# Créer une application de quiz avec HTML, CSS et Javascript  
  
![Écran d’accueil](./images/cover.png)  
  
Créez une application de quiz avec HTML, CSS et Javascript pour améliorer votre développement Web de base  
  
Vous souhaitez améliorer vos compétences \*\*de base en développement Web\*\*? Vous souhaitez améliorer vos connaissances de \*\*HTML, CSS et Javascript\*\*? Dans ce cours, vous apprendrez comment créer une application de jeu-questionnaire \*\*sans l’aide de bibliothèques ou de cadres\*\*. Voici quelques-uns des sujets que nous aborderons!

 Enregistrer les scores élevés dans le stockage local  
- Créer une barre de progression  
- Créer une icône de chargeur rotatif  
- Générer dynamiquement du HTML en Javascript  
- Récupérez les questions de trivia de l’API de base de données Open Trivia  
  
- Javascript - Fonctions de tableau (splice, map, sort), stockage local, API Fetch  
- ES6 Javascript Features - Arrow Functions, the Spread Operator, Const and Let, Template Literals  
- CSS - Flexbox, Animtations et unités REM  
  
## Introduction au cours et ressources

Bienvenue à "Build a Quiz App with HTML, CSS, and JS". Je suis tellement excité que vous avez décidé de prendre l’initiative d’améliorer vos compétences de base de développement Web!  
  
Dans ce cours, nous allons utiliser des compétences fondamentales en HTML, CSS et Javascript pour créer une application de quiz. Cette application sera en mesure de charger des questions à partir d’une API tierce partie, suivre et afficher des scores élevés, et bien plus encore! Vous apprendrez à utiliser Local Storage, créer des animations, utiliser les fonctionnalités Javascript modernes ES6, et plus encore.

Ressources  
  
- [Code source du cours] (https://github.com/jamesqquick/Design-And-Build-A-Quiz-App)  
- [Newsletter Learn Build Teach] (https://www.learnbuildteach.com/)  
- [Learn Build Teach Youtube Channel](https://www.youtube.com/c/jamesqquick)  
  
## 1. Création et style de la page d’accueil  
  
Dans cette vidéo, nous allons créer la page d’accueil avec un bon morceau de la CSS nécessaire. La page d’accueil se compose de quelques liens pour le jeu et les pages High Scores. Nous allons également créer des classes d’aide CSS pour Flexbox, boutons, et les éléments de masquage.

Je vous encourage tous à jeter un coup d’oeil aux extraits d’Emmet pour générer HTML et CSS.  
  
Ressources  
  
- [Emmet in Visual Studio Code](https://www.youtube.com/watch?v=5guZjNDcVnA)  
- [Understanding REM Units](https://www.sitepoint.com/understanding-and-using-rem-units-in-css/)  
- [A Complete Guide to Flexbox](<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>)

## 2. Créer et personnaliser la page du jeu  
  
Dans cette vidéo, nous allons créer la page de jeu et afficher les questions et réponses statiques. Finalement, nous chargerons les questions à partir d’une API, mais pour l’instant, nous allons coder une question afin d’établir le style.  
  
No 3. Afficher les questions codées en dur et les réponses  
  
Dans cette vidéo, nous chargerons les questions à partir d’un tableau codé en dur et les itérerons à travers les questions disponibles comme l’utilisation y répond. Nous utiliserons des attributs de données personnalisés, l’opérateur de propagation ES6, et les fonctions fléchées Javascript.

Ressources  
  
- [Création d’extraits de code dans Visual Studio Code](https://www.youtube.com/watch?v=K3gLlZm-m\_8)  
- [Utilisation des attributs de données] (https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use\_data\_attributes)  
- [Sélecteur de requête de document] (https://developer.mozilla.org/en-US/docs/Web/API/Document\_object\_model/Locating\_DOM\_elements\_using\_selectors)  
- [Document Get by ID] (https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById)  
- [Spread Operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread\_syntax)  
- [Arrow Functions](<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions>)

No 4. Afficher les commentaires pour les bonnes ou les mauvaises réponses  
  
Dans cette vidéo, nous vérifions si la réponse de l’utilisateur est correcte et nous affichons les commentaires à l’utilisateur avant de charger la question suivante.  
  
Ressources  
  
- [Bootstrap 4 Colors](https://www.w3schools.com/bootstrap4/bootstrap\_colors.asp)  
- [Triple vs Double Equals](https://codeburst.io/javascript-double-equals-vs-triple-equals-61d4ce5a121a)  
  
## 5. Créer un affichage tête vers le haut (HUD)  
  
Dans cette vidéo, nous allons créer un Heads Up Display (HUD) pour notre application de quiz. Cela affichera le score de l’utilisateur et le numéro de question actuel.

Ressources  
  
- [ES6 Template Literals](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template\_literals)  
  
## 6. Créer une barre de progression  
  
Dans cette vidéo, nous ferons un pas de plus en créant une barre de progression visuelle pour suivre les progrès de l’utilisateur à travers les questions.  
  
## 7. Créer et styliser la page finale  
  
Dans cette vidéo, nous allons créer notre page de fin où nous allons afficher le score de l’utilisateur atteint. Cet écran fournira un formulaire pour sauvegarder la partition et des liens pour jouer à nouveau ou rentrer à la maison.  
  
Ressources  
  
- [Local Storage](<https://www.w3schools.com/jsref/prop_win_localstorage.asp>)

## 8. Enregistrer les notes élevées dans le stockage local  
  
Dans cette vidéo, nous allons enregistrer et maintenir un tableau de scores élevés dans le stockage local. Pour ce faire, nous aurons besoin de JSON.stringify() et JSON.parse() pour convertir notre tableau de score élevé en une chaîne et visa versa.  
  
Ressources  
  
- [Local Storage](https://www.w3schools.com/jsref/prop\_win\_localstorage.asp)  
  
## 9. Charger et afficher les scores élevés du stockage local  
  
Dans cette vidéo, nous allons créer notre page High Scores. Nous devrons charger les scores élevés du stockage local, les itérer et les afficher à l’écran.

Ressources  
  
- [JSON Parse and Stringify](https://alligator.io/js/json-parse-stringify/)  
- [Tableau de tri] (https://www.w3schools.com/js/js\_array\_sort.asp)  
- [Tableau cartographique](https://www.w3schools.com/jsref/jsref\_map.asp)  
- [Array Join](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/join)  
  
## 10. Extraction de l’API pour charger les questions à partir du fichier JSON local  
  
Dans cette vidéo, nous déplacerons nos questions échantillon d’un tableau codé en dur à un fichier externe . json. Cela aidera à nettoyer notre fichier Game.js et à nous préparer à demander des questions à partir d’une API dans la prochaine vidéo.

Ressources  
  
- [Comment utiliser l’API Fetch](https://scotch.io/tutorials/how-to-use-the-javascript-fetch-api-to-get-data)  
- [Promesses](https://developer.mozilla.org/fr-US/docs/Web/JavaScript/Reference/Global\_Objects/Promise\_)  
  
## 11. Récupérez l’API pour charger les questions à partir de l’API Open Trivia  
  
Dans cette vidéo, nous utiliserons Fetch pour demander une liste de questions à partir de l’API de base de données Open Trivia.  
  
Réources  
  
- [Comment utiliser l’API Fetch](https://scotch.io/tutorials/how-to-use-the-javascript-fetch-api-to-get-data)  
- [Ouvrir la base de données Trivia] (https://opentdb.com/)  
- [Tableau cartographique](https://www.w3schools.com/jsref/jsref\_map.asp)  
- [Array For Each](<https://www.w3schools.com/jsref/jsref_foreach.asp>)

## 12. Créer un chargeur rotatif  
  
Dans cette vidéo, nous allons créer un simple chargeur rotatif en CSS qui sera affiché jusqu’à ce que nous ayons fini de demander / charger des questions à partir de l’API.  
  
Ressources  
  
- [Créer un chargeur CSS](https://www.w3schools.com/howto/howto\_css\_loader.asp)  
  
## 13. Clôture  
  
Merci beaucoup d’avoir suivi ce cours. J’espère sincèrement que vous l’avez apprécié et que vous avez amélioré vos compétences de base en développement Web!