

Разработка образовательной игры «Висельница» на HTML, CSS и JavaScript: методика поэтапного создания и интеграции игровых механик

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Статья посвящена разработке интерактивной веб-версии классической игры «Висельница» с использованием современных фронтенд-технологий: HTML5 для семантической структуры, CSS3 для адаптивного дизайна и анимаций, а также JavaScript для реализации игровой логики. Подробно рассматривается поэтапное создание приложения — от проектирования SVG-графики висельницы до разработки виртуальной клавиатуры с динамической обработкой событий.

Ключевые слова: веб-разработка, HTML5, CSS3, JavaScript, интерактивные игры, образовательные технологии.

Development of the educational game "Hangman" in HTML, CSS and JavaScript: a method for step-by-step creation and integration of game mechanics

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The article is devoted to the development of an interactive web version of the classic Hangman game using modern frontend technologies: HTML5 for semantic structure, CSS3 for responsive design and animations, as well as JavaScript for implementing game logic. The step-by-step creation of the application is considered in detail, from designing the SVG graphics of the gallows to developing a virtual keyboard with dynamic event processing.

Keywords: web development, HTML5, CSS3, JavaScript, interactive games, educational technologies.

1 Введение

1.1 Актуальность

В современном мире, где цифровые технологии активно проникают в сферу образования, интерактивные игры становятся эффективным инструментом для обучения. Разработка веб-приложений, сочетающих игровые механики с образовательной составляющей, особенно востребована в

условиях роста популярности онлайн-обучения и необходимости повышения мотивации учащихся.

1.2 Обзор исследований

В статье С.В. Рогатина, В.А. Антошкина рассматривается методика разработки компьютерных игр по технологии HTML5 с помощью CASE-средства GameMakerStudio компании YoYoGamesLtd. Проводится тестирование производительности работы компьютерных игр в браузере в зависимости от сложности использованных спрайтов игровых объектов [1]. Разработали игровое приложение «league of football heads» с использованием html5, css3, javascript И.А. Фёдоров, Е.Ю. Кузьменкова[2]. Королук Я. А., Куликова Н. Ю. показали опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике [3]. В работе рассмотрели возможности языка разметки гипертекста HTML5 для создания макета электронного учебника Д.М. Размологов, А.С. Панина, П.В. Захаров. Привели разработанный макет, а также часть таблицы стилей для дизайна макета [4]. И.С. Шегаев, А.Г. Полозов разработали новейший политологический глоссарий: основные категории, понятия, термины [5].

1.3 Цель исследования

Целью исследования является разработка веб-игры «Висельница» с использованием HTML, CSS и JavaScript.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовым редактор Notepad++, оптимизированным для языков программирования.

3 Результаты

Открываем Notepad++ и создаём файл index.html, для этого в меню выбираем файл → новый. Далее нажимаем сочетание клавиш "Ctrl+S" и в поле "Имя файла" вводим название файла "index" с расширением .html, сохраняем данный файл на рабочем столе.

Создадим основную структуру HTML-документа для игры "Висельница": объявляем <!DOCTYPE html> и открываем корневой тег <html> с атрибутом lang="ru". В разделе <head> указываем кодировку UTF-8, настройки адаптивности через <meta name="viewport">, заголовок страницы <title> и пустой блок <style> для будущих CSS-стилей. В теле документа (<body>) размещаем контейнер игры (game-container), содержащий заголовок <h1>, блок для отображения висельницы (hangman-container), выпадающий список выбора категорий слов (category-select с четырьмя вариантами), поле для отображения загаданного слова (word-display), блок сообщений (message), панель игровой информации (счётчик ошибок и очков), виртуальную

клавиатуру (keyboard) и кнопку перезапуска игры. В конце документа добавлен пустой тег <script> для будущего JavaScript-кода. Весь интерфейс свёрстан с использованием семантических HTML-тегов и классов для дальнейшего стилового оформления.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Висельница</title>
  <style>

  </style>
</head>
<body>
  <div class="game-container">
    <h1>Висельница</h1>

    <div class="hangman-container">

    </div>

    <div class="category-select">
      <select id="category-select">
        <option value="animals">Животные</option>
        <option value="fruits">Фрукты и овощи</option>
        <option value="countries">Страны</option>
        <option value="professions">Профессии</option>
      </select>
    </div>

    <div class="word-display" id="word-display"></div>

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

    <div class="game-info">
      <div class="info-box">Ошибки: <span id="mistakes">0</span>/6</div>
      <div class="info-box">Счет: <span id="score">0</span></div>
    </div>

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

    <button id="restart-btn">Новая игра</button>
  </div>

  <script>

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

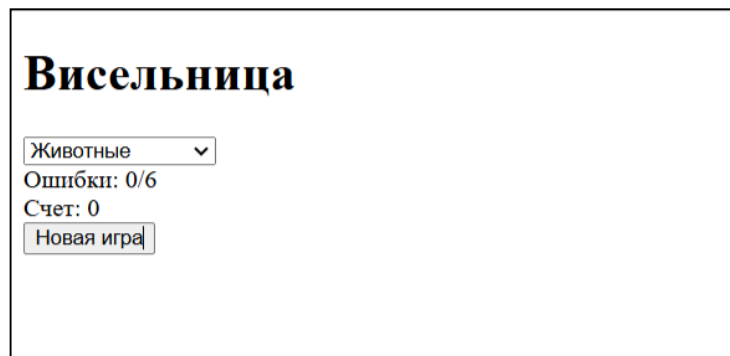


Рисунок 1. Основная структура HTML-документа

Добавим стили: задаем для `body` плавный градиентный фон, центрируем содержимое с помощью `Flexbox` и устанавливаем минимальную высоту в `100vh`. Контейнер игры (`game-container`) оформлен белым фоном, закругленными углами и тенью для эффекта "карточки". Заголовок `h1` стилизован фиолетовым цветом с легкой тенью. Блок `hangman-container` содержит серый фон с внутренней тенью для имитации доски. Виселица (`hangman-drawing`) рисуется SVG-линиями с плавными переходами. Отображение слова (`word-display`) использует большой шрифт с широкими пробелами между буквами, а каждая буква (`letter`) имеет анимированное подчеркивание при угадывании. Виртуальная клавиатура (`keyboard`) построена на `Flexbox` с адаптивными кнопками (`key`), которые меняют цвет при правильном/неправильном выборе и имеют эффекты наведения. Игровая информация (`game-info`) представлена в виде карточек с градиентным фоном. Сообщения (`message`) меняют цвет при победе/поражении. Кнопка перезапуска и выпадающий список стилизованы с градиентами, тенями и `hover`-эффектами. Добавлены медиазапросы для мобильных устройств, уменьшающие размеры элементов, а также анимации (`bounce`, `shake`) для визуальной обратной связи. Все переходы (`transition`) и трансформации (`transform`) создают плавные интерактивные эффекты.

Данный код пропишем на следующей строчке после тега `<title>` Висельница `</title>` в секции `<head>` HTML-документа.

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  margin: 0;
  padding: 20px;
  color: #2d3436;
}

.game-container {
  background-color: white;
  border-radius: 15px;
  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
  padding: 30px;
```

```
max-width: 800px;
width: 100%;
text-align: center;
transition: all 0.3s ease;
}

h1 {
color: #6c5ce7;
margin-bottom: 25px;
font-size: 2.5rem;
text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
}

.hangman-container {
margin: 25px auto;
position: relative;
width: 300px;
height: 250px;
background-color: #f9f9f9;
border-radius: 10px;
padding: 15px;
box-shadow: inset 0 0 10px rgba(0, 0, 0, 0.05);
}

.hangman-drawing {
stroke: #636e72;
stroke-width: 4;
stroke-linecap: round;
fill: none;
filter: drop-shadow(1px 1px 2px rgba(0,0,0,0.1));
}

.hidden-part {
opacity: 0;
transition: opacity 0.3s ease;
}

.word-display {
font-size: 2.2rem;
letter-spacing: 12px;
margin: 35px 0;
font-weight: bold;
min-height: 60px;
color: #2d3436;
text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
}

.letter {
display: inline-block;
margin: 0 6px;
position: relative;
transition: all 0.3s ease;
}

.letter::after {
content: "";
position: absolute;
bottom: -5px;
left: 0;
width: 100%;
height: 3px;
background-color: #6c5ce7;
transform: scaleX(0);
```

```
        transition: transform 0.3s ease;
    }

    .letter.revealed::after {
        transform: scaleX(1);
    }

    .keyboard {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        gap: 10px;
        margin: 25px 0;
    }

    .key {
        width: 45px;
        height: 45px;
        font-size: 1.3rem;
        font-weight: bold;
        background: linear-gradient(145deg, #a8e6cf, #dcedc1);
        color: #2d3436;
        border: none;
        border-radius: 8px;
        cursor: pointer;
        transition: all 0.2s;
        box-shadow: 0 3px 6px rgba(0,0,0,0.1);
    }

    .key:hover:not(:disabled) {
        transform: translateY(-3px);
        box-shadow: 0 5px 10px rgba(0,0,0,0.15);
        background: linear-gradient(145deg, #dcedc1, #a8e6cf);
    }

    .key:active:not(:disabled) {
        transform: translateY(1px);
    }

    .key:disabled {
        color: white;
        cursor: not-allowed;
        opacity: 0.8;
        box-shadow: none;
    }

    .key.wrong {
        background: linear-gradient(145deg, #ffaaa5, #ff8b94);
        color: white;
    }

    .key.correct {
        background: linear-gradient(145deg, #a8e6cf, #6c5ce7);
        color: white;
    }

    .game-info {
        display: flex;
        justify-content: space-around;
        margin: 25px 0;
        gap: 20px;
    }
```

```
.info-box {
  background: linear-gradient(145deg, #ffffff, #f1f1f1);
  padding: 12px 25px;
  border-radius: 8px;
  font-size: 1.1rem;
  box-shadow: 0 3px 6px rgba(0,0,0,0.05);
  flex: 1;
  max-width: 200px;
}

.info-box span {
  font-weight: bold;
  color: #6c5ce7;
  font-size: 1.3rem;
}

.message {
  margin: 25px 0;
  font-size: 1.2rem;
  min-height: 27px;
  font-weight: bold;
  transition: all 0.3s ease;
}

.message.win {
  color: #00b894;
}

.message.lose {
  color: #d63031;
}

button {
  background: linear-gradient(145deg, #6c5ce7, #a29bfe);
  color: white;
  border: none;
  padding: 14px 30px;
  font-size: 1.1rem;
  border-radius: 50px;
  cursor: pointer;
  transition: all 0.3s;
  margin-top: 15px;
  box-shadow: 0 5px 15px rgba(108, 92, 231, 0.3);
  font-weight: bold;
  letter-spacing: 1px;
}

button:hover {
  transform: translateY(-3px);
  box-shadow: 0 8px 20px rgba(108, 92, 231, 0.4);
  background: linear-gradient(145deg, #a29bfe, #6c5ce7);
}

button:active {
  transform: translateY(1px);
}

.category-select {
  margin: 20px 0;
}

select {
  padding: 10px 20px;
```

```
border-radius: 50px;
border: 2px solid #dfe6e9;
font-size: 1rem;
background-color: white;
cursor: pointer;
transition: all 0.3s;
outline: none;
box-shadow: 0 3px 6px rgba(0,0,0,0.05);
}

select:hover {
  border-color: #a8e6cf;
}

select:focus {
  border-color: #6c5ce7;
}

@media (max-width: 600px) {
  .game-container {
    padding: 20px;
  }

  h1 {
    font-size: 2rem;
  }

  .hangman-container {
    width: 250px;
    height: 200px;
    padding: 10px;
  }

  .word-display {
    font-size: 1.8rem;
    letter-spacing: 8px;
    margin: 25px 0;
  }

  .key {
    width: 35px;
    height: 35px;
    font-size: 1.1rem;
  }

  .game-info {
    flex-direction: column;
    align-items: center;
  }

  .info-box {
    max-width: 100%;
    width: 100%;
    margin-bottom: 10px;
  }
}

/* Анимации */
@keyframes bounce {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(-10px); }
}
```

```
.bounce {  
  animation: bounce 0.5s;  
}  
  
@keyframes shake {  
  0%, 100% { transform: translateX(0); }  
  20%, 60% { transform: translateX(-5px); }  
  40%, 80% { transform: translateX(5px); }  
}  
  
.shake {  
  animation: shake 0.5s;  
}
```

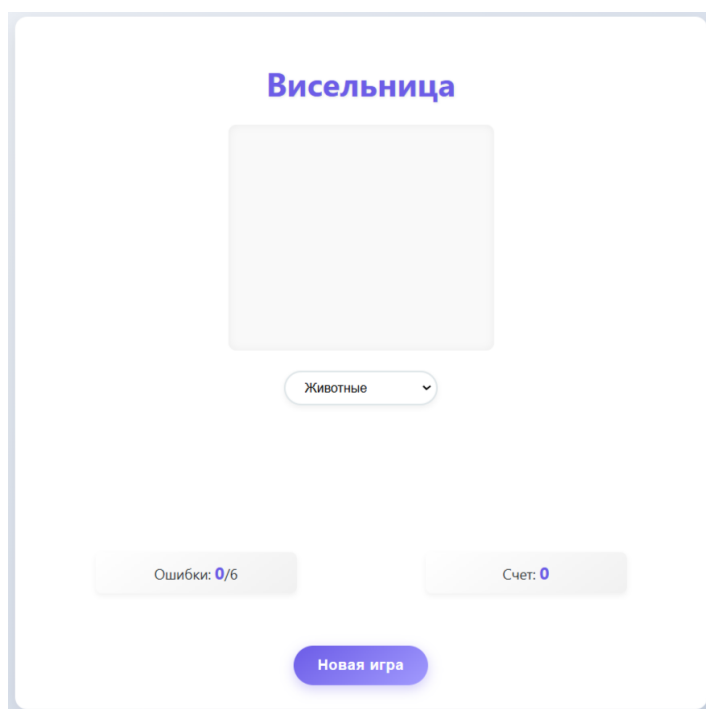


Рисунок 2. Добавление CSS-стилей

Теперь напишем код, который создает SVG-изображение виселицы и человечка для игры: используем элемент `<svg>` с областью просмотра 200×200 пикселей. Основа виселицы состоит из четырех линий (`<line>`): горизонтальной основы (base), вертикальной стойки (pole), верхней перекладины (top) и веревки (rope). Части тела человечка изначально скрыты (класс `hidden-part`) и включают: голову (head) в виде круга (`<circle>`) радиусом 15 пикселей, тело (body) в виде вертикальной линии, руки (left-arm и right-arm) как диагональные линии, и ноги (left-leg и right-leg), также расходящиеся по диагонали. Все элементы имеют общий класс `hangman-part`, задающий стиль линий (цвет, толщину, скругление), а скрытые части (`hidden-part`) имеют нулевую прозрачность, которая будет меняться при ошибках. Координаты каждого элемента точно рассчитаны для создания целостного изображения, где голова расположена ниже веревки, а конечности соединены с телом. Этот SVG-рисунок будет динамически обновляться в процессе игры, показывая новые части тела при неправильных ответах.

Вставьте данный код после открывающего тега `<div class="hangman-container">` и закрывающего тега `</div>`:

```
<svg viewBox="0 0 200 200" class="hangman-drawing">
  <!-- Основа -->
  <line x1="20" y1="180" x2="100" y2="180" class="hangman-part" id="base"/>
  <line x1="40" y1="180" x2="40" y2="20" class="hangman-part" id="pole"/>
  <line x1="40" y1="20" x2="120" y2="20" class="hangman-part" id="top"/>
  <line x1="120" y1="20" x2="120" y2="40" class="hangman-part" id="rope"/>

  <!-- Части тела (изначально скрыты) -->
  <circle cx="120" cy="55" r="15" class="hangman-part hidden-part" id="head"/>
  <line x1="120" y1="70" x2="120" y2="120" class="hangman-part hidden-part" id="body"/>
  <line x1="120" y1="80" x2="90" y2="100" class="hangman-part hidden-part" id="left-arm"/>
  <line x1="120" y1="80" x2="150" y2="100" class="hangman-part hidden-part" id="right-arm"/>
  <line x1="120" y1="120" x2="100" y2="150" class="hangman-part hidden-part" id="left-leg"/>
  <line x1="120" y1="120" x2="140" y2="150" class="hangman-part hidden-part" id="right-leg"/>
</svg>
```

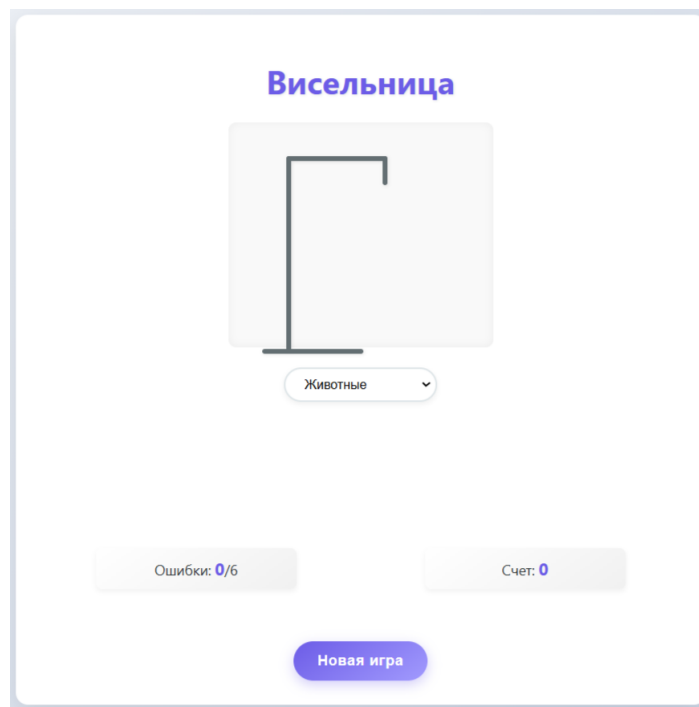


Рисунок 3. Создание изображения виселицы

Далее напишем код, который определяет логику игры: создаем объект `wordsByCategory`, содержащий массивы слов для четырех категорий (животные, фрукты/овощи, страны, профессии), где все слова записаны в верхнем регистре. Получаем ссылки на элементы интерфейса (`wordDisplay`, `keyboard`, `messageEl` и др.) с помощью `getElementById`. Инициализируем игровые переменные: `selectedWord` (загаданное слово), `guessedLetters` (угаданные буквы), `mistakes` (количество ошибок, максимум 6), `score` (счет игрока) и `gameActive` (статус игры). Создаем строку `russianAlphabet`, содержащую все буквы русского алфавита для генерации виртуальной клавиатуры. Определяем массив `hangmanParts`, в котором перечислены ID частей тела человечка (голова, тело, руки, ноги), которые будут последовательно отображаться при ошибках игрока. Этот код формирует

основу для последующей реализации игровой механики, включая выбор слова, обработку ввода букв, отображение состояния игры и обновление интерфейса.

Вставьте данный код после закрывающего тега `</div>` и перед открывающим тегом `<script>`:

```
// Слова по категориям
const wordsByCategory = {
  animals: ['СЛОН', 'ТИГР', 'ЖИРАФ', 'КРОКОДИЛ', 'БЕГЕМОТ', 'ЗЕБРА', 'НОСОРОГ',
'ПАНДА'],
  fruits: ['ЯБЛОКО', 'БАНАН', 'АПЕЛЬСИН', 'ВИНОГРАД', 'АРБУЗ', 'МОРКОВЬ', 'ПОМИДОР',
'ОГУРЕЦ'],
  countries: ['РОССИЯ', 'КИТАЙ', 'БРАЗИЛИЯ', 'КАНАДА', 'ТЕРМАНИЯ', 'ФРАНЦИЯ',
'ИСПАНИЯ', 'ИТАЛИЯ'],
  professions: ['ВРАЧ', 'УЧИТЕЛЬ', 'ПРОГРАММИСТ', 'СТРОИТЕЛЬ', 'ПОВАР', 'ХУДОЖНИК',
'МУЗЫКАНТ', 'ПИЛОТ']
};

// Элементы интерфейса
const wordDisplay = document.getElementById('word-display');
const keyboard = document.getElementById('keyboard');
const messageEl = document.getElementById('message');
const mistakesEl = document.getElementById('mistakes');
const scoreEl = document.getElementById('score');
const restartBtn = document.getElementById('restart-btn');
const categorySelect = document.getElementById('category-select');

// Игровые переменные
let selectedWord = '';
let guessedLetters = [];
let mistakes = 0;
const maxMistakes = 6;
let score = 0;
let gameActive = false;

// Русский алфавит для клавиатуры
const russianAlphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ';

// Части висельницы
const hangmanParts = [
  'head', 'body', 'left-arm', 'right-arm', 'left-leg', 'right-leg'
];
```

Теперь займёмся реализацией функций инициализации игры: функция `initGame()` выбирает случайное слово из выбранной категории (значение берётся из выпадающего списка `categorySelect`), сбрасывает игровые переменные (`guessedLetters`, `mistakes`), активирует игру (`gameActive = true`) и обновляет интерфейс через вспомогательные функции `updateWordDisplay()`, `updateMistakes()` и `updateScore()`. Затем очищается поле сообщений и создаётся виртуальная клавиатура с помощью функции `initKeyboard()`, которая генерирует кнопки для каждой буквы русского алфавита (перебирается строка `russianAlphabet`), назначая им класс `key`, текст с буквой и обработчик клика `handleLetterClick()`. Все части висельницы скрываются через `hideAllHangmanParts()` для подготовки к новой игровой сессии. Таким образом, код обеспечивает полную подготовку игрового поля перед началом

раунда, включая выбор слова, сброс состояния и создание интерактивных элементов управления.

Вставьте данный код в продолжение тега `<script>`:

```
// Инициализация игры
function initGame() {
  const category = categorySelect.value;
  const words = wordsByCategory[category];
  selectedWord = words[Math.floor(Math.random() * words.length)];
  guessedLetters = [];
  mistakes = 0;
  gameActive = true;

  // Обновляем интерфейс
  updateWordDisplay();
  updateMistakes();
  updateScore();
  messageEl.textContent = "";
  messageEl.className = 'message';

  // Создаем клавиатуру
  initKeyboard();

  // Скрываем все части висельницы
  hideAllHangmanParts();
}

// Инициализация клавиатуры
function initKeyboard() {
  keyboard.innerHTML = "";
  for (let letter of russianAlphabet) {
    const key = document.createElement('button');
    key.className = 'key';
    key.textContent = letter;
    key.dataset.letter = letter;
    key.addEventListener('click', () => handleLetterClick(letter));
    keyboard.appendChild(key);
  }
}
```

Реализуем основные игровые механизмы: функция `handleLetterClick(letter)` обрабатывает выбор буквы, проверяя активность игры и состояние кнопки, затем либо добавляет правильную букву в `guessedLetters` с анимацией `bounce`, либо увеличивает счётчик ошибок с анимацией `shake` и отображает часть висельницы. Функция `updateWordDisplay()` динамически формирует отображение слова, показывая угаданные буквы зелёным цветом и скрывая остальные подчёркиваниями. Проверка победы `checkWin()` анализирует, все ли буквы слова угаданы. Вспомогательные функции `updateMistakes()` и `updateScore()` синхронизируют интерфейс с игровым состоянием, а `showMessage()` выводит временные сообщения с автоматическим скрыванием через 2 секунды, применяя стили для победы/поражения. Весь код обеспечивает интерактивное взаимодействие, визуальную обратную связь и контроль игрового прогресса.

Вставьте данный код в продолжение тега `<script>`:

```
// Обработка нажатия на букву
function handleLetterClick(letter) {
    if (!gameActive) return;

    const key = document.querySelector(`.key[data-letter="${letter}"]`);
    if (key.disabled) return;

    key.disabled = true;

    if (selectedWord.includes(letter)) {
        // Правильная буква
        key.classList.add('correct');
        key.classList.add('bounce');
        guessedLetters.push(letter);
        updateWordDisplay();

        // Проверка на победу
        if (checkWin()) {
            endGame(true);
        }
    } else {
        // Неправильная буква
        key.classList.add('wrong');
        key.classList.add('shake');
        mistakes++;
        updateMistakes();
        showHangmanPart();

        // Проверка на проигрыш
        if (mistakes >= maxMistakes) {
            endGame(false);
        } else {
            showMessage(`Буквы "${letter}" нет в слове!`, 'lose');
        }
    }
}

// Обновление отображения слова
function updateWordDisplay() {
    wordDisplay.innerHTML = '';
    for (let letter of selectedWord) {
        const letterSpan = document.createElement('span');
        letterSpan.className = 'letter';
        if (guessedLetters.includes(letter)) {
            letterSpan.textContent = letter;
            letterSpan.classList.add('revealed');
            letterSpan.style.color = '#00b894'; // Зеленый для угаданных букв
        } else {
            letterSpan.textContent = '_';
            letterSpan.style.color = '#2d3436'; // Темный для скрытых букв
        }
        wordDisplay.appendChild(letterSpan);
    }
}

// Проверка на победу
function checkWin() {
    return selectedWord.split('').every(letter => guessedLetters.includes(letter));
}

// Обновление счетчика ошибок
function updateMistakes() {
    mistakesEl.textContent = mistakes;
}
```

```
}

// Обновление счета
function updateScore() {
    scoreEl.textContent = score;
}

// Отображение сообщения
function showMessage(msg, type) {
    messageEl.textContent = msg;
    messageEl.className = 'message';
    if (type) {
        messageEl.classList.add(type);
    }

    setTimeout(() => {
        if (messageEl.textContent === msg) {
            messageEl.textContent = '';
            messageEl.className = 'message';
        }
    }, 2000);
}
```

Добавим завершающие игровые функции: `showHangmanPart()` последовательно отображает части тела висельницы при каждой ошибке, удаляя класс `hidden-part` у соответствующего SVG-элемента, `hideAllHangmanParts()` полностью сбрасывает визуализацию, добавляя класс `hidden-part` всем элементам. Функция `endGame(isWin)` обрабатывает завершение раунда - при победе увеличивает счет на основе длины слова и подсвечивает все буквы зеленым, при поражении показывает загаданное слово с красной подсветкой неотгаданных букв, в обоих случаях деактивируя игру (`gameActive = false`). Обработчик события `click` для кнопки перезапуска вызывает `initGame()`, а начальный вызов этой функции при загрузке страницы запускает первый игровой раунд. Весь код обеспечивает полный игровой цикл с визуализацией прогресса, обработкой завершения игры и возможностью рестарта.

Вставьте данный код в продолжение тега `<script>`:

```
// Показать часть висельницы
function showHangmanPart() {
    if (mistakes > 0 && mistakes <= hangmanParts.length) {
        const partId = hangmanParts[mistakes - 1];
        document.getElementById(partId).classList.remove('hidden-part');
    }
}

// Скрыть все части висельницы
function hideAllHangmanParts() {
    hangmanParts.forEach(partId => {
        document.getElementById(partId).classList.add('hidden-part');
    });
}

// Завершение игры
function endGame(isWin) {
    gameActive = false;
```

```
if (isWin) {
    score += selectedWord.length * 10;
    updateScore();
    showMessage(`Поздравляем! Вы угадали слово: ${selectedWord}`, 'win');

    // Показываем все буквы
    const letters = wordDisplay.querySelectorAll('.letter');
    letters.forEach(letter => {
        if (letter.textContent === '_') {
            letter.textContent = selectedWord[letters.indexOf(letter)];
            letter.classList.add('revealed');
            letter.style.color = '#00b894';
        }
    });
} else {
    showMessage(`Игра окончена! Загаданное слово: ${selectedWord}`, 'lose');

    // Показываем все буквы
    const letters = wordDisplay.querySelectorAll('.letter');
    letters.forEach(letter => {
        if (letter.textContent === '_') {
            letter.textContent = selectedWord[letters.indexOf(letter)];
            letter.classList.add('revealed');
            letter.style.color = '#d63031'; // Красный для неотгаданных букв
        }
    });
}
}

// Обработчики событий
restartBtn.addEventListener('click', initGame);

// Начало игры при загрузке
initGame();
```

Теперь игра полностью завершена и готова к использованию: реализована логика отображения частей висельницы через функцию `showHangmanPart()`, которая активирует соответствующую часть SVG-рисунка при каждой ошибке игрока, и `hideAllHangmanParts()`, сбрасывающая визуальное состояние перед новой игрой. Функция `endGame()` обрабатывает финал раунда, обновляя счет при победе, раскрывая загаданное слово и меняя цвет букв в зависимости от результата (зеленый для победы, красный для поражения). Кнопка рестарта инициирует новую игру через `initGame()`, а автоматический вызов этой функции при загрузке страницы обеспечивает готовность к игре сразу после открытия. Весь проект представляет собой полноценную интерактивную игру "Виселица" с выбором категорий, подсчетом очков, визуальной обратной связью и анимациями, работающую в любом современном браузере.

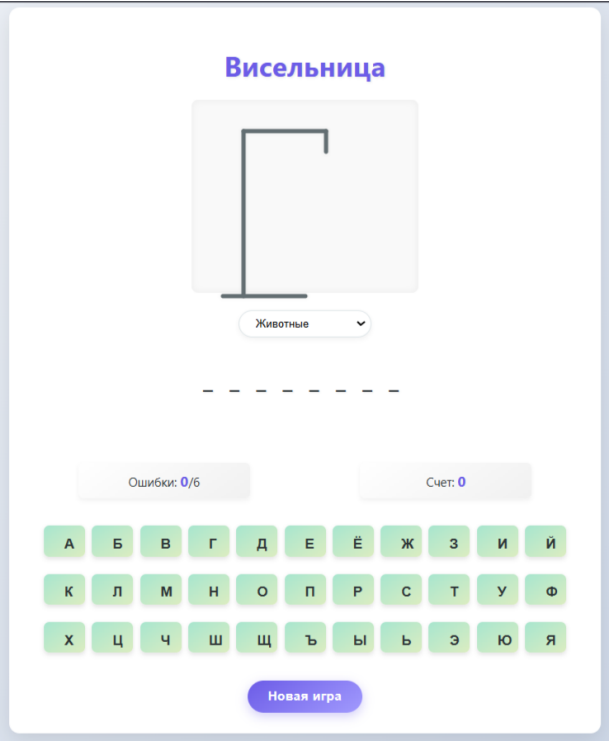


Рисунок 4. Игра

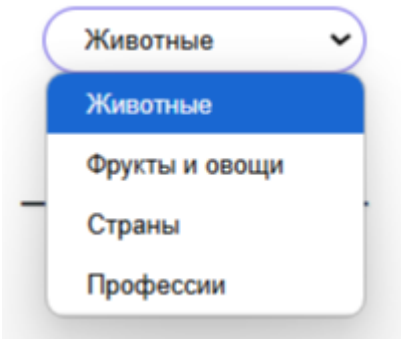


Рисунок 5. Выбор категорий



Рисунок 6. Готовое слово

Если игрок выбирает неправильную букву, функция `handleLetterClick()` проверяет активность игры (`gameActive`) и состояние кнопки, затем добавляет класс `wrong` (красный градиент) и анимацию `shake` (тряска) к нажатой клавише, увеличивает счетчик ошибок (`mistakes++`) через `updateMistakes()` и отображает следующую часть висельницы с помощью `showHangmanPart()`, которая удаляет класс `hidden-part` у соответствующего SVG-элемента. Если количество ошибок достигает максимума (`maxMistakes = 6`), игра завершается через `endGame(false)`, показывая загаданное слово с красной подсветкой неотгаданных букв и сообщение о поражении. В противном случае выводится временное предупреждение `showMessage()` о неправильной букве, которое автоматически исчезает через 2 секунды. Весь процесс обеспечивает визуальную обратную связь, постепенно раскрывая рисунок висельницы и информируя игрока о состоянии игры.

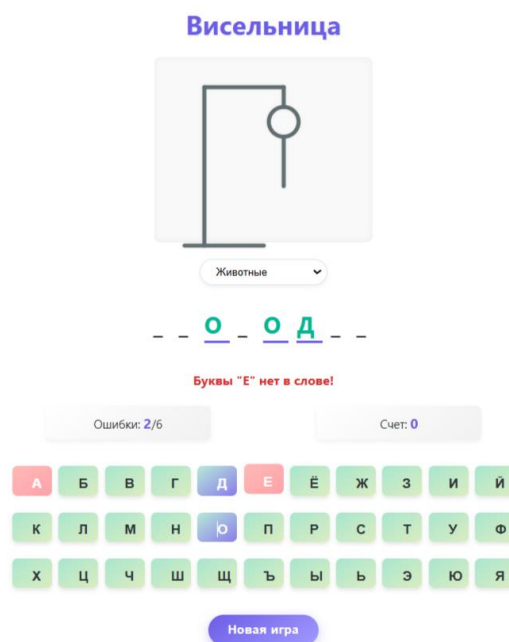


Рисунок 7. Ошибка

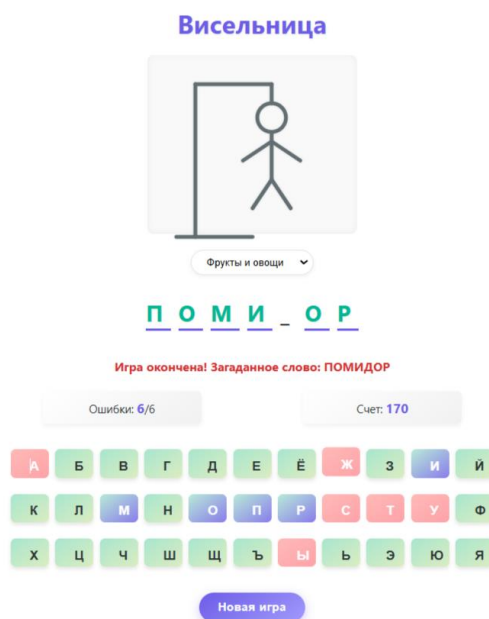


Рисунок 8. Ошибка

4 Выводы

В данной работе реализована веб-версия игры "Висельница", которая наглядно демонстрирует возможности современных веб-технологий в создании интерактивных образовательных приложений. Разработанное решение объединяет в себе несколько ключевых аспектов: эффективную визуализацию игрового процесса с использованием SVG-графики, интуитивно понятный интерфейс с адаптивным дизайном, а также продуманную систему обратной связи, реализованную через CSS-анимации и динамическое обновление контента. Особую ценность проекту придает его образовательная направленность - игра не только развлекает пользователя, но и способствует

расширению словарного запаса и улучшению орфографических навыков. Полученные результаты подтверждают перспективность использования подобных интерактивных решений в цифровом образовании, а модульная структура кода делает проект удобной базой для дальнейшего развития и адаптации под различные учебные задачи. Предложенное решение ориентировано на преподавателей и студентов, заинтересованных в применении современных веб-технологий для повышения эффективности образовательного процесса и превращения обучения в более увлекательный и интерактивный опыт.

Библиографический список

1. Рогатин С. В., Антошкин В. А. Разработка компьютерных игр по технологии html5 //Информатика и прикладная математика. 2015. №. 21. С. 12-16.
2. Фёдоров И. А., Кузьменкова Е. Ю. Разработка игрового приложения «league of football heads» с использованием html5, css3, javascript //Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. 2018. С. 311-311.
3. Королюк Я. А., Куликова Н. Ю. Опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике //XXV Всероссийская студенческая научно-практическая конференция Нижневартковского государственного университета. 2023. С. 48-55.
4. Размологов Д. М., Панина А. С., Захаров П. В. Использование html5 для разработки электронного учебника //Современные тенденции развития науки и технологий. 2016. №. 4-4. С. 92-94.
5. Шегаев И. С., Полозов А. Г. Новейший политологический глоссарий: основные категории, понятия, термины. Общество с ограниченной ответственностью Издательство Перо, 2015. 242 с.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Висельница</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
      display: flex;
      flex-direction: column;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
      margin: 0;
      padding: 20px;
      color: #2d3436;
    }

    .game-container {
      background-color: white;
      border-radius: 15px;
      box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
      padding: 30px;
      max-width: 800px;
      width: 100%;
      text-align: center;
      transition: all 0.3s ease;
    }

    h1 {
      color: #6c5ce7;
      margin-bottom: 25px;
      font-size: 2.5rem;
      text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
    }

    .hangman-container {
      margin: 25px auto;
      position: relative;
      width: 300px;
      height: 250px;
      background-color: #f9f9f9;
      border-radius: 10px;
      padding: 15px;
      box-shadow: inset 0 0 10px rgba(0, 0, 0, 0.05);
    }

    .hangman-drawing {
      stroke: #636e72;
      stroke-width: 4;
      stroke-linecap: round;
      fill: none;
      filter: drop-shadow(1px 1px 2px rgba(0,0,0,0.1));
    }

    .hidden-part {
      opacity: 0;
      transition: opacity 0.3s ease;
    }
```

```
.word-display {
  font-size: 2.2rem;
  letter-spacing: 12px;
  margin: 35px 0;
  font-weight: bold;
  min-height: 60px;
  color: #2d3436;
  text-shadow: 1px 1px 2px rgba(0,0,0,0.1);
}

.letter {
  display: inline-block;
  margin: 0 6px;
  position: relative;
  transition: all 0.3s ease;
}

.letter::after {
  content: ";";
  position: absolute;
  bottom: -5px;
  left: 0;
  width: 100%;
  height: 3px;
  background-color: #6c5ce7;
  transform: scaleX(0);
  transition: transform 0.3s ease;
}

.letter.revealed::after {
  transform: scaleX(1);
}

.keyboard {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 10px;
  margin: 25px 0;
}

.key {
  width: 45px;
  height: 45px;
  font-size: 1.3rem;
  font-weight: bold;
  background: linear-gradient(145deg, #a8e6cf, #dcedc1);
  color: #2d3436;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.2s;
  box-shadow: 0 3px 6px rgba(0,0,0,0.1);
}

.key:hover:not(:disabled) {
  transform: translateY(-3px);
  box-shadow: 0 5px 10px rgba(0,0,0,0.15);
  background: linear-gradient(145deg, #dcedc1, #a8e6cf);
}

.key:active:not(:disabled) {
```

```
        transform: translateY(1px);
    }

    .key.disabled {
        color: white;
        cursor: not-allowed;
        opacity: 0.8;
        box-shadow: none;
    }

    .key.wrong {
        background: linear-gradient(145deg, #ffaaa5, #ff8b94);
        color: white;
    }

    .key.correct {
        background: linear-gradient(145deg, #a8e6cf, #6c5ce7);
        color: white;
    }

    .game-info {
        display: flex;
        justify-content: space-around;
        margin: 25px 0;
        gap: 20px;
    }

    .info-box {
        background: linear-gradient(145deg, #ffffff, #f1f1f1);
        padding: 12px 25px;
        border-radius: 8px;
        font-size: 1.1rem;
        box-shadow: 0 3px 6px rgba(0,0,0,0.05);
        flex: 1;
        max-width: 200px;
    }

    .info-box span {
        font-weight: bold;
        color: #6c5ce7;
        font-size: 1.3rem;
    }

    .message {
        margin: 25px 0;
        font-size: 1.2rem;
        min-height: 27px;
        font-weight: bold;
        transition: all 0.3s ease;
    }

    .message.win {
        color: #00b894;
    }

    .message.lose {
        color: #d63031;
    }

    button {
        background: linear-gradient(145deg, #6c5ce7, #a29bfe);
        color: white;
        border: none;
```

```
padding: 14px 30px;
font-size: 1.1rem;
border-radius: 50px;
cursor: pointer;
transition: all 0.3s;
margin-top: 15px;
box-shadow: 0 5px 15px rgba(108, 92, 231, 0.3);
font-weight: bold;
letter-spacing: 1px;
}

button:hover {
  transform: translateY(-3px);
  box-shadow: 0 8px 20px rgba(108, 92, 231, 0.4);
  background: linear-gradient(145deg, #a29bfe, #6c5ce7);
}

button:active {
  transform: translateY(1px);
}

.category-select {
  margin: 20px 0;
}

select {
  padding: 10px 20px;
  border-radius: 50px;
  border: 2px solid #dfe6e9;
  font-size: 1rem;
  background-color: white;
  cursor: pointer;
  transition: all 0.3s;
  outline: none;
  box-shadow: 0 3px 6px rgba(0,0,0,0.05);
}

select:hover {
  border-color: #a8e6cf;
}

select:focus {
  border-color: #6c5ce7;
}

@media (max-width: 600px) {
  .game-container {
    padding: 20px;
  }

  h1 {
    font-size: 2rem;
  }

  .hangman-container {
    width: 250px;
    height: 200px;
    padding: 10px;
  }

  .word-display {
    font-size: 1.8rem;
    letter-spacing: 8px;
  }
```

```

        margin: 25px 0;
    }

    .key {
        width: 35px;
        height: 35px;
        font-size: 1.1rem;
    }

    .game-info {
        flex-direction: column;
        align-items: center;
    }

    .info-box {
        max-width: 100%;
        width: 100%;
        margin-bottom: 10px;
    }
}

/* Анимации */
@keyframes bounce {
    0%, 100% { transform: translateY(0); }
    50% { transform: translateY(-10px); }
}

.bounce {
    animation: bounce 0.5s;
}

@keyframes shake {
    0%, 100% { transform: translateX(0); }
    20%, 60% { transform: translateX(-5px); }
    40%, 80% { transform: translateX(5px); }
}

.shake {
    animation: shake 0.5s;
}
</style>
</head>
<body>
    <div class="game-container">
        <h1>Висельница</h1>

        <div class="hangman-container">
            <svg viewBox="0 0 200 200" class="hangman-drawing">
                <!-- Основа -->
                <line x1="20" y1="180" x2="100" y2="180" class="hangman-part" id="base"/>
                <line x1="40" y1="180" x2="40" y2="20" class="hangman-part" id="pole"/>
                <line x1="40" y1="20" x2="120" y2="20" class="hangman-part" id="top"/>
                <line x1="120" y1="20" x2="120" y2="40" class="hangman-part" id="rope"/>

                <!-- Части тела (изначально скрыты) -->
                <circle cx="120" cy="55" r="15" class="hangman-part hidden-part" id="head"/>
                <line x1="120" y1="70" x2="120" y2="120" class="hangman-part hidden-part" id="body"/>
                <line x1="120" y1="80" x2="90" y2="100" class="hangman-part hidden-part" id="left-arm"/>
                <line x1="120" y1="80" x2="150" y2="100" class="hangman-part hidden-part" id="right-arm"/>
                <line x1="120" y1="120" x2="100" y2="150" class="hangman-part hidden-part" id="left-leg"/>
                <line x1="120" y1="120" x2="140" y2="150" class="hangman-part hidden-part" id="right-
leg"/>
            </svg>

```

```
</div>

<div class="category-select">
  <select id="category-select">
    <option value="animals">Животные</option>
    <option value="fruits">Фрукты и овощи</option>
    <option value="countries">Страны</option>
    <option value="professions">Профессии</option>
  </select>
</div>

<div class="word-display" id="word-display"></div>

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

<div class="game-info">
  <div class="info-box">Ошибки: <span id="mistakes">0</span>/6</div>
  <div class="info-box">Счет: <span id="score">0</span></div>
</div>

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

<button id="restart-btn">Новая игра</button>
</div>

<script>
  // Слова по категориям
  const wordsByCategory = {
    animals: ['СЛОН', 'ТИГР', 'ЖИРАФ', 'КРОКОДИЛ', 'БЕГЕМОТ', 'ЗЕБРА', 'НОСОРОГ',
'ПАНДА'],
    fruits: ['ЯБЛОКО', 'БАНАН', 'АПЕЛЬСИН', 'ВИНОГРАД', 'АРБУЗ', 'МОРКОВЬ', 'ПОМИДОР',
'ОГУРЕЦ'],
    countries: ['РОССИЯ', 'КИТАЙ', 'БРАЗИЛИЯ', 'КАНАДА', 'ТЕРМАНИЯ', 'ФРАНЦИЯ',
'ИСПАНИЯ', 'ИТАЛИЯ'],
    professions: ['ВРАЧ', 'УЧИТЕЛЬ', 'ПРОГРАММИСТ', 'СТРОИТЕЛЬ', 'ПОВАР', 'ХУДОЖНИК',
'МУЗЫКАНТ', 'ПИЛОТ']
  };

  // Элементы интерфейса
  const wordDisplay = document.getElementById('word-display');
  const keyboard = document.getElementById('keyboard');
  const messageEl = document.getElementById('message');
  const mistakesEl = document.getElementById('mistakes');
  const scoreEl = document.getElementById('score');
  const restartBtn = document.getElementById('restart-btn');
  const categorySelect = document.getElementById('category-select');

  // Игровые переменные
  let selectedWord = '';
  let guessedLetters = [];
  let mistakes = 0;
  const maxMistakes = 6;
  let score = 0;
  let gameActive = false;

  // Русский алфавит для клавиатуры
  const russianAlphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ';

  // Части висельницы
  const hangmanParts = [
    'head', 'body', 'left-arm', 'right-arm', 'left-leg', 'right-leg'
  ];
```

```
// Инициализация игры
function initGame() {
  const category = categorySelect.value;
  const words = wordsByCategory[category];
  selectedWord = words[Math.floor(Math.random() * words.length)];
  guessedLetters = [];
  mistakes = 0;
  gameActive = true;

  // Обновляем интерфейс
  updateWordDisplay();
  updateMistakes();
  updateScore();
  messageEl.textContent = "";
  messageEl.className = 'message';

  // Создаем клавиатуру
  initKeyboard();

  // Скрываем все части висельницы
  hideAllHangmanParts();
}

// Инициализация клавиатуры
function initKeyboard() {
  keyboard.innerHTML = "";
  for (let letter of russianAlphabet) {
    const key = document.createElement('button');
    key.className = 'key';
    key.textContent = letter;
    key.dataset.letter = letter;
    key.addEventListener('click', () => handleLetterClick(letter));
    keyboard.appendChild(key);
  }
}

// Обработка нажатия на букву
function handleLetterClick(letter) {
  if (!gameActive) return;

  const key = document.querySelector(`.key[data-letter="${letter}"]`);
  if (key.disabled) return;

  key.disabled = true;

  if (selectedWord.includes(letter)) {
    // Правильная буква
    key.classList.add('correct');
    key.classList.add('bounce');
    guessedLetters.push(letter);
    updateWordDisplay();

    // Проверка на победу
    if (checkWin()) {
      endGame(true);
    }
  } else {
    // Неправильная буква
    key.classList.add('wrong');
    key.classList.add('shake');
    mistakes++;
    updateMistakes();
    showHangmanPart();
  }
}
```

```
// Проверка на проигрыш
if (mistakes >= maxMistakes) {
  endGame(false);
} else {
  showMessage(`Буквы "${letter}" нет в слове!`, 'lose');
}
}

// Обновление отображения слова
function updateWordDisplay() {
  wordDisplay.innerHTML = "";
  for (let letter of selectedWord) {
    const letterSpan = document.createElement('span');
    letterSpan.className = 'letter';
    if (guessedLetters.includes(letter)) {
      letterSpan.textContent = letter;
      letterSpan.classList.add('revealed');
      letterSpan.style.color = '#00b894'; // Зеленый для угаданных букв
    } else {
      letterSpan.textContent = '_';
      letterSpan.style.color = '#2d3436'; // Темный для скрытых букв
    }
    wordDisplay.appendChild(letterSpan);
  }
}

// Проверка на победу
function checkWin() {
  return selectedWord.split("").every(letter => guessedLetters.includes(letter));
}

// Обновление счетчика ошибок
function updateMistakes() {
  mistakesEl.textContent = mistakes;
}

// Обновление счета
function updateScore() {
  scoreEl.textContent = score;
}

// Отображение сообщения
function showMessage(msg, type) {
  messageEl.textContent = msg;
  messageEl.className = 'message';
  if (type) {
    messageEl.classList.add(type);
  }

  setTimeout(() => {
    if (messageEl.textContent === msg) {
      messageEl.textContent = "";
      messageEl.className = 'message';
    }
  }, 2000);
}

// Показать часть висельницы
function showHangmanPart() {
  if (mistakes > 0 && mistakes <= hangmanParts.length) {
    const partId = hangmanParts[mistakes - 1];
```

```
        document.getElementById(partId).classList.remove('hidden-part');
    }
}

// Скрыть все части висельницы
function hideAllHangmanParts() {
    hangmanParts.forEach(partId => {
        document.getElementById(partId).classList.add('hidden-part');
    });
}

// Завершение игры
function endGame(isWin) {
    gameActive = false;

    if (isWin) {
        score += selectedWord.length * 10;
        updateScore();
        showMessage('Поздравляем! Вы угадали слово: ${selectedWord}', 'win');

        // Показываем все буквы
        const letters = wordDisplay.querySelectorAll('.letter');
        letters.forEach(letter => {
            if (letter.textContent === '_') {
                letter.textContent = selectedWord[letters.indexOf(letter)];
                letter.classList.add('revealed');
                letter.style.color = '#00b894';
            }
        });
    } else {
        showMessage('Игра окончена! Загаданное слово: ${selectedWord}', 'lose');

        // Показываем все буквы
        const letters = wordDisplay.querySelectorAll('.letter');
        letters.forEach(letter => {
            if (letter.textContent === '_') {
                letter.textContent = selectedWord[letters.indexOf(letter)];
                letter.classList.add('revealed');
                letter.style.color = '#d63031'; // Красный для неотгаданных букв
            }
        });
    }
}

// Обработчики событий
restartBtn.addEventListener('click', initGame);

// Начало игры при загрузке
initGame();
</script>
</body>
</html>
```