<!DOCTYPE html>

<html lang="ja">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>スライドルーレット</title>

    <!-- Tailwind CSS CDN -->

    <script src="[https://cdn.tailwindcss.com](https://cdn.tailwindcss.com/)"></script>

    <link href="[https://fonts.googleapis.com/css2?family=Inter:wght@400;700&display=swap](https://fonts.googleapis.com/css2?family=Inter%3Awght%40400%3B700&display=swap)" rel="stylesheet">

    <style>

        body {

            font-family: 'Inter', sans-serif;

            background-color: #f0f4f8; /\* ライトブルーグレーの背景 \*/

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            margin: 0;

            padding: 20px;

            box-sizing: border-box;

        }

        .container {

            background-color: #ffffff; /\* 白いコンテナ \*/

            border-radius: 20px;

            box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);

            padding: 40px; /\* パディングを少し大きく \*/

            text-align: center;

            width: 100%;

            max-width: 900px; /\* 最大幅を拡大 \*/

            display: flex;

            flex-direction: column;

            align-items: center;

            gap: 25px; /\* ギャップも少し大きく \*/

            overflow: hidden; /\* 子要素がはみ出さないように \*/

        }

        .item-list-wrapper {

            width: 100%;

            overflow: hidden; /\* スクロールバーを隠す \*/

            position: relative;

            border-radius: 15px; /\* 角を少し丸く \*/

            background-color: #e2e8f0; /\* リストの背景色 \*/

            padding: 15px 0; /\* パディングを少し大きく \*/

            box-shadow: inset 0 2px 5px rgba(0,0,0,0.05);

        }

        /\* カスタムスクロールバー非表示（Webkit系） \*/

        .item-list-wrapper::-webkit-scrollbar {

            display: none;

        }

        .item-list-wrapper {

            -ms-overflow-style: none;  /\* IE and Edge \*/

            scrollbar-width: none;  /\* Firefox \*/

        }

        .item-list {

            display: flex;

            flex-wrap: nowrap; /\* 折り返しを禁止 \*/

            overflow-x: scroll; /\* 横スクロールを有効に \*/

            scroll-behavior: smooth; /\* スムーズスクロール \*/

            -webkit-overflow-scrolling: touch; /\* iOSでのスムーズスクロール \*/

            padding: 0 calc(50% - 100px); /\* 両端のアイテムが中央に来るように調整 (アイテム幅の半分を引く) \*/

            box-sizing: border-box;

            will-change: transform; /\* アニメーションのパフォーマンス最適化 \*/

        }

        .item {

            flex: 0 0 auto; /\* アイテムのサイズ固定 \*/

            width: 200px; /\* アイテムの幅を拡大 \*/

            height: 200px; /\* アイテムの高さを拡大 \*/

            display: flex;

            flex-direction: column;

            justify-content: center;

            align-items: center;

            background-color: #ffffff;

            border-radius: 20px; /\* 角を丸く \*/

            margin: 0 15px; /\* マージンを少し大きく \*/

            box-shadow: 0 6px 15px rgba(0, 0, 0, 0.08); /\* 影を強調 \*/

            transition: transform 0.2s ease-in-out;

            cursor: default;

            user-select: none; /\* テキスト選択を無効化 \*/

            pointer-events: none; /\* スピン中は手動スクロールさせない \*/

        }

        .item-emoji {

            font-size: 5rem; /\* 絵文字のサイズを拡大 \*/

            line-height: 1;

            margin-bottom: 8px; /\* マージンを少し大きく \*/

        }

        .item-name {

            font-size: 1.4rem; /\* フォントサイズを拡大 \*/

            font-weight: 700; /\* フォントウェイトを強調 \*/

            color: #333;

            white-space: nowrap; /\* テキストの折り返しを禁止 \*/

        }

        .spin-button, .reset-button {

            background: linear-gradient(145deg, #6366f1, #4f46e5); /\* グラデーションボタン \*/

            color: white;

            padding: 18px 35px; /\* パディングを大きく \*/

            border: none;

            border-radius: 35px; /\* 角を丸く \*/

            font-size: 1.4rem; /\* フォントサイズを拡大 \*/

            font-weight: 700;

            cursor: pointer;

            box-shadow: 0 6px 18px rgba(80, 70, 229, 0.45); /\* 影を強調 \*/

            transition: all 0.3s ease;

            outline: none;

            width: 100%;

            max-width: 300px; /\* ボタンの最大幅を拡大 \*/

        }

        .spin-button:hover, .reset-button:hover {

            transform: translateY(-4px); /\* ホバー時の浮き上がりを大きく \*/

            box-shadow: 0 10px 25px rgba(80, 70, 229, 0.6);

        }

        .spin-button:active, .reset-button:active {

            transform: translateY(0);

            box-shadow: 0 4px 12px rgba(80, 70, 229, 0.4);

        }

        .spin-button:disabled, .reset-button:disabled {

            background: #aeb5ed; /\* 無効時の色 \*/

            cursor: not-allowed;

            box-shadow: none;

            transform: none;

        }

        .reset-button {

            background: linear-gradient(145deg, #ef4444, #dc2626); /\* リセットボタンは赤系のグラデーション \*/

            box-shadow: 0 6px 18px rgba(220, 38, 38, 0.45);

        }

        .reset-button:hover {

            box-shadow: 0 10px 25px rgba(220, 38, 38, 0.6);

        }

        .reset-button:active {

            box-shadow: 0 4px 12px rgba(220, 38, 38, 0.4);

        }

        .buttons-container {

            display: flex;

            flex-direction: column;

            gap: 15px; /\* ボタン間のスペース \*/

            width: 100%;

            max-width: 300px;

        }

        #result {

            font-size: 2rem; /\* 結果フォントサイズを拡大 \*/

            font-weight: 800; /\* フォントウェイトを強調 \*/

            color: #1a202c; /\* ダークグレー \*/

            margin-top: 15px; /\* マージンを大きく \*/

            min-height: 40px; /\* 結果表示エリアの最小高さを確保 \*/

            transition: opacity 0.5s ease-in-out;

        }

        /\* 画面が狭い場合の調整 \*/

        @media (max-width: 768px) { /\* タブレットとモバイルのブレークポイントを調整 \*/

            .container {

                padding: 25px;

                gap: 20px;

            }

            .item {

                width: 150px; /\* モバイルでのアイテム幅を調整 \*/

                height: 150px;

                margin: 0 10px;

            }

            .item-emoji {

                font-size: 4rem;

            }

            .item-name {

                font-size: 1.2rem;

            }

            .spin-button, .reset-button {

                padding: 15px 30px;

                font-size: 1.25rem;

                max-width: 250px;

            }

            #result {

                font-size: 1.6rem;

            }

            .item-list {

                padding: 0 calc(50% - 75px); /\* モバイル向け調整 (アイテム幅の半分を引く) \*/

            }

            .buttons-container {

                max-width: 250px;

            }

        }

        /\* 中央のポインター \*/

        .center-pointer {

            position: absolute;

            top: 0;

            bottom: 0;

            left: 50%;

            width: 2px; /\* 細い線 \*/

            background-color: #ef4444; /\* 赤い線 \*/

            transform: translateX(-50%);

            z-index: 10;

            box-shadow: 0 0 5px rgba(239, 68, 68, 0.5); /\* 影 \*/

        }

    </style>

</head>

<body>

    <div class="container">

        <h1 class="text-3xl font-bold text-gray-800 mb-4">野菜と果物ルーレット</h1>

        <div class="item-list-wrapper">

            <div class="item-list" id="itemList">

                <!-- アイテムはJavaScriptでここに挿入されます -->

            </div>

            <div class="center-pointer"></div>

        </div>

        <div class="buttons-container">

            <button id="spinButton" class="spin-button">スタート！</button>

            <button id="resetButton" class="reset-button" disabled>リセット</button>

        </div>

        <p id="result" class="text-xl font-semibold text-gray-700 mt-4">何が当たるかな？</p>

    </div>

    <script>

        const itemListContainer = document.getElementById('itemList');

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

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

        const resultDisplay = document.getElementById('result');

        const itemListWrapper = document.querySelector('.item-list-wrapper');

        // Web Audio APIのコンテキストを作成

        const audioContext = new (window.AudioContext || window.webkitAudioContext)();

        // ルーレットの項目データ

        const items = [

            { name: 'ナス', emoji: '🍆' },

            { name: 'トマト', emoji: '🍅' },

            { name: 'ピーマン', emoji: '🫑' },

            { name: 'ニンジン', emoji: '🥕' },

            { name: 'キャベツ', emoji: '🥬' },

            { name: 'ジャガイモ', emoji: '🥔' },

            { name: 'かぼちゃ', emoji: '🎃' },

            { name: 'とうもろこし', emoji: '🌽' },

            { name: 'リンゴ', emoji: '🍎' },

            { name: 'ぶどう', emoji: '🍇' },

            { name: 'バナナ', emoji: '🍌' },

            { name: 'スイカ', emoji: '🍉' },

            { name: 'レモン', emoji: '🍋' },

            { name: '玉ねぎ', emoji: '🧅' },

            { name: 'さつまいも', emoji: '🍠' },

            { name: '梨', emoji: '🍐' },

            { name: '桃', emoji: '🍑' },

            { name: 'キュウリ', emoji: '🥒' }

        ];

        const spinTimeTotal = 10000; // スピンの合計時間（ミリ秒）

        let spinning = false; // スピン中かどうかのフラグ

        let selectedItemIndices = []; // 選ばれた項目の元のインデックスを格納する配列

        // キャッシュする変数

        let cachedItemWidthWithMargin = 0;

        let cachedInitialScrollOffset = 0;

        /\*\*

         \* ルーレットの項目を動的に生成し、HTMLに挿入します。

         \* アイテムリストに、項目を複数回追加することで、スライド時に連続性を持たせます。

         \* これにより、スピン中にアイテムが途切れることなく流れるように見えます。

         \*/

        function createRouletteItems() {

            // スムーズなスピンのために、項目を複数回（例えば5回）繰り返して追加します

            const repeatCount = 5; // リストの繰り返し回数

            itemListContainer.innerHTML = ''; // 既存のアイテムをクリア

            for (let r = 0; r < repeatCount; r++) {

                items.forEach((item, index) => {

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

                    itemDiv.classList.add('item');

                    itemDiv.dataset.originalIndex = index; // 元のアイテムのインデックスを保存

                    const emojiSpan = document.createElement('span');

                    emojiSpan.classList.add('item-emoji');

                    emojiSpan.textContent = item.emoji;

                    const nameSpan = document.createElement('span');

                    nameSpan.classList.add('item-name');

                    nameSpan.textContent = [item.name](http://item.name/);

                    itemDiv.appendChild(emojiSpan);

                    itemDiv.appendChild(nameSpan);

                    itemListContainer.appendChild(itemDiv);

                });

            }

        }

        /\*\*

         \* スピン終了時の効果音を再生します。

         \* 高周波の短い音を生成します。

         \*/

        function playSpinEndSound() {

            const oscillator = audioContext.createOscillator();

            const gainNode = audioContext.createGain();

            oscillator.type = 'triangle'; // 三角波

            oscillator.frequency.setValueAtTime(800, audioContext.currentTime); // 高い周波数

            oscillator.connect(gainNode);

            gainNode.connect(audioContext.destination);

            gainNode.gain.setValueAtTime(0.5, audioContext.currentTime);

            gainNode.gain.exponentialRampToValueAtTime(0.001, audioContext.currentTime + 0.5); // フェードアウト

            oscillator.start(audioContext.currentTime);

            oscillator.stop(audioContext.currentTime + 0.5); // 短く再生して停止

        }

        /\*\*

         \* スピンアニメーションを開始します。

         \* ランダムな項目を選択し、その項目が中央に来るようにリストをスクロールさせます。

         \*/

        function spin() {

            if (spinning) return;

            // 未選択のアイテムをフィルタリング

            const availableItems = items.filter((\_, index) => !selectedItemIndices.includes(index));

            if (availableItems.length === 0) {

                resultDisplay.textContent = '全ての項目が選ばれました！リセットしてください。';

                spinButton.disabled = true;

                resetButton.disabled = false;

                return;

            }

            spinning = true;

            spinButton.disabled = true;

            resetButton.disabled = true; // スピン中はリセットボタンも無効化

            itemListWrapper.style.pointerEvents = 'none'; // スピン中は手動スクロールを無効化

            resultDisplay.style.opacity = '0'; // 結果を一時的に非表示にする

            // キャッシュされた値を使用

            itemListContainer.scrollLeft = cachedInitialScrollOffset; // キャッシュされた初期位置に設定

            let startScrollLeft = itemListContainer.scrollLeft;

            // 利用可能なアイテムの中からランダムに選択

            const selectedAvailableItem = availableItems[Math.floor(Math.random() \* availableItems.length)];

            const selectedOriginalIndex = items.indexOf(selectedAvailableItem); // 元のitems配列でのインデックスを取得

            // 選ばれた項目を追跡リストに追加

            selectedItemIndices.push(selectedOriginalIndex);

            // スピンの目標位置を計算

            // リストの最後の繰り返しセットの中から、ランダムに選ばれたアイテムの位置を目指します。

            const targetSetIndex = (Math.floor(itemListContainer.children.length / items.length) - 1); // 最後の繰り返しセット

            const targetIndexInFullList = targetSetIndex \* items.length + selectedOriginalIndex; // 最後のセットにある選択されたアイテムのインデックス

            const targetItem = itemListContainer.children[targetIndexInFullList];

            if (!targetItem) {

                console.error("目標アイテムが見つかりません。スピンできません。");

                spinning = false;

                spinButton.disabled = false;

                itemListWrapper.style.pointerEvents = 'auto';

                updateButtonStates(); // ボタンの状態を適切に更新

                return;

            }

            const itemStyle = getComputedStyle(targetItem);

            const itemTotalWidth = parseFloat(itemStyle.width) + parseFloat(itemStyle.marginLeft) + parseFloat(itemStyle.marginRight);

            // 目標アイテムがラッパーの中央に来るように計算

            const targetScrollLeft = targetItem.offsetLeft - (itemListWrapper.offsetWidth / 2) + (itemTotalWidth / 2);

            let startTime = null;

            function animateSpin(currentTime) {

                if (!startTime) startTime = currentTime;

                const elapsed = currentTime - startTime;

                const progress = Math.min(elapsed / spinTimeTotal, 1);

                // イーズアウトカーブ（徐々に減速）

                const easing = 1 - Math.pow(1 - progress, 3); // CUBIC EASE OUT

                // スクロール位置を計算

                const currentScrollLeft = startScrollLeft + (targetScrollLeft - startScrollLeft) \* easing;

                itemListContainer.scrollLeft = currentScrollLeft;

                if (progress < 1) {

                    requestAnimationFrame(animateSpin);

                } else {

                    // アニメーション終了

                    spinning = false;

                    itemListWrapper.style.pointerEvents = 'auto';

                    playSpinEndSound(); // 終了音を再生

                    showResult([selectedAvailableItem.name](http://selectedavailableitem.name/)); // 選ばれたアイテムの名前を表示

                    updateButtonStates(); // ボタンの状態を更新

                }

            }

            requestAnimationFrame(animateSpin);

        }

        /\*\*

         \* スピンの結果を表示します。

         \* @param {string} resultName - 当たったアイテムの名前

         \*/

        function showResult(resultName) {

            resultDisplay.textContent = `結果: ${resultName}！`;

            resultDisplay.style.opacity = '1';

        }

        /\*\*

         \* ボタン（スタート、リセット）の状態を更新します。

         \* 全てのアイテムが選ばれたかどうかによって、スタートボタンの有効/無効を切り替えます。

         \*/

        function updateButtonStates() {

            if (selectedItemIndices.length === items.length) {

                spinButton.disabled = true;

                resultDisplay.textContent = '全ての項目が選ばれました！リセットしてください。';

            } else {

                spinButton.disabled = false;

            }

            // 選択されたアイテムがある場合はリセットボタンを有効にする

            resetButton.disabled = selectedItemIndices.length === 0;

        }

        /\*\*

         \* 選ばれた項目をリセットし、全てを再度選べるようにします。

         \*/

        function resetRoulette() {

            selectedItemIndices = []; // 選択済みリストをクリア

            resultDisplay.textContent = '何が当たるかな？';

            resultDisplay.style.opacity = '1';

            updateButtonStates(); // ボタンの状態を更新

            // ルーレットを初期位置に戻す（キャッシュされた初期位置を使用）

            itemListContainer.scrollLeft = cachedInitialScrollOffset;

        }

        /\*\*

         \* ページの読み込み時とリサイズ時にルーレットを初期化します。

         \* アイテムを生成し、リストを中央にスクロールします。

         \*/

        function initializeRoulette() {

            createRouletteItems(); // アイテムを生成

            updateButtonStates(); // 初期状態のボタン設定

            const itemElement = document.querySelector('.item');

            if (itemElement) {

                // ここで計算し、グローバル変数にキャッシュ

                cachedItemWidthWithMargin = itemElement.offsetWidth + parseFloat(getComputedStyle(itemElement).marginLeft) + parseFloat(getComputedStyle(itemElement).marginRight);

                cachedInitialScrollOffset = (Math.floor(itemListContainer.children.length / items.length / 2)) \* items.length \* cachedItemWidthWithMargin;

                itemListContainer.scrollLeft = cachedInitialScrollOffset;

            }

        }

        // 初期設定

        window.onload = initializeRoulette;

        // スピンボタンのクリックイベントリスナー

        spinButton.addEventListener('click', spin);

        // リセットボタンのクリックイベントリスナー

        resetButton.addEventListener('click', resetRoulette);

        // ウィンドウのリサイズ時にアイテムを再生成し、初期位置を調整

        window.addEventListener('resize', initializeRoulette);

    </script>

</body>

</html>