Drap'O (HTML, CSS et JavaScript)

1. Structure de base (HTML)

```
html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Quiz des Drapeaux</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="app">
        <!-- Écran de connexion/inscription -->
        <div id="auth-screen" class="screen">
            <div class="auth-form">
                <h2>Connexion</h2>
                <input type="text" id="login-username" placeholder="Nom
d'utilisateur">
                <input type="password" id="login-password"
placeholder="Mot de passe">
                <button id="login-btn">Se connecter</button>
                <p>Pas encore de compte? <a href="#"
id="show-register">S'inscrire</a></p>
            </div>

            <div class="auth-form" style="display: none;">
                <h2>Inscription</h2>
                <input type="text" id="register-username"
placeholder="Nom d'utilisateur">
                <input type="password" id="register-password"
placeholder="Mot de passe">
                <button id="register-btn">S'inscrire</button>
                <p>Déjà un compte? <a href="#" id="show-login">Se
connecter</a></p>
            </div>
        </div>

        <!-- Écran de jeu -->
        <div id="game-screen" class="screen" style="display: none;">
```

```
            <div class="game-header">
                <div class="timer">Temps: <span
id="time">20</span>s</div>
                <div class="score">Score: <span
id="current-score">0</span>/10</div>
            </div>

            <div class="flag-container">
                <img id="flag-img" src="" alt="Drapeau du pays à
deviner">
            </div>

            <div class="answer-form">
                <input type="text" id="country-input" placeholder="Nom
du pays">
                <button id="submit-answer">Valider</button>
            </div>

            <div id="feedback" class="feedback"></div>
        </div>

        <!-- Écran des résultats -->
        <div id="results-screen" class="screen" style="display: none;">
            <h2>Résultats</h2>
            <p>Votre score: <span id="final-score">0</span>/10</p>

            <h3>Classement</h3>
            <table id="leaderboard">
                <thead>
                    <tr>
                        <th>Position</th>
                        <th>Joueur</th>
                        <th>Score</th>
                    </tr>
                </thead>
                <tbody id="leaderboard-body">
                    <!-- Les scores seront ajoutés ici dynamiquement -->
                </tbody>
            </table>

            <button id="play-again">Rejouer</button>
            <button id="logout-btn">Déconnexion</button>
        </div>
    </div>

    <script src="script.js"></script>
```

```
</body>
</html>
```

2. Style (CSS)

css

```css
/* styles.css */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f5f5f5;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}

.screen {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
    padding: 30px;
    width: 90%;
    max-width: 600px;
    text-align: center;
}

.auth-form {
    margin: 0 auto;
    width: 80%;
}

h2 {
    color: #2c3e50;
    margin-bottom: 20px;
}

input {
    display: block;
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ddd;
```

```
    border-radius: 5px;
    font-size: 16px;
}

button {
    background-color: #3498db;
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    margin-top: 10px;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #2980b9;
}

.game-header {
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
    font-size: 18px;
    font-weight: bold;
}

.flag-container {
    margin: 20px 0;
    height: 150px;
    display: flex;
    justify-content: center;
    align-items: center;
}

#flag-img {
    max-height: 100%;
    max-width: 100%;
    border: 1px solid #ddd;
}

.answer-form {
    display: flex;
    gap: 10px;
    margin-top: 20px;
```

```css
}

#country-input {
    flex-grow: 1;
}

.feedback {
    margin-top: 20px;
    min-height: 24px;
    font-weight: bold;
}

.correct {
    color: #27ae60;
}

.incorrect {
    color: #e74c3c;
}

#leaderboard {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}

#leaderboard th, #leaderboard td {
    padding: 10px;
    border-bottom: 1px solid #ddd;
}

#leaderboard th {
    background-color: #f2f2f2;
}

#play-again {
    background-color: #2ecc71;
    margin-right: 10px;
}

#play-again:hover {
    background-color: #27ae60;
}

#logout-btn {
    background-color: #e74c3c;
```

```
}

#logout-btn:hover {
    background-color: #c0392b;
}

.timer.warning {
    color: #e74c3c;
    animation: pulse 1s infinite;
}

@keyframes pulse {
    0% { opacity: 1; }
    50% { opacity: 0.5; }
    100% { opacity: 1; }
}
```

3. Logique du jeu (JavaScript)

javascript

```
// script.js
document.addEventListener('DOMContentLoaded', () => {
    // Éléments DOM
    const authScreen = document.getElementById('auth-screen');
    const gameScreen = document.getElementById('game-screen');
    const resultsScreen = document.getElementById('results-screen');

    const loginForm = document.querySelector('.auth-form');
    const registerForm = document.querySelectorAll('.auth-form')[1];

    const showRegisterLink = document.getElementById('show-register');
    const showLoginLink = document.getElementById('show-login');

    const loginBtn = document.getElementById('login-btn');
    const registerBtn = document.getElementById('register-btn');

    const timeDisplay = document.getElementById('time');
    const scoreDisplay = document.getElementById('current-score');
    const flagImg = document.getElementById('flag-img');
    const countryInput = document.getElementById('country-input');
    const submitAnswerBtn = document.getElementById('submit-answer');
    const feedbackDisplay = document.getElementById('feedback');

    const finalScoreDisplay = document.getElementById('final-score');
```

```
    const leaderboardBody = document.getElementById('leaderboard-body');
    const playAgainBtn = document.getElementById('play-again');
    const logoutBtn = document.getElementById('logout-btn');

    // Variables du jeu
    let countries = [];
    let currentGameCountries = [];
    let currentCountryIndex = 0;
    let score = 0;
    let timer;
    let timeLeft = 20;
    let currentUser = null;

    // API URL pour les pays
    const API_URL = 'https://restcountries.com/v3.1/all';

    // Basculer entre les formulaires de connexion et d'inscription
    showRegisterLink.addEventListener('click', (e) => {
        e.preventDefault();
        loginForm.style.display = 'none';
        registerForm.style.display = 'block';
    });

    showLoginLink.addEventListener('click', (e) => {
        e.preventDefault();
        registerForm.style.display = 'none';
        loginForm.style.display = 'block';
    });

    // Gestion de l'inscription
    registerBtn.addEventListener('click', () => {
        const username =
document.getElementById('register-username').value.trim();
        const password =
document.getElementById('register-password').value.trim();

        if (!username || !password) {
            alert('Veuillez remplir tous les champs');
            return;
        }

        // Stockage local simple (en production, utiliser un backend
sécurisé)
        const users = JSON.parse(localStorage.getItem('flagQuizUsers'))
|| [];
```

```
        if (users.some(user => user.username === username)) {
            alert('Ce nom d\'utilisateur est déjà pris');
            return;
        }

        users.push({ username, password, scores: [] });
        localStorage.setItem('flagQuizUsers', JSON.stringify(users));

        alert('Inscription réussie! Vous pouvez maintenant vous
connecter.');
        registerForm.style.display = 'none';
        loginForm.style.display = 'block';
    });

    // Gestion de la connexion
    loginBtn.addEventListener('click', () => {
        const username =
document.getElementById('login-username').value.trim();
        const password =
document.getElementById('login-password').value.trim();

        if (!username || !password) {
            alert('Veuillez remplir tous les champs');
            return;
        }

        const users = JSON.parse(localStorage.getItem('flagQuizUsers'))
|| [];
        const user = users.find(u => u.username === username &&
u.password === password);

        if (!user) {
            alert('Nom d\'utilisateur ou mot de passe incorrect');
            return;
        }

        currentUser = user;
        startGame();
    });

    // Déconnexion
    logoutBtn.addEventListener('click', () => {
        currentUser = null;
        authScreen.style.display = 'block';
        gameScreen.style.display = 'none';
        resultsScreen.style.display = 'none';
```

```
    });

    // Rejouer
    playAgainBtn.addEventListener('click', startGame);

    // Charger les pays depuis l'API
    async function fetchCountries() {
        try {
            const response = await fetch(API_URL);
            if (!response.ok) throw new Error('Erreur de chargement des
pays');

            const data = await response.json();

            // Filtrer les pays qui ont un drapeau et un nom commun
            return data.filter(country =>
                country.flags?.png && country.name?.common
            ).map(country => ({
                name: country.name.common,
                flag: country.flags.png
            }));
        } catch (error) {
            console.error('Erreur:', error);
            return [];
        }
    }

    // Démarrer une nouvelle partie
    async function startGame() {
        if (countries.length === 0) {
            countries = await fetchCountries();
            if (countries.length === 0) {
                alert('Impossible de charger les pays. Veuillez
réessayer plus tard.');
                return;
            }
        }

        // Sélectionner 10 pays aléatoires uniques
        currentGameCountries = [];
        const shuffled = [...countries].sort(() => 0.5 - Math.random());
        currentGameCountries = shuffled.slice(0, 10);

        currentCountryIndex = 0;
        score = 0;
        scoreDisplay.textContent = score;
```

```
        authScreen.style.display = 'none';
        gameScreen.style.display = 'block';
        resultsScreen.style.display = 'none';

        loadNextCountry();
    }

    // Charger le pays suivant
    function loadNextCountry() {
        if (currentCountryIndex >= currentGameCountries.length) {
            endGame();
            return;
        }

        const currentCountry =
currentGameCountries[currentCountryIndex];
        flagImg.src = currentCountry.flag;
        countryInput.value = '';
        countryInput.focus();
        feedbackDisplay.textContent = '';
        feedbackDisplay.className = 'feedback';

        // Réinitialiser et démarrer le timer
        clearInterval(timer);
        timeLeft = 20;
        timeDisplay.textContent = timeLeft;
        timeDisplay.classList.remove('warning');

        timer = setInterval(() => {
            timeLeft--;
            timeDisplay.textContent = timeLeft;

            if (timeLeft <= 5) {
                timeDisplay.classList.add('warning');
            }

            if (timeLeft <= 0) {
                clearInterval(timer);
                handleTimeOut();
            }
        }, 1000);
    }

    // Gérer la fin du temps
    function handleTimeOut() {
```

```
        feedbackDisplay.textContent = `Temps écoulé! La réponse était: 
${currentGameCountries[currentCountryIndex].name}`;
        feedbackDisplay.className = 'feedback incorrect';

        setTimeout(() => {
            currentCountryIndex++;
            loadNextCountry();
        }, 2000);
    }

    // Soumettre la réponse
    submitAnswerBtn.addEventListener('click', checkAnswer);
    countryInput.addEventListener('keypress', (e) => {
        if (e.key === 'Enter') checkAnswer();
    });

    function checkAnswer() {
        const userAnswer = countryInput.value.trim().toLowerCase();
        const correctAnswer = 
currentGameCountries[currentCountryIndex].name.toLowerCase();

        clearInterval(timer);

        if (userAnswer === correctAnswer) {
            score++;
            scoreDisplay.textContent = score;
            feedbackDisplay.textContent = 'Correct!';
            feedbackDisplay.className = 'feedback correct';
        } else {
            feedbackDisplay.textContent = `Incorrect! La réponse était: 
${currentGameCountries[currentCountryIndex].name}`;
            feedbackDisplay.className = 'feedback incorrect';
        }

        setTimeout(() => {
            currentCountryIndex++;
            loadNextCountry();
        }, 2000);
    }

    // Terminer la partie
    function endGame() {
        gameScreen.style.display = 'none';
        resultsScreen.style.display = 'block';

        finalScoreDisplay.textContent = score;
```

```
        // Enregistrer le score
        if (currentUser) {
            const users =
JSON.parse(localStorage.getItem('flagQuizUsers'));
            const userIndex = users.findIndex(u => u.username ===
currentUser.username);

            users[userIndex].scores.push({
                score,
                date: new Date().toISOString()
            });

            // Garder seulement les 10 meilleurs scores
            users[userIndex].scores.sort((a, b) => b.score - a.score);
            if (users[userIndex].scores.length > 10) {
                users[userIndex].scores =
users[userIndex].scores.slice(0, 10);
            }

            localStorage.setItem('flagQuizUsers',
JSON.stringify(users));
        }

        // Mettre à jour le classement
        updateLeaderboard();
    }

    // Mettre à jour le tableau des scores
    function updateLeaderboard() {
        const users = JSON.parse(localStorage.getItem('flagQuizUsers'))
|| [];

        // Trier les utilisateurs par leur meilleur score
        const sortedUsers = users.map(user => ({
            username: user.username,
            bestScore: Math.max(...user.scores.map(s => s.score), 0)
        })).sort((a, b) => b.bestScore - a.bestScore);

        leaderboardBody.innerHTML = '';

        sortedUsers.slice(0, 10).forEach((user, index) => {
            const row = document.createElement('tr');

            if (currentUser && user.username === currentUser.username) {
                row.classList.add('current-user');
```

```
            }

            row.innerHTML = `
                <td>${index + 1}</td>
                <td>${user.username}</td>
                <td>${user.bestScore}</td>
            `;

            leaderboardBody.appendChild(row);
        });
    }
});
```

4. Explications du code

1. Structure HTML

- **3 écrans principaux** : authentification, jeu et résultats
- **Authentification** : formulaire de connexion et d'inscription
- **Jeu** : affichage du drapeau, champ de réponse et compteurs
- **Résultats** : affichage du score et classement

2. Style CSS

- Design responsive et moderne
- Animations pour le timer quand il reste peu de temps
- Feedback visuel pour les réponses correctes/incorrectes
- Mise en évidence du joueur actuel dans le classement

3. Logique JavaScript

A. Initialisation

javascript

```
document.addEventListener('DOMContentLoaded', () => { ... });
```

- Le code s'exécute une fois la page chargée.

B. Authentification

- Stockage des utilisateur :
  - Utilise **localStorage** pour sauvegarder les comptes (en production, utilisez un backend sécurisé !).

- Structure d'un utilisateur :
  javascript
  {
  username: "pseudo",
  password: "motdepasse", // Non sécurisé (à hasher en production)
  scores: [{ score: 8, date: "2024-05-01" }, ...] // Historique des parties
  }

- Connexion/Inscription :
  - Basculer entre les formulaires avec **showRegisterLink**/ **showLoginLink**.
  - Vérification des champs et de l'unicité du pseudo.

C. Chargement des Pays

javascript

```
async function fetchCountries() {
  const response = await fetch('https://restcountries.com/v3.1/all');
  const data = await response.json();
  return data.filter(country => country.flags?.png &&
country.name?.common)
             .map(country => ({ name: country.name.common, flag:
country.flags.png }));
}
```

- API REST Countries : Récupère tous les pays avec leurs drapeaux.
- Filtrage : Garde seulement les pays avec un drapeau (**flags.png**) et un nom commun (**name.common**).

D. Mécanique de Jeu

1. Initialisation d'une partie (`**startGame**`) :
   - Sélectionne 10 pays aléatoires uniques :
     ```javascript
     const shuffled = [...countries].sort(() => 0.5 - Math.random());
     currentGameCountries = shuffled.slice(0, 10);
     ```

2. Chargement d'un pays (**loadNextCountry**) :

   - Affiche le drapeau (flagImg.src = currentCountry.flag).
   - Lance un timer de 20 secondes :

javascript

```
timer = setInterval(() => {
       timeLeft--;
       timeDisplay.textContent = timeLeft;
       if (timeLeft <= 0) handleTimeOut();
     }, 1000);
```

3. Vérification de la réponse (**checkAnswe**r) :

- Compare la saisie utilisateur (**userAnswe**r) avec le nom du pays (correctAnswer).
- Met à jour le score si correct :

javascript

```
if (userAnswer === correctAnswer) {
       score++;
       feedbackDisplay.textContent = "Correct!";
     }
```

4. Fin du jeu (**endGame**) :
- Affiche le score final.
- Sauvegarde le score dans le **localStorage**.
- Met à jour le classement (**updateLeaderboard**).

E. Gestion du Temps

- Timer:
  - Décompte de 20 à 0 secondes.
  - Si temps écoulé (**handleTimeOut**), passe au pays suivant sans ajouter de point.
- Feedback urgent :
  - Changement de couleur et animation quand il reste ≤ 5 secondes.

F. Classement

- Tri des scores :

  - Chaque utilisateur conserve ses 10 meilleurs scores.
  - Le classement général trie les utilisateurs par leur meilleur score.

- Affichage :
  - Highlight du joueur actuel dans le tableau (**current-user**).

4. Points Clés à Retenir

- API : Les données viennent de REST Countries (gratuit et sans clé API).

- Aléatoire : Sélection de pays uniques avec **sort(() => 0.5 - Math.random())**.

- Persistance :

  - Les comptes et scores sont stockés dans le **localStorage** (limité à 5-10 Mo).

  - En production, il faudrait un backend avec une vraie base de données.
- Sécurité :
  - Les mots de passe sont en clair ici (à hasher avec bcrypt en réel).