УДК 004

Создание браузерной игры на языке JavaScript по мотивам Mortal Kombat

Халиманенков Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматриваются разработка игры Mortal Kombat на языке JavaScript для браузеров, в которой используется работа с RESTAPI при помощи AJAX запросов для получения списков персонажей. Реализованы функции случайных атак и блоков соперника.
Ключевые слова: браузерная игра, JavaScript, HTML, CSS, RESTAPI, веб-приложение.

Creating a JavaScript browser game based on Mortal Kombat

Khalimanenkov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the development of the game Mortal Kombat in JavaScript for browsers, which uses RESTAPI using AJAX requests to get lists of characters. The functions of random opponent attacks and blocks are implemented.
Keywords: browser game, JavaScript, HTML, CSS, RESTAPI, web application.

Создание игр в браузере с помощью JavaScript является отличным способом для изучения основ этого языка программирования. Такие сайты уже являются полноценными небольшими веб-приложениями, так как используют AJAX запросы к API сервера для работы и рендера в DOM некоторых модулей меню игры. А также к таким играм есть доступ с любого устройства, будь то ПК или смартфон. Единственное условие – наличие интернета.

Ключевые вопросы разработки однопользовательских двумерных игр на языке JavaScript рассматриваются в учебном пособии С. А. Беляева [1]. Е. В. Пантелеева отразила сущность разработки сайта с использованием языка разметки HTML, особенности таблицы стилей CSS и языка программирования JavaScript [2]. В работе Е.В.Соловьева и Т.А.Максимова рассматривается разработка браузерной игры на языке JavaScript [3]. В своей статье Е.Н. Поварницын рассматривает создание игры «Тетрис» на языке JavaScript, HTML и таблицей стилей CSS [4]. S. Nabhani и соавторы рассматривают сетевой прототип игры «Pharmacy Challenge» на студентах-

фармацевтах для проведения исследования на оценку игры с точки зрения дизайна и оценивают влияние на успеваемость и уверенность учащихся [5].

Для создания игры по Mortal Kombat использовался чистый JavaScript [6] без сторонних библиотек и фреймворков, а также язык разметки HTML5 [7] и язык каскадных стилей CSS [8].

В начале игры выводится меню из списка всех персонажей. Получаем этот список через fetch запрос

```
const players = await fetch('https://reactmarathon-api.herokuapp.com/api/mk/players').then(res => res.json());
```

Потом на основе полученного массива со всеми игроками создаём меню (рис. 1), в котором для каждого элемента добавляем слушателя событий по нажатию левой кнопки мыши. Сначала персонажа выбирает игрок, после чего переменная canChooseCharacter меняется на false и у игрока отключается возможность выбирать персонажа и вызывать зелёную обводку при наведении на картинки бойцов. После этого имитируется выбор персонажа компьютером. У компьютера обводка выбранных бойцов красная (рис. 2). Выбранные персонажи записываются в local storage браузера, после чего происходит переход на другую страницу, где происходит основной игровой процесс.

```
let canChooseCharacter = true;
players.forEach(item => {
        const el = createElement('div', ['character', `div${item.id}`]);
        const img = createElement('img');
        let mouseMove = () => {
            if (canChooseCharacter && imgSrc === null) {
                imgSrc = item.img;
                const $img = createElement('img');
                $img.src = imgSrc;
                $player.appendChild($img);
            }
        }
        let mouseOut = () => {
            if (canChooseCharacter && imgSrc) {
                imgSrc = null;
                $player.innerHTML = '';
            }
        }
        el.addEventListener('mouseout', mouseOut);
        el.addEventListener('mousemove', mouseMove);
        el.addEventListener('click', (e) => {
            e.preventDefault();
            canChooseCharacter = false;
            $parent.style.pointerEvents = 'none';
            localStorage.setItem('player1', JSON.stringify(item));
            el.classList.add('active');
            async function initEnemy() {
                imgEnemySrc = null;
                $enemy.innerHTML = '';
                let enemyCharacter = players[getRandom(23)-1];
```

```
                    if (enemyCharacter.id == 11) {
                        enemyCharacter.id == 12;
                    }
                                                        let      enemyEl      =
document.querySelector(`.div${enemyCharacter.id}`);

                    if (!!(document.querySelector(`.character-enemy`))) {
                        enemyEl.classList.remove(`character-enemy`);
                    }
                    enemyEl.classList.add(`character-enemy`);
                    console.log(enemyCharacter);
                                                localStorage.setItem('player2',
JSON.stringify(enemyCharacter));
                    imgEnemySrc = enemyCharacter.img;
                    const $img = createElement('img');
                    $img.src = imgEnemySrc;
                    $enemy.appendChild($img);
                }
                setInterval(async () => {
            // здесь имитируется выбор персонажа компьютером
                    if (!!(document.querySelector(`.character-enemy`))) {
                            let  enemyEl  =  document.querySelector(`.character-
enemy`);
                        enemyEl.classList.remove(`character-enemy`);
                    }
                    initEnemy();
                }, 1000);
                setTimeout(async() => {
                    window.location.pathname = 'arena.html';
                }, 4500);
            });
            img.src = item.avatar;
            img.alt = item.name;
            el.appendChild(img);
            $parent.appendChild(el);
        });
```

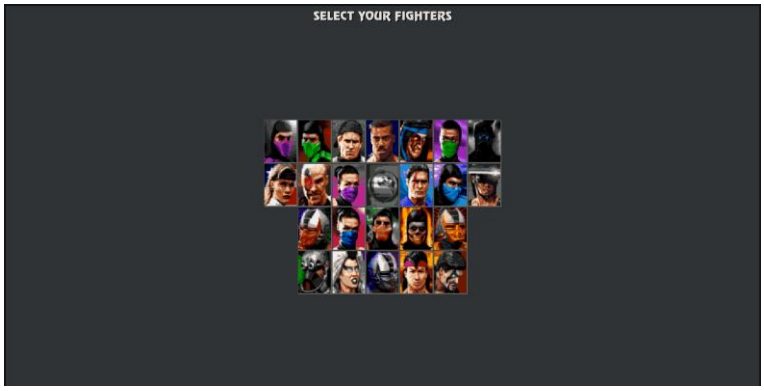

Рисунок 1 – Меню выбора персонажей

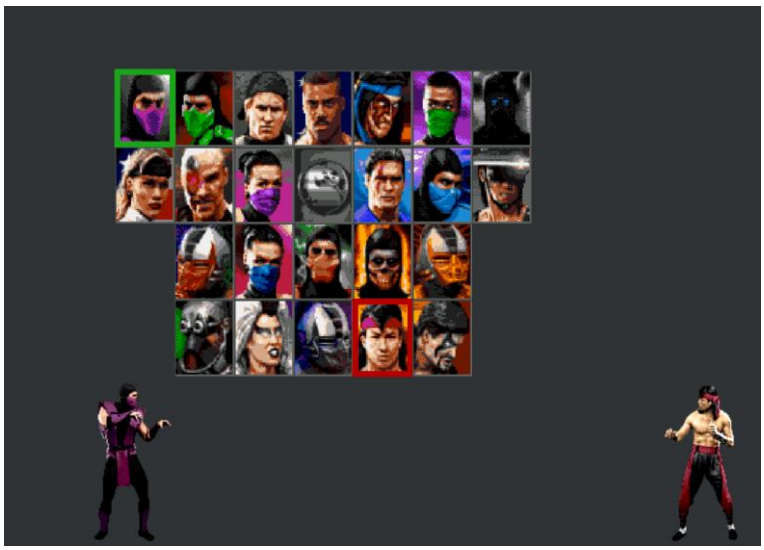

Рисунок 2 – Выбранный игроком и компьютером персонаж

После начала игры появляется анимированное изображение в формате GIF (рис. 3), а после её исчезновения появляются выбранные раннее персонажи, которые загрузились из local storage.

Рисунок 3 – Начало игры

Игровой процесс осуществляется с помощью формы с выбором места для удара и защиты (рис. 4). Сила удара определяется случайно от 1 до

30/25/20 при попадании в голову/тело/ноги соответственно. После чего выбранные игроком удар и блок отправляются в виде тела запроса POST на сервер, где сравниваются со случайным выбором сервера. Если игрок ударил в защищённое противником место, то удар не засчитывается и наоборот (рис. 5 и 6). Такая же схема работает и с противником.

```
const HIT = {
    head: 30,
    body: 25,
    foot: 20,
}

playerAttack = () => {
        const attack = {};

        for (let item of $formFight) {
            if (item.checked === true && item.name === 'hit') {
                attack.value = getRandom(HIT[item.value]);
                attack.hit = item.value;
            }
            if (item.checked === true && item.name === 'defence') {
                attack.defence = item.value;
            }
            item.checked = false;
        }

        return attack;
    }
```

Рисунок 4 – форма для выбора удара и блока

Рисунок 5 – Удар игрока

Рисунок 6 – Удар противника

У противника эти места генерируются случайным образом на сервере.

```
      enemyAttack = async ({hit, defence} = playerAttack()) => {

                        let    attack    =    await    fetch('http://reactmarathon-
api.herokuapp.com/api/mk/player/fight', {
                    method: 'POST',
                    body: JSON.stringify({
                        hit,
                        defence,
                    })
                });

                let result = await attack.json();

                return result;
            }
```

```
> fetch('http://reactmarathon-api.herokuapp.com/api/mk/player/fight', {
            method: 'POST',
            body: JSON.stringify({
                hit:'body',
                attack:'body',
            })
        }).then(response => console.log(response.json()));
< ▶ Promise {<pending>}

  ▼ Promise {<pending>} i
    ▶ [[Prototype]]: Promise
      [[PromiseState]]: "fulfilled"
    ▼ [[PromiseResult]]: Object
      ▶ player1: {value: 13, hit: 'body'}
      ▶ player2: {value: 15, hit: 'body', defence: 'foot'}
      ▶ [[Prototype]]: Object
```

Рисунок 7 – Ответ сервера на отправку формы

В игре предусмотрено логирование хода боя, где записываются удары, их сила и блоки. Также логируются начало и конец боя с результатами (рис. 8 и 9). После окончания раунда персонажи принимают позы победителя и проигравшего в зависимости от исхода поединка. Сверху появляется кнопка перезапуска (рис. 9), которая возвращает на экран выбора персонажа.

Рисунок 8 – Логирование игры

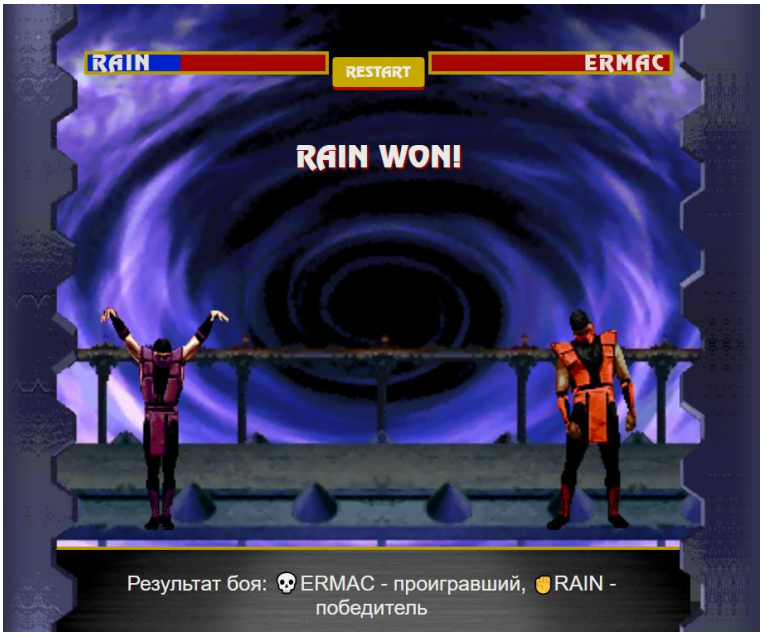

Рисунок 9 – Конец поединка с логом и кнопкой перезапуска

Таким образом, была разработана браузерная игра по мотивам Mortal Kombat, в которой использовались обработчики событий addEventListener, работа с сервером через AJAX и fetch методы языка JavaScript. Проверить работоспособность игры возможно на сервисе Netlify [9], который предоставляет бесплатный хостинг для маленьких проектов.

Библиографический список

1. Беляев С.А. Разработка игр на языке JavaScript: Учебное пособие. СПб.: Лань, 2016. 128с
2. Пантелеева Е. В. Разработка сайта с использованием языка разметки HTML, таблицы стилей CSS и языка программирования JavaScript. // Информационные системы и технологии в образовании, науке и бизнесе. 2020. С. 94-96.
3. Соловьева Е. В., Максимова Т. А. Компьютерная игра-головоломка «Судоку» на языке JavaScript // Студенческие научные достижения. 2020. С. 21-25.
4. Поварницын Е. Н. Создание браузерной игры «Тетрис» // Форум молодых ученых. 2020. №. 2. С. 289-296.
5. Nabhani S. et al. Development and evaluation of an educational game to support pharmacy students //Currents in Pharmacy Teaching and Learning. 2020. Т. 12. №. 7. С. 786-803.
6. JavaScript URL: https://ru.wikipedia.org/wiki/JavaScript (дата обращения: 03.01.2022).
7. HTML URL: https://ru.wikipedia.org/wiki/HTML (дата обращения: 03.01.2022).
8. CSS URL: https://ru.wikipedia.org/wiki/CSS (дата обращения: 03.01.2022).
9. Игра из статьи URL: https://andreykhalmk.netlify.app/ (дата обращения: 03.01.2022).