Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2021
WEBG2
DBO, DNA, MCD, NRI,
NVS, SDR, TNI

WEBG2 – Développement Web I

Projet 2 - Super quiiiiiizzzzzzz

Table des matières

Échéances

Publication de l'énoncé	vendredi 19 mars à 18h
Remise du projet	vendredi 2 avril à 18h
Défense	semaine du 19 avril

Ce projet compte pour 20% dans votre cote finale de l'UE.

Version du document
19 mars 2021

1 Informations pratiques

Bienvenue dans le deuxième projet de développement WEB. Il vous permettra de mettre en œuvre les concepts vus au cours théorique, pendant vos laboratoires ainsi que d'exercer votre capacité à trouver les bonnes informations sur la toile.

Nous vous demandons de lire attentivement et complètement ce document. Votre professeur respectif vous donnera plus d'information en début de projet.

Git et remise

Votre projet doit être développé sous GIT avec le dépôt que nous vous avons créé pour vous : `webg2-etd/2021/projet2/webg2-projet2-xxxxx` [1].

⚠ Commits réguliers

Vous **devez** faire un *commit* régulier de votre projet, – au minimum **quand c'est explicitement demandé** [a] à la fin de chaque étape – et le *pusher* régulièrement sur GITLAB.

a. **Deux oublis** sont tolérés, pas plus. Tout oubli supplémentaire conduira à la **non correction du projet**.

Même si vous avez des soucis avec GIT, n'essayez en aucun cas de ré-initialiser votre dépôt ou d'en utiliser un autre : demandez à votre enseignant ce qu'il y a lieu de faire. Dans le doute : ne faites pas de push et posez la question !

Évaluation

La cote que nous vous attribuerons dépendra de ce que vous nous remettrez mais également de la **défense** (modifications à apporter au projet ou interrogation de projet). Si cette défense est ratée, votre travail ne sera pas évalué ($\Rightarrow 0$).

Nous attirons votre attention sur certains points qui peuvent faire baisser la note de façon drastique :

- ▷ Code illisible, mal indenté ou mal documenté ;
- ▷ Non respect des consignes de remise (en retard c'est 0/20).

⚠ Coopération vs tricherie

Le projet est un **travail individuel**. Il vous est interdit de copier en tout ou en partie le travail d'un autre étudiant. En cas de copie manifeste, le **copieur et le copié seront sanctionnés**. Toutefois, nous encourageons la collaboration : des échanges sur la compréhension de l'énoncé, sur les algorithmes à mettre en œuvre, la comparaison de pratiques, l'aide au débogage.

Technologies

Ce projet fait appel aux technologies : HTML, CSS et JAVASCRIPT.

- ▷ Vous êtes fortement *encouragé* à utiliser JQUERY.
- ▷ Vous *pouvez* utiliser BOOTSTRAP si vous le voulez.
- ▷ Le projet ne prévoit pas de créer des classes en JAVASCRIPT.

1. où `xxxxx` est votre matricule.

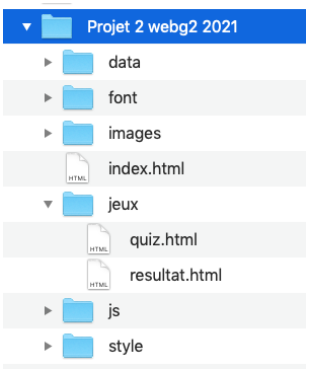
Les ressources

Vous trouverez sur poÉSI les dossiers suivants :

- ▷ `images` : contient l'ensemble des images réparties dans trois sous dossiers.
- ▷ `data` : contient le fichier `quizzes.js` décrivant les quiz ainsi qu'un fichier par quiz reprenant son contenu.

Respect des consignes

Nous allons procéder par étapes. Celle-ci sont placées de telle sorte que la difficulté sera progressive. Évitez la surcharge de « gadgets » inutiles ou qui risquerait de vous faire perdre du temps lors de la conception du projet.



Nous vous demandons d'organiser votre projet avec des dossiers dédiés aux différents types de fichiers.

Nous vous demandons également de soigner l'indentation et de **commenter** toutes vos fonctions JAVASCRIPT de façon rigoureuse (comme en DEV1/2).

N'oubliez pas qu'il est **obligatoire** de commiter votre projet à la fin de chaque étape. Le message doit être celui que nous proposons (ex : `Étape 1.1 : Page d'accueil`).
Si vous désirez réaliser des commits supplémentaires, le message accompagnant doit commencer par le label « perso » + numéro de l'étape et être le plus explicite possible (ex : `perso Étape 2.1: problème formulaire résolu`).

2 Présentation du projet

Description

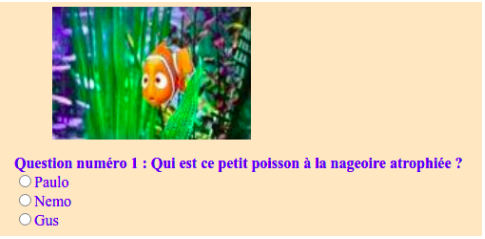
On vous demande de coder un site qui propose des quiz.

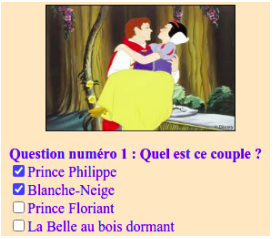
L'application commencera par une page proposant un formulaire de choix du quiz parmi une série.

Après le choix du quiz, un compte à rebours de trois secondes sera lancé avant apparition du quiz.

Suivant le choix deux types de quiz seront proposés. Les quiz « Les animaux marins dans les dessins animés » et « De quel jeu provient cette image ? » sont à sélection unique (une réponse possible). Le quiz « Les couples de dessins animés« demande un

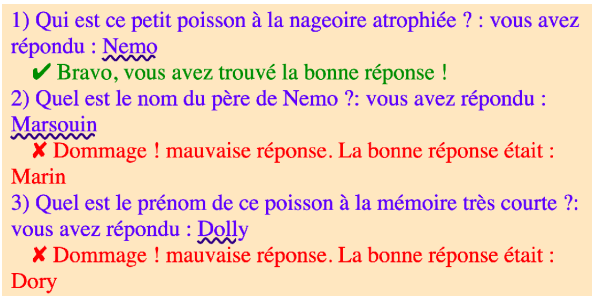
couple de réponses (logique non !). Le dernier quiz contient des questions de tous les types.





Particularité du quiz : le joueur doit répondre à toutes les questions (nous reviendrons là-dessus en temps utile)

Fin du jeu : vous afficherez les bonnes et mauvaises réponses. En cas de mauvaise réponse, la correction apparait à côté.



3 Étape 1.1 : Mise en place

Commençons par la mise en place du projet. Créez une page d'accueil (`index.html`) et placez-y les informations suivantes [2] : un titre, votre nom, votre matricule, votre groupe et le nom de votre professeur.

Dans un dossier `style` créez un fichier `style.css`. Il contiendra toutes les règles communes de mise en forme de votre site. Il doit être correctement indenté. Prenez soin de d'utiliser les bons sélecteurs.

Lorsque c'est fait, faites un commit (`Étape 1.1 : Page d'accueil`) et un push de votre projet.

Rappel : ces commits sont **obligatoires**. En cas de manquement, votre travail ne sera pas corrigé.

4 Étape 1.2 : Comprendre les données

La page d'accueil doit contenir un formulaire qui permet de choisir le quiz qu'on souhaite faire.

Mais ne le faites pas tout de suite ! Nous aimerions que la liste des quiz se **construise de façon dynamique**. Ce qui veut dire que du code JAVASCRIPT remplit la liste (le `<select>`) avec les données fournies dans le fichier `quizzes.js` fourni.

2. Elles devront se trouver sur chaque page

```
const quizzes = {
    "mer": {
        title: "Les animaux marins dans les dessins animés",
        description: "Des poissons et mammifères marins vous sont données. À vous de me donner leur nom.",
    },
    "jeux": {
        title: "De quel jeu provient cette image ?",
        description: "On vous donne l'image, retrouvez le jeu !",
    },
    "couples": {
        title: "Les couples de dessins animés",
        description: "Essayez de reconnaitre ces célèbres couples de dessins animés.",
    },
    "webg2": {
        title: "La matière de Webg2",
        description: "Testez vos connaissances sur le cours de webg2",
    },
};
```

resources/poesi/js/data/quizzes.js

Afin de vérifier que vous savez manipuler ces données, on vous demande à cette étape d'écrire une fonction JAVASCRIPT qui affiche sur la console la liste des identifiants des quiz et le titre correspondant à partir des informations se trouvant dans `quizzes`.

```
mer - Les animaux marins dans les dessins animés
jeux - De quel jeu provient cette image ?
couples - Les couples de dessins animés
webg2 - La matière de Webg2
```

Aide :

- Le `for..in` (cf. slide 367 du cours) vous permet de parcourir tous les *identifiants* (`mer`, `jeux`, `couples` et `webg2`).
- Si la variable `quizId` contient un identifiant, alors `quizzes[quizId]` donne tout l'*objet* associé et `quizzes[quizId].title` donne le titre du quiz.

Pensez à bien placer votre code dans des fichiers se trouvant dans un dossier dédié.

Lorsque vous obtenez la liste sur la console, faites un commit (`Étape 1.2: liste des quiz`) et un push de votre projet.

5 Étape 1.3 : Le formulaire de choix du quiz

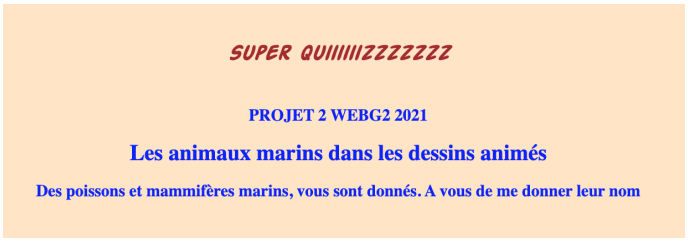
Il est temps d'ajouter le formulaire et de remplir dynamiquement la liste déroulante.

Le bouton "Commencer" transfère l'identifiant du quiz choisi à la page `quiz.html` (que vous coderez à l'étape suivante).

Lorsque le menu du formulaire s'affiche correctement et que votre code est bien indenté et documenté, faites un commit (`Étape 1.3: choix du quiz`) et un push de votre projet.

6 Étape 2.1 : Page du quiz - création de l'entête

Le joueur a choisi un quiz, et il arrive sur la page `quiz.html` qui détaille le quiz : titre et description.



Soignez votre page de jeu afin qu'elle ressemble à la page d'accueil en

- ▷ copiant les éléments HTML communs aux deux pages ;
- ▷ réutilisant le CSS de la première page (pas de copie).

Dans le fichier `pageQuiz.js` créez le code nécessaire pour afficher les informations demandées. Il faudra bien sûr récupérer l'id du quiz choisi dans la première page (cf. slide 327 du cours).

Pour rappel, pour insérer une valeur dans une page vous pouvez

1. *réserver* un emplacement dans la page HTML via le code `<span id="unID"></span>` ;
2. y insérer une valeur via le code `$("#unID").text(uneValeur)`.

À ce stade, la page doit afficher le titre et la description du quiz. Soignez et documentez votre code, faites un commit (`Étape 2.1 : création de l'entête`) et un push de votre projet.

7 Étape 2.2 : Page du quiz - création d'un compte-à-rebours

Le quiz ne se lance pas directement. Un compte à rebours est lancé 3...2...1...0 après quoi il disparait et le quiz s'affiche.

Dans le fichier `timer.js`, vous écrivez le code qui gère le timer. Remarque : vous serez peut-être tenté de chercher un code tout fait pour ce timer. C'est très bien de faire des recherches et de s'inspirer d'un code trouvé mais il faut le comprendre. Lors de la défense, il vous sera peut-être demandé de l'expliquer et/ou de le modifier.

À ce stade, la page doit afficher les instructions du quiz et un compte-à-rebours. Faites un commit (`Étape 2.2: compte-à-rebours`) et un push de votre projet.

8 Étape 3.1 : Parcourir le quiz

Lorsque le compte-à-rebours est arrivé à 0, le quiz choisi est affiché.

Structure des fichiers de données

Vous trouverez sur `poÉSI` un dossier **data** comprenant un fichier JAVASCRIPT par quiz reprenant le détail de ce quiz.

En voici un extrait :

```
const questions_jeux = [
    {
        id: "id1",
        question: "De quel jeu est issue cette scène ?",
        reponses: [
            "Final Fantasy X",
            "Kingdom Hearts",
            "Dragon Quest"
        ],
        bonneReponses: [0],
        image: "jv/img1.jpg"
    },
    {
        id: "id2",
        question: "De quelle série de jeu vient ce screen ?",
```

resources/poesi/js/data/jeux.js

Ils ont tout la même structure. Chaque élément du tableau décrit une question du quiz :

- ▷ id : identifiant unique ;
- ▷ question : la question posée ;
- ▷ reponses : tableau comprenant les réponses possibles ;
- ▷ bonneReponse : numéro de la réponse [3] [4] ;
- ▷ image : lien relatif du chemin de l'image.

Utiliser les données du fichier

Le problème qui doit nous occuper à présent c'est de pouvoir sélectionner le bon quiz. Pour ce faire :

1. Modifiez le fichier `quizzes.js` afin d'y inclure les variables contenant les quiz. Voici ce qu'il devient :

```
const quizzes = {
    "mer": {
        title: "Les animaux marins dans les dessins animés",
        description: "Des poissons et mammifères marins vous sont données. À vous de me donner leur
        ↪ nom.",
        data: questions_mer,
    },
    "jeux": {
        title: "De quel jeu provient cette image ?",
        description: "On vous donne l'image, retrouvez le jeu !",
        data: questions_jeux,
    },
    "couples": {
        title: "Les couples de dessins animés",
        description: "Essayez de reconnaitre ces célèbres couples de dessins animés.",
        data: questions_couples,
    },
    "webg2": {
        title: "La matière de Webg2",
        description: "Testez vos connaissances sur le cours de webg2",
        data: questions_webg2,
    },
};
```

2. Dans le fichier HTML, incluez tous les fichiers du dossier `data` (`quizzes.js` en dernier).
3. À présent, si la variable `quizId` contient l'id du quiz choisi alors le code suivant sélectionne les données du bon questionnaire : `quizzes[quizId].data`.

3. Tableau comprenant un entier pour les quiz mer et jeux mais deux entiers pour couples.
4. Les réponses sont numérotées à partir de 0. Si la bonne réponse est 1, c'est la deuxième qui est désignée.

Afin de vérifier que vous arrivez à choisir le bon questionnaire et que vous comprenez bien sa structure, nous vous demandons d'écrire une fonction qui affiche la liste des questions dans la console.

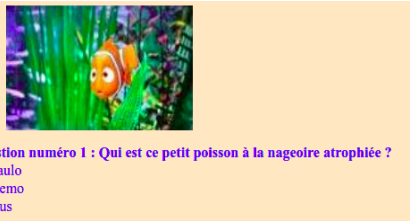
Par exemple, pour le questionnaire sur la mer, on voudrait ceci sur la console.

```
Qui est ce petit poisson à la nageoire atrophiée ?
Quel est le nom du père de Nemo ?
Quel est le prénom de ce poisson à la mémoire très courte ?
Comment se nomme ce gentil requin que l'on retrouve dans le film ' Gang de Requins
Quel est le nom de cette baleine blanche sortie de l'imagination d'Herman Melville ?
Quel est le nom ce célèbre dauphin ?
Quel est le prénom de ce poisson rouge, qui pousse le bouchon un peu loin ?
```

Lorsque vous obtenez sur la console l'affichage des questions du quiz choisi, faites un commit (`Étape 3.1 : Parcourir quiz`) et un push de votre projet.

9 Étape 3.2 : Afficher le quiz

Nous allons enfin concevoir le quiz. Pour le moment, nous allons nous concentrer sur les quiz **mer.js** et **jeux.js** qui sont des quiz à réponse unique. Nous reviendrons dans une étape ultérieure sur **couples.js** qui contient des questions à choix multiple.



Le quiz est un formulaire qui va reprendre les informations se trouvant dans le quiz sélectionné à l'étape précédente.

Écrivez une fonction qui affiche les images, questions et propositions de réponses. Le formulaire utilise des radios boutons.

À la fin du questionnaire, placez un bouton "Vérification" qui envoie les résultats à la page `resultats.html` que vous écrirez lors d'une prochaine étape.

Lorsque le formulaire contenant le quiz choisi s'affiche correctement, faites un commit (`Étape 3.2 : Afficher quiz`) et un push de votre projet.

10 Étape 3.3 : Correction du quiz

1) Qui est ce petit poisson à la nageoire atrophiée ? : vous avez répondu : Nemo
✔ Bravo, vous avez trouvé la bonne réponse !
2) Quel est le nom du père de Nemo ?: vous avez répondu : Marsouin
✘ Dommage ! mauvaise réponse. La bonne réponse était : Marin
3) Quel est le prénom de ce poisson à la mémoire très courte ?: vous avez répondu : Dolly
✘ Dommage ! mauvaise réponse. La bonne réponse était : Dory

On arrive à la fin du jeu, il faut afficher la correction. On vous demande d'écrire les infos suivantes pour chaque question :

- ▷ l'intitulé de la question ;
- ▷ la réponse donnée ;
- ▷ en vert si c'est la bonne réponse « Bravo, vous avez trouvé la bonne réponse ! »
- ▷ en rouge si c'est la mauvaise réponse « Dommage ! mauvaise réponse. La bonne réponse était : *la bonne réponse* »

Aide :

- ▷ Cette page recevra les réponses de l'utilisateur mais elle devra aussi savoir de quel quiz il s'agit. Vous pouvez passer des informations supplémentaires grâce à `<input type="hidden"...`

Faites un commit (`Étape 3.3 : Correction Radio`) et un push de votre projet.

11 Étape 4.1 : Questions à réponses multiples

Ce que vous avez codé ne fonctionne pas pour l'instant pour le troisième quiz (les couples) puisqu'il y a 2 réponses à cocher pour chaque question.

Nous vous demandons d'adapter votre code pour pouvoir gérer des quiz avec plusieurs réponses possibles.

Votre code doit pouvoir fonctionner avec des quiz mixtes qui contiennent des questions à réponse unique ainsi que des questions à plusieurs réponses (pas forçément 2).

Question numéro 1 : Quel est ce couple ?

- ☑ Prince Philippe
- ☑ Blanche-Neige
- ☐ Prince Floriant
- ☐ La Belle au bois dormant

Nous vous laissons réfléchir à comment faire ça au mieux ;)

Aide :

- ▷ Vous savez récupérer la valeur d'un paramètre de l'URL. S'il est présent plusieurs fois – ce qui risque d'être le cas en cas de réponse multiple – vous pouvez utiliser la fonction `getAll` pour récupérer toutes les valeurs dans un tableau.

Faites un commit (`Étape 4.1 : Questions à réponses multiples`) et un push de votre projet.

12 Étape 4.2 : Valider le quiz

On ne peut pas *"ne pas répondre"*. Avant d'envoyer le formulaire, il faut vérifier que l'utilisateur a bien répondu à toutes les questions. Plus exactement, vous devez vérifier que le nombre de réponses données à chaque question correspond bien au nombre de réponses attendues.

Si ce n'est pas le cas, un message clair doit accompagner la ou les questions qui posent problème et on ne doit pas envoyer le formulaire.

Aide :

- ▷ Le formulaire ne sera envoyé à la page suivante que si la fonction appelée lors du *click* retourne *vrai*.

Faites un commit (`Étape 4.2 : validation`) et un push de votre projet.

13 Remise finale

Bravo ! Votre projet est terminé. Prenez le temps de repasser dessus :

- ▷ Est-ce que tout est bien indenté, lisible, documenté ?
- ▷ Est-il possible de créer des fonctions pour éviter des répétitions de code ?
- ▷ Est-ce que ça fonctionne encore si on change un peu les données : un quiz supplémentaire, un nombre différent de réponses… ?

Lorsque vous êtes satisfait, faites le dernier commit (`Remise finale`) et un push de votre projet.