IUT- Département d'Informatique
G. Rey
2016

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

1 Objectifs

Ce TD illustre la partie du cours sur la programmation web et en particulier les concepts liés à la conception d'application riches coté client, c'est-à-dire dans le navigateur. Les concepts principaux abordés seront:

- Révision de l'ensemble des connaissances sur JavaScript, le DOM, les formats de fichier XML et JSON et AJAX
- Utilisation de la bibliothèque jQuery

Un compte rendu donnant la réponse à chacune des questions posées devra être rendu à votre responsable de TD par email à la fin de chacune des séances. L'objet (sujet) de l'email devra être le suivant :

[S4T][JS][Gx][TDx]Nom-Prénom / Nom-Prénom

Avec Gx le numéro de votre groupe (exemple G1 pour le groupe 1) et TDx le numéro du TD inscrit sur le sujet du TD (exemple TD1 pour le premier TD).

Vous ne pouvez pas faire 2 séances avec le même binôme. De même, vous ne pouvez pas faire plus de la moitié des séances seul sauf si votre groupe de TD est assez petit pour que chaque étudiant soit sur une machine de l'IUT.

Veuillez, sauf indication contraire de votre responsable de TD, faire l'ensemble des exercices marqués « Obligatoire » dans l'ordre de la feuille de TD. Les exercices « Conseillés » sont à faire si vous avez terminé les précédents ou lors de vos révisions. Les exercices « optionnels » sont là pour les plus passionnés d'entre vous.

2 Prise en main de l'environnement (obligatoire)

Pour réaliser ce TD, vous aurez également besoin d'un éditeur de texte pour écrire le code de vos pages. Vous pouvez par exemple utiliser un des logiciels Notepad++, ConText, Quanta+, WebExpert,... Microsoft vous propose de télécharger gratuitement Expression Web.

Pour ce TD, nous aurons besoin de faire fonctionner nos pages sur un véritable serveur web, capable de répondre à nos requêtes http et d'interpréter des scripts PHP (coté serveur). Pour cela nous utiliserons le serveur mis à votre disposition par le département informatique de l'IUT. Les informations nécessaires sont disponibles sur cette page.

Pour déposer vos fichiers sur l'espace web, vous pouvez utiliser soit votre partage de fichiers (p:\web\) depuis les machines du département, soit une connexion ssh, scp ou sftp via différents outils (WinSCP, FileZilla, Cyberduck, Core FTP LE, ...) vers le server linserv1 (depuis le réseau interne) soit lindmz.unice.fr depuis l'extérieur.

Concernant la bibliothèque jQuery, vous devrez obligatoirement utiliser la version jQuery 1.12.1. Pour plus d'information sur l'intégration de la bibliothèque dans vos pages, reportez-vous aux supports de cours.

Dans votre compte rendu, vous devrez donc d'une part rendre l'ensemble des fichiers (html, css et js) que vous avez écrit. Mais également le lien sur votre travail. Le lien aura la forme suivante : https://linserv3.iutnice.unice.fr/~xy123456. De fait, vous ne devez pas utiliser un autre serveur web que celui qu'on vous fournit. Plus d'info sur la page du SIDI.

Pensez à sauvegarder régulièrement votre travail et à commenter votre code. N'hésitez pas à sauvegarder les différentes versions, évolutions d'un même exercice de manière à pouvoir facilement réviser ou le réutiliser par la suite.

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

3 Conseils : lisibilité, maintenance et performance

Avant de vous aventurer dans cet ultime TD du semestre, voici un petit résumé de conseils à appliquer à votre code JavaScript et à l'utilisation de bibliothèques. Nombre de ces conseils restent applicable à tous types de programmes indépendamment du langage.

3.1 lisibilité, maintenance

- **Structurez et organisez votre code** : regrouper votre code (classes, fonctions, …) en respectant une décomposition logique (souvent fonctionnelle, et/ou s'appuyant sur un modèle ou des patrons de conceptions).
- **Indentez votre code** : une bonne indentation améliore la lisibilité du code. Attention, de bien respecter toujours le même schéma de mise en forme de votre code.
- **Commentez votre code** : en plus des entêtes de classes et/ou de fonctions, n'hésitez pas à ajouter des commentaires utiles le long de votre code. Même si votre code vous semble clair et compréhensible sur l'instant, il n'en sera pas forcément de même dans 1 semaine ou 1 mois, ni pour d'autres personnes. Ajoutes pas des commentaires du style : « i++ ; // augment i de 1 ». Ça ne sert à rien. Un commentaire est là pour vous aider à comprendre rapidement et précisément votre code.
- **Choisissez des noms adaptés** : le nom d'une variable/fonction/classe doit indiquer son rôle et/ou l'élément qu'elle contient. Par exemple en jQuery, une convention de nommage est de préfixé par $ les noms de variables qui contiennent des éléments jQuery. Pour facilement les différencier des variables classiques.
- **Séparez les préoccupations** : un fichier de code source ne doit pas contenir des constantes ni du texte. Il est préférable d'externaliser ces éléments. De même, il est préférable d'avoir des fichiers HTML sans style ni script. Le style étant défini dans des fichiers CSS et les scripts dans des fichiers JavaScript.

3.2 Performance

- **Sauvez les résultats de calculs plutôt que les refaire** : Lorsque vous êtes amené à utiliser un élément du DOM ou un objet jQuery plus d'une fois dans votre fonction ou votre script entier, mettez cet objet dans une variable. Cependant, avec jQuery, on peut chainer les commandes, ce qui est encore mieux : par exemple si je veux cacher un élément, changer son arrière-plan et le faire apparaitre en fondu, je peux écrire la commande chainée suivante : « *$('#id').hide().css('background-color', 'red').fadeIn();* »
- **Utilisez des ids** : la recherche par « id » étant très performante, quand vous en avez la possibilité, utilisez cette solution. Attention, avec jQuery, si vous faites une recherche du type « tag#id », la recherche se fera en premier sur les tags. Donc, privilégier l'approche « #id ». De même, au lieu de chercher dans toute la page, il est souvent préférable de cherche un élément (tag, classe, …) depuis l'id le plus proche (exemple « #id tag.class » au lieu de « tag.class ».
- **Utilisez « tag.classe » plutôt que « .classe » :** avec jQuery, la recherche par « tag.classes » est plus efficace que la recherche « .classes ».
- **Modifiez le DOM en une seule fois :** quand cela est possible, il est préférable de ne pas modifier le DOM à chaque itération d'une boucle mais de mémoriser cela dans un variable et de faire la modification du DOM en une fois à la fin.

Voilà quelques conseils, j'espère qu'ils vous seront utiles et pour vérifier cela, je vous demande de vous forcer à tous les appliquer dès le prochain exercice et jusqu'à la fin de ce TD.

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

4 Exercice 1 : retour sur le taquin (obligatoire)

A partir du fichier HTML suivant (que vous n'avez pas le droit de modifier), réaliser le jeu de taquin en utilisant jQuery et en respectant les consignes ci-dessous. **Avant de vous lancer dans le codage du jeu, il est conseillé de réaliser les fonctions demandées par les consignes, dans l'ordre demandé**.

```
	<!DOCTYPE html>
<html lang="fr">
	<head>
		<title>Jeu de taquin</title>
		<meta charset="UTF-8">
		<link rel="stylesheet" type="text/css" href="td5.css" />
		<script	charset="utf-8"	src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.2.min.js"></script>
		<script type="text/javascript" src="./script.js"> </script>
	</head>
	 <body>
		<div id="titre">Jeu de taquin</div>
		<div id="interface">
			<label for="form-large">Largeur (nb cases) : </label>
			<input id="form-large" type="number" value="2" min="2" max="10" size="4"/><br/>
			<label for="form-haut">Hauteur (nb cases) : </label>
			<input id="form-haut" type="number" value="2" min="2" max="10" size="4"/><br/>
			<label for="form-taille">Taille de la case : </label>
			<input id="form-taille" type="number" value="10" min="5" max="20"/><br/>
			<input id="form-nouveau" type="button" value="Nouveau plateau" />
		</div>
		<div id="presentation">Ordonnez les cases de la plus petite valeur à la plus grande, de gauche à droite puis de bas en haut. La case en haut à gauche devant être la case sans numéro. Celle en bas à droite aura la plus grande valeur.</div>
		<div id="jeu"></div>
		<p>Bravo !!</p>
	 </body>
 </html>
```

4.1 Mise en forme

- Créez un fichier CSS qui prendra en charge la mise en page de votre page. Par défaut, les balises de type « paragraphe » ne sont pas visible.

4.2 En JavaScript standard

Pour des questions de simplicité, dans les consignes ci-dessous un tableau sera un tableau à une dimension alors qu'une grille sera un tableau à deux dimensions.

- Ecrivez une fonction initialiseLesValeurs(*tableau*) qui prend en paramètre un tableau vide et qui ajoute dans chaque case une valeur. Chaque case du tableau devra avoir une valeur différente. La liste des valeurs possible va de 0 à taille du tableau-1.
- Ecrivez une fonction *melangerLesValeurs(tableau)* qui prend en paramètre un tableau d'éléments et qui mélange les éléments de ce tableau.
- Ecrivez une fonction *obtenirLElement(grille, valeur)* qui retourne les coordonnées (x, y) de la case de la grille qui contient la valeur recherchée. Chaque case de la grille a une valeur et toutes les valeurs sont différentes. Au besoin on construira un objet *coordonnee*.

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

- Ecrire une fonction estVictoire(grille) qui retourne vrai si les valeurs de la grille sont ordonnées de la plus petite à la plus grande. La case *grille[0][0]* devant être la case avec la valeur 0, *grille[l_max][c_max]* aura la plus grande valeur. Et *grille[n][p+1]* aura une valeur inférieure à *grille[n+1][p]*.

4.3 A l'aide de jQuery

Pour vous forcer à utiliser jQuery, vous n'avez pas le droit d'utiliser les fonctions JavaScript *getElementXXX* ni les fonctions *querySelectorXXX*. De même, toutes les modifications du DOM (structure et style) devront se faire via jQuery.

Attention, comme nous l'avons vu dans le TD précédent avec l'exercice sur 2048, il ne faut pas vous appuyer sur le DOM (la partie graphique de l'application) pour faire des calculs. Mais il faut les faire sur votre modèle.

- Ecrivez une fonction *initialisation()* qui sera exécutée dès que possible. La fonction devra :
  → Récupérer les valeurs du nombre de ligne, de colonne et la taille de chaque case
  → Construire, à l'aide des fonctions précédente, un objet grille de valeurs mélangées allant de 0 à nb_lignes*nb_colonnes-1.
  → Mettre à jour la page HTML (le DOM) en affichant un plateau de jeu (dans la balise identifiée avec l'id « jeu). Ce plateau de jeu devra être une représentation de l'objet grille. Attention à bien prendre en compte la taille des cases et à adapter le texte en fonction de celle-ci.
  → Ajouter un écouteur d'évènement sur le bouton de la page. Cet écouteur sera à usage unique (utilisable une seul fois par initialisation). Si l'utilisateur appuie sur le bouton, il faudra effacer le plateau de jeu, supprimer l'objet grille et relancer une nouvelle initialisation. Attention, il faut aussi réinitialiser les éléments de la page d'origine que vous auriez modifiés avec vos scripts (mais pas les modifications faites par l'utilisateur dans les inputs).
  → Ajouter un écouteur d'évènement « click » sur le plateau de jeu. Cet écouteur testera si on clique sur case. Si c'est le cas, il appellera la fonction *echangeDeuxCases(case)* et testera après cela la victoire ou non. En cas de victoire, on affiche le message de la balise paragraphe et on bloque les actions sur le plateau de jeu.
- Ecrivez une fonction *echangeDeuxCases(case)* qui si et seulement si la case a un bord commun avec la case vide, échangera les cases et mettra à jour l'objet grille. Attention à bien faire en vous appuyant sur votre modèle.

5 Exercice 2 : Images et slideShow (obligatoire)

Dans ce dernier exercice de l'année nous allons essayer de mettre en œuvre les différents éléments vus dans ce module tout en poursuivant notre apprentissage de la bibliothèque jQuery.

En partant de la page HTML ci-dessous (et sans modifier celle-ci), mettez en œuvre les effets demandés :

```
<!DOCTYPE html>
<html lang="fr">
   <head>
      <title>Images et SlideShow</title>
      <meta charset="UTF-8">
      <link rel="stylesheet" type="text/css" href="TD5.2.css" />
      <script charset="utf-8" src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.2.min.js"></script>
      <script type="text/javascript" src="./script2.js"> </script>
   </head>
    <body>
      <div id="titre">Images et SlideShow</div>
      <div id="conteneur"></div>
      <div id="interface"></div>
    </body>
</html>
```

Pour cet exercice, les consultations et manipulations des nœuds du DOM (Core ou HTML) et ou de ces propriétés (incluant les propriétés CSS) devront être effectuées via jQuery.

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

- Dès que la page est opérationnelle, chargez un ensemble de 10 à 20 figures qui se trouveront sur votre serveur. Une figure est une nouvelle balise d'HTML5 qui permet d'associer une légende à une image. Si vous ne connaissez pas cette balise, vous pouvez rapidement consulter cette page. Cela se fera via la fonction *obtenirLaListeDesFigures()*.
- Ajoutez dans un premier temps ces figures à votre page HTML et vérifiez que vous voyez bien l'ensemble des images et de leur légende.
- Modifiez votre CSS et vos scripts, pour afficher 4 images en largeur avec la légende écrite en blanc sur un fond noir légèrement transparent. La légende devra être proprement positionnée sur l'image. On notera que pour cela, on peut, par exemple, mettre la figure en position relative, la légende en positon absolue, lui définir une hauteur et une largeur, une position par rapport au haut de l'image et ses autres caractéristiques (n'oublié pas le z-index pour avoir un élément au-dessus de l'autre).
- Modifiez votre CSS et vos scripts, pour que la légende soit cachée par défaut. Celle-ci apparaitra lors du survole de la figure par la souris. La première version de votre fonction devra utiliser *.hover(handlerIn, handlerOut)* alors qu'une deuxième version devra utiliser *.hover(handlerInOut)*. Dans la version 1, deux fonctions différentes seront appelées lors de l'entrée et de la sortie de la zone de survole. Dans la version 2, une même fonction sera appelée dans les deux cas.
- Ajoutez une règle CSS sur l'élément d'id « *conteneur* » pour qu'une seule ligne d'images soit affichée.
- Ecrivez une fonction *creerLeCarousel(liste,index)* qui va modifier les figures de la liste en paramètre en réduisant à 60% de leur taille les 1ère et 5ème figures de la liste (si on compte à partir de 1) et à 80% de leur taille les 2ème et 4ème figures. Attention à bien aligner ces figures vers le bas. Pour une question de simplicité, on pourra estimer que toutes nos images ont une même taille. L'illustration ci-dessous peut vous donner une idée du résultat attendu.
- Modifiez également votre page pour que seule la 3ème image puisse laisser apparaitre sa légende.
- Toujours sans modifier votre page HTML, ajoutez deux boutons (flèche gauche et flèche droite). Lors d'un clic que l'un des boutons, les images se décalent dans le sens indiqué. Cela devra être fait pas les méthodes *deplacementAGauche(liste,index)* et *deplacementADroite(liste,index)*.

5.1 Pour les plus rapide d'entre vous (conseillé)

Une fois que le reste du TD est terminé et que celui-ci est bien fait (code propre, commenté, …) essayez d'ajouter les fonctionnalités suivantes à votre carrousel.

- Quand si un clic survient sur une des images réduites, le carrousel tourne jusqu'à cette image.
- Le carrousel doit donner l'impression d'être sur un cylindre. On pourra garder l'image centrale comme elle est.
- Votre code doit fonctionner avec des images de taille différentes (la plus petites ayant la taille de l'image centrale). Le ratio largeur/hauteur de toutes les images est le même.
- Même fonctionnalités que la précédente mais cette fois-ci le ratio L/H peut être différents entre les images. On ne déformera pas les images à l'affichage.

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)

6 Exercice 3 : Atom Reader (optionnel)

Un document au format Atom est appelé un « fil de syndication Atom » ou fil Web. Ces fils peuvent être affichés aussi bien sur un site Web que directement dans un agrégateur, qui est un logiciel prévu à cet effet. Cela permet de suivre, ou « s'abonner », à un fil. Le propriétaire d'un site web peut quant à lui utiliser un logiciel spécialisé, tel qu'un système de gestion de contenu, pour publier une liste de ressources, dans un format standardisé et lisible par une machine, et dont il souhaite notifier des mises à jour.

Le développement d'Atom a été justifié par le manque de flexibilité commun aux nombreuses variantes de RSS. [Wikipédia]

6.1 Ajax avec jQuery

L'idée est ici de refaire la même chose que précédemment avec RSS mais en utilisant jQuery et en analysant le format Atom.

Puisque nous recevons un fichier Atom, il s'agit d'un fichier XML si vous passer par le proxy.php ou un fichier JSON si vous utilisez le proxyJson.php. Il vous reste donc mettre en place un script capable d'analyser ces documents et de nous les présenter de manière plus agréable dans notre page.

Si vous avez besoin de flux Atom, en voici quelques-uns :

- http://www.senat.fr/rss/presse.xml
- http://videos.senat.fr/video/videos.xml

Vous trouverez ci-dessous le code des différents proxys.

6.2 Exemples de proxy

```
<?php
	$url = $_POST["rssURL"];
	if ($url != ""){
		$monRSSDoc = new DOMDocument();
		if ($monRSSDoc->load($url)){
			header('Content-Type: text/xml');
			echo $monRSSDoc->saveXML();
		}else{
			echo "erreur de lecture";
		}
	}else{
		echo "URL vide";
	}
?>
```

Vous n'avez pas le droit de modifiez ce fichier proxy.php.

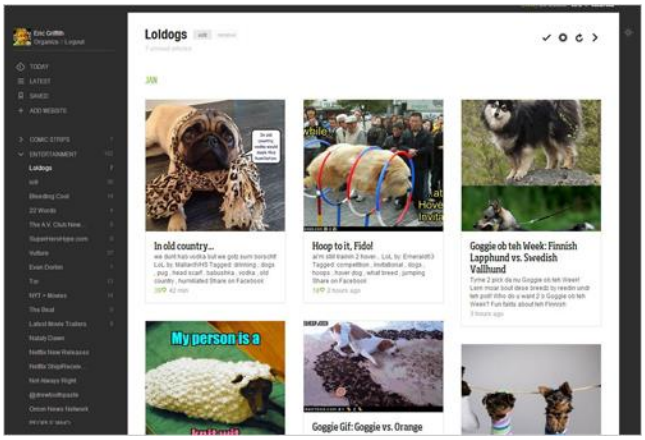
```
<?php
	$url = $_POST["rssURL"];
	if ($url != ""){
		$monRSSDoc = new DOMDocument();
		if ($monRSSDoc->load($url)){
			$xml = simplexml_load_string($monRSSDoc->saveXML());
			echo json_encode($xml);
		}else{
			echo "erreur de lecture";
```

Programmation Web – client riche
M4103 - TD n° 5 (séance 7)


```
                }
        }else{
                echo "URL vide";
        }
?>
```

Vous n'avez pas le droit de modifiez ce fichier proxyJson.php.

6.3 Exemple d'illustrations



7 Conclusion

J'espère que ce cinquième TD vous a permis de découvrir ou redécouvrir jQuery. Ayant refait des choses similaires aux précédents TD en JavaScript, vous a dû pouvoir comparer et mieux appréhender les possibilités qu'offre cette bibliothèque.