

Javascript, HTML dynamique

version 2

Animateur : Alexandre Beaugrand
Contact : alex.beaugrand@gmail.com
Site : www.alexandrebeaugrand.com



Objectifs et prérequis

- Objectif
 - Prendre en main javascript pour agir sur vos pages web, les rendre dynamiques et interactives
 - Manipuler des données XML et utiliser AJAX
- Prérequis
 - Bonnes connaissances du HTML
 - Connaissances de base de la programmation

Plan de cours

- 1) Les technologies du Web
 - 2) Le langage JavaScript
 - 3) Manipulation du DOM XML
 - 4) Événements et données
 - 5) Interaction avec les feuilles de style en cascade
 - 6) Gestion de formulaires HTML
 - 7) Ajax
 - 8) Besoins divers
- Annexe : Introduction à JQuery



Chapitre 1

Les technologies du Web

Dans cette partie

- Les composants Web : HTML, XHTML, CSS, Javascript, Flash, Java... Les protocoles HTTP, HTTPS.
- Présentation et caractéristiques
- Positionnement et illustration des technologies DHTML.
- Utilisation et importance de JavaScript sur les sites web 2.0. Impact de la conformité XHTML sur l'utilisation dynamique de Javascript.

Dans cette partie

- Présentation des types de navigateurs et leur impact sur la portabilité des programmes JavaScript.
- Impact de Javascript sur l'accessibilité et le référencement.
- Les outils de développement (éditeur, débogueur...).

Composants web

- HTML et XHTML
- CSS
- PHP
- Javascript
- Flash
- Java
- HTTP
- HTTPS

Présentation

- Créé par Brenden Eich pour Netscape
- ECMAScript



Caractéristiques

- Javascript permet de modifier les balises d'une page, en agissant sur leurs propriétés, notamment la propriété style
- Il peut écouter et réagir aux évènements
- Syntaxe proche du C, du Java, et de l'actionsript, orienté objet et de type prototypé
- Exécuté côté client, c'est un langage interprété par un navigateur (donc dépendant de celui ci)
- Chaque instruction se termine par un ; optionnel. La casse est prise en compte même pour le nom des fonctions, les espaces blancs sont ignorés

Technologies DHTML

- DHTML = Dynamic HTML
- Désigne l'ensemble des techniques utilisées par l'auteur d'une page web pour que celle-ci soit capable de se modifier elle-même en cours de consultation

Exemples d'utilisation

- Formulaires dynamiques (comboboxs dépendantes)
- Vérifications de formulaires
- Modifications d'une page sans rafraichissement (Google maps)
- Images apparaissant progressivement ou changeant au survol
- Applications Ajax
- Menus dynamiques
- Horloge animée, affichage de la date actuelle
- Etc.

Javascript pour le web 2.0

- Web 2.0 = évolution du web
 - Participation (contenus)
 - Interactions (commentaires, réseaux sociaux)
 - Interfaces simples
 - Standards
 - Design...
- Utilisation de technologies
 - Javascript
 - AJAX
 - Web Services
 - RSS...

Navigateurs et versions de javascript

- Variantes de Javascript par Netscape et Microsoft
- Langage portable (norme ECMASCRIPT)
- Problème d'interprétation des modèles d'objets (DOM - Document Object Model) fournis par les navigateurs

Performances selon les navigateurs



- Source : <http://www.pcinpact.com/news/49706-comparatif-chrome-peacekeeper-safari-firefox.htm>

Javascript et accessibilité

- Recommandation W3C sur l'accessibilité (anglais) :
 - <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/full-checklist>
- Traduction en Français :
 - <http://www.la-grange.net/w3c/wcag1/full-checklist.html>
- *6.3 Assurez-vous que les pages sont utilisables quand les scripts, applets, ou tout autre objets programmés sont désactivés ou non supportés. Si ce n'est pas possible, fournissez une information équivalente sur une page alternative et accessible.*
- *6.4 Pour les scripts et les applets, assurez-vous que les gestionnaires d'évènements sont indépendants du périphérique d'entrée.*

Javascript et référencement

- Les spiders se servent de textes et liens pour le référencement
- Impossibilité d'interpréter le javascript

- Des astuces existent, par exemple :

`lien`

- Peut être remplacé par :

`<a href="http://www.site.com"
onclick="windows.open(this.href); return false;">lien`

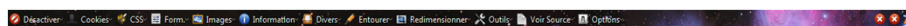
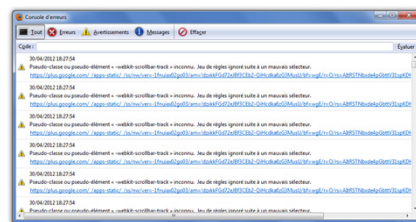
Outils de développement

- Editeurs de textes

- Mozilla Firefox

- Console
- Web developer toolbar

- <https://addons.mozilla.org/fr/firefox/addon/web-developer/>



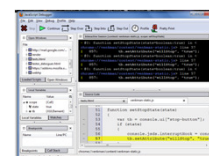
- Firebug

- <https://addons.mozilla.org/fr/firefox/addon/firebug/>



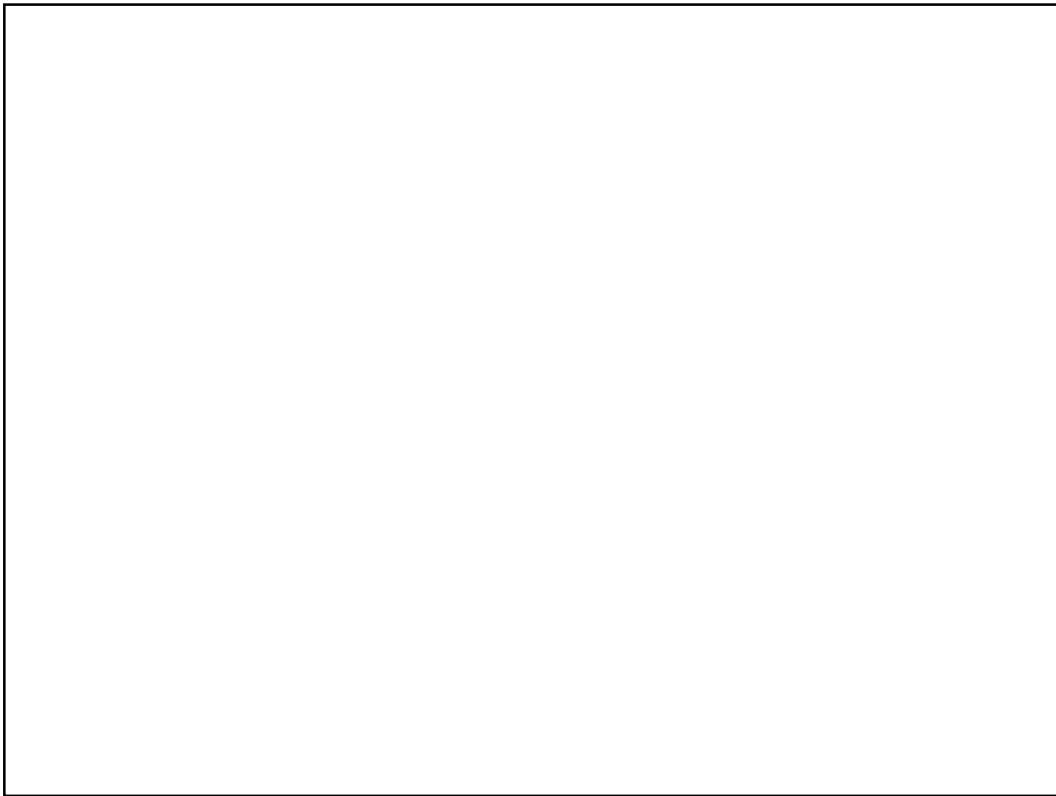
- Dom inspector (outils / inspecteur DOM):

- <https://addons.mozilla.org/en-US/firefox/addon/dom-inspector-6622/>




- Venkman Javascript Debugger (outils / Javascript Debugger)

- <https://addons.mozilla.org/fr/firefox/addon/javascript-debugger/>



Formation - Création web



Alexandre Beaugrand

Chapitre 2

Le langage JavaScript

Dans cette partie

- Où mettre son code ?
- Premières instructions, commentaires
- Déclaration de variables
- Types de données (Number, Boolean, Date, Math, String, Array). Typage et conversion de type. Détection de type avec typeof
- Opérateurs
- Structures de contrôle
- Boucles (for, while...)
- Fonctions et paramètres
- Codage sécurisé : gestion d'erreurs, exceptions.

Dans cette partie

- Rappels sur les concepts objets. Développement Objet en JavaScript : création de classes (méthodes, propriétés). Constructeur. Surcharge. Usage des mots réservés prototype, this. Création d'instance. Extension des objets prédéfinis. Porté des variables
- Utilisation du format JSON pour la création de classes
- Les objets prédéfinis du langage (Array, Date, Math, String, Regexp ...) et leur utilisation
- Travaux pratiques
 - *Mise en oeuvre des différentes fonctionnalités Javascript à travers de multiples travaux pratiques*

Où mettre son code ?

- Directement sur la page (dans head ou body)

```
<script type="text/javascript"><!--
```

Code

```
// --></script>
```

- Dans un fichier externe

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

- Dans une balise (pour écouter des evenements)

```
<a href="#" onClick='code'>Cliquez ici</a>
```

```
<a href="javascript:code">lien</a>
```

Premières instructions

- Pop up :

```
alert("helloworld");
```

- Ecrire dans la page :

```
document.write("Helloworld");
```

- Ecrire dans la console (installer le module FireBug dans firefox, et penser à l'activer) :

```
console.log('test');
```

- **Pratique : tester les 3 modes d'intégration avec ces fonctions**

Commentaires

- Sur une ligne

`//mon commentaire`

- Sur plusieurs lignes

`/* mon commentaire sur
plusieurs lignes */`

Variables

- Les variables ont un `nom` (commence par une lettre ou `_`, contient chiffres, lettres, `_` et `&` sans espaces)
- Elles sont liées à des `valeurs modifiables` au cours du script
- On peut utiliser le mot clé `var` pour les déclarer (conseillé)
- Elles ont un `type`, qui n'a pas besoin d'être précisé

Types de données

- Nombres : entiers (int), à virgule (float)
- Chaînes de caractères (string) : encadrés de guillemets
- Booléens (boolean) : 0 (false) ou 1 (true)
- Null (pas de valeur)
- Function
- Objets : standard (Object), Tableau (Array), Date (Date), etc.
- Exemples

```
var maVariable=4; //entier
var maVariable2='chaine'; //chaine
var o={prop1:42, prop2:'chaine'};//objet standard
var a=new Array(19,24); //objet tableau
var v=null; // pas de valeur
```

Typage, conversion, détection de type

- Conversion par fonctions :
 - [parseInt\(\)](#) : variable vers nombre
 - [parseFloat\(\)](#) : variable vers nombre décimal
- Conversion par casting : on entoure une valeur de parenthèse précédées du type voulu
 - Exemple : `console.log(Boolean("A"));`
- Détection de type : `typeof()`
 - Exemple : `var v="test";`
`console.log(typeof(v));`

Opérateurs

- Arithmétiques :

+ - / % * % ++ --

- D'assignation :

= += -= *= /= %=

- De concaténation :

+

Opérateurs

- Comparaisons :

== === != < > >= <=

- Logiques :

&& || !

- Conditionnel :

var variable=(condition)?valeur1:valeur2

- *Pratique : exercice 2.1 – âge minimum*

Structures de contrôle

- If / If, else / If, else if, else :

```
if (condition1){  
    code 1  
}else if (condition2){  
    code 2  
}else{  
    code 3  
}
```

Structures de contrôle

- switch :

```
switch(variable){  
    case valeur1:  
        code 1  
        break;  
    case valeur2:  
        code 2  
        break;  
    default:  
        code 3  
}
```


Boucles

- For :

```
var i=0;
for (i=valeurDepart;i<=valeurArrivee;i++){
    code
}
```

- While :

```
var i=0;
while (i<=5){
    code
    i++;
}
```

Pratique : exercice 2.2 - mois

Boucles

- For, in :

```
for (variable in objet){
    code
}
```

- Exemple :

```
var x;
var elements = new Array("e1","e2","e3");
for (x in elements){
    document.write(elements[x] + "<br />");
}
```

Break and continue

- Break, au sein d'une boucle, en fait sortir
- Continue, au sein d'une boucle, passe à l'itération suivante

```
var i=0
for (i=0;i<=10;i++){
    if (i==3){
        continue;
    }
    document.write(i+'<br/>');
}
```

- *Pratique : tester cet exemple et remplacer continue par break*

Fonctions

- Pour définir une fonction :

```
function nomDeFonction(){
    code
}
```

- Pour utiliser une fonction :

```
nomDeFonction();
```

- *Pratique : faire une fonction qui affiche “Bonjour” dans la page*

Paramètres de fonction

- On peut passer des paramètres à une fonction:

```
function nomDeFonction(param1,param2){  
    code  
}
```

- On l'appelle en lui donnant le même nombre de paramètres :

```
nomDeFonction(param1,param2);
```

- On peut passer des paramètres optionnels en plaçant un test en début de fonction :

```
if ( param === undefined ) {var param = 'salut';}
```

Pratique : Exercice 2.3 - salutations

Codage sécurisé : erreurs, exceptions

- On peut utiliser les blocs try...catch pour détecter des erreurs

```
try{  
    alert(eval("20+*5")); //code avec erreur  
}catch(err){  
    document.writeln("Erreur : " + err.name+"<BR/>")  
    document.writeln("Description : " + err.message)  
}
```

- On peut « lancer » nos propres erreurs avec throw

```
throw new Error("mon exception");
```

Programmation orientée objet

- EtreVivant
 - Animal
 - Mammifère
 - Chat
 - nom : Viking
 - race : Siamois
 - miauler()
 - Chien
 - Reptile
 - Lézard
 - ...
- Les objets sont des structures de données
 - Hierarchisés
 - Ayant des propriétés (variables) et des méthodes (fonctions)

Le chat :
EtreVivant.Animal.Mammifère.Chat

Son nom :
EtreVivant.Animal.Mammifère.Chat.nom

Le faire miauler :
EtreVivant.Animal.Mammifère.Chat.miauler()

Programmation orientée objet

- 3 principes importants :
- Encapsulation
 - Propriétés et méthodes sont liées dans le même objet
- Héritage
 - Un objet dérive d'un autre (son ancêtre), et hérite également de ses propriétés et méthodes, en plus des siennes
- Polymorphisme
 - Un objet peut librement modifier les méthodes de ses ancêtres
 - Exemple : bateau, voiture, avion auront une méthode avancer() héritée de leur ancêtre véhicule, mais la redéfinirons

Développement objet en Javascript

- Si on place une image avec name="A" on a la hiérarchie :
 - window
 - document
 - images
 - 0
 - name="A"
 - src="a.jpg"
 - forms
 - ...
- La première image :
window.document.images.A
ou
window.document.images[0]
- Son attribut width (lecture / écriture) :
window.document.images.A.width
- *Pratique : exercice 2.4 – image changeante*

Création d'instances

- On peut créer des objets prédéfinis avec le mot clé `new` et le type de l'objet, selon la forme :
`var nom_variable = new Type(param_optionnel);`
- Exemples :
`var v = new Number(4.05); // équivaut à var v = 4.05`
`var v = new String("test"); // équivaut à var v = "test"`
- On peut utiliser ses propriétés et ses méthodes avec le `point`
`var v = new String("test");`
`alert(v.length); //accès à la propriété length`
`document.write(v.bold()); //utilisation de la méthode bold()`

Développement objet en Javascript

- On peut créer nos propres objets standards regroupants propriétés et méthodes, avec le mot clé `new` et le type `Object` :

```
var o=new Object();
o.propriete='valeur1';
o.methode=function(){alert('ok')};
var mafonction=function(){alert('ok2')};
o.methode2=mafonction;

document.write('Propriété : '+o.propriete);
o.methode();
o.methode2();
```

Utilisation de JSON

- On peut aussi déclarer des objets en utilisant la notation JSON :

```
var o={
    propriete:'valeur1',
    methode:function(){alert('ok')}
}

//utilisation
alert(o.propriete);
o.methode();
```

Classes

- Contrairement à PHP ou Java, il n'existe pas de mot clé `class` !
- On peut faire un équivalent de classes avec le mot clé `function`

```
function maClasse(){  
    //propriété  
    this.nom = "Monsieur";  
  
    //méthode  
    this.saluer = function(){ alert("Bonjour "+ this.nom); }  
}
```

- Utilisation :

```
var o = new maClasse(); //création d'un objet maClasse  
alert(o.nom); //accès à la propriété nom  
o.saluer(); //utilisation de la méthode saluer()
```

This

- Le mot clé `this` sert à faire référence à l'objet qui invoque la fonction « classe »
- Dans l'exemple suivant, `this` fait référence à l'objet `o`

```
function maClasse(){  
    this.nom = "Monsieur";  
    this.saluer = function(){ alert("Bonjour "+ this.nom); }  
}  
var o = new maClasse();
```

Paramètres et constructeur

- On peut passer des paramètres à notre classe, qui agit alors comme un constructeur en initialisant des propriétés
- Il est alors intéressant de séparer **déclaration** et **affectation**

```
function maClasse(lenom){  
    var nom; //déclaration de variable  
    this.saluer = function(){ alert("Bonjour "+ this.nom); }  
    this.nom = lenom; //affectation d'une valeur  
}
```

- Instanciation de l'objet

```
var o = new maClasse("Monsieur");
```

- *Pratique : exercice 2.5 – chat : questions 1 à 5*

Surcharge

- On peut redéfinir les méthodes d'un objet, c'est la surcharge

```
function maClasse(){  
    this.saluer = function(){ alert("Bonjour"); }  
}  
  
var o = new maClasse();    o.saluer(); //1er test  
o.saluer=function(){ alert("Au revoir"); } //surcharge  
o.saluer(); //2nd test
```

- Plus élégamment, il est souvent utile de définir une méthode de nom différent et d'appeler l'ancienne fonction en son sein avant de la compléter

- *Pratique : exercice 2.5 – chat : question 6*

Extensions d'objets prédéfinis

- Il est possible d'ajouter des propriétés et des méthodes à des objets après qu'il aient été instanciés
- Exemple :

```
function cercle(){ }  
var petit_cercle=new cercle();  
var grand_cercle=new cercle();  
petit_cercle.pi=3.14159; //extension  
alert(petit_cercle.pi);  
alert(grand_cercle.pi);
```

Pratique : exercice 2.5 – chat : questions 7 à 8

Prototype

- Les objets ont la propriété `prototype`, qui permet de redéfinir leur « type » par l'ajout de propriété et de méthodes à tous les objets

```
typeObjet.prototype.nom=valeur;
```

- La valeur peut être une variable ou un nom de fonction
- Exemple :

```
function cercle(){ }  
var petit_cercle=new cercle();  
var grand_cercle=new cercle();  
cercle.prototype.pi=3.14159;  
alert(petit_cercle.pi+' '+grand_cercle.pi);
```

Pratique : exercice 2.5 – chat : question 9

Portée des variables

- Il n'y a pas d'accesseurs (public, private, etc.) en javascript
- Propriétés
 - Publiques : utiliser `this.mavariante`. Accessibles et modifiables partout.
 - Privées : utiliser `var` pour la déclarer dans l'objet. Elles ne sont accessibles et modifiables que par les fonctions privées et protégées.
- Méthodes
 - Publiques : utiliser la déclaration via `prototype`. Modifiables partout.
 - Privées : déclaration avec `var` dans le constructeur : `var functionName=function(){...}`. A appeler depuis les méthodes protégées.
 - Protégées : utiliser `this.methodName=function(){...}`. On peut aussi les appeler depuis du code extérieur à l'objet. On ne peut les changer.

Objet String : chaîne de caractères

- Les objets ont des propriétés et des méthodes
- Déclaration d'un objet String

```
var txt = new String();    ou    var txt="test";
```
- Exemple de propriété

```
document.write(txt.length);
```
- Exemple de méthode

```
document.write(txt.toUpperCase());
```
- Référence de String

```
http://www.w3schools.com/jsref/jsref\_obj\_string.asp
```

Pratique : exercice 2.6 – à l'envers

Objet String : chaîne de caractères

- Pour str et str2 étant deux chaînes de caractères, i, et j entiers :
- str.length // le nombre de lettres
- str.charAt(i) // caractère à la position i (1ère lettre = position 0)
- str.indexOf(str2) // 1ère position de str2 ou -1 si non trouvée
- str.lastIndexOf(str2) // idem mais avec la dernière position
- str.toLowerCase() // str en minuscules
- str.toUpperCase() // str en majuscules
- str.concat(str2) // str concaténée avec str2
- str.substring(i,j) // sous chaîne de str, de i inclus à j exclus
- str.substr(i,j) // sous chaîne de i jusqu'à j caractères
- str.split(str2) // tableau de str scindé à chaque str2

Pratique : exercice 2.7 – string

Objet Regexp

- Les expressions régulières sont des chaînes codées définissant un motif, c'est à dire décrivant d'autres chaînes
- Elles sont encadrées par des slashes suivis éventuellement d'un « modifier » : /expression/modifier
- Modifier : i (insensible à la casse), g (global), m (multiligne)

signe	description	signe	description
^ dans /^abc/	Commence par abc	\S dans /\S/	Tout sauf espaces blancs
\$ dans /abc\$/	Fini par abc	\w dans /\w/	Tout signe alphanum. et _
* dans /abc*/	abc suivi de 0 ou plus c	\W dans /\W/	Inverse du précédent
+ dans /abc+/	abc suivi de 1 ou plus c	\b dans /\b\b/	Toutes les lettres a seules
. dans /abc./	abc suivi d'un caractère	\d dans /\d\d/	Toutes chiffre entier
\s dans /\s/	Tout espace blanc	\D dans /\D.+D/	Caractères sans chiffres
- Dans a-z	Lettres de a à z	[] dans /[a-zA-Z]/	Minuscules ou majuscules

- Exemple de motifs : /Mamarque/g
/^\\w+@\\w+\\.\\w+\$/g

Objet Regexp

- Les motifs des expressions régulières sont plus ou moins précis
- Exemple de détection d'e-mail simple :
 - `/^\w+@\w+\.\w+$/g`
- Plus précis :
 - `/^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$/g`
- etc.

Pratique : Expressions régulières

- Il existe des méthodes de l'objet String et de l'Objet RegExp qui utilisent les expressions régulières. Pour str et str2 = chaînes et reg = expression régulière :
- `str.search(reg)` // position de la 1ère occurrence de reg ou -1
- `str.match(reg)` // tableau des occurrences de reg dans str
- `str.replace(reg,str2)` // chaîne avec les reg remplacées par str2
- `reg.exec(str)` // renvoie le premier élément correspondant ou null
- `Reg.test(str)` // renvoie vrai ou faux
- Exemple :

```
var exp=/toto/g;  
document.write(exp.test("bonjour toto"));
```

Ref : http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Pratique : exercice 2.8 – regExp

Objet Date

- Les dates sont souvent décrites en **millisecondes**. Elles font toutes références aux nombres de millisecondes s'étant écoulées depuis le 1er janvier 1970
- Pour la suite, définissons la variable moment =
"mois jour, année heures:minutes:secondes"
- Il y a 4 façons pour déclarer une Date :
 - **new Date()** //maintenant (en millisecondes)
 - **new Date(millisecondes)** //à la date indiquée en millisecondes
 - **new Date(moment)** // ex : "October 13, 1975 11:13:00"
 - **new Date(année, mois, jours, heures, minutes, secondes, millisecondes)**
- Ref : http://www.w3schools.com/jsref/jsref_obj_date.asp

Objet Date

- Pour d=date, moment = "mois jour, année heures:minutes:secondes" et X ayant pour valeur possibles :
- FullYear (année), Month(mois), Date(jour du mois), Day (jour de la semaine – 0 = dimanche), Hours (heures de 0 à 23), Minuts (minutes), Seconds (secondes), Milliseconds (millisecondes)
- On a les méthodes suivantes :
 - **d.getX()** // retourne X
 - **d.setX(valeur)** // modifie X
 - **d.getUTFX()** // retourne X pour Greenwich
 - **d.setUTFX()** // modifie X pour Greenwich
 - **Date.parse(moment)** // nombre de millisec. depuis le 1/1/1970
 - **Date.UTC(a,m,j,h,s,m)** // idem, mais paramètres différents

Pratique : exercice 2.9 – date

Objet Math

- L'objet Math à des propriétés et méthodes utiles pour des calculs
- Propriétés :
 - Math.PI //retourne PI
- Méthodes (pour x,y,n = nombres) :
 - Math.abs(x) // valeur absolue
 - Math.cos(x) // cosinus. Il existe acos pour l'inverse. sin et tan également.
 - Math.ceil(x) // arrondi supérieur. Il existe floor et round également.
 - Math.pow(x,y) // x puissance y. Sqrt retourne la racine
 - Math.random() // nombre « aléatoire » compris entre 0 et 1
 - Math.max(x,y,...,n) // valeur max. Il existe min également.
- Ref : http://www.w3schools.com/jsref/jsref_obj_math.asp

Pratique : exercice 2.10 – math

Objet Array

- Un Array est un tableau, qui contient de multiples valeurs et tous types, manipulables par des méthodes associées à l'objet Array
- Déclarations de tableaux :
 - Déclaration 1 (simple) : `var t=new Array(19,24);`
 - Déclaration 2 (simple) : `var t=[19,24];`
 - Déclaration 3 (simple) : `var t=new Array(); t[0]=19; t[1]=24;`
 - Déclaration 4 (associatif) : `var t=new Array() t["A"] = "valeur";`
 - Déclaration 5 (imbriqués) `var t=new Array(); t[0]=new Array('A');`

Accéder aux données d'un Array

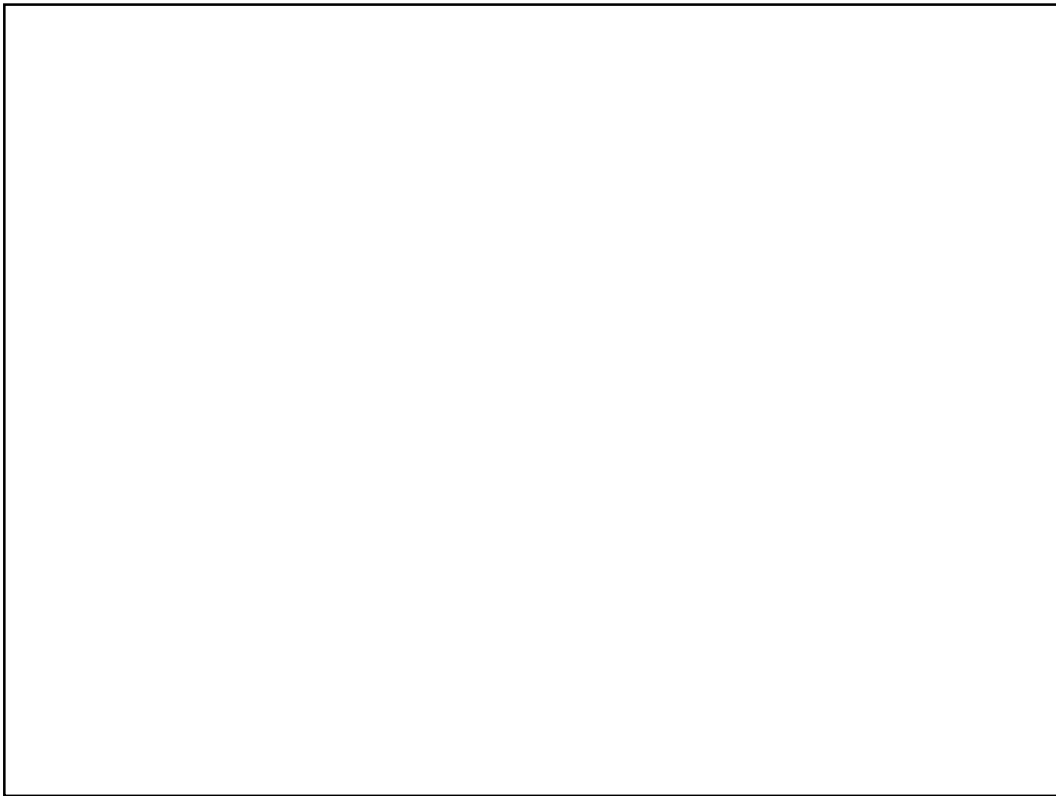
- Accès directs (avec t = objet Array) :
 - Tableau simple : `t[0];`
 - Tableau associatif : `t["A"];`
 - Tableaux imbriqués : `t[0]['A'];`
- Comme pour les objets Object, on peut utiliser des boucles for, while et surtout for in. Penser à imbriquer les boucles pour les tableaux imbriqués. Rappel :

```
var x;  
var elements = new Array("e1","e2","e3");  
for (x in elements){  
    document.write(elements[x] + "<br />");  
}
```


Méthodes d'un Array

- Pour t1, t2 = Array et v1, v2 = chaînes ou nombres
- `t1.length` // longueur de t1
- `t1.concat(t2)` // t1 concaténé à t2
- `t1.join()` // chaîne avec les éléments de t1 séparés par des ,
- `t1.pop()` // supprime le dernier élément de t1 et le retourne
- `t1.push(v1,v2)` // ajoute v1 et v2 à la fin de t1
- `t1.reverse()` // inverse l'ordre des éléments de t1
- Ref : http://www.w3schools.com/js/js_obj_array.asp

Pratique : exercice 2.11 – array



Formation - Création web



Alexandre Beaugrand

Chapitre 3

Manipulation du DOM XML

Dans cette partie

- Présentation du langage XML (éléments attributs).
- Implémentation des analyseurs XML
- Compatibilité des navigateurs
- Les objets du DOM (window, document, ...) et leur manipulation.
- Manipulation des objets du DOM (lecture, ajout, suppression, modification de noeuds).
- **Travaux pratiques**
 - *Familiarisation à la création, suppression, et sélection de noeuds DOM, ainsi qu'à l'affichage et la modification de leurs propriétés*

Le langage XML

- eXtensible Markup Language
- Sert à **contenir** et **transporter** des **données**
- Langage de **balises** comparable à l'HTML mais où nous pouvons inventer nos propres balises

Eléments et attributs XML

- Exemple de fichier XML :

```
<?xml version="1.0"?>
<voitures>
  <voiture couleur="bleue" annee="2000">Z4</voiture>
  <voiture couleur="rouge" annee="1998">F40</voiture>
  <voiture couleur="jaune" annee="2007">BMW</voiture>
</voitures>
```

- Un fichier XML comporte une balise racine, ici voitures
- La balise racine contient une série d'éléments : les **nœuds**
- Chaque nœud peut avoir **des attributs** et une **valeur**
- Si un élément n'a pas de valeur, on peut utiliser des balises autofermantes : `<voiture couleur="jaune" annee="2007" />`

Analyseurs XML des navigateurs

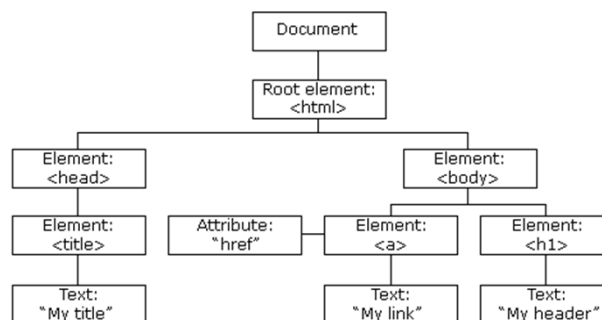
- Il est possible de visualiser les fichiers xml dans des navigateurs. Chaque navigateur dispose de son propre analyseur XML



- On trouve dans les références, la compatibilité des méthodes du DOM en fonction du navigateur :
- <http://fr.selfhtml.org/javascript/objets/node.htm>

Objets du DOM

- Le Document Object Model (DOM) est un standard du W3C pour hierarchiser le contenu d'un document XML ou HTML
- Comme pour le XML, une page HTML est constituée d'une série de nœuds ayant valeurs et attributs et des liens entre eux



Compatibilité des navigateurs

- Le DOM est interprété différemment en fonction des navigateurs
- Si certaines méthodes fonctionnent pour tous, il est parfois nécessaire d'adapter le code en fonction des navigateurs
- Compatibilités :
- http://www.w3schools.com/html/dom/dom_methods.asp
- http://www.quirksmode.org/dom/w3c_css.html
- On peut utiliser le quirksmode d'IE pour vérifier la compatibilité d'un script avec un navigateur IE plus ancien (sous IE8 faire F12 ou outils / outils de dev. / onglet scripts / Mode document en haut)

Compatibilité des navigateurs

- Il est possible de mettre des instructions conditionnelles en fonction du navigateur (pas du mode document) :

```
<!--[if (IE 5)|(IE 6)|(IE 7)]>
    <script type="text/javascript">
        alert("<= ie7");
    </script>
<![endif]-->
<!--[if (gt IE 7)|!(IE)]><!-->
    <script type="text/javascript">
        alert("pas ie7");
    </script>
<!--<![endif]-->
```

Compatibilité des navigateurs

- On peut aussi mettre des instructions conditionnelles via javascript :
- document.all n'existe que pour IE
- document.documentMode n'existe que pour IE et précise la version du document (5, 7, etc.)

```
if(document.all) {
    if(document.documentMode <= 7){
        alert("ie < 8");
    } else{    alert(">= ie8");    }
}else{
    alert("pas IE");
}
```

Accès aux noeuds

- On peut faire référence à des noeuds de plusieurs façons :
- Par sa balise : `document.getElementsByTagName("balise")`
- Par son nom : `document.getElementsByName("nom")`
- Par son id : `document.getElementById("id")`
- Par sa classe : `document.getElementsByClassName("nom")`
- Eviter `getElementsByClassName()` car non géré par IE
- Attention : pour pouvoir accéder à un nœud, il faut que le code javascript soit exécuté APRES le chargement de la page. Pour ce faire, on appelle une fonction depuis la balise BODY comme suit : `<BODY onLoad="init();" >` et on définit la fonction correspondante dans notre script

Pratique : exercice 3.1 – nœuds – questions 1 à 3

HTML collections

- Lorsqu'on utilise les fonctions `getElementsByTagName` ou `getElementsByName`, ou `getElementsByClassName`, javascript renvoie une `HTMLCollection`, c'est à dire une `liste de noeuds HTML`
- Pour accéder au premier noeud, on peut utiliser : la méthode `item(numero)` ou bien utiliser les `[crochets]` comme les tableaux
- Exemples :
`getElementsByTagName('p').item(0); //retourne le 1er noeud`
`getElementsByTagName('p')[0] //idem`
- Pour récupérer le nombre d'éléments de cette collection, on utilise la propriété `length`
- Exemple :
`getElementsByTagName('p').length;`

Propriétés des noeuds

- `noeud.nodeType` → type de nœud
 - 1 : Élément
 - 2 : Attribut
 - 3 : Texte
 - 8 : Commentaire
 - 9 : Document
- `noeud.nodeName` → nom du nœud
 - Retourne la valeur si nœud est un attribut ou un élément.
 - Retourne #text ou #document le cas échéant.
 - Lecture seulement.
- `noeud.nodeValue` → valeur du nœud
 - undefined si nœud est un élément.
 - Utile pour les textes et attributs.
 - Pour les nœuds textes, nodeValue est l'équivalent de data.
 - Ré-écriture possible si le nœud existe déjà : `noeud.nodeValue=valeur`;

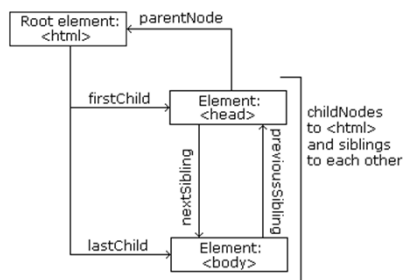
Propriétés des noeuds

- `noeud.attributes` → liste des attributs
 - Cette méthode récupère un nœud de type attribut. Elle est déconseillée car très incompatible ("a bloody mess") : http://www.quirksmode.org/dom/w3c_core.html#attributes
 - Il est préférable d'utiliser `getAttribute()` ou `setAttribute()` (voir ci-après)
 - `attributes['id']` : attribut 'id' (cette méthode est presque compatible)
 - `attributes.id` : attribut 'id' (ne fonctionne pas avec class sur IE par exemple)
 - `attributes[0]` : premier attribut (non reconnu avant IE8)
- `noeud.innerHTML` → le contenu du nœud
 - Ecriture possible : `noeud.innerHTML=valeur`

Pratique : exercice 3.1 – nœuds – question 4

Hierarchie des noeuds

- `noeud.childNodes` → les sous noeuds du noeud
- `noeud.childNodes.length` → nombre de sous noeuds du noeud
- `noeud.childNodes[0]` → premier sous noeud du noeud
- `noeud.parentNode` → nom du parent
- `noeud.firstChild` → premier sous noeud
- `noeud.lastChild` → dernier sous noeud
- `noeud.previousSibling` → noeud avant
- `noeud.nextSibling` → noeud après



Pratique : exercice 3.1 – noeuds – question 5

Selection de noeuds imbriqués

- On peut utiliser plusieurs fonctions de sélection pour sélectionner des noeuds imbriqués

```
document.getElementById("monform")
    .getElementsByTagName("input")
```

- Attention, on ne peut appliquer les méthodes de sélection que sur les noeuds seuls, pas sur les collections de noeuds (d'où l'utilisation de `item(0)` dans l'exemple suivant :

```
document.getElementsByTagName("form")
    .item(0)
    .getElementsByTagName("input")
```

Ajout de noeud

- 3 étapes :
 - Création d'un noeud
 - Localisation de l'endroit où l'insérer
 - Insertion
- 1^{ère} étape : création du nœud
 - Texte : `document.createTextNode()`
`var texte = document.createTextNode("Nouveau texte");`
 - Element : `document.createElement()` et `noeud.setAttribute()`
`var lien = document.createElement('a');`
`lien.setAttribute('href', 'mapage.html');`
`lien.innerHTML = "bouton";`

Ajout de noeud

- 2^{nde} étape : sélection de la destination (on pointe sur un nœud)
 - Utiliser les méthodes d'accès vues précédemment
`var dest = document.getElementsByTagName("p").item(0);`
- 3^{ème} étape : insertion
 - Comme dernier enfant du nœud de destination :
`dest.appendChild(lien);`
 - Juste avant le premier enfant du nœud de destination :
`dest.insertBefore(texte, dest.firstChild);`

Suppression de noeud

- 2 étapes :
 - Localisation du nœud où la suppression aura lieu
 - Suppression
- 1^{ère} étape : localisation

```
var noeud = document.getElementsByTagName("p").item(0);
```
- 2^{nde} étape : suppression

```
var disparu = noeud.removeChild(noeud.firstChild)
```

Modification de noeud

- Au niveau des attributs
 - `noeud.setAttribute(nom,valeur);` → ajoute un attribut
 - `noeud.getAttribute(nom);` → lit un attribut
 - `noeud.removeAttribute(nom);` → supprime un attribut
- Ne pas utiliser ces fonctions pour l'attribut "class", "style" et les attributs d'évènements (onClick, onLoad, etc.)
- Pour IE < 8, remplacer si besoin "class" par "className"

```
var nomClasse="class"; //FF et compatibles
if(document.all) { //IE
    if(document.documentMode <= 7){
        nomClasse="className"; //IE7 et <
    }
}
lien.setAttribute(nomClasse, 'en_rouge');
```

Modification de noeud

- Au niveau du contenu html
 - utiliser `innerHTML` en écriture :
`noeud.innerHTML= valeur`

Pratique : exercice 3.1 – noeuds – à finir

Autres problèmes liés à l'arbre DOM

- Suivant les navigateurs, les espaces blancs et les retours à la ligne peuvent être comptés comme des noeuds
- On peut dans ce cas utiliser des fonctions qui 'nettoient' les espaces blancs (exemple : `cleanWhitespace` sur <http://reference.sitepoint.com/javascript/domcore>)
- Il est parfois utile de considérer l'enfant d'un noeud. Par exemple, pour accéder au texte d'un paragraphe :

```
var paragraphes = document.getElementsByTagName('p');  
var paragraphe2=paragraphes[1].firstChild.nodeValue;
```

Le DOM, récapitulatif

- N'agir sur le DOM qu'après la fin de chargement de la page
- Ne pas utiliser `getElementsByClassName` mais `getElementById` et `getElementsByTagName`
- Ne pas utiliser `attributes[]` mais `getAttributes()` et `setAttributes()` sauf pour les id, les classes, les styles, et les attributs d'évènements
- Utiliser un code de détection de navigateur pour gérer les cas particulier où on ne peut utiliser un code générique



Chapitre 4

Événements et données

Dans cette partie

- Organisation des événements. Impact des événements sur les types de navigateurs et versions de DOM.
- Comment programmer des écouteurs sur des événements : méthode inline, méthode traditionnelle. Règles pour faire un codage multi-navigateurs.
- Détruire des écouteurs.
- Les traitements événementiels JavaScript : gestionnaires clavier, souris, événements liés aux formulaires

Dans cette partie

- L'objet Event et son utilisation.
- Travaux pratiques
 - *Programmation d'événements multiples sur les éléments formulaire, souris, clavier...*

Evènements

- Un évènement est une action qui peut être détectée par Javascript
- Chaque élément de page web peut lancer un évènement, et une fonction Javascript est alors exécutée
- On reconnaît un évènement car il commence par 'on' (onClick, onLoad, onMouseOver...)
- Références des Evènements :
 - http://www.w3schools.com/jsref/dom_obj_event.asp
 - <http://www.quirksmode.org/dom/events/index.html>

Ecouteurs d'évènements inline

- On peut ajouter un évènement à une balise, en lui disant quel code exécuter lors du déclenchement de l'évènement

- HTML :

```
<a onclick='clic_bouton(event)'>mon bouton</a>
```

- Javascript :

```
function clic_bouton(e){  
    alert('clic détecté');  
}
```

Objet Event

- Le paramètre passé aux fonctions réagissant aux évènements est de type event. On s'assure qu'il soit récupéré pour tous les navigateurs ainsi :

```
function clic_bouton(e){  
    if (!e) var e = window.event;  
}
```

- On peut l'exploiter grâce à ses propriétés, notamment :

- `e.type`
- `e.target` (compatibles) ou `e.srcElement` (IE)
- `e.keyCode` (compatibles) ou `e.which` (IE)
- [...] et toutes les propriétés propre à chaque type d'évènements

Ecouteurs d'évènements traditionnels

- On peut également éviter de placer un attribut sur la balise, et tout faire par javascript
- HTML

```
<body onLoad="init();"><a id='bt' href='#'>mon bouton</a></body>
```

- Javascript

```
function init(){  
    var bouton = document.getElementById("bt");  
    bouton.onclick=clic_bouton; //pas de ()  
}  
  
function clic_bouton(){  
    alert('clik détecté');  
}
```

Ecouteurs d'évènements avancés

- On peut aussi utiliser les fonctions avancées addEventListener() (compatibles) et attachEvent() (IE)

```
if(bouton.addEventListener){ //compatibles  
    bouton.addEventListener('click',clic_bouton,false);  
}else{  
    bouton.attachEvent('onclick',clic_bouton); //IE  
}  
  
function clic_bouton(){  
    alert('clik détecté');  
}
```

Évènements fenêtre

- Évènements fenêtre
 - `load, unload` → chargement, départ de la page
 - `resize` → redimensionnement
 - `scroll` → scroll de la page

Pratique : exercice 4.1 – évènements fenêtres

Évènements souris

- Évènements souris
 - `onclick, ondblclick` → clic, double clic
 - `onmouseup, onmousedown` → bouton levé, abaissé
 - `onmouseover, onmouseout` → entrée, sortie du pointeur
 - `onmousemove` → déplacement du pointeur

Pratique : exercice 4.2 – évènements souris

Évènements clavier

- Évènements clavier
 - `onkeydown`, `onkeypress` → touche enfoncée
 - `onkeyup` → touche relachée

Pratique : exercice 4.3 – évènements clavier

Évènements formulaire

- Évènements formulaire
 - `onfocus` , `onblur` → gain, perte de focus
 - `onselect` → selection de texte d'un input ou textarea
 - `onchange` → modification d'un élément
 - `onreset` → réinitialisation du formulaire
 - `onsubmit` → envoi du formulaire

Pratique : exercice 4.4 – évènements formulaire, et 4.5 - recopiage

Codage multi-navigateurs

- L'interprétation des événements dépend du navigateur. On doit donc se référer à la documentation afin de vérifier la compatibilité
- Il est utile d'utiliser des attributs pour gérer les événements. On privilégie les attributs universels afin d'assurer une compatibilité multi-navigateurs :
- <http://www.quirksmode.org/dom/events/index.html#t02>

Pratique : exercice 4.6 - event

Détruire des écouteurs

- Si on a placé un écouteur via une balise :
`noeud.removeAttribute("onclick");`
- Si on a placé un écouteur via la méthode traditionnelle :
`noeud.onclick=null;`
- Si on a placé un écouteur via la méthode avancée :

```
if(bouton.removeEventListener){
    bouton.removeEventListener('click', clic_bouton, false); //compatibles
}else{
    bouton.detachEvent('onclick', clic_bouton); //IE
}
```

Les évènements, récapitulatif

- Préférer placer les écouteurs en javascript plutôt qu'au niveau des balises html.
- Préférer retirer ou ajouter des écouteurs en utilisant la méthode traditionnelle ou avancée, plutôt que d'agir au niveau des attributs des balises html avec `getAttribute()` ou `setAttribute()`



Chapitre 5

Interaction avec les feuilles de style en cascade

Dans cette partie

- Rappel sur les feuilles de style en cascade, les outils pour les manipuler.
- Implémentation des CSS en tant que propriétés des objets du DOM.
- Modification directe des propriétés CSS des objets du DOM et de l'objet CSS stylesheets.
- **Travaux pratiques**
 - *Réalisation de pages simples afin de se familiariser à l'utilisation de feuilles de style et à leur manipulation à travers JavaScript.*

Les feuilles de style en cascade (CSS)

- Le CSS décrit l'affichage d'une page
- Organisation du code CSS :

```
Sélecteur {  
    Propriété1: valeur1;  
    Propriété2: valeur2;  
}
```

- le Sélecteur permet d'affecter la liste des propriétés / valeurs à un ou plusieurs éléments de la page
- Sélecteur = `E` → balises `E` affectées
- Sélecteur = `#E` → balises ayant l'attribut `id="E"` affectées
- Sélecteur = `.E` → balises ayant l'attribut `class="E"` affectées

CSS : principe d'insertion "en bloc"

- Insertion de CSS via la balise `<STYLE>` dans la partie `<HEAD>` :

```
<style>code css</style>
```

- Insertion par la balise `<LINK>` dans la partie `<HEAD>` :

```
<link rel="stylesheet" type="text/css" href="style.css" />
```

CSS : principes d'insertion inline

- On peut insérer du code css directement via les attributs `style`, `class`, et `id` d'une balise html qui sera affectée
- Principe de codage inline :

```
<balise style="propriété1:valeur1;propriété2:valeur2;"  
        class='nom_classe' id='nom_id' />
```

Insertion d'id, class et style inline

- Ajout de l'attribut `id` :

```
noeud.id="header"; → applique l'id "header"
```

- Ajout de l'attribut `class` :

```
noeud.className="bt"; → applique la classe "bt"
```

- Ajout de l'attribut `style` :

```
var style_complet ="border:5px solid black;background-color:red";
```

```
noeud.style.cssText=style_complet; → applique le style complet
```

- Ajout d'une propriété CSS particulière :

```
noeud.style.backgroundColor="blue"; → applique un fond bleu
```

Lecture d'id, class, et style inline

- Lecture de l'attribut id :

```
alert(noeud.id)
```

- Lecture de l'attribut classe :

```
alert(noeud.className)
```

- Lecture de l'attribut style :

```
alert(noeud.style.cssText)
```

- Lecture d'une propriété CSS particulière :

```
alert(noeud.style.backgroundColor)
```

Pratique : exercice 5.1 – styles inline

Alternatives pour id et class

- On peut utiliser les méthodes `setAttribute()` (écriture) et `getAttribute()` (lecture) également, mais il faut dans ce cas passer un paramètre différent en fonction du navigateur : `className` pour IE6 et IE7, et `class` pour les autres navigateurs

```
lien.setAttribute("id","mon_id");//compris par tous
```

```
lien.getAttribute("id");
```

```
lien.setAttribute(nomClasse,"bt");//attention nomClasse est une variable
```

```
alert(lien.getAttribute(nomClasse)); //idem
```

Alternatives pour style

- En fonction du navigateur, on peut accéder à une propriété CSS d'un nœud ainsi (seulement en lecture) :

```
if( window.getComputedStyle ) { //navigateurs compatibles
    alert(window.getComputedStyle(noeud,null).propriété);
} else if( noeud.currentStyle ) { //IE
    alert(noeud.currentStyle.propriété);
}
```

Style en bloc : objet CSS styleSheet

- Il est possible d'agir sur les feuilles CSS incluses dans la page, soit par la balise style, soit par la balise link. Chaque feuille liée et chaque bloc <script> correspond à un objet CSS styleSheet

- On accède à un objet stylesheet n° N de la façon suivante :

```
document.styleSheets[N]
```

- On peut récupérer le chemin du fichier css source le cas échéant :

```
document.styleSheets[N].href
```

- On peut désactiver un objet styleSheet via la propriété disabled :

```
document.styleSheets[0].disabled = true;
```


CSS styleSheet : lecture d'une règle

- IE (toutes versions) :
 - Valeur :

```
alert(document.styleSheets[0].rules[0].style.cssText);
```
 - Sélecteur :

```
alert(document.styleSheets[0].rules[0].selectorText);
```
- IE9 et autres navigateurs :
 - Valeur :

```
alert(document.styleSheets[0].cssRules[0].style.cssText);
```
 - Sélecteur :

```
alert(document.styleSheets[0].cssRules[0].selectorText);
```

CSS styleSheet : insertion d'une règle

- IE :

```
document.styleSheets[0].addRule('.bt','border:5px solid green');
```
- Autres navigateurs :

```
var noeud=document.styleSheets[0];  
noeud.insertRule('.bt{border:5px solid green}',noeud.cssRules.length);
```

CSS styleSheet : suppression d'une règle

- IE :

```
document.styleSheets[0].removeRule(0);
```

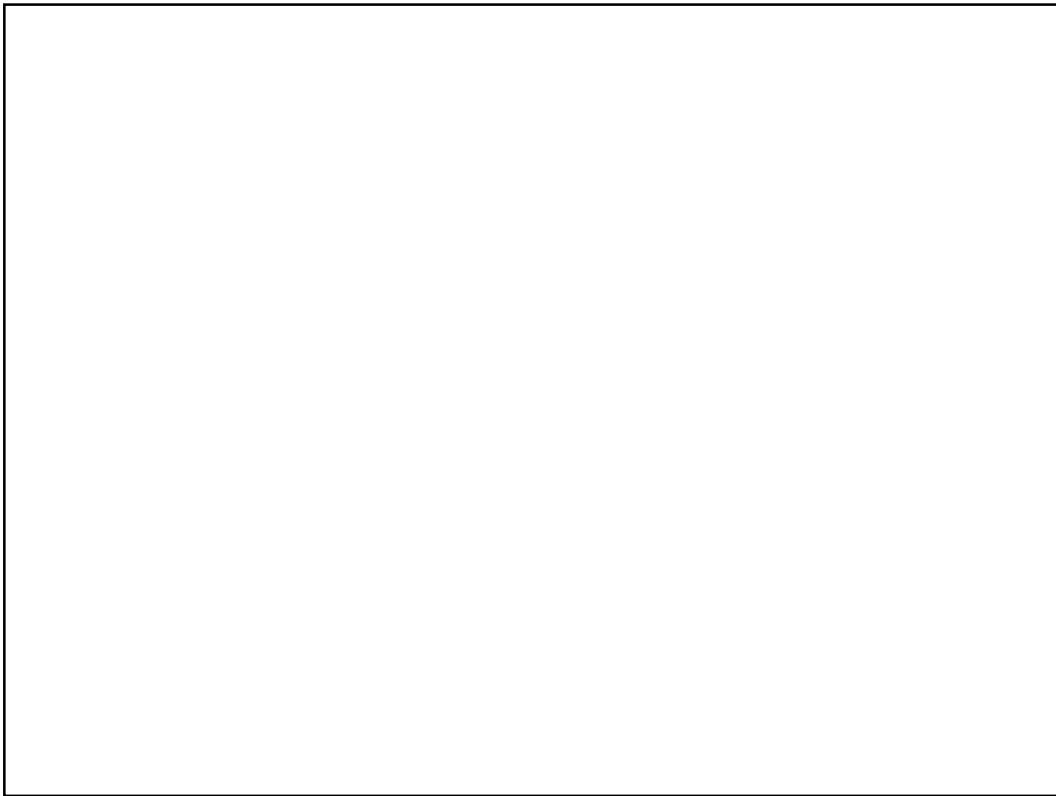
- Autres navigateurs :

```
document.styleSheets[0].deleteRule(0);
```


Pratique : exercice 5.2 – CSS styleSheet et exercice 5.3 – menu

Les CSS, récapitulatif

- Utiliser de préférence `noeud.id`, `noeud.className`, et `noeud.style.proprieteCSS` pour lire et définir les styles inline
- Éviter d'agir sur les feuilles de styles externes en javascript car la compatibilité est mauvaise



Formation - Création web



Alexandre Beaugrand

Chapitre 6

Gestion des formulaires HTML

Dans cette partie

- Manipulation de contenu de formulaires.
- Accès et modification dynamique des composants du formulaire : zone de saisie, cases à cocher, cases d'options, manipulation de listes déroulantes, textarea.
- Événements liés aux éléments de formulaire.
- Fonctions de validation de formulaire.
- Travaux pratiques
 - *Conception de fonctions personnalisées contrôlant les activités de l'utilisateur*

Balise form

`<form name="nom" method="X" action="Y">...balises...</form>`

- Avec X= POST ou GET et Y=chemin vers la page cible. Rajouter `enctype="multipart/form-data"` dans la balise ouvrante si envoi de fichiers
- Pour envoyer ou faire un reset d'un formulaire, on peut placer à l'intérieur les balises suivantes :
 - `<input type="submit" value="envoyer" />` → Envoi
 - `<input type="reset" value="remise à zéro" />` → Remise à zéro
- Événements liés à la balise form :
 - `onSubmit` → Détecte la soumission du formulaire
 - `onReset` → Détecte la réinitialisation

Button

```
<INPUT type="button" value="Mon bouton" name="bouton" />
```

- On peut utiliser cette notation pour créer un bouton générique

- Contrairement à submit et reset, il n'y a pas d'action prédéfinie pour ce bouton

- Il faut donc appliquer un écouteur à ce bouton

```
<INPUT type="button" value="Bouton" name="bouton" onclick="clic()" />
```

- Évènements liés :

- `onClick` → Détecte le clic sur le bouton

Accès aux formulaires

- Le DOM possède un objet forms listant tous les formulaires

```
document.forms[0] ou document.forms.item(0)
```

- Les éléments de chaque formulaire sont listés dans leur propriété elements

```
document.forms[0].elements[0] ou  
document.forms.item(0).elements.item(0)
```

- On peut accéder directement aux éléments de formulaire en utilisant les noms mis avec leur attribut `name`

```
document.monform.ch1  
document.forms["monform"].elements["ch1"]
```

Évènements de formulaires

- On peut écouter des évènements en plaçant des écouteurs sur les balises des formulaires ou de leurs enfants

- Exemple :

```
<form name="nom" onSubmit="traitement(this)"></form>
```

- Le mot **this** est optionnel et peut être utilisé pour passer à la fonction une référence à la balise sur lequel l'écouteur est placé

- Si on place l'écouteur sur une balise enfant d'un formulaire, on peut également utiliser this, mais il est parfois utile d'utiliser this.form pour faire référence au formulaire parent de la balise

```
<input name="ch1" onChange="traitement(this.form)" />
```

Propriétés et méthodes des formulaires

- Pour presque tous les éléments de formulaires, on peut récupérer les propriétés suivantes :
 - **name** → nom du champ
 - **type** → type du champ
 - **value** → valeur du champ (seulement pour les enfants des forms)
 - [...]
- Certains champs ont cependant des propriétés supplémentaires qui seront abordées après

Manipulation des textfields

```
<INPUT type="text" value="V" name="N" />
```

- Avec V=valeur par défaut, N=nom du champ
- On peut remplacer text par `password` ou `hidden` pour avoir un champ avec caractères masqués ou un champ invisible
- Évènements applicables :
 - `onFocus` → Gain de focus
 - `onBlur` → Perte du focus
 - `onChange` → Perte du focus après un changement de valeur

Pratique : exercice 6.1 - textfields

Manipulation de checkboxes

```
<INPUT type="checkbox" name="N" checked />
```

- Avec N=nom du champ. Checked est optionnel et permet de cocher la case par défaut
- Propriétés supplémentaires :
 - `checked` → true ou false suivant que la case est cochée ou non
- Évènements applicables :
 - `onFocus` → Gain de focus
 - `onBlur` → Perte du focus
 - `onClick` → Détecte le clic

Manipulation de checkboxes

- Il est pratique d'utiliser le même nom pour toutes les checkboxes. On distingue alors les diverses checkboxes en utilisant l'attribut `value`. Exemple :

```
<INPUT type="radio" name="voiture" value="break" />
```

```
<INPUT type="radio" name="voiture" value="collection" />
```

- On peut alors accéder au nombre d'éléments du tableau :

```
document.monform.elements["voiture"].length
```

- et à un élément particulier :

```
document.monform.elements["voiture"][0]
```

Pratique : exercice 6.2 - checkboxes

Manipulation des boutons radio

```
<INPUT type="radio" name="N" value="V" checked />
```

- Avec V=valeur, N=nom du champ. Checked est optionnel et permet de cocher le bouton radio par défaut

- Propriétés supplémentaires :

- `checked` → true ou false suivant que la case est cochée ou non

- Évènements applicables :

- `onFocus` → Gain de focus
- `onBlur` → Perte du focus
- `onClick` → Détecte le clic

Manipulation des boutons radio

- On récupère un tableau de valeurs correspondant au nom que l'on a donné aux boutons radio via l'attribut name
- On a donc accès au nombre de boutons radios ayant ce name :
`document.monform.nom_radio.length`
- Et on a la possibilité d'accéder à un élément radio particulier :
`document.monform.nom_radio[0]`

Pratique : exercice 6.3 - radios

Manipulation de listes déroulantes

```
<select name="N">  
  <option value="V1" selected>I1</option>  
  <option value="V2">I2</option>  
</select>
```

- Avec N=nom, V1=valeur 1, I1=intitulé 1, etc. On peut aussi lui appliquer les propriétés `size`, et `multiple`
- On peut placer un `selected` de façon à sélectionner une option par défaut
- Évènements applicables :
 - `onFocus` → Gain de focus
 - `onBlur` → Perte du focus
 - `onChange` → Détecte le clic

Manipulation de listes déroulantes

- Les listes ont 2 propriétés intéressantes :
 - `.selectedIndex` → numero de l'option sélectionnée
 - `.options` → liste des options (HTML collection)
- On peut donc récupérer l'option d'une liste, et le nombre d'options :
`liste.options[0] //première option`
`liste.options.length //nombre d'options`
- Propriétés des éléments option :
 - `.selected` → true ou false suivant que l'option est sélectionnée ou non
 - `.text` → texte de l'option
 - `.value` → valeur de l'option
 - `.index` → numéro de l'option *Pratique : exercice 6.4 - comboboxes*

Manipulation de listes déroulantes

- On dispose de 2 fonctions pour ajouter ou supprimer des options :
 - `ma_liste.add(new Option("intitulé", "valeur"))`
 - `ma_liste.remove(num_option)`

Pratique : exercice 6.5 – comboboxes dépendantes

Manipulation de textareas

`<textarea cols="C" rows="R" name="N">V</textarea>`

- Avec C=colonnes, R=lignes, N=nom, V=valeur par défaut
- Il existe aussi la propriété `readonly` pour que le textarea soit en lecture seule

• Événements applicables :

- `onFocus` → Gain de focus
- `onBlur` → Perte du focus
- `onChange` → Détecte le clic
- `onScroll` → Déroulement de la liste

Validation de formulaire

- On utilise l'événement `onSubmit()` dans les balises du formulaire, et on y appelle une fonction précédée du mot `return`

`<form action="#" method="post" onSubmit="return verifier(this)">`

- Cette fonction teste les champs du formulaire (form.champ1...)
- Si la fonction renvoie `true` les données sont envoyées
- Si la fonction renvoie `false` les données ne sont pas envoyées

Pratique : exercice 6.6 – validation de formulaire

Validation de formulaire

- On peut faire des fonctions plus ou moins évoluées appelées depuis la fonction présente dans le onsubmit

- Exemple pour la validation d'email :

```
function validateEmail(champ) {  
    var x=champ.value;  
    var atpos=x.indexOf("@");  
    var dotpos=x.lastIndexOf(".");  
    if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length){  
        alert("Not a valid e-mail address");  
        return false;  
    }  
}
```



Chapitre 7 AJAX

Dans cette partie

- Présentation de Asynchronous Javascript And Xml
- Enjeux, solutions et alternatives.
- Les bibliothèques du marché.
- HTTP et Ajax : échanges HTTP et l'objet XMLHttpRequest.
- **Travaux pratiques**
 - *Récupération de données XML et affichage des données. Interrogation d'une base de données. Création d'un formulaire de connexion.*

Présentation d'AJAX

- AJAX : Asynchronous JavaScript And XML
- Permet l'échange d'informations avec un serveur distant sans rechargement de page
- Asynchrone : le script n'attend pas la réponse pour continuer

Enjeux, problèmes, solutions

- Avantages :
 - Gain de temps (pas de préchargement)
 - Meilleure ergonomie
 - Possibilités d'applications étendues
- Problèmes :
 - Compatibilité entre navigateurs
- Solutions :
 - Prévoir les différentes façons de coder en fonction du navigateur
 - Utiliser des bibliothèques

Principe d'ajax

- 1) on crée un objet capable d'appeler une page distante (en lui communiquant ou non des données) et de recevoir une réponse (l'objet est de type XMLHttpRequest)
- 2) on définit clairement la fonction qui récupérera l'objet et donc les données distantes. Elle peut afficher ces données, etc.
- 3) on lie cet objet à la fonction, de telle sorte qu'elle puisse récupérer l'objet lorsqu'il recevra une réponse, lors de sa prochaine communication (cette fonction est un callback)
- 4) on lance une communication vers un script distant. Il y a la méthode GET et la méthode POST. On peut transmettre au script distant des données si besoin
- 5) on gère ensuite le côté serveur (PHP, ASP, XML, etc.) qui perçoit des paramètres optionnels et retourne une réponse. Comme convenu la fonction définie à l'étape 2 et liée à l'objet à l'étape 3 reçoit les données et les traite

1) on crée XMLHttpRequest

- Il y a plusieurs manières pour l'instancier XMLHttpRequest

```
if (window.XMLHttpRequest){      // IE7+, et autres
    xhr=new XMLHttpRequest();
}else{                            //IE6, IE5
    try {
        xhr = new ActiveXObject("Msxml2.XMLHTTP");
    } catch(e) {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

Fonction pour l'étape 1

- Il est pratique de créer une fonction pour instancier l'objet :

```
function getXMLHttpRequest() {  
    var xhr = null;  
    if (window.XMLHttpRequest || window.ActiveXObject) {  
        code précédent  
    } else {  
        alert("Pas de support de l'objet XMLHttpRequest");  
        return null;  
    }  
    return xhr;  
}
```

XMLHttpRequest

onreadystatechange	Evènement déclenché dès que la propriété readyState de l'objet change. C'est ici qu'on appelle la fonction qui récupère les données
readyState	Etat de la requête
responseText	Retour texte du serveur
responseXML	Retour XML du serveur
status	Code de retour serveur (200, 404, 500...)
statusText	Message décrivant le status

2) on prépare la fonction qui reçoit

- Définissons par exemple la fonction `afficherDonnees` qui est prête à recevoir un objet `xhr` de type `XMLHttpRequest` et à afficher les données qu'il contient
- Ces données sont dans la propriété `responseText` de `xhr` pour un appel de script `php`, `asp`, `txt`, etc.
- Ces données sont dans propriété `responseXML` de `xhr` pour un appel de fichier `XML`

```
function afficherDonnees(xhr) {  
    alert("données recues : "+xhr.responseText);  
}
```

3) on lie l'objet à la fonction

- On crée un objet `XMLHttpRequest` avec la fonction précédente
- On écoute l'évènement `onreadystatechange`
- Si la communication s'est bien déroulée, on appelle la fonction de l'étape 2 en lui passant l'objet `xhr` (et donc la réponse du script distant)

```
var xhr = getXMLHttpRequest();  
xhr.onreadystatechange = function() {  
    if (this.readyState == 4 && (this.status == 200 || this.status == 0)) {  
        afficherDonnees(this);  
    }  
};
```

Fonction pour l'étape 3

- Là encore il est pratique de faire une fonction

```
function linkXhrToCallback(xhr,callback){  
  xhr.onreadystatechange = function() {  
    if (this.readyState == 4 && (this.status == 200 || this.status == 0)) {  
      callback(this);  
    }  
  };  
}
```

- Le script devient alors tout simple

```
var xhr = getXMLHttpRequest(); //étape 1 : on crée l'objet  
linkXhrToCallback(xhrafficherDonnees); //étape 2 : liaison xhr / fonction
```

4) envoi des données en GET

- On peut envoyer les données en GET. Comme les variables sont transmises par l'url on les encode avec `encodeURIComponent()`
- Pour contourner les problèmes de cache avec IE, il faut passer une variable unique, par exemple une date

```
var unique = new Date().valueOf();  
xhr.open( "GET",  
  "ajax.php?rand="+unique+"&param="  
    +encodeURIComponent(ma_variable),true);  
xhr.send(null);
```

- `open()` : `open(methode,url du script, asynchrone ou non)`

4) envoi des données en POST

- On peut envoyer les données en POST.
- Pour que la communication POST se déroule correctement, ne pas oublier de rajouter un appel de `setRequestHeader()`

```
xhr.open("POST", "ajax.php", true);  
xhr.setRequestHeader("Content-Type",  
                     "application/x-www-form-urlencoded");  
xhr.send("param="+ma_variable);
```

5) traitement coté serveur

- Le script du serveur précise l'entête et affiche les données en PHP

```
<?php  
header("Content-Type: text/plain"); //si on renvoie des données texte  
header("Access-Control-Allow-Origin: *"); //sécurité de Firefox  
//GET  
$param = (isset($_GET["param"])) ? $_GET["param"] : NULL;  
  
//POST (ou get au choix, en activer un)  
//$param = (isset($_POST["param"])) ? $_POST["param"] : NULL;  
  
if ($param) {    echo "OK : ".$param; }  
else {    echo "FAIL";    }  
?>
```

Pratique : exercice 7.1 – fonction ajax et 7.2 – login

Chargement de XML

- On crée d'abord un fichier XML. Exemple, livres.xml :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<livres>
  <livre titre="Le Rouge et le Noir">
    <auteur>Stendhal</auteur>
  </livre>
  <livre titre="L'alchimiste">
    <auteur>Paulo Coelho</auteur>
  </livre>
</livres>
```

Chargement de XML

- On appelle le fichier XML avec une URL unique :

```
var unique = new Date().valueOf();
xhr.open("GET","livres.xml?rand="+unique,true);
xhr.send(null);
```

Chargement de XML

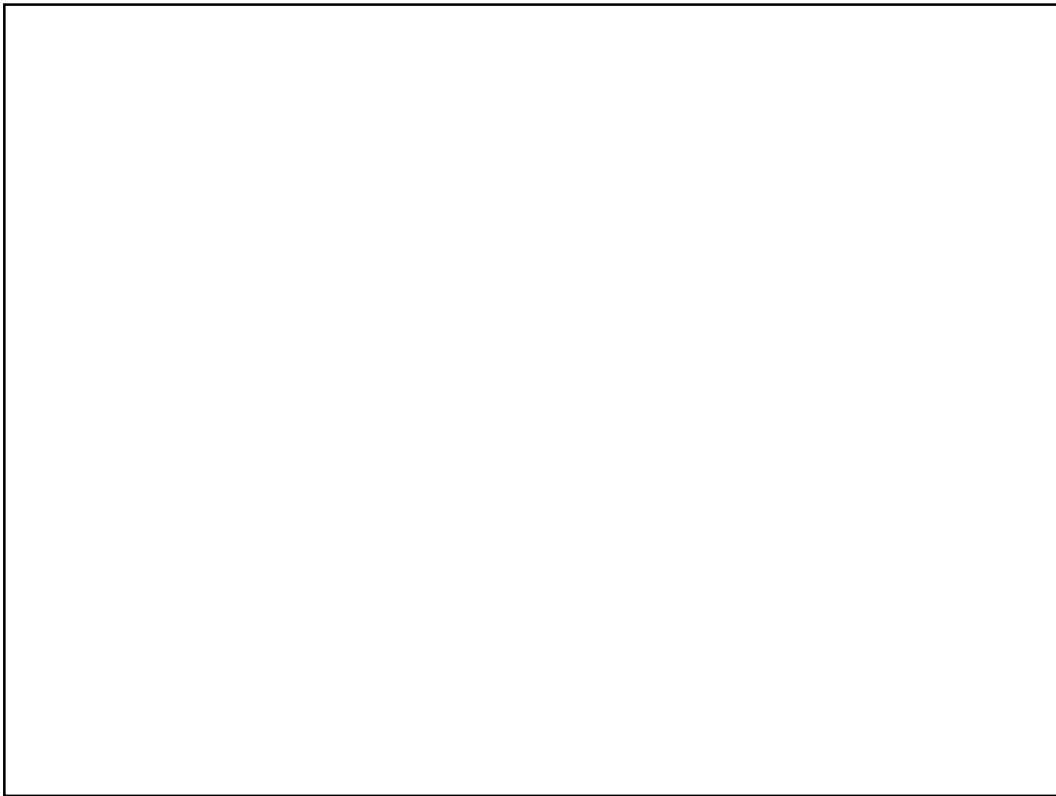
- La fonction de réception utilise `xhr.responseXML` au lieu de `xhr.responseText` et récupère les données ainsi :

```
var resultat="";
livre=xhr.responseXML.getElementsByTagName("livre");
for (var i=0;i<livre.length;i++){
    titre=livre[i].getAttribute("titre");
    auteur=livre[i].getElementsByTagName("auteur").item(0).firstChild.
nodeValue;
    resultat+=titre+"<BR/>" +auteur+"<BR/>";
}
document.write(resultat);
document.close();
```


Pratique : exercice 7.3 – pages dynamiques XML

Fonctions téléchargeables

- Pour palier au problème de compatibilité multi-navigateurs, les fonctions se sont compliquées
- Il est donc intéressant d'utiliser des fonctions testées, et disponibles gratuitement sur internet
- Exemples :
- <https://github.com/ilinsky/xmlhttprequest>
- <http://www.quirksmode.org/js/xmlhttp.html>



Formation - Création web



Alexandre Beaugrand

Chapitre 8

Besoins divers

Dans cette partie

- Timers
- Manipulation des URL (redirections http, ...)
- Gestion des cookies (lecture et écriture)
- Travaux pratiques
 - *Exercices de mise en application des points abordés*

Timers

- On utilise la fonction `setTimeout(code,delaiEnMillisecondes)`
- Exemple :

Dans head :

```
function decompte(){  
    var t=setTimeout("alert('2 secondes après !')",2000);  
}
```

Dans body :

```
<a href='#' onClick="decompte();return false">  
    Compte 2 secondes  
</a>
```

Manipulation des URL

- Si l'URL est `http://www.monsite.com/index.html?param1=toto`
- `location.href` → `http://www.monsite.com/index.html?param1=toto`
- `location.host` → `www.monsite.com`
- `location.hostname` → `www.monsite.com`
- `location.pathname` → `/index.html?param1=toto`
- `location.search` → `?param1=toto`

Redirections

- Il suffit de forcer les URL en donnant une valeur à `window.location`

`window.location = "http://www.google.com/"`

Pratique : exercice 8.1 – redirection avec délai

Récupération des paramètres URL

```
function getUrlParameter(name) {  
    var searchString = location.search.substring(1).split('&');  
    for (var i = 0; i < searchString.length; i++) {  
        var parameter = searchString[i].split('=');  
        if(name == parameter[0]) return parameter[1];  
    }  
    return false;  
}
```

//utilisation : getUrlParameter("param")

Cookies

- Ce sont des petits fichiers textes utiles pour sauvegarder et récupérer des informations d'une visite de site à l'autre
- Cela peut être utile, par exemple, pour sauvegarder les préférences des utilisateurs, ou le nombre de visites, ou la date de la dernière visite, etc.
- La sauvegarde est de la forme :

```
document.cookie = "nom=valeur; expires=date; path=/"
```

Écriture de cookies

- Le plus simple consiste à utiliser cette fonction

```
function setCookie(nom,valeur,jours) {  
  if (jours) {  
    var date = new Date();  
    date.setTime(date.getTime()+(jours*24*60*60*1000));  
    var expires = "; expires="+date.toGMTString();  
  }  
  else var expires = "";  
  document.cookie = nom+"="+valeur+expires+"; path=/";  
}
```

Lecture de cookies

- On peut également utiliser une fonction :

```
function getCookie(nom) {  
  var nameEgal = nom + "=";  
  var ca = document.cookie.split(';');  
  for(var i=0;i < ca.length;i++) {  
    var c = ca[i];  
    while (c.charAt(0)==' ') c = c.substring(1,c.length);  
    if (c.indexOf(nameEgal) == 0) return  
    c.substring(nameEgal.length,c.length);  
  }  
  return null;  
}
```

Suppression de cookies

- Pour supprimer un cookie, il suffit de le redéfinir à une date passé (expire dans un nombre de jours négatif)

```
function deleteCookie(nom){  
    setCookie(nom,"",-1);  
}
```

Pratique : exercice 8.2 – cookies



Annexe Introduction à JQuery

Dans cette partie

- Frameworks Javascript
- Installation de JQuery
- Sélections de noeuds
- Modifier un noeud
- Créer, ajouter, supprimer un noeud
- Évènements
- Ajax
- JQuery UI

Frameworks Javascript

- Il existe des bibliothèques pour développer plus vite en Javascript
- Avantages :
 - Gain de temps
 - Développement plus intuitif
 - Support multi-navigateurs
 - Nombreux plugins
 - Communauté
- 3 bibliothèques connues :
 - JQuery : <http://jquery.com/>
 - Mootools : <http://mootools.net/>
 - Prototype : <http://www.prototypejs.org/>

Installation de JQuery

- Téléchargement sur <http://jquery.com/>
- Placer le fichier dans un répertoire, créer une page web, et l'appeler depuis la page :

```
<script src="jquery.js" type="text/javascript" ></script>
<script type="text/javascript">
$(function(){
    alert('helloworld !');
})
</script>
```

Sélections de noeuds : sélecteurs

- JQuery permet de sélectionner et d'agir sur un ou des nœuds directement. Exemple :

```
$("div").css("border","1px solid blue");
```

- On peut remplacer **div** par les sélecteurs suivants :
- **#intro** ou **p#intro** : les éléments ou les paragraphes d'id intro
- **.red** ou **p.red** : les éléments ou les paragraphes de classe red
- **p strong** : les éléments strong inclus dans des paragraphes
- **a,strong** : les éléments a et les éléments strong
- etc.

Sélections de noeuds : filtres

- On peut filtrer les listes de nœuds sélectionnées. Exemple :

```
$("div:nth(1)").css("border","1px solid blue");
```

- On peut remplacer **:nth(1)** par les filtres suivants :
- **:nth-child (1)** : les div 1ers fils de leur nœud parent
- **:not (.red)** : les div qui n'ont pas la classe red
- **:empty** : les div qui n'ont pas d'enfants
- **:gt(1)** : les div dont la position est supérieure à 1
- **:lt(3)** : les div dont la position est inférieure à 3
- **:contains(3e)** : les div qui contiennent le texte 3^e
- **[title=intro]** : les div qui ont un attribut title = intro

Agir sur un noeud

- **Styles :**
 - Lecture : `nœud.css("border")`
 - Ecriture : `nœud.css ("border","1px solid blue")`
- **Attributs :**
 - Lecture : `nœud.attr('title')`
 - Ecriture : `nœud.attr("title","second")`
 - Suppression : `nœud.removeAttr("title")`
- **Classes :**
 - Ajout : `nœud.addClass('red')`
 - Suppression : `nœud.removeClass('red')`
 - Ajout / Suppression : `nœud.toggleClass('red')`
- **Html :**
 - Lecture : `nœud.html()`
 - Ecriture : `nœud.html(...)`
- **Valeur :**
 - Lecture : `nœud.val()`
 - Ecriture : `nœud.val(...)`

Créer, ajouter, supprimer un noeud

- **Création :**
 - `$("<div>nouveau div</div>")`
- **Ajout :**
 - Intérieur, au début : `nœud.prependTo(nœud_destination);`
 - Intérieur, à la fin : `nœud.appendTo(nœud_destination);`
 - Avant : `nœud.insertBefore(nœud_destination);`
 - Après : `nœud.insertAfter(nœud_destination);`
- **Suppression :**
 - vider : `nœud.empty()`
 - supprimer le nœud : `nœud.remove()`

Évènements

- On ajoute un évènement ainsi :

```
$('#bouton').click(function(){  
    $(this).toggleClass("red");  
})
```

- On peut remplacer **click** par :
 - **focus** / **blur** : gain / perte de focus
 - **change** : changement
 - **dblclick** : double clic
 - **keydown** / **keypress** / **keyup**
 - etc.

Ajax

- Envoi de données en POST :

```
$.post( "ajax.php",  
    {mavARIABLE:'mavaleur'},  
    function done(data){  
        alert(data);  
    });
```

- Script serveur :

```
<?php echo 'Et voila : '.$_POST['mavARIABLE'].' !'; ?>
```


Jquery UI

- UI = User Interface. Jquery dispose d'une série de composants que l'on peut personnaliser.
- Permet d'utiliser des effets (redimensionner, glisser-déposer, etc.)
- Permet d'utiliser des widgets (accordéon, menu, calendrier, etc.)
- Permet d'utiliser des effets (animations de couleurs, etc.)
- Téléchargeable ici : <http://jqueryui.com/>
- Nécessite jquery

Webliographie

- <http://www.w3schools.com/js/>
- <http://www.w3schools.com/jsref>
- <http://fr.selfhtml.org/javascript/>
- <http://jacques-guizol.developpez.com/javascript/Objets/Objets.php>
- <http://www.commentcamarche.net/contents/javascript/>
- <http://jpvincent.developpez.com/tutoriels/javascript/javascript-oriente-objet-syntaxe-base-classes-js-intention-developpeurs-php/>
- <http://www.javascriptkit.com/javatutors/proto4.shtml>
- http://www.w3schools.com/html/dom/dom_nodetree.asp
- http://www.nanoum.net/blog/9_setAttribute.html
- http://www.toutjavascript.com/savoir/savoir06_1.php3
- <http://ppk.developpez.com/tutoriels/javascript/gestion-cookies-javascript/>
- <http://www.quirksmode.org/dom/intro.html>

Sites utiles

- JQuery
- Mootools
- Références DOM:
 - http://www.w3schools.com/html/dom/dom_methods.asp
 - http://www.w3schools.com/jsref/dom_obj_style.asp
 - <http://www.quirksmode.org/dom/intro.html>
 - http://www.quirksmode.org/dom/w3c_css.html
- Références Evènements :
 - http://www.w3schools.com/jsref/dom_obj_event.asp
- Gestion des formulaires :
 - http://www.toutjavascript.com/savoir/savoir06_3.php3
- Ajax :
 - <http://www.siteduzero.com/tutoriel-3-100294-l-objet-xmlhttprequest.html>
 - http://www.w3schools.com/ajax/ajax_xmlfile.asp