Développement d’une application en Blockly

Nom du développeur 1 : Sophie Crowley			
Nom du développeur 2 : —			
Titre de l’application : Jeu de Vocabulaire Français			
Type d’application	Jeu X	Exerciseur ☐	Utilitaire ☐

Présentation de l’application

Vous pouvez trouver mon application en ligne ici : https://sophieccc.github.io/js-game/

Le jeu de Vocabulaire Français est un jeu dans lequel un mot anglais est affiché et le joueur doit deviner la traduction française correcte. Le joueur peut choisir entre trois niveaux de difficulté, ainsi que deux modes de jeu. Le premier mode de jeu affiche trois mots français et le joueur choisit celui qu'il pense être correct. Le deuxième mode de jeu permet au joueur de taper ce qu'il pense être la bonne traduction dans une zone de saisie. Une fois que le joueur a fait son choix, le programme lui donne la bonne réponse et lui indique s'il a été correct ou incorrect. Le jeu dure cinq questions et donne au joueur un score total à la fin, ainsi que la possibilité de rejouer.

L'intérêt de ce jeu est de faire plaisir aux joueurs tout en les aidant à apprendre de nouveaux mots français. De cette façon, on pourrait l'appeler soit un jeu, soit un exerciseur. Il est donc à la fois éducatif et ludique. On peut commencer avec la difficulté la plus facile et choisir le mode de jeu "options" et augmenter la difficulté à partir de là. Le mode de jeu "saisie de texte" est évidemment plus difficile que le mode "options" car il ne donne aucune piste au joueur. Le joueur est libre de choisir comment il veut jouer et dans quelle mesure il veut se mettre au défi. Les questions sont également aléatoires, de sorte qu'on ne peut pas apprendre par cœur l'ordre des questions.

Copie d’écran du code de l’application

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>JS Game</title>
    <meta name="description" content="JS Game">
    <meta name="author" content="Sophie Crowley">

    <link rel="icon" type="image/png" href="src/french-flag.png">
    <link rel="stylesheet" href="src/styles.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

    <script src="src/index.js"></script>
</head>
<body>
    <div class="centred">
        <p id="statement" class="h4">Pick a difficulty & game mode and begin!</p>
        <div id="start">
            <div>
                <p class="choice h5">Difficulty:</p>
                <select id="difficulty">
                <option>easy</option>
                <option>medium</option>
                <option>hard</option>
              </select>
            </div>
            <div>
                <p class="choice h5">Game Mode:</p>
                <select id="game-choice">
                <option>choose from options</option>
                <option>type your answer</option>
              </select>
            </div>
            <button class="btn btn-primary btn-lg" onclick="startGame()">Start</button>
        </div>
        <div id="options-game" class="originally-hidden">
            <button class="btn btn-primary" id="one" onclick="answerQuestion(0)"></button>
            <button class="btn btn-primary" id="two" onclick="answerQuestion(1)"></button>
            <button class="btn btn-primary" id="three" onclick="answerQuestion(2)"></button>
        </div>
        <div id="text-game" class="originally-hidden">
            <input class="form-control" type="text" id="text-input" />
            <button class="btn btn-primary" id="text-button" onclick="sendTextAnswer()">submit</button>
        </div>
        <button class="btn btn-primary" id="next" onclick="askQuestion()">Next</button>
    </div>
</body>

</html>
```

styles.css

```
styles.css > #statement
body {
    text-align: center;
    background-image: url('export.png');
    background-size: cover;
}

a:any-link {
    color: navy;
}

.centred {
    position: fixed;
    top: 35%;
    left: 50%;
    transform: translate(-50%, -50%);
}

.choice {
    display: inline-block;
}

.originally-hidden {
    display: none;
}

#statement {
    white-space: pre;
}

#text-input {
    margin-bottom: 10px;
}

#next {
    display: none;
```

input.txt

```
src > input.txt
1   easy,book,livre,livre,ouvrage,cahier
2   easy,house,maison,chambre,habitation,maison
3   easy,happy,heureux,triste,heureux,fière
4   easy,small,petit,grand,faible,petit
5   easy,pen,stylo,crayon,stylo,papier
6   easy,door,porte,porte,mur,chaise
7   easy,to be,être,avoir,aller,être
8   easy,tree,arbre,fleur,arbre,herbe
9   easy,car,voiture,voiture,vélo,avion
10  easy,from,de,sans,de,avec
11  medium,teacher,enseignant,étudiant,enseignant,apprentissage
12  medium,wise,sage,vif,sage,sensé
13  medium,university,université,université,collège,lycée
14  medium,computer,ordinateur,portable,ordinateur,technique
15  medium,screen,écran,soleil,papier,écran
16  medium,to want,vouloir,vouloir,aimer,devoir
17  medium,to have,avoir,devoir,avoir,aller
18  medium,since,depuis,après,avant,depuis
19  medium,curtain,rideau,volet,tapis,rideau
20  medium,before,avant,avant,après,pendant
21  hard,to accommodate,loger,accommoder,recevoir,loger
22  hard,character,personnage,caractère,personnage,identité
23  hard,squirrel,écureuil,renard,hérisson,écureuil
24  hard,travel,voyage,voyage,travail,marche
25  hard,coding language,langage,langue,langage,programme
26  hard,scale,échelle,échelle,importance,adage
27  hard,cog,rouage,appareil,fer,rouage
28  hard,challenge,défi,échec,défi,tiers
29  hard,encrypted,chiffré,mélangé,chiffré,caché
30  hard,heat,chaleur,chaleur,humidité,chaud
```

index.js

```
src > JS index.js > processTextFile
1   var currentQuestion;
2   var score;
3   var questions = [];
4   var gameMode;
5
6   /* This is called for initial set-up each time you play the game.
7   It retrieves the difficulty and game mode that the player has chosen
8   and acts accordingly */
9   function startGame() {
10      currentQuestion = 0;
11      score = 0;
12      gameMode = document.getElementById("game-choice").selectedIndex;
13      var result = document.getElementById("difficulty");
14      var difficulty = result.options[result.selectedIndex].text;
15      var allQuestions = processTextFile(difficulty);
16      getRandomisedQuestions(allQuestions);
17      var introduction = document.getElementById("start");
18      introduction.style.display = "none";
19      askQuestion();
20  }
21
22  /* This makes sure that there are no repeat questions within games,
23  while also randomising which questions are picked so that there is
24  variety in the questions asked and in their order */
25  function getRandomisedQuestions(allQuestions) {
26      questions = [];
27      numbers = []
28      while (numbers.length < 5) {
29          var rand = Math.floor(Math.random() * allQuestions.length);
30          if (numbers.indexOf(rand) === -1) {
31              numbers.push(rand);
32          }
33      }
34      for (var i = 0; i < 5; i++) {
35          questions.push(allQuestions[numbers[i]]);
```

```
    }
}

/* This retrieves and processes the text file containing the questions.
If the question is of the required difficulty level, it is pushed
to an array of question objects */
function processTextFile(difficulty) {
    var allQuestions = [];
    var file = new XMLHttpRequest();
    file.open("GET", "src/input.txt", false);
    file.overrideMimeType('text/plain; charset=UTF-8');
    file.onreadystatechange = function() {
        if (file.readyState === 4) {
            var lines = file.responseText.split('\n');
            for (var i = 0; i < lines.length; i++) {
                if (lines[i].startsWith(difficulty)) {
                    var words = lines[i].split(',');
                    allQuestions.push({
                        english: words[1],
                        french_options: [words[3], words[4], words[5]],
                        answer: words[2]
                    })
                }
            }
        }
    }
    file.send();
    return allQuestions;
}

/* This sets up and displays the question-and-answer part of the game.
In the case of the options mode, the 3 options are loaded into their respective buttons.
If all of the questions have been asked, the game is ended. Based on the players score,
different commentary is output. An option is given to replay */
function askQuestion() {
    document.getElementById("next").style.display = "none";
    var statement = document.getElementById("statement");
    if (currentQuestion < questions.length) {
        statement.textContent = "Guess the french word for '" +
            questions[currentQuestion].english + "':";
        if (gameMode == 0) {
            document.getElementById("one").innerText = questions[currentQuestion].french_options[0];
            document.getElementById("two").innerText = questions[currentQuestion].french_options[1];
            document.getElementById("three").innerText = questions[currentQuestion].french_options[2];
            document.getElementById("options-game").style.display = "block";
        } else if (gameMode == 1) {
            document.getElementById("text-game").style.display = "inline-block";
        }
    } else {
        var commentary;
        if (score < 3) {
            commentary = "You should keep practising..."
        } else if (score < 5) {
            commentary = "You're making good progress!"
        } else {
            commentary = "You're practically fluent!"
        }
        statement.textContent = "Game over! You got " + score +
            " out of " + currentQuestion + " correct.\r\n " + commentary + "\r\n\r\nPlay again?";
        document.getElementById("start").style.display = "inline-block";
    }
}

// This retrieves the text input and sends it to Answer()
function sendTextAnswer() {
    var input = document.getElementById("text-input").value;
    document.getElementById("text-input").value = "";
    answerQuestion(input);

}

/* This checks if the guess parameter is the correct answer.
Based on the result, different messages are output. A link is given to
the wordreference dictionary page for the correct french word */
function answerQuestion(guess) {
    if (gameMode == 0) {
        document.getElementById("options-game").style.display = "none";
        guess = questions[currentQuestion].french_options[guess]
    } else if (gameMode == 1) {
        document.getElementById("text-game").style.display = "none";
    }
    var answer = questions[currentQuestion].answer;
    var resultMessage;
    if (guess == answer) {
        resultMessage = "Correct!\r\n The answer was '";
        score++;
    } else {
        resultMessage = "Incorrect, sorry.\r\n The answer was '";
    }
    var answerRef = "https://www.wordreference.com/fren/" + answer;
    var statement = document.getElementById("statement");
    statement.innerHTML = resultMessage + '<a target="_blank" href="' + answerRef + '">' +
        answer + '</a>' + "'.";
    currentQuestion++;
    document.getElementById("next").style.display = "inline-block";
}
```

Explications sur le code de l'application

Components :

L'application est construite sous la forme d'une application web qui utilise un fichier HTML et un fichier JavaScript. Le fichier HTML contient ce qui est affiché à l'utilisateur, tandis que le fichier JavaScript gère les algorithmes, les variables et les réactions de l'application aux entrées de l'utilisateur. Un fichier CSS est également utilisé pour rendre la présentation plus agréable. Enfin, un fichier texte est utilisé pour saisir les questions et les réponses nécessaires pour le jeu.

Démarche Générale :

Une déclaration qui guide l'utilisateur tout au long du jeu est toujours affichée. Au départ, elle a "Pick a difficulty & game mode and begin!" pour valeur. Des listes déroulantes de difficultés de jeu et de modes de jeu sont également affichées. Une fois que l'utilisateur a choisi ses paramètres de jeu, il clique sur le bouton "Start" qui appelle la fonction startGame(). L'utilisateur est alors invité à choisir ou à taper la traduction française correcte pour un mot anglais donné.

Si l'utilisateur a choisi le mode de jeu "options", trois boutons cliquables affichent trois mots français différents. L'utilisateur clique sur le bouton contenant la traduction qu'il pense être correcte. En cliquant sur un bouton, la fonction answerQuestion() est appelée. Selon le bouton choisi, un nombre différent est envoyé comme paramètre à cette fonction (bouton gauche = 0, bouton central = 1, bouton droit = 2). Si l'utilisateur a choisi le mode de jeu "texte", une zone de texte et un bouton à cliquer une fois qu'il a tapé sa réponse sont affichés. En cliquant sur le bouton, la fonction sendTextAnswer() est appelée.

Quel que soit le mode de jeu, le jeu procède de la même manière. L'utilisateur est informé s'ils sont corrects ou incorrects et la traduction correcte lui est présentée. La traduction est affichée sous la forme d'un lien qui, si l'utilisateur choisit de cliquer dessus, l'envoie à la page 'wordreference.com' du mot. En dessous, il y a un bouton 'Next' qui, lorsqu'on clique dessus, appelle la fonction askQuestion(). L'utilisateur reçoit alors un nouveau mot anglais et continue à jouer.

Une fois que le joueur a répondu à cinq questions, le jeu se termine. Au lieu de recevoir une nouvelle question en cliquant sur le bouton 'Next', l'utilisateur est informé que le jeu est terminé. On lui indique son score total sur cinq et, selon ce score, il reçoit un commentaire sur sa performance. Les listes déroulantes de difficultés et de modes de jeu sont affichées à nouveau et l'utilisateur a la possibilité de rejouer en cliquant sur le bouton 'Start'.

Variables Globales :

- **currentQuestion** - un nombre entier qui indique combien de questions ont été déjà posées (dans l'intervalle 0 <= x < 5)
- **score** - un nombre entier qui indique combien de questions l'utilisateur a bien répondu
- **questions** - un tableau de questions (objets 'dict' JavaScript) à utiliser dans ce tour de jeu
- **gameMode** - un nombre entier qui indique quelle version du jeu est jouée (0 = jeu d'options, 1 = jeu de texte)

Descriptions de Fonctions :

startGame() : Cette fonction est appelée à chaque fois qu'un nouveau jeu est lancé. Elle fixe les quatre variables globales à leurs valeurs respectives. *Score* et *currentQuestion* sont fixés à 0. *GameMode* et la variable locale *difficulty* sont fixés selon ce que l'utilisateur a choisi, ce qui est récupéré en utilisant le DOM HTML. La fonction *processTextFile()* alors appelée et utilise *difficulty* pour retourner un tableau de questions [voir :

processTextFile()]. *Questions* est alors fixé quand la fonction *getRandomisedQuestions()* est appelée, qui utilise la valeur de retour de *processTextFile()* comme paramètre [voir : getRandomisedQuestions()]. L'affichage initial (c'est-à-dire les listes déroulantes et le bouton de lancement) est caché et la fonction *askQuestion* est appelée [voir: askQuestion()].

processTextFile() : Cette fonction, qui prend comme paramètre une chaîne décrivant la difficulté du jeu, traite un fichier texte pour récupérer un tableau de questions (objets 'dict' JavaScript) du niveau de difficulté correct. Une requête GET est réalisée comme une XMLHttpRequest synchrone pour extraire les données du fichier texte. Si la requête est réussie, le texte renvoyé est divisé en lignes. Une boucle est utilisée pour parcourir chaque ligne, en commençant par la première ligne (i=0) et en terminant par la dernière ligne (i=*lines.length* - 1). À l'intérieur de cette boucle, on fait un test afin de ne traiter que les lignes qui ont la difficulté appropriée, c'est-à-dire les lignes qui commencent par la chaîne de caractères que la fonction prend en paramètre. Si cela est le cas, la ligne est divisée par des caractères "," pour former un tableau. Un 'dict' est créé avec les clés "english", "french_options" et "answer". Les valeurs sont remplies à l'aide du tableau créé et le 'dict' est ajouté à la variable locale *allQuestions* (un tableau de questions). Une fois que la boucle est terminée, *allQuestions* est envoyée comme valeur de retour de la fonction.

getRandomisedQuestions() : Cette fonction prend un tableau de questions, *allQuestions*, comme paramètre et y choisit cinq questions aléatoires à ajouter à *questions*. Pour ce faire, un tableau d'entiers appelés *nombres* est créé. Une boucle 'while' est utilisée pour ajouter 5 nombres aléatoires compris entre 0 et 9 (c'est-à-dire la plage d'index de *allQuestions*). Dans une boucle, chaque nombre de ce tableau est alors utilisé comme un index pour obtenir un objet question de *allQuestions*. Ces questions sont ajoutées à *questions*.

sendTextAnswer() : Cette fonction, utilisée lorsque le mode de jeu est "texte", permet de récupérer les réponses des utilisateurs dans la zone de saisie en utilisant le DOM HTML. Elle appelle ensuite la fonction *answerQuestion*() en utilisant le texte récupéré comme paramètre [voir : answerQuestion()].

askQuestion() : Cette fonction permet d'afficher le résultat approprié si le jeu est en cours ou terminé. Elle contrôle cela selon la valeur de *currentQuestion*. Si *currentQuestion* est inférieure au nombre de questions à poser, une nouvelle question est alors posée. L'énoncé de la question qui est affiché est créé en obtenant la valeur de la clé "english" pour la question courante dans *question*. Si le mode de jeu est "options", trois boutons sont affichés. Le texte affiché par chaque bouton est obtenu en obtenant la valeur de la clé "french_options", qui est un tableau contenant trois traductions. Si le mode de jeu est "texte", la zone de saisie est affichée. En revanche, si *currentQuestion* n'est pas inférieur au nombre de questions à poser, alors le score final (la variable globale *score*) est affiché. De plus, l'utilisateur reçoit un commentaire correspondant à son score. Il existe trois commentaires différents : un commentaire négatif si l'utilisateur obtient un score inférieur à 3, un commentaire neutre si l'utilisateur obtient 3 ou 4, et un commentaire positif si l'utilisateur obtient 5. Les listes déroulantes de difficulté et de mode de jeu sont également affichées, ainsi que le bouton 'Start'.

answerQuestion() : Cette fonction, qui prend la supposition de l'utilisateur comme paramètre, vérifie si la supposition est correcte et affiche ensuite la réponse appropriée. Tout d'abord, gameMode est vérifié et la possibilité de répondre à la question est masquée. La supposition est ensuite comparée à la réponse de la question courante (qui est obtenue en accédant à la valeur de la clé "réponse" dans le tableau questions). Si la supposition et la réponse sont égales, l'utilisateur est informé qu'il avait raison. Dans le cas contraire, l'utilisateur est informé qu'il n'avait pas raison. Un lien est également créé, utilisant la réponse comme texte, qui est affiché à l'utilisateur afin qu'il puisse rechercher le mot plus en profondeur. Pour avancer à la question suivante, *currentQuestion* est incrémenté et le bouton 'Next' est affiché.

Difficultés rencontrées

J'ai rencontré trois difficultés principales. Le premier est la gestion du fichier texte. Je n'étais pas sûr de la manière dont je voulais stocker les données des questions. Initialement, j'ai créé un fichier JavaScript séparé qui contenait un dictionnaire d'objets de questions. Cependant, j'ai estimé que ce n'était pas une bonne programmation et qu'elle ne serait pas efficace en cas d'entrées de données importantes. J'ai donc décidé de mettre les questions dans un fichier texte. De cette façon, je peux traiter les données avec plus de flexibilité dans mon fichier JavaScript principal.

La deuxième difficulté que j'ai rencontrée était liée à l'accès aux éléments HTML. Je voulais pouvoir modifier dynamiquement le texte et l'affichage de mon application web en utilisant JavaScript. Pour y parvenir, j'ai dû jouer avec le DOM HTML et trouver une façon efficace et propre de regrouper mes éléments HTML. J'ai trouvé que l'utilisation de "document.getElementById" était très utile.

Finalement, j'ai eu des difficultés à récupérer mon fichier texte. Je n'ai pas pu trouver un moyen simple et rapide de charger le fichier. À la fin, j'ai choisi d'utiliser XMLHTTPRequest car cela me semblait le plus simple et j'ai pu trouver des exemples de son utilisation. En utilisant cela, j'ai rencontré des problèmes liés au fait que la requête était asynchrone. J'ai finalement trouvé un moyen de le rendre synchrone en changeant un paramètre de vrai à faux.

Propositions d’amélioration

En ce qui concerne le mode de jeu "saisie de texte", la saisie de texte n'est soumise que lorsque l'on clique sur le bouton ‘Submit’. Il serait pratique que, lorsque l'on appuie sur la touche "Entrée", la demande réagisse de la même manière que lorsque l'on appuie sur le bouton.

Il serait également toujours bénéfique d'ajouter beaucoup plus de questions afin qu'il y ait plus de variété et plus de mots que l'on puisse apprendre.

Enfin, il serait bon qu'une sorte de tableau de bord soit établi. Pour ce faire, il faudrait entrer un nom d'utilisateur (ou se connecter) avant de jouer. De plus, une forme de base de données ou de fichier texte serait nécessaire pour stocker les données entre les sessions Internet. L'algorithme devrait stocker les scores dans cette base de données et ils seraient ensuite affichés dans l'ordre du plus haut au plus bas