

| | |
|--|---|
| 420-355-LI - Programmation Web Cégep Limoilou Département d'Informatique Automne 2023 | Tp1 (10 %) Jeu-Questionnaire (comme un quiz) |
|--|---|

Objectifs

- Construire une petite application Web, client, en JavaScript offrant une interface HTML et CSS, dont tout le traitement se fait en JavaScript.
- Séparer correctement les différents composants de votre application (html, css, code dans des fonctions et objets).
- Valider correctement le fonctionnement de l'application (Structures de données, navigation, résultats ...).

Lecture

- Je mets ici des références au sujet des formulaires HTML :
 - Formulaire HTML :
<https://developer.mozilla.org/fr/docs/Learn/Forms>
ET
https://www.w3schools.com/html/html_forms.asp
 - Pour valider une partie des valeurs d'un formulaire avec du HTML (faire attention à la compatibilité avec votre browser) :
https://developer.mozilla.org/fr/docs/Learn/Forms/Form_validation
Et
https://www.w3schools.com/js/js_validation.asp

Conditions et remise

- Le travail se fait **obligatoirement** en équipe de 2 (3 s'il y a une personne seule, elle sera placée aléatoirement par le professeur).
- Vous avez 3 semaines pour faire et remettre ce travail à partir de sa date de réception.

Agenda – à mettre à votre agenda

Date de réception :

Groupe 3 - 19 septembre

Groupe 1 - 20 septembre

Groupe 2 - 22 septembre

Semaine 1 - Vérification d'avancement par le prof :

Groupe 1 - le 25 septembre entre 11h et 14h

Groupe 2 - le 25 septembre entre 16h et 18h

Groupe 3 – le 26 septembre entre 8h et 11h

- Prise en note, par votre professeur, de l'avancement de votre travail. La présence des équipes est obligatoire. Cette vérification sera notée.

Semaine 2 - Vérification d'avancement par le prof :

Groupe 1 - le 2 octobre entre 11h et 14h

Groupe 2 - le 2 octobre entre 16h et 18h

Groupe 3 – le 3 octobre entre 8h et 11h

- Prise en note, par votre professeur, de l'avancement de votre travail. La présence des équipes est obligatoire. Cette vérification sera notée.

Semaine 2 - Remise du fichier JS de questions :

Groupe 1 - le 28 septembre avant 17h

Groupe 2 - le 28 septembre avant 17h

Groupe 3 – le 28 septembre avant 17h

- Remettre le fichier final JS « **questionJSON.js** » de questions produit dans ce travail. Renommez le fichier comme suit « **Noms des équipiers – questionJSON.js** » avant de le remettre. Copiez le fichier renommé dans le dossier du « **Tp 1 - JSON** » selon votre groupe. Le dossier « Tp 1 - JSON » est dans le dépôt sur le réseau du département dans le dossier « 420-355-A23 ».

Semaine 3 - Remise du code :

Groupe 1 - le 10 octobre avant 17h

Groupe 2 - le 11 octobre avant 17h

Groupe 3 – le 13 octobre avant 17h

- Remettre le projet « PhpStorm » et seulement le projet « **Noms des équipiers – Tp 1-Questionnaire – A23** » produit dans ce travail pratique, dans le dossier du « **Tp 1** » selon votre groupe. Le dossier « Tp 1 » est dans le dépôt sur le réseau du département dans le dossier « 420-355-A23 ».

- NOTE IMPORTANTE :

Un projet « PHPStorm » de départ pour ce travail est disponible dans le dossier de ce travail sur le réseau.

Mise en situation

Réalisez une petite application qui permet de jouer à un jeu-questionnaire (quiz). Votre jeu-questionnaire sera basé sur un ensemble d'au moins 15 vraies questions, cet ensemble sera vos données d'entrée dans le programme.

Au départ, vous affichez le but du jeu à l'utilisateur et vous permettez à l'utilisateur de jouer. Vous choisissez 5 questions aléatoirement dans votre ensemble de questions, ce qui constituera votre questionnaire. Par la suite, vous présentez une question à la fois, lorsque la question posée est répondue, vous présentez sa correction et ainsi de suite pour les 5 questions. Lorsque les 5 questions ont été répondues, vous présentez à l'utilisateur ses points (sa note finale) et un commentaire personnalisé d'encouragement. Dans ce dernier affichage, vous permettez aussi à l'utilisateur de rejouer avec 5 nouvelles questions prises aléatoirement sans remise dans votre « pool ». À tout moment il est possible d'abandonner, à ce moment vous présentez le résultat accumulé jusque-là.

Comme dit précédemment, les questions sont choisies aléatoirement sans remise. Ce qui veut dire que lorsqu'une question est choisie pour un questionnaire de 5 questions, elle ne peut être redemandée dans le même questionnaire. Il est important aussi que chaque question soit posée au moins une fois après avoir répondu à quelques questionnaires.

Contraintes au niveau des données

- **Chaque semaine**, selon l'agenda proposé ci-haut, votre professeur vérifiera au prêt de chaque équipe l'avancement de votre projet. Ce qui est important, c'est d'être présent (tous les membres de l'équipe) et de répondre aux questions de votre enseignant.
- Votre projet de départ, copiez le projet « **Noms des équipiers – Tp 1-Questionnaire – A23** » qui est fourni dans le dossier du travail « Tp 1 » et renommez-le correctement.
- Le code HTML de départ est réduit à son minimum. Ce code HTML est dans le fichier « **tp1-Questionnaire.html** » à la racine du projet.

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>TP 1 - Questionnaire - A23</title>
  <link href="css/tp1-questionnaire.css" rel="stylesheet">
</head>
<body>

<div id="zoneDeDonnees">

  <!-- Le HTML qui doit venir ici, doit être généré par programmation en JS! -->

</div>

  <!-- Insérez vos balises <script> ici -->
</body>
</html>
```

- Toute la logique est dans des fichiers et/ou classes JavaScript. Utilisez des structures de données, des fonctions et des classes pour gérer vos questions et votre questionnaire.
- Utilisez les CSS pour mettre l'accent sur les bonnes et mauvaises réponses. Mettre vos styles dans le fichier « tp1-Questionnaire.css » dans le dossier « css ».
- **Un fichier JavaScript indépendant** doit être utilisé pour contenir, un tableau associatif de format JSON, de toutes les questions. Cet objet JSON sera utilisé pour remplir un tableau **indexé** d'objets « Question » au démarrage de votre application. C'est avec ce dernier que votre application devra travailler, pas avec l'objet JSON. Donc on veut :
 - Une classe « Question » doit être utilisée pour contenir l'information pour une question.

IMPORTANT : Cette classe doit posséder tous les attributs et méthodes nécessaires pour gérer l'information concernant une question. Il faut bien réfléchir ici, car plus la classe (les objets) « Question » est bien conçue, plus elle va être utile.

- Un tableau indexé d'objet « Question » pour contenir toutes les questions du fichier de données JSON. On prend l'information du tableau associatif JSON (fichier questionsJSON.js) et on fabrique des objets « Question » que l'on place dans un tableau index (à [])
- Une classe « QuestionnaireQuiz » doit être utilisée pour contenir les 5 questions du quiz (jeu-questionnaire) courant. Il contient les 5 objets « Question » choisis aléatoirement sans remises à partir du tableau indexé d'objets « Question » précédemment créés.

Cette classe doit posséder tous les attributs et méthodes nécessaires pour gérer l'ajout

et la disponibilité des questions du questionnaire et la note s'y rattachant. Il faut bien réfléchir ici, car plus la classe (les objets) « QuestionnaireQuiz » est bien conçue, plus elle va être utile.

- Les textes et les fenêtres présentés **ci-après** représentent l'affichage et le fonctionnement minimal attendu, si vous le voulez, vous pouvez faire mieux, mais il ne faut pas altérer le fonctionnement demandé.

Contraintes au niveau du fonctionnement (étapes de développement)

1. Fichier de questions

Le fichier principal de questions doit porter obligatoirement le nom de « **questionJSON.js** ». Il est au format JSON (tableau associatif JavaScript) et contient au minimum la définition de 15 questions réelles.

Une question, dans ce fichier, doit être définie selon les informations **minimums** suivantes :

- Le texte de la question.
- Le texte de 5 réponses (5 obligatoires).
- La solution, la bonne réponse.
- Le nombre de points pour une bonne réponse.

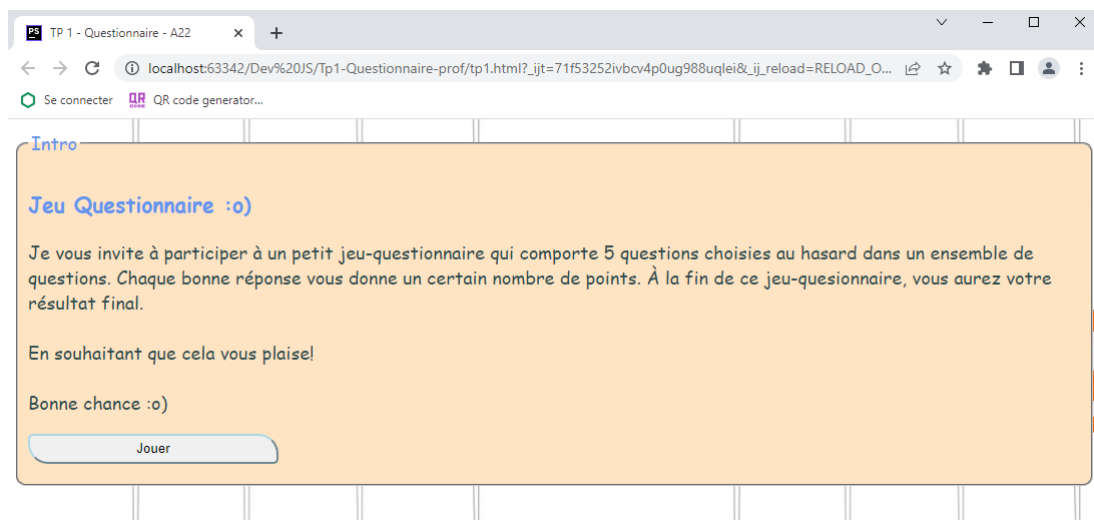
IMPORTANT : Les questions doivent être réelles et toutes composées en relation avec la matière du cours. Sur la matière JavaScript vue dans les 6 premières semaines de cours (à partir des notes de cours, des formatifs et des sites Web identifiés dans le cours).

2. Chargement des questions dans l'application

Au départ de l'application, les questions contenues dans le tableau associatif JSON (fichier questionsJSON.js) doivent être parcourues pour créer des objets « Question » qui seront placés dans un tableau indexé. Cette structure servira de tableau, « pool » de questions par la suite. Ce n'est pas le questionnaire.

3. Premier affichage de l'application

Le premier affichage de votre application devrait présenter à l'utilisateur vos règles de fonctionnement et la possibilité de jouer. Voici l'exemple :



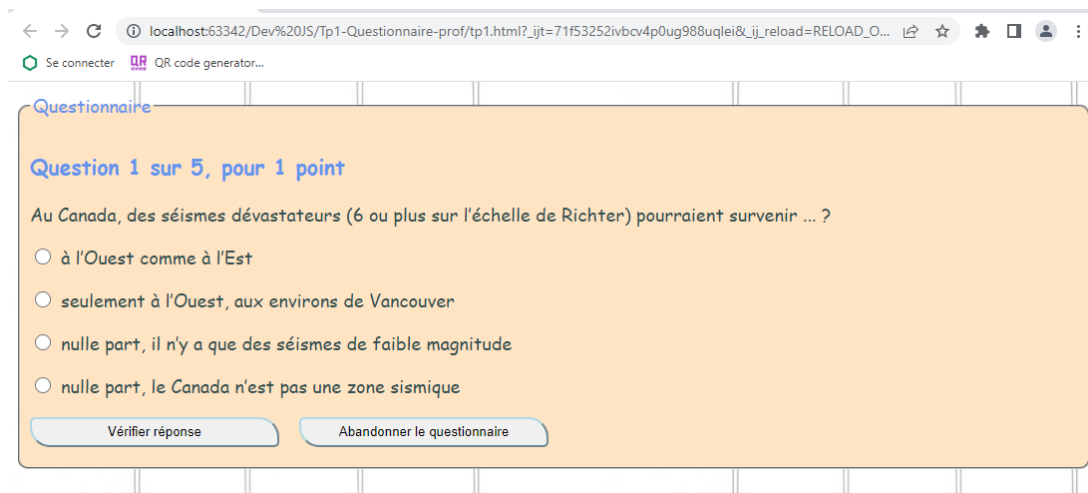
Ici le bouton « Jouer » permet de commencer.

4. Choix des questions du questionnaire (quiz)

Si le bouton « Jouer » est cliqué, il faut choisir aléatoirement sans remise, parmi l'ensemble des questions du tableau d'objet « Question », 5 questions pour fabriquer un objet « QuestionnaireQuiz ». Il ne faut pas que les questions se répètent dans le même questionnaire (quiz).

5. Présentation d'une question

Maintenant que l'on a un objet « QuestionnaireQuiz », il faut poser les questions. Chaque question est présentée une à la fois à l'utilisateur. On doit voir le numéro de question à laquelle on est rendu à répondre et comment passer d'une question à l'autre. La navigation doit être simple et claire. On ne peut pas revenir en arrière. Voici un exemple de présentation :



Questionnaire

Question 1 sur 5, pour 1 point

Au Canada, des séismes dévastateurs (6 ou plus sur l'échelle de Richter) pourraient survenir ... ?

- ☐ à l'Ouest comme à l'Est
- ☐ seulement à l'Ouest, aux environs de Vancouver
- ☐ nulle part, il n'y a que des séismes de faible magnitude
- ☐ nulle part, le Canada n'est pas une zone sismique

Vérifier réponse Abandonner le questionnaire

Ici le bouton « Vérifier réponse » permet de montrer à l'utilisateur la correction de la question. Le bouton « Abandonner le questionnaire » termine et présente le résultat ou note partiel à l'utilisateur. Cette dernière éventualité sera présentée ci-après.

Il est très important que l'on puisse donner qu'une seule réponse, étant donné que nous avons des cases à choix (boutons radio).

6. Bouton « Vérifier réponse »

Lorsque ce bouton est cliqué, la correction de la question est présentée. Cela peut ressembler à ceci :

Si je n'ai pas coché de réponse :

localhost:63342 indique
Vous devez répondre à la question ou abandonner !

OK

Questionnaire

Question 1 sur 5, pour 1 point

Au Canada, des séismes dévastateurs (6 ou plus sur l'échelle de Richter) pourraient survenir ... ?

- ☐ à l'Ouest comme à l'Est
- ☐ seulement à l'Ouest, aux environs de Vancouver
- ☐ nulle part, il n'y a que des séismes de faible magnitude
- ☐ nulle part, le Canada n'est pas une zone sismique

Vérifier réponse Abandonner le questionnaire

Si vous avez coché la bonne réponse :

Questionnaire

Question 1 sur 5, pour 1 point

Au Canada, des séismes dévastateurs (6 ou plus sur l'échelle de Richter) pourraient survenir ... ?

- ☒ à l'Ouest comme à l'Est ✓
- ☐ seulement à l'Ouest, aux environs de Vancouver
- ☐ nulle part, il n'y a que des séismes de faible magnitude
- ☐ nulle part, le Canada n'est pas une zone sismique

Passer à la prochaine question Abandonner le questionnaire

La correction met en **évidence la bonne réponse** (CSS) et me permet de passer à la question suivante.

IMPORTANT : Remarquez que je ne peux plus sélectionner de réponses.

Si vous avez coché la mauvaise réponse :

Questionnaire

Question 2 sur 5, pour 1 point

Quel est l'âge de l'Univers ?

- ☐ 6000 ans
- ☒ 4,6 milliards d'années
- ☐ 13,7 milliards d'années ✓
- ☐ L'univers est éternelle et existe depuis toujours.

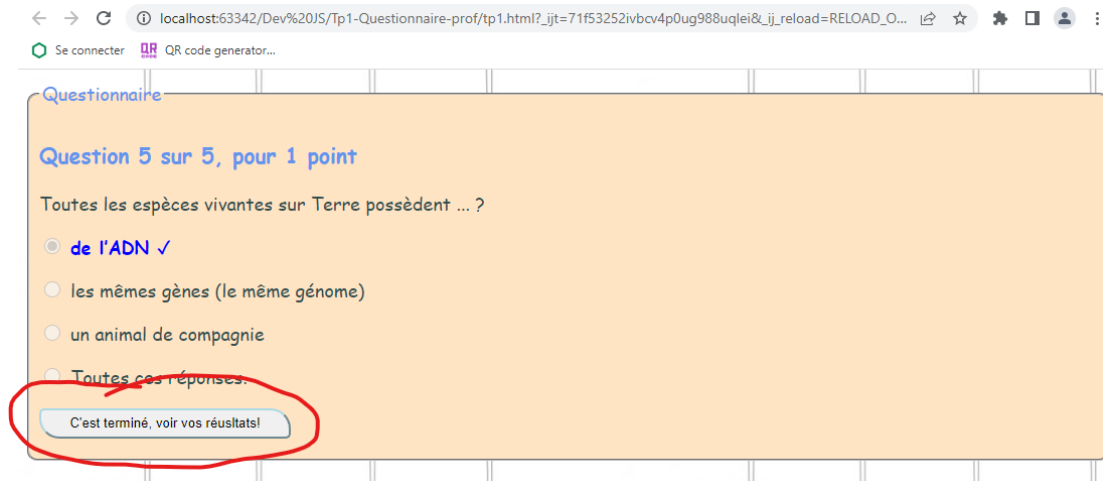
Passer à la prochaine question Abandonner le questionnaire

La correction me montre que j'ai la ~~mauvaise réponse~~, en mettant en **évidence la bonne réponse** et me permet de passer à la question suivante ou d'abandonner.

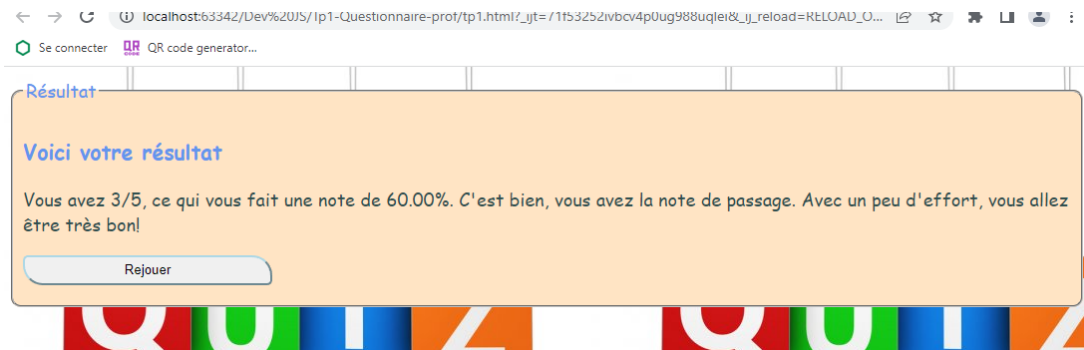
IMPORTANT : Remarquez que je ne peux plus sélectionner, là aussi, de réponses.

7. Fin du questionnaire sans abandon

Lorsque vous affichez la correction de la **dernière question**, remarquez ici que le bouton « Abandonner le questionnaire » a disparu, car de toute façon le questionnaire est fini. Maintenant si je clique sur le bouton « C'est terminé, voir vos résultats! », je vais voir les résultats.



Voici la présentation des résultats finaux pour un questionnaire.



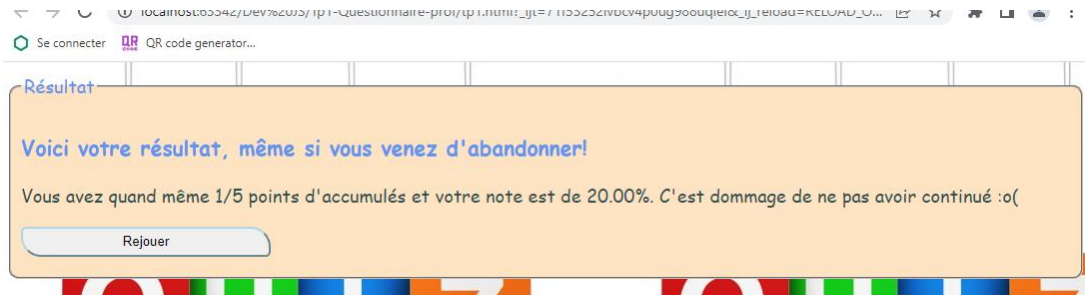
Vous affichez le nombre de points sur le nombre total qu'on aurait pu avoir, la note en pourcentage et un commentaire **d'encouragement personnalisé** selon la note en pourcentage. Voici la liste des différents intervalles permettant d'émettre des **messages personnalisés d'encouragement croissant**.

- De 0% à 29.99%
- De 30% à 59.00%
- De 60% à 69.99%
- De 70% à 84.99%
- De 85% à 94.99%
- De 95% à 100%

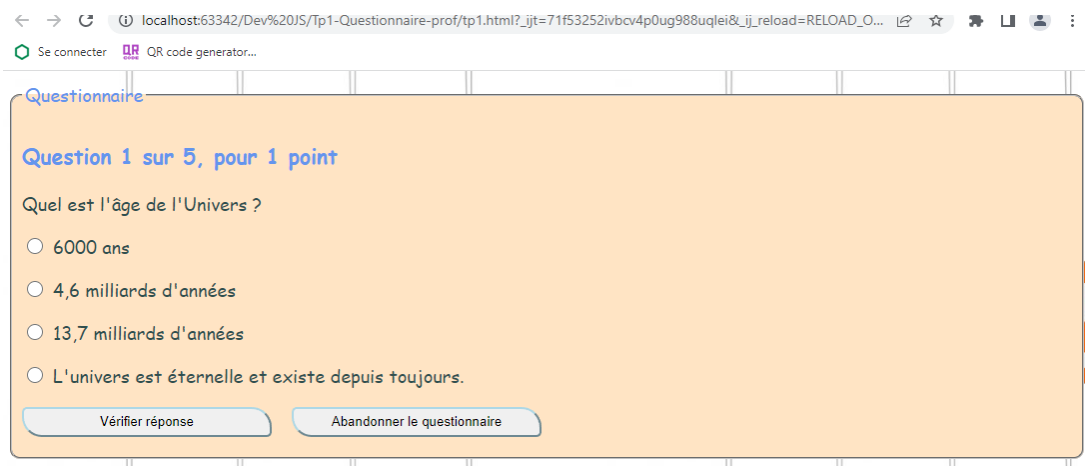
À vous d'écrire les messages d'encouragement voulus selon l'intervalle

8. Bouton « Abandonner le questionnaire »

Lorsque ce bouton est cliqué, vous présentez un résultat partiel à l'utilisateur. On doit comprendre que l'on a abandonné, sans vraiment avoir terminé. Un peu comme ceci :



Remarquez, tout comme quand je termine normalement, je peux « Rejouer ». Si je rejoue, j'ai de nouveau 5 nouvelles questions et je recommence le jeu directement, comme ceci :



Grille de correction

La grille de correction vous sera fournie la semaine prochaine.

Conseils

- Vous êtes une équipe, c'est important de travailler en équipe! Respect, écoute et collaboration sont de mise.
- Prendre le temps de bien comprendre l'ensemble du problème.
- Dresser une liste détaillée de petites actions ou étapes de développement à faire et ordonner cette dernière.
- Développer une action à la fois et tester correctement son fonctionnement.
- Ne pas essayer de développer des comportements finaux trop vite, mais passer plutôt par une évolution itérative vers un état final.
- Servez-vous de F12 dans le « browser » pour voir et comprendre vos erreurs.
- Ne surestimez pas votre programmation, programmez à petits pas, prenez le temps de tester, de tracer et de déboguer votre code au fur et à mesure que vous avancez.