Philippe.Renevier@unice.fr

Organisation du cours

- Jeudi 10 mars
  - Contrôle et présentation du projet
- Semaine du 21 au 25 février : fin du premier sujet, début du second
- Semaine du 7 – 11 mars : second sujet
- Semaine 14-18 mars
  - Tutoriaux php – image
  - Tutoriaux outil javascript

Philippe.Renevier@unice.fr

Javascript et Php : interaction plus fluide

Javascript : Principes de bases

AJAX et XML

AJAX et l'Architecture

JavaScript

- langage de scripts
  - incorporé aux balises Html,
  - améliorer la présentation et l'interactivité
- JavaScript n'est pas JAVA
- JavaScript partage les mêmes objets DOM (Document Object Model) que ceux de document HTML.

JavaScript et Info Générale

- http://deptinfo.unice.fr/~renevier/infogene/tutojs/
- En résumé :
  - document.write : (bof) -> plutôt pour php
  - innerHTML : permet de modifier le contenu
  - Évenements
    - Attribut html (onclick, onmouseover, etc.)
    - http://www.w3.org/TR/html4/interact/scripts.html
  - Objet html
    - this (lors de l'événement)
    - document.getDocumentByID

JavaScript : syntaxe

- Similaire à java et php…
- Types : JavaScript comprend les types de données suivants :
  - les nombres : 2, 2.90, 314E-2 . Ils peuvent être en base 10, 16 (0x.. ou 0X..) ou 8 (commençant par 0). Les réels flottants peuvent contenir un point décimal, un exposant (E)
  - les booléen : true ou false
  - les chaînes de caractères : "coucou" ou 'coucou'. Les caractères spéciaux peuvent être utilisés dans les chaînes (\b backspace, \f form feed , \n new line , \r CR, \t tabulation). Le caractère \ permet d'insérer une double quote dans une chaîne
  - les tableaux : tableau[0], tableau[1]
  - les tableaux associatifs : X["Poitier"] , Y["Poitier"]

Javascript : manipulation des variables

- déclaration
  - la déclaration peut être implicite (utilisation sans autre instruction) ou explicite, avec le mot clef "var".
    Les noms doivent commencer par une lettre et ne doivent pas contenir les caractères spéciaux : espace, +, *, /, &, %, $, #, @, !, |, etc.
    Les noms des variables sont sensibles à la casse.
- mots réservés
  - abstract, boolean, break, byte, case, catch, char, class, const, continue, default, delete, do, double, else, export, extends, false, final, finally, float, for, function, goto, if, implements, in, instanceof, int, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, while, with
- affectation
  - Simplement, juste avec "=". Attention, une affectation peut modifier le type d'une variable.
- déterminer le type d'une variable
  - cela est possible avec l'opérateur "typeof". Il retourne undefined, number (pour entier ou réel), string, un type d'objet ou fonction
- conversion
  - d'une chaîne à un nombre : "parseInt" ou "parseFloat". Ces fonctions peuvent tronquer la valeur. Elles peuvent aussi retourner "NaN" (not a number). La fonction isNaN(x) retourne vrai si x n'est pas un nombre.
  - vers une chaîne : "toString" (une méthode) ou ""+variable

Javascript objet

- propriétés et méthodes
  - simplement accessibles par un ".". Par exemple, si str est une chaîne, str.length permet d'accéder à la propriété "length" de la chaîne str. La propriété peut être un objet, duquel il est possible d'accéder aux propriétés et méthodes.
    Pour une méthode, c'est semblable, sauf qu'il y a les "(" ")" et les paramètres.
    La création se fait avec le mot clef "new".
- existance d'un objet
  - Il est possible de tester si un objet existe, avec un "if". Par exemple if (window.XMLHttpRequest) permet de savoir si l'objet XMLHttpRequest existe (ce qui est le cas avec un "mozilla", mais ne l'est pas avec un "ie").
- liste de propriétés
  - Une version de la boucle for permet de parcourir la liste des propriétés d'un objet. Le mot clef "in" et le nom de l'objet permettent le parcours facilement.
    Par exemple, for(prop in window) {...window[prop]..} permet de parcourir toutes les propriétés de l'objet document.
- this : objet en cours
  - this n'est exploitable que dans une fonction en cours, ou pour faire référence à un objet web avec lequel l'utilisateur agit (voir les événements).

JavaScript : String

- Outre l'attribut length, il existe beaucoup de méthodes.
- charAt(i) : permet d'obtenir le caractère à l'indice i
- contact("...") : pour joindre deux chaînes, équivalent de l'opérateur +
- replace(str1, str2) : remplace la première occurrence de la chaîne str1 par la chaîne str2. Pour les remplacer toutes, il faut faire une boucle.
- substring(i, j) ; substr(i, l) : permettent d'extraire une partie de la chaîne, entre deux indices pour substring, depuis un indice pour un certain nombre de caractères pour substr. Note : slice fonctionne aussi
- toLowerCase() et toUpperCase() : permettent de mettre en minuscule ou (respectivement) en majuscule
- split(sep) : découpe la chaîne en tableau, les éléments étant définis par le séparateur "sep", qui n'apparait plus
- search(str) ; indexOf(str, i) ; lastIndexOf(str) : permettent de rechercher l'indice de l'apparition d'une chaîne dans la "string", ou retournent -1 si elle n'y est pas. search et indexOf retourne la première occurence, sauf qu'avec indexOf il est possible de préciser l'indice de début de recherche. lastIndexOf retourne la dernière occurence

Philippe.Renevier@unice.fr

Javascript : Math

- L'objet Math définit des constantes (attributs PI, E, LN10, LN2, SQRT2, etc.) et les fonctions mathématiques usuelles :
- random() : nombre aléatoire entre 0 et 1
- abs() : valeur absolue (du paramètre)
- trigonométrique (cos, sin, tan, acos, asin, atan) : le tout en radian
- log ; exp : logarithme et exponentielle
- sqrt : racine carrée
- pos(x,y) = xy (puissance de x à l'exposant y)
- round ; floor ; ceil : les arrondis, respectivement standar, entier le plus proche le plus petit et entier le plus proche le plus grand
- min(x, y) ; max(x, y) : déterminent le plus petit (ou le plus grand) des nombres x et y
- Utilisation : Math.sqrt(2) , Math.min(10, y), etc.

Javascript : Date

- Créer sans paramètre, il s'agit de la date du jour (et de l'heure) de création. Avec des paramètres, il est possible de préciser n'importe quelle date. Il y a trois formats :
  - chaîne : new Date("Nov 28, 2005 15:45:00")
  - valeurs : new Date(2005, 10, 28) pour le 28 novembre 2005
  - valeurs étendues : new Date(2005, 10, 28, 15, 45, 00) pour la même date que dans 1)
- Les dates sont définies en milisecondes à partir du 1er janvier 1970. Il faudra parfois user de "/" et "%" pour faire des calculs entre dates. Les méthodes suivantes sont disponibles :
  - pour obtenir un champ : getYear() (pour l'année, les « deux » derniers chiffres, e.g. 05), getFullYear(), getMonth() (pour le mois, e.g. 0->janvier, 11->décembre) getDate() (pour le jour du mois, e.g. 28), getDay() (pour le jour de la semaine, e.g. 1-> lundi, 7 -> dimanche), getHours() getMinutes() et getSeconds() (pour l'heure, les minutes et les secondes)
  - pour modifier un champ : setYear(05) ou setYear(2005), setMonth(10), setDate(28), setHours(15), setMinutes(45), setSeconds(0)
  - pour obtenir la date en milisecondes depuis le 1er janvier 1970 : getTime(). Pour mettre la date via un temps en ms, setTime(val)
  - conversion d'un nombre en milisecondes en un format "lisible" : toGMTString() qui retourne une chaîne du type "Mon, 28 Nov 2005 14:45:00 UTC". toLocaleString() donnera l'heure locale, sous le format "11/28/2005 15:45:00". toString() sur un objet Date donnera encore un autre résultat : "Mon Nov 28 15:45.00 MDT 2005". A l'inverse, à partir d'un des formats de création (c.f. ci-dessus), il est possible d'obtenir le temps en milisecondes avec la méthode Date.UTC(...)

JavaScript : opérateur

- Opérateurs : Comme en C ou en Java les opérateurs suivants sont valides : + (adition), - (soustraction) , * (multiplication) , ++ (incrément) , -- (décrément) , / (division), % (modulo), & (ET bit à bit), | (OU bit à bit), ^ (XOR bit à bit), << (décalage de bits vers la gauche), >> (décalage de bits vers la droite),>>> (idem mais en mettant les nouveaux digits à 0), && (ET logique entre deux booléens), || (OU logique entre deux booléens), ! (négation d'un booléen)
- Les comparaisons sont faites par les opérateurs : == (test dégalité) , > , >=, <, <=, != (test d'inégalité)
- Les opérateurs peuvent être assignés à l'affectation, par exemple :
  - x += 4 équivaut à x = x + 4
  - x++ signifie x+=1 c'est à dire x = x +1
  - x-- signifie x-=1 c'est à dire x=x-1

Javascript : les instructions de contrôle

- for : permet de répéter une séquence d'instructions tant qu'une condition est vraie; elle peut prendre deux formes :
  - for (i = 0; i < 255; i++) { fonction(i) }
  - for (var in obj)in { instructions }
- Les instructions break et continue :
- Tests
  - if (prix == 0) { x++ } else { result = result + prix }
- while / do { } while …

JavaScript : les fonctions

- Les fonctions : Les fonctions en JavaScript comme dans tous les autres langages désignent des suites d'instructions répétitives, qui sont exécutés plusieurs fois mais écrites une seule fois.
  - Syntaxte : function nom (param1, param2,...) { ...}
  - Remarques :
    - Nombre de paramètres quelconque (propriété arguments de la fonction, un tableau)
    - Une fonction peut appeler des arguments de tout type, ainsi que des fonctions et les fonctions peuvent être appelées récursivement.
    - Une fonction retourne une valeur quand elle rencontre le mot clé return suivie de la valeur retournée qui peut être de tout type. Les fonctions suivantes permettent de manipuler les chaînes de caractères :
      - eval convertit une chaîne de caractères en valeur entière
      - parseInt convertit une chaîne de caractères en valeur entière dans la base spécifiée
      - parseFloat convertit une chaîne de caractères en valeur réelle flottante
  - Mot-clé **var** pour déclarer des variables locales

JavaScript et HTML (1/2)

- Insérer du code JavaScript
  - par la balise SCRIPT
    <script type= "text/javascript" >
    // code javascript
    </script>
  - Du code dans un fichier source
    <script type= "text/javascript" src= "source.js " >
    </script>
  - Par morceaux de code dans les événements (c.f. plus loin)

JavaScript et HTML (2/2)

- Exemple d'insertion de la date avec document.write

```
<script type="text/javascript">
var jour = new Date();
document.write("<p> Nice, le ", jour.getDate(), "/",
jour.getMonth() + 1, "/", jour.getFullYear(), "</p>");
</script >
```

- Manipulation de l'attribut innerHTML des objets web
  - Chaque objet HTML a une existance javascript
  - Ces objets sont manipulable c.f. DOM
  - Par manipulation du code HTML entier !

JavaScript et HTML : Par les évènement

- Une action utilisateur (déplacement de la souris sur une zone, clic sur un bouton souris, ...) crée un événement que javascript peut alors traiter par un handler associé
- le document HTML deviendra réactif à un événement si à cet événement est associée une fonction javascript : Ces événements sont surveillés dans certaines balises HTML en particulier dans un formulaire ou bien sur un lien hypertexte.
- Syntaxe :
- <Balise-HTML Attributs eventhandler= “Code JAVASCRIPT”>
- Exemple : déclencher une fonction JavaScript, appelée control() pour contrôler la donnée d’un champ INPUT d’un formulaire :

```
<form>
<input name="telephone" onChange= "control(this); ">
</form>
```

JavaScript et HTML : Les événements

- onload = script The onload event occurs when the user agent finishes loading a window or all frames within a FRAMESET. This attribute may be used with BODY and FRAMESET elements.
- onunload = script The onunload event occurs when the user agent removes a document from a window or frame. This attribute may be used with BODY and FRAMESET elements.
- onclick = script The onclick event occurs when the pointing device button is clicked over an element. This attribute may be used with most elements.
- ondblclick = script The ondblclick event occurs when the pointing device button is double clicked over an element. This attribute may be used with most elements.
- onmousedown = script The onmousedown event occurs when the pointing device button is pressed over an element. This attribute may be used with most elements.
- onmouseup = script The onmouseup event occurs when the pointing device button is released over an element. This attribute may be used with most elements.
- onmouseover = script The onmouseover event occurs when the pointing device is moved onto an element. This attribute may be used with most elements.
- onmousemove = script The onmousemove event occurs when the pointing device is moved while it is over an element. This attribute may be used with most elements.
- onmouseout = script The onmouseout event occurs when the pointing device is moved away from an element. This attribute may be used with most elements.

JavaScript et HTML : Les événements

- onfocus = script The onfocus event occurs when an element receives focus either by the pointing device or by tabbing navigation. This attribute may be used with the following elements: A, AREA, LABEL, INPUT, SELECT, TEXTAREA, and BUTTON.
- onblur = script The onblur event occurs when an element loses focus either by the pointing device or by tabbing navigation. It may be used with the same elements as onfocus.
- onkeypress = script The onkeypress event occurs when a key is pressed and released over an element. This attribute may be used with most elements.
- onkeydown = script The onkeydown event occurs when a key is pressed down over an element. This attribute may be used with most elements.
- onkeyup = script The onkeyup event occurs when a key is released over an element. This attribute may be used with most elements.
- onsubmit = script The onsubmit event occurs when a form is submitted. It only applies to the FORM element.
- onreset = script The onreset event occurs when a form is reset. It only applies to the FORM element.
- onselect = script The onselect event occurs when a user selects some text in a text field. This attribute may be used with the INPUT and TEXTAREA elements.
- onchange = script The onchange event occurs when a control loses the input focus and its value has been modified since gaining focus. This attribute applies to the following elements: INPUT, SELECT, and TEXTAREA.

Objet Web

- Image
  - Cet objet permet notamment le préchargement d'image, par les instructions
  - img = new Image(); img.src = "url de l'image";
  - La propriété complete est un booléen qui est vrai lorsque l'image est chargée.

```
var img = new Array();
function load() {
    var nb_img = load.arguments.length;
    for(var i =0; i < nb_img; i++) {
            img[i] = new Image();
            img[i].src = load.arguments[i];
            }
}
```

Objet Web

- history
  - Cet objet permet de manipuler l'historique de navigation.
  - Sa propriété length donne le nombre de pages visitées.
  - Les méthodes sont les suivantes :
    - back() (page précédente),
    - forward() (page suivante)
    - et go(n) (passer à une URI dans l'historique, placée en nième position, n peut être négatif ou égal à zéro).

Objet Web

- location
  - L'objet location permet d'avoir des informations sur la "localisation" du document (propriétés .href, .hostname, .pathname, etc.).
  - Avec la méthode replace il est possible d'effacer le lien actuel de l'historique. Par exemple, c'est le cas de ce lien vers deptinfo. Son code est le suivant :
  - ce lien vers deptinfo
  - La fonction "replace" retourne la chaîne passée en paramètre.
  - location offre aussi la fonction "reload()" équivalente à hitory.go(0);

Objet web

- window
  - C'est l'objet de plus haut niveau, celui qui contient tout les autres. Par exemple, ci-dessus, "window." est omis devant "location".
  - Il possède diverses propriétés, dont status qui donne accès à la barre de status de la fenêtre. (si les préférences du navigateur le permet).
  - Parmis les fonctions, il y a : moveTo(x,y) / moveBy(dx, dy) et resizeTo(w,h) / resizeBy(dw, dh).
  - Il y a aussi les fonctions pour les boîtes de dialogues :
    - alert(msg),
    - prompt(msg, valpardef) permet à l'utilisateur de répondre une valeur puis cliquer "ok" ou cliquer "cancel" . Il y a un retour (la valeur si "ok" est appuyé ou null sinon). "valpardef" est la valeur par défaut du champ à remplir. S'il n'y en a pas, la valeur sera "undefined". Si vous voulez qu'il n'y ait rien, il faut passer "" en paramètre.
    - confirm(msg). De même pour confirm, qui laisse le choix entre "ok" et "cancel" et qui retourne "true" (ok) ou "false" (cancel).
    - Le paramètre "msg" est toujours un message (explicatif, une question).

Objet Web

- navigator
  - L'objet navigator permet d'obtenir des informations sur le navigateur via les propriétés appCodeName (nom de code ex : mozilla), appName (ex : Netscape), appVersion (la version) et platform (le système d'exploitation).
- document
  - L'objet document permet l'accès aux différents éléments (xhtml) qui composent le document. Outre les fonctions liées au DOM (voir plus loin), document a la méthode write("...") pour écrire dans le document.
  - Il a également les propriétés "lastModified" (donne la date de dernière modification) et URL (qui donne l'url du document, probablement codée par "escape(url)" à cause des caractères spéciaux, décodable par "unescape(url)").

DOM

- DOM est un acronyme pour Document Object Model. C'est une API (application programming interface) pour du html et xml valide.
- En suivant le DOM, les programmeurs peuvent construire des documents, naviguer dedans, ajouter, modifier ou effacer des éléments à ces documents.
- En tant que recommandation du W3C, l'objectif du DOM est de fournir une interface de programmation standard pour être utilisée par tous (applications, OS).
- Suivant le DOM, un document a une structure logique d'arbre (ou de forêt).
- Le nom DOM provient de l'approche retenue par le W3C : une approche proche des modèles objets (propriétés, méthodes, description).
- Le DOM a pour but d'uniformiser la manipulation des documents "web", notamment par les scripts.
- c.f. ECMAscript (une sorte de standardisation de javascript).
  - http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/ecma-script-binding.html

Quelques éléments du DOM

- document
  - Element createElement(in DOMString tagName)
  - NodeList getElementsByTagName(in DOMString tagname)
  - **Element getElementById(in DOMString elementId)**
- Document HTML (hérite de document)
  - title : titre de la page
  - referrer, domain, URL
  - body
  - images, applets, links, forms, anchors, cookie [des collections]
  - write(), writeln()
- NodeList
  - length
  - item(index)

Quelques éléments du DOM

- node
  - nodeName, nodeValue, nodeType , parentNode
  - childNodes [une NodeList]
  - insertBefore(newChild, refChild) , replaceChild(newChild, oldChild), appendChild(newChild) , removeChild(oldChild)
- element
  - tagName
- Element html
  - id
  - title
  - lang
  - dir
  - className [class CSS]
  - innerHTML : code html contenu dans la balise

Animation : exemple/anim.html

- Animation
  - comptes à rebours : setTimeout
    - la fonction setTimeout prend deux paramètres : en premier des instructions (entre "", séparées par des ';') (qui peuvent être un appel de fonction) et en second un délai en millisecondes. Ce délai est le temps à attendre avant l'exécution des instructions passer en premier paramètre.
  - déclenchement périodique : setInterval
    - setInterval fonctionne comme setTimeout, à la différence que les instructions passées en premier paramètre sont exécutées périodiquement (toutes les délai milisecondes).
  - stoper une minuterie : clearInterval
    - la fonction "clearInterval" permet de stopper un déclenchement périodique. Pour ce faire, il faut concerver un pointeur sur le retour de setInterval et de le passer en paramètre de la fonction clearInterval.
    - exemple : var inter = setInterval("alert('p\u00e9riode de 3 secondes');", 3000);...
      clearInterval(inter);
- Anim.html

AJAX

- Principe : faire une requête asynchrone
  - Pour vérifier
  - Pour obtenir
- Requête asynchrone HTTP (uniquement sur le serveur)
- On obtient en retour du XML
  - À exploiter… (c.f. getElementById + innerHTML)
- Le tout sans recharger…

Ajax : mécanismes

- Envoie : objet XMLHttpRequest
- http://www.w3.org/TR/XMLHttpRequest/

```
xmlhttpinside=new XMLHttpRequest();
xmlhttpinside.onreadystatechange= navcontext;
```

```
xmlhttpinside.open("GET",url,true);
xmlhttpinside.send(null);
```

Nom de la fonction javascript pour recevoir la réponse à la requête http

Si post, chaine de caractère du style "var1=va1l&var2=val2"

Envoie de la requête en post ou get

Recevoir les réponses de la requête

```
function navcontext( ) {
// test de l'etat d'avancement du telechargement (http request)
if ((xmlhttpinside.readyState==4) &&
    (xmlhttpinside.status==200))
        { // recuperation du fichier XML au format XML ou Text
        var myXML = xmlhttpinside.responseXML;
        var myXML = xmlhttpinside.responseText;
        // ...
        }
 }
```

Génération de la réponse

- Fonction header avec Content-type
  header('Content-type: text/xml;');
- http://deptinfo.unice.fr/~renevier/L3/cours5/getCaddie.php.txt

- http://deptinfo.unice.fr/~renevier/L3/cours5/generationXMLenPhp.html

Javascript et Php

- Le php peut générer du javascript
- Javascript et Php peuvent communiquer par requête
  - La génération du xml s'intègre alors à l'architecture logiciel…
  - Exemple en PAC : un getView qui retourne du HTML encapsulé dans du php
    <![CDATA[….]]>
  - On récupère avec reponseText et on applique avec getElementById et innerHTML…