JAVASCRIPT

Cours Nouvelles Technologies du web

Javascript - objectif

- Dynamique simple de sites HTML coté client :
  - Validation de formulaires, calculs, messages,
  - Modification de la page web,
  - Communication avec le serveur (AJAX)
- Développement d'applications coté client :
  - Interfaces et traitements complexes (ex : google doc)
- Développement d'applications coté serveur :
  - Node.js

Caractéristiques principales

- Le Javascript est :
  - Un langage spécifique inclus dans du code HTML
  - Exécuté chez le client
  - Sans compilation a priori
- Supporté par tous les navigateurs web (récents)
- Syntaxe proche du C mais
  - Langage objet orienté prototype
  - Fonctionnel (fonctions d'ordre supérieur)

Outils

- Débug et efficacité :
  - Console (firebug) + Yslow
- Validation javascript :
  - jslint.com
- Optimisation (minification) :
  - google closure compiler
- …

Plan du cours

- Syntaxe : tests, boucles, fonctions
- Fonctions d'ordre supérieur
- Objets
- Événements / Manipulation de page
- Ajax
- Cookies
- Optimisation de code

JAVASCRIPT SYNTAXE

Cours Nouvelles Technologies du web

HTML et JavaScript

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

- Utilisation d'une balise spécifique :
  - <script type="text/javascript">…</script>
  - <script language="text/javascript">…</script>
  - <script language="javascript">…</script>
  - Les deux derniers sont obsolètes

Insertion dans une page HTML

- Dans le corps de la page HTML
  - Entre les balises <body> et </body>
  - Code exécuté lors du chargement de la page

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

Insertion dans une page HTML

- Dans l'entête de la page
  - Entre les balises <head> et </head>
  - Code exécuté lors d'un événement utilisateur
  - L'événement se trouve dans le corps du document.

```
<html>
<head>
<script type="text/javascript">
  function f () { alert('Au revoir'); }
</script>
</head>
<body onUnload="f();"> // fermeture de la page courante
</body>
</html>
```

Insertion dans une page HTML

- A placer dans l'entête ou dans le fichier
  - Fichier en format texte (ne contient pas de balise html)
  - Avantage : réutilisation du script dans plusieurs pages
  - Inconvénient : requête supplémentaire vers le serveur

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

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

Structure d'un script

- Suite d'instructions (voir C)

- Règles générales
  - On sépare les commandes par des point-virgule ";"
    - Pas obligatoire mais généralement recommandé
  - 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)

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

//Déclaration et affectation d'un tableau
var car = ["a", "b", "c"];
var moi = {"nom":"guillaume", "prenom":"jean-loup"};
alert(car[1] + moi.nom);
```

Les types de variables

- Types :
  - String
  - Number : 10^-308 >nombre < 10^308
    - 3 valeurs spéciales : Positive Infinity, Negative Infinity, Nan
  - Boolean : true et false
  - Undefined : type d'une variable avant qu'elle soit affectée
  - Null : absence de données dans une variable

- Le type d'une variable dépend de son contenu
  - var maVariable = "tutu"; // type String
    maVariable =10; // type Number

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

Syntaxe, les boucles

- Boucle for :
  - for (i=0; i<5; i++) {…}
- Boucle while :
  - while (test) {…}
  - do {…} while (test)

Syntaxe, les conditions

- if (test) {} else {}

- Tests possibles :
  - Egalité : ==, !=
  - Inférieur, supérieur : =<, >=, >, <
  - Opérations bit à bit : &, |
  - Identique à : ===, !== (teste valeur et type)
    - ('1' == 1) // true
    - ('1' === 1) // false
  - Opérations logiques : &&, ||

Les fonctions

- Définition :
  - function maFonction(arg1,arg2) {instr;}

- Appel :
  - maFonction("12",13);

JAVASCRIPT FONCTIONS D'ORDRE SUPÉRIEUR

Fonctions d'ordre supérieur

- Une fonction peut prendre une fonction en argument

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

function somme2 (a,b,f) {
  return f(a)+f(b);
}

alert(somme2(2,3,carre));
```

Fonctions d'ordre supérieur

- Une fonction peut retourner une fonction
- Notion de clôture
  - La variable a est conservée dans la fonction b

```
function foo(a) {
   function ajouter(b) {return a+b;};
   return ajouter;
}
var bar = foo(2);
alert(bar(1));
```

Clôture

- On souhaite afficher les éléments d'un tableau toutes les secondes
  - Résultat ?

```
var a = ["a","b","c","d"];
for(var i = 0; i < 3; i++) {
  window.setTimeout(
    function() { alert(a[i]) },
    1000*(i+1)
  );
}
```

Clotûre

- Résultat : affiche trois fois "d"
  - La seule variable i dans le contexte vaut 4
- Version avec clôture :
  - arg1 défini en dehors de la fonction
  - Pas détruit tant que la fonction est utilisée

```
var a = ["a","b","c","d"];
for(var i = 0; i < 3; i++) {
  window.setTimeout(
    ( function(arg1) {
        return function() {alert( a[arg1] ); };
    } ) ( i ),
    1000*(i+1)
  );
}
```

Map

```
function carre(a) {
  var res = new Array();
  for (var i=0 ; i<a.length ; i++) {
    res.push(a[i]*a[i]);
  }
  return res;
}

function double(a) {
  var res = new Array();
  for (var i=0 ; i<a.length ; i++) {
    res.push(a[i]*2);
  }
  return res;
}

alert(carre([1,2,3]));
alert(double([1,2,3]));
```

Map

```
function map (a,f) {
  var res = new Array();
  for (var i=0 ; i<a.length ; i++) {
    var v = a[i]; // !!!
    res.push(f(v));
  }
  return res;
}

alert(map([1,2,3], function(x){return x*x;}));

alert([1,2,3].map(
  function(current, index, array){
    return current*current;
}));
```

Reduce

```
function sum(a) {
  var res = 0;
  for (i = 0; i < a.length; i++)
    res += a[i];
  return res;
}

function concat (a) {
  var res = "";
  for (i = 0; i < a.length; i++)
    res += a[i];
  return res;
}

alert(sum([1,2,3,4,5]));
alert(concat(["a","b","c","d","e"]));
```

Reduce

```
function reduce(f, a, start) {
  var s = start;
  for (i = 0; i < a.length; i++) {
    var v = a[i];
    s = f( s, v);
  }
  return s;
}

alert(reduce(
   function(a, b){ return a + b; },
  [1,2,3,4,5],
  0));

alert([1,2,3,4,5].reduce(
  function(previous, current, index, array){
    return previous + current;
}));
```

Filter

```
function impair(a) {
  var res = new Array();
  for (var i = 0; i < a.length; i++) {
    if (a[i]%2==1)
      res.push(val);
  }
  return res;
}

alert(impair([1,2,3,4,5]));
```

Filter

```
function filter(f, a) {
  var res = new Array();
  for (var i = 0; i < a.length; i++) {
    var val = a[i];
    if (f(val))
      res.push(val);
  }
  return res;
}

alert(filter(
  function(a){ return a%2==1; },
  [1,2,3,4,5]));

alert([1,2,3,4,5].filter(
  function(current, index, array){
    return current%2==1;
}));
```

JAVASCRIPT OBJETS / PROTOTYPE

Cours Nouvelles Technologies du web

Les Objets de base

- String – chaine de caractères
  - length, substr(), concat(), …
- Array – tableaux
  - length, concat(), slice(), sort(), …
- Math – fonctions mathématiques
  - abs(), max(), min(), sqrt(), round( ), random( ), …
- Date – gestion des dates/heures
  - getDate(), getTime( ), …

Objets liés au navigateur

- Le navigateur est un objet qui s'appelle "navigator"
- La fenêtre du navigateur se nomme "window"
- La page HTML est un objet, qui s'appelle "document"
- Un formulaire, un lien hypertexte, … sont des objet
- Ces objets peuvent réagir à des "Evénements"
  - Cf plus loin

- On peut définir n'importe quel nouvel objet

Création d'un objet

- New : créer une instance d'un objet
  - Objet défini par l'utilisateur
  - Objets prédéfinis, Array, Boolean, Date, Function, Image, Number, Object, ou String.
  - nouvel_objet = new type_objet(parametres)

```
texte = new String("Une chaîne de caractère");
```

Création directe

- Similaire aux tableaux associatifs
- Séparation par des virgules ","
- On crée un seul objet

```
var obj = {
  monAttribut: "valeur",
  maMethode: function(p) { alert("parametre : " + p);}
};

alert(obj.monAttribut);
obj.maMethode("test");
```

Création par clonage

- Création d'un objet via une fonction
  - Équivalent du constructeur
  - + définition de tous les slots

```
var obj = {
  monAttribut: "valeur",
  maMethode: function(p) { alert("parametre : " + p);}
};

var obj2=Object.create(obj);
alert(obj2.monAttribut);
```

Langage orienté prototype

- Programmation orientée usuelle :
  - Classe = ensemble de comportements et de structures
  - Instance = données spécifiques (état)
  - Classes majoritairement statiques

- Programmation orienté prototypes
  - Objets clonés à partir d'un prototype
  - On ne distingue pas attributs et méthodes (slot)
  - L'héritage se fait par clonage
  - Un prototype et les slots sont dynamiques (mutables)

Prototypage

- Ajout de slot à un objet, on voudrait :
  - pouvoir ajouter un slot à une instance
  - pouvoir ajouter un slot à toutes les instances

```
function Cercle(rayon){
  this.rayon=rayon;
}
var cercle1=new Cercle(1);
cercle1.pi=3.14;

var cercle2 = new Cercle(2);
alert(cercle2.pi); // undefined

Cercle.prototype.pi=3.14;
alert(cercle2.pi); // 3.14
```

Prototypage

- Modification d'un objet déjà existant

```
function inverse() {
  var t = "";
  for (i=this.length-1;i>=0;i--)
    t += this.charAt(i);
  return t;
}

String.prototype.inverse = inverse;
var s = new String("abcdef");
alert(s.inverse());
```

Héritage

- Via le prototype
  - Pas unique solution (cf inheritFrom)

```
function Sup(){}
function Sub(){}
Sub.prototype = new Sup();
b = new Sub();
console.log(b instanceof Sub, b instanceof Sup);
```

JAVASCRIPT ÉVÉNEMENTS

Cours Nouvelles Technologies du 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;">…</balise>

Les événements de base

- Evénement onLoad
  - Exécuté quand la page web est chargée (images comprises)
- Evénement onClick
  - Exécuté lors d'un clic sur un élément (balise)
- Événement onMouseover
  - Exécuté lorsque la souris est passée sur un élément
- Événement onMouseout
  - Exécuté lorsque la souris quitte un élément

Une liste plus longue

- Evénements globaux :
  - 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

- Evénements 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

- Evénements 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

- Evénements clavier :
  - onKeydown : appui sur une touche du clavier
  - onKeypress : appui et maintient sur une touche
  - onKeypress : relâchement d'une touche

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>
```

JAVASCRIPT MANIPULATION DE PAGE

Cours Nouvelles Technologies du web

Entrées/sorties

- 3 types de boites de messages peuvent être affichés en utilisant Javascript
  - Méthode alert()
    - sert à afficher à l'utilisateur des informations simples de type texte. Une fois que ce dernier a lu le message, il doit cliquer sur OK pour faire disparaître la boîte
  - Méthode confirm()
    - permet à l'utilisateur de choisir entre les boutons OK et Annuler.
  - Méthode prompt()
    - La méthode prompt() permet à l'utilisateur de taper son propre message en réponse à la question posée

- La méthode document.write permet d 'écrire du code HTML dans la page WEB

Les objets du navigateur

- L'objet le plus haut dans la hiérarchie est "window" qui correspond à la fenêtre même du navigateur.
- L'objet "document" fait référence au contenu de la fenêtre :
  - "document" = ensemble des éléments HTML de la page.

- On peut accéder ces éléments avec :
  - méthodes propres à l'objet document :
    - getElementById( ) trouve l'élément avec son identifiant (ID)
    - getElementByName
  - soit des collections d'objets qui regroupent sous forme de tableaux Javascript tous les éléments de type déterminé.

L'objet document

- Propriétés :
  - applets, forms, images, links : retourne les collection d'applets java, formulaires… présente dans le document
  - cookie : permet de stocker un cookie
  - domain : nom de domaine du serveur
  - referrer : indique l'adresse de la page précédente
  - title : titre du document

```
<html><head><title>Test</title></head>
<body>
<a href="http://www.yahoo.fr/">Yahoo</a>
<a href="http://www.google.fr/">Google</a>
<script type="text/javascript">
  alert(document.title);
  for(var i=0; i < document.links.length; ++i)
    alert("<br>" + document.links[i]);
</script>
</body></html>
```

Manipulation des objets

- Position dans l'arborescence (DOM)
  - Le navigateur utilise par défaut la fenêtre courante
  - On peut omettre window.document
    - Mais c'est non standard…

```
<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 liststyle(color) {
  var ml = document.getElementById("maliste");
  ml.setattribute("style","color:"+color);
}
</script>
</head><body>

<ul id="maliste">
  <li><a href="javascript: liststyle('red');">rouge</a></li>
  <li><a href="javascript: liststyle('blue');">bleu</a></li>
  <li><a href="javascript: liststyle('black');">noir</a></li>
</ul>
</body></html>
```

Exemples

```
<html><head
<script type="text/javascript">
function liststyle(color) {
  var ml = document.getElementById("maliste");
  ml.innerHTML="<p>tout est cassé</p>";
}
</script>
</head><body>

<ul id="maliste">
  <li><a href="javascript: liststyle('red');">rouge</a></li>
  <li><a href="javascript: liststyle('blue');">bleu</a></li>
  <li><a href="javascript: liststyle('black');">noir</a></li>
</ul>
</body></html>
```

Pour aller plus loin

- Toutes les commandes sur :
  - http://www.toutjavascript.com/reference/index.php
  - https://developer.mozilla.org/fr/DOM/element#Propriétés

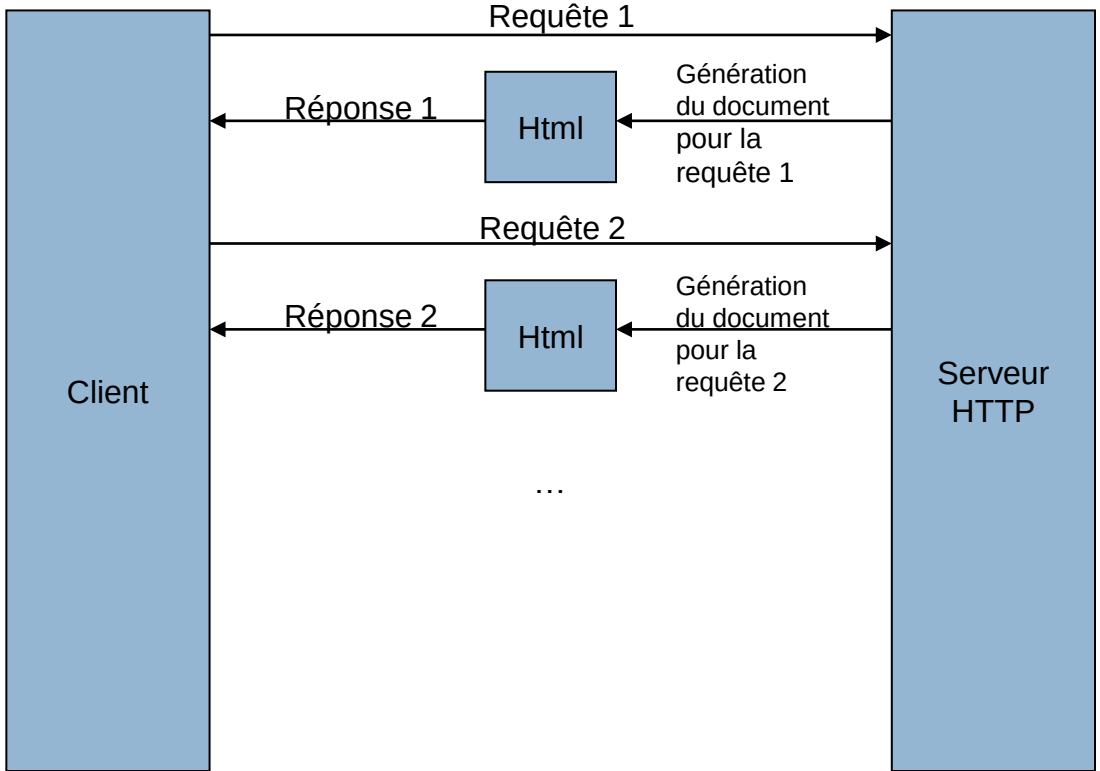
- Pour tester/débugger :
  - Firebug (firefox), Outils de développement (Chrome)
  - Console d'erreur.
  - Utiliser des alertes.

AJAX

Cours Nouvelles Technologies du web

Application traditionnelle

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



Application traditionnelle

- Inconvénients :
  - Consommation inutile de la bande passante :
    - Une grande partie du code est commun aux différentes pages. Le navigateur gère un cache mais pas pour des parties d'un fichier HTML
  - Le chargement d'une nouvelle page n'est pas ergonomique :
    - Délai variable pendant lequel la page n'est pas affichée ou seulement partiellement
    - A comparer à une application traditionnelle

Application traditionnelle

- Avantages :
  - 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
  - Tout ne peut pas être confié au client :
    - Manque de sécurité/confiance
      - On ne veut pas envoyer la base de données au client
    - Limitations en puissance de calcul

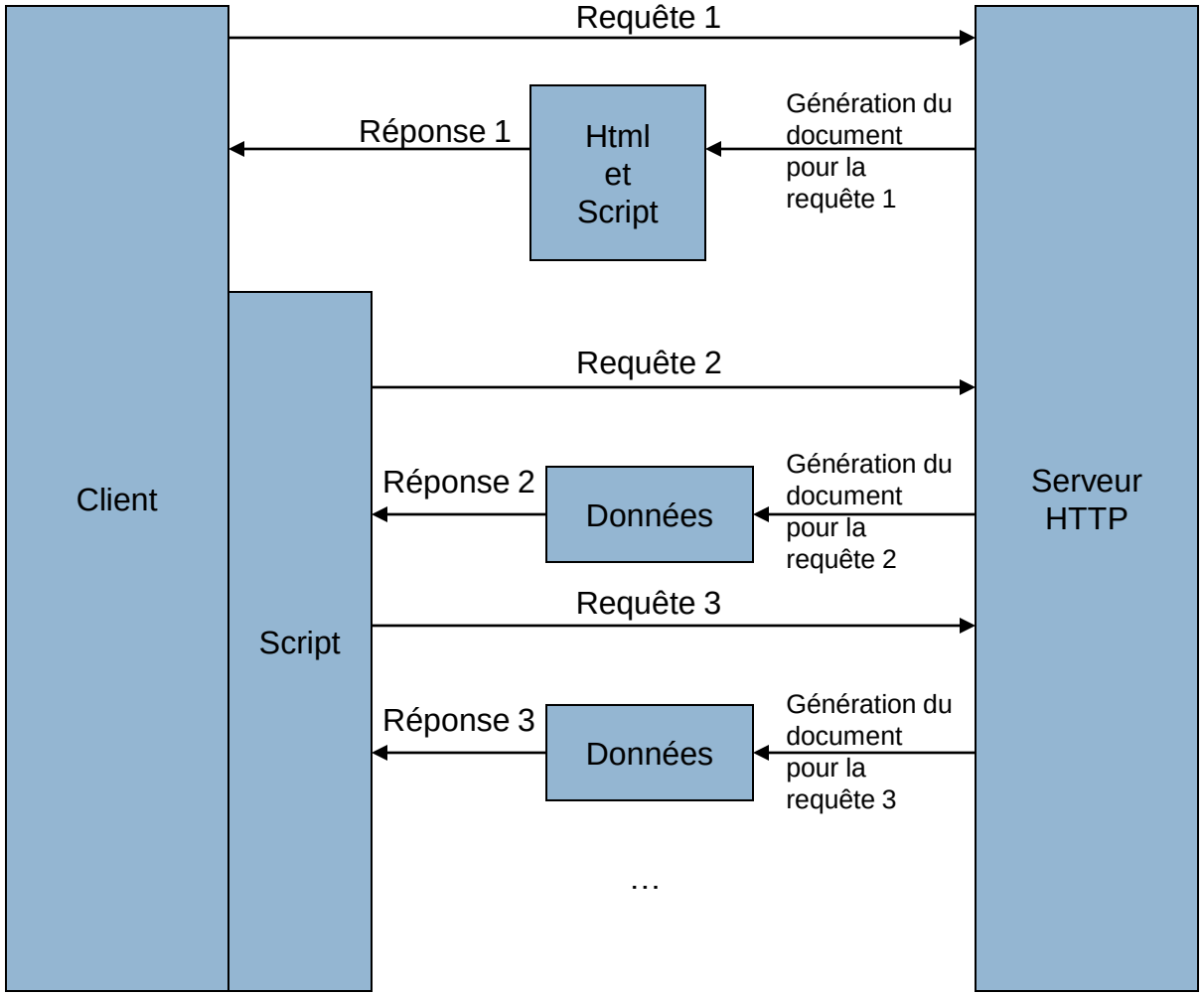
Pourquoi AJAX

- Nécessité de communication entre client et serveur
- Volonté de ne pas perdre de bande passante

- Principe de base :
  - Javascript émet des requêtes vers le serveur
  - Le serveur répond avec les informations demandées
  - Javascript traite les données et modifie la page si nécessaire
  - Tout se passe sans rechargement de la page

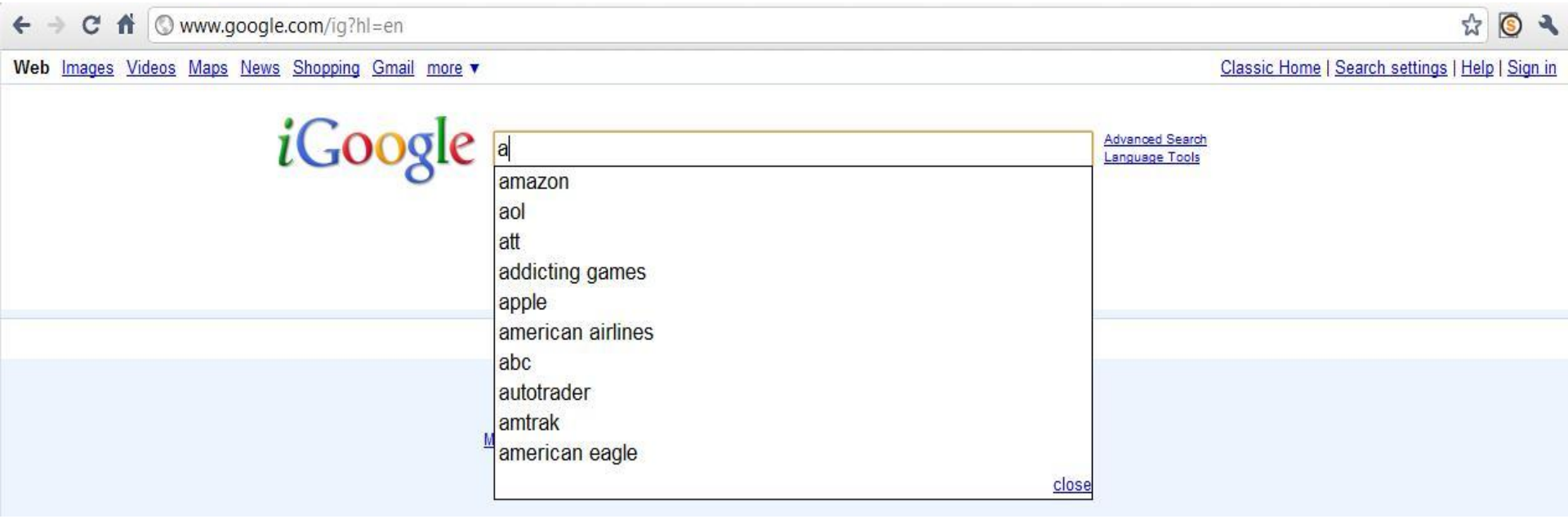
= AJAX / Asynchronous Javascript and XML

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
…

Exemple d'utilisation

- Auto complétion d'un moteur de recherche :
  - L'utilisateur commence à taper "xxx"
  - La liste des requêtes possibles s'affiche



Exemple d'utilisation

```
<html><head>
<script type="text/javascript">
function getAnswers(query) {...}

function displayAnswers(ml) {
  var answers = getAnswers(ml);
  var list = "";
  for (i=0; i<answers.length ; i++)
    list += "<li>"+answers[i]+"</li>";
  document.getElementById("answers").innerHTML = list;
}
</script>
</head><body><form>
<input id="search" type ="text" value="" onKeyUp="displayAnswers(this.value);">
<ul id="answers">
</ul>
</form></body></html>
```

Exemple d'utilisation

- Moteur de recherche :
  - L'utilisateur commence à taper "xxx"
  - Javascript récupère le "xxx"
  - Puis demande au serveur les recherches fréquentes commençant par "xxx"
  - Et affiche la réponse du serveur
- Il faut juste écrire la fonction getAnswers()

- Le tout prend moins d'une seconde, c'est transparent pour l'utilisateur

Qui utilise Ajax

- Clients de messagerie : Gmail, Yahoo Mail, HotMail
- Google, Google Maps
- FlickR, Picasa
- Deezer
- Youtube, Dailymotion
- Myspace, Facebook

Fonctionnement

- Via l'objet XmlHttpRequest :
  - 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 sur Internet

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;
}
```

Méthodes de l'objet

- open(method, url, async [, user, password])
  - Prépare une requête en indiquant
    - La méthode (GET ou POST), l'URL, le mode de synchronisation
    - Et éventuellement le nom d'utilisateur et le mot de passe
- send (contenu)
  - Effectue la requête, éventuellement en envoyant les données
- setRequestHeader("champ","valeur")
- abort()
- getAllResponseHeaders()
- getResponseHeader("champEntete")

Propriétés de l'objet

- responseText / responseXML
  - Réponse sous forme de chaîne de caractères / XML

- status / statusText :
  - Code numérique de réponse du serveur HTTP
  - À tester pour s'assurer que tout est bon
    - 200 : OK
    - 404 : page introuvable
    - ...
  - message accompagnant le code de réponse :
    - 404 : Not Found…

Exemple simple

```
<html>
<body>
<script language="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>
```

Synchrone ou asynchrone

- Requête synchrone :
  - Tout est bloqué en attendant la réponse
  - Simple à coder (pas ergonomique)
  - Les réponses arrivent forcément dans l'ordre

- Requête asynchrone :
  - Non bloquant
  - Les réponses peuvent arriver dans le désordre

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() { ...};

Autres propriétés de l'objet

- readyState :
  - État 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

- onreadystatechange :
  - callback appelé à chaque changement d'état
  - Sans doute au moins une fois

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);}};
    xhr.open("GET", "test2.html", true);
    xhr.send(null);
}

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

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 ou du réseau
- Nécessite de faire très attention à ce qu'on fait :
  - Cf exemple de recherche google, on ne veut pas afficher pour "a" après celles pour "ab"
  - Une seule requête par objet XMLHttpRequest !

Utilisation de clôture

```
function runAjax() {
  var xhr = new XMLHttpRequest();

  xhr.onreadystatechange = function () {
    if(xhr.readyState == 1)
      document.getElementById("resultat").innerHTML += "connexion établie<br />";
    if(xhr.readyState == 2)
      document.getElementById("resultat").innerHTML+= "requête reçue<br />";
    if(xhr.readyState == 3)
      document.getElementById("resultat").innerHTML += "réponse en cours<br />";
    if(xhr.readyState == 4) {
      if(xhr.status == 200)
        document.getElementById("resultat").innerHTML += "Réponse : " + xhr.responseText+"<br />";
      else
        document.getElementById("resultat").innerHTML += "Error code " + xhr.status+"<br />";
    }
  }

  xhr.open("GET", "test.php", true);
  xhr.send(null);
}
```

HTTP GET ou POST

- GET pour récupérer des données
  - Les informations sont passées dans l'url
    - www.google.fr/index.php?q=3&req=fdg
  - Les requêtes GET doivent pouvoir être bookmarkées ou mises en cache
    - Elle ne doivent donc pas provoquer de mises à jour sur le serveur
- POST pour envoyer des données
  - Pour tout ce qui ne correspond pas à un GET

Inconvénients

- JavaScript doit être activé
- Les données chargées de façon dynamique ne font pas vraiment 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 :
  - on peut recharger une page mais plus difficilement annuler des modifications faites par Javascript

JSON
JAVASCRIPT OBJECT NOTATION

Cours Nouvelles Technologies du web

JSON ?

- Format d'échange de données.

- Objectifs :
  - Simple.
  - Extensible.
  - Ouvert.
  - Lisible par un humain.

- Similaire à
  - la définition des objets Javascript
  - Les tableaux associatifs 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":"Laurent",
  "Age":53,
  "Adresse": {"rue":"12 rut du pot",
              "cp":"75016","ville":"Paris"},
  "notes": [1, 2, 4, 8, 16, 32]
}
```

JSON et Javascript

- JSON peut être utilisé directement :
  - Inclusion dans du HTML
    - <script …> var data = JSONdata; </script>
  - Peut être converti en un objet Javascript
    - responseData = eval('(' + responseText + ')');

JSON ou XML ?

- JSON est très utilisé avec AJAX (le X de AJAX est pour XML)

- JSON :
  - {"nom": "Guillaume", "prenom": "Jean-Loup"}

- XML :
  - <?xml version='1.0‘ encoding='UTF-8'?>
  - <element>
  - <nom>Guillaume</nom>
  - <prenom>Jean-Loup</prenom>
  - </element>

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) {
          var doc = eval('(' + req.responseText + ')');
     }
}
req.send(null);
```

Exemple d'utilisation de JSON

- Coté serveur :
  - Génération de JSON à partir d'objets en Php ou JAVA
- L'échange de données :
  - Envoi avec Ajax

```
<?php
   $arr = array ('a'=>1,'b'=>2,'c'=>3,'d'=>4,'e'=>5);
   echo json_encode($arr);
?>

Affiche :
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

JAVASCRIPT OPTIMISATION DE CODE

Optimisation

- Librairies :
  - Utiliser des outils pour combiner plusieurs fichiers Javascript ou CSS afin de créer un seul gros fichier pour diminuer la charge sur le serveur
    - Une seule requête
    - Le fichier peut être mis en cache
- Eviter les appels multiples :
  - Mieux vaut faire une seule grosse requête au début que plein de petites requêtes.

Un navigateur

- Fonctionne mais :
  - Problèmes de sécurité
  - Problèmes de performance :
    - Ouvrir 20 onglets sous Firefox pose quelques problèmes
  - Pas prévu spécifiquement pour Ajax
    - Très gourmand en ressources
    - Modification du DOM et autres
- Autant aider le navigateur en codant proprement

Du code correct avant tout !

- Ne pas forcément optimiser avant que ça ne marche :
  - Si ça ne fonctionne pas, on se moque que ça aille vite
  - Mais y penser avoir d'avoir 10k lignes de code

- Tester dans des situations réelles :
  - Réseau, client ou serveur lent (tester en local peut masquer de nombreux problèmes de performance)
  - Utilisation de Yslow avec Firebug

Optimisation

- Approches complémentaires :
  - Suivre des cours d'algorithmique
  - Comprendre que Javascript est un langage interprété :
    - pas de compilateur pour optimiser le code !
  - Tester différentes façons de faire la même chose

Algorithmique - exemple

- Calcul de Fibonacci en récursif :
- Exemple : calcul de fibonacci(30)
  - Retourne 832 040
  - Combien d'appels à la fonction fibonacci ?

```
var fibonacci = function (n) {
  return n < 2 ? n :
    fibonacci(n - 1) + fibonacci(n - 2);
};
```

Algorithmique - exemple

- Nombre d'appels : 2 692 537
- Pourquoi :
  - on recalcule de nombreuses fois les mêmes chose
  - autant tout stocker en mémoire et ne pas refaire les calculs

```
function fibonacci(n) {
  var fibo_tab = [0,1];
  var fibonacci_aux = function(n) {
    var result = fibo_tab[n];
    if (typeof result !== 'number') {
      result = fibonacci_aux(n-1)+fibonacci_aux(n-2);
      fibo_tab[n] = result;
    }
    return result;
  };
  return fibonacci_aux(n);
}
```

Langage interprété

```
var i;
for (i = 0; i < divs.length; i += 1) {
    divs[i].style.color = "black";
    divs[i].style.border = thickness + "px solid blue";
    divs[i].style.backgroundColor = "white";
}
```

- Que peut-on améliorer ?

Langage interprété

- Les choses "couteuses" :
  - Calcul de la longueur du tableau
  - Recherche de divs[i].style
  - Concaténation de chaîne

```
var border = thickness + 'px solid blue',
    nrDivs = divs.length,
    ds, i;
for (i = 0; i < nrDivs; i += 1) {
    ds = divs[i].style;
    ds.color = "black";
    ds.border = border;
    ds.backgroundColor = "white";
}
```

Langage interprété

- La plupart des compilateurs font (par défaut) :
  - La suppression de sous-expression communes
  - La suppression des invariants de boucle
  - …
- Mais pas Javascript !
  - Ne pas trafiquer le code à outrance non plus

Travailler sur des chaînes

- Qu'est ce qui est le plus rapide ?
  - str = 'test'; str += 'test';
  - str = 'test' + 'test';

- Répété 1 million de fois :
  - str = 'test'; str += 'test'; : 432 ms
  - str = 'test' + 'test'; : 70 ms
  - Résultat pas forcément intuitif…

Optimisation

- Ce qui est couteux :
  - Modification du DOM (Document Object Model)
  - InnerHTML en particulier est très couteux
    - Éviter de faire des obj.innerHTML += chaine
    - Plutôt construire le nouveau contenu et l'insérer en une fois

Optimiser ou ne pas optimiser ?

- Certains navigateurs ont des fonctions mal implémentées.
  - Deux versions d'une même navigateur peuvent présenter des caractéristiques très différentes.
  - Eviter les astuces qui peuvent fonctionner sur un navigateur mais pas sur un autre

- Plutôt penser à long terme.

Optimiser en connaissance

- Mesurer avec précision :
  - debut = new Date().valueOf();
  - fonction_a_evaluer ();
  - fin = new Date().valueOf();
  - duree = end_time - start_time;

- Attention :
  - Un seul essai ne veut rien dire en général.
  - Tester avec différents paramètres (complexité)

JAVASCRIPT COOKIES

Cours Nouvelles Technologies du web

Les Cookies

- Chaîne de caractères pouvant être écrite sur le disque dur.
  - A un endroit défini.
  - Ne peut être lue que par le seul serveur qui l'a générée.

- Que faire avec un cookie
  - Transmettre des valeurs d'une page HTML à une autre.
    - Exemple : gestion d'un caddie sur un site de vente en ligne.
  - Personnaliser les pages présentées à l'utilisateur.

Les Cookies

- Limitations (théoriques, en pratique c'est faux) :
  - Limites en nombre : Pas plus de 20 cookies par serveur.
  - Limites en taille : un cookie ne peut excéder 4 Ko.
  - Limites du poste client : Un poste client ne peut stocker plus de 300 cookies en tout.

- Stockage - dépend du navigateur
  - IE :
    - C:\Users\admin\AppData\Local\Microsoft\Windows\Temporary Internet Files
  - Firefox :
    - …

Les Cookies - structure

- Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure
  - Nom=Contenu;
    - En-tête du cookie : deux variables suivies d'un ";"
    - La variable Nom contient le nom à donner au cookie.
    - La variable Contenu contient le contenu du cookie
    - Exemple mon_cookie="oui:visite"

Les Cookies - structure

- Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure
  - Expires= expdate;
    - Le mot réservé expires suivi du signe "=" (égal).
    - Date à laquelle le cookie sera supprimé du disque dur.
    - Format :
      - Wdy, DD-Mon-YYYY HH:MM:SS GMT
    - A générer avec les fonctions de l'objet Javascript Date
    - Règle générale : indiquer un délai en nombre de jours (ou d'années) avant disparition du Cookie.

Les Cookies - structure

- Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure
  - path=Chemin;
    - chemin de la page qui a créé le cookie.
  - domain=NomDeDomaine;
    - le nom du domaine de la page.
  - secure (false par défaut)
    - "true" (HHTPS) ou "false" (HTTP) : protocole utilisé.
  - path, domain et secure sont facultatifs.

Un exemple – fnac.com

- 27 cookies, dont :
  - Nom=FMBasketProductsCount
    - Contenu=0
    - Domaine=.fnac.com
    - Chemin=/
    - Expire=mercredi 2 avril 2014 23:59:49
  - Nom=soldes
    - Contenu=06%2F01%2F2010+12%3A40%3A09
    - Domaine=.fnac.com
    - Chemin=/
    - Expire=lundi 6 janvier 2020 12:39:34
  - …

Les Cookies - écriture

- propriété de l'objet document :
  - document.cookie = Nom + "=" + Contenu + "; expires=" + expdate.toGMTString() ;

```
// creation des variables
var Nom = "MonCookie" ;
var Contenu = "contenu" ;
var expdate = new Date () ;

// modification de la date d'expiration (10 jours)
expdate.setTime (expdate.getTime() + ( 10 * 24 * 60 * 60 * 1000)) ;

// ecriture du cookie sur le disque
document.cookie = Nom + "=" + Contenu + "; expires=" +
   expdate.toGMTString() ;
```

Les Cookies - lecture

- Accéder à la propriété cookie de l'objet document :
  - Document.cookie

```
var lesCookies ;
lesCookies = document.cookie ;
```

Les Cookies - modification

- Modification d'un cookie
  - Modifier le contenu de la variable Contenu puis réécrire le cookie sur le disque dur du client

```
Contenu = "Le cookie a été modifié..." ;
document.cookie = Nom + "=" + Contenu + "; expires=" +
   expdate.toGMTString() ;
```

Les Cookies - suppression

- Modifier la date de péremption :
  - Mettre une date dans le passé.
  - Le navigateur se charge du reste

```
// on enlève une seconde a la date courante
expdate.setTime (expdate.getTime() - (1000)) ;

// écriture sur le disque
document.cookie = Nom + "=" + Contenu + "; expires=" +
   expdate.toGMTString() ;
```

A faire en TME / toujours

- Vérifier vos fonctions récursives.
- Ecrire des fonctions pour faire du profiling de code.
- Tester plusieurs approches.

Un peu de temps à perdre

- http://steve-yegge.blogspot.fr/2006/03/execution-in-kingdom-of-nouns.html