TP

3 Construction d'un site en utilisant AJAX

Nom et prénom :

Introduction et objectifs du TP

Construire un site web permettant au directeur d'un magasin de planches de surfs d'être tenu au courant de son chiffre d'affaires. Au cours de ce TP les points suivants vont être abordés :

1. Analyse des besoins, mise au point de la maquette et du diagramme de l'application
2. Conception (Base de données, code HTML, PHP ... puis AJAX)

Ce TP a pour objectif de vous remettre en mémoire les bases du HTML/CSS/PHP et de vous amener progressivement vers une solution comportant de l'AJAX.
Pour faire fonctionner ce TP (si vous êtes sous windows), vous devrez télécharger le logiciel movamp à l'adresse suivante :
http ://www.movamp.net/telechargement Une fois lancé vous avez accès à un serveur web, une base de données et un interpréteur PHP.
Dans une fenêtre web tapez $127.0.0.1$ pour vérifier que vous avez bien accès au serveur web.
si vous êtes sous linux, vous n'avez rien à faire. LAMP est déjà installé.

3.1 Quelques précisions sur Javascript

Avant d'entrer dans le vif du sujet voici quelques précisions/tests concernant Javascript. Il s'agit d'un langage interprété dont le code est chargé en même temps que la page. Ce code peut être soit embarqué directement dans le HTML (exemple 3.1)

```
<html>
  <head>
    <script language="Javascript">
      // Code javascript
    </script>
  </head>
</html>
```

ou lié en tant que ressource externe comme le sont les pages CSS (exemple 3.1).

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

Tout le code qui sera chargé, soit entre les balises `<script>`, soit depuis un fichier externe, sera aussitôt exécuté. Ceci dit, il est également possible de définir des fonctions. Le script 3.1 montre l'exemple d'une fonction permettant d'afficher une boite de dialogue (appel de la fonction `alert`).

```
<script language="Javascript">
  function bonjour()
  {
    alert("Salut !");
```

```
  }
</script>
```

Ces fonctions sont généralement utilisées en tant que fonction de retour (*callback*) asso ciée à des événements. Pour mettre en place le lien, il suffit de déclarer la fonction comme une valeur de l'attribut pour l'événement concerné. L'exemple `hellob` montre comment associer l'exécution de la fonction `bonjour` au clic sur un bouton.

```
<html>
  <head>
    <script language="Javascript">
      // Code javascript de la fonction bonjour
    </script>
  </head>
  <body>
    <button onclick="bonjour();">Cliquez ici</button>
  </body>
</html>
```

3.2 Analyse des besoins et conception

Le directeur d'un magasin de surf vous demande de développer une solution permettant de suivre, de chez lui, la quantité de surfs vendus ainsi que le bénéfice total réalisé. Après avoir un (bref) entretien, il vous indique que son critère est la rapidité d'exécution (quand il veut son chiffre d'affaire : il le veut tout de suite !). Il vous indique qu'une interface sobre lui conviendra parfaitement. Pour calculer le bénéfice net, vous disposez des informations suivantes : planches vendues (variable), prix de vente et prix de revient.
Après avoir réfléchi, vous décidez de réaliser une page web affichant le nombre de planches vendues, le prix de vente d'une planche, le prix de revient et le bénéfice net réalisé. Vous décidez également d'insérer un bouton dans la page web. A chaque nouveau clic sur ce bouton, la page web sera mise à jour. Cette mise à jour modifie le nombre de planches vendues et après calcul le bénéfice net réalisé par le directeur.
Après une réunion avec votre groupe de "designers" vous décidez d'opter pour l'interface suivante (le code source des exercices de ce TP est sur mon espace perso du réseau).

Question 1
N'étant pas encore rompu à l'AJAX, vous décidez de réaliser votre site de manière synchrone.
Proposez un schéma simulant les échanges entre l'utilisateur, le navigateur et le serveur web/php. Vous devez placer les 4 évènements suivants sur votre schéma : *le navigateur charge la page que le serveur renvoie et l'affiche, clique sur le bouton pour mettre à jour les totaux, le serveur exécute un script php qui* renvoie ce nombre ainsi que le bénéfice dans une page HTML complètement neuve, le serveur reçoit une requête demandant l'exécution

Question 2
Etudiez le code source de *planches.html* et indiquez à quoi correspondent les lignes suivantes :

```
1
2 <link rel="stylesheet" type="text/css" href="boards.css"/>
3
4 <td><span id="planches-vendues">1012</span></td></tr>
5
6 <form method="GET" action="getVentesActualisees.php">
7 <input value="Actualiser" type="submit"/>
8 </form>
```

En étudiant le code source de getVentesActualisees.php. Vous remarquerez qu'il n'y a pas d'appel à une base SQL (ça sera pour plus tard). A la place, on tire aléatoirement le nombre de planches vendues. Appuyez sur le bouton "actualiser" et vérifiez que la mise à jour est correctement réalisée.

Question 3
Editez le code source de cette page générée. Quelles sont les parties de cette page qui ont "réellement" changé après actualisation de la page ? Que déduisez vous sur ce modèle consistant à faire régénérer l'ensemble de la page par le serveur ?

3.3 Réalisation du site avec le protocole XMLHTTPRequest()

3.3.1 Quelques informations sur AJAX

Qu'est-ce que cet Ajax devenu à la mode ces derniers temps ? Il s'agit d'un terme inventé en février 2005 par Jesse James Garret pour désigner l'usage commun de différentes technologies [1]. Pour savoir lesquelles sont utilisées, il suffit d'étudier ce que veut dire ce terme : AJAX = « Asynchronous Javascript And XML ». Visiblement, il y a du Javascript. Sa présence ici indique que les sites Ajax mettront en place un certain niveau d'interaction avec l'utilisateur. Au tour du XML maintenant : ce formalisme, qui permet de représenter n'importe quel ensemble structuré de données, désigne ici l'utilisation du DOM (il s'agit de la représentation de la page sous la forme d'un arbre XML) pour modifier le contenu des pages mais pas seulement. Le XML est aussi employé pour coder les transferts entre le navigateur Web et le serveur du site visité. Il ne nous reste plus qu'à traiter le cas du « Asynchronous »... En fait, c'est à ce terme qu'Ajax doit son succès. Il désigne une technologie mise au point par Microsoft en 1999 et qui jusqu'à présent n'était pas particulièrement célèbre. Celle-ci permet à un programme Javascript d'envoyer une requête au serveur puis de continuer son exécution sans attendre que la réponse n'arrive. Une fois cette réponse effectivement arrivée, une fonction est appelée et le résultat peut être utilisé dans la page.

Et c'est tout ?
En fait, oui, Ajax ce n'est rien que ça. Mais ceci dit, si elles sont utilisées judicieusement, les requêtes asynchrones peuvent à la fois alléger la charge des serveurs et permettre la création de sites plus interactifs. Ce qui est pratique, c'est qu'il est possible (grâce à DOM), de ne modifier qu'une partie de la page afin de faire apparaître le résultat d'une requête. Etant donné que le serveur n'aura pas eu besoin de renvoyer toute la page, celui-ci aura une charge moindre et puisque l'on peut se passer de ce re-chargement complet de la page, le site Web devient plus interactif. L'AJAX fait donc partie des technologies permettant d'améliorer significativement les IHM du WEB.

3.3.2 Création d'un objet requête

Pour envoyer une requête au serveur il vous faut créer un objet. Le côté délicat de la chose est que, selon le navigateur, il faut créer des objets différents. Le constructeur XMLHttpRequest permet de créer un objet requête pour Mozilla, Firefox et Safari. ActiveXObject permet de créer un objet requête pour IE.

Question 4
A l'aide de vos connaissances en Java, commentez le contenu du fichier "CreerRequete.js". Vous détaillerez tout particulièrement l'intérêt de bloc "catch et try".

Question 5
En vous référant au rappel sur le Javascript du premier paragraphe, modifiez l'entête du fichier HTML afin de pouvoir appeler la fonction CreerRequete() depuis n'importe quel endroit de ce fichier.

3.3.3 Ecriture de la fonction GetPlanchesVendues()

Dans la version actuelle de votre site, le clic sur le bouton actualiser envoie un formulaire au serveur qui vous renvoie du code HTML. Nous désirons maintenant que le clic sur ce bouton déclenche, à la place, l'envoi d'une requête asynchrone.

1. En cela Ajax est comparable au DHTML. DHTML = Dynamic HTML = Javascript + (X)HTML (+ CSS)

Question 6
Modifiez le type de la balise input afin qu'elle devienne un bouton. Ajoutez également un gestionnaire d'évènements sur ce bouton (utilisez l'évènement onClick) afin que la fonction getPlanchesVendues() soit appelée à chaque clic de souris.

Nous allons maintenant écrire le contenu de cette fonction. 3 étapes sont nécessaires pour envoyer une requête au serveur.

1. Créer l'objet **requete** en question
2. Initialiser la connexion à l'aide de la méthode open : requete.open("GET",URL,true) ; Le premier argument indique si l'envoi de la requête est en mode GET ou POST, le deuxième contient l'url du script php qui s'exécute sur le serveur WEB, le dernier argument indique si la requête doit être asynchrone ou non.
3. Envoi de la requête au serveur via la méthode send qui prend en argument les données à envoyer au script (mot-clef **NULL** si vous n'avez rien à lui envoyer)

Question 7
En suivant ces indications et en vous inspirant de ce que vous avez vu en cours, écrivez le contenu de la fonction getPlanchesVendues()

Du côté du serveur, il faut modifier le script php. Là où le script calculait les nouveaux chiffres, puis renvoyait du code HTML au navigateur, il va falloir maintenant que le script renvoie simplement la valeur du nombre de planches vendues.

Question 8
A l'aide de l'instruction `echo`, faites en sorte que le script php retourne uniquement le nombre de planches vendues.

3.3.4 Gestion de la réponse du serveur

Maintenant que vous avez réussi à gérer l'envoi d'une requête et que le serveur vous a répondu, il va falloir vous occuper de cette réponse et mettre à jour les chiffres du tableau. Pour cela nous allons écrire la fonction actualiserPage() et indiquer dans quel cas de figure il faut la traiter.

Nous allons tout d'abord modifier la fonction getPlanchesVendues() afin qu'il soit explicitement indiqué ce qui doit être fait au retour de la réponse. On parle de fonction "callback".

Question 9
La propriété onreadystatechange de l'objet requête permet justement de faire cela. Il faut assigner une fonction à cette propriété pour effectuer des traitements sur les données renvoyées après la requête.

```
requete.onreadystatechange=functionaExecuter
```

Insérer la ligne de code dans getPlanchesVendues() permettant d'appeler la fonction actualiserPage(). Attention, l'endroit où vous insérez cette fonction a une importance !

Les données retournées par le serveur sont placées dans la propriété `responseText` de l'objet requête.

Question 10
A l'aide de l'attribut `responseText` de la requête, écrivez la première ligne de la fonction getPlanchesVendues() afin de récupérer la réponse du serveur et de l'affecter à une variable.

Il ne reste plus qu'à mettre à jour la valeur dans les tableaux. Pour cela vous aurez besoin de 3 fonctions :

1. La méthode JavaScript qui s'applique sur l'objet document (qui représente la page HTML) est getElementbyId("id"). Par exemple, si vous voulez obtenir une référence sur la valeur ou est indiqué le nombre de planches vendues, il suffit d'écrire le code suivant :

```
1  var nbPlanches=document.getElementbyId("planches-vendues")
```

2. Les valeurs à modifier étant dans une balise **span**, il n'y a pas de champ "value" qui lui est associé. Il faut obligatoirement passer par des fonctions JavaScript basées sur le DOM. Pas de panique, elles sont déjà écrites dans le fichier "texte-utils.js". Vous devrez donc utiliser la fonction

```
1  remplacerTexte("nomDeLaVariableaModifier","nouvelleValeur");
```

3. Ce qui est renvoyé par getElementbyId() est un élément DOM. Pour pouvoir manipuler la valeur récupérée (par exemple pour faire des additions, multiplications...), il faut utiliser la fonction getTexte("element"). Exemple :

```
1
2 var prixEl = document.getElementById("prix");
3 var prix = getTexte(prixEl);
```

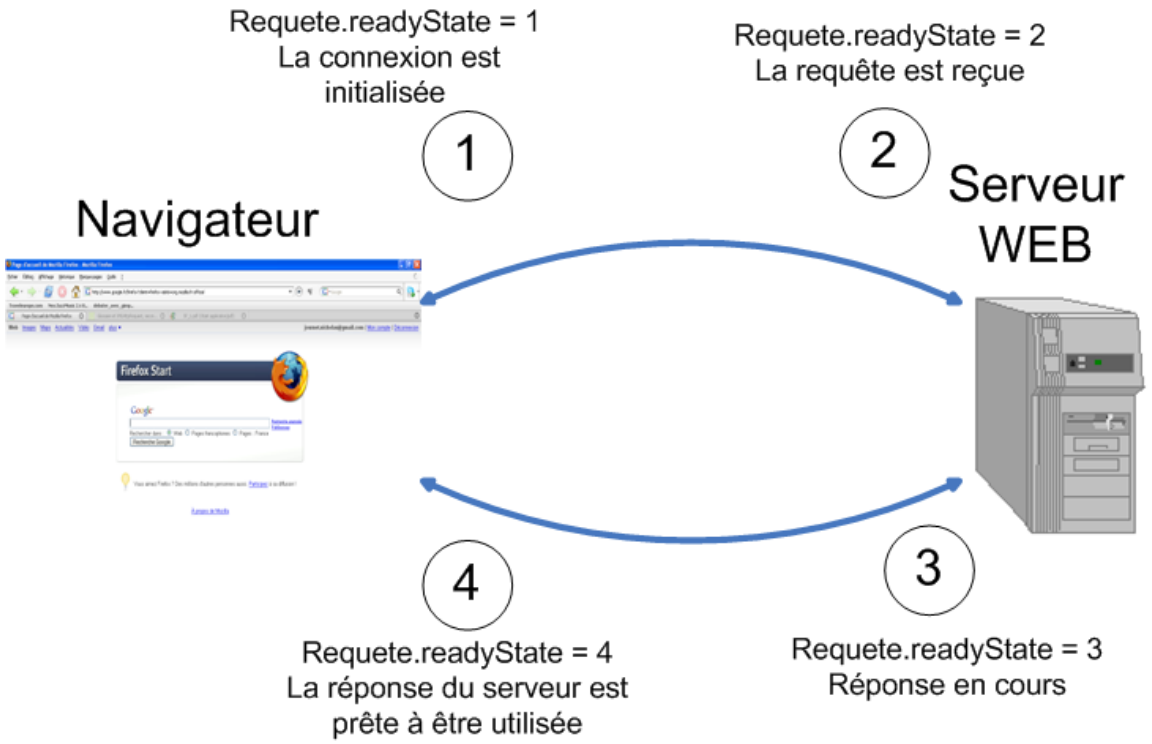
Question 11
Ecrivez la fin de getPlanchesVendues(). Aide : en plus de la réponse du serveur, vous devez récupérer toutes les valeurs contenues dans le fichier HTML (prix unitaire, cout) et calculer le nouveau bénéfice.

```
1      var planchesVenduesEl =                                    ;
2      var benefEl =                          ;
3      remplacerTexte(                        );
4
5      /* Calculer le benefice realise */
6      var prixEl =                           ;
```

```
7        var prix = getTexte(        );
8        var coutEl =                                    ;
9        var cout =                  ;
10       var benefParPlanche =                ;
11       var benef =                                     ;
12
13       /* Mettre a jour le benefice net dans le formulaire */
14       benef = Math.round(benef * 100) / 100;//petite astuce pour affichage
15
```

3.3.5 Vérifier que le serveur a fini

Pour le moment, votre fonction `actualiserPage()` suppose que le serveur a fini quand elle s'exécute... mais ce n'est pas le cas ! Pour comprendre il faut s'intéresser aux codes d'état. L'objet requête possède un attribut (`readyState`) qui informe le navigateur de l'état dans lequel se trouve la requête. C'est à chaque changement de ce code d'état que la fonction actualiserPage() est appelée.
La figure suivante résume les 4 états possibles de la requête.



Question 12
Rajoutez la ligne de code dans `actualiserPage()` qui permet de n'exécuter que la mise à jour des variables uniquement si l'on se trouve dans l'état indiquant que la réponse du serveur est prête à être utilisée.

3.4 Déjouer les pièges de l'Ajax

A partir d'ici votre site web devrait fonctionner. Faites quelques essais en appuyant sur le bouton actualiser et observez qu'il n'y a pas de rechargement de page. Si vous travaillez sur IE et que le nombre de planches vendues ne change qu'une seule fois alors que vous avez appuyé plusieurs fois sur le bouton Actualiser, la fin de ce TP va vous permettre d'aller un peu plus loin dans la mise en place d'un site comportant de l'AJAX et vous éviter ainsi de faire quelques erreurs de débutants ; $-$)

3.4.1 Vérifier l'ansynchronisme

Avant toutes choses nous allons vérifier que les requêtes sont bien asynchrones.

Question 13
Effectuez les modifications suivantes :

1. Dans le fichier php, exécutez une boucle allant de 1 à 5000000 et qui ne fait rien. Cette astuce permet de simuler un serveur qui met du temps à traiter la requête.
2. Dans le fichier HTML, rajoutez un champ texte dans lequel vous pouvez écrire au clavier. Une fois la modification effectuée appuyez sur le bouton Actualiser et essayez d'écrire dans le champ de texte. Qu'observez-vous ?
3. Modifiez la méthode open() afin de repasser en mode synchrone. Une fois la modification effectuée recommencez la même expérience. Vous devriez observer que l'appel est bloquant (visiblement ce test ne fonctionne plus que sous IE)

3.4.2 Envoi de plusieurs requêtes

L'un des points sur lequel il faut être attentif lorsqu'on utilise des requêtes asynchrones, c'est l'envoi multiple de requêtes à un serveur par le même objet, alors que les réponses précédentes ne sont pas encore revenues. Effectuons le test suivant : dans ActualiserPage() ajoutez tout à là fin la ligne suivante :

```
1 document.getElementById("ZoneTexte").value+=nouveauTotal+"|";
```

Où ZoneTexte est l'id de la zone de texte de la question 13 et nouveauTotal est la variable dans laquelle vous avez récupéré la valeur renvoyée par le script php.

Question 14
Appuyez deux fois de suite sur le bouton Actualiser. Qu'observez vous ? Qu'en déduisez vous ? Décrivez une solution permettant de résoudre ce problème.

3.4.3 Vérification du statut de la requête

Suite à une erreur de manipulation sur le serveur php, le script "getVentesActualisees.php" est effacé.

Question 15
Simulez l'effacement du fichier (changer le nom du fichier devrait suffire). Qu'observez-vous ? Pourquoi ?

Pour éviter ce genre de problème, il y a une une propriété de votre objet requête, nommée **status**, que vous pouvez utiliser pour déterminer s'il y a un problème. Le serveur va associer la valeur 200 à la propriété **status** de la requête si tout va bien et 404 sinon. Donc même si une erreur intervient au niveau du serveur, le navigateur exécute TOUJOURS votre fonction callback.

Question 16
Modifiez actualiserPage() afin que la mise à jour de la page web ne s'exécute que si le status retourné est de 200.

3.4.4 Problème de cash

Regardons de plus près comment les navigateurs tels que IE et Opéra procèdent exactement et essayons de comprendre ce qui pose problème pour nos requêtes asynchrones.

1. Après avoir appuyé sur le bouton actualiser, votre code envoie une requête à un serveur web : JavaScript construit une URL et envoie une requête à cette URL.
2. Le serveur retourne une réponse : la fonction callback est appelée et la page WEB est mise à jour
3. Vous appuyez encore une fois sur le bouton actualiser. L'URL envoyée au serveur web est la même qu'au premier point. Le serveur web le remarquant et croyant vous faire gagner du temps, il n'exécute pas le script php et vous renvoie la valeur qu'il a en cache (donc la même que celle du point 1).

Le navigateur se moque de la manière dont l'URL diffère ; il faut simplement qu'elle soit différente. Nous pouvons donc ajouter un paramètre factice à l'URL et lui donner une autre valeur chaque fois que nous envoyons une requête.
Modifiez la fonction getPlanchesVendues() afin de rajouter un attribut bidon à l'URL de la requête envoyée.

```
1 var url="getVentesActualis\'ees.php";
2 url=url+"?bidon="+new Date().getTime();
```

Question 17
Pourquoi l'ajout de `Date().getTime()` génère bien une URL différente à chaque fois ?