AJAX

Table of Contents

1 L'arbre DOM

1.1 Introduction

La page web est organisée en mémoire sous une forme d'arbre. Le nœud racine est la balise `<html>` dans laquelle on trouve les balises `<head>` et `<body>`, ...

Lorsque l'on veut modifier la page web au travers du javascript, il est nécessaire d'agir au travers de l'arbre DOM. Cela permet aussi bien d'accéder à un élément en lecture que de modifier la totalité de la page.

1.2 Accéder à un élément

1.2.a getElementById()

La manière la plus simple est de retrouver l'élément par la valeur de l'attribut `ID` qui est précisée dans l'élément. Pour ce faire, on utilisera la fonction[1] *getElementById()* qui sera appelée sous la forme

```
el=base_el.getElementById(contenu_id);
```

où la variable *el* contiendra l'élément concerné, *base_el* est une variable contenant l'élément à partir duquel la recherche est effectuée et *contenu_id* est la chaîne de caractères qui se trouve dans l'id de l'élément recherché.

1.2.b getElementsByTagName()

Il est également possible de retrouver tous les éléments d'un type donné à l'aide de *getElementsByTagName()*[2] qui retourne un tableau d'éléments. On pourra l'utiliser de la manière suivante :

```
var all_span=base_el.getElementsByTagName("span");
var num_span=all_span.length;
var i;
for (i=0;i<num_span;i++) {
   var el=all_span[i];
   ...
}
```

Ici, la boucle parcourra tous les éléments `<span>` situés à l'intérieur de l'élément *base_el*.

On peut également utiliser la notation

```
   var el=all_span.item(i);
```

1.2.c base du document HTML : *document*

Si on désire examiner la totalité du document HTML, on utilisera l'élément fictif[3] *document* comme *base_el*.

1 De par la nature "objet" du javascript, il faudrait plutôt parler de méthode
2 A noter qu'ici, le mot "elements" est au pluriel
3 Il s'agit en fait d'un objet, tout comme les éléments

On obtiendra ainsi *document.getElementById()* ou *document.getElementsByTagName()* qui permettront de faire la recherche sur la totalité de la page telle qu'elle est affichée (et non telle que téléchargée à l'origine)

1.2.d *parentNode* et *firstChild*

Si on désire retrouver le premier élément contenu dans un élément donné, on utilisera le champ[4] firstChild de l'élément.

```
var child=el.firstChild;
```

De même, si on désire remonter à l'élément père d'un élément donné (l'élément qui le contient), on utilisera le champ parentNode de l'élément.

```
var parent=el.parentNode;
```

1.3 Modification de la structure

1.3.a *innerHTML*

La propriété innerHTML d'un élément permet de lire ou écrire le contenu d'un élément directement sous la forme d'une chaîne de caractères en HTML.

```
el.innerHTML='<span>Hello World</span><img src="img/icon.jpg" />';
```

D'un autre côté, cette méthode permet d'injecter un code HTML incorrect et ne permet pas d'accrocher des gestionnaires d'événements ou des propriétés calculées ou supplémentaires aux éléments contenus dans la chaîne en HTML.

Une utilisation courante est d'effacer rapidement la totalité du contenu d'une division :

```
div.innerHTML="";
```

1.3.b *createElement()*

La méthode *createElement()* permet de créer un élément HTML. On pourra par la suite modifier ses propriétés et l'attacher dans l'arbre DOM afin de le faire apparaître dans la page web.

```
var el=document.createElement("input");
el.type="button";
el.value="OK";
el.id="OKButton";
```

L'élément sauvé dans la variable aura les mêmes caractéristiques que les éléments obtenus par *getElementById()* et *getElementsByTagName()*. On pourra donc les exploiter de la même manière (changement de style par exemple).

1.3.c *appendChild()*

Pour attacher dans l'arbre DOM un élément généré par *createElement()*, on utilisera la méthode *appendChild()*.

Dans le cas où l'on désire accrocher un ensemble d'éléments imbriqués (une table par exemple), on effectuera les *appendChild()* en partant des éléments les plus à l'intérieur pour remonter vers l'élément qui contient les autres. Cela permettra d'éviter des modifications répétées de la page affichée.

4 On parlera de propriété

```
var baseel=document.getElementById("tablediv");
var table=document.createElement("table");
var tr=document.createElement("tr");
var td=document.createElement("td");
tr.appendChild(td);
td=document.createElement("td");
tr.appendChild(td);
table.appendChild(tr);
baseel.appendChild(table);
```

1.3.d *removeChild()* et *replaceChild()*

Lorsque l'on désire retirer un élément (ou un ensemble d'éléments), on utilise la méthode *removeChild()*. Cette méthode s'utilise de la même manière que *appendChild()* et détache l'élément précisé (et tous ses éléments fils) de l'arbre DOM.

La méthode *replaceChild()* permet elle de remplacer un élément par un autre. Le nouvel élément est placé là où se trouvait l'ancien (alors qu'un *removeChild()* suivi d'un *appendChild()* placera le nouvel élément à la fin de la liste).

```
var el1=document.getElementById("DeleteMe");
var baseel1=el1.parentNode;
baseel1.removeChild(baseel);

var el2=document.getElementById("ChangeMe");
var elnew=document.createElement("div");
var baseel2=el2.parentNode;
baseel2.replaceChild(elnew,el2);
```

1.4 Le texte

Dans l'arbre DOM, les morceaux de texte se trouvent dans des nœuds d'un type spécial. Lorsqu'un élément contient un mélange de texte et d'autres éléments, chaque morceau de texte se trouve dans un nœud texte séparé.

1.4.a *createTextNode()*

Pour créer un nouveau nœud texte, on utilisera la méthode *createTextNode()*. Ce noeud pourra ensuite être ajouté dans l'arbre DOM comme on ajouterait d'autres éléments.

```
var el=document.getElementById("MyText");
var tn=document.createTextNode("Hello World");
el.appendChild(tn);
```

1.4.b *nodeValue*

Lorsque l'on désire lire ou modifier la valeur d'un nœud texte, on utilise la propriété *nodeValue* du nœud en question.

```
var tn=document.getElementById("Span1").firstChild;
tn="Hello World";
var msg=tn.nodeValue;
```

1.4.c *getAttribute()*

Pour lire la valeur d'un attribut dans un élément XML, on utilisera simplement la méthode *getAttribute()* qui prend en paramètre le nom de l'attribut et en retourne le texte.

```
attrib=el.getAttribute("disabled");
```

2 Les événements

2.1 Introduction

Un site AJAX étant une application interactive, il s'agit d'une programmation événementielle.

Le fil d'exécution est déterminé par différents événements qui peuvent provenir d'actions de l'utilisateur (boutons, déplacement de souris, clavier) mais également par l'arrivée de données demandées au serveur web (page, frame ou image chargée, timeout, contenu demandé par l'objet XMLHttpRequest, ...)

La gestion des événements est donc un aspect important de l'application.

2.2 Evénements dans l'arbre DOM

2.2.a Les événements liés aux éléments

Les éléments dans l'arbre DOM supportent une série d'événements. Chaque type d'élément supportera des événements différents.

onload	L'élément a été chargé du serveur web	FRAME, BODY, IMG,...
onclick	L'utilisateur a cliqué sur l'élément concerné	Tous les éléments sauf IFRAME[5]
onmousedown onmouseup	L'utilisateur a appuyé sur/relaché un des boutons de la souris	Tous les éléments sauf IFRAME
onmouseover onmousemove onmouseout	La souris est entrée dans/s'est déplacée dans/ a quitté l'élément.	Tous les éléments sauf IFRAME
onkeydown onkeyup	L'utilisateur a enfoncé/relaché une touche du clavier alors que le focus est sur l'élément ou un des éléments inclus dans celui qui gère l'événement	Tous les éléments sauf IFRAME
onresize	Les dimensions de l'élément ont changé	Principalement utilisé sur BODY
onfocus onblur	L'élément a reçu/perdu le focus	A(HREF), INPUT, TEXTAREA, SELECT
onchange	La valeur de l'élément a changé	SELECT, INPUT(TEXT, PASSWORD, RADIO, CHECKBOX), TEXTAREA

Si le gestionnaire d'événement retourne true ou ne retourne rien, l'action par défaut sera effectuée et l'événement sera transmis à l'élément qui contient celui sur lequel le gestionnaire se trouve. Si le gestionnaire retourne false, l'action par défaut ne sera pas effectuée et l'événement ne sera pas remonté.

5 Les événements click sur une IFRAME sont envoyés à la page affichée dans cette dernière et non à la page qui contient l'IFRAME

2.2.b Attacher un événement

2.2.b.1) Attribut dans une balise HTML

On peut préciser un code javascript dans la balise HTML sous la forme

```
<div onclick="code_javascript">
```

Le javascript inclus ainsi ne pourra pas contenir de " vu que ces derniers servent à délimiter l'attribut.

Cette façon de procéder peut cependant poser problème lorsque l'on veut exploiter l'objet Event et le comportement exact peut dépendre du navigateur internet.

2.2.b.2) on... sur l'élément

Lorsque l'on dispose de l'élément DOM dans une variable (soit après un *getElementById()* ou autre, soit parce qu'on l'a créé avec *document.createElement()*), on peut directement associer le gestionnaire d'événement sous la forme

```
var el=document.getElementById("xyz");
el.onclick=fonction;
```

Lorsque la fonction sera appelée, elle recevra automatique l'objet Event en paramètre. Il n'est cependant pas possible de préciser d'autres paramètres à la fonction chargée de gérer l'événement.

2.2.b.3) addEventListener

Le W3C propose une méthode *addEventListener()* qui permet d'ajouter un événement à un objet. Contrairement aux autres méthodes présentées ci-dessus, il est possible d'associer plusieurs gestionnaires d'événements à un même élément.

Le nom de l'événement à utiliser ne contient pas le "on"... On aura ainsi "click", "change", "load", ... et non "onclick", "onchange", "onload", ...

MS Internet Explorer (jusqu'à la version 8 et dans les versions 9 et suivantes en mode "compatibilité ancienne version"[6]) utilise à la place la méthode *attachEvent()*. Il sera donc nécessaire de déterminer quelle est la méthode qui est effectivement disponible

```
var el=document.getElementById("xyz") ;
if (el.addEventListener) {
  el.addEventListener("click",fonction,false);
} else {
  el.attachEvent("onclick",fonction);
}
```

Il est possible de retirer un gestionnaire d'événement en utilisant les méthodes *removeEventListener()* et *detachEvent()* qui prennent les mêmes paramètres[7].

2.2.c L'objet Event

Lorsqu'un gestionnaire d'événement est appelé, il reçoit comme paramètre un objet Event (cet objet n'est pas disponible si le gestionnaire d'événement a été associé à l'aide d'un attribut dans l'élément comme décrit en 2.2.b.1)

6 Ce mode sera déclenché par certains DOCTYPE ou peut avoir été configuré au niveau du navigateur.
7 A noter que cela n'est pas possible si on a utilisé des fonctions anonymes

Cet objet contient des informations supplémentaires sur l'événement (bouton enfoncé par exemple) mais contiendra également une référence à l'élément qui a déclenché l'événement (le bouton cliqué, le contrôle modifié, ...) et ce, même si le gestionnaire d'événement a été placé sur un élément qui contient celui qui est à l'origine de l'événement. On trouve cet élément dans la propriété *target* de l'objet Event.

Dans le cas de MS Internet Explorer, l'événement peut ne pas être passé en paramètre à la fonction, il faut alors le récupérer dans *window.event*. De même, l'élément où a eu lieu l'événement peut ne pas se trouver dans *target* mais dans *srcElement*. Cela dépendra de la version et du mode de fonctionnement (il n'est donc pas suffisant de tester si on est sous MS Internet Explorer).

On utilisera dès lors le code suivant :

```
function onOKClick(ev) {
   if (!ev) ev=window.event;
   var targ=ev.target;
   if (!targ) targ=ev.srcElement;
   ...
}
```

2.2.d Quelques propriétés de Event

2.2.d.1) Touche enfoncée

Pour lire la touche enfoncée, on utilisera le code suivant :

```
var code = ev.keyCode;
if (!code) code = ev.which;
```

La variable code contiendra le "scancode" de la touche (qui permet de désigner toutes les touches, y compris les touches de fonction, de curseur, ...) Si on désire récupérer la lettre correspondante, on utilisera

```
var character = String.fromCharCode(code)
```

pour faire la conversion.

2.2.d.2) Position de la souris

Selon les navigateurs, on trouve 6 paires de propriétés qui fournissent la position de la souris (chaque navigateur pouvant en définir plusieurs).

Une méthode plus ou moins fiable pour repérer le pointeur de souris est la suivante :

```
function onMouseEvent(ev) {
   var posx = 0;
   var posy = 0;
   if (!ev) ev = window.event;
   if (ev.pageX || ev.pageY) {
      posx = ev.pageX;
      posy = ev.pageY;
   }
   else if (ev.clientX || ev.clientY) {
      posx = ev.clientX + document.body.scrollLeft
         + document.documentElement.scrollLeft;
      posy = ev.clientY + document.body.scrollTop
         + document.documentElement.scrollTop;
   }
}
```

posx et posy contiendront la position du pointeur de souris par rapport au début du document. Si on désire connaître la position par rapport à l'élément, il "suffit" de soustraire la position de l'élément.

Pour la position par rapport à la fenêtre, on peut utiliser *ev.clientX* et *ev.clientY*.

2.3 Autres événements

Il est possible de programmer des événements qui doivent se déclencher à un moment donné. On utilisera pour cela *setTimeout()* et *setInterval()*. *setTimeout()* appellera la fonction une fois alors que *setInterval()* l'appellera de manière répétée.

Le premier paramètre est la fonction à exécuter lorsque le délai est expiré, le second est le délai avant exécution (ou répétition) exprimé en millisecondes.

Ces fonctions retournent un identifiant qui permettent d'annuler le timer à l'aide de *clearTimeout()* ou de *clearInterval()*.

```
var tmr=setTimeout(onTimerDone,2000);
clearTimeout(tmr);
```

3 Les fonctions

3.1 Fonctions anonymes

En Javascript, la fonction est une entité comme un nombre, une chaîne de caractères, un élément, un objet, ... Elle peut être sauvée dans une variable, transmise comme paramètre lors d'un appel de fonction, retournée par une fonction, ...

Lorsque l'on défini une fonction par

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

ce qui est fait en réalité est de créer une fonction avec un paramètre et la sauver dans la variable globale *carre*.

La fonction pourrait également être définie de la façon suivante :

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

Ici, on sauve la fonction à un paramètre sans nom dans la variable *carre*. Ce type de fonction est appelé **fonction anonyme**.

Lorsque l'on veut définir un gestionnaire d'événements, on peut utiliser une fonction anonyme plutôt que de devoir définir une fonction nommée :

```
el.onclick = function (ev) {
  if (!ev) ev=window.event;
  var targ=ev.target;
  if (!targ) targ=ev.srcElement;
  var tn = document.createTextNode("X");
  targ.parentNode.replaceChild(tn,targ);
};
```

A noter le point virgule après l'accolade fermante qui termine l'instruction `el.onclick= ...;`

3.2 Closure

3.2.a Contexte de variables

Lorsqu'on exécute une fonction, ses variables locales et ses paramètres forment son contexte. Ce contexte reprend la liste de tous les noms de variables et de paramètres ainsi que les références vers leurs contenus.

Ainsi, lorsque l'on exécute la fonction suivante :

```
function (a,b) {
  var c=document.getElementById("xxx");
  var d=5;
  c.innerHTML=d;
}
```

le contexte associera les symboles a, b, c et d aux contenus de ces variables.

A noter qu'il s'agit de références vers la valeur de la variable. C'est ainsi qu'on ne fait pas de copie de l'élément dont l'ID est xxx dans la variable c (si le symbole avait été associé à une copie de l'élément, une modification n'aurait pas changé l'élément d'origine et n'aurait pas eu d'effet sur la page web).

De même, le symbole d n'est pas associé à la valeur 5 mais à une zone mémoire qui contient cette valeur.

Lorsque la fonction se termine, le contexte est effacé mais pas les valeurs référencées. Ces valeurs ne seront effacées que lorsqu'il n'existera plus aucune référence vers elles[8]. C'est ainsi qu'un élément créé dans une fonction et attaché à l'arbre DOM ne disparaîtra pas lorsque la fonction se terminera.

Pour les paramètres, certains types de données seront copiés lors de l'appel de fonction (comme les chaînes de caractères et les nombres) et d'autres pas (les fonctions, les éléments, ...)

3.2.b Héritage du contexte

Lorsqu'une fonction anonyme est créée à l'intérieur d'une autre fonction, elle hérite du contexte de la fonction dans laquelle elle est créée.

On pourra donc, à l'intérieur de cette fonction anonyme, accéder aussi bien aux variables liées au contexte de la fonction anonyme qu'à celles de la fonction dans laquelle elle a été créée.

```
function A(b,c) {
   var d;
   ...
   ... = function (z) {
      var y ;
      ...
   };
   ...
}
```

Le contexte de la fonction A contiendra les variables b, c et d. Le contexte de la fonction interne contiendra les variables y et z (ce contexte sera créé à chaque fois que la fonction interne sera appelée).

La fonction interne aura accès à toutes ces variables (b, c, d, y, z). On parle de closure pour désigner la copie du contexte de la fonction externe (ici, la closure contient les variables b, c et d)

3.2.c Problème de la boucle

Le contexte associé à la fonction interne possède une référence vers la valeur de la variable. Lorsque cette variable est modifiée, la référence ne change pas mais la valeur change.

Si on utilise une boucle pour générer des boutons auxquels on associe des gestionnaires d'événement, la valeur disponible au niveau de la closure sera celle au moment où l'événement se déclenchera. Ce sera donc la valeur de la variable à la fin de l'exécution de la fonction externe (il est peu probable que l'événement se déclenche avant)

```
function mkInput() {
   var base=document.getElementById("xxx");
   var i, inp;
   for (i=0;i<5;i++) {
      inp=document.createElement("input");
      inp.type="button";
      inp.value="Affiche "+i;
      inp.onclick= function() { alert("Bouton "+i+" enfoncé"); };
      base.addChild(inp);
   }
}
```

Dans l'exemple ci-dessus, on crée 5 boutons appelés "Affiche 0", "Affiche 1", ... mais, quand on cliquera sur un bouton, le message sera toujours "Bouton 5 enfoncé".

8 Cet effacement se fera lors d'un processus appelé "garbage collect"

Le problème vient du fait que la valeur de i utilisée par les différentes fonctions "onclick" est celle au moment où le bouton est enfoncé. A ce moment, la fonction *mkInput()* aura eu le temps de se terminer et la valeur de i à ce moment est de 5.

3.2.d Contournement du problème de la boucle

Il existe plusieurs moyens pour contourner le problème cité en 3.2.c. L'idée derrière ces méthodes est de créer un nouveau contexte à chaque fois que l'on associe la fonction interne à une ressource (événement, cellule dans un tableau, ...)

3.2.d.1) Fonction d'association d'événement

On peut utiliser une fonction intermédiaire qui sera chargée de sauver la fonction dans un gestionnaire d'événement. Cette fonction intermédiaire prendra la valeur qui change comme paramètre. A chaque fois qu'elle sera appelée, un nouveau contexte sera créé et la valeur courante sera sauvée dans ce contexte.

```
function addCallBack(el,i) {
  el.onclick=function() { alert("Bouton "+i+" enfoncé"); };
}
function mkInput() {
  var base=document.getElementById("xxx");
  var i, inp;
  for (i=0;i<5;i++) {
    inp=document.createElement("input");
    inp.type="button";
    inp.value="Affiche "+i;
    addCallBack(inp,i);
    base.addChild(inp);
  }
}
```

A noter également que la closure ne contiendra que les variables *el* et *i* (paramètres de *addCallBack()*) et ne contiendra pas les variables *base* et *inp* de la fonction *mkInput()* (ni la variable *i* de la fonction *mkInput()* d'ailleurs qui pourra donc être libérée par le Garbage Collector).

3.2.d.2) Fonction génératrice

Comme une fonction peut en retourner une autre, au lieu d'associer directement la fonction interne au gestionnaire d'événement, on peut retourner la fonction à *mkInput()* et laisser *mkInput()* faire cette association.

```
function genCallBack(num) {
  return function() { alert("Bouton "+num+" enfoncé"); };
}
function mkInput() {
  var base=document.getElementById("xxx");
  var i, inp;
  for (i=0;i<5;i++) {
    inp=document.createElement("input");
    inp.type="button";
    inp.value="Affiche "+i;
    inp.onclick=genCallBack(i);
    base.addChild(inp);
  }
}
```

Ici, la closure de la fonction interne ne contiendra que la valeur *num* et n'aura plus à contenir l'élément dans lequel cette fonction est accrochée.

3.2.d.3) Tout dans l'anonymat

L'étape suivante peut être de rendre anonyme cette fonction intermédiaire (*addCallBack()* ou *genCallBack()*). Cela amène à un code très concis mais qui est cependant moins efficace.

```
function mkInput() {
  var base=document.getElementById("xxx");
  var i, inp;
  for (i=0;i<5;i++) {
    inp=document.createElement("input");
    inp.type="button";
    inp.value="Affiche "+i;
    inp.onclick=( function (num) {
      return function() { alert("Bouton "+num+" enfoncé");};
      } ) (i);
    base.addChild(inp);
  }
}
```

A noter que les parenthèses autour de la définition de la fonction génératrice ont la même signification que celles que l'on peut mettre dans (4+5)*6 et que celles autour du paramètre *i* servent à effectuer un appel de fonction.

Par rapport au cas précédent, la fonction génératrice doit être créée à chaque passage dans la boucle, au lieu d'avoir été créée une seule fois ce qui peut poser des problèmes de lenteur.

4 XMLHttpRequest

4.1 Introduction

L'objet XMLHttpRequest permet de faire une requête au serveur Web à partir du Javascript. Attention, on ne peut interroger que le serveur d'où vient la page web (pour des raisons de sécurité).

Même s'il existe d'autres méthodes pour effectuer ce transfert (dont une utilisée pour les envois de fichiers), il s'agit de la principale méthode utilisée pour cette communication.

Elle permet d'effectuer les différents types de requêtes[9]

4.2 Créer l'objet sous tous les navigateurs

Si Mozilla/Firefox, Opera et Chrome l'ont supporté dès le début, Microsoft Internet Explorer a préféré obliger à créer un objet ActiveX (leur technologie propriétaire) et il est donc nécessaire de procéder différemment selon les navigateurs internet. Les versions les plus récentes de Microsoft Internet Explorer supportent maintenant l'objet XMLHttpRequest.

On utilisera donc un code comme celui présenté ci-dessous qui testera les différents cas de figure.

```
function getXHR() {
  var xhr = null;
  try {
     xhr=new XMLHttpRequest();
  } catch (e) {
     try {
        xhr=new ActiveXObject('Msxml2.XMLHTTP');
     } catch (e) {
        try {
           xhr= new ActiveXObject('Microsoft.XMLHTTP');
        } catch (e) {
           xhr = null;
        }
     }
  }
  return xhr;
}
```

Si l'objet n'a pas pu être créé, la fonction retourne *null*.

4.3 Les requêtes GET

4.3.a Effectuer la requête

Pour effectuer une requête **GET** sans paramètre sur une page, il suffit d'instancier un objet XMLHttpRequest, de s'en servir pour *ouvrir* une page web (généralement PHP ou semblable) et envoyer la requête.

```
var xhr=getXHR();
xhr.open("GET","fichier.php",true);
xhr.send(null);
```

9 GET et POST mais également HEAD, PUT, DELETE et autres

Le *true* dans l'instruction xhr.open() sert à indiquer que la demande est asynchrone[10], c'est-à-dire que le navigateur n'attend pas que la requête soit terminée pour continuer l'exécution du javascript.

Lorsque le *send()* sera effectué, la requête HTTP sera réalisée. L'aspect asynchrone a pour effet que le code javascript continue à s'exécuter et la fonction qui aura effectué cette requête peut donc se terminer. Si on désire conserver une référence à l'objet xhr, il sera donc nécessaire d'utiliser le mécanisme de closure[11].

4.3.b Passer des paramètres

Pour une requête **GET**, les paramètres sont passés dans l'URL, après un caractère "?".

Chaque paramètre apparaîtra sous la forme nom=valeur et les paramètres sont séparés par le caractère "&".

Il est cependant nécessaire d'encoder certains caractères spéciaux. Par exemple, le caractère & sert à séparer les différents paramètres transmis. Pour ce faire, on utilise la fonction *encodeURIComponent()*[12].

Il faut également noter que la taille de la requête est limitée (la limitation varie selon le navigateur et le serveur web) et qu'il faut donc éviter d'utiliser des paramètres GET si certains paramètres peuvent être relativement longs ou si les paramètres sont trop nombreux.

```
var url="ajax/send_data.php";
var param="param1="+encodeURIComponent(valeur1);
param=param+"&param2="+encodeURIComponent(valeur2);
          [...]
var xhr=getXHR();
xhr.open("GET",url+"?"+param,true);
xhr.send(null);
```

4.4 Les requêtes POST

Une requête **POST** permet d'envoyer des paramètres d'une taille quelconque (dans les limites configurées au niveau du serveur Web et du PHP).

La chaîne de paramètres est créée comme pour une requête **GET** (nom=valeur&nom=valeur...) mais elle est transférée différemment au serveur web.

```
var url="ajax/send_data.php";
var param="param1="+encodeURIComponent(valeur1);
param=param+"&param2="+encodeURIComponent(valeur2);
          [...]
var xhr=getXHR();
xhr.open("POST",url,true);
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
xhr.send(param);
```

Il est nécessaire de préciser le type de données envoyées afin que le PHP puisse les interpréter et les sauver dans le tableau $_POST[].

10 Le A de AJAX est pour Asynchronous

11 Il serait également possible d'utiliser une variable globale mais cela peut poser certains problèmes, surtout lorsque la communication réseau est lente.

12 Attention, certains tutoriaux sur internet utilisent de manière erronée encodeURI(). Cette fonction ne doit en AUCUN CAS être utilisée pour encoder les paramètres dans la requête car certains caractères ne seront pas encodés correctement.

Les deux autres en-têtes permettent au PHP de connaître la taille de la requête (et donc, de savoir quand il a reçu tous les paramètres) et de fermer la connexion une fois la requête terminée (sinon, certaines combinaisons de navigateur et de serveur peuvent garder la connexion ouverte indéfiniment).

4.5 Récupérer le résultat

4.5.a Gestion de l'asynchronisme

Comme la requête est asynchrone, le seul moyen de réagir quand le résultat est disponible est d'utiliser un événement. A cette fin, on dispose de l'événement *onreadystatechange* qui se déclenchera a différents moments clés de la communication entre l'objet XMLHttpRequest et le serveur[13]

L'objet XMLHttpRequest dispose d'une propriété *readyState* qui vaudra 4 lorsque le transfert avec le serveur sera terminé. Il dispose également d'une propriété status qui indiquera le status HTTP de la requête (200 pouir OK, 404 pour Page non trouvée, ...).

Une fois que readyState vaudra 4, si status vaut 200, le résultat de la requête sera disponible. Sinon, la valeur du status permettra d'identifier l'erreur[14]

```
function requeteAJAX() {
  var xhr=getXHR();
  xhr.onreadystatechange= function() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {

        [TRAITEMENT RESULTAT]

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

Il est souvent plus simple d'appeler une fonction pour effectuer le traitement effectif des résultats (surtout si ce traitement est assez complexe).

4.5.b Lire les résultats

Le texte envoyé par le serveur lors de la requête AJAX sera disponible tel quel dans la propriété *responseText* de l'objet XMLHttpRequest. Il peut s'agir d'un simple texte mais également d'une chaîne formatée de manière à pouvoir être décomposée en plusieurs éléments par le javascript (en utilisant la méthode *split()* par exemple) ou d'un morceau de HTML (qui ne devra pas contenir les balises <HTML>, <HEAD> ou <BODY>).

Si le type transmis par le navigateur est **text/xml** et que le texte envoyé par le serveur est un document XML valide, il sera également disponible sous forme d'arbre dans la propriété *responseXML*.

Il est possible de récupérer un des en-têtes HTTP à l'aide de la méthode *getResponseHeader()*.

13 A noter que sous certaines versions de Microsoft Internet Explorer, seule la dernière étape peut être utilisée de manière fiable.

14 En pratique, on ignore souvent les résultats autres que 200.

```
function requeteAJAX() {
  var xhr=getXHR();
  xhr.onreadystatechange= function() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        requeteAJAX_cb(xhr.responseText);
      }
    }
  };
  xhr.open("GET","get_data.php",true);
  xhr.send(null);
}

function requeteAJAX_cb(str) {

  [TRAITEMENT DU RESULTAT]

}
```

5 XMHHttpRequest et XML

Il est possible de transmettre directement un objet XML du PHP vers le javascript. L'objet XML permet de communiquer simplement des données structurées.

5.1 Le XML

5.1.a Syntaxe XML

- La syntaxe d'un fichier XML est proche de celle d'un fichier HTML
- Le document XML est composé d'un ensemble d'éléments organisés en arbre et délimités par des balises
- Une ligne d'en-tête permet d'identifier le document comme étant un document XML et de préciser l'encodage
- Un élément peut contenir d'autres éléments et/ou du texte
- On peut préciser des informations complémentaires (méta-data[15]) à un élément sous forme d'attributs

5.1.b En-tête XML

Tout document XML doit commencer par une ligne d'en-tête de la forme suivante[16] :

```
<?xml version="1.0" ?>
```

Cette ligne permet à certains programmes d'identifier le fichier comme étant un fichier XML (magic name[17]). Elle permet également de fournir quelques informations supplémentaires concernant le fichier.

L'option qui est la plus souvent présente est l'option `encoding` qui permet de définir le jeu de caractères utilisé. L'encodage préféré pour le XML est le UTF-8.

```
<?xml version="1.0" encoding="utf-8"?>
```

5.1.c Eléments et balises

Un élément est composé d'une balise ouvrante, d'un contenu et d'une balise fermante.

La balise ouvrante est composée du caractère <, d'un nom, éventuellement d'attributs et du caractère >. La balise fermante est composée des caractères </, du nom de la balise ouvrante correspondante et du caractère >

```
<nom> ... </nom>
```

Le nom d'une balise doit commencer par une lettre et peut être composé de lettres, chiffres et de quelques caractères spéciaux (à éviter sauf le caractère de soulignement _). Les majuscules et minuscules sont des caractères différents. Les noms ne peuvent pas commencer par le mot XML (que ce soit en majuscules, en minuscules ou un mélange des deux).

15 Des méta-data sont des informations qui servent principalement à ordonner, identifier ou décrire les données. Par exemple, un numéro de référence, un attribut signalant que la valeur a été modifiée, ...

16 Il s'agit de la forme des *Processing Instructions*, également utilisée quand on insère du PHP dans de l'HTML

17 Le *magic name* est une séquence d'octets qui permet d'identifier le type d'un fichier. Par exemple, les images GIF commencent par les 5 caractères GIF87 ou GIF89 et les archives .ZIP par les deux lettres PK

Un élément sans contenu peut s'écrire à l'aide d'une balise *auto-fermante*. Cette balise est composée du caractère <, du nom, des attributs (s'il y en a) et des caractères />.

```
<nom />
```

5.1.d Les attributs

Les attributs sont de la forme nom="valeur" ou nom='valeur'. Contrairement au HTML 4 (et précédents), il est obligatoire de fournir une valeur.

Dans une balise, on peut préciser autant d'attributs que l'on désire mais ils doivent avoir des noms différents.

Les noms des attributs suivent les mêmes règles que les noms des balises.

```
<adresse type="maison">Square Hiernaux, 2</adresse>
```

5.1.e Le contenu d'un élément

Un élément peut contenir du texte, d'autres éléments ou un mélange des deux.

Lorsque l'on insère des éléments dans un autre élément, il est nécessaire de respecter les règles d'imbrication et de fermer les éléments intérieurs avant l'élément extérieur.

Contrairement à l'HTML où un nombre quelconque d'espaces, de retours de ligne et de tabulations sont remplacés par un espace unique, en XML les espaces sont conservés tels quels.

5.1.f Les entités

Certains caractères ont une signification spéciale en XML. Dès lors, lorsque l'on veut les faire apparaître dans un attribut ou dans un élément, il faut utiliser l'entité correspondante.

Caractère	Entité	description
<	&lt;	Plus petit que (less than)
>	&gt;	Plus grand que (greater than)
&	&amp;	ET commercial (ampersand)
'	&apos;	Apostrophe
"	&quot;	Guillemet (quote)

5.1.g Les blocs CDATA

Lorsqu'un grand bloc de données doit être pris tel quel, sans aucun traitement (par exemple, un morceau de code en C, qui contient des caractères tels que <, >, &,...), plutôt que de remplacer tous ses caractères spéciaux par les entités correspondantes, on peut l'enfermer dans un bloc CDATA.

La syntaxe est la suivante :

```
<listing>
<![CDATA[
if ((c >= 'a') && (c <= 'z')) c-=0x20;
]]>
</listing>
```

En pratique, dès que le texte XML qui doit être renvoyé pourra potentiellement contenir des caractères tels que <, > et &, on utilisera un CDATA.

5.1.h Les commentaires

Pour préciser un commentaire dans le document XML, on l'enferme entre <!-- et -->

```
<!-- commentaire -->
```

5.2 Génération de XML à partir du PHP

5.2.a Envoi d'un fichier XML en PHP

Un fichier XML étant un simple fichier texte, il peut être généré par des instructions *echo*.

Pour que le navigateur web puisse reconnaître que les données sont un fichier XML, il faut le prévenir en envoyant un en-tête décrivant le type de contenu.

Ceci doit être fait AVANT que le moindre texte n'ait été envoyé au navigateur[18].

```
<?php
header("Content-Type: text/xml");
echo '<?xml version="1.0" ?>';
echo '<racine><element>texte</element></racine>';
?>
```

5.2.b Documents "HERE"

Lorsque l'on doit définir une longue chaîne en PHP, on peut utiliser une forme spéciale appelée *document here*.

La chaîne commence par <<< suivi d'un mot et finit lorsque l'on a une ligne qui ne contient que ce mot suivi d'un ; terminant la commande (attention, le mot doit **commencer** la ligne, sans aucun caractère avant, pas même des espaces ou des tabulations)

```
echo <<<EOT
Voici un exemple de chaîne qui est répartie sur plusieurs lignes.
Les variables sont étendues comme dans une chaîne entre
guillemets. $name sera donc remplacé par le contenu de la variable name
EOT;
```

Dans un *document here*, on peut préciser des variables qui seront remplacées par leur contenu, comme c'est le cas pour les chaînes entre "". Les caractères guillemets n'ont pas besoin d'être échappés.

On peut aussi utiliser les *documents here* dans une assignation de variable :

```
$v= <<<FIN
Voici une variable définie par "document here".
FIN;
```

Les *documents here* permettent donc de renvoyer facilement un texte XML à partir du PHP.

5.3 Lire le XML à partir du javascript

Lorsque l'on fait une requête XMLHttpRequest, si le RPC retourne un objet XML (entête signalant le XML et contenu XML valide), on dispose de l'objet XML dans la propriété responseXML.

Cet objet peut être manipulé avec les mêmes méthodes que l'arbre DOM pour la page HTML (*getElementByID*, *getElementsByTagName*, *parentNode*, *firstChild*, *nodeValue*, ...)

18 tout comme quand on envoie un cookie ou lorsqu'on utilise une session

L'objet retourné est le nœud racine. Il n'est donc pas possible de le rechercher par les fonctions de navigation.

```
function requete() {
  var xhr=getXHR() ;
  xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
      if (xhr.status == 200) {
        requete_cb(xhr.responseXML);
      }
    }
  };
  xhr.open("GET","exemple.php",true);
  xhr.send(null);
}

function requete_cb(xml) {
  var subtree,listelem,elem,i,nomel,nom;

  subtree=xml.getElementById("brancheXML");

  listelem=xml.getElementsByTagName("joueur");
  for (i=0;i<listelem.length;i++) {
    elem=listelem[i];
    nomel=elem.getElementsByTagName("nom");
    if (nomel[0]) {
      if (nomel[0].firstChild) {
        nom=nomel[0].firstChild.nodeValue;
      }
    }
  }
}
```

subtree contiendra le nœud qui possède un attribut id égal à brancheXML (ou null si l'id n'est pas présent),

listelem sera un tableau contenant les élements <joueur>.

On parcourt le tableau listelem et à chaque itération, elem pointera sur un des éléments <joueur> et permettra d'accéder aux éléments contenus dans ce dernier

nomel est l'élément <nom> présent dans l'élément <joueur> courant. Avant d'en extraire le texte (firstChild.nodeValue), il est préférable de s'assurer qu'il y a bien un élément <nom> présent et qu'il possède bien un contenu.

A noter que, si côté PHP il est nécessaire de vérifier que les données reçues sont correctes pour des raisons de sécurité (accès au script en contournant les vérifications effectuées par le javascript), ici, le code s'exécute sur la machine de l'utilisateur. Il n'est donc pas nécessaire de se prémunir d'éventuelles actions malicieuses de sa part.

Dès lors, si le script PHP envoie toujours un élément, il n'est pas nécessaire de tester sa présence. De même, si le script PHP envoie toujours un contenu non vide et non blanc dans un élement, il n'est pas nécessaire non plus de vérifier sa présence.

Ainsi, si on sait que le nom ne sera jamais une chaîne vide[19] et que l'élément nom sera toujours présent, les deux conditions (nomel[0] et nomel[0].firstChild) n'ont pas besoin d'être testées.

19 Un nœud qui ne contient que des espaces et des retours de chariot est considéré comme vide

I. Compléments de HTML

HTML 5

La structure de base d'un document HTML 5 est la suivante :

```
<!doctype html>
<html lang="fr">
<head>
<meta charset="UTF-8" />
<title>...</title>
</head>
<body>
...
</body>
</html>
```

Le doctype est en minuscule, l'attribut lang de la balise `html` sert à indiquer aux systèmes de synthèse vocale la prononciation à utiliser ou aux moteurs de recherche la langue dans lequel la page est rédigée et le `meta charset` indique le jeu de caractères.

Si l'on désire utiliser des fonctionnalités comme la balise <video>, les canvas et autres, il est nécessaire de corectement structurer le document en HTML 5.

A noter que le HTML 5 contrairement au X-HTML est permissif et permet l'emploi de balises non fermées telles que `<br>`.

SCRIPT

Pour spécifier un script Javascript, il faut utiliser la forme suivante :

```
<script type="text/javascript" src="fichier.js"></script>
<script  type="text/javascript">code_javascript();</script>
```

Il est nécessaire de placer la balise fermante </script> telle qu'elle, sans espace ou autres, car certains navigateurs recherchent cette séquence de 9 caractères pour détecter la fin du script et considéreront que les caractères <, > et & perdent leur signification spéciale dans le script.

Comme certains navigateurs conservent le caractère spécial des caractères <, > et & dans les scripts, on peut être tenté d'enfermer le script dans un CDATA... Sauf que si on désire que le script fonctionne dans tous les cas, il faut en arriver à utiliser une syntaxe particulièrement tortueuse :

```
<!--//--><![CDATA[//><!--
code_javascript();
//--><!]]>
```

Au final, il est plus simple de placer le script dans un fichier extérieur et de n'utiliser de script dans le document que pour des tâches simples telles que

```
window.onload=init;

window.onload=function() {
  var ctrl=new Controller();
};
```

Pour appeler une fonction javascript à partir d'un élément HTML, il existe deux moyens : soit un gestionnaire d'événement

```
<span onclick="fonction(paramètres)">click</span>
```

soit une URL

```
<a href="javascript:fonction(paramètres)">click</a>
```

A noter que le gestionnaire d'événements peut être associé à un objet ou autre directement à partir du Javascript. Cependant, on essaiera dans la mesure du possible de mettre les gestionnaires d'événements en place à partir du Javascript.

La balise script possède maintenant un attribut defer qui permet de retarder l'exécution du script jusqu'à la fin du téléchargement du document.

```
<script src="scripts.js" defer> </script>
```

II. Complements de CSS

IMAGES

Lorsque l'on place une image dans une page web elle est par défaut en mode inline. Dans ce mode, le navigateur prévoit la place pour les ascents (parties des lettres qui débordent vers le haut comme dans b, d, f, h, k, l et t) et les descents (parties des lettres qui débordent vers le bas comme dans g, j,p et q).

Ces ascents et descents ont pour effet d'ajouter un espace en haut et en bas de l'image. Pour l'éviter, il existe plusieurs méthodes (selon la situation) :

Soit on passe l'image en mode block, mais si on a plusieurs images, elles se trouveront en dessous l'une de l'autre (à moins qu'on n'utilise en plus float)

```
img { display: block; }
```

Soit on utilise un alignement vertical (il faut alors faire attention en cas de présence de texte à côté de l'image)

```
img { vertical-align: middle; }
```

On peut aussi utiliser l'image en image de fond d'une division de taille fixe

```
div {
  background: url('img/xxx.jpg') no-repeat;
  height: 123px;
  width:  42px;
}
```

PRIORITE DES REGLES

Lorsque plusieurs règles CSS sont en conflit sur un élément (règles héritées et plusieurs définitions qui conviennent), le navigateur se basera sur la précision du sélecteur pour déterminer la valeur conservée.

Pour calculer la spécificité d'une règle, on détermine les valeurs (a,b,c,d) comme suit :

Pour chaque élément ou pseudo-élément précisé, on augmente d de 1

Pour chaque classe, pseudo-classe ou attribut précisé, on augmente c de 1

Pour chaque id précisé, on augmente b de 1

Pour un style déclaré dans la balise de l'élément, on augmente a de 1

Par exemple, si on a

```
#main div div div.active
```

La spécificité sera **(0, 1** (#main)**, 1** (.active)**, 3** (3 div)**).** Pour comparer deux règles, on compare les valeurs dans l'ordre, d'abord a, puis b, puis c et enfin d. Dès qu'une niveau a une valeur supérieure dans une règle, on s'arrête.

Les `:first-line`, `:after`,... sont des pseudo-éléments et comptent donc comme des éléments.

Les `:hover`, `:link`, `:focus` sont des pseudo-classes et comptent comme des classes.

Une condition comme `[id=xyz]`, même si elle teste l'ID, est une déclaration de type attribut et est prise en compte comme telle.

Barre fluide / division fluide

Lorsque l'on désire utiliser un même jeu d'images pour définir une barre de séparation de taille variable ou une bordure de division de taille variable (par exemple pour donner un aspect de fenêtre), on parle de style fluide (par opposition au style taille fixe).

Barre fluide

Pour une barre fluide, on utilise deux images, une partie gauche qui reprend l'extrémité gauche et la zone centrale et une partie droite qui ne reprendra que l'extrémité droite. La taille maximale de la barre fluide dépendra de la taille de l'image gauche.

La taille de la barre fluide doit être précisée dans la largeur de la division extérieure. La hauteur de l'image doit être précisée dans la division interne (ce qui forcera la division externe à cette hauteur).

```
.bar {
     background: url('img/barleft.png') no-repeat 0% 0%;
     width: 400px;
     }

.bar div {
     background: url('img/barright.png') no-repeat 100% 0%;
     height:12px;
}
```

Le code HTML sera extrèmement simple :

```
     <div class="bar"><div></div></div>
```

Division fluide

Le même principe sera utilisé pour une division de taille variable. Cela permet de faire des effets de relief, de coins arrondis, ...

```
.popup {
    background: url('img/popuptl.jpg') no-repeat 0 0;
    margin: 0px;
    padding: 0px;
}
.popup div {
    background: url('img/popuptr.jpg') no-repeat 100% 0;
    margin: 0px;
    padding: 0px;
}
.popup div div {
    background: url('img/popupbl.jpg') no-repeat 0 100%;
    margin: 0px;
    padding: 0px;
}
.popup div div div {
    background: url('img/popupbr.jpg') no-repeat 100% 100%;
    color: #000000;
    padding: 30px;
    margin: 0px;
}
```

Ici, les images seront (dans l'ordre), le coin supérieur gauche+les bordures supérieures et gauche+le centre, le coin supérieur droit et la bordure droite, le coin inférieur gauche et le bord inférieur, le coin inférieur droit.

L'HTML utilisé contiendra 4 divisions imbriquées. Le padding de la 4ème division indiquera la taille de la zone de l'image qui correspond à la bordure (et donc, où le texte ne devra pas apparaître).

A noter que l'on ne peut pas utiliser d'images transparentes (les morceaux successifs doivent cacher les morceaux précédents).

Si on désire utiliser des divisions supplémentaires à l'intérieur de cette fenêtre, il sera nécessaire de la préciser au niveau CSS afin d'éviter que l'image de fond et le padding de la 4ème division ne la contaminent.

Positionnement absolu

L'attribut CSS `position: absolute;` permet d'indiquer que la position d'un élément est fixe par rapport à un point d'ancrage. Ce mécanisme est un peu le contre-pied des layouts fluides mais reste nécessaire pour certaines tâches.

La position de l'élement sera généralement définie à l'aide des attributs `top:` et `left:` (mais on peut également recourir à `bottom:` et `right:`)

Le point d'ancrage est fixé sur le premier élément parent qui contienne un attribut `position:` quel qu'il soit. A cet effet, on utilisera souvent une division aux dimensions fixées qui pourra elle même être positionnée de façon absolue ou au contraire utiliser `position: relative;` qui correspond au positionnement normal dans le flux.

Lorsque l'on place des éléments d'une telle manière, il peut arriver qu'ils se trouvent en totalité ou en partie hors de la division parent servant de référence. Pour éviter que les parties hors de la division parent ne soient affichées, on précisera sur cette dernière l'attribut `overflow: hidden;`.

Si aucune division parent ne précise l'attribut `position:`, le positionnement se fera par rapport à la balise `<body>`.

III. Compléments de Javascript

OBJET Math

Une série de fonctions mathématiques avancées se trouvent dans l'objet Math. On y trouve entre autres :

π	Math.PI	retourne la valeur de π
$\lvert x \rvert$	Math.abs(x)	Retourne la valeur absolue de x
⌈x⌉	Math.ceil(x)	Retourne le plus petit entier supérieur ou égal à x
⌊x⌋	Math.floor(x)	Retourne le plus grand entier inférieur ou égal à x
sin(x)	Math.sin(x)	Retourne le sinus de x (exprimé en radians)
cos(x)	Math.cos(x)	Retourne le cosinus de x (exprimé en radians)
tan(x)	Math.tan(x)	Retourne la tangente de x (exprimé en radians)
x^y	Math.pow(x,y)	Retourne la valeur de x exposant y
Random	Math.random()	Retourne un nombre aléatoire compris dans l'intervalle [0,1[

TABLEAUX

Pour créer un nouveau tableau, on utilisera

```
var a=new Array();
var b=new Array("a","b","c");
var c=["x",1,"y",2];
```

La seconde forme est à éviter car dans le cas où il n'y aurait qu'un paramètre, il sera considéré non comme un tableau de un élément mais un tableau dont le nombre d'éléments est la valeur du paramètre.

Pour accéder à un élément du tableau, on utilisera

```
a[2]="abc";
b[0]=c[2];
```

Tout comme les variables, les cellules d'un tableau peuvent contenir n'importe quel type d'entité : nombre, chaîne de caractères, fonction, objet, ...

Le premier élément d'un tableau est l'élément 0. Si on essaye d'écrire un élément après le dernier élément du tableau, la taille est automatiquement adaptée.

Pour déterminer la taille d'un tableau, il suffit de regarder sa propriété *length* :

```
var l=a.length;
```

On peut transformer un tableau en chaîne ou une chaîne en tableau à l'aide de *join()* et de *split()*.

```
var a=new Array(1,2,3);
var s=a.join(":"); // s vaudra "1:2:3"
var a2=s.split(":"); // a2 vaudra [1,2,3]
```

Pour envoyer un tableau lors d'une requête AJAX, on envoie les éléments un après l'autre dans des variables de la forme nom[]

```
param=param+"&nom[]="+encodeURIComponent(tbl[0]) ;
param=param+"&nom[]="+encodeURIComponent(tbl[1]) ;
 ...
```

Pour créer un tableau à deux dimensions, il faut créer un tableau de tableau :

```
var a=new Array();
for(i=0;i<n;i++) {
  a[i]=new Array();
}
```

On accédera aux éléments très simplement :

```
a[2][3]=123;
b=a[0][4];
```

COMMON .JS

La bibliothèque de fonction common.js fournie avec le cours contient une documentation de chacune de ses fonctions. On y trouve les fonctions suivantes :

$(elem)	Cette fonction permet de récupérer un élément HTML dans l'arbre DOM. Si on lui passe une chaîne de caractère, elle recherchera l'élément ayant cette chaîne comme ID. Sinon, elle retourne ce qu'on lui a passé en paramètre. Cela permet de réaliser des fonctions qui agissent sur un élément mais acceptent que l'on passe indifféremment cet élément directement ou par son id
getChildrenByTagName(elem,tag)	Cette fonction permet de trouver les éléments qui sont des enfants directs de l'élément fourni en premier paramètre et dont la balise est la chaîne passée en second élément. Le nom de la balise doit être fourni en minuscules et la fonction forcera en minuscules les noms des balises trouvées. Cela permet d'éviter des problèmes avec certains navigateurs mais doit être pris en compte si on utilise cette fonction pour récupérer des éléments dans un arbre XML.
createElement(el)	Cette fonction est généralement un synonyme de document.createElement. En X-HTML, elle s'assurera que le « namespace » soit correctement défini.
getvalfromxml(xml,tag)	Retourne le texte contenu dans la première balise de type tag trouvée dans l'objet xml fourni. Si la balise n'existe pas ou ne contient pas de texte, retourne une chaîne vide.
hide(elem)	Cache une division (ou un autre élément de type bloc).
show(elem)	Affiche une division (ou un autre élément de type bloc).
toggle(elem)	Affiche ou cache une division selon qu'elle était cachée ou visible.

selectclear(elem)	Vide un <select> de toutes les <option> qu'il contient
selectappend(elem,text,value,sel)	Ajoute une <option> à un <select>. Le texte affiché pour cette option sera text et la valeur associée sera value. Si sel est vrai, l'option sera sélectionnée par défaut.
selectgetval(elem)	retourne la valeur de l'<option> sélectionnée dans le <select> passé en argument (value)
selectgetname(elem)	retourne le texte affiché pour l'<option> sélectionnée dans le select passé en argument (text)
addevent(elem,event,handler)	Associe la fonction handler (nom de fonction ou fonction anonyme) à l'événement event (sans le "on" au début) de l'élément précisé. Cette fonction permet d'associer plusieurs fonctions à un même événement d'un même objet. (les événements déjà associés ne sont pas supprimés) Cette fonction retourne le gestionnaire d'événement associé nécessaire pour un removeevent(). Le gestionnaire d'événement sera toujours appelé avec une structure event (même sur MSIE) et la propriété target contiendra toujours l'élément associé au déclenchement.
removeevent(elem,event,handler)	Permet de retirer un gestionnaire d'événement associé précédemment à l'aide de addevent. La valeur handler devra contenir la valeur retournée lors de l'appel à addevent.
isanumber(text)	Retourne true si le paramètre est une valeur numérique correcte
checkEmail(text)	Retourne true si le paramètre est une adresse E-Mail valide.
css.getElementByClass(base,class,tag)	Parcours les éléments situés sous l'élément base (inclus) et retourne ceux qui ont le type décrit par tag et la classe CSS class sous la forme d'un tableau d'éléments
css.hasClass(el,class)	Retourne true si l'élément fourni utilise la classe CSS indiquée.
css.addClass(el,class)	Ajoute la classe indiquée à l'élément fourni en paramètre
css.removeClass(el,class)	Retire la classe indiquée de l'élément fourni en paramètre

Objets Javascript

Le Javascript utilise le principe de la programmation objet par prototype. Un objet est un container qui associe à des propriétés des valeurs. Ces valeurs peuvent être d'autres objets, des valeurs numériques, des chaînes, des tableaux ou des fonctions.

Lorsque l'on instancie un objet, le Javascript commence par créer un objet vide. Ensuite, il copie dans ce dernier le contenu du prototype de l'objet créé pour finalement appeler la fonction

« constructeur » dont le nom est le même que celui de l'objet créé. Ce constructeur doit retourner le pointeur `this` (qui pointe vers l'objet nouvellement créé)

Toutes les propriétés de cet objet peuvent être modifiées après leur création, y compris celles qui contiennent des fonctions. De même, on peut à tout moment ajouter de nouvelles propriétés, simplement en y sauvant un contenu.

Contrairement à la programmation objet par classe, la structure d'un objet (les propriétés présentes) peut changer tout au long de l'existence de ce dernier.

```
function Test(a) {
     this.a = a;
     this.b = 5;
     this.c = null;
     this.d = function(v) { return v*v;};
     return this;
}

Test.prototype.f1 = function (x) {
     return this.a * x;
};

var o=new Test(10);
var res=Test.f1(3);
```

Le mécanisme de closure sera souvent utilisé pour des gestionnaires d'événements afin de corriger la valeur de this pour qu'elle pointe sur l'objet créé et non celui qui a déclenché l'événement

```
Test.prototype.addevents = function () {
     var oThis=this; // oThis sera utilisé pour la closure
     this.c.onclick = function(e) { oThis.myonclick(e); };
};
```

Local Storage

Introduction

Si le mécanisme des cookies permet de sauver des informations de petites taille au niveau des navigateurs, ce mécanisme est limité à un volume de quelques Ko et la totalité des cookies relatifs à un site est transmise à chaque requête.

Dans un contexte AJAX, il serait intéressant de pouvoir sauver des volumes de données plus conséquents et de disposer d'une interface simple pour y accéder. C'est ce que permettent les objets *localStorage* et *sesssionStorage*.

Utilisation

L'objet *sessionStorage* permet de sauver des données relatives à un site dans une fenêtre. Plusieurs fenêtres ouvertes sur le même site auront des objets sessionStorage disctincts. Le contenu des objets sessionStorage est perdu une fois le navigateur fermé (comme le sont les cookies de session).

L'objet *localStorage* permet une sauvegarde persistante de données au niveau du navigateur. Il permet de sauver des données relatives à un site et disponible à partir de n'importe qu'elle fenêtre ouverte sur le site.

Les objets sessionStorage et localStorage disposent des mêmes méthodes :

.length	Nombre d'éléments stockés dans l'objet
.key(idx)	Retourne la idx^{ème} "clé"
.getItem(clé)	Récupère l'élément associé à la clé passée en paramètre
.setItem(clé,valeur)	Change la valeur associée à une clé (crée la clé si nécessaire)
.removeItem(clé)	Efface l'élément associé à la clé passée en paramètre
.clear()	Efface tous les éléments présents dans le stockage (pour le site)

On peut également utiliser la notation tableau[20] pour récupérer et modifier la valeur d'un élément sauvé dans un *localStorage* ou un *sessionStorage*.

```
var i,s;

localStorage.setItem("abc",15);
localStorage["ghi"] = "Don't Panic !";

s="Valeur abc : "+localStorage.getItem("abc");
s=s+" Valeur ghi : "+localStorage["ghi"];

for(i=0;i<localStorage.length;i++) {
   s=s+" Clé["+i+"] : "+localStorage.key(i);
   s=s+" valeur : "+localStorage.getItem(localStorage.key(i));
}

localStorage.removeItem("ghi");
localStorage.clear();
```

Références

Définition des Storage sur le site du W3C : http://www.w3.org/TR/webstorage/

Dive Into HTML5 propose une explication moins "stricte": http://diveintohtml5.info/storage.html

20 C'est normalement le cas pour tous les objets Javascript qui proposent les fonctions getItem/setItem

IV. Les Canvas

Introduction

Les canvas, ajoutés au HTML 5, sont des éléments HTML dont le contenu est complètement défini par le Javascript.

On peut comparer un canvas à une feuille blanche sur laquelle il est possible de tracer graphiques, images et texte.

Vu que le contenu affiché dans un canvas est totalement généré par le Javascript, ils sont principalement (si pas exclusivement) utilisé dans des applications de type AJAX.

Pour une application de type mathématique, les canvas permettent d'afficher différents graphiques à partir des données calculées ou téléchargées par le javascript.

Pour une application plus graphique, ils permettent d'implémenter des fonctions de traitement de l'image (recadrage par exemple) ou des affichages graphiques complexes

Pour une application ludique, ils permettent de réaliser un jeu complètement graphique sans utiliser d'astuces telles que des layers multiples, mais également de profiter des accélérations matérielles de la carte graphique (tant en 2D qu'en 3D).

Une fois un canvas placé sur une page web, il sera nécessaire de décider si on compte l'utiliser en 2D (tracés de formes géométriques par exemple) ou en 3D (en utilisant WebGL, qui permet une accélération 3D matérielle). Dans le cadre de ce cours, nous nous limiterons aux canvas 2D.

Elément Canvas

Pour placer un canvas dans un page HTML, il est nécessaire de fixer ses dimensions et de lui donner un ID. Les dimensions permettront au moteur HTML de réserver l'espace nécessaire au canvas et l'ID permettra au Javascript de récupérer le canvas pour le contrôler.

Il va de soi que si l'élément est créé directement à partir du javascript, on peut se passer de l'ID et conserver l'élément dans une variable.

```
<canvas id="canvasID" height="300" width="400"></canvas>
```

Il est possible d'utiliser des styles CSS pour modifier l'apparence du canvas, comme ajouter une bordure.

Chaque fois qu'une des dimensions du canvas est modifiée, la totalité du canvas est effacée et il est nécessaire de le retracer complètement. Toutes les informations liées au canvas sont effacées.

Par contre, changer la dimension au travers des CSS provoquera une mise à l'échelle de l'image au niveau de l'affichage. Par défaut, les dimensions CSS seront égales aux dimensions précisées dans l'élément canvas.

Propriétés de l'élément canvas

L'élément canvas contient les propriétés (et fonctions) suivantes :

- width, height : dimensions du canvas (pas les dimensions CSS). Les changer remet le canvas à zé&ro (même si on remet l'ancienne valeur)

- getContext("2d") : retourne un contexte 2D pour le canvas. Tous les accès pour dessiner le canvas se fait au travers du contexte 2D.[21]

- toDataURL() : retourne une Data URI contenant l'image affichée dans le canvas. On peut préciser le type MIME demandé (toDataURL("image/png") par exemple). Le format PNG est celui par défaut et le seul exigé par le standard. Pour certains types, il sera nécessaire de fournir des paramètres supplémentaires (par exemple, toDataURL("text/jpeg",0.9) où le second paramètre permet de préciser la qualité de l'image)

La valeur retournée par toDataURL() peut être utilisée comme source pour une balise <img> par exemple, ou être envoyée au niveau du réseau (POST).

La quasi totalité des actions sur le canvas se fera au travers de l'objet contexte 2D qui sera représenté par la suite par `ctx`.

Notion de Path

Une fois que l'on a obtenu le contexte 2D, on pourra l'utiliser pour tracer des "chemins". Ces chemins sont des successions de déplacements, de lignes, d'arcs de cercles et autres courbes.

Une fois le path terminé, on peut demander de le tracer (`ctx.stroke()`) ou de le "remplir" (`ctx.fill()`). Différents paramètre du contexte permettront de modifier la façon dont des tracés seront effectués

Pour commencer un tracé, on utilisera la commande

```
ctx.beginPath() ;
```

Cette instruction effacera le chemin précédent (mais pas les tracés effectués à l'aide de ce dernier) et commencera un nouveau chemin vide.

On pourra alors utiliser les commandes

`ctx.moveTo(x,y);`	Déplace le curseur à la position (x,y). Permet de commencer un autre segment pour le chemin en cours
`ctx.lineTo(to_x,to_y);`	Ajoute un segment du point courant au point (to_x,to_y). Le nouveau point courant devient (to_x,to_y).
`ctx.rect(x,y,width,height);`	Permet de tracer un rectangle horizontal dont on fourni le coin supérieur gauche (x,y) et les dimensions.
`ctx.arc(x,y,r,startAngle,endAngle, counterclockwise);`	Ajoute un arc de cercle au chemin courant. Le cercle aura (x,y) comme centre et r comme rayon. Les angles de départ et de fin sont mesurés en radians (360°=2*Math.PI). Le paramètre counterclockwise permet de préciser le sens de tracé. Le nouveau point courant est la fin de l'arc.
`ctx.arcTo(x1,y1,x2,y2,r);`	Trace un arc de cercle qui commence au point

21 On peut utiliser .getContext("webgl") pour obtenir un contexte 3D en WebGL

	courant, termine au point (x2,y2) , présent sur un cercle de rayon r. Les extrémités sont tangentes aux droites passant par les points de départ/arrivée et par le point (x1,y1). Le point courant sera déplacé en (x2,y2)
`ctx.quadraticCurveTo(cpx,cpy,x,y);`	Trace une courbe de bézier quadratique qui commence au point courant et termine au point (x,y). Le point de contrôle de la courbe est (cpx,cpy)
`cpx.bezierCurveTo(cp1x,cp1y,cp2x,cp2y, x,y);`	Trace une courbe de bézier qui commence au point courant et termine au point (x,y). Les deux points de contrôle sont (cp1x,cp1y) et (cp2x,cp2y).

Si on désire créer un chemin fermé, on utilisera la commandes

```
ctx.closePath();
```

Il est préférable de terminer le chemin ainsi plutôt que de terminer par un segment vers le point de départ du chemin. En effet, cette seconde méthode ne forme pas un chemin fermé mais un chemin où les points de départ et de fin sont à la même position, ce qui peut impacter négativement le rendu.

Tracé de chemin

La commande

```
ctx.stroke();
```

permet de tracer les segments du chemin courant. Plusieurs paramètres permettent de modifier l'apparence du tracé.

```
ctx.strokeStyle=couleur/gradient/pattern;
```

Permet de définir la couleur du tracé. La couleur peut être fournie soit par le nom CSS ("red"), soit par le code couleur hexadécimal. On peut également utiliser des gradiants (dégradés) ou patterns (motifs).

`ctx.lineCap="butt|round|square";`	Permet de définir le style d'extrémité de ligne
`ctx.lineJoin="bevel|round|miter";`	Permet de définir le style d'angle entre deux traits successifs d'un tracé
`ctx.lineWidth=largeur;`	Permet de définir la largeur de tracé
`ctx.miterLimit=taille;`	Dans le cas où deux segments se suivent, indique la taille maximum entre le point le plus à l'intérieur et celui le plus à l'extérieur. Permet de tronquer le joint entre les deux lignes.

Remplissage d'un chemin

La commande

```
ctx.fill();
```

permet de remplir la forme définie par le chemin. Si le chemin n'est pas fermé, on le considérera comme tel pour le remplissage (mais cela n'affectera pas le tracé).

```
ctx.fillStyle=couleur/gradient/pattern;
```

Permet de définir le style de remplissage (de manière semblable au strokeStyle).

Pour déterminer si une zone est à l'intérieur ou à l'extérieur (et donc, doit être remplie ou pas), on compte le nombre de segments croisés à partir du point concerné. Chaque segment "tournant" dans le sens horlogique augmente la valeur de 1, chaque segment "tournant" dans le sens contraire diminuera cette valeur de 1. Si la valeur vaut 0, le point est considéré à l'extérieur.

Si on prend deux carrés imbriqués l'un dans l'autre, si le tracé se fait dans le même sens pour les deux carrés, l'intérieur du carré situé dans l'autre sera considéré comme "à l'intérieur". Par contre, si les deux carrés sont tracés dans des sens opposés, l'intérieur du carré interne sera considéré comme à l'extérieur et ne sera pas rempli.

Ombrage

Les propriétés ctx.shadowColor, ctx.shadowBlur, ctx.shadowOffsetX et ctx.shadow.offsetY permettent de définir une ombre qui devra être tracée en dessous des tracés effectués sur un chemin.

La couleur sera précisée par code couleur (#rrggbb) ou par nom de couleur, l'intensité de l'ombre à l'aide du paramètre blur et les deux derniers paramètres permettent de décaller l'ombre générée.

Tracés hors "path"

Il est possible de tracer du texte ou des rectangles sans passer par un chemin.

Les paramètres de remplissage, tracé et ombrage sont également actifs pour ce genre de tracés.

`ctx.strokeRect(x,y,width,height);`	Permet de tracer un rectangle horizontal dont on fourni le coin supérieur gauche (x,y) et les dimensions. Seul le tracé extérieur est réalisé
`ctx.fillRect(x,y,width,height);`	Permet de tracer un rectangle horizontal dont on fourni le coin supérieur gauche (x,y) et les dimensions. Seul le remplissage intérieur est réalisé
`ctx.clearRect(x,y,width,height);`	Permet de vider un rectangle horizontal dont on fourni le coin supérieur gauche (x,y) et les dimensions.
`ctx.fillText(text,x,y,maxWidth);`	Trace un texte aux coordonnées x,y. Le texte prendra maximum maxWidth pixels de large. Le texte sera rempli
`ctx.strokeText(text,x,y,maxWidth);`	Trace un texte aux coordonnées x,y. Le texte

	prendra maximum maxWidth pixels de large. Seul le tracé extérieur sera effectué
`ctx.font=cssfont`	Défini la fonte utilisée pour les tracés de texte. La fonte est définie de la même façon que dans les CSS (par exemple, "40px Courrier")
`ctx.textAlign="left/center/right"`	Indique le mode de centrage du texte par rapport au point d'ancrage spécifié

BitBlit

La commande

```
ctx.drawImage();
```

permet de copier tout ou partie d'une image dans un canvas. A noter qu'il est important que l'image soit déjà chargée (utiliser l'événement onload de l'image en cas de doute).

Selon les paramètres, l'image sera simplement copiée à une position donnée, redimensionnée ou un morceau de l'image sera utilisé.

`ctx.drawImage(image,x,y)`	Copie l'image à la position x,y
`ctx.drawImage(image,x,y,w,h)`	Copie l'image à la position x,y. L'image sera redimensionnée aux dimensions w,h
`ctx.drawImage(image,sx,sy,sw,sh,x,y,w,h)`	Copie le morceau rectangulaire de l'image dont le coin supérieur gauche est (sx,sy) et la taille (sw,sh) à la position (x,y) dans le canvas, en la redimentionnant à la taille (w,h)

Les commandes `createImageData(w,h)`, `getImageData(x,y,w,h)` et `putImageData(img,x,y)` permettent de créer une image vide ou de récupérer une image du canvas sous la forme d'un objet qui contient les propriétés `height`, `width` et `data`, un tableau de 4*w*h nombres. Chaque pixel est représenté par 4 nombres consécutifs : rouge, vert, bleu, transparence.

Coordonnées

Par défaut, le point (0,0) se trouve au niveau du coin supérieur gauche du canvas et le coin opposé correspond à la taille de l'image.

Les fonctions `scale(sx,sy)`, `rotate(angle)`, `translate(dx,dy)` permettent de modifier la position des différentes coordonnées. En interne, cela est effectué par une matrice de transformation de 3x3 (la dernière ligne contient les valeurs 0, 0 et 1) qui sera appliquée sur le vecteur des coordonnées augmenté de la valeur 1 : (x,y,1) (cette valeur 1 finale permet de gérer les translations par la matrice de transformation et la dernière ligne de la matrice de transformation permet de conserver cette valeur).

La fonction `transform(a11,a12,a13,a21,a22,a23)` permet d'effectuer une transformation quelconque définie par sa matrice de transformation (ce qui permet des transformations plus complexes) et la fonction

`setTransform(a11,a12,a13,a21,a22,a23)` permet de fixer la valeur de la matrice de transformation.

Autres

Les commandes `ctx.save()` et `ctx.restore()` permettent de sauver et récupérer la configuration du canvas : chemin en cours, transformation, …

La ressource `ctx.globalCompositeOperation` contient une chaîne qui indique comment combiner les nouveaux tracés avec les anciens. On peut découvrir un bon descriptif de ce paramètres sur le site de la W3School : http://www.w3schools.com/tags/canvas_globalcompositeoperation.asp

La ressource `ctx.globalAlpha` contient le niveau d'alpha pour les tracés qui suivent (entre 0, transparent, et 1, opaque).

V. Compléments de PHP

TABLEAUX

Tout comme en Javascript il est possible de transformer une chaîne en tableau à l'aide de split et un tableau en chaîne à l'aide de join, on peut faire ces opérations en PHP à l'aide des fonctions implode et explode.

```
$array=explode("|",$str);    // Décompose la chaîne en tableau
$str=implode("|",$array);    // Crée une chaîne à partir d'un tableau et
                             // d'un séparateur
```

On peut aussi créer un tableau élément par élément aisément :

```
$a=array(); // un tableau vide
$a[]="valeur 1"; // Ajoute un élément à la fin du tableau
$a[]="valeur 2"; // Ajoute un deuxième élément en fin de tableau
```

Il est possible de passer tous les éléments du tableau en revue (qu'il s'agisse d'un tableau normal ou associatif)

```
foreach($a as $i => $v) {
        [...]
}
```

Pour chaque valeur présente dans *$a*, la variable *$i* contiendra l'index et la variable *$v* la valeur.

```
($v=$a[$i])
```

Pour réaliser un tableau de tableaux (ou tableau à plusieurs dimensions), il faut penser à initialiser chaque niveau intermédiaire

```
$tbl=array() ;
$tbl[0]=array() ;
$tbl[1]=array() ;
$tbl[0]["abc"]=... ;
$tbl[0]["def"]=... ;
$tbl[1]["xyz"]=... ;
```

Lorsque l'on fait un foreach, il s'effectue sur le premier niveau non précisé

```
foreach($tbl as $i) ==> 0, 1, ...
foreach($tbl[0] as $i) ==> "abc", "def", ...
```

SESSIONS

Un des problèmes qui se pose lorsque l'on crée une application web est celui de conserver des variables entres deux visites de page.

En effet, chaque requête effectuée par le navigateur est indépendante des autres.

En PHP, on dispose du mécanisme des sessions qui permet de conserver des variables entre deux accès de pages. Ces sessions sont liées au navigateur qui fait les requêtes. Si on utilise deux navigateurs différents pour accéder au même site, chacun d'entre eux aura une session différente.

De même, lorsque l'on quitte un navigateur, les sessions sont également perdues.

Pour utiliser les sessions, il faut utiliser la commande

```
session_start();
```

En début de page PHP (en même temps que les *header()* et autres). On dispose alors d'un tableau associatif *$_SESSION[]* qui sera conservé d'une page à l'autre. Il est nécessaire d'utiliser *session_start()* dans TOUTES les pages qui doivent accéder à *$_SESSION[],* que ce soit pour y lire ou pour y écrire

```
session_start();
$_SESSION["msg"]=0;
$usr=$_SESSION["uid"];
```

Ces lignes permettent de sauver 0 dans la variable de session *msg* et de lire dans *$usr* la variable de session *uid*.

CHAINES DE CARACTERES UTF-8

Si l'UTF-8 permet de représenter tous les caractères (alphabet latin, grec, cyrillique, japonnais, thailandais,...),cela nécessite plus que les 8 bits présents dans un octet. Certains caractères seront encodés sur plusieurs octets.

La méthode la plus simple est de préciser l'encodage utilisé en début de programme à l'aide de

```
mb_internal_encoding("UTF-8");
```

et de remplacer les appels aux fonctions de traitement de chaîne standard (qui considèrent que 1 caractère = 1 octet) par leur équivalent mb_string (qui utilisent les mêmes paramètres)

Chaîne ISO-8859-1	Chaîne UTF-8	
explode(*séparateur*,*chaine*)	mb_split	découpe la *chaîne* à l'aide du *séparateur* et place les résultats dans un tableau[22]
implode(*séparateur*,*tableau*)	implode	Crée une chaîne composée des chaînes dans le tableau séparées par le séparateur.
strpos(*foin*,*aiguille*)	mb_strpos	Trouve la première occurrence de *aiguille* dans *foin* (deux chaînes de caractères). Retourne la position par rapport au début de la chaîne (le premier caractère est le caractère 0) Si la chaîne n'est pas trouvée, retourne FALSE (à tester avec 3 signes "=") `if (mb_strpos("abcdef","z")===FALSE) ...`
stripos(*aiguille*,*foin*)	mb_stripos	Semblable à strpos mais ne fait pas de différence entre majuscules et minuscules
strlen(*chaine*)	mb_strlen	Retourne le nombre de caractères
strtoupper(*chaîne*)	mb_strtoupper	Met la chaîne en majuscules[23]
strtolower(*chaîne*)	mb_strtolower	Met la chaîne en minuscules

22 Pour être exact, mb_split correspond à preg_split et découpe selon une "regexp" (expression régulière). Mais si on utilise une chaîne contenant un caractère à l'exception de +, *, $, ^, |, . (point), ?, [,], (,), { ou }, cette fonction se comporte comme explode. Pour les caractères précités, il faudra utiliser \ suivi du caractère ('\+' ou "\\+").

23 Dans le cas de mb_strtoupper, cela inclus la mise en majuscule des caractères accentués, grecs, cyrilliques,...

substr(*chaine*,*debut*[,*taille*])	mb_substr	Découpe un morceau de la chaîne de caractères *chaîne*, commençant par le caractère à la position *debut* (0 = le premier caractère) et contenant *taille* caractères. Si *taille* n'est pas précisé (ou trop grand), la fonction prendra tous les caractères jusqu'à la fin de la chaîne.
str_repeat(*chaîne*,*nombre*)	str_repeat	répète la *chaîne* de caractères *nombre* fois.
str_replace(*chercher*,*remplacer*, *chaîne*)	str_replace	retourne une chaîne où les occurrences de la chaîne *chercher* sont remplacées par la chaîne *remplacer* dans *chaîne*

A noter que les comparaisons entre chaînes (== et !=) et la concaténation ("abc" **.** "def") sont correctes quel que soit l'encodage des caractères utilisés.

DATE/TIME

Les fonctions qui suivent demandent que le fuseau horaire par défaut aie été défini. Si ce n'est pas le cas, le PHP retournera un "warning".

```
date_default_timezone_set('Europe/Brussels");
```

Sous Unix, la méthode standard pour stocker les heures est le nombre de secondes depuis le 1er Janvier 1970 à 00:00:00. Cette méthode a l'avantage de ne demander qu'un entier 32 bits, facile à manipuler (type int en C, INTEGER en SQL, ...). Cette valeur est souvent appelée timestamp.

time()	retourne l'heure courante (en secondes depuis EPOCH)
date(format)	synonyme de date(format,time())
date("d",ts)	retourne le jour (de 01 à 31) du timestamp fourni
date("m",ts)	retourne le mois (de 01 à 12) du timestamp
date("y",ts)	retourne l'année sur 2 chiffres (2013 -> 13) du timestamp
date("Y",ts)	retourne l'année sur 4 chiffres du timestamp
date("d/m/y",ts)	retourne la date du timestamp de la forme 23/04/13 pour le 23 avril 2013
date("H",ts)	retourne l'heure (de 00 à 23)
date ("i",ts)	retourne les minutes (de 00 à 59)
date("s",rs)	retourne les secondes (de 00 à 59)
date("H:i:s",ts)	retourne l'heure associée au timestamp de la forme 13:45:32
mktime(heure,minute,seconde,mois,jour,annee)	retourne un timestamp pour le moment décrit.

INCLUDE

Dans un script PHP, il est possible d'inclure un autre fichier. Pour cela, on utilisera la fonction

include(nomfichier) ;

Il est fortement conseillé de nommer les include .inc et non .php.

En effet, lorsque, suite à un problème, le moteur PHP cesse de fonctionner, une personne qui essaye de charger une page .php verra son listing.

En déplaçant les mots de passe de base de donnée (et autres informations sensibles) dans un fichier .inc, il est possible en bloquant le chargement des fichiers .inc par le navigateur à l'aide du fichier .htaccess (ou de la configuration de apache) de garder ces informations secrètes.

Afin d'éviter les problèmes avec le mécanisme de session et la fonction header(), un fichier d'include doit commencer directement par la balise <?php (sans espace, saut de ligne ou autre avant) et il ne doit y avoir aucun caractère après la balise fermante?> (donc, pas de retour de chariot ou d'espace).

Quand le fichier include définit des variables (paramètres, chaînes de connexion, ...) ils seront déclarés comme variables globales. Pour y accéder à partir d'une fonction, il faudra, dans le corps de la fonction, préciser que ces variables sont globales par le mot clé "global" suivi des variables en question.

SERIALIZE

La fonction *serialize($variable)* permet de créer une chaîne de caractère qui contient toutes les informations de la variable passée en argument. On peut ainsi obtenir une chaîne à partir d'un entier, d'une chaîne ou d'un tableau.

La fonction *unserialize($chaîne)* permet de recréer l'objet initial à partir de la chaîne retournée par serialize.

A noter que le format de la chaîne sérialisée n'est pas défini et pourrait être changé à tout moment. Cette chaîne ne doit donc être utilisée qu'au moyen de la fonction *unserialize()*.

VI. Bases de données en PHP

PDO

Connexion

Pour créer une connexion sur une DB à l'aide de PDO, on utilise une des syntaxes :

```
$dbh = new PDO(connectstr);
$dbh = new PDO(connectstr,user,pass)
```

Le premier paramètre est la chaîne de connexion. Cette chaîne dépendra du moteur de base de données utilisé.

La seconde forme permet de fournir un nom d'utilisateur et un mot de passe pour la connexion.

La valeur retournée est la ressource de connexion de la base de données concernée (on peut effectuer des connexions vers plusieurs DB différentes dans le même script). En cas de problème, une valeur *null* est retournée.

Preparer la requête

Avant d'effectuer des requêtes, il faut les préparer. On peut disposer de plusieurs requêtes préparées en même temps et on peut réutiliser la même requete préparée plusieurs fois[24].

```
$stm = $dbh->prepare(requeteSQL);
```

Lorsque l'on prépare une requête, on remplace les valeurs par des ?. C'est ainsi que l'on transformera les requêtes

```
  SELECT nom,prenom FROM utilisateurs WHERE prenom='Albert'
INSERT INTO produit(nom,prix) VALUES('clé USB',15)
UPDATE score SET highscore=1000 WHERE idusr=15
```

en :

```
SELECT nom,prenom FROM utilisateurs WHERE prenom= ?
INSERT INTO produit(nom,prix) VALUES(?,?)
UPDATE score SET highscore=? WHERE idusr=?
```

Exécuter la requête

Pour exécuter une requête préparée, on utilisera la syntaxe suivante :

```
$stm->execute();
$stm->execute(array(valeur1,valeur2,...));
```

La seconde forme permet de fournir des valeurs qui remplaceront les points d'interrogation (?) dans l'ordre où ils apparaissent dans la requête SQL.

L'utilisation de requêtes préparées permet de ne pas devoir s'inquiéter de la présence de caractères spéciaux dans les valeurs transmises.

24 Cela permet de ne pas "perdre" plusieurs fois le temps nécessaire à la préparation de la requête

Lire les résultats

Une fois une requête exécutée, on peut lire les résultats de la manière suivante :

```
while ($row = $stm->fetch()) {
   ... $row["colonne1"] ...
   ... $row["colonne2"] ...
}
```

La boucle sera effectuée pour chaque ligne retournée par la requête. On peut aussi utiliser[25] $*row*[0], $*row*[1],... mais cette notation est moins lisible (sauf s'il n'y a qu'une valeur retournée).

Si on n'est sensé ne récupérer que 0 (pas trouvé) ou 1 (trouvé) ligne, on peut utiliser

```
if ($row=$stm->fetch()) {
   ... [trouvé] ...
}
```

SQLite

Le système SQLite est un moteur SQL intégré en natif dans PHP 5. Il utilise un fichier pour stocker les tables et les index. La chaîne de connexion PDO pour une DB SQLite est :

```
sqlite:nomfichier
```

Les principaux types de données sont

- INTEGER
- TEXT
- REAL

respectivement pour sauver des entiers, des chaînes de caractères et des nombres en virgule flottante. Les types tels que TINYINT, VARCHAR, ... sont convertis en interne vers un des trois types ci-dessus.

Il est possible de déclarer une colonne comme

INTEGER PRIMARY KEY AUTOINCREMENT

Cette colonne recevra automatiquement des valeurs qui se suivent. Il ne faut pas la préciser dans les INSERT et UPDATE.

L'utilitaire sqlite3.exe permet d'accéder à la DB à partir de la ligne de commande. On l'appellera avec le nom du fichier en paramètre.

Dans cet utilitaire, la commande `.help` permettra d'avoir une page d'aide. Pour exécuter une commande SQL, il faudra la terminer par un point-virgule(;). En l'absence du ";", on peut continuer la commande SQL sur la ligne suivante.

25 On utilisera par exemple cette notation lorsque l'on utilise un SELECT COUNT(*) FROM ...