

Интерактивная игра «Фруктовая корзина» на языке JavaScript с использованием jQuery

Кокухина Софья Александровна

Приамурский государственный университет им.Шолом-Алейхема

Студент

Аннотация

Целью данного исследования является разработка браузерной игры «Фруктовая корзина» с использованием HTML, CSS и библиотеки jQuery. Реализация проекта позволяет пользователю управлять игровым процессом: ловить фрукты и ягоды в разные корзины, накапливать баллы и сравнивать результаты в таблице рекордов. Работа демонстрирует основы веб-разработки, взаимодействие фронтенда и бэкенда (через PHP-скрипт сохранения результатов), применение таймеров, обработку событий и анимацию объектов. Проект может использоваться в образовательных целях для изучения динамики DOM, работы с JavaScript и принципов интерактивных интерфейсов.

Ключевые слова JavaScript, jQuery, HTML, CSS, PHP, браузерная игра.

Interactive game “Fruit Basket” using JavaScript and jQuery

Kokukhina Sofya Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of this research is to develop a Fruit Basket browser game using HTML, CSS, and the jQuery library. The implementation of the project allows the user to control the gameplay: catch fruits and berries in different baskets, accumulate points and compare the results in the high score table. The work demonstrates the basics of web development, the interaction of the front-end and backend (through a PHP script for saving results), the use of timers, event processing and animation of objects. The project can be used for educational purposes to study DOM dynamics, work with JavaScript, and principles of interactive interfaces.

Keywords JavaScript, jQuery, HTML, CSS, PHP, browser game.

1 Введение

1.1 Актуальность

Разработка браузерных игр является эффективным способом изучения веб-технологий. Язык JavaScript и библиотека jQuery позволяют реализовывать динамические интерфейсы, а HTML и CSS отвечают за визуальную часть. Создание игры «Фруктовая корзина» помогает на практике

освоить основы клиентской логики, обработку событий и работу с DOM, а также взаимодействие с серверной частью для хранения рекордов.

1.2 Обзор исследований

В статье М.И. Притчиной и Е.А. Зотовой рассматривается разработка приложения с использованием технологий `phonegap`, `javascript`, `html` и `jquery mobile` [1]. Б.Б. Картабаева и А.Ш. Бегалин описали процесс разработки и использования интернет-сайта на конкретном примере с использованием фреймворка Angular Java-Script (AngularJS), библиотек `jQuery` и `Modernizr` [2]. В статье Е.И. Белоруковой рассматривается проблема выбора языка веб-программирования при обучении веб-разработке. Представлены примеры выражения концепций программирования, которые могут помочь продемонстрировать различия и сходства языков [3]. И.А. Кожунков рассказал о использовании языка `javascript` при создании веб-страниц [4]. Д.Е. Панов разработал программу, предназначенную для изучения студентами основ, необходимых для полноценного начала работы в сфере Web-программирования и Web-дизайна; правил создания, применения и использования JavaScript [5].

1.3 Цель исследования

Цель исследования — разработка единого проекта, включающего фронтенд (HTML, CSS, `jQuery`) и бэкенд (PHP), для реализации интерактивной игры «Фруктовая корзина» с учётом учёта времени, очков и отображения таблицы рекордов.

2 Материалы и методы

Для реализации проекта используется набор классических веб-технологий:

HTML5 — язык разметки, определяющий структуру приложения. Вёрстка построена по принципу экранов («стартовый экран», «игровой экран», «экран окончания»), каждый из которых скрывается или показывается динамически с помощью классов CSS.

CSS3 — каскадные таблицы стилей, отвечающие за визуальное оформление. В проекте используются гибкие контейнеры (`flexbox`), адаптивные блоки и стили для элементов интерфейса: корзин, фруктов, кнопок и таблицы рекордов.

JavaScript (с библиотекой `jQuery`) — основной язык логики игры. `jQuery` обеспечивает простое взаимодействие с DOM: выбор элементов, изменение их свойств, анимация падения фруктов, обработка событий кликов и перетаскивания.

PHP — серверная технология для обработки и сохранения результатов игры. Через AJAX-запросы результаты передаются на сервер и заносятся в файл (или базу данных). В дальнейшем сервер возвращает список лучших результатов для отображения в таблице рекордов.

Среда выполнения: проект запускается в любом современном веб-браузере (Google Chrome, Mozilla Firefox, Microsoft Edge). Для разработки проекта выбран редактор Visual Studio Code, но может быть выполнен в любой IDE, поддерживающей работу с веб-технологиями.

3 Результаты

Для начала необходимо создать сам проект в Visual Studio Code, в нем создаются файлы необходимые для реализации игры:

index.html — основной каркас игры: стартовый экран, поле с корзинами, экран завершения с таблицей рекордов

styles.css — оформление: стили для экранов, корзин, фруктов, кнопок и таблицы рекордов

save_score.php — серверный скрипт для сохранения и загрузки рекордов.

game.js (подключён в index.html) — основная логика игры: таймер, падение фруктов/ягод, обработка корзин, начисление баллов и переход между экранами.

В главном файле game.js задаются константы игры: время партии (20 секунд), набор фруктов и ягод, начисляемые очки и штрафы. Для хранения текущего состояния создаётся объект gameState, в котором содержатся имя игрока, количество очков, таймер, интервалы, выбранный фрукт и состояние корзин.

Такое решение позволяет централизованно управлять логикой игры и сбрасывать параметры при рестарте (Рис. 1).

```

3      const GAME_DURATION = 20; // 20 секунд
4      const FRUIT_TYPES = {
5          fruits: ['apple', 'banana', 'pear'],
6          berries: ['strawberry', 'blackberry', 'raspberry']
7      };
8      const POINTS_PER_FRUIT = 10;
9      const PENALTY_PER_MISSED = 1;
10
11      // Состояние игры
12      let gameState = {
13          playerName: '',
14          score: 0,
15          timeLeft: GAME_DURATION,
16          isPaused: false,
17          isTestMode: false,
18          timerInterval: null,
19          fruitInterval: null,
20          selectedFruit: null,
21          baskets: {
22              fruits: 0,
23              berries: 0
24          }
25      };
26

```

Рисунок 1 - Константы и структура состояния игры

Через jQuery выбираются все основные DOM-элементы (стартовый экран, игровое поле, экран окончания, счётчики, кнопки) (Рис.2).

```

28      const $startScreen = $('#start-screen');
29      const $gameScreen = $('#game-screen');
30      const $endScreen = $('#end-screen');
31      const $playerNameInput = $('#player-name');
32      const $startGameBtn = $('#start-game-btn');
33      const $currentPlayer = $('#current-player');
34      const $timer = $('#timer');
35      const $score = $('#score');
36      const $pauseBtn = $('#pause-btn');
37      const $gameField = $('#game-field');
38      const $finalScore = $('#final-score');
39      const $highScoresTable = $('#high-scores-table tbody');
40      const $restartGameBtn = $('#restart-game-btn');
41

```

Рисунок 2 – Добавление DOM элементов

Функция `initGame()` настраивает обработчики событий: ввод имени игрока, старт игры, пауза по кнопке или пробелу, рестарт. При переключении экранов используется добавление/удаление класса `active`, что соответствует стилям в CSS (Рис. 3).

```
43     initGame();
44
45     function initGame() {
46         // Обработчики событий стартового экрана
47         $playerNameInput.on('input', function() {
48             const name = $(this).val().trim();
49             $startGameBtn.prop('disabled', name.length === 0);
50
51             // Проверка на тестовый режим
52             gameState.isTestMode = name.toLowerCase() === 'tester';
53         });
54
55         $startGameBtn.on('click', startGame);
56
57         // Обработчики событий игры
58         $pauseBtn.on('click', togglePause);
59         $(document).on('keydown', function(e) {
60             if (e.key === ' ') {
61                 togglePause();
62             }
63         });
64
65         // Обработчик кнопки "Играть снова"
66         $restartGameBtn.on('click', function() {
67             $endScreen.removeClass('active');
68             $startScreen.addClass('active');
69             resetGame();
70         });
71     }
72 }
```

Рисунок 3 – Инициализация игры и обработка экранов

Метод startGame() выполняет подготовку:

сохраняет имя игрока; обнуляет состояние с помощью resetGame(); запускает таймер и генерацию фруктов; подключает обработчики кликов по фруктам и корзинам (Рис.4).

```

73 function startGame() {
74     gameState.playerName = $playerNameInput.val().trim();
75
76     // Отображение имени игрока (скрываем если тестовый режим)
77     $currentPlayer.text(gameState.isTestMode ? 'Тестовый режим' : gameState.playerName);
78
79     // Сброс состояния игры
80     resetGame();
81
82     // Переключение экранов
83     $startScreen.removeClass('active');
84     $gameScreen.addClass('active');
85
86     // Запуск таймера (если не тестовый режим)
87     if (!gameState.isTestMode) {
88         startTimer();
89     }
90
91     // Запуск генерации фруктов
92     startFruitGeneration();
93
94     // Обработчики кликов на фрукты и корзины
95     $gameField.on('click', '.fruit', onFruitClick);
96     $gameField.on('click', '.basket', onBasketClick);
97 }
98

```

Рисунок 4 – Функция запуска игры

Функция `resetGame()` очищает игровое поле, обнуляет счёт, таймер и корзины, а также сохраняет результат в `localStorage` (если это не тестовый режим) (Рис.5).

```

99 function resetGame() {
100     // Сохраняем текущий результат перед сбросом
101     if (gameState.playerName && !gameState.isTestMode && gameState.score > 0) {
102         saveScoreToDatabase(gameState.playerName, gameState.score);
103     }
104     // Очистка игрового поля
105     $gameField.find('.fruit').remove();
106
107     // Сброс состояния
108     gameState = {
109         ...gameState,
110         score: 0,
111         timeLeft: GAME_DURATION,
112         isPaused: false,
113         selectedFruit: null,
114         baskets: {
115             fruits: 0,
116             berries: 0
117         }
118     };
119
120     // Обновление UI
121     updateTimerDisplay();
122     $score.text('0');
123     $('.basket-count').text('0');
124     $pauseBtn.text('Пауза');
125
126     // Остановка интервалов
127     clearInterval(gameState.timerInterval);
128     clearInterval(gameState.fruitInterval);
129 }
130

```

Рисунок 5 – Функция перезапуска игры

Функция `startTimer()` запускает интервал с шагом в 1 секунду. В каждом шаге уменьшается оставшееся время и обновляется отображение в формате «мм:сс». Когда счётчик достигает нуля, вызывается `endGame()`.

Это обеспечивает чёткое ограничение игровой сессии (рис. 6).

```
131     function startTimer() {  
132         updateTimerDisplay();  
133  
134         gameState.timerInterval = setInterval(function() {  
135             if (!gameState.isPaused) {  
136                 gameState.timeLeft--;  
137                 updateTimerDisplay();  
138  
139                 if (gameState.timeLeft <= 0) {  
140                     endGame();  
141                 }  
142             }  
143         }, 1000);  
144     }
```

Рисунок 6 – Запуск и остановка таймера

Функция `startFruitGeneration()` каждую секунду создаёт новый фрукт (`createFruit()`).

При создании объекта случайным образом выбирается тип (фрукт или ягода), затем элемент `<div class="fruit">` добавляется в игровое поле и начинает анимироваться вниз с помощью `setInterval`.

Если фрукт «пролетает» мимо корзины и достигает нижней границы, он удаляется, а игрок получает штраф (минус баллы).

Такая механика создаёт динамику игры и стимулирует быструю реакцию игрока (рис. 7).

```

152     function startFruitGeneration() {
153         gameState.fruitInterval = setInterval(function() {
154             if (!gameState.isPaused) {
155                 createFruit();
156             }
157         }, 1000);
158     }
159
160     function createFruit() {
161         // Случайный выбор типа плода (фрукт или ягода)
162         const isFruit = Math.random() > 0.5;
163         const type = isFruit ? 'fruits' : 'berries';
164         const fruitType = FRUIT_TYPES[type][Math.floor(Math.random() * FRUIT_TYPES[type].length)];
165
166         // Создание элемента фрукта
167         const $fruit = $('<div class="fruit"></div>');
168         $fruit.css({
169             'background-image': `url('images/${fruitType}.png')`,
170             left: `${Math.random() * ($gameField.width() - 50)}px`,
171             top: '-50px'
172         });
173         $fruit.data('type', type);
174
175         // Добавление на игровое поле
176         $gameField.append($fruit);
177
178         // Анимация падения
179         const fallSpeed = 2 + Math.random() * 3; // Случайная скорость
180         const fallAnimation = setInterval(function() {
181             if (!gameState.isPaused) {
182                 const currentTop = parseInt($fruit.css('top'));
183                 const newTop = currentTop + fallSpeed;
184                 $fruit.css('top', `${newTop}px`);
185
186                 // Проверка достижения нижней границы
187                 if (newTop > $gameField.height()) {

```

Рисунок 7 – Генерация фруктов и анимация падения

При завершении (endGame()): останавливаются все интервалы; переключается экран на финальный; выводится результат; загружается таблица рекордов (Рис.8).

```

function endGame() {
    // Остановка интервалов
    clearInterval(gameState.timerInterval);
    clearInterval(gameState.fruitInterval);

    // Переключение экранов
    $gameScreen.removeClass('active');
    $endScreen.addClass('active');

    // Отображение финального счета
    $finalScore.text(gameState.score);

    // Загрузка и отображение таблицы рекордов
    loadHighScores();
}

```

Рисунок 8 – Конец игры

Метод loadHighScores() извлекает данные из localStorage, сортирует их и отображает первые 10 лучших. Текущий игрок подсвечивается цветом, а если он не попал в топ-10, его результат добавляется отдельно внизу (Рис.9).

```
function loadHighScores() {  
  // Получаем результаты из localStorage  
  const scores = JSON.parse(localStorage.getItem('fruitBasketScores')) || [];  
  
  // Добавляем текущий результат, если он попадает в топ-10  
  let currentPlayerInTop = false;  
  const allScores = [...scores];  
  
  // Если текущий игрок не в тестовом режиме и набрал баллы, добавляем его результат  
  if (!gameState.isTestMode && gameState.score > 0) {  
    allScores.push({ name: gameState.playerName, score: gameState.score });  
    allScores.sort((a, b) => b.score - a.score);  
  }  
  
  // Очистка таблицы  
  $highScoresTable.empty();  
  
  // Заполнение таблицы  
  for (let i = 0; i < 10; i++) {  
    if (i >= allScores.length) break;  
  
    const player = allScores[i];  
    const isCurrent = player.name === gameState.playerName && player.score === gameState.score;  
  
    if (isCurrent) currentPlayerInTop = true;  
  
    const $row = $('`  
      <tr ${isCurrent ? 'class="current-player-score"' : ''}>  
        <td>${i + 1}</td>  
        <td>${player.name}</td>  
        <td>${player.score}</td>  
      </tr>  
    `);  
  
    $highScoresTable.append($row);  
  }  
}
```

Рисунок 9 – Загрузка результатов в таблицу

Функция saveScoreToDatabase() отвечает за добавление новой записи в список рекордов и обрезает его до топ-100 (Рис. 10).

```
327 function saveScoreToDatabase(name, score) {
328     // Получаем текущие результаты из localStorage
329     let scores = JSON.parse(localStorage.getItem('fruitBasketScores')) || [];
330
331     // Добавляем новый результат
332     scores.push({ name, score, date: new Date().toISOString() });
333
334     // Сортируем по убыванию баллов
335     scores.sort((a, b) => b.score - a.score);
336
337     // Сохраняем только топ-100 результатов
338     if (scores.length > 100) {
339         scores = scores.slice(0, 100);
340     }
341
342     // Сохраняем обратно в localStorage
343     localStorage.setItem('fruitBasketScores', JSON.stringify(scores));
344 }
```

Рисунок 10 – Сохранение результатов

После написания кода проект запускается из файла Index.html, игроку предоставляется две корзины: для фруктов и для ягод. На игровом поле сверху начинают появляться и падать объекты — изображения различных фруктов (яблоко, банан, груша) и ягод (клубника, малина, ежевика) (Рис.11).

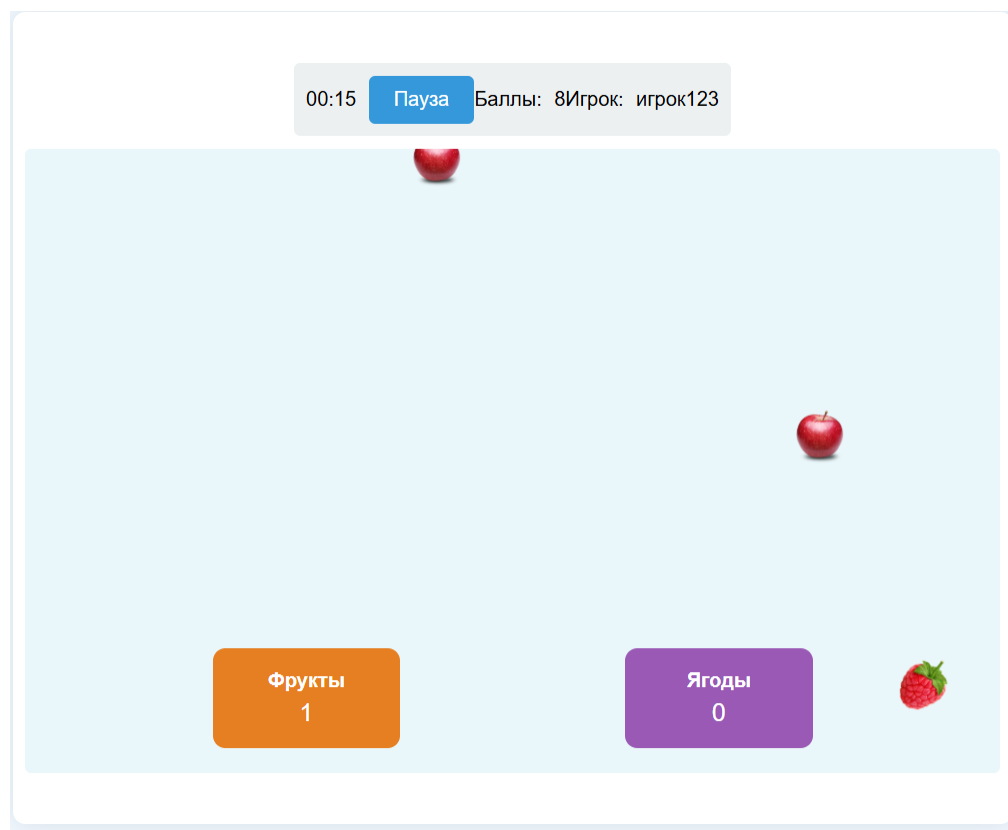


Рисунок 11 – Игровое поле

Задача игрока — щелчком мыши выбрать фрукт и перенести его в соответствующую корзину. При правильном действии начисляются очки, а

счётчик корзины увеличивается. Если же объект оказывается в неправильной корзине, очки вычитаются, а содержимое корзины сбрасывается.

После завершения времени игра переключается на экран с результатами, где показывается финальный счёт и таблица рекордов (Рис. 12). Текущий игрок подсвечивается отдельным цветом, что позволяет легко найти свой результат.



Рисунок 12 – Финальное окно игры

4

Выводы. Разработка игры «Фруктовая корзина» продемонстрировала возможности создания полноценного интерактивного приложения на основе простых веб-технологий. Использование связки HTML, CSS и jQuery позволило реализовать динамический интерфейс, включающий таймер обратного отсчёта, анимацию падения объектов и обработку действий пользователя. Дополнительная интеграция с локальным хранилищем браузера обеспечила сохранение и отображение результатов, что придало игре соревновательный характер.

В ходе работы удалось объединить визуальное оформление и игровую механику в единую систему, где игрок вовлекается в процесс сортировки фруктов и ягод, а итоговый результат фиксируется в таблице рекордов. Такой подход делает проект полезным не только как средство развлечения, но и как учебный пример, позволяющий на практике освоить основы событийного программирования, работу с DOM-структурой и принципы построения интерактивных интерфейсов.

Библиографический список

1. Притчина М. И., Зотова Е. А. Разработка приложения с использованием технологий PhoneGap, JavaScript, HTML и jQuery Mobile // Вестник МГУП имени Ивана Федорова. 2013. № 2. С. 50-54.
2. Картабаева Б. Б., Бегалин А. Ш. Разработка интернет-сайта с использованием фреймворка Angular Java-Script, библиотек jquery и Modernizr // Теория и практика современной науки. 2016. № 5(11). С. 411-418.
3. Белорукова Е. И., Государев И. Б. Использование возможностей языков GO и JAVASCRIPT при обучении веб-разработке на различных уровнях образования // Современное образование: традиции и инновации. 2021. № 2. С. 103-106.
4. Кожунков И. А. Использование языка JavaScript при создании веб-страниц // Информационные и коммуникационные технологии в экономике, управлении и образовании : доклады участников Международной заочной научно-практической конференции. Энгельс: Редакционно-издательский центр Поволжского кооперативного института. 2015. С. 98-99.
5. Свидетельство о государственной регистрации программы для ЭВМ № 2015615497 Российская Федерация. Web-решение для изучения языков программирования HTML и JavaScript с использованием библиотеки jQuery: № 2015612647 : заявл. 01.04.2015 : опубл. 19.05.2015 / Панов Д. Е., Жафярова Ф. С., Журавлев А. Ю.; заявитель Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Ивановский государственный университет».