

Разработка образовательной игры «Мемори»

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель исследования - показать процесс разработки математической игры «Мемори». При создании игры использовались языки программирования HTML, CSS, JavaScript. Данная игра может использоваться в учебных целях для учащихся.

Ключевые слова: HTML, CSS, JavaScript, разработка, игра, код.

Development of the educational game "Memory"

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The purpose of the study is to show the process of developing the mathematical game "Memory". When creating the game, the programming languages HTML, CSS, and JavaScript were used. This game can be used for educational purposes for students.

Keywords: HTML, CSS, JavaScript, development, game, code.

1. Введение

1.1. Актуальность

В данной статье рассматривается создание образовательной игры «Мемори» с помощью HTML, CSS и JavaScript. Для написания кода использовался Notepad++.

Notepad++ – это бесплатный текстовый редактор и исходный код, который предоставляет множество функций и инструментов для работы с текстом. Он доступен только для операционной системы Windows.

1.2. Обзор исследований

А.С. Заблоцкая посвятила статью вопросам разработки математических настольных игр для обучающихся среднего школьного звена. Автором была проанализирована роль изучения алгебры в школе с упором на структуру математических способностей личности. В статье описаны особенности разработки настольных игр по математике, а также представлен обзор авторской системы игр для автоматизации математических навыков. [1]. В статье рассмотрела функции учебных игр на занятиях по РКИ, модульные

игры как особый вид учебных игровых заданий. Дала примеры вариантов учебных игр на основе «Мемори» для обучающихся РКИ на начальном этапе Н.А. Власова [2]. Е.Д. Макарова, В.И. Титова рассмотрели вопрос включения игр в урок математики. Для обсуждения представлена игра: «Геометрическое мемори», разработанная нами самостоятельно и адаптированная под урок математики [3]. Н.А. Власова рассмотрела модульные учебные игры, приводится пример игры «Неприкосновенный запас» (на основе «Мемори»), направленной на отработку прилагательных, с описанием целей игры, правил ее проведения, порядка подведения итогов и видов пост игровой активности [4]. Б.П. Бондаренко рассмотрел методы вычисления теоретических вероятностей возможных исходов в игре «Мемори». В ходе исследований проводится анализ возможных игровых ситуаций и вычисление вероятностей выпадения каждой из них. В итоге был разработан определитель вероятностей возможных исходов в игре «Мемори», что позволило грамотно спроектировать электронную версию игры «Мемори», рассчитанную для двух участников[5].

1.3. Цель исследования

Цель исследования – показать процесс разработки математической игры «Мемори».

2. Материалы и методы

Для реализации необходимы стандартные для веба инструменты: HTML, CSS и JavaScript.

3. Результаты

Создадим новый файл index.html и напомним код страницы:



```
1 <!DOCTYPE html>
2 <html lang="ru">
3 <head>
4 <meta charset="UTF-8">
5 <title>Найди пары</title>
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3/jquery.min.js">
8 <link rel="stylesheet" href="/style.css">
9 </head>
10 <body>
11 <div class="wrap">
12 <div class="timer">Time: 0</div>
13 <div class="game"></div>
14 <div class="modal-overlay">
15 <div class="modal">
16 <div class="winner">Победил</div>
17 <div class="time">Проели: <span class="timer"></span></div>
18 <button class="restart">Сыграть еще</button>
19 </div>
20 </div>
21 </div>
22
23 <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3/jquery.min.js"></script>
24 <script src="/script.js"></script>
25 </body>
26 </html>
```

Рисунок 1 – Код страницы index.html

Пока ничего нет из оформления, поэтому на странице видим только время, текст и кнопку:

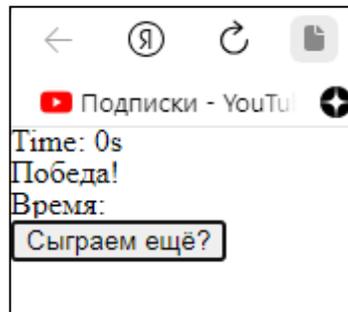


Рисунок 2 – Итог страницы index.html

Создадим файл style.css и добавим туда сначала общие настройки для всей страницы:

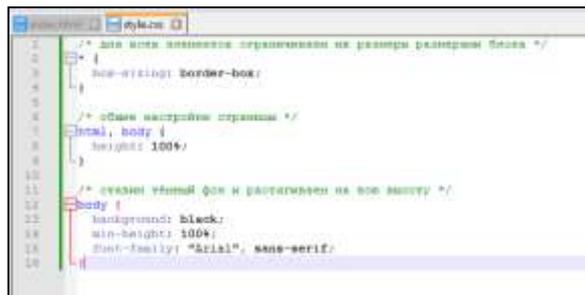


Рисунок 3 – Код страницы style.css

Теперь оформим модальное окно. Добавим этот код в файл со стилями:



Рисунок 4 – Код страницы style.css

```
21 text-align: center;
22 color: #444444;
23 text-decoration: 0px 0px 0 black;
24 }
25
26 /* кнопка кнопки окна победы, кнопки призыва */
27 @media (max-width: 480px) {
28   .modal .button {
29     font-size: 40px;
30   }
31 }
32
33 /* кнопка кнопки переключения игры */
34 @media .button {
35   margin: 20px auto;
36   padding: 20px 30px;
37   display: block;
38   font-size: 30px;
39   border: none;
40   background: #444444;
41   background: linear-gradient(#444444, #222);
42   border: 1px solid #222;
43   border-radius: 5px;
44   color: white;
45   text-decoration: 0px 0px 0 black;
46   cursor: pointer;
47 }
48
49 /* кнопки для при наведении мыши на кнопку */
50 @media .button:hover {
51   background: linear-gradient(#222, black);
52 }
53
54 /* выравниваем кнопки на странице окна по центру */
55 @media .button {
56   text-align: center;
57 }
```

Рисунок 5 – Код страницы style.css

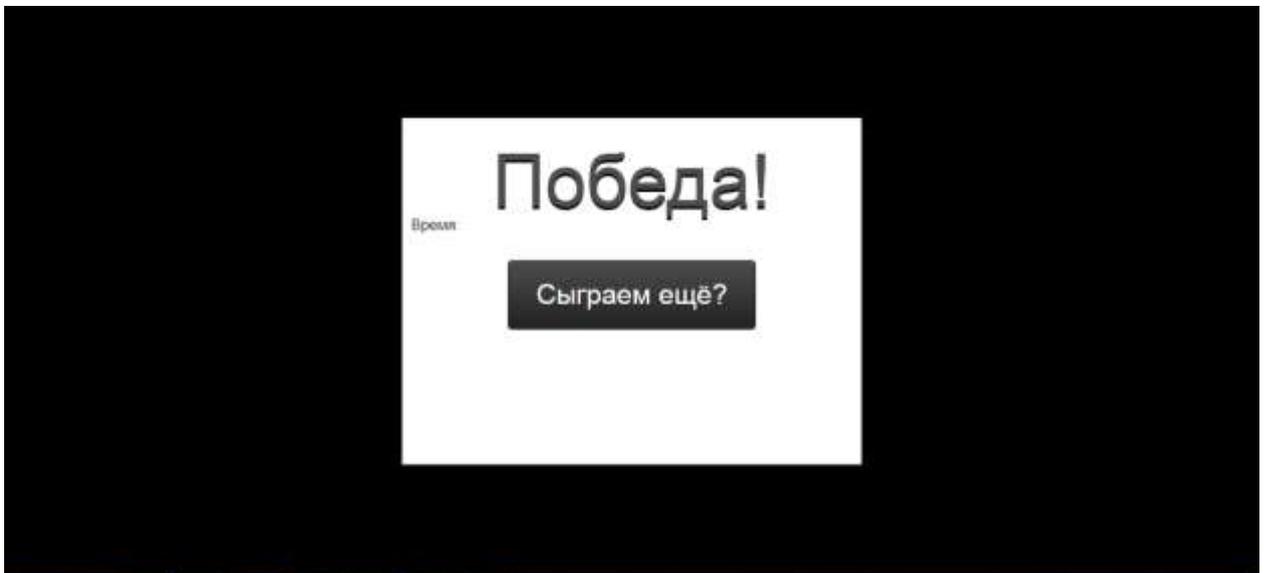


Рисунок 6 – Итог страницы style.css

Теперь скроем окно: добавим `display: none;` в оба первых блока. Это скроет окно, но всё оформление останется – как только окно понадобится, уберём это свойство через скрипт.

Далее понадобится скрипт. Создадим файл `script.js` и добавим в него код.

```

1 // Объявление массива
2 // Массив картинок = []
3 { id: '1', imageUrl: "155a*1155" },
4 { id: '2', imageUrl: "1505a*1000" },
5 { id: '3', imageUrl: "1405a*1500" },
6 { id: '4', imageUrl: "741a*670" },
7 { id: '5', imageUrl: "801a*701" },
8 { id: '6', imageUrl: "2334a*1600" },
9 { id: '7', imageUrl: "140a*270" },
10 { id: '8', imageUrl: "101a*701" },
11 { id: '9', imageUrl: "101a*1101" },
12 { id: '10', imageUrl: "420a*520" },
13 { id: '11', imageUrl: "81a*110" },
14 { id: '12', imageUrl: "55a*150" },
15
16 { id: '17', imageUrl: "a*0" },
17 { id: '18', imageUrl: "a*0" },
18 { id: '19', imageUrl: "a*0" },
19 { id: '20', imageUrl: "a*0" },
20 { id: '21', imageUrl: "a*0" },
21 { id: '22', imageUrl: "a*0" },
22 { id: '23', imageUrl: "a*0" },
23 { id: '24', imageUrl: "a*0" },
24 { id: '25', imageUrl: "a*0" },
25 { id: '26', imageUrl: "a*0" },
26 { id: '27', imageUrl: "a*0" },
27 { id: '28', imageUrl: "a*0" },
28 { id: '29', imageUrl: "a*0" },
29 { id: '30', imageUrl: "a*0" },
30

```

Рисунок 7 – Создание страницы script.js

На экране пока ничего не появится, так как просто написали код на картинке и сложили их в массив. Чтобы сформировать сами карточки, понадобится объект – переменная, внутри которой можно объявлять и вызывать разные методы.

Для «рубашки» карточек возьмём бесплатную фотографию кода из сервиса Unsplash.

```

73
74 // Объявление объекта, внутри которого будем прописывать основные методы игры
75 var Memory = {}
76
77 // создание карточки
78 init: function(card) {
79     // создание объекта в классе
80     this.Spam = {} ("spam");
81     this.Serial = {} ("serial");
82     this.Overlay = {} ("modal-overlay");
83     this.ResetCardButton = {} ("button-reset");
84     // создание из карточки объекта - игровой блок
85     this.cardArray = $.merge(cards, card);
86     // переименование карточек
87     this.shuffleCards(this.cardArray);
88     // и возвращаем их
89     this.setup();
90 }
91
92 // как переименовать карточки
93 shuffleCards: function(cardArray) {
94     // используем встроенный метод .shuffle
95     this.cards = $(this.shuffle(cardArray));
96 }
97
98 // возвращаем карточку
99 setup: function() {
100     // создаём объект card и вставляем на страницу
101     this.html = this.buildHTML();
102     // добавляем код в блок с игрой
103     this.$game.html(this.html);
104     // получаем доступ к сформированным карточкам
105     this.$memoryCards = $(".card");
106     // на экран вы не ждём завершения загрузки карточек
107     this.paused = false;
108     // на экране у нас все переключатель первой карточкой
109     this.$card = null;

```

Рисунок 8 – Создание страницы script.js


```
228     };  
229     // запускаем игру  
230     Memory.init(cards);
```

Рисунок 12 – Создание страницы script.js

Теперь появилось много всякого разного на странице, но выглядит хаотично, а всё потому, что нужно настроить стили для карточек.



Рисунок 13 – Итог страницы script.js

Сейчас разместим карточки и расположим по сетке 6×4 . Для этого подготовим два блока – общий и блок с карточками:

```
81 /* стили основного блока */  
82 var {  
83     /* устанавливаем относительные позиционирование */  
84     position: relative;  
85     /* высота изменена */  
86     height: 100%;  
87     /* минимальная высота и ширина */  
88     min-height: 300px;  
89     padding-bottom: 30px;  
90 }  
91  
92 /* блок с игрой */  
93 var {  
94     /* добавляем прозрачность для эффекта всплывающего */  
95     transition: opacity 3d;  
96     opacity: 0.5;  
97     /* чтобы элементы всегда все доступны на экране */  
98     min-height: 100%;  
99     height: 100%;  
100 }  
101
```

Рисунок 14 – Код страницы style.css

Теперь сделаем сетку – добавим стили именно для карточек, где укажем их высоту и ширину. Добавим медиазапрос, чтобы на небольших экранах карточки тоже выглядели хорошо:

```

110 /* стили карточек */
111 @media (max-width: 1000px) {
112     /* параметры расположения, высоты и ширины карточек */
113     float: left;
114     width: 16.6666%;
115     height: 120px;
116     /* отступы */
117     padding: 5px;
118     /* выравнивание по центру */
119     text-align: center;
120     /* подравниваем блочные элементы и переносим */
121     display: block;
122     margin-bottom: 50px;
123     /* добавляем относительные позиционирования */
124     position: relative;
125     pointer-events: none;
126     z-index: 50;
127 }
128
129 /* настроим размеры карт при максимальной ширине экрана 1200 пикселей */
130 @media (max-width: 1200px) {
131     @each $i in 1..6 {
132         width: 16%;
133         height: 16.666%;
134     }
135 }

```

Рисунок 15 – Код страницы style.css

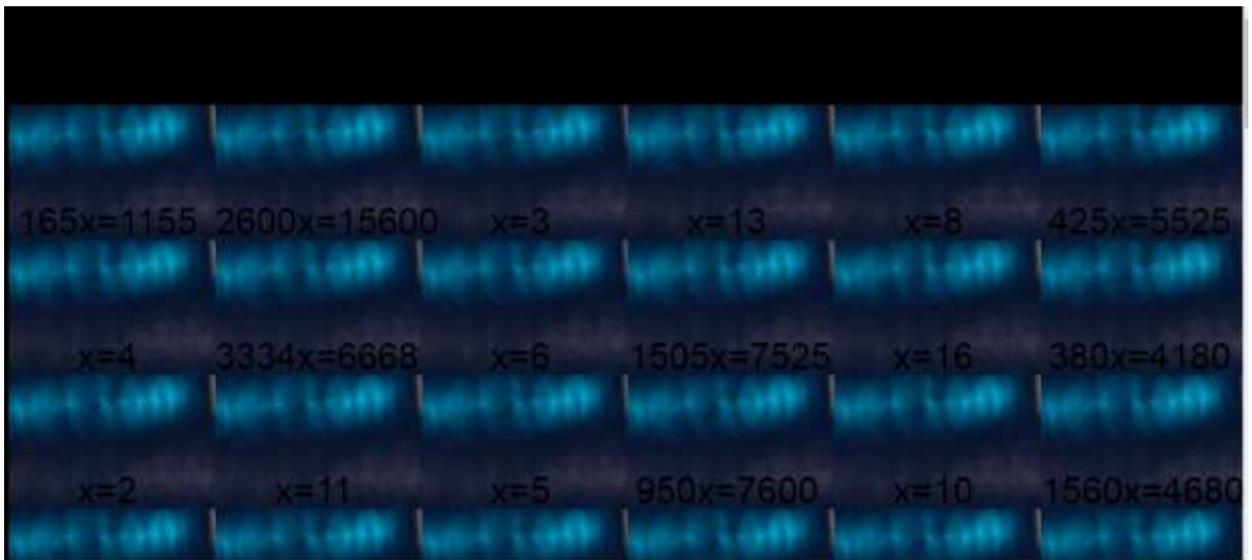


Рисунок 16 – Вид страницы

Появилась сетка, но это всё ещё не похоже на нормальные карточки – всё налезает друг на друга и непонятно, как с этим работать. Чтобы это исправить, нужно разделить карточки на лицевую и обратную стороны, для этого нужно написать код:

```

136 /* обратная сторона карточек */
137 @each $i in 1..6 {
138     /* ширина и высота всей карточки */
139     width: 100%;
140     height: 100%;
141     display: block;
142     /* ширина и высота */
143     transform: rotate(180deg);
144     transform-origin: center;
145     /* у каждой стороны будет белый фон */
146     background: white;
147 }
148
149 /* общие настройки для обеих сторон карточек */
150 @each $i in 1..6 {
151     /* размер шрифта */
152     font-size: 12px;
153     background: 1px solid black;
154     /* скрываем обратную сторону */
155     backface-visibility: hidden;
156     /* скрываем лицевую сторону */
157     backface-visibility: hidden;
158     /* абсолютное позиционирование */
159     position: absolute;
160     top: 0;
161     left: 0;
162     /* размеры и отступы */
163     width: 100%;
164     height: 100%;
165     padding: 20px;
166 }
167
168 /* настроим отображение на лицевой и обратной стороне */
169 @each $i in 1..6 {
170     /* картинка занимает всю ширину */

```

Рисунок 17 – Код страницы style.css

```

1371
1372
1373 /* настройки изображения на основной и игровой экранах */
1374 .main .front img, .main .back img {
1375     /* ширина задается по ширине */
1376     max-width: 100%;
1377     /* изображение как большой элемент, без отступов */
1378     display: block;
1379     margin: 0 auto;
1380     max-height: 100%;
1381 }
1382
1383 /* настройки игровой панели */
1384 .main .front {
1385     /* поворачиваем игровую панель на 90 градусов */
1386     transform: rotateY(-90deg);
1387 }
1388
1389 /* настройки при включенной игре экран 4:3 пикселей */
1390 @media (max-width: 400px) {
1391     .main .front {
1392         padding: 5px;
1393     }
1394     .main .back {
1395         padding: 10px;
1396     }
1397 }
1398
1399 /* кнопки кнопки переключаются при игре на экран */
1400 .main .slide-right, .main .slide-left {
1401     transform: rotateY(180deg);
1402 }
1403

```

Рисунок 18 – Код страницы style.css



Рисунок 19 – Итог игры

В самом конце показывает время, за которое прошли игру.

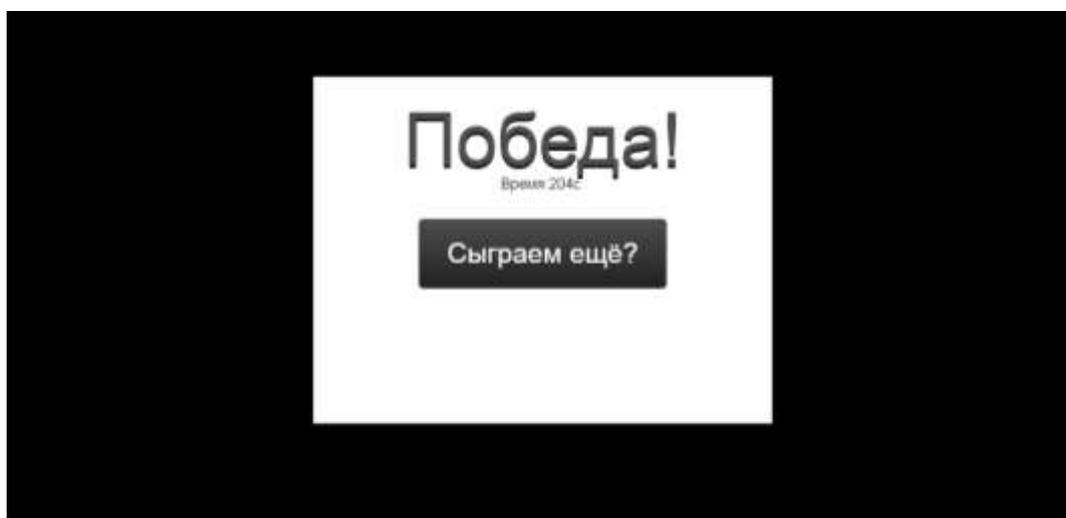


Рисунок 2.20 – Конец игры

4. Выводы

В данной статье был показан процесс разработки образовательной игры «Мемори». Игру можно использовать в учебных целях.

Библиографический список

1. Заблоцкая Л. В. Разработка математической настольной игры по тригонометрии для обучающихся старшего школьного звена //Рекомендовано к печати Ученым советом института педагогики и психологии образования ГАОУ ВО МГПУ Ответственный редактор: АИ Савенков. 2022.
2. Власова Н. А. Модульная игра " Мемори" в обучении русскому языку как иностранному на начальном этапе //На пересечении языков и культур. Актуальные вопросы гуманитарного знания. 2017. №. 1. С. 85-87.
3. Красовский Д. А. Использование игровых форм обучения на уроках математики на примере дидактической игры «домино» //Управление качеством образования: от проектирования к практике. 2018. С. 202-209.
4. Власова Н. А. Использование модульной игры на основе " Мемори" в обучении иностранных студентов лексике русского языка //Актуальные вопросы реализации образовательных программ на подготовительных факультетах для иностранных граждан. 2017. С. 70-75.
5. Бондаренко Б. П. Разработка определителя вероятностей возможных исходов в игре «Мемори» //Современные научные исследования и инновации. 2021. №. 3. С. 10-10.