Backup

Ce que je dois faire : j’ai plus le code qui crée un quizz,

Il faut les faire être sur le même quizz

Améliorer l’interface

Disposition des questions etc

En direct

Les réponses

Ajout de personnages

oui plein de choses à faire, déjà il faut que l'interface fonctionne bien entre le host et les players, que les players puissent choisir un pseudo, qu'il y ait un classement, qu'on fasse un set de questions aléatoires et qu'on puisse à chaque partie choisir le nombre de questions du set, qu'il y ait un ranking qui se fasse selon si la réponse est juste ou pas mais aussi selon la rapidité, gerer aussi l'accessibilité dans le sens comment on se connecte à la partie

et aussi que ce soit le host qui choisisse quand on passe à la prochaine question

aussi mettre une fonctionnalité de plusieurs réponses correctes

Code pin pour entrer dans la partie

Ouvrir une partie avec les joueureuses

Démarrer la partie

Ajouter une autre interlude, à côté du bouton passer à la question suivante on peut mettre un autre bouton qui dit afficher la bonne réponse et

oui et aussi ce que je veux c'est que si je rafraichi la page du host j'arrive à nouveau sur la page de démarrage, comme sur la page host je reviens au moment des pseudos mais je continuerai une prochaine fois aussi le truc du code pour entrer dans une partie, bref j'écris ca la pour pas oublier

server avant modif :

const express = require('express');

const app = express();

const http = require('http').createServer(app);

const io = require('socket.io')(http);

const questionsFixes = require('./questions');

const port = 3000;

app.use(express.static('public'));

let currentQuestionIndex = -1; // Phase d'attente initiale

let questionStartTime = null;

const players = {}; // { socketId: { pseudo, score, answer } }

io.on('connection', (socket) => {

  console.log('Un joueur est connecté:', socket.id);

  socket.on('pseudo', (pseudo) => {

    players[socket.id] = { pseudo, score: 0, answer: null };

    console.log(`Pseudo défini : ${pseudo}`);

    // Informer le joueur qu’il doit patienter

    socket.emit('waitingForStart');

    // Met à jour la liste des joueurs côté host

    io.emit('playerList', Object.values(players).map(p => p.pseudo));

  });

  socket.on('startQuiz', () => {

    currentQuestionIndex = 0;

    for (let id in players) {

      players[id].answer = null;

      players[id].score = 0;

    }

    questionStartTime = Date.now();

    io.emit('question', questionsFixes[currentQuestionIndex]);

  });

  socket.on('nextQuestion', () => {

    currentQuestionIndex++;

    if (currentQuestionIndex < questionsFixes.length) {

      for (let id in players) {

        players[id].answer = null;

      }

      questionStartTime = Date.now();

      io.emit('question', questionsFixes[currentQuestionIndex]);

    } else {

      const classement = Object.values(players)

        .map(p => ({ pseudo: p.pseudo, score: p.score }))

        .sort((a, b) => b.score - a.score);

      io.emit('quizEnd', classement);

    }

  });

  socket.on('answer', (answerIndex) => {

    if (!players[socket.id] || players[socket.id].answer !== null) return;

    players[socket.id].answer = answerIndex;

    const delay = (Date.now() - questionStartTime) / 1000; // en secondes

    const maxDelay = 7;

    let bonus = 0;

    if (answerIndex === questionsFixes[currentQuestionIndex].bonne) {

      players[socket.id].score += 1;

      const maxBonus = 0.5;

      if (delay < maxDelay) {

        bonus = ((maxDelay - delay) / maxDelay) \* maxBonus;

        bonus = Math.max(0, Math.min(maxBonus, bonus));

        players[socket.id].score += Number(bonus.toFixed(2));

      }

    }

    socket.emit('waiting');

    io.emit('playersAnswers', players);

  });

  socket.on('disconnect', () => {

    console.log('Joueur déconnecté:', socket.id);

    delete players[socket.id];

    io.emit('playerList', Object.values(players).map(p => p.pseudo));

    io.emit('playersAnswers', players);

  });

  // Si le quiz a commencé, envoyer la question courante

  if (currentQuestionIndex >= 0) {

    socket.emit('question', questionsFixes[currentQuestionIndex]);

  } else {

    socket.emit('waitingForStart');

  }

});

http.listen(port, () => {

  console.log(`Serveur démarré sur http://localhost:${port}`);

});

Host avant modif :   
<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>Host Quiz</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>🎮 Host Quiz</h1>

  <div id="waitingPhase">

    <h2>Le quiz va bientôt démarrer...</h2>

    <p>Joueurs connectés :</p>

    <ul id="playerList"></ul>

    <button id="startQuizBtn">🚀 Démarrer le quiz</button>

  </div>

  <div id="quizPhase" style="display: none;">

    <div id="question"></div>

    <div id="reponses"></div>

    <button id="nextBtn">➡️ Question suivante</button>

    <div id="resultats"></div>

  </div>

  <script src="/socket.io/socket.io.js"></script>

  <script>

    const socket = io();

    const questionEl = document.getElementById('question');

    const reponsesEl = document.getElementById('reponses');

    const resultatsEl = document.getElementById('resultats');

    const nextBtn = document.getElementById('nextBtn');

    const waitingPhase = document.getElementById('waitingPhase');

    const quizPhase = document.getElementById('quizPhase');

    const playerList = document.getElementById('playerList');

    const startQuizBtn = document.getElementById('startQuizBtn');

    let couleurs = ['c0', 'c1', 'c2', 'c3'];

    // Affiche la liste des joueurs connectés

    socket.on('playerList', (pseudos) => {

      playerList.innerHTML = '';

      pseudos.forEach(p => {

        const li = document.createElement('li');

        li.textContent = p;

        playerList.appendChild(li);

      });

    });

    // Le quiz démarre

    socket.on('question', (q) => {

      waitingPhase.style.display = 'none';

      quizPhase.style.display = 'block';

      questionEl.textContent = q.question;

      reponsesEl.innerHTML = '';

      resultatsEl.innerHTML = '';

      q.reponses.forEach((rep, i) => {

        const div = document.createElement('div');

        div.className = `color-box ${couleurs[i]}`;

        div.textContent = rep;

        reponsesEl.appendChild(div);

      });

    });

    socket.on('playersAnswers', (players) => {

      resultatsEl.innerHTML = '<h3>Réponses des joueurs :</h3>';

      for (let id in players) {

        const p = players[id];

        if (p.answer !== null) {

          resultatsEl.innerHTML += `<p>${p.pseudo || id} : ✅ ${p.score} pts</p>`;

        } else {

          resultatsEl.innerHTML += `<p>${p.pseudo || id} : en attente...</p>`;

        }

      }

    });

    socket.on('quizEnd', (classement) => {

      questionEl.textContent = "✅ Fin du quiz !";

      reponsesEl.innerHTML = '<h3>Classement final :</h3><ol>' +

        classement.map(p => `<li>${p.pseudo} – ${p.score} pts</li>`).join('') +

        '</ol>';

    });

    // Bouton pour commencer le quiz

    startQuizBtn.addEventListener('click', () => {

      socket.emit('startQuiz');

    });

    nextBtn.addEventListener('click', () => {

      socket.emit('nextQuestion');

    });

  </script>

</body>

</html>

Player avant modif :

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>Player Quiz</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h2>Entre ton pseudo</h2>

  <input id="pseudoInput" placeholder="Ton nom" />

  <button id="startBtn">Commencer</button>

  <div id="waitingScreen" style="display: none; margin: 20px;">

    <h2>👀 Tiens-toi prêt, le quiz va bientôt commencer !</h2>

  </div>

  <div id="quiz" style="display:none;">

    <div id="waiting" style="margin:20px;"></div>

    <div id="colorButtons"></div>

  </div>

  <script src="/socket.io/socket.io.js"></script>

  <script>

    const socket = io();

    const pseudoInput = document.getElementById('pseudoInput');

    const startBtn = document.getElementById('startBtn');

    const quiz = document.getElementById('quiz');

    const waiting = document.getElementById('waiting');

    const colorButtons = document.getElementById('colorButtons');

    const waitingScreen = document.getElementById('waitingScreen');

    let couleurs = ['c0', 'c1', 'c2', 'c3'];

    let ready = false;

    startBtn.addEventListener('click', () => {

      const pseudo = pseudoInput.value.trim();

      if (pseudo) {

        socket.emit('pseudo', pseudo);

        document.querySelector('h2').style.display = 'none';

        pseudoInput.style.display = 'none';

        startBtn.style.display = 'none';

        waitingScreen.style.display = 'block';

        ready = true;

      }

    });

    socket.on('question', (q) => {

      if (!ready) return;

      waitingScreen.style.display = 'none';

      quiz.style.display = 'block';

      waiting.textContent = "Choisis une couleur :";

      colorButtons.innerHTML = '';

      q.reponses.forEach((\_, i) => {

        const btn = document.createElement('button');

        btn.className = `color-button ${couleurs[i]}`;

        btn.onclick = () => socket.emit('answer', i);

        colorButtons.appendChild(btn);

      });

    });

    socket.on('waiting', () => {

      if (!ready) return;

      waiting.textContent = "🕒 En attente des autres joueurs...";

      colorButtons.innerHTML = '';

    });

    socket.on('quizEnd', (classement) => {

      if (!ready) return;

      waiting.textContent = "✅ Quiz terminé !";

      colorButtons.innerHTML = '<h3>Classement final :</h3><ol>' +

        classement.map(p => `<li>${p.pseudo} – ${p.score} pts</li>`).join('') +

        '</ol>';

    });

  </script>

</body>

</html>