PHP

Bertrand Estellon

Aix-Marseille Université

13 mars 2012

Rappels – HTML

Le **HTML** est un langage qui permet de décrire une page Web.

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
 <head>
  <title>
   Exemple
  </title>
 </head>
 <body>
  izgz hzegizihzi zihzg zeighze <a href="toto.html">toto</a>
  <b>aaaa</b>
 </body>
</html>
```

Rappels – CSS

Le **Cascading Style Sheets** (**CSS**) est un langage qui sert à décrire la présentation des documents HTML et XML.

```
body{
   overflow-y:scroll;
   background-color: #FFFFFF;
   white-space: nowrap;
   text-align:center;
}

.main {
      text-align:left;
      white-space: normal;
      display:inline-block;
      color: #000000;
      background-color: #FFFFFF;
      border: 0px;
}
```

Rappels – JavaScript

Aix*Marseille université

Le **JavaScript** est le langage principalement utilisé dans les pages Web :

```
function addPointToChart(time, id, value) {
  if (data.length <= id)
        data.push({ label : "objective "+(id+1), data : [] });
  if (data[id].data.length >= 10)
        data[id].data = data[id].data.slice(1);
  data[id].data.push([time, value]);
}

function analyseMessageForChart(msg) {
  var expression = /([0-9]*) sec.*obj = \((.*)\)/;
  var res = msg.match(expression);
  if (res!=null) {
        var time = parseInt(res[1]);
        var points = res[2].split(", ");
        for(var i=0;i<points.length;i++)
              addPointToChart(time,i,parseInt(points[i]));
        updateChart();
  }
}
```

Rappels – DOM

Le **Document Object Model** (**DOM**) est une recommandation du W3C qui décrit une interface permettant à des programmes d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML.

```
<script type="text/javascript">
   document.firstChild.firstChild.lastChild;
   document.firstChild.childNodes[0].lastChild;
   document.firstChild.childNodes[0].childNodes[1];
   var x = document.getElementsByTagName('P')[0].lastChild;
   document.getElementById('a')
           .firstChild.nodeValue='bold bit of text';
   var x = document.createElement('br');
   document.getElementById('e').appendChild(x);
   var x = document.createTextNode('exemple');
   document.getElementById('b').appendChild(x);
   ...
</script>
```

jQuery

jQuery est une bibliothèque JavaScript libre qui permet de simplifier la programmation en JavaScript (AJAX et interaction avec HTML).

Elle est disponible sous Licence MIT, GNU GPL.

"Installation" :

- Télécharger le code JavaScript sur le site de JQuery ;
- Inclure le code suivant dans l'en-tête de votre fichier HTML :

```
<script type="text/javascript" src="jquery-xxx.min.js">
</script>
```

Exemple d'utilisation :

```
<script type="text/javascript">
$(function() {
    $("a").click(function() { alert("Hello world!"); });
});
</script>
```

jQuery – Sélectionner des éléments

Aix*Marseille université

Sélectionner un élément par son identifiant :

```
$("#id_element").addClass("maClasseCss");
```

Sélectionner un élément par sa classe :

```
$(".classeCss").click(function() { alert("coucou"); });
```

Sélectionner des elements contenus dans un autre élément :

```
$("#list > li").addClass("blue"); ou
$("#list").find("li").addClass("blue");
```

Tout sélectionner :

```
$(*).addClass("blue");
```

Appliquer une fonction sur un ensemble d'éléments :

```
$("#list").find("li").each(function(i) {
      $(this).append( "Je suis le numero " + i);
});
```

jQuery – Modifier les attributs et les propriétés

Aix*Marseille université

Les attributs (valeurs initiales présentes dans le HTML) :

```
$("#id_element").attr("title","toto");
a = $("#id_element").attr("title");
$("#id_element").removeAttr("title");
```

Les propriétés (valeurs courantes dans le DOM) :

```
$("#id_element").prop("checked",true);
a = $("#id_element").prop("checked");
$("#id_element").removeProp("toto");
```

La valeur :

```
$("#id_element").val("coucou");
v = $("#id_element").val();
```

jQuery – Parcourir les éléments

Aix*Marseille université

Appliquer une fonction sur un ensemble d'éléments :

```
$("#list").find("li").each(function(i) {
      $(this).append( "Je suis le numero " + i);
});
```

Récupérer la liste des fils d'un élément :

```
$('#list').children().css('background-color', 'red');
```

Récupérer le premier élément :

```
$('#list').children().first().css('background-color', 'red');
```

Récupérer le dernier élément :

```
$('#list').children().last().css('background-color', 'red');
```

Filtrer les éléments :

```
$('#list').children().filter(function(i) {return i%3== 0;})
          .css('background-color', 'red');
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – Ajouter, supprimer des éléments

Aix*Marseille université

Modifier et récupérer le contenu des éléments :

```
$("#list").find("li").html("<b>toto</b>");
$("#list").find("li").text("<b>toto</b>");
c_html = $("#list").find("li").html();
c_text = $("#list").find("li").text();
```

Ajouter dans des éléments :

```
$("#list").find("li").append("toto");
```

Supprimer un ensemble d'éléments du DOM :

```
$("#list").find("li").remove();
```

Insérer avant et après des éléments :

```
$('#test').children().before("<li>avant chaque element</li>");
$('#test').children().after("<li>apres chaque element</li>");
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – CSS

Aix*Marseille université

Les classes *css* :

```
r = $("#id_element").hasClass("maClasseCss");
$("#id_element").removeClass("maClasseCss");
$("#id_element").toggleClass("maClasseCss");
```

Jouer avec le style :

```
$("#id_element").css('background-color', 'red');
color = $("#id_element").css('background-color');
```

Jouer avec la hauteur et la largeur :

```
$("#id_element").height(100);
h = $("#id_element").height();
$("#id_element").width(100);
w = $("#id_element").width();
```

Connaître la position d'un élément :

```
p = $("#id_element").offset(); (par rapport au document)
alert(p.left+" "+p.top);
p = $("#id_element").position(); (par rapport au parent)
alert(p.left+" "+p.top);
```

jQuery – Les événements

Aix*Marseille université

La souris :

```
$("#id_element").click(function() {  alert("click."); });
$("#id_element").mouseup(function(){$(this).append('up ');}).
                 .mousedown(function(){$(this).append('down ');});
```

Le clavier :

```
$('#input_text').keydown(function(event) {
   if (event.keyCode == '13') {
      alert("ok");
    }
});
```

Les changements :

```
$("#id_element").change(function() {  alert("change."); });
```

(Voir les autres fonctionnalités sur le site de jQuery)

jQuery – Les autres fonctionnalités

- Les effets et animations
- Des outils pour simplifier la programmation en JavaScript
- JQuery UI :
  - Librairie de widgets
  - Interaction (Drag & drop, tri, etc).
  - Thèmes
  - Effets

Ajax (Asynchronous JavaScript and XML)

Ajax est un acronyme pour Asynchronous JavaScript and XML.

Il est un ensemble de technologies libres couramment utilisées sur le Web :

- **HTML** (ou XHTML) pour la structure sémantique des informations ;
- **CSS** pour la présentation des informations ;
- **DOM** et **JavaScript** pour afficher et interagir dynamiquement avec l'information présentée ;
- l'objet **XMLHttpRequest** pour échanger les données de manière asynchrone avec le serveur Web ;
- **XML** (parfois les fichiers texte ou JSON) pour le format des données informatives.

XMLHttpRequest : Création

Fonction Javascript pour créer un objet XMLHttpRequest :

```
function createXhrObject()
{
    if (window.XMLHttpRequest)
        return new XMLHttpRequest();

    if (window.ActiveXObject) {
        var names = [
            "Msxml2.XMLHTTP.6.0",
            "Msxml2.XMLHTTP.3.0",
            "Msxml2.XMLHTTP",
            "Microsoft.XMLHTTP"
        ];
        for(var i in names) {
            try{ return new ActiveXObject(names[i]); }
            catch(e){}
        }
    }
    window.alert("pas de XMLHTTPRequest.");
    return null;
}
```

XMLHttpRequest : Les méthodes et propriétés

Aix*Marseille université

```
interface XMLHttpRequest {

  // requête
  void open(DOMString method, DOMString url);
  void open(DOMString method, DOMString url, boolean async);
  ...
  void setRequestHeader(DOMString header, DOMString value);
  void send();
  void send(Document data);
  ...
  void abort();

  // réponse
  readonly attribute unsigned short status;
  readonly attribute DOMString statusText;
  DOMString getResponseHeader(DOMString header);
  DOMString getAllResponseHeaders();
  readonly attribute DOMString responseText;
  readonly attribute Document responseXML;

  ...
}
```

XMLHttpRequest : Les méthodes et propriétés

Aix*Marseille université

```
interface XMLHttpRequest {

  ...

  // gestionnaire d'évènement
  attribute Function onreadystatechange;

  // états
  const unsigned short UNSENT = 0;
  const unsigned short OPENED = 1;
  const unsigned short HEADERS_RECEIVED = 2;
  const unsigned short LOADING = 3;
  const unsigned short DONE = 4;
  readonly attribute unsigned short readyState;

};
```

XMLHttpRequest : Les états

Aix*Marseille université

Les différents états de XMLHttpRequest :

- `UNSENT` (0) : L'objet a été construit.
- `OPENED` (1) : La méthode `open` a été invoquée avec succès. Le header de la requête peut être modifié avec la méthode `setRequestHeader`.
- `HEADERS_RECEIVED` (2) : Le header HTTP de la réponse a été reçu.
- `LOADING` (3) : Le corps de la réponse est en cours de téléchargement.
- `DONE` (4) : Le transfert des données est terminé (ou erreur !).

XMLHttpRequest : Requête asynchrone GET

Exemple de requête GET :

```
xhr = createXhrObject();

xhr.onreadystatechange  = function() {
    if(xhr.readyState  == 4)
    {
        if(xhr.status  == 200) {
            alert(xhr.responseText);
        }
        else {
            alert("Error : "+xhr.status);
        }
    }
};

xhr.open("GET", "partie.php?a=12&b=13",  true);
xhr.send(null);
```

XMLHttpRequest : Requête asynchrone POST

Aix*Marseille université

Exemple de requête POST :

```
xhr = createXhrObject();

xhr.onreadystatechange = function() {
    if(xhr.readyState == 4)
    {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
        else {
            alert("Error : "+xhr.status);
        }
    }
};

xhr.open("POST", "partie.php", true);
xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
xhr.send("a=12&b=13");
```

AJAX et jQuery

Exemple de requête POST :

```
$.ajax({
   type: "POST",
   url: "toto.php",
   data: {numero : "123", nom : "bob"}
}).done(function( msg ) { alert( msg ); })
   .fail(function() { alert("erreur"); })
   .always(function() { alert("fini"); });
```

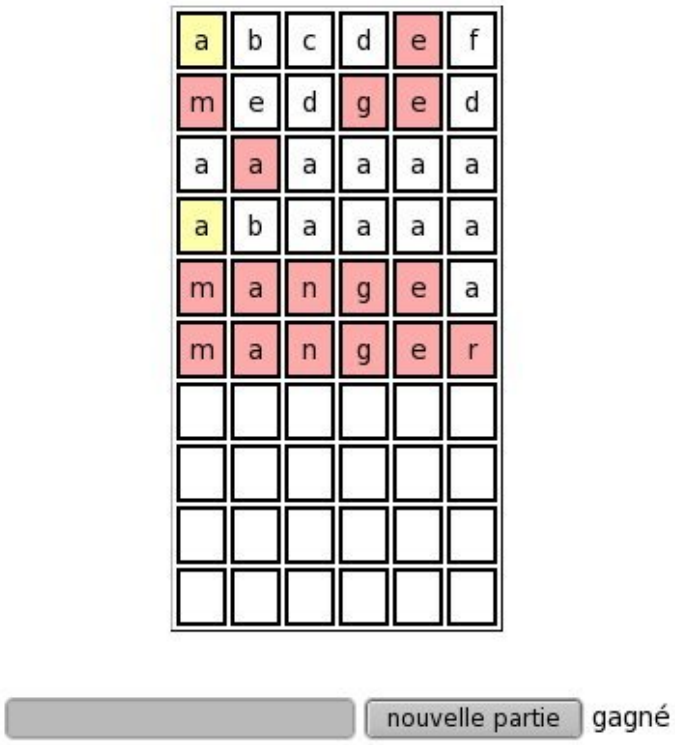
Exemple de requête GET :

```
$.get("test.php", { 'choices[]': ["Jon", "Susan"]} );
```

Projet AJAX

Présentation du projet : Réalisation d'un jeu client/serveur en Ajax.

Le "plateau" de jeu est constitué d'une grille de 6 colonnes et 10 lignes :



Projet AJAX

Aix*Marseille université

Déroulement de la partie :

- La grille est vide et le serveur choisit un mot à faire deviner au joueur.
- Le joueur tape un mot de 6 lettres dans la zone de texte puis appuie sur la touche entrée pour soumettre le mot au serveur.
- Le serveur associe alors une couleur à chaque lettre du mot :
  - La couleur rouge est associée aux lettres bien placées.
  - La couleur jaune est associée aux lettres mal placées mais présentes dans le mot à trouver.
- Les couleurs associées à chaque lettre sont envoyées au client.
- La partie s'arrête une des deux conditions est vérifiées :
  - Le joueur a trouvé le mot $\rightarrow$ gagné ;
  - Le joueur a rempli toutes les lignes de la grille $\rightarrow$ perdu ;

Projet AJAX

Aix*Marseille université

Travail à réaliser : Réaliser ce jeu en utilisant la technologie Ajax.

- Le client (écrit en JavaScript et HTML) doit être capable de :
  - afficher la grille
  - afficher les réponses du serveur
  - envoyer des requêtes au serveur
- Le serveur (écrit en PHP) doit être capable de :
  - répondre au client
  - suivre la partie en utilisant le mécanisme des sessions.

Projet AJAX

Le serveur répond à deux types de requêtes :

- **Action d'initialisation** : demande du client au serveur de choisir un nouveau mot et de lui communiquer la taille de la grille via l'URL `"jeu.php?action=init"`.
- **Action de jeu** : Le client soumet un mot au serveur via l'URL `"jeu.php?action=jeu&mot=lemotsoumis"`.

A la suite de chacune des requêtes, le serveur envoie une réponse au client. Cette réponse doit être **facilement interprétable** par le code JavaScript qui est exécuté sur le client.

JSON

Aix*Marseille université

Pour faire transiter les données du serveur vers le client, nous allons utiliser le format de données textuel JSON (JavaScript Oject Notation) :

```
{"machin": {
  "taille": "12",
  "style": "gras",
  "bidule": {
    "machin": [
      {"style" : "italique" },
      {"style" : "gras" }
    ]
  }
}}
```

L'avantage de JSON est qu'il est reconnu nativement par JavaScript (mais le langage XML aurait également pu être utilisé)

JSON

Aix*Marseille université

Les éléments en JSON :

- Les objets : `{chaîne : valeur, chaîne : valeur...}`
- Les tableaux : `[valeur, valeur, ...]`
- Les valeurs : chaîne, nombre, objet, tableau, true, false, null
- Les chaînes : `"abcdef"` ou `"abcd\n\t"`
- Les nombres : `-1234.12`

```
{"unObjet": {
  "unTableau": [12, 13, 53],
  "unNombre" : 53,
  "unChaîne" : "truc\n"
  "unObjet" : { "style" : "gras" }
}}
```

JSON, AJAX et PHP

Aix*Marseille université

Côté serveur :

```
header('Content-type: application/json');
$a = array('init', 10, 6);
echo json_encode($a);
```

Côté client :

```
var reponse = eval('(' + xhr.responseText + ')');
if (reponse[0]=='init')
   construireGrille(reponse[1], reponse[2]);
```

jQuery, AJAX, JSON et PHP

Aix*Marseille université

Côté serveur :

```
header('Content-type: application/json');
$a = array('nom'=>$_POST['nom'],
           'maj'=>strtoupper($_POST['nom']));
echo json_encode($a);
```

Côté client :

```
$.ajax({
  type: "post",
  url: "a.php",
  data: { nom : "bob" }
}).done(function( msg ) {
  alert( msg['nom']+"=>"+msg['maj'] );
});
```

Projet AJAX

Exemple de protocole de communication du serveur vers le client :

- `["init", 6, 10]`
- `["ligne", "l", "#FFAAAA", "a", "#FFFFAA", ...]`
- `["perdu", "l", "#FFFFFF", "a", "#FFFFFF", ...]`
- `["gagne", "l", "#FFAAAA", "a", "#FFAAAA", ...]`

Le serveur ne doit jamais envoyé le mot à trouver au client !

Projet AJAX

Aix*Marseille université

Améliorations possibles :

- Comptes utilisateurs
- Scores (temps pour résoudre dix mots par exemple)
- Classements
- Vérification des mots dans un dictionnaire
- Jeux à plusieurs