

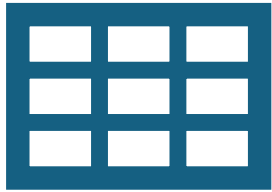
A hiker with a large green backpack is walking away from the viewer on a rocky mountain trail. The hiker is wearing a brown jacket and a hat. The trail is surrounded by lush green grass and yellow wildflowers. In the background, there are majestic snow-capped mountains under a blue sky with white clouds. The overall scene is a beautiful, stylized illustration of a mountain landscape.

# Adventure Spin

---

Adventure Game by Ungerhofer Julian

# Überblick

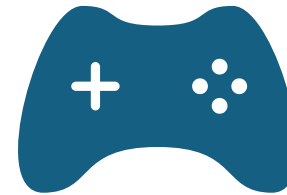


## **Pages: Landingpage mit Infopage**

Data Page mit eingabe der Daten

Map Page (3 unterseiten, Map, Shop und Charakter)

Game Pages



## **Games & Funktionen:**

5 verschiedene Spannende Spiele

Tolle Tools und extras



# Adventure Spin

## Choose your Character


Pick your favorite character and dive into the adventure! Unlock new characters by collecting coins and overcoming exciting challenges. Each character has a unique style—find the one that suits you best!

## Thrilling Levels & Challenges

Jump, collect coins, and dodge dangerous obstacles! Complete levels to earn a spin on the special gambling machine for a chance to win even more coins. Only the best players can unlock them all!

[How to?](#)

Start Game



Coin: 0  
Level: 0  
Daily Lucky Wheel: unused

Name (Full Name):

Birthdate:

Kreditcardnumber:

Nickname:

Sound:

test

## How to play?

Welcome to the ultimate collecting challenge! Your goal is simple: move the harrel left and right to catch the right items while avoiding the wrong ones. Each level gets harder as more fake items drop and the speed increases!

### Main Gameplay

- Move the harrel left and right to collect the correct items.
- Avoid the wrong items—catching them will cost you points!
- As you progress, more wrong items appear, and everything falls faster!
- Complete levels to earn coins and unlock new characters!

### Bonus Level

- Every few levels, you'll enter a 15-second Bonus Round!
- During this time, only coins fall—catch as many as you can!
- Use your collected coins to unlock new skins for your character!

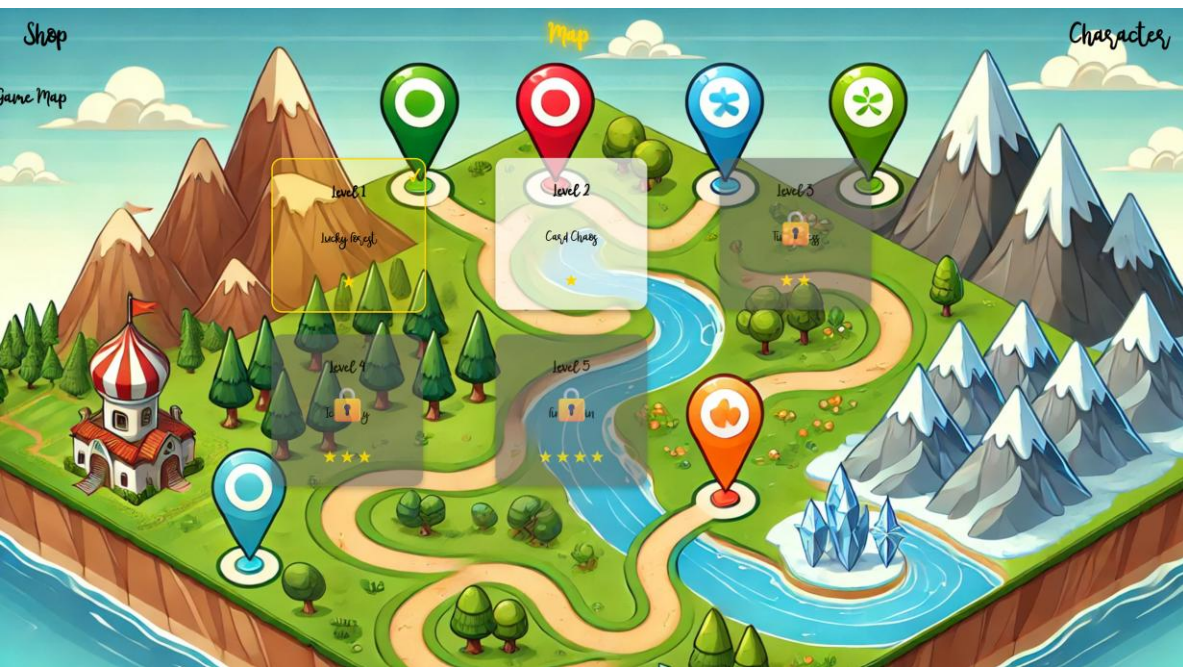
Are you ready to test your reflexes and collect all the rewards? Start playing now!

Back

## Starter-Seiten-Überblick



# Map-Page



=>



# Code-Level-System

```
.level-box {
  width: 180px;
  height: 180px;
  background-color: rgba(255, 255, 255, 0.7);
  border-radius: 15px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  cursor: pointer;
  transition: transform 0.3s ease, box-shadow 0.3s ease;
  position: relative;
  padding: 15px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
  font-family: 'Amy', sans-serif;
  color: black;
}

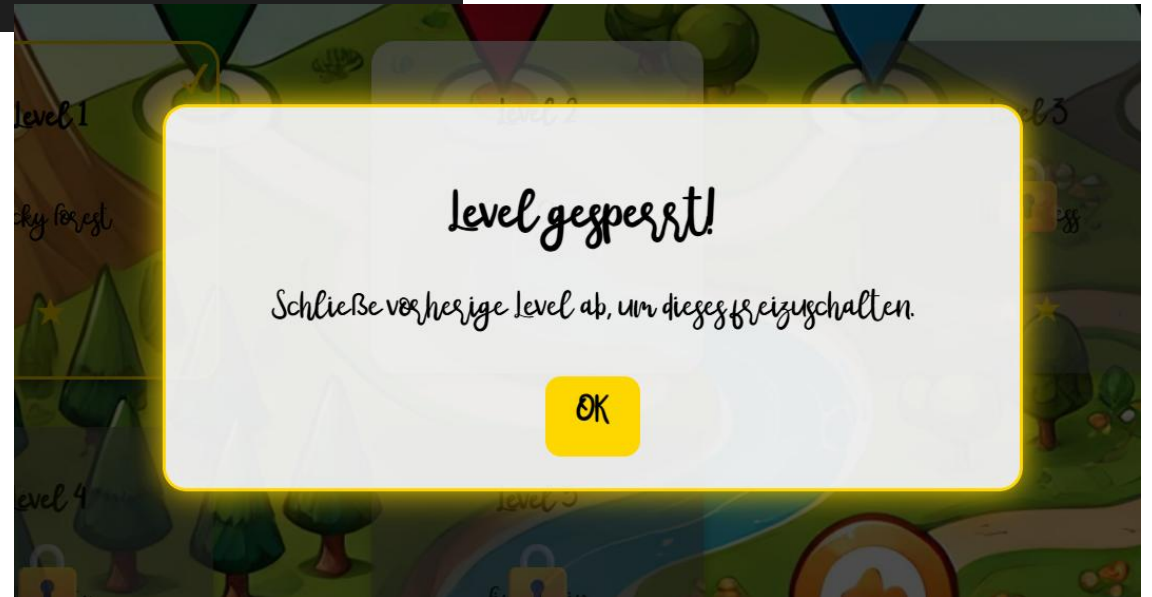
.level-box:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 15px rgba(0, 0, 0, 0.4);
  background-color: rgba(255, 255, 255, 0.9);
}

.level-box.locked {
  background-color: rgba(100, 100, 100, 0.7);
  opacity: 0.8;
  cursor: not-allowed;
}

.level-box.locked::after {
  content: "🔒";
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  font-size: 3em;
}
```

```
//Unlocked Level
function openLevelModal() {
  document.getElementById("levelLockedModal").style.display
  playSound("errorSound");
}

function closeLevelModal() {
  document.getElementById("levelLockedModal").style.display
}
```





# Character-Page

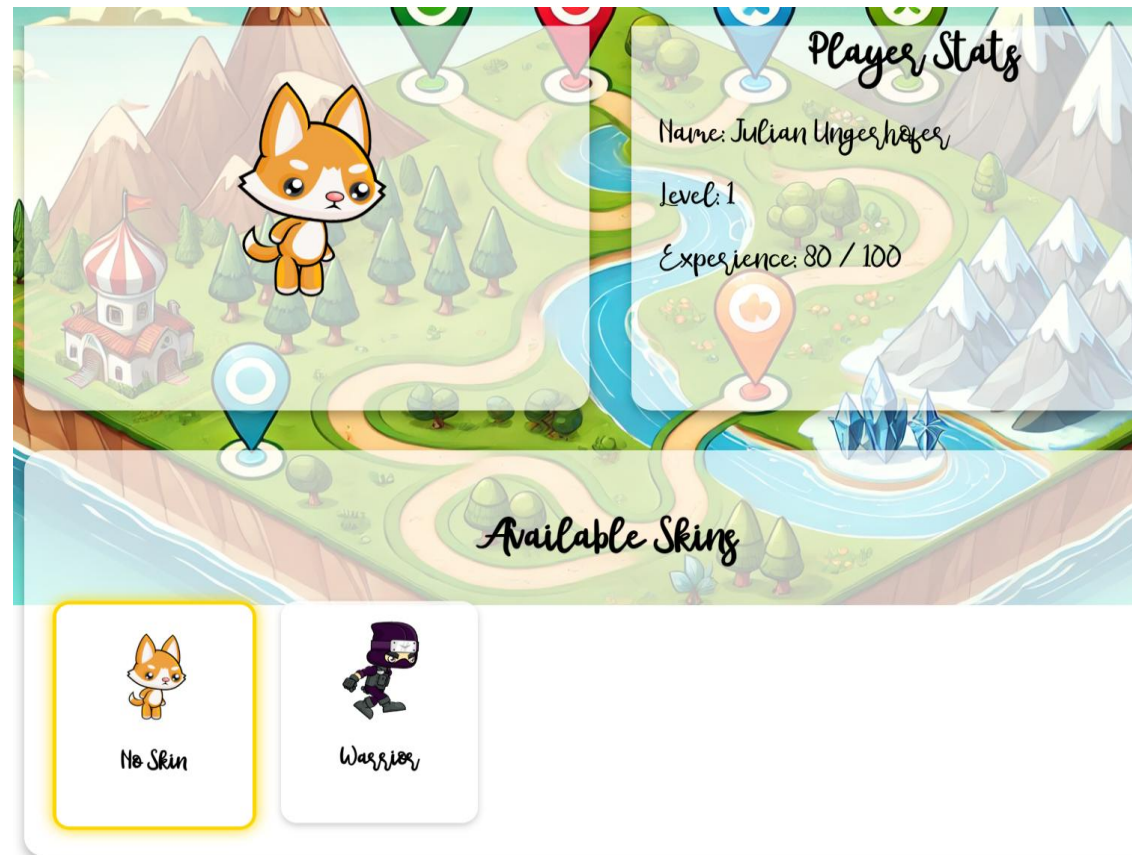
```
<div id="sigmaCharacter">
  <h2>Character Customization</h2>

  <div class="character-display">
    <div class="character-preview">
      
    </div>

    <div class="character-stats">
      <h3>Player Stats</h3>
      <div class="stat">Name: <span id="playerName">Player</span></div>
      <div class="stat">Level: <span id="playerLevel">1</span></div>
      <div class="stat">Experience: <span id="playerExp">0</span> / <span id="playerMaxExp">100</span></div>
    </div>
  </div>

  <div class="skin-selection">
    <h3>Available Skins</h3>
    <div class="skins-grid">
      <div class="skin-option selected" onclick="changeSkin('NoSkin')">
        
        <h4>No Skin</h4>
      </div>
      <!-- Other skins would be added when purchased -->
    </div>
  </div>

  <div class="reset-box">
    <h3>Spiel zurücksetzen</h3>
    <p>Alle Spielstände und Fortschritte werden gelöscht.</p>
    <button onclick="resetGame()">Zurücksetzen</button>
  </div>
</div>
```

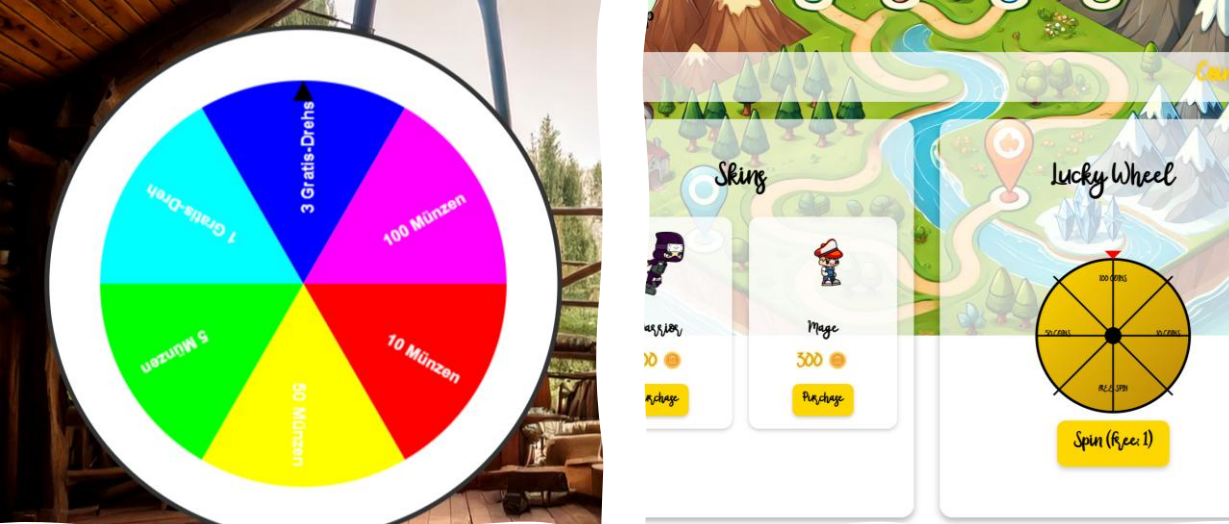


Charakter hinzufügen + gespeicherte Inhalte anzeigen.

```
// Add skin to selection
function addSkinToSelection(skinName) {
  playSound("click");
  const skinsGrid = document.querySelector('.skins-grid');
  const skinDiv = document.createElement('div');
  skinDiv.className = 'skin-option';
  skinDiv.onclick = function() { changeSkin(skinName); };

  //todo richtiger pfad funktioniert noch nicht -> funktioniereirt
  skinDiv.innerHTML = `
    
    <h4>${skinName.charAt(0).toUpperCase() + skinName.slice(1)}</h4>
  `;

  skinsGrid.appendChild(skinDiv);
}
```



# Shop-Page

- Fehlermeldung falls zu wenig Münzen vorhanden sind
- Extra Lucky Wheel Seite
- Bei kauf werden die Münzen sowie der Skin im Local-Storage gespeichert und aktualisiert



# Shop-Wheel-Code

```
const segments = [
  { label: '10 Münzen', type: 'coins', value: 10 },
  { label: '50 Münzen', type: 'coins', value: 50 },
  { label: '5 Münzen', type: 'coins', value: 100 },
  { label: '1 Gratis-Dreh', type: 'spin', value: 1 },
  { label: '3 Gratis-Drehs', type: 'spin', value: 3 },
  { label: '100 Münzen', type: 'coins', value: 200 }
];
```

```
function updateGameData(prize) {
  if (prize.type === 'coins') {
    gameData.coins += prize.value;
  } else if (prize.type === 'spin') {
    gameData.freeSpins += prize.value;
  }

  localStorage.setItem('gameData', JSON.stringify(gameData));

  setTimeout(() => {
    window.location.href = '../html/map.html';
  }, 3000);
}
```

```
function drawWheel() {
  for (let i = 0; i < segmentCount; i++) {
    const angle = startAngle + i * anglePerSegment;
    ctx.beginPath();
    ctx.moveTo(250, 250);
    ctx.arc(250, 250, 200, angle, angle + anglePerSegment);
    ctx.fillStyle = `hsl(${(i * 360) / segmentCount}, 100%, 50%)`;
    ctx.fill();
    ctx.save();
    ctx.translate(250, 250);
    ctx.rotate(angle + anglePerSegment / 2);
    ctx.textAlign = 'right';
    ctx.fillStyle = '#fff';
    ctx.font = 'bold 16px sans-serif';
    ctx.fillText(segments[i].label, 180, 10);
    ctx.restore();
  }

  // Zeiger
  ctx.fillStyle = '#000';
  ctx.beginPath();
  ctx.moveTo(250, 50);
  ctx.lineTo(240, 70);
  ctx.lineTo(260, 70);
  ctx.closePath();
  ctx.fill();

  if (gameData.freeSpins < 0) {
    setTimeout(() => {
      window.location.href = '../html/map.html';
    }, 2000);
  }
}
```



```
function startGeneratingItems() {
  setInterval(function() {
    if (itemCount >= 15 || gamePhase !== "game") return;

    const item = document.createElement("div");
    item.className = "item";
    item.style.left = Math.random() * 770 + "px";
    item.style.top = "-40px";

    const itemType = Math.random();

    if (itemType < 0.6) {
      // 60% Wahrscheinlichkeit für ein einsammelbares
      item.classList.add("collectible");
      item.dataset.type = "collectible";
    } else if (itemType < 0.9) {
      // 30% Wahrscheinlichkeit für eine Münze
      item.classList.add("coin");
      item.dataset.type = "coin";
    } else {
      // 10% Wahrscheinlichkeit für ein gefährliches
      item.classList.add("dangerous");
      item.dataset.type = "dangerous";
    }

    document.getElementById("gameScreen").appendChild(item);

    items.push({
      element: item,
      speed: 1 + Math.random() * 3
    });
  }, 800);
}
```

```
function startGame() {
  document.getElementById("startScreen").style.display = "none";
  document.getElementById("entryPhase").style.display = "block";
  gamePhase = "entry";

  const character = document.querySelector("#entryPhase #character");
  if (character) {
    character.style.left = "10%";
    character.style.bottom = "50%";
    characterPos.x = parseInt(character.style.left);
    characterPos.y = parseInt(character.style.bottom);
  }

  setupKeyboardControls();
  loadCharacterFrames();
  gameLoop();
}

function loadCharacterFrames() {
  for (let i = 0; i < 8; i++) {
    characterFrames[i] = `../img/NoSkin/Run${i+1}.png`;
  }

  console.log(characterFrames);

  const entryCharacter = document.querySelector("#entryPhase #character");
  const gameCharacter = document.querySelector("#gameScreen #character");

  if (entryCharacter) {
    entryCharacter.style.backgroundImage = `url('${characterFrames[0]}')`;
    entryCharacter.style.backgroundSize = 'contain';
    entryCharacter.style.backgroundRepeat = 'no-repeat';
  }

  if (gameCharacter) {
    gameCharacter.style.backgroundImage = `url('${characterFrames[0]}')`;
    gameCharacter.style.backgroundSize = 'contain';
    gameCharacter.style.backgroundRepeat = 'no-repeat';
  }
}
```

# Level 1

Sammel-Spiel:

- ➔ Items fallen von den Bäumen
- ➔ Spieler muss mit seinem Charakter der ein Faß mitsich führt die Items einsammeln
- ➔ Leben, Zeit und Erfahrung

# Erfahrung + Belohnungen

```
function spin() {
  const symbols = ["🍒", "🍋", "🍌", "🍉", "🍊"];
  const slots = [
    document.getElementById("slot1"),
    document.getElementById("slot2"),
    document.getElementById("slot3")
  ];

  //disable button während gedrückt wird => Verhinderung doppeltes drehen
  document.getElementById("gambleButton").disabled = true;

  // Kurze Animation mit GSAP
  //GSAP für Gambling
  gsap.to(slots, {
    rotation: 360,
    duration: 0.5,
    ease: "power1.inOut",
    onComplete: function() {
      let finalSymbols = [];

      for (let i = 0; i < 3; i++) {
        const symbolIndex = Math.floor(Math.random() * symbols.length);
        finalSymbols.push(symbols[symbolIndex]);
        slots[i].textContent = symbols[symbolIndex];
        slots[i].dataset.symbol = symbols[symbolIndex];
      }

      gsap.set(slots, { rotation: 0 });

      checkWin();

      // beenden wenn spins 0 ergeben
      if (spinsLeft <= 0) {
        completeLevel(); //wenn alle spins fertig sind wird das Spiel eben
        setTimeout(endGame, 2000);
      }
    }
  });
}

// Prüfen ob Gewinn
```

## Gsap Animation (Drehen, Text)

```
// Alle gleich?
if (symbols[0] === symbols[1] && symbols[1] === symbols[2]) {
  // Gewinn!
  let winAmount = 0;

  // Je nach Symbol unterschiedliche Beträge
  switch (symbols[0]) {
    case "🍒": winAmount = 10; break;
    case "🍋": winAmount = 50; break;
    case "🍌": winAmount = 20; break;
    case "🍉": winAmount = 100; break;
    case "🍊": winAmount = 30; break;
  }

  document.getElementById("winnings").textContent = `Gewonnen: ${winAmount} Münzen!`;

  // Münzen zum Spielstand hinzufügen
  coinCount += winAmount;
  document.getElementById("coins").textContent = `Münzen: ${coinCount}`;

  // Münzen zum localStorage hinzufügen
  const storedCoins = localStorage.getItem("playerCoins") || "0";
  localStorage.setItem("playerCoins", parseInt(storedCoins) + winAmount);

  // GSAP Animation für den Gewinn
  // Kommentar: GSAP für die Animation des Gewinntextes
  gsap.from("#winnings", {
    scale: 0,
    duration: 0.5,
    ease: "back.out(1.7)"
  });
} else {
  document.getElementById("winnings").textContent = "Leider kein Gewinn.";
}
```

```
function endGame() {
  gamePhase = "gameOver";
  clearInterval(timerInterval);

  // Erfahrung berechnen basierend auf Zeit und verbleibenden Leben und items
  const elapsedSeconds = Math.floor((Date.now() - startTime) / 1000);
  const timePoints = Math.max(0, 60 - elapsedSeconds);
  const lifePoints = lives * 3;
  const itemPoints = itemCount * 0.5;

  // Erfahrung zwischen 1 und 20 begrenzen!
  experience = Math.min(20, Math.max(1, Math.floor(timePoints + lifePoints + itemPoints)));

  // Erfahrung zum localStorage hinzufügen
  const storedXP = localStorage.getItem("playerXP") || "0";
  localStorage.setItem("playerXP", parseInt(storedXP) + experience);

  // Markiere das Level als abgeschlossen, wenn genügend Items gesammelt wurden
  if (itemCount >= 15) {
    completeLevel();
  }

  updatePlayerProgress();

  // Game Over Anzeige anzeigen
  document.getElementById("finalScore").textContent = `Items gesammelt: ${itemCount}/15`;
  document.getElementById("finalCoins").textContent = `Münzen gesammelt: ${coinCount}`;
  document.getElementById("finalExperience").textContent = `Erfahrung erhalten: ${experience}`;

  document.getElementById("gameOver").style.display = "block";
}
```



# Level 2 - Memory

---

---

Kartenspiel

---

Bei gleichen Symbol verschwinden die Karten

---

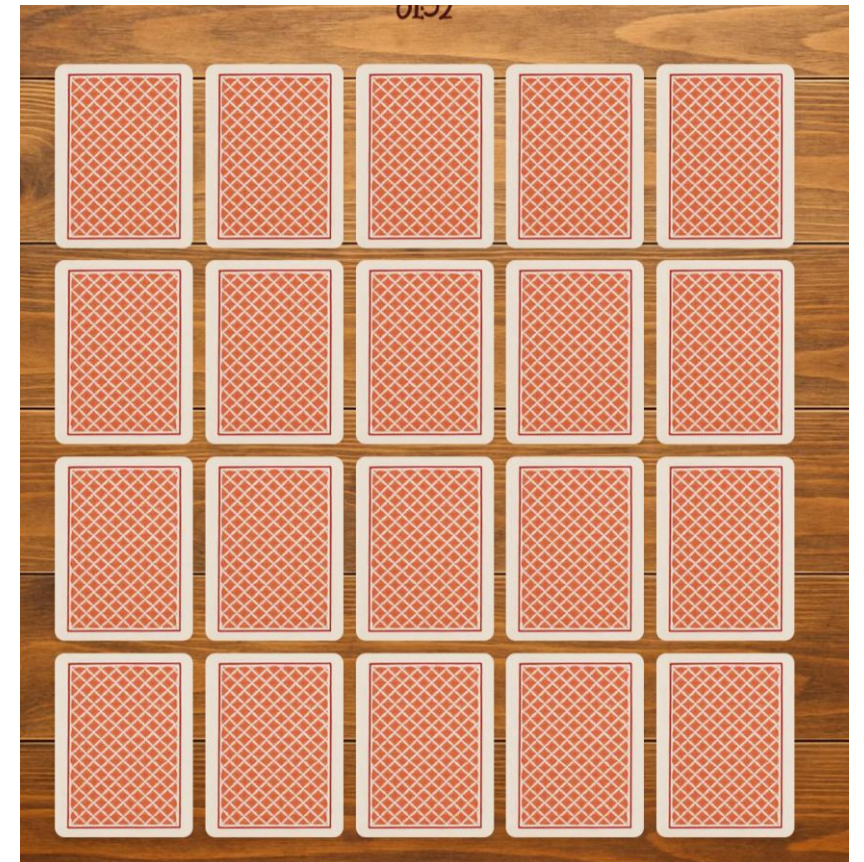
Erfahrung wird berechnet durch Versuche und Zeit

---

Hinterseite (Karten Img)

---

Vorderseite (Weiß mit Symbol)



# Logik + Animation

```
.match-animation {  
  animation: matchFound 0.5s ease-out;  
}  
  
@keyframes matchFound {  
  0% {  
    transform: scale(1);  
    opacity: 1;  
  }  
  50% {  
    transform: scale(1.2);  
    opacity: 0.8;  
  }  
  100% {  
    transform: scale(0);  
    opacity: 0;  
  }  
}
```

```
// Prüfen, ob Karten übereinstimmen  
function checkForMatch() {  
  const isMatch = firstCard.dataset.value === secondCard.dataset.value;  
  
  if (isMatch) {  
    setTimeout(() => {  
      firstCard.classList.add('match-animation');  
      secondCard.classList.add('match-animation');  
  
      setTimeout(() => {  
        firstCard.style.visibility = 'hidden';  
        secondCard.style.visibility = 'hidden';  
        resetBoard();  
        pairsFound++;  
  
        if (pairsFound === emojis.length) {  
          clearInterval(timer);  
          endGame(true);  
        }  
      }, 500);  
    }, 500);  
  } else {  
    lockBoard = true;  
    setTimeout(() => {  
      firstCard.classList.remove('flipped');  
      secondCard.classList.remove('flipped');  
      resetBoard();  
    }, 1000);  
  }  
}
```



# Level 3 – Time Guess

- Vorbereitungszone
- Count wird angezeigt
- Bestätigen mit Leertaste
- 3 Level
- Abweichung von +- 0.5 Sekunden
- Local-storage Aktualisierung (Ordner Struktur)

```
function updateProgress() {  
  try {  
    // Extrahiere die Level-Nummer aus der URL  
    const pathSegments = window.location.pathname.split('/');  
    const gameFolder = pathSegments[pathSegments.length - 2]; // z.B. "game1"  
    const levelNumber = parseInt(gameFolder.replace('game', '')) || 1;  
  
    // Markieren das das Level erfolgreich weurd  
    localStorage.setItem(`level${levelNumber}Completed`, 'true');  
  
    console.log(`Level ${levelNumber} als abgeschlossen markiert!`);  
  
  } catch (error) {  
    console.error('Fehler beim Speichern des Fortschritts:', error);  
  }  
}
```

# Level 3 - Code

---

- -> drawTimer wird regelmäßig ausgeführt
- -> Aktuellen Zeitstempel in Ms
- -> Zeit nach Start
- -> Berechnung der verbleibenden Ms bis Ende
- -> Übrige Countdown wird als Text ausgegeben

```
function endGame(win) {
  cancelAnimationFrame(animationFrameId);
  instructions.style.display = 'none';
  timerDisplay.style.display = 'none';

  if (win) {
    message.style.color = '#0f0';
    message.textContent = `Glückwunsch! Alle Level geschafft!`;
    // Gambling Maschine zeigen
    gamblingMachine.style.display = 'block';
    spinsLeft = 5;
    spinsLeftDisplay.textContent = `Versuche übrig: ${spinsLeft}`;
    winnings.textContent = '';

    // Level im LocalStorage aktualisieren
    updateProgress();
  } else {
    message.style.color = '#f00';
    message.textContent = `Leider verloren! Du warst zu weit weg.`;
    // Button zum neu starten zeigen
    setTimeout(() => {
      startScreen.style.display = 'flex';
      levelStatus.style.display = 'none';
    }, 3000);
  }
}
```

```
function drawTimer() {
  animationFrameId = requestAnimationFrame(drawTimer);
  const now = performance.now();
  let elapsed = now - countdownStart;

  if (timerPhase === 'showing') {
    let timeLeft = Math.max(0, countdownDuration - elapsed);
    timerDisplay.textContent = (timeLeft / 1000).toFixed(2);
  }
}
```



# Level 4 – Color Game

- Farben müssen in der Höhle an der richtigen Position gebracht werden.
- Gsap für Countdown

```
function showCountdown() {
  const counter = document.getElementById("counter");
  counter.style.display = "block";
  let count = 3;

  const countdownInterval = setInterval(() => {
    counter.textContent = count;

    // GSAP Animation für Countdown
    gsap.fromTo(counter,
      { scale: 0, opacity: 0 },
      { scale: 1.2, opacity: 1, duration: 0.3, ease: "back.out(1.7)" }
    );

    gsap.to(counter, {
      scale: 0.8,
      opacity: 0.5,
      duration: 0.7,
      delay: 0.3
    });

    count--;

    if (count < 0) {
      clearInterval(countdownInterval);
      counter.textContent = "START!";

      gsap.fromTo(counter,
        { scale: 0, opacity: 0 },
        { scale: 1.5, opacity: 1, duration: 0.5, ease: "back.out(1.7)" }
      );

      setTimeout(() => {
        counter.style.display = "none";
        actuallyStartGame();
      }, 1000);
    }
  }, 1000);
}
```

## Level 5 – Final Spin

- Slot aus 3 Zeilen und 4 Reihen
- Gewinne nur bei 3 gleichen Reihen
- Gewinne je nach Symbol verschieden
- Ziel ist es mit 50 Coins Gewinn rauszukommen (sehr schwer)

```
function createSlots() {  
    const grid = document.getElementById('slotsGrid');  
    grid.innerHTML = '';  
  
    for (let i = 0; i < 12; i++) {  
        const slot = document.createElement('div');  
        slot.className = 'slot';  
        slot.id = `slot${i}`;  
        slot.textContent = '🎯';  
        grid.appendChild(slot);  
    }  
}
```

# Abschluss

