IUT - département Informatique
2ème année
Cours : Programmation Client WEB

janvier 2018
J-M. Ilié,
C. Kurtz

TP JavaScript «programmation d'un qcm»

Objectif technique : Séparation de code HTML et javascript, Programmation événementielle des pages web, requête dynamique AJAX vers le serveur WEB.

CONTEXTE du TP

On désire réaliser un jeu composé de plusieurs questions à réponse unique.

- Une invite à cliquer est affichée avant la question, afin que l'utilisateur lance le jeu.
- le lancement du jeu consiste à afficher une question et à lancer un timer (le joueur a 1mn pour répondre avant l'échec). La question est composée d'un ensemble de réponses proposées et l'utilisateur doit cliquer sur l'une d'elle pour répondre.
- S'il répond correctement en cliquant sur la bonne réponse, un message d'alerte le félicite, et en cas contraire, un message d'alerte lui explique son erreur. S'il ne répond à temps, un autre message d'alerte indiquant son abandon est affiché.
- Dans tous ces cas, prévoir de relancer le jeu après le message d'alerte affiché.

Partie 1 : Implémentation d'un qcm à une question

1- Attacher le `jeu.js` au fichier `jeu.html`
Pour ce faire, modifier l'attribut `src` de la balise script, sachant que `jeu.js` est dans le sous répertoire `scriptJS`.

2- Compléter le fichier `jeu.js`.
Dans la page html, un bloc div (id="`QUEST`") est prévu pour l'affichage de l'invite ou de la question.
- Compléter `jeu.html` pour que la procédure `init()` soit invoquée dans jeu.html, au chargement de la page (`onload`).
- Compléter `init()`. Cette procédure initialise la variable `q` servant de référence sur la div `QUEST`, puis lance le jeu en invoquant `lance()`.
- Compléter la fonction `lance()` pour qu'elle affiche dans la page le message d'invite `quest`. Vérifier que l'invite est un message interactif et l'expliquer.
- Compléter la procédure question() pour qu'elle affiche la question dans la page et lance le timer. Ce dernier doit lancer une procédure `abandon()` à échéance d'un temps imparti.
- Compléter la procédure `reponse()` pour qu'elle analyse la réponse cliquée par l'utilisateur et affiche un message en correspondance.
- Compléter la procédure `abandon()` pour indiquer à l'utilisateur l'abandon du jeu.
- Terminer ces trois dernières procédures en relançant le jeu.

3- Séparation du code `html` et `javascript`
Déporter les déclarations évenementielles vers *`jeu.js`*
Ainsi, au lieu d'utiliser `onload` dans la balise `body` du fichier `html`,
déclarer le gestionnaire d'événement `load` dans le fichier javascript.
Redéfinir pour cela la fonction `onload()` de l'objet window.

Partie 2 : Etendre le jeu à une série de questions

Transformer le programme pour que le jeu soit composé de plusieurs questions tirées aléatoirement, au lieu d'une seule question. Chaque question sera vue sous la forme d'un objet de 3 attributs : l'intitulé de la question, les choix possibles de réponse et la réponse attendue. L'ensemble des questions seront regroupées dans un tableau `quest`.

- Déclarer le tableau et son contenu sous la forme d'une expression `javascript-json`. Les tableaux indicés sont déclarés entre [] et les objets entre {}. Par exemple, ['va','vb','vc'] est un tableau contenant 3 valeurs : va, vb et vc. De plus {a:'va',b:'vb',c:'vc'} est un objet de 3 attributs (a, b et c) de valeurs respectives va, vb et vc. Toute expression json `exprJson` peut être interprétée avec la fonction `eval(exprJson)` de javascript.
- Comprendre le code de la fonction `question(temps_imparti)`. Notamment, vérifier que l'indice de la question à afficher (`numQ`) est tiré au sort dans cette fonction, en utilisant la fonction `rand`() de javascript, sur les indices du tableau `tabObject`.
  Noter que cette fonction lance `htmlQuestion(numQ)` pour l'affichage HTML de la question à l'utilisateur.
- Ecrire la fonction `htmlQuestion(numQ)`, dont le paramètre `numQ` est l'indice de la question sélectionnée dans `tabObject` . Son but est de rendre au format html, une chaîne de caractères correspondant à l'intitulé de la question et ses choix de réponse. Le format d'affichage désiré est celui d'une liste `<ul>`.
- Faire une boucle indicée pour créer les éléments de liste `<li>`. Chaque élément de cette liste contiendra un bouton radio présentant un des choix de réponse possible. L'attribut value de ce bouton devra mémoriser l'indice de cette réponse (`iChoix`).
- Pour chaque bouton radio, un clic sur le bouton devra lancer la fonction `reponse(iChoix, repGood)`, qui testera si l'indice de la réponse choisie (`iChoix`) correspond à celui de la bonne réponse pour la question traitée. Ecrire cette fonction.

Partie 3 : Accès aux données du qcm à partir une application serveur WEB-PHP

Utiliser le script `qcm.php` fourni, dont le rôle est de produire en sortie le tableau des questions du qcm, au format json. Le but est désormais d'initialiser la variable `tabObject` via une requête AJAX, invoquant à distance `qcm.php`.

Comprendre le contenu du ficher `qcm.php` qui linéarise un tableau de questions, sous la forme d'une chaîne au format `json`.

- la linéarisation en php est faite par la fonction `json_encode($D)` , où $D est la donnée à linéarisée, par exemple un tableau PHP. On obtient une chaîne formatée à transmettre vers le navigateur, avec un `echo()` par exemple.
- La fonction inverse est `json_decode($D)` . Elle pourra servir à récupérer une donnée json transmise en paramètre d'une requête http.

Lancer qcm.php sur serveur web-php via une requête du navigateur :
`http://vs-wamp/...../qcm.php`
et voir le résultat du echo au format JSON.

Réaliser une requête en javascript, destinée à lancer à distance le script `qcm.php` afin d'en récupérer la sortie (à savoir, le résultat des `echo` du script).
Par souci de simplification et en avance sur le cours jquery à venir, réaliser cette requête en jquery, de la façon suivante :

- Dans un répertoire du serveur web-php, placer : `index.html`, `qcm.js` et `qcm.php` .
- Modifier le code javascript de la fonction `window.onload()` pour utiliser l'instruction suivante réalisant dynamiquement une requête de type `GET` vers un serveur web (AJAX).

- La fonction `jquery` suivante réalise cette requête pour invoquer le fichier `qcm.php`,

```
$.getJSON("qcm.php ", function(data){tabObject = data})
```

La fonction déclarée en 2ème paramètre est invoquée à la suite de cette requête pour traiter les sorties reçues du fichier php (`echo…`). Le paramètre data de cette fonction est un paramètre formel correspondant à ce qui a été reçu, à savoir le tableau json émis par le script `qcm.php`.

- Noter la modification de l'entête du fichier index.html ajoutant un fichier de script pour `jquery`:

```
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>
```

- Faire fonctionner et visualiser la notification de la requête dans le débogueur du navigateur.