

Создание мини-игры «Пицца-мастер» на чистом JavaScript: от HTML-структуры до игровой логики

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается методика создания интерактивной образовательной игры «Пицца-мастер» с применением технологий HTML5, CSS3 и JavaScript. Разработанное решение позволяет воссоздать реалистичный процесс приготовления пиццы с механизмом последовательного добавления ингредиентов, системой проверки соответствия рецепту и динамическим таймером. Особое внимание уделено реализации алгоритмов генерации случайных заказов, визуализации игрового процесса с анимацией элементов, организации уровней сложности с прогрессивным увеличением требований и обработке пользовательских взаимодействий. Представленное решение демонстрирует эффективное сочетание современных веб-технологий с игровыми механиками для создания образовательных приложений. Материал предназначен для преподавателей и студентов, заинтересованных в применении геймификации в обучении и стремящихся сделать процесс образовательного взаимодействия более эффективным и увлекательным.

Ключевые слова: Пицца-мастер, HTML5, CSS, JavaScript.

Creating a Pizza Wizard mini-game in pure JavaScript: from HTML structure to game logic

Ekimova Yana Sergeevna

Sholom-Aleichem Priamurskiy State University

Student

Abstract

The article discusses the method of creating an interactive educational game "Pizza Master" using HTML5, CSS3 and JavaScript technologies. The developed solution allows you to recreate a realistic pizza cooking process with a sequential ingredient addition mechanism, a recipe verification system and a dynamic timer. Special attention is paid to the implementation of algorithms for generating random orders, visualizing the gameplay with animation of elements, organizing difficulty levels with a progressive increase in requirements, and processing user interactions. The presented solution demonstrates an effective combination of modern web technologies with game mechanics to create educational applications. The material is intended for teachers and students interested in using gamification in teaching and seeking to make the process of educational interaction more effective and exciting.

Keywords: Pizza Wizard, HTML5, CSS, JavaScript.

1 Введение

1.1 Актуальность

В современном мире веб-технологий разработка игровых приложений на HTML5 и JavaScript приобретает особую значимость, поскольку позволяет создавать кроссплатформенные проекты, работающие непосредственно в браузере без необходимости установки дополнительного программного обеспечения. Актуальность реализации классической игры Тетрис заключается в том, что она служит идеальным учебным полигоном для освоения ключевых аспектов программирования: работы с DOM-структурами, обработки пользовательского ввода, реализации игровой логики и анимации, а также демонстрирует эффективное применение алгоритмов для проверки коллизий и оптимизации производительности. Кроме того, возрождение интереса к ретро-играм и их адаптация под современные веб-стандарты отвечает запросам широкой аудитории пользователей, обеспечивая доступность и сохранение культурного наследия игровой индустрии.

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 создание полнофункциональной веб-версии образовательной игры "Пицца-мастер" с использованием языков HTML5, CSS3 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: объявим doctype, создадим корневой элемент html с языком ru, в разделе head добавим метатеги для кодировки и адаптивности, установим заголовок страницы. В теле документа разместим игровой интерфейс с контейнерами для отображения уровня, таймера, заказа, основы пиццы, ингредиентов и кнопок управления, оставив области для CSS-стилей и JavaScript-кода пустыми для последующей реализации игровой логики.

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

</style>
</head>
<body>

<div id="game">
  <h1>🍕 Пицца-мастер</h1>
  <p class="subtitle">Готовь по заказу — быстро и правильно!</p>

  <div class="info-row">
    <div>Уровень: <span id="level">1</span></div>
    <div>Осталось: <span id="left">3</span></div>
    <div>Время: <span id="timer">60</span> сек</div>
  </div>

  <div id="order">Загрузка...</div>

  <div id="pizza-base"></div>

  <div id="ingredients"></div>

  <button id="serve">Подать пиццу</button>
  <button id="reset">Начать сначала</button>

  <div id="result"></div>
</div>

<script>

</script>
```

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

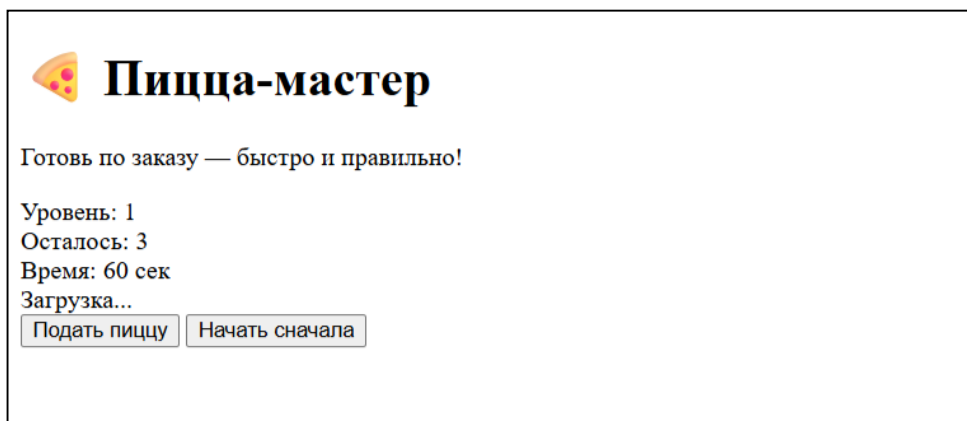


Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей: зададим базовые настройки для `body`, включая фоновый цвет и выравнивание, стилизуем заголовки и информационные блоки с использованием теплой цветовой палитры, соответствующей кулинарной тематике. Определим стили для игровой области с закругленными углами и тенями, создадим визуальное представление основы пиццы в виде круглого элемента с фоновым изображением теста, реализуем интерактивные кнопки ингредиентов с анимацией при наведении, а также стилизуем кнопки управления и различные состояния текстовых сообщений с цветовой индикацией успеха или ошибки для улучшения пользовательского опыта.

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

```
body {  
  font-family: 'Arial', sans-serif;  
  background: #f5f0e6;  
  margin: 0;  
  padding: 20px;  
  text-align: center;  
}  
  
h1 {  
  color: #d35400;  
  margin-bottom: 10px;  
}  
  
p.subtitle {  
  color: #7f8c8d;  
  font-size: 1.1em;  
  margin-bottom: 20px;  
}
```

```
#game {
  max-width: 900px;
  margin: 0 auto;
  padding: 20px;
  background: #fff8e1;
  border-radius: 15px;
  box-shadow: 0 4px 15px rgba(0,0,0,0.1);
}

.info-row {
  display: flex;
  justify-content: space-around;
  font-size: 1.3em;
  margin: 15px 0;
  font-weight: bold;
  color: #2c3e50;
}

#order {
  font-size: 1.5em;
  margin: 20px 0;
  padding: 15px;
  background: #ffecd2;
  border-radius: 10px;
  color: #c0392b;
  min-height: 50px;
  display: flex;
  align-items: center;
  justify-content: center;
}

#timer {
  font-size: 2em;
  color: #27ae60;
}

#pizza-base {
  width: 220px;
  height: 220px;
  background: #f9d371 url('https://img.freepik.com/premium-photo/pizza-dough-round-flat-white-background_77019-1014.jpg?w=600') center/cover no-repeat;
  margin: 20px auto;
  border-radius: 50%;
  position: relative;
  border: 4px solid #c8a45d;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

.ingredient {
  width: 55px;
  height: 55px;
  position: absolute;
```

```
border-radius: 50%;
display: flex;
align-items: center;
justify-content: center;
font-size: 20px;
font-weight: bold;
color: white;
cursor: pointer;
box-shadow: 0 3px 6px rgba(0,0,0,0.2);
border: 1px solid #555;
}

#ingredients {
display: flex;
flex-wrap: wrap;
justify-content: center;
gap: 20px;
margin: 30px 0;
}

.ingredient-item {
display: flex;
flex-direction: column;
align-items: center;
width: 90px;
}

.ingredient-btn {
width: 70px;
height: 70px;
border: 3px solid #ddd;
border-radius: 15px;
cursor: pointer;
transition: all 0.2s ease;
display: flex;
align-items: center;
justify-content: center;
font-size: 28px;
box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

.ingredient-btn:hover {
transform: scale(1.1);
border-color: #d35400;
box-shadow: 0 6px 12px rgba(0,0,0,0.15);
}

.ingredient-label {
margin-top: 6px;
font-size: 0.9em;
font-weight: bold;
color: #333;
```

```
text-align: center;
word-wrap: break-word;
}

button {
padding: 12px 24px;
font-size: 1.2em;
border: none;
border-radius: 10px;
cursor: pointer;
margin: 10px;
box-shadow: 0 3px 6px rgba(0,0,0,0.1);
}

#serve {
background: #27ae60;
color: white;
}

#reset {
background: #3498db;
color: white;
}

button:disabled {
cursor: not-allowed;
opacity: 0.6;
}

#result {
margin: 20px 0;
font-size: 1.4em;
font-weight: bold;
min-height: 36px;
}

.success { color: #27ae60; }
.error { color: #c0392b; }
.level-complete { color: #8e44ad; }
.game-over { color: #e74c3c; font-size: 1.5em; }
```

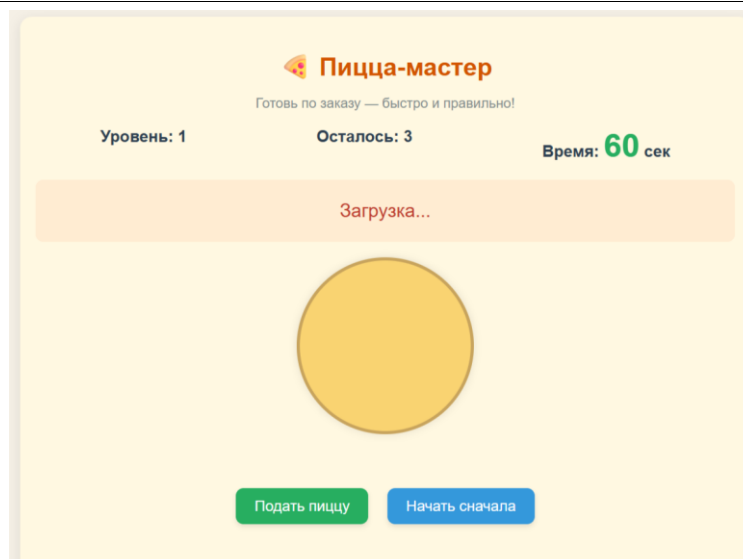


Рисунок 2. Добавление CSS-стилей

Затем инициализируем данные игры: создадим массив ингредиентов с понятными названиями, цветами и иконками для визуального представления, определим коллекцию рецептов пицц с точными последовательностями добавления компонентов, а также настроим параметры уровней сложности с постепенно увеличивающимся количеством целевых пицц и уменьшающимся временем на их приготовление, что создает прогрессирующий вызов для игрока.

Вставим данный код после закрывающего тега `</div>` и после открывающего тега `<script>`:

```
// Ингредиенты с понятными названиями и иконками
const ingredients = [
  { name: "tomato", label: "Томатный соус", color: "#d94135", symbol: "🍅", textColor:
"white" },
  { name: "cheese", label: "Сыр", color: "#f9d758", symbol: "🧀", textColor: "black" },
  { name: "pepperoni", label: "Пепперони", color: "#9e2a2b", symbol: "🍖", textColor:
"white" },
  { name: "mushroom", label: "Грибы", color: "#b89470", symbol: "🍄", textColor:
"white" },
  { name: "olive", label: "Оливки", color: "#3d5623", symbol: "🫒", textColor: "white"
},
  { name: "onion", label: "Лук", color: "#f2d79c", symbol: "🧅", textColor: "black" },
  { name: "ham", label: "Ветчина", color: "#e57e6e", symbol: "🍖", textColor: "white"
},
  { name: "pineapple", label: "Ананас", color: "#f7c548", symbol: "🍍", textColor:
"black" }
];

// Рецепты пицц
const recipes = [
  { name: "Маргарита", order: "Томатный соус, Сыр", sequence: ["tomato", "cheese"]
},
```

```

    { name: "Пепперони", order: "Томатный соус, Сыр, Пепперони", sequence:
["tomato", "cheese", "pepperoni"] },
    { name: "Гавайская", order: "Томатный соус, Сыр, Ветчина, Ананас", sequence:
["tomato", "cheese", "ham", "pineapple"] },
    { name: "Грибная", order: "Томатный соус, Сыр, Грибы", sequence: ["tomato",
"cheese", "mushroom"] },
    { name: "Оливковая", order: "Томатный соус, Сыр, Оливки", sequence: ["tomato",
"cheese", "olive"] },
    { name: "С луком и грибами", order: "Томатный соус, Сыр, Грибы, Лук", sequence:
["tomato", "cheese", "mushroom", "onion"] }
  ];

  // Уровни: количество пицц и время
  const levels = [
    { target: 3, time: 60, name: "Новичок" },
    { target: 5, time: 50, name: "Мастер" },
    { target: 7, time: 40, name: "Гуру пиццы" }
  ];

```

Следом объявим ключевые игровые переменные для отслеживания текущего уровня, оставшихся пицц, времени и состояния игры, получим ссылки на DOM-элементы для последующего манипулирования ими. Реализуем функцию `initIngredients` для динамического создания интерактивных кнопок ингредиентов на основе предопределенного массива данных, каждая из которых будет иметь уникальный цвет, символ и обработчик события, а также разработаем механизм `addIngredient` для добавления визуальных элементов на основу пиццы со случайным расположением и одновременным сохранением последовательности в специальном массиве для последующей проверки соответствия рецепту.

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

```

// Игровые переменные
let currentLevel = 0;
let pizzasLeft = 3;
let timer = 60;
let timerInterval = null;
let currentRecipe = null;
let pizzaStack = [];

// DOM-элементы
const levelEl = document.getElementById("level");
const leftEl = document.getElementById("left");
const timerEl = document.getElementById("timer");
const orderEl = document.getElementById("order");
const pizzaBase = document.getElementById("pizza-base");
const ingredientsContainer = document.getElementById("ingredients");
const serveBtn = document.getElementById("serve");
const resetBtn = document.getElementById("reset");
const resultEl = document.getElementById("result");

```

```
// Инициализация кнопок ингредиентов
function initIngredients() {
  ingredientsContainer.innerHTML = "";
  ingredients.forEach(ing => {
    const item = document.createElement("div");
    item.className = "ingredient-item";

    const btn = document.createElement("div");
    btn.className = "ingredient-btn";
    btn.style.backgroundColor = ing.color;
    btn.dataset.name = ing.name;
    btn.title = ing.label;
    btn.innerHTML = ing.symbol;
    btn.style.color = ing.textColor;
    btn.addEventListener("click", () => addIngredient(ing));

    const label = document.createElement("div");
    label.className = "ingredient-label";
    label.textContent = ing.label; // Например: "Томатный соус"

    item.appendChild(btn);
    item.appendChild(label);
    ingredientsContainer.appendChild(item);
  });
}

// Добавление ингредиента на пиццу
function addIngredient(ing) {
  if (timer <= 0 || !currentRecipe) return;

  const ingr = document.createElement("div");
  ingr.className = "ingredient";
  ingr.style.backgroundColor = ing.color;
  ingr.style.color = ing.textColor;
  ingr.innerHTML = ing.symbol;
  ingr.dataset.name = ing.name;

  const angle = Math.random() * 360;
  const radius = 60 + Math.random() * 30;
  const x = 100 + radius * Math.cos(angle * Math.PI / 180);
  const y = 100 + radius * Math.sin(angle * Math.PI / 180);
  ingr.style.left = `${x - 27.5}px`;
  ingr.style.top = `${y - 27.5}px`;

  pizzaBase.appendChild(ingr);
  pizzaStack.push(ing.name);
}
```

Теперь реализуем основные игровые процессы: создадим функцию `startLevel` для инициализации параметров уровня, `newOrder` для генерации случайного заказа и `startTimer` для обратного отсчета времени. Добавим

обработчик события для кнопки "Подать пиццу", который проверяет соответствие собранной пиццы рецепту через функцию `arraysEqual`, и реализуем логику перехода на следующий уровень при успешном выполнении всех заказов или завершения игры при истечении времени. Завершим код функциями сброса состояния игры, сравнения массивов и обработки кнопки перезапуска, после чего инициализируем игровые элементы и запустим первый уровень для начала игрового процесса.

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

```
// Начало уровня
function startLevel() {
  const levelData = levels[currentLevel];
  pizzasLeft = levelData.target;
  timer = levelData.time;

  leftEl.textContent = pizzasLeft;
  levelEl.textContent = currentLevel + 1;
  timerEl.textContent = timer;

  resultEl.textContent = `🕒 Уровень ${currentLevel + 1}: "${levelData.name}". Сделай ${pizzasLeft} пицц!`;
  resultEl.className = "success";

  setTimeout(newOrder, 1500);
}

// Новый заказ
function newOrder() {
  resetPizza();
  currentRecipe = recipes[Math.floor(Math.random() * recipes.length)];
  orderEl.textContent = `Пицца "${currentRecipe.name}": ${currentRecipe.order}`;
  startTimer();
}

// Запуск таймера
function startTimer() {
  clearInterval(timerInterval);
  timerEl.textContent = timer;
  timerInterval = setInterval(() => {
    timer--;
    timerEl.textContent = timer;
    if (timer <= 0) {
      clearInterval(timerInterval);
      gameOver("🕒 Время вышло!");
    }
  }, 1000);
}

// Проверка пиццы
serveBtn.addEventListener("click", () => {
```

```
if (!currentRecipe) return;

if (arraysEqual(pizzaStack, currentRecipe.sequence)) {
  pizzasLeft--;
  leftEl.textContent = pizzasLeft;
  resultEl.textContent = "✅ Правильно! Готово!";
  resultEl.className = "success";

  if (pizzasLeft <= 0) {
    levelComplete();
  } else {
    setTimeout(newOrder, 1000);
  }
} else {
  resultEl.textContent = "❌ Неправильно! Проверь последовательность.";
  resultEl.className = "error";
}
});

// Уровень пройден
function levelComplete() {
  clearInterval(timerInterval);
  resultEl.textContent = `🎉 Уровень ${currentLevel + 1} пройден!`;
  resultEl.className = "level-complete";
  serveBtn.disabled = true;

  currentLevel++;

  if (currentLevel >= levels.length) {
    setTimeout(() => {
      resultEl.textContent = "🏆 Поздравляем! Ты — мастер пиццы!";
      resultEl.className = "success";
      resetBtn.textContent = "Сыграть снова";
    }, 1500);
  } else {
    setTimeout(() => {
      startLevel();
      serveBtn.disabled = false;
    }, 1500);
  }
}

// Игра окончена
function gameOver(msg) {
  clearInterval(timerInterval);
  resultEl.textContent = msg;
  resultEl.className = "game-over";
  serveBtn.disabled = true;
  orderEl.textContent = "🏹 Игра окончена!";
}
```

```
// Очистка пиццы
function resetPizza() {
  pizzaStack = [];
  Array.from(pizzaBase.children).forEach(child => {
    if (child.classList.contains("ingredient")) {
      pizzaBase.removeChild(child);
    }
  });
}
```

```
// Сравнение последовательности
function arraysEqual(a, b) {
  if (a.length !== b.length) return false;
  for (let i = 0; i < a.length; i++) {
    if (a[i] !== b[i]) return false;
  }
  return true;
}
```

```
// Кнопка "Начать сначала"
resetBtn.addEventListener("click", () => {
  currentLevel = 0;
  clearInterval(timerInterval);
  serveBtn.disabled = false;
  resetPizza();
  resultEl.textContent = "";
  resetBtn.textContent = "Начать сначала";
  startLevel();
});
```

```
// Запуск игры
initIngredients();
startLevel();
```



Рисунок 3. Главная страница



Рисунок 4. Процесс игры

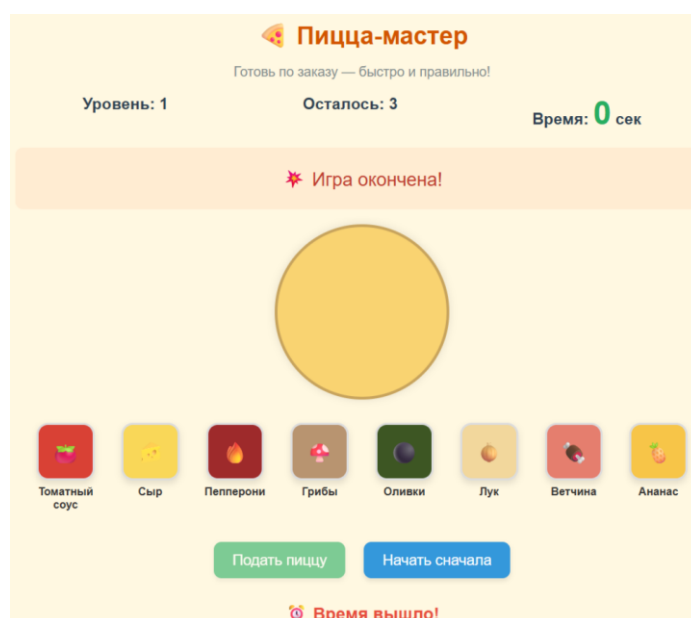


Рисунок 5. Проигрыш

4 Выводы

В данной работе подробно рассмотрено создание образовательной игры "Пицца-мастер" на современном веб-стеке с использованием технологий HTML5, CSS3 и JavaScript, которая включает реалистичную механику приготовления пиццы, систему проверки соответствия рецепту, таймер обратного отсчета и адаптивный интерфейс для различных устройств. Реализованы ключевые игровые механизмы: генерация случайных заказов, визуализация процесса добавления ингредиентов, многоуровневая система сложности с прогрессивным увеличением требований и уменьшением времени на выполнение, а также система обратной связи с пользователем. Разработан интуитивно понятный игровой интерфейс с визуальным представлением ингредиентов, областью сборки пиццы и анимированными уведомлениями о результатах игры.

Предложенное решение ориентировано на преподавателей и студентов, заинтересованных в применении геймификации и современных веб-технологий для повышения эффективности образовательного процесса, развития навыков программирования 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.
3. Королюк Я. А., Куликова Н. Ю. Опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайнобучения 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. М.: Перо, 2015. 242 с.