

JavaScript в действии: Как создать динамическую викторину с тремя уровнями сложности

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается процесс разработки динамической викторины с тремя уровнями сложности, реализованной с использованием современных веб-технологий: HTML, CSS и JavaScript. Представленная методика позволяет создать интерактивную систему тестирования знаний, подходящую как для образовательных целей, так и для развлекательных задач. Статья содержит описание структуры проекта, примеры кода, а также демонстрацию функциональности викторины, включая выбор вопросов по уровню сложности, проверку ответов, отображение результатов и элементы пользовательского интерфейса. Материал предназначен для преподавателей и студентов, заинтересованных в применении современных технологий в обучении и стремящихся сделать процесс образовательного взаимодействия более эффективным и увлекательным.

Ключевые слова: HTML5, CSS, JavaScript, динамическая викторина.

JavaScript in action: How to create a dynamic quiz with three difficulty levels

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

This article discusses the process of developing a dynamic quiz with three levels of difficulty, implemented using modern web technologies: HTML, CSS and JavaScript. The presented methodology makes it possible to create an interactive knowledge testing system suitable for both educational and entertainment purposes. The article contains a description of the project structure, code examples, and a demonstration of the quiz's functionality, including selecting questions by difficulty level, checking answers, displaying results, and user interface elements. The material is intended for teachers and students interested in using modern technologies in teaching and striving to make the process of educational interaction more effective and exciting.

Keywords: HTML5, CSS, JavaScript, dynamic quiz.

1 Введение

Актуальность

Современные образовательные технологии активно используют интерактивные методы обучения для повышения вовлеченности и эффективности усвоения материала. Одним из таких методов являются динамические викторины, которые позволяют не только проверять знания, но и делать процесс обучения увлекательным. JavaScript, как один из ключевых языков веб-разработки, предоставляет широкие возможности для создания интерактивных приложений, включая тесты с адаптивными уровнями сложности.

Разработка викторины на чистом JavaScript (без использования фреймворков) демонстрирует фундаментальные принципы работы с DOM, обработкой событий, асинхронными операциями и локальным хранилищем данных. В данной статье рассматривается создание динамической викторины с тремя уровнями сложности, что позволяет адаптировать тест под разный уровень подготовки пользователей.

Такой подход особенно актуален в дистанционном обучении, где важно не только оценить знания, но и мотивировать учащихся за счет геймификации (игровых элементов: таймер, система очков, таблица лидеров). Использование HTML5, CSS3 и JavaScript обеспечивает кроссплатформенность, анимации и адаптивный интерфейс, что делает викторину доступной на любых устройствах.

1.2 Обзор исследований

В статье рассматривается организация подготовки педагогических работников к использованию технологий HTML5 для разработки электронных образовательных ресурсов в формате веб-приложений. Отмечена актуальность использования мультимедийных интерактивных электронных образовательных ресурсов в современной школе и обоснована эффективность разработки подобных электронных образовательных ресурсов именно в формате офлайновых или онлайновых веб-приложений [1]. Показала процесс создания и использования мультимедийных интерактивных плакатов по информатике К.А. Цилинская. Представила опыт разработки мультимедийных интерактивных плакатов средствами языка HTML5 и Интернет-сервисов [2]. Я. А. Королюк, Н. Ю. Куликова 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 динамической викторины с тремя уровнями сложности, реализованной с использованием современных веб-технологий: HTML, CSS и JavaScript.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовый 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.

Создаем структуру HTML-документа для интерактивной викторины «Программирование: Квест знаний», которая включает подключение метатегов для корректного отображения и адаптации на мобильных устройствах, заголовок страницы, иконки с использованием Font Awesome, а также разметку интерфейса с тремя основными экранами — начальным, игровым и экраном результатов. Интерфейс содержит элементы отображения счета, времени, уровня сложности, динамически обновляемые блоки вопросов и вариантов ответов, индикаторы прогресса и таймер, а также таблицу лидеров и анимационные эффекты для повышения пользовательского опыта.

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

</style>
  <link          rel="stylesheet"          href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
</head>
<body>
  <div class="game-container">
    <header class="game-header">
      <h1>Программирование: Квест знаний</h1>
      <p>Проверь свои навыки в увлекательной викторине!</p>
    </header>

    <div class="game-stats">
      <div class="stat-item">
        <i class="fas fa-star stat-icon"></i>
        <span id="score">0 очков</span>
```

```
</div>
<div class="stat-item">
  <i class="fas fa-trophy stat-icon"></i>
  <span id="high-score">Рекорд: 0</span>
</div>
<div class="stat-item">
  <i class="fas fa-clock stat-icon"></i>
  <span id="timer">60 сек</span>
</div>
</div>

<main class="game-content">
  <div id="start-screen" class="game-screen active">
    <h2>Готов проверить свои знания?</h2>
    <p>Ответь правильно на максимальное количество вопросов за ограниченное
время</p>

    <div class="difficulty-select">
      <h3>Выбери уровень сложности:</h3>
      <div class="difficulty-buttons">
        <button class="difficulty-btn active" data-level="easy">Легкий</button>
        <button class="difficulty-btn" data-level="medium">Средний</button>
        <button class="difficulty-btn" data-level="hard">Сложный</button>
      </div>
    </div>

    <button id="start-btn" class="game-btn">
      <i class="fas fa-play"></i> Начать игру
    </button>
  </div>

  <div id="game-screen" class="game-screen">
    <div class="timer-container">
      <div class="timer-bar" id="timer-bar"></div>
    </div>

    <div class="question-container">
      <span class="category-tag" id="category-tag">HTML</span>
      <div class="difficulty" id="difficulty-display">
        <div class="difficulty-dot"></div>
        <div class="difficulty-dot"></div>
        <div class="difficulty-dot"></div>
      </div>

      <img src="" class="question-image" id="question-image" style="display: none;">
      <h2 id="question-text">Вопрос загружается...</h2>

      <div class="answers-grid" id="answers-container">
        <!-- Варианты ответов будут здесь -->
      </div>

      <div class="explanation" id="explanation" style="display: none;"></div>
```

```
</div>

<div class="progress-container">
  <div class="progress-bar" id="progress-bar"></div>
</div>
</div>

<div id="result-screen" class="game-screen">
  <div class="result-emoji" id="result-emoji"> 🏆 </div>
  <h2>Игра завершена!</h2>
  <p id="result-message">Ты набрал <span id="final-score">0</span> из <span
id="total-questions">0</span> возможных очков</p>

  <div class="achievement-badge" id="achievement-badge" style="display: none;">
    <i class="fas fa-trophy"></i> Новый рекорд!
  </div>

  <div class="stats-container">
    <p><i class="fas fa-check"></i> Правильных ответов: <span id="correct-
answers">0</span></p>
    <p><i class="fas fa-times"></i> Неправильных ответов: <span id="wrong-
answers">0</span></p>
    <p><i class="fas fa-clock"></i> Оставшееся время: <span id="time-left">0</span>
сек</p>
  </div>

  <h3>Лучшие игроки</h3>
  <table class="leaderboard" id="leaderboard">
    <thead>
      <tr>
        <th>#</th>
        <th>Имя</th>
        <th>Очки</th>
        <th>Уровень</th>
      </tr>
    </thead>
    <tbody>
      <!-- Данные таблицы лидеров будут здесь -->
    </tbody>
  </table>

  <button id="restart-btn" class="game-btn">
    <i class="fas fa-redo"></i> Играть снова
  </button>
</div>
</main>
</div>

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

Программирование: Квест знаний

Проверь свои навыки в увлекательной викторине!

★ 0 очков
 🏆 Рекорд: 0
 ⌚ 60 сек

Готов проверить свои знания?

Ответь правильно на максимальное количество вопросов за ограниченное время

Выбери уровень сложности:

HTML

Вопрос загружается...



Игра завершена!

Ты набрал 0 из 0 возможных очков

✓ Правильных ответов: 0

✗ Неправильных ответов: 0

⌚ Оставшееся время: 0 сек

Лучшие игроки

Имя Очки Уровень

Рисунок 1. Основная структура HTML-документа

Теперь добавим CSS-стили, которые определяют внешний вид и анимацию веб-приложения «Программирование: Квест знаний». Стили задают цветовую схему с использованием CSS-переменных, реализуют градиентный фон, стилизованные контейнеры для игровых блоков, анимации при наведении и переходах, оформление кнопок, таблицы лидеров и элементов управления. Также используются шрифтовые иконки через Font Awesome, применяются адаптивные решения для корректного отображения на разных устройствах, обеспечивается высокая читаемость и интерактивность интерфейса благодаря продуманной стилизации состояний элементов, теням, эффектам анимации и цветовой индикации.

```
:root {
  --primary: #4361ee;
  --secondary: #3a0ca3;
  --correct: #4cc9f0;
  --wrong: #f72585;
  --light: #f8f9fa;
  --dark: #212529;
  --accent: #7209b7;
  --highlight: #4895ef;
}

* {
  box-sizing: border-box;
  margin: 0;
```

```
padding: 0;
font-family: 'Montserrat', Arial, sans-serif;
}

@font-face {
font-family: 'Montserrat';
src:
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap');
}

body {
background: linear-gradient(135deg, #1a1a2e 0%, #16213e 100%);
color: var(--light);
min-height: 100vh;
display: flex;
justify-content: center;
align-items: center;
padding: 20px;
}

.game-container {
max-width: 900px;
width: 100%;
background: rgba(255, 255, 255, 0.1);
backdrop-filter: blur(10px);
border-radius: 20px;
box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);
overflow: hidden;
border: 1px solid rgba(255, 255, 255, 0.2);
transition: all 0.3s ease;
}

.game-header {
background: linear-gradient(to right, var(--accent), var(--secondary));
color: white;
padding: 25px;
text-align: center;
position: relative;
overflow: hidden;
}

.game-header::before {
content: "";
position: absolute;
top: -50%;
left: -50%;
width: 200%;
height: 200%;
background: radial-gradient(circle, rgba(255,255,255,0.1) 0%, transparent 70%);
animation: rotate 20s linear infinite;
}
```

```
@keyframes rotate {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

.game-header h1 {
  font-size: 2.2rem;
  margin-bottom: 10px;
  position: relative;
  text-shadow: 0 2px 5px rgba(0,0,0,0.3);
}

.game-header p {
  font-size: 1.1rem;
  opacity: 0.9;
  position: relative;
}

.game-stats {
  display: flex;
  justify-content: space-between;
  background: rgba(0, 0, 0, 0.3);
  padding: 15px 25px;
  border-radius: 0 0 20px 20px;
  font-size: 1.1rem;
  position: relative;
}

.stat-item {
  display: flex;
  align-items: center;
  gap: 8px;
}

.stat-icon {
  font-size: 1.2rem;
}

.game-content {
  padding: 30px;
}

.game-screen {
  display: none;
  flex-direction: column;
  align-items: center;
  text-align: center;
}

.game-screen.active {
  display: flex;
}
```

```
.game-btn {
  background: linear-gradient(to right, var(--primary), var(--highlight));
  color: white;
  border: none;
  padding: 15px 30px;
  font-size: 1.1rem;
  border-radius: 50px;
  cursor: pointer;
  margin-top: 30px;
  transition: all 0.3s;
  box-shadow: 0 4px 15px rgba(67, 97, 238, 0.4);
  position: relative;
  overflow: hidden;
  z-index: 1;
}

.game-btn::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: linear-gradient(to right, var(--highlight), var(--primary));
  z-index: -1;
  opacity: 0;
  transition: opacity 0.3s;
}

.game-btn:hover {
  transform: translateY(-3px);
  box-shadow: 0 6px 20px rgba(67, 97, 238, 0.6);
}

.game-btn:hover::before {
  opacity: 1;
}

.game-btn:active {
  transform: translateY(1px);
}

.question-container {
  width: 100%;
  background: rgba(255, 255, 255, 0.05);
  border-radius: 15px;
  padding: 30px;
  margin-bottom: 25px;
  border: 1px solid rgba(255, 255, 255, 0.1);
  position: relative;
  overflow: hidden;
```

```
}

.question-container::after {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 5px;
  background: linear-gradient(to right, var(--primary), var(--accent));
}

#question-text {
  margin-bottom: 25px;
  font-size: 1.4rem;
  line-height: 1.5;
  color: white;
  font-weight: 500;
}

.question-image {
  width: 100%;
  max-height: 200px;
  object-fit: contain;
  border-radius: 10px;
  margin-bottom: 20px;
  box-shadow: 0 5px 15px rgba(0,0,0,0.2);
}

.answers-grid {
  display: grid;
  grid-template-columns: 1fr;
  gap: 15px;
  width: 100%;
}

.answer-btn {
  background: rgba(255, 255, 255, 0.1);
  border: 2px solid rgba(255, 255, 255, 0.2);
  border-radius: 12px;
  padding: 18px;
  font-size: 1rem;
  cursor: pointer;
  transition: all 0.3s;
  text-align: left;
  color: white;
  display: flex;
  align-items: center;
  gap: 12px;
  position: relative;
  overflow: hidden;
}
```

```
.answer-btn::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 5px;
  height: 100%;
  background: var(--primary);
  transition: width 0.3s;
}

.answer-btn:hover {
  background: rgba(255, 255, 255, 0.2);
  transform: translateX(5px);
}

.answer-btn:hover::before {
  width: 100%;
  opacity: 0.2;
}

.answer-letter {
  background: var(--primary);
  color: white;
  width: 30px;
  height: 30px;
  border-radius: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
  font-weight: bold;
  flex-shrink: 0;
}

.answer-btn.correct {
  background: rgba(76, 201, 240, 0.2);
  border-color: var(--correct);
}

.answer-btn.correct .answer-letter {
  background: var(--correct);
}

.answer-btn.wrong {
  background: rgba(247, 37, 133, 0.2);
  border-color: var(--wrong);
}

.answer-btn.wrong .answer-letter {
  background: var(--wrong);
}
```

```
.progress-container {
  width: 100%;
  height: 8px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 10px;
  margin: 25px 0;
  overflow: hidden;
}

.progress-bar {
  height: 100%;
  background: linear-gradient(to right, var(--primary), var(--accent));
  width: 0%;
  transition: width 0.5s ease;
  border-radius: 10px;
  position: relative;
}

.progress-bar::after {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background: linear-gradient(90deg, transparent, rgba(255,255,255,0.3),
transparent);
  animation: progressShine 2s infinite;
}

@keyframes progressShine {
  0% { transform: translateX(-100%); }
  100% { transform: translateX(100%); }
}

.timer-container {
  width: 100%;
  height: 8px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 10px;
  margin-bottom: 20px;
  overflow: hidden;
}

.timer-bar {
  height: 100%;
  background: linear-gradient(to right, #4cc9f0, #4361ee);
  width: 100%;
  border-radius: 10px;
  transition: width 1s linear;
}
```

```
.explanation {  
  background: rgba(0, 0, 0, 0.3);  
  padding: 15px;  
  border-radius: 10px;  
  margin-top: 20px;  
  font-size: 0.95rem;  
  line-height: 1.6;  
  text-align: left;  
  border-left: 4px solid var(--highlight);  
  animation: fadeIn 0.5s ease-out;  
}
```

```
.category-tag {  
  display: inline-block;  
  background: var(--accent);  
  color: white;  
  padding: 5px 12px;  
  border-radius: 50px;  
  font-size: 0.8rem;  
  margin-bottom: 15px;  
  font-weight: bold;  
}
```

```
.difficulty {  
  display: flex;  
  gap: 5px;  
  margin-bottom: 15px;  
  justify-content: center;  
}
```

```
.difficulty-dot {  
  width: 12px;  
  height: 12px;  
  border-radius: 50%;  
  background: rgba(255,255,255,0.2);  
}
```

```
.difficulty-dot.active {  
  background: var(--highlight);  
}
```

```
.result-emoji {  
  font-size: 4rem;  
  margin-bottom: 20px;  
  animation: bounce 1s infinite alternate;  
}
```

```
@keyframes bounce {  
  from { transform: translateY(0); }  
  to { transform: translateY(-15px); }  
}
```

```
.achievement-badge {
  background: linear-gradient(45deg, #ff9a9e 0%, #fad0c4 99%, #fad0c4 100%);
  color: #d32f2f;
  padding: 8px 15px;
  border-radius: 50px;
  font-weight: bold;
  margin: 15px 0;
  display: inline-block;
  box-shadow: 0 3px 10px rgba(0,0,0,0.2);
  animation: pulse 1.5s infinite;
}

@keyframes pulse {
  0% { transform: scale(1); }
  50% { transform: scale(1.05); }
  100% { transform: scale(1); }
}

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

.leaderboard th, .leaderboard td {
  padding: 12px;
  text-align: left;
  border-bottom: 1px solid rgba(255,255,255,0.1);
}

.leaderboard th {
  background: rgba(255,255,255,0.1);
  color: var(--highlight);
}

.leaderboard tr:nth-child(even) {
  background: rgba(255,255,255,0.05);
}

.leaderboard tr:hover {
  background: rgba(255,255,255,0.1);
}

/* Анимации */
@keyframes fadeIn {
  from { opacity: 0; transform: translateY(20px); }
  to { opacity: 1; transform: translateY(0); }
}

@keyframes shake {
  0%, 100% { transform: translateX(0); }
```

```
20%, 60% { transform: translateX(-5px); }
40%, 80% { transform: translateX(5px); }
}

.shake {
  animation: shake 0.5s;
}

/* Адаптивность */
@media (min-width: 768px) {
  .answers-grid {
    grid-template-columns: 1fr 1fr;
  }

  .game-header h1 {
    font-size: 2.5rem;
  }
}

@media (max-width: 480px) {
  .game-content {
    padding: 20px;
  }

  .question-container {
    padding: 20px;
  }

  .game-btn {
    padding: 12px 25px;
    font-size: 1rem;
  }

  #question-text {
    font-size: 1.2rem;
  }
}
```

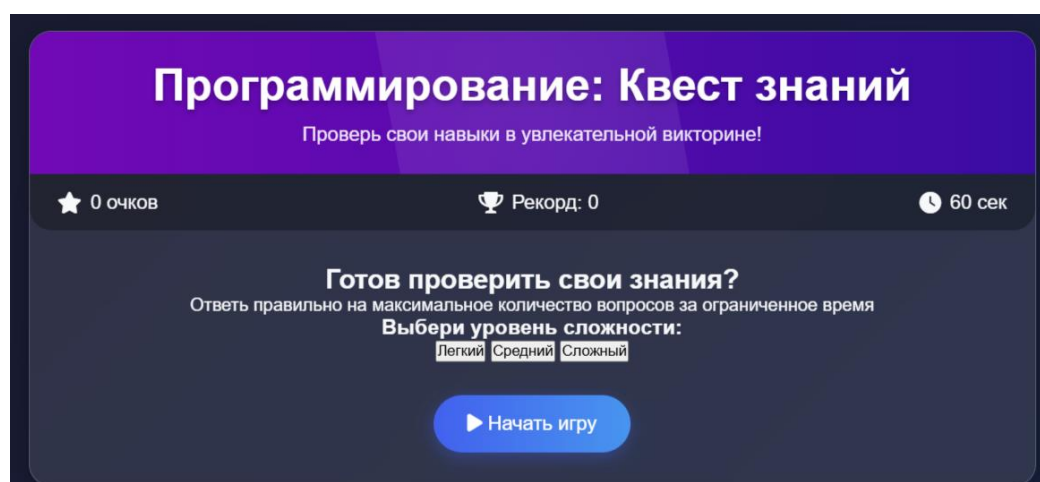


Рисунок 2. Добавление CSS -стилей

Далее добавим код, который определяет структуру массива quizData с вопросами для викторины, разделёнными на уровни сложности: легкий, средний и сложный. Каждый уровень содержит объекты с полями вопросов, вариантов ответов, индексом правильного ответа, пояснением, категорией и, при необходимости, изображением. Этот код лежит в основе игровой механики, обеспечивая динамическое отображение вопросов в зависимости от выбранного уровня сложности и позволяя проверять ответы пользователей, выдавать объяснения и подсчитывать результаты.

```
const quizData = {
  easy: [
    {
      question: "Какой тег используется для создания ссылки в HTML?",
      answers: ["<link>", "<a>", "<href>", "<url>"],
      correct: 1,
      explanation: "Тег <a> (anchor) используется для создания гиперссылок в HTML.",
      category: "HTML",
      image: ""
    },
    {
      question: "Какой язык используется для стилизации веб-страниц?",
      answers: ["JavaScript", "HTML", "CSS", "Python"],
      correct: 2,
      explanation: "CSS (Cascading Style Sheets) - это язык стилей, используемый для описания внешнего вида документа, написанного на HTML.",
      category: "CSS",
      image: ""
    },
    {
      question: "Как объявить переменную в JavaScript?",
      answers: ["variable x = 5", "var x = 5", "let x = 5", "Все варианты верны"],
      correct: 2,
      explanation: "В JavaScript можно объявлять переменные с помощью ключевых слов var, let и const. В данном случае правильны варианты с var и let.",
      category: "JavaScript",
      image: ""
    }
  ],
  medium: [
    {
      question: "Что такое API в веб-разработке?",
      answers: [
        "Язык программирования",
        "Интерфейс программирования приложений",
        "База данных",
        "Фреймворк для тестирования"
      ],
      correct: 1,
      explanation: "API (Application Programming Interface) - это набор методов и правил, позволяющих программам взаимодействовать друг с другом.",
    }
  ]
}
```

```
category: "Веб-разработка",
image: "https://example.com/api-image.jpg"
},
{
  question: "Какой метод массива создает новый массив с результатами
вызова функции для каждого элемента?",
  answers: ["forEach()", "map()", "filter()", "reduce()"],
  correct: 1,
  explanation: "Метод map() создает новый массив, вызывая
предоставленную функцию для каждого элемента исходного массива.",
  category: "JavaScript",
  image: ""
},
{
  question: "Что делает CSS-свойство 'display: flex'?",
  answers: [
    "Создает гибкий контейнер",
    "Скрывает элемент",
    "Делает элемент прозрачным",
    "Изменяет шрифт"
  ],
  correct: 0,
  explanation: "Flexbox - это модель макета, которая позволяет легко
выравнивать элементы по горизонтали и вертикали.",
  category: "CSS",
  image: ""
}
],
hard: [
  {
    question: "Что такое замыкание (closure) в JavaScript?",
    answers: [
      "Способ скрытия переменных",
      "Функция вместе с лексическим окружением",
      "Метод закрытия модального окна",
      "Специальный тип объекта"
    ],
    correct: 1,
    explanation: "Замыкание - это функция вместе со всеми внешними
переменными, которые ей доступны. Это мощный инструмент в JavaScript.",
    category: "JavaScript",
    image: ""
  },
  {
    question: "Какой алгоритм использует React для эффективного
обновления DOM?",
    answers: [
      "Virtual DOM",
      "Real DOM",
      "Shadow DOM",
      "Double DOM"
    ],
    correct: 1,
  }
]
```

```

        correct: 0,
        explanation: "React использует Virtual DOM (виртуальный DOM) для
минимизации количества операций с реальным DOM, что повышает производительность.",
        category: "React",
        image: ""
    },
    {
        question: "Что такое асинхронный код в JavaScript?",
        answers: [
            "Код, который выполняется параллельно",
            "Код, который выполняется с задержкой",
            "Код, который не блокирует основной поток",
            "Код, который выполняется только ночью"
        ],
        correct: 2,
        explanation: "Асинхронный код позволяет выполнять операции без
блокировки основного потока выполнения, используя колбэки, промисы или async/await.",
        category: "JavaScript",
        image: ""
    }
]
};

```

Теперь добавим код, который реализует подключение к ключевым элементам интерфейса викторины и инициализирует переменные для управления игровой логикой: объявляются константы для экранов игры (стартовый, игровой, результатов), кнопок, текстовых полей и элементов отображения данных (вопросы, ответы, таймер, счёт, уровень сложности), а также создаются переменные для хранения текущего состояния игры — список вопросов, индекс текущего вопроса, очки, время, статус активности игры, количество правильных и неправильных ответов, выбранный уровень сложности и массив случайно выбранных вопросов — всё это формирует основу для динамического взаимодействия с пользователем и обновления интерфейса в реальном времени.

```

const startScreen = document.getElementById('start-screen');
const gameScreen = document.getElementById('game-screen');
const resultScreen = document.getElementById('result-screen');
const startBtn = document.getElementById('start-btn');
const restartBtn = document.getElementById('restart-btn');
const questionText = document.getElementById('question-text');
const answersContainer = document.getElementById('answers-container');
const scoreElement = document.getElementById('score');
const highScoreElement = document.getElementById('high-score');
const timerElement = document.getElementById('timer');
const timerBar = document.getElementById('timer-bar');
const finalScoreElement = document.getElementById('final-score');
const totalQuestionsElement = document.getElementById('total-questions');
const resultMessage = document.getElementById('result-message');
const progressBar = document.getElementById('progress-bar');

```

```
const explanationElement = document.getElementById('explanation');
const categoryTag = document.getElementById('category-tag');
const difficultyDisplay = document.getElementById('difficulty-display');
const questionImage = document.getElementById('question-image');
const resultEmoji = document.getElementById('result-emoji');
const achievementBadge = document.getElementById('achievement-badge');
const correctAnswersElement = document.getElementById('correct-answers');
const wrongAnswersElement = document.getElementById('wrong-answers');
const timeLeftElement = document.getElementById('time-left');
const leaderboard = document.getElementById('leaderboard');
const difficultyBtns = document.querySelectorAll('.difficulty-btn');
```

```
// Переменные игры
let currentQuestions = [];
let currentQuestionIndex = 0;
let score = 0;
let highScore = localStorage.getItem('highScore') || 0;
let timeLeft = 60;
let timer;
let gameActive = false;
let correctAnswers = 0;
let wrongAnswers = 0;
let currentDifficulty = 'easy';
let selectedQuestions = [];
```

Далее добавим код, который реализует логику работы викторины, включая выбор вопросов в зависимости от уровня сложности, запуск игрового процесса, отображение текущего вопроса и вариантов ответа, обработку выбора пользователем, начисление очков за правильные ответы, а также завершение игры с выводом результатов. В начале функция `startGame ()` инициализирует параметры игры: выбирает 10 случайных вопросов из массива `quizData`, соответствующих выбранному уровню сложности (легкий, средний или сложный), перемешивает их для случайного порядка, устанавливает начальные значения счётчика времени, баллов и индекса текущего вопроса.

Функция `startTimer ()` управляет обратным отсчетом времени, обновляя состояние таймера на экране и анимируя прогресс-бар, который уменьшается пропорционально оставшемуся времени. При приближении к нулевой отметке активируется визуальный эффект — текст начинает «трястись» и изменяется цвет на красный, что повышает интерактивность и вовлечённость игрока. Если время истекает, игра заканчивается автоматически.

Функция `showQuestion ()` отвечает за отрисовку текущего вопроса и вариантов ответов на странице. Она обновляет заголовок категории, уровень сложности и, при наличии, изображение к вопросу. Каждый вариант ответа представлен кнопкой, клик по которой вызывает проверку правильности и начисление бонусов или штрафов за неверные ответы. Также реализуется отображение пояснения к каждому вопросу, которое показывается на короткое время перед переходом к следующему вопросу.

В конце игры срабатывает функция `endGame ()`, которая рассчитывает итоговый результат, определяет процент правильных ответов и формирует соответствующее сообщение с эмодзи и рекомендацией по дальнейшим действиям. Результат также сохраняется в таблице лидеров, где отображаются имя, количество набранных очков, уровень сложности и позиция игрока. Это позволяет создать элемент соревнования и мотивации для повторной игры.

Также реализованы дополнительные вспомогательные функции:

- `shuffleArray ()` — обеспечивает случайное перемешивание вопросов;
- `updateScore ()` и `updateProgress ()` — обновляют отображение счёта и прогресса прохождения;
- `playSound ()` — может быть использована для воспроизведения звуковых эффектов (в данном случае выведена в виде консольного лога);
- `restartGame ()` — сбрасывает все игровые параметры и возвращает пользователя на стартовый экран.

Этот блок JavaScript-кода лежит в основе игровой механики и обеспечивает динамическое взаимодействие с пользователем, делая процесс обучения не только полезным, но и увлекательным.

```
// Инициализация игры
document.addEventListener('DOMContentLoaded', () => {
  highScoreElement.textContent = `Рекорд: ${highScore}`;

  startBtn.addEventListener('click', startGame);
  restartBtn.addEventListener('click', restartGame);

  // Обработчики выбора сложности
  difficultyBtns.forEach(btn => {
    btn.addEventListener('click', () => {
      difficultyBtns.forEach(b => b.classList.remove('active'));
      btn.classList.add('active');
      currentDifficulty = btn.dataset.level;
    });
  });

  // Заполняем таблицу лидеров
  updateLeaderboard();
});

// Начало игры
function startGame() {
  startScreen.classList.remove('active');
  gameScreen.classList.add('active');
  gameActive = true;

  // Выбираем вопросы в зависимости от сложности
  selectedQuestions = [...quizData[currentDifficulty]];
  shuffleArray(selectedQuestions);
  selectedQuestions = selectedQuestions.slice(0, 10); // Берем 10 вопросов
```

```
score = 0;
currentQuestionIndex = 0;
timeLeft = 60;
correctAnswers = 0;
wrongAnswers = 0;

updateScore();
startTimer();
showQuestion();
}

// Запуск таймера
function startTimer() {
  timerElement.textContent = `${timeLeft} сек`;
  timerBar.style.width = '100%';

  timer = setInterval(() => {
    timeLeft--;
    timerElement.textContent = `${timeLeft} сек`;
    timerBar.style.width = `${(timeLeft / 60) * 100}%`;

    if (timeLeft <= 10) {
      timerElement.classList.add('shake');
      timerElement.style.color = '#ff6b6b';
      timerElement.style.fontWeight = 'bold';
    }

    if (timeLeft <= 0) {
      endGame();
    }
  }, 1000);
}

// Отображение вопроса
function showQuestion() {
  if (currentQuestionIndex >= selectedQuestions.length) {
    endGame();
    return;
  }

  const question = selectedQuestions[currentQuestionIndex];
  questionText.textContent = question.question;
  answersContainer.innerHTML = "";
  explanationElement.style.display = 'none';

  // Отображаем категорию
  categoryTag.textContent = question.category;

  // Отображаем сложность
  const difficultyLevel = currentDifficulty === 'easy' ? 1 : currentDifficulty ===
'medium' ? 2 : 3;
```

```

const dots = difficultyDisplay.querySelectorAll('.difficulty-dot');
dots.forEach((dot, index) => {
  dot.classList.toggle('active', index < difficultyLevel);
});

// Отображаем изображение, если есть
if (question.image) {
  questionImage.src = question.image;
  questionImage.style.display = 'block';
} else {
  questionImage.style.display = 'none';
}

// Создаем кнопки ответов
const letters = ['A', 'B', 'C', 'D'];
question.answers.forEach((answer, index) => {
  const button = document.createElement('button');
  button.className = 'answer-btn';
  button.innerHTML = `
    <span class="answer-letter">${letters[index]}</span>
    <span class="answer-text">${answer}</span>
  `;
  button.addEventListener('click', () => selectAnswer(index));
  answersContainer.appendChild(button);
});

// Обновляем прогресс
updateProgress();
}

// Обработка выбора ответа
function selectAnswer(answerIndex) {
  if (!gameActive) return;

  const question = selectedQuestions[currentQuestionIndex];
  const answerButtons = document.querySelectorAll('.answer-btn');

  // Отключаем все кнопки после выбора
  answerButtons.forEach(button => {
    button.disabled = true;
  });

  // Проверяем ответ
  if (answerIndex === question.correct) {
    answerButtons[answerIndex].classList.add('correct');
    score += currentDifficulty === 'easy' ? 5 : currentDifficulty === 'medium' ? 10 :
15;

    correctAnswers++;
    updateScore();

    // Воспроизводим звук правильного ответа
    playSound('correct');
  }
}

```

```
    } else {
      answerButtons[answerIndex].classList.add('wrong');
      answerButtons[question.correct].classList.add('correct');
      wrongAnswers++;

      // Воспроизводим звук неправильного ответа
      playSound('wrong');

      // Штрафуем за неправильный ответ
      timeLeft = Math.max(0, timeLeft - 3);
      timerElement.classList.add('shake');
      setTimeout(() => timerElement.classList.remove('shake'), 500);
    }

    // Показываем объяснение
    explanationElement.textContent = question.explanation;
    explanationElement.style.display = 'block';

    // Переход к следующему вопросу через 2 секунды
    setTimeout(() => {
      currentQuestionIndex++;
      showQuestion();
    }, 2000);
  }

  // Обновление счета
  function updateScore() {
    scoreElement.textContent = `${score} очков`;
  }

  // Обновление прогресс-бара
  function updateProgress() {
    const progress = ((currentQuestionIndex) / selectedQuestions.length) * 100;
    progressBar.style.width = `${progress}%`;
  }

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

    gameScreen.classList.remove('active');
    resultScreen.classList.add('active');

    finalScoreElement.textContent = score;
    totalQuestionsElement.textContent = selectedQuestions.length;
    correctAnswersElement.textContent = correctAnswers;
    wrongAnswersElement.textContent = wrongAnswers;
    timeLeftElement.textContent = timeLeft;

    // Проверяем рекорд
    if (score > highScore) {
```

```

        highScore = score;
        localStorage.setItem('highScore', highScore);
        highScoreElement.textContent = `Рекорд: ${highScore}`;
        achievementBadge.style.display = 'block';
        resultEmoji.textContent = '🏆';
        playSound('highscore');
    } else {
        achievementBadge.style.display = 'none';
    }

    // Выбираем emoji в зависимости от результата
    const percentage = (correctAnswers / selectedQuestions.length) * 100;
    if (percentage >= 80) {
        resultEmoji.textContent = '🏆';
        resultMessage.textContent = `Отличный результат! Ты настоящий эксперт в
        ${currentDifficulty === 'easy' ? 'основах' : currentDifficulty === 'medium' ? 'продвинутых
        темах' : 'сложных концепциях'}!`;
    } else if (percentage >= 50) {
        resultEmoji.textContent = '👍';
        resultMessage.textContent = `Хороший результат! Продолжай учиться и
        улучшать свои навыки!`;
    } else {
        resultEmoji.textContent = '😓';
        resultMessage.textContent = `Попробуй еще раз! Рекомендуем начать с более
        простого уровня.`;
    }

    // Обновляем таблицу лидеров
    updateLeaderboard();
}

// Перезапуск игры
function restartGame() {
    resultScreen.classList.remove('active');
    startScreen.classList.add('active');
    timerElement.style.color = '';
    timerElement.style.fontWeight = '';
    timerElement.classList.remove('shake');
}

// Обновление таблицы лидеров
function updateLeaderboard() {
    // В реальном приложении здесь была бы загрузка данных с сервера
    const leaderboardData = JSON.parse(localStorage.getItem('leaderboard')) || [];

    // Добавляем текущий результат, если он достаточно высок
    if (score > 0) {
        const playerName = prompt('Введите свое имя для таблицы лидеров:',
        'Аноним') || 'Аноним';
        leaderboardData.push({
            name: playerName,

```

```
        score: score,
        difficulty: currentDifficulty,
        date: new Date().toLocaleDateString()
    });

    // Сортируем и сохраняем только топ-5
    leaderboardData.sort((a, b) => b.score - a.score);
    const topPlayers = leaderboardData.slice(0, 5);
    localStorage.setItem('leaderboard', JSON.stringify(topPlayers));
}

// Заполняем таблицу
const tbody = leaderboard.querySelector('tbody');
tbody.innerHTML = "";

const displayData = JSON.parse(localStorage.getItem('leaderboard')) || [];
displayData.forEach((player, index) => {
    const row = document.createElement('tr');
    row.innerHTML = `
        <td>${index + 1}</td>
        <td>${player.name}</td>
        <td>${player.score}</td>
        <td>${getDifficultyName(player.difficulty)}</td>
    `;
    tbody.appendChild(row);
});
}

// Вспомогательные функции
function shuffleArray(array) {
    for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [array[i], array[j]] = [array[j], array[i]];
    }
    return array;
}

function getDifficultyName(difficulty) {
    return difficulty === 'easy' ? 'Легкий' : difficulty === 'medium' ? 'Средний' :
'Сложный';
}

function playSound(type) {
    // В реальном приложении здесь было бы воспроизведение звука
    console.log(`Playing ${type} sound`);
}

// Добавляем обработчик для сброса анимации таймера
timerElement.addEventListener('animationend', () => {
    timerElement.classList.remove('shake');
});
```

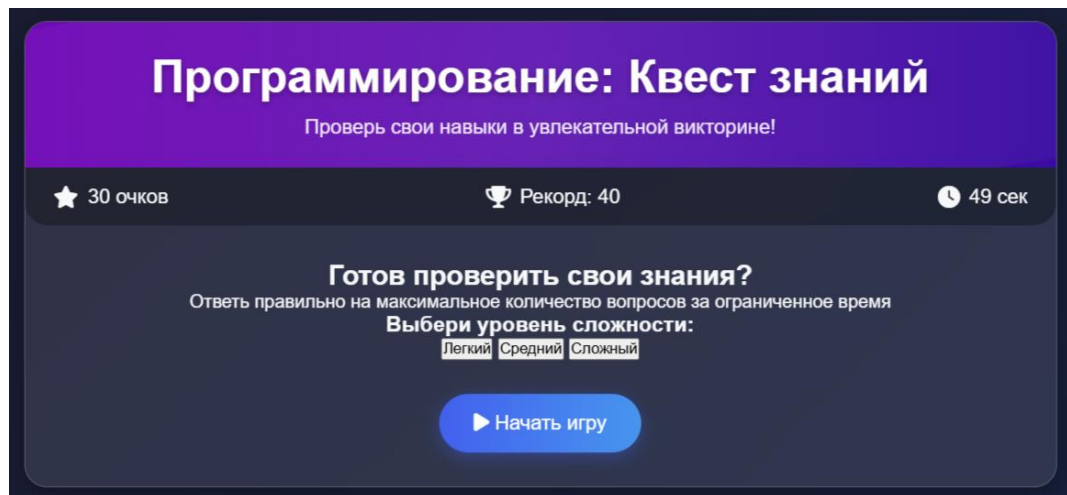


Рисунок 3. Готовая система

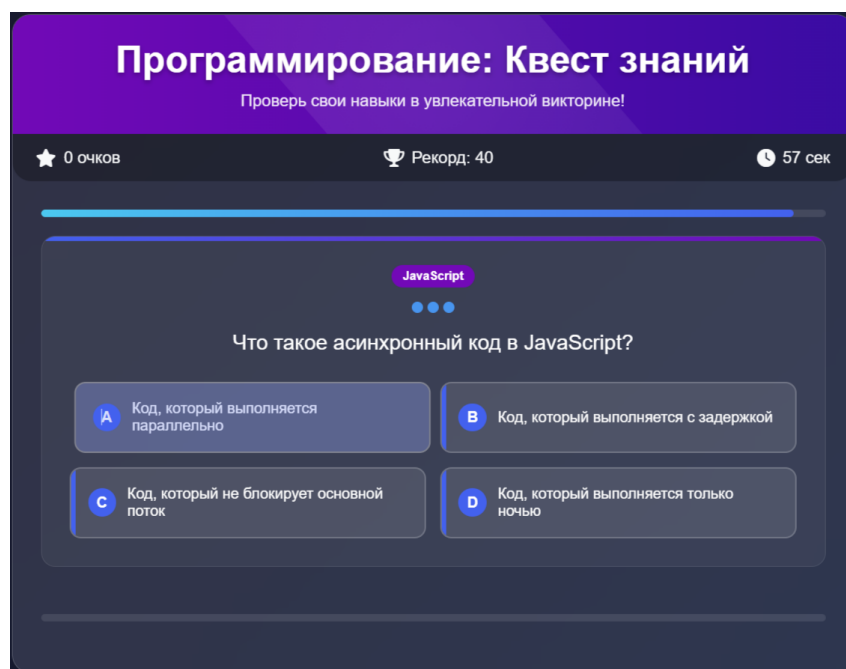


Рисунок 4. Прохождение теста

Выбранный пользователем ответ выделен красным цветом, указывая на ошибку. Правильный ответ подсвечивается зеленым, что помогает пользователю понять, какой ответ был правильным. Нижняя часть экрана содержит пояснение к вопросу: "React использует Virtual DOM (виртуальный DOM) для минимизации количества операций с реальным DOM, что повышает производительность." Это объясняет, почему вариант А является верным.

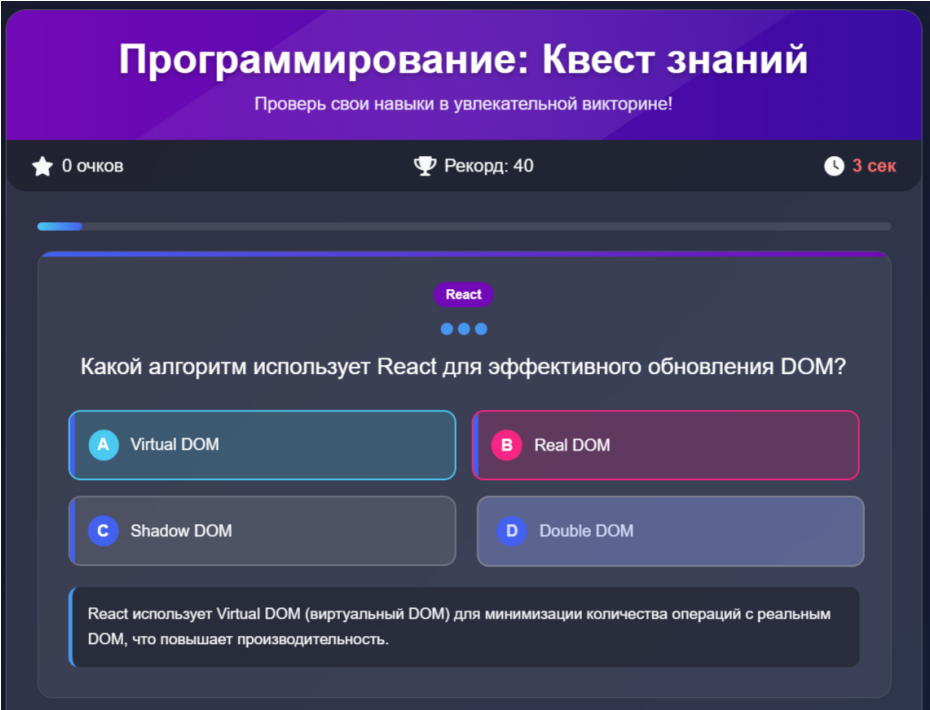


Рисунок 5. Прохождение теста

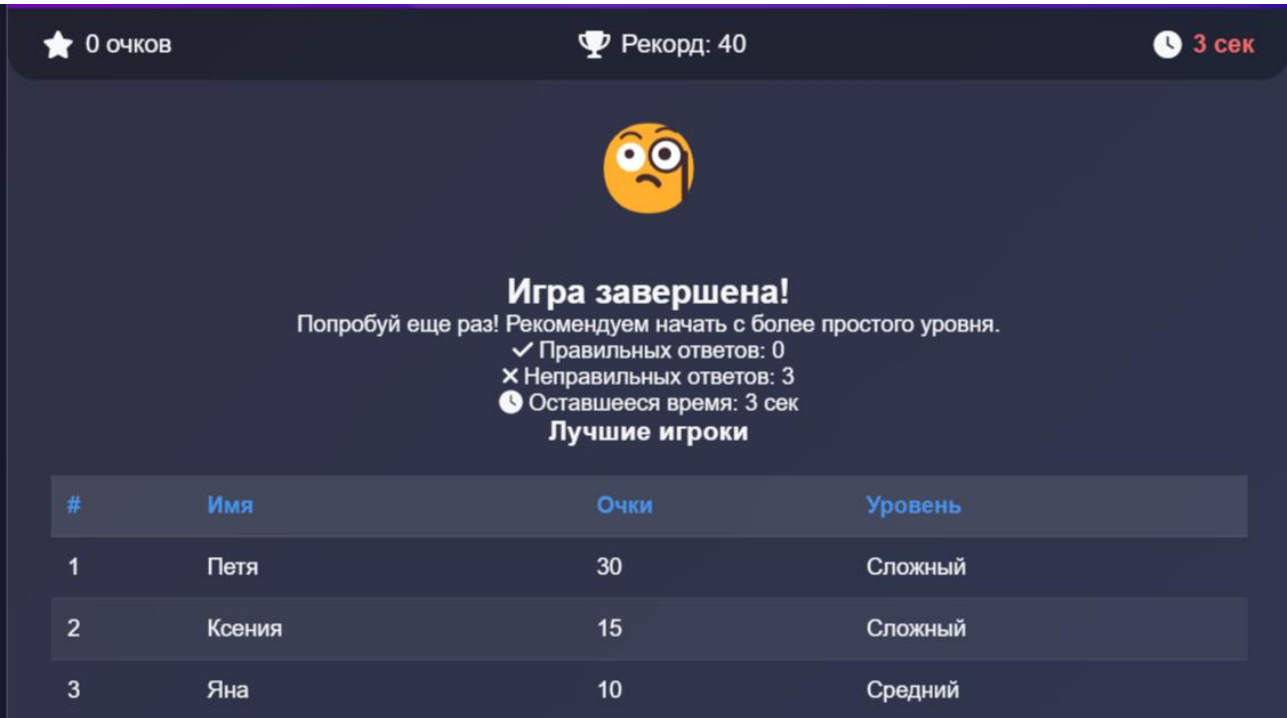


Рисунок 6. Таблица лучших игроков

4 Выводы

В данной работе подробно рассмотрено создание интерактивной викторины «Программирование: Квест знаний» с использованием HTML, CSS и JavaScript, которая включает динамическую генерацию вопросов трех уровней сложности (легкий, средний, сложный) для персонализации тестирования, интерактивный интерфейс с анимациями, таймером, прогресс-баром и подсветкой ответов, локальное сохранение результатов через

localStorage с таблицей лидеров, а также адаптивный дизайн для корректного отображения на различных устройствах. 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. Бакулевская С. С. Подготовка педагогических работников к использованию технологий html5 для разработки электронных образовательных ресурсов //Информатика и образование. 2019. №. 5. С. 32-40.
2. Цилинская К.А. Создание и использование мультимедийных интерактивных плакатов по информатике //Будущее науки-2019. 2019. С. 340-344.
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.