# Analyse de la partie « Choisir le QCM »

Le diagramme suivant illustre les séquences du fonctionnement :



On note 2 parties dans l’application :

* L’obtention de la liste des QCM
* Après le choix du QCM, son affichage question par question. C’est l’enseignant qui décide de passer à la question suivante.

# Réalisation de l’IHM :

Pour réaliser l’affichage des 3 pages web j’ai utilisé de l’HTML (Langage à balise). Ce langage m’a permis l’insertion d’objet et de texte sur la page web et le CSS m’a permis l’amélioration de la présentation de la page web que ce soit au niveau du positionnement des objets et des textes, le style de texte, la couleur d’un objet, etc.

Actuellement, 3 pages du site on été réalisées :

## Choix du questionnaire :

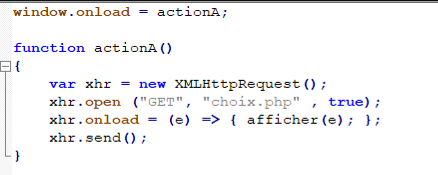
Cette première page va afficher les différents questionnaires présents dans la Base de données.



La copie écran montre le résultat après l’exécution de la requête faite avec AJAX : Les deux vignettes Quizz sont ajoutées dynamiquement à partir des données enregistrées.

**Code source associé :**

Préparation de la requête AJAX :



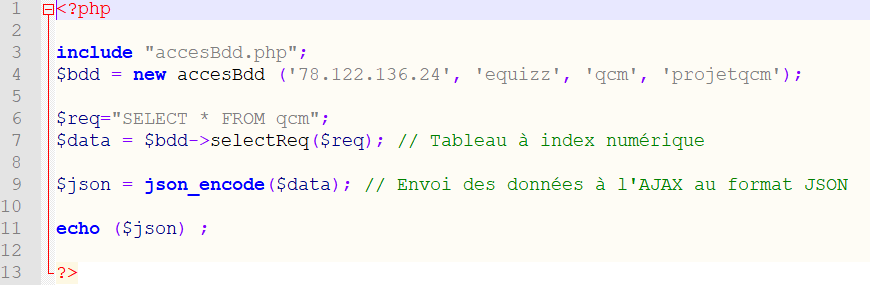
L’objet AJAX est de type XMLHttpRequest et il fait partie des bibliothèques standards de JavaScript.

On note l’utilisation de la variable « e », générée automatiquement par le système des événements.

On m’a également montré la notation des fonctions « arrow » du JavaScript : () => { }

Il permet de faire une requête asynchrone : la page « choix.php » est appelée en arrière plan et la fonction « afficher » sera exécutée lorsqu’il y aura une réponse. Entre temps, le navigateur n’est pas bloqué (c’est la définition de « asynchrone » dans ce cas).

La page PHP consulte la base de données et renvoi le résultat au format JSON :

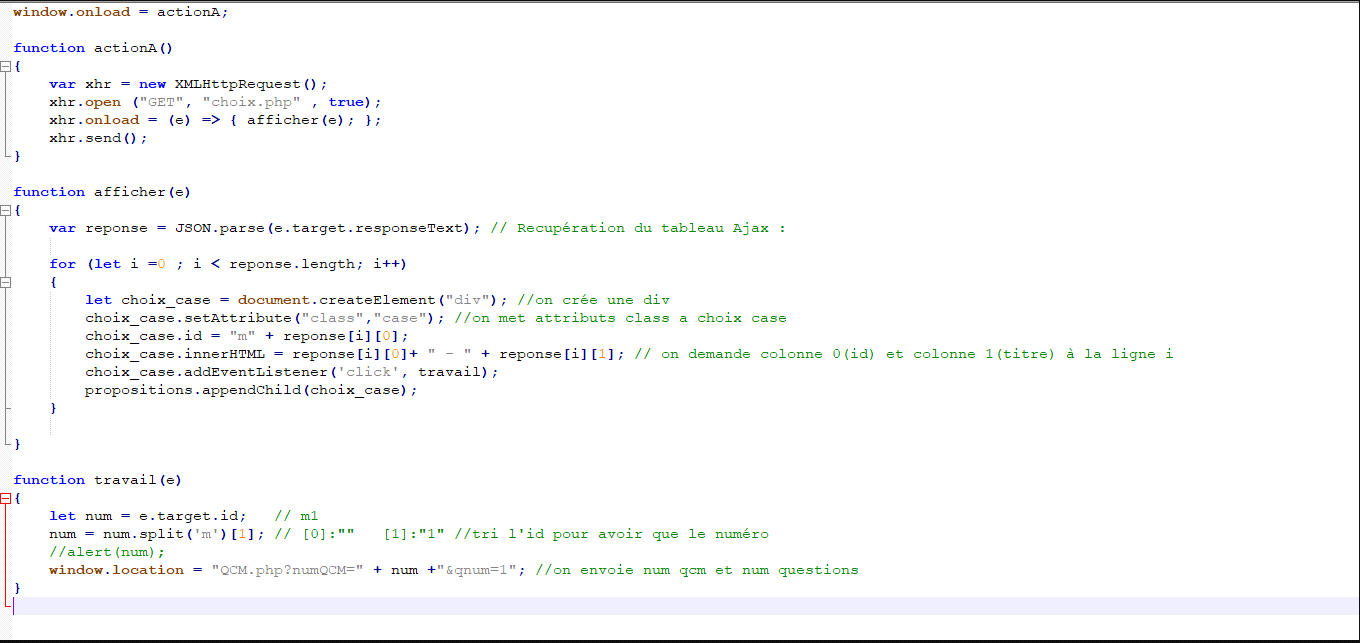


La connexion à la base de données utilise un module externe « accessBDD.php ». Cette librairie contient une classe PHP qui utilise les requêtes préparées PDO (PHP Data Object).

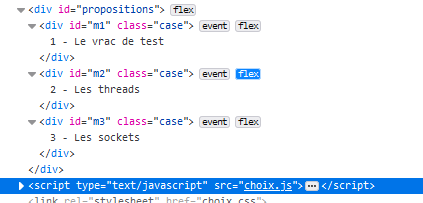
Quand les données PHP JSON sont reçues par le navigateur, le code de la fonction « afficher » est exécuté :

Le JSON est converti en tableau JavaScript : *JSON.parse()*

On parcoure ensuite le tableau pour créer une <div> html pour chaque questionnaire. On associe à chaque <div> un évènement « click » qui appellera le questionnaire correspondant :



Après l’exécution de ce script, le HTML sera le suivant :

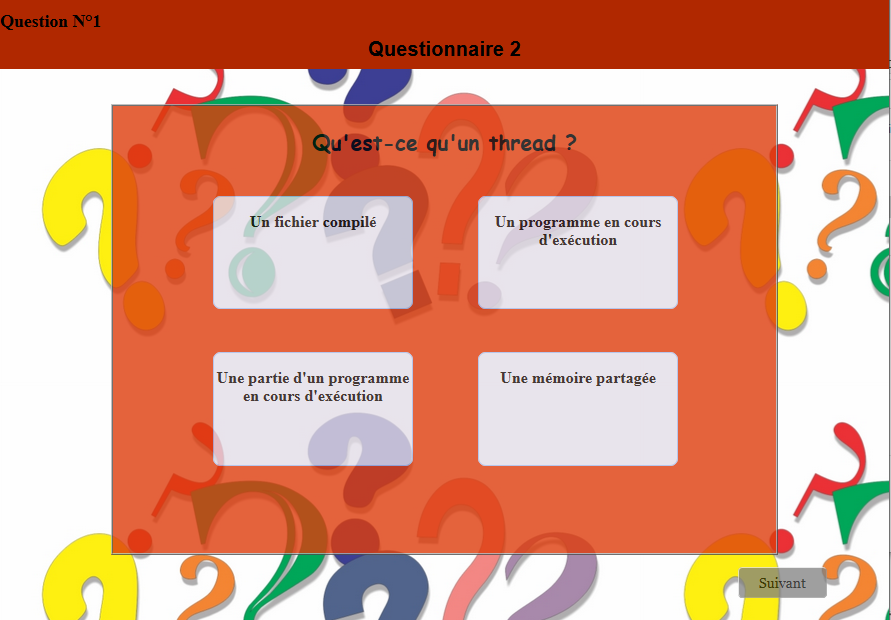


On voit les <div> ajoutées, et l’association avec un « event » qui correspond à l’événement « click ». Le code du script correspond à la fonction « travail » ci-dessus.

## La page du QCM :

La page appelée « QCM.php » est la suivante :

C’est une page créée dynamiquement par le PHP :

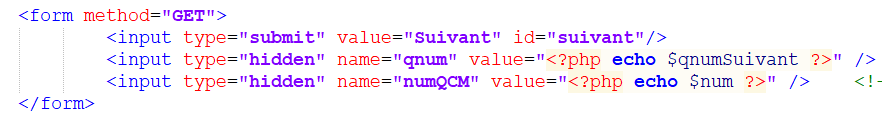


L’extrait de code ci-dessous contient la requête GET qui va récupérer le contenu des variables *numQCM* (Numéro du formulaire) et *qnum* (numéro de la question) venant de la page « choix.html ». Au départ, on demande la question 1.



A chaque appui sur le bouton Suivant, la page se recharge et renvoi une requête SQL avec un rappel de la variable *numQCM* et la variable modifiée *qnum* pour avoir la question suivante.

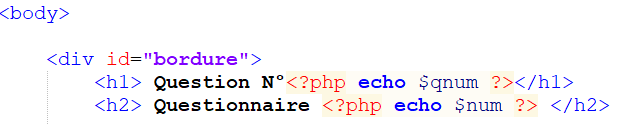
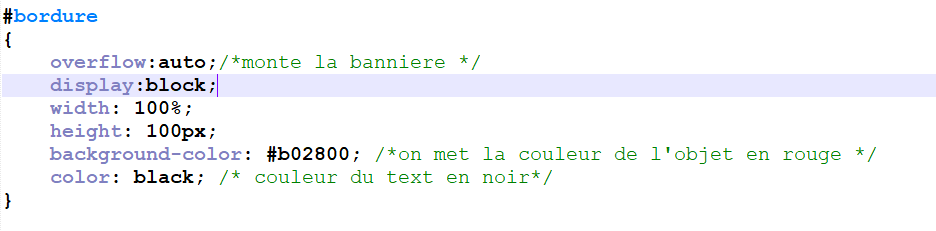
La variable *qnum* est modifiée dans une variable cachée du formulaire lors de l’appui sur le bouton <suivant> :



Et lorsque « $data » sera égale à 0 ou NULL , alors la condition *empty* se réalisera et la page fin.html sera chargée. Dans le futur, on appellera un page qui affiche le résultat du QCM.

## L’utilisation du CSS :

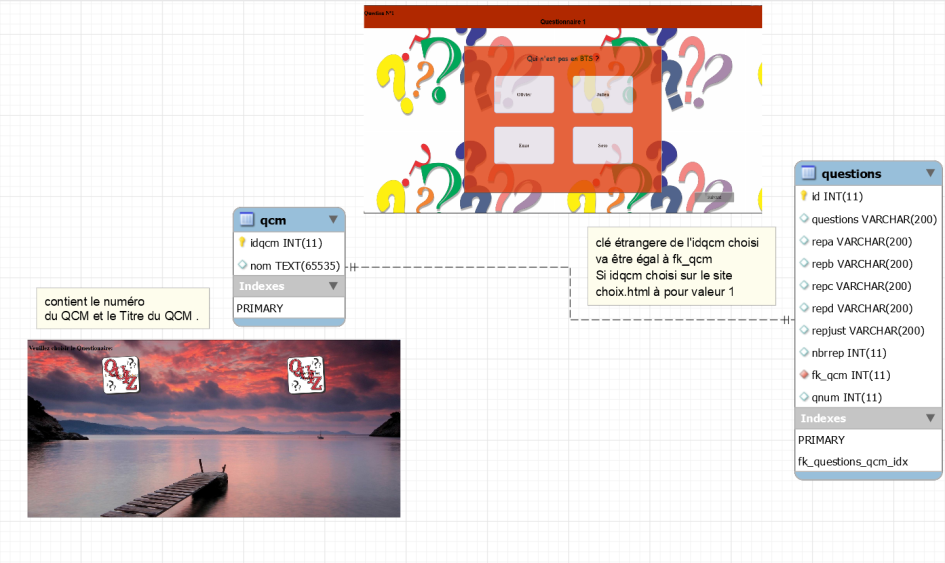
HTML : CSS :



Nous avons déclaré la balise avec l’id « bordure » dans le HTML, suivie du texte qui sera présent dans la balise. Puis nous lui avons déclaré des paramètres tels que le positionnement en hauteur et largeur, la couleur du texte et de l’arrière plan, et « overflow :auto » qui permet d’étirer la bannière vers le haut.

# Modélisation des tables utilisées :

## Partie stockage des QCM :



La table « qcm » est composée de 2 colonnes : « idqcm » qui est le numéro unique du qcm ; et « nom » qui est une variable texte qui contient le nom du qcm.

La table « questions » quand à elle est composée du champ « questions » qui contient les questions du qcm, et une colonne pour chacune des 4 réponses possibles.

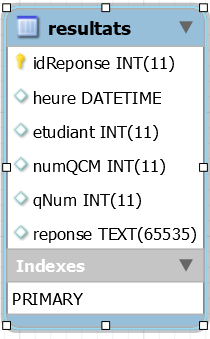
« **repjust** » contient les numéros (a, b, c, ou d) des réponses exactes.

Exemple : a,d. Une comparaison avec la réponse de l’éleve sera faite en comparant avec la table « resultats » de la base de données.

« **qnum** » permet de numéroter les questions dans le formulaire

« **nbrep** » est une valeur entière qui permettra de savoir le nombre de réponses possibles. Cela permettra par exemple de faire une question avec 2 ou 3 réponses possibles et donc un affichage différent de la page selon le nombre de réponses.

## Résultats :



La base de donnée resultats est composé des champs :

« **heure** » contient l’heure de l’enregistrement

« **etudiant**» contient l’id de l’etudiant , clé étrangére qui sera lié au champ de l’id des Etudiants dans la table Etudiant .

« **numQCM**» numéro du QCM

« **qNum** » Numéro de la question

« **reponse** » les réponse choisi pas l’etudiant

Une connection entre les table « resulats» et « question» devra être faite .

Cela permetra de faire une comparaison entre les résultat mis par l’éleve présent dans **reponse** et **repjust** (qui contient les réponses juste).

A la fin cela permetra une notation des éléves qui sera afficher sous forme de tableau a la fin de la Session .