nombre") //retourne false
```

Quelques fonctions de base

- parseInt(string, base)
  - Analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée
  - La base peut prendre les valeurs 16 (hexadécimal) 10 (décimal), 8 (octal), 2 (binaire)

- parseFloat
  - Analyse une chaîne de caractères et retourne un nombre décimal
  - Si l'argument évalué n'est pas un nombre, renvoie NaN (Not a Number)

```
alert(5+"2"); // affiche 52
alert(5+parseInt("2")); // affiche 7

parseInt("10.33") // 10
parseInt("40 years") // 40
parseInt("He was 40") // NaN
parseInt("010") // base 8 (0…)
parseInt("0x10") // base 10 (0x…)
parseInt("10",3) // base 3
```

Quelques fonctions de base

- Number
  - Convertit l'objet spécifié en valeur numérique
- String
  - Convertit l'objet spécifié en chaîne de caractères
- Escape
  - Retourne la valeur hexadécimale à partir d'une chaîne codée en ISO-Latin-1.

```
var jour = new Date("December 17, 1995 03:24:00");
alert (Number(jour));

jour = new Date(430054663215);
alert (String(jour));

escape("!&") //retourne %21%26%
```

L'objet String

- Propriété :
  - length : retourne la longueur de la chaîne de caractères
- Méthodes :
  - anchor("lien") : formate la chaîne avec la balise <A>
  - bold( ) : formate la chaîne avec la balise <B>
  - charAt( ) : renvoie le caractère se trouvant à une certaine position
  - charCodeAt( ) : renvoie le code du caractère se trouvant à une certaine position
  - concat( ) : permet de concaténer 2 chaînes de caractères
  - fromCharCode( ) : renvoie le caractère associé au code
  - indexOf( ) : trouve l'indice d'occurrence d'un caractère dans une chaîne
  - substr( ), substring( ) : retourne une portion de la chaîne
- http://www.toutjavascript.com/reference/index.php

L'objet Array

- Propriété :
  - length : retourne le nombre d'éléments du tableau

- Méthodes :
  - concat( ) : permet de concaténer 2 tableaux
  - reverse( ) : inverse le classement des éléments du tableau
  - slice( ) : retourne une section du tableau
  - sort( ) : permet le classement des éléments du tableau

L'objet Math

- Propriétés :
  - E, LN2, LN10, LOG2E, LOG10E, PI, SQRT2, SQRT1_2

- Méthodes :
  - abs(), max(), min(), sqrt(), pow(), exp(), …
  - ceil( ) : retourne le plus petit entier supérieur à un nombre
  - floor( ) : retourne le plus grand entier inférieur à un nombre
  - round( ) : arrondi un nombre à l'entier le plus proche
  - random( ) : retourne un nombre aléatoire entre 0 et 1

L'objet Date

- Propriété : aucune

- Méthodes :
  - getFullYear(), getYear(), getMonth(), getDay(), getDate(), getHours(), getMinutes(), getSeconds(), getMilliseconds()
  - getUTCFullYear( ), … idem en temps universel
  - setFullYear( ), setYear( ), … modification d'une date
  - getTime( ) : retourne le temps stocké dans l'objet Date
  - getTimezoneOffset( ) : retourne la différence entre l'heure du client et le temps universel
  - toGMTString( ), toLocaleString( ), toUTCString( ) : convertissent la date en chaîne de caractère selon la convention GMT, selon la convention locale ou en temps universel

JAVASCRIPT ÉVÉNEMENTS / MANIPULATION

LI288 – web et développement web

Gestionnaire d'événements

- Les événements servent à interagir avec l'utilisateur
  - On peut détecter les clics, les modifications de formulaires, …
- Chaque événement a un identifiant
  - De la forme onQuelqueChose
  - Par exemple : onLoad, onClick, onMouseOver, etc.
- Un événement peut exécuter du code javascript
  - Une ou plusieurs instructions, en général un appel de fonction
- Activation :
  - <balise … onQuelqueChose="code javascript;">

Les événements de base

- Evénement onLoad
  - Se produit lorsque une page web est chargée dans la fenêtre du navigateur
  - Toute la page (y compris les images qu'elle contient si leur chargement est prévu) doit avoir été chargée pour qu'il ait lieu
  - Cet événement peut être associé à une image seulement (il se produit alors une fois le chargement terminé)

Les événements de base

- Evénement onClick
  - Se produit lorsque l'utilisateur clique sur un élément spécifique dans une page, comme un lien hypertexte, une image, un bouton, du texte, etc.
  - Ces éléments sont capables de répondre séparément à cet événement
  - Il peut également être déclenché lorsque l'utilisateur clique n'importe où sur la page s'il a été associé non pas à un élément spécifique, mais à l'élément body tout entier

Les événements de base

- Événement onMouseOver
  - Analogue à onClick sauf qu'il suffit que l'utilisateur place le pointeur de sa souris sur l'un des éléments précités (lien hypertexte, image, bouton, texte, etc.) pour qu'il ait lieu

- Événement onMouseOut
  - A l'inverse de onMouseover, cet événement se produit lorsque le pointeur de la souris quitte la zone de sélection d'un élément.

Une liste plus longue

- Globales :
  - onAbort : chargement d'une image interrompu
  - onError : une erreur durant le chargement de la page
  - onLoad : chargement de la page
  - onUnload : l'utilisateur quitte la page

- Souris :
  - onBlur : un élément perd le focus
  - onClick : clic sur l'élément
  - onDblclick: double clic sur l'élément
  - onDragdrop : drag and drop sur la fenêtre du navigateur
  - onFocus : le focus est donné à un élément
  - onMouseOver : la souris passe sur un élément
  - onMouseOut : la souris quitte un élément
  - onResize : la fenêtre est redimensionnée

Une liste plus longue

- Formulaires :
  - onChange : modification d'un champ de données
  - onReset : effacement d'un formulaire à l'aide du bouton Reset.
  - onSelect : sélection d'un texte dans un champ "text" ou "textarea"
  - onSubmit : clic sur le bouton de soumission d'un formulaire

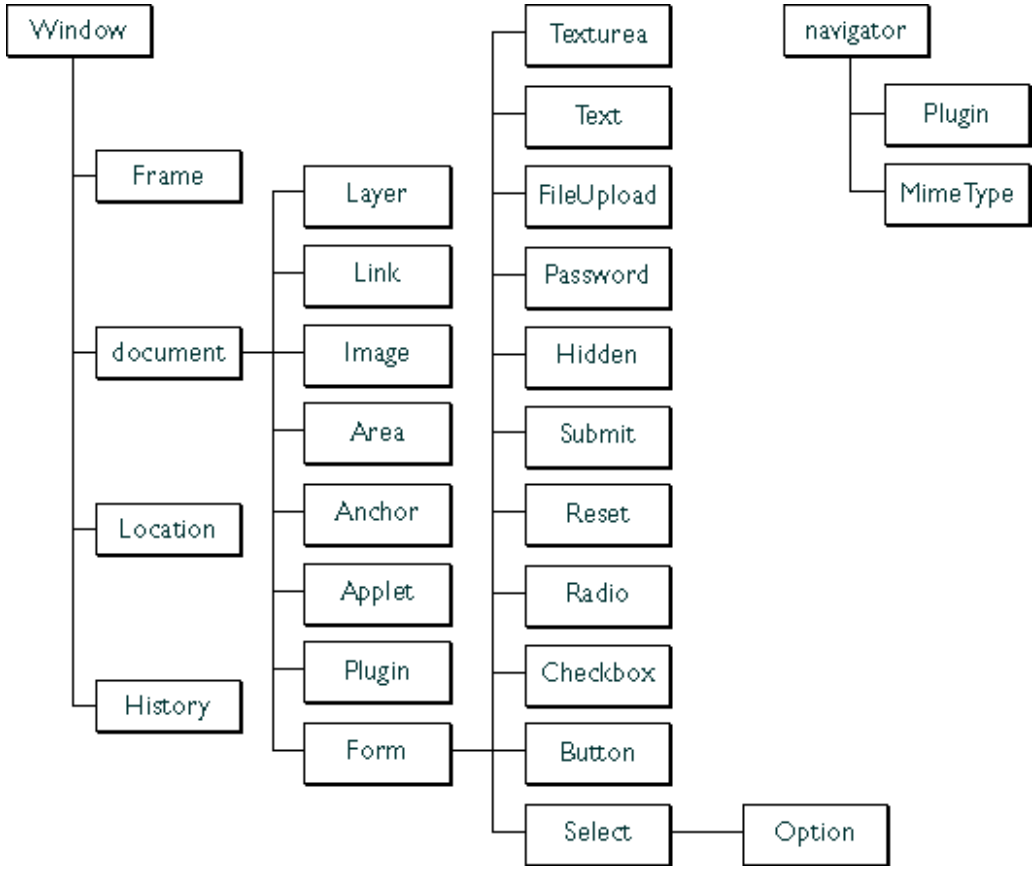
- Clavier :
  - onKeyDown : appui sur une touche du clavier
  - onKeyPress : appui et maintient sur une touche
  - onKeyUp : relâchement d'une touche

Entrées/sorties

- Affichage pour l'utilisateur :
  - alert()
  - confirm()
    - Alert + boutons ok/annuler
  - prompt()
    - Récupération entrée tapée par l'utilisateur
- document.write() :
  - permet d 'écrire du code HTML dans la page WEB
- Console.log() :
  - Affichage dans la console (debug)

Les objets du navigateur

- Window : fenêtre du navigateur
- Document : contenu de la fenêtre
  - Ensemble des éléments HTML
- Accès aux éléments :
  - Méthodes de document :
    - getElementById( )
    - getElementsByName
  - Tableaux Javascript contenant certains éléments :
    - document.links



Manipulation des objets

- Pour récupérer un objet, il faut préciser son « chemin d'accès » dans l'arborescence.
  - On peut omettre window.document (un seul objet "document")

```
<html>
<body onLoad="window.document.f1.zone.value='Bonjour';">
<form name="f1">
  <input name="zone" type="text">
</form>
</body>
</html>
```

Exemples

```
<html><head>
<script type="text/javascript">
function changeCouleur(color) {
  var ml = document.getElementById("maListe");
  ml.setAttribute("style","color:"+color);
}
</script>
</head><body>

<ul id="maListe">
  <li><a href="javascript: changeCouleur ('red');">Rouge</a></li>
  <li><a href="javascript: changeCouleur ('blue');">Bleu</a></li>
  <li><a href="javascript: changeCouleur ('black');">Noir</a></li>
</ul>
</body></html>
```

Fonctions de modification

- Recherche :
  - document.getElementById()
  - document.getElementsByTagName()
  - document.getElementsByClassName()
- Modification éléments :
  - element.innerHTML=
  - element.setAttribute(attribute,value)
  - element.style.property=
- Modification document :
  - document.createElement()
  - document.removeChild(child)
  - element.appendChild(child)
  - element.replaceChild(newchild, oldchild)
- Evenements :
  - element.on****=function(){…}

Pour aller plus loin

- Toutes les commandes sur :
  - https://developer.mozilla.org/fr/DOM/element#Propriétés

- Pour tester/débugger :
  - Firebug
  - Console d'erreur.
  - Utiliser des alertes.

AJAX

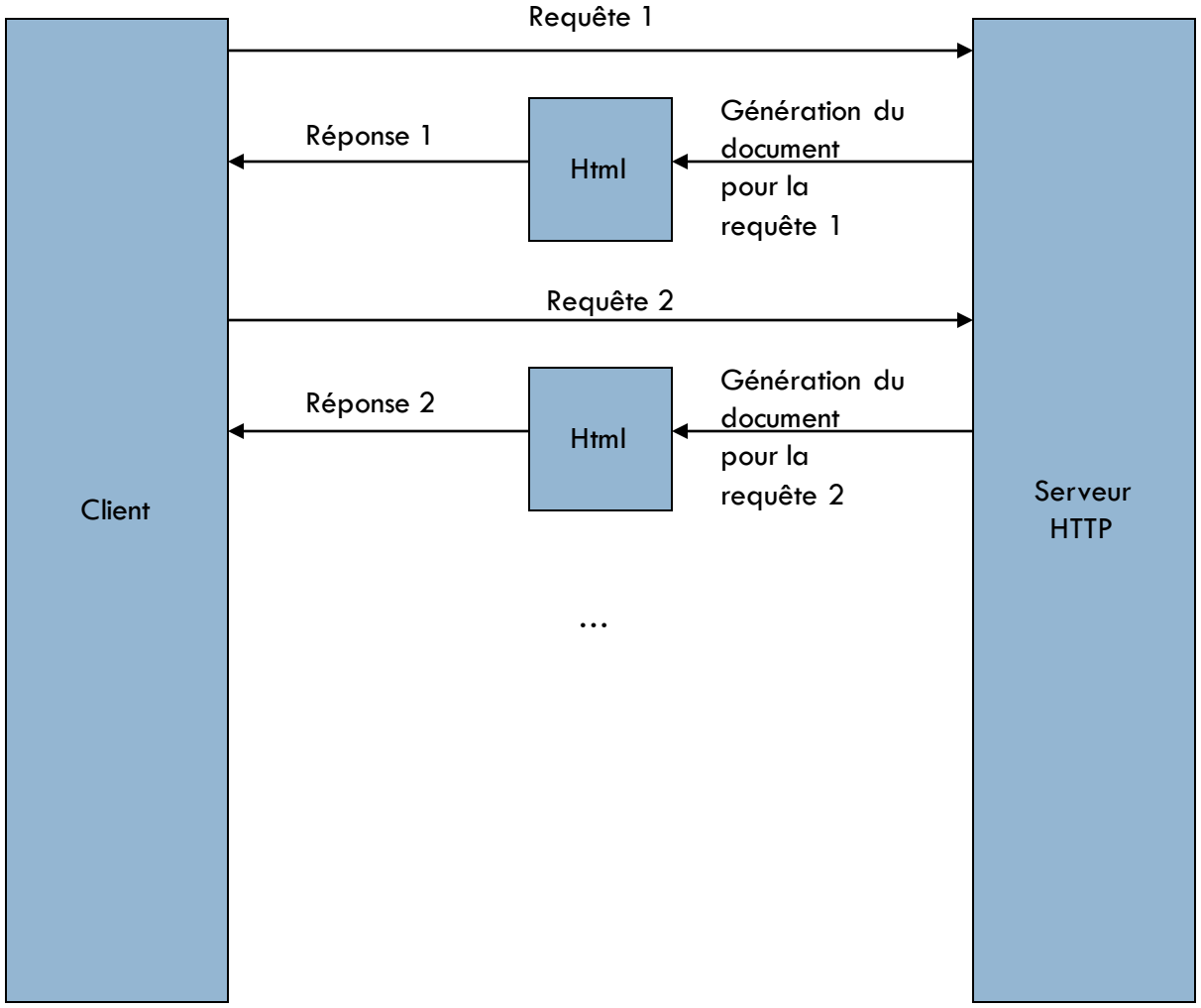
LI288 – web et développement web

Application traditionnelle

- Application WEB traditionnelle :
  - Le client envoie une requete HTTP
  - Le serveur renvoie une page

- Consommation inutile de la bande passante :
  - Une grande partie du code HTML est commun aux différentes pages de l'application.
- Le chargement d'une nouvelle page à chaque requête n'est pas ergonomique

Application traditionnelle



AJAX

- Qu'est-ce qu'AJAX ?
  - Asynchronous Javascript and XML

- Pourquoi AJAX:
  - Javascript est très utilisé au niveau du client :
    - validation de formulaire, modifications de la page, …
  - Tout ne peut pas être confié au client :
    - Manque de sécurité/confiance
    - Limitations

AJAX

- Principe de base :
  - Le client et le serveur dialoguent.

- Autant faire en sorte que les messages soient le plus petits possibles.
  - Le client n'a pas besoin de toute la base de données, juste de suffisamment de données pour le client.
- Le serveur et le client ont chacun un travail
  - L'application ne doit donc pas être prise en charge entièrement d'un coté ou de l'autre.

Principe de base

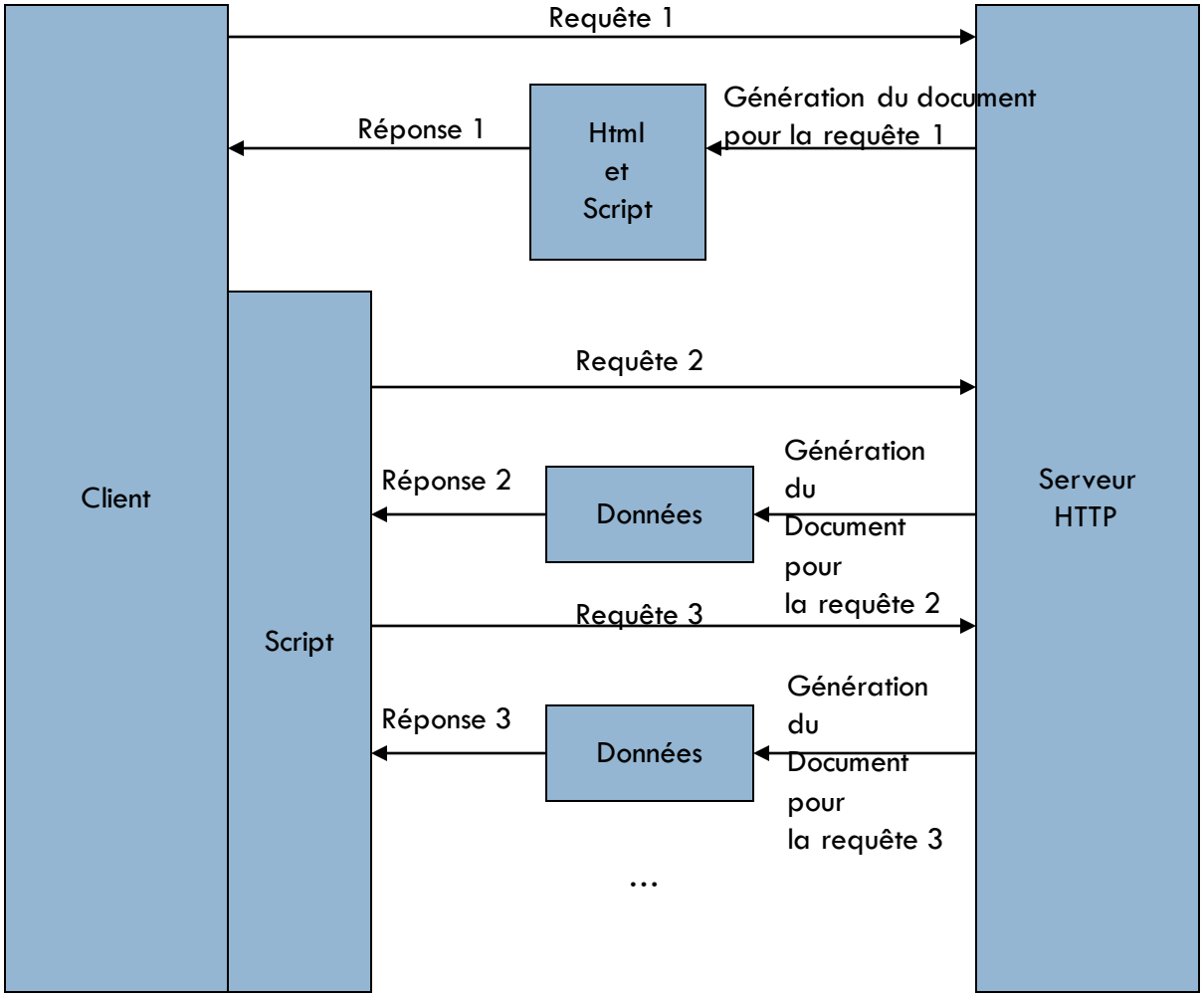
1. L'application Javascript émet des requêtes vers le serveur avec un protocole donné

2. Le serveur répond avec les informations demandées

3. Tout se passe sans rechargement de la page
   - Mode synchrone ou asynchrone pour le client

4. Javascript traite les données reçues et modifie la page en conséquence

AJAX


Client
Script
Serveur HTTP
Requête 1
Réponse 1
Html et Script
Génération du document pour la requête 1
Requête 2
Réponse 2
Données
Génération du Document pour la requête 2
Requête 3
Réponse 3
Données
Génération du Document pour la requête 3
...

Comment ça marche

- Un exemple sans AJAX :
  - requête faite automatiquement par le navigateur
  - récupération d'une image à distance

```
<body onLoad="javascript:document.images[0].src =
   'http://...'">


<img id="img01" src="vide.jpg"></a>


</body>
```

L'objet XmlHttpRequest

- AJAX se base sur XmlHttpRequest
  - Initialement développé par Microsoft, en tant qu'objet ActiveX, pour Internet Explorer 5
  - Puis repris et implémenté sous Mozilla 1 Safari 1.2, Konqueror 3.4 et Opera 8.
  - Pas supporté par certains vieux navigateurs.
  - Proposé en 2006 pour devenir une recommandation du W3C :
    - http://www.w3.org/TR/XMLHttpRequest/
    - Draft novembre 2009

L'objet XmlHttpRequest

- Problèmes :
  - Nécessite un navigateur compatible, autorisant le Javascript et XMLHTTP.
  - Nécessite plus de tests car il existe de grandes différences entre les navigateurs.
    - XMLHttpRequest n'est pas implémenté de la même manière selon les navigateurs (et les versions des navigateurs).
- Solution la plus simple :
  - Aller chercher le code générique avec google.

Création

```
function getXMLHttpRequest() {
    if (window.XMLHttpRequest)
            return new XMLHttpRequest();
    if (window.ActiveXObject) {
            var names = [
                    "Msxml2.XMLHTTP.6.0",
                    "Msxml2.XMLHTTP.3.0",
                    "Msxml2.XMLHTTP",
                    "Microsoft.XMLHTTP" ];
            for(var i in names) {
                    try {
                            return new ActiveXObject(names[i]);
                    } catch(e) {}
            }
    }
    window.alert("Votre navigateur ne prend pas en charge l'objet XMLHTTPRequest.");
    return null;
}
```

Exemple simple

```
<html>
<body>
<script type="text/javascript">
function getXMLHttpRequest() {...}

function ajax() {
   var xhr=getXMLHttpRequest();
   xhr.open("GET", "test.html", false);
   xhr.send(null);
   alert(xhr.responseText);
}
</script>
<a href="javascript:ajax();">Cliquez-moi !</a>
</body>
</html>
```

Propriétés de l'objet

- onreadystatechange :
  - Fonction appelée à chaque changement d'état.

- readyState = statut de l'objet :
  - 0 : non initialisé.
  - 1 : ouverture = méthode open() appelée avec succès.
  - 2 : envoyé = méthode send() appelée avec succès.
  - 3 : en train de recevoir = données en cours de transfert.
  - 4 : terminé = données chargées.

- responseText / responseXML
  - Réponse sous forme de chaîne de caractères / objet DOM.

Propriétés de l'objet

- status :
  - code numérique de réponse du serveur HTTP, à tester quand les données sont chargées (readyState=4)
    - 200 : OK.
    - 404 : page introuvable.
    - ...
  - En local (sans serveur web), 0=OK

- statusText :
  - message accompagnant le code de réponse :
    - 404 : Not Found…

Méthodes principales de l'objet

- open(method, url, async, user, password)
  - Prépare une requête en indiquant la méthode, l'URL, la drapeau de synchronisation (et éventuellement le nom d'utilisateur et le mot de passe).
- send (contenu)
  - Effectue la requête, éventuellement en envoyant les données.
- Modification/récupération des entêtes, etc.

Synchrone ou asynchrone

- Requête synchrone :
  - Tout est bloqué en attendant la réponse.
    - Mauvais pour l'utilisateur.
  - Les réponses arrivent forcément dans l'ordre.
  - C'est l'approche classique et simple.

- Requête asynchrone :
  - Le navigateur continue à répondre aux événements en attendant la réponse.
  - Attention à ne pas faire n'importe quoi.

Javascript Asynchrone

- Le choix entre synchrone et asynchrone se fait dans l'appel à XMLHttpRequest (méthode open) :
  - true pour asynchrone
  - false pour synchrone

- Dans le cas d'un appel asynchrone, le résultat est récupéré par une fonction :
  - xhr.onreadystatechange = function() { ...};
  - Cette fonction sera appelée à chaque changement d'état de notre objet.

Pour résumer

- Deux méthodes principales :
  - open : pour établir une connexion.
  - send : pour envoyer une requête au serveur.
- Récupération des données :
  - Champs responseXml ou responseText.
- Créer un nouvel objet XmlHttpRequest, pour chaque fichier à charger.

Un exemple

```
function ajax() {
  var xhr=getXMLHttpRequest();
  xhr.onreadystatechange  = function() {
    if(xhr.readyState  == 4) {
      if(xhr.status  == 200)
        alert("Received : "  + xhr.responseText);
      else
        alert("Error code :  " + xhr.status);
    } else {
      alert("en cours : " + xhr.readyState);
    }
  };
  xhr.open("GET", "eval.html", true);
  xhr.send(null);
}

<body><a href="javascript:ajax();">Cliquez-moi !</a></body>
```

HTTP GET ou POST

- GET pour récupérer des données
  - Ne devrait pas provoquer de mises à jour sur le serveur.
  - Les requêtes GET doivent pouvoir être bookmarkées ou mises en cache.
- POST pour envoyer des données
  - Pour tout ce qui ne correspond pas à un GET

Attention !

- Les requêtes AJAX asynchrones passent par Internet
  - Aucune garantie que les paquets arrivent dans l'ordre.
  - Aucune garantie qu'une requête soit terminée avant qu'une autre ne soit lancée :
    - Les délais peuvent varier énormément à cause de la charge du serveur et du réseau.

Inconvénients

- JavaScript doit être activé.
- Les données chargées de façon dynamique ne font pas partie de la page. Prise en compte par les moteurs de recherche pas claire.
- Asynchrone => affichage avec délai, peut poser problème à l'utilisateur.
- Le bouton « Page précédente » ne marche pas en général.

Conclusions sur Ajax

- Combinaison des langages standards du WEB (Javascript, DOM HTML, XML)
- Grâce à l'objet XMLHttpRequest
- WEB dynamique « coté client »
- Utilisé par tous les sites « WEB 2.0 »

JSON
JAVASCRIPT OBJECT NOTATION

LI288 – web et développement web

JSON ?

- Format d'échange de données.
- Objectifs :
  - Simple.
  - Extensible.
  - Ouvert.
  - Lisible par un humain.
- Similaire à la définition des objets Javascript.

JSON : JavaScript Object Notation

- Les types de base
  - Nombres entiers, rééls ou à virgule flottante
  - Chaînes de caractères
  - Booléen true et false
  - Tableaux [..., ...] ou tableaux associatifs (objets) "clé":valeur : {..., ...}
  - null

```
{
  "nom":"Guillaume",
  "ue": {"nom":"harmweb", "lieu":"IUT"},
  "notes": [1, 2, 4, 8, 16, 32]
}
```

JSON et Javascript

- Inclusion dans du HTML
  - <script> var data = JSONdata; </script>

- Peut être converti en un objet Javascript
  - responseDATA = JSON.parse(responseText);
  - responseData = eval('(' + responseText + ')');

Exemple d'utilisation de JSON

- Coté client :
  - JSON inclus dans JavaScript.
  - Le contenu est assigné à une variable et devient un objet.

```
// Création de la connexion :
var req = new XMLHttpRequest();
req.open("GET", "fichier.json", true);
req.onreadystatechange = function() {
  if (req.readyState == 4) {
    console.log(JSON.parse(xhr.responseText));
  }
}
req.send(null);
```