Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра МКиИТ

Проектирование клиент-серверных приложений

Лабораторная работа №9
“Дальнейшее изучение библиотеки jQuery, добавление эффекта параллакса для главной страницы блога”

Выполнил:
студент 3 курса,
группы БФИ2001
Лушин Е. А.

Москва 2023

Цель работы: научиться разрабатывать эффект параллакса с помощью библиотеки jQuery.

Задание:

Дальнейшее изучение библиотеки jQuery, добавление эффекта параллакса для главной страницы блога

- Настройте также эффект параллакса для картинки-логотипа вашего сервиса. Параметры для скорости перемещения задайте самостоятельно.

Краткая теория

Основные принципы работы с библиотекой jQuery для языка JavaScript

jQuery — библиотека, которая позволяет значительно сократить количество монотонного кода, сделать ваш скрипт более читаемым и менее склонным к ошибкам. jQuery в первую очередь направлена на манипуляцию элементами DOM, так что вряд ли с помощью неё вам удастся выполнить какие-то сложные научные расчеты или какую-то подобную сложную вычислительную работу. Так что обязательно запомните — jQuery лишь помогает работать с DOM, почти все остальные задачи, возложенные на JavaScript, вам придется решать другими путями, хотя тут есть редкие исключения, например создание запросов на сервер. В jQuery есть метод ajax(), который позволяет отправлять и получать данные с сервера без перезагрузки страницы.

Банальное описание принципов работы jQuery выглядит примерно так:

1. Подключение файла библиотеки, например через тег <script>, после которого в вашей программе появляется глобальная переменная с коротким названием «$» — да, название этого объекта — это всего лишь знак доллара. Вот и первое сокращение ненужного кода, не нужно писать длинного имени переменной document для манипуляций элементами.

2. Затем в любом месте программы вы можете вызвать объект «$» как самую обычную функцию, передав ей в качестве параметра CSS-селектор. Например, $(‘div.post′) вернет список всех div-элементов, у которых есть класс «post». Вызов такой функции всегда возвращает специальный объект JavaScript, содержащий массив элементов DOM, соответствующий селектору. У этого специального объекта есть много методов, которые собственно и позволяют различным образом манипулировать элементами.
3. У объекта, который вернул вызов функции $() вызвать любую jQuery функцию. Например, функция addClass(‘‘new-class′′) добавит всем элементам в списке новый класс «new-class».
4. Любой выполненный метод ещё раз возвращает специальный объект jQuery, так что можно продолжить работу с тем же списком элементов и таким образом создать цепочку манипуляций над элементами: $(‘div.post′). addClass(‘‘new-class′′).hide().remove() — сначала добавить класс элементам, потом их скрыть, а затем и вовсе удалить.

Из-за того что только объекты jQuery обладают всеми этими широкими возможностями, обычные элементы DOM нужно сначала «обрамить» вызовом функции jQuery, которая вернет тот же список тех же элементов, только уже с нужными возможностями.

Исчезновение элемента с последующим добавлением потомка с помощью jQuery:

$ (e.target).hide().append(‘ hey! ')

В данном случае, внутри переменной e.target хранится уже знакомый вам обычный узел DOM-а, однако, «обрамив» этот узел вызовом jQuery, вы сможете к данному элементу применить все возможности библиотеки.

Также jQuery позволяет гораздо более удобным способом добавлять обработчики на различные события. Это становится еще более актуально, если один и тот же обработчик нужно добавить сразу к нескольким элементам. С помощью данной библиотеки можно всего за одну строку добавить

обработчик какого-либо события сразу всем элементам, которые подходят на заданные селекторы, и совсем отпадает нужда в создании циклов.

Добавление функции-обработчика на событие click сразу для нескольких элементов с помощью jQuery:

```
$ (`.one-post').click(function(e){
    console.log(`you clicked me');
})
```

Помимо приведенных выше методов, jQuery предоставляет богатейший набор возможностей, список которых вы можете увидеть на официальном сайте проекта http://jquery.com/.

Выполнение

Скопируем предыдущий проект (лабораторная работа №8).

Для того чтобы разместить картинки одна под другой с эффектом смещения и пересечения добавим следующие стили.

Листинг 1. Содержимое файла article.css

```
body {
    background: #1abc9c;
    font-family: Tahoma, Arial, sans-serif;
    color: #ffffff
}

img {
    display: block;
    width: 318px;
    margin-left: auto;
    margin-right: auto;
}

.archive {
    width: 960px;
    margin-left: auto;
    margin-right: auto;
}

post-title a {
    color: #ffffff;
}

.article-author {
```

```
    width: 50%;
    float: Left;
}

.article-created-date {
    text-align: right;
}

.article-image {
    display: block;
    width: 318px;
    margin-left: 0;
}

.link {
    color: white;
    font-weight: bold;
    position: absolute;
    right: 470px;
    top: 180px;
}

.article-border p {
    text-align: right;
}

.article-text {
    width: 960px;
    text-align: justify;
}

.article-created-data {
    text-align: right;
}

.content {
    text-align: center;
    padding-top: 70px;
}

input[name="title"] {
    padding: 5px;
    margin-bottom: 10px;
    border: 1px solid #888;
    outline: none;
    -moz-appearance: none;
    width: 200px;
    text-align: center;
    border-radius: 40px;
}
```

```css
textarea[name="text"] {
    padding: 25px;
    margin-bottom: 10px;
    border: 1px solid #888;
    outline: none;
    -moz-appearance: none;
    width: 650px;
    height: 350px;
    resize: none;
    border-radius: 40px;
    scrollbar-width: thin;
}

.create {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.save_button {
    padding: 10px;
    width: 150px;
    background-color: white;
    border: none;
    border-radius: 40px;
    color: #1abc9c;
    font-weight: bold;
    letter-spacing: 0.06em;
    margin-top: 10px;
}

.save_button:hover {
    color: white;
    background-color: #1abc9c;
    box-shadow: 1px 1px 10px 10px;
    transition-duration: 0.3s;
}

.one-post {
    position: relative;
    /* Добавьте эту строку, чтобы создать контекст для z-index */
}

.one-post-shadow {
    position: absolute;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    background: black;
    z-index: -1;
```

```
    /* это свойство «отодвигает» элемент на задний план */
}

.icons-for-parallax {
    margin-top: 50px;
    height: 335px;
    position: relative;
}

.icons-for-parallax img {
    /* здесь находятся общие для всех картинок стили */
    position: absolute;
    top: 0;
}

.icon-for-parallax-first {
    width: 200px;
    margin-top: 120px;
    z-index: 3;
    /* эта картинка на переднем плане, у неё больший z-index*/
}

.icon-for-parallax-second {
    width: 175px;
    /* уменьшить ширину для картинки, чтобы визуально */
    /* находилась дальше для пользователя */
    margin-top: 60px;
    /* эта картинка слегка смещена вправо и вниз */
    left: 150px;
    /* за счет левой и верхней границ */
    z-index: 2;
}

.icon-for-parallax-third {
    width: 150px;
    left: 260px;
    z-index: 1;
    /* эта картинка на заднем плане, у неё меньший z-index */
}
```

После внесения всех изменений в верстку, добавим в папку articles/static/js/ новый файл parallax.js. Именно в этом файле будет содержаться исходный код скрипта, который изменяет координаты изображений при прокрутке страницы.

Для выполнения всех шагов начнем со стандартной для каждого файла, где подключен jQuery, операции: ожидания события ready для документа.

Листинг 2. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ...
1   $(document).ready(function () {
2       // Код будет здесь
3   });
```

Теперь можно быть уверенным, что код запустится только после окончательно загрузки DOM. Для шагов инициализации, описанных выше, добавим в свой файл следующий исходный код:

Листинг 3. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ...
1   $(document).ready(function () {
2       var yPosition;
3       var scrolled = 0;
4       var $parallaxElements = $('.icons-for-parallax img');
5
6   });
```

Чтобы можно было при каждой прокрутке выполнять какие-либо операции, на событие scroll объекта window добавим функцию-обработчик:

Листинг 4. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ...
1   $(document).ready(function () {
2       var yPosition;
3       var scrolled = 0;
4       var $parallaxElements = $('.icons-for-parallax img');
5       $(window).scroll(function () {
6           // код для изменения при прокрутке страницы
7       });
8
9   });
```

Внутри этой функции сначала следует узнать количество прокрученных пикселей, для чего используем следующую функцию так, как показано ниже.

Листинг 5. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ready() callback > scroll() callback
1   $(document).ready(function () {
2       var yPosition;
3       var scrolled = 0;
4       var $parallaxElements = $('.icons-for-parallax img');
```

```
    $(window).scroll(function () {
        // код для изменения при прокрутке страницы
        scrolled = $(window).scrollTop();
    });

});
```

Естественно, эту операцию нужно выполнять внутри функции-обработчика события scroll. Затем там же, в этой функции запустим цикл по всем элементам внутри переменной $parallaxElements.

Листинг 6. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ready() callback > scroll() callback
$(document).ready(function () {
    var yPosition;
    var scrolled = 0;
    var $parallaxElements = $('.icons-for-parallax img');
    $(window).scroll(function () {
        // код для изменения при прокрутке страницы
        scrolled = $(window).scrollTop();
        for (var i = 0; i < $parallaxElements.length; i++) {
            // Изменение каждого изображения
        }

    });

});
```

Внутри цикла сначала посчитаем на сколько пикселей вам нужно визуально опустить текущий элемент.

Листинг 7. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ready() callback > scroll() callback
$(document).ready(function () {
    var yPosition;
    var scrolled = 0;
    var $parallaxElements = $('.icons-for-parallax img');
    $(window).scroll(function () {
        // код для изменения при прокрутке страницы
        scrolled = $(window).scrollTop();
        for (var i = 0; i < $parallaxElements.length; i++) {
            // Изменение каждого изображения
            yPosition = (scrolled * 0.15 * (i + 1));
        }

    });

});
```

Осталось только установить уже подсчитанные координаты для текущего элемента:

Листинг 8. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ...
1   $(document).ready(function () {
2       var yPosition;
3       var scrolled = 0;
4       var $parallaxElements = $('.icons-for-parallax img');
5       $(window).scroll(function () {
6           // код для изменения при прокрутке страницы
7           scrolled = $(window).scrollTop();
8           for (var i = 0; i < $parallaxElements.length; i++) {
9               // Изменение каждого изображения
10              yPosition = (scrolled * 0.15 * (i + 1));
11              $parallaxElements.eq(i).css({top: yPosition});
12          }
13
14      });
15
16  });
```

Метод eq(), позволяет из всего списка элементов, который хранится в jQuery-переменной, выбрать именно тот элемент, позиция которого совпадает с переданным методу числом. В данном случае в цикле последовательно из списка берём 0-ой элемент, затем 1-ый, и, наконец, 2-ой. Для каждого из них устанавливаем координату, которая напрямую зависит от номера позиции, ведь в операции подсчета пикселей также фигурирует переменная i.

В конце изменим размер header для визуального вмещения в него картинок при прокрутке.

В конечном итоге код нашего скрипта имеет следующий вид:

Листинг 9. Содержимое файла parallax.css

```
blog > articles > static > js > JS parallax.js > ready() callback
1   $(document).ready(function () {
2       var yPosition;
3       var scrolled = 0;
4       var $parallaxElements = $('.icons-for-parallax img');
5       var $header = $('.header')
6
```

```
7     $(window).scroll(function () {
8         scrolled = $(window).scrollTop();
9         for (var i = 0; i < $parallaxElements.length; i++) {
10            yPosition = (scrolled * 0.15 * (i + 1));
11            $parallaxElements.eq(i).css({top: yPosition});
12        }
13    });
14
15    $header.height(initHeight + scrolled * 0.32)
16
17 });
```

Результат работы параллакса представлен на рисунках 1 и 2 на страницах 11 и 12 соответственно.

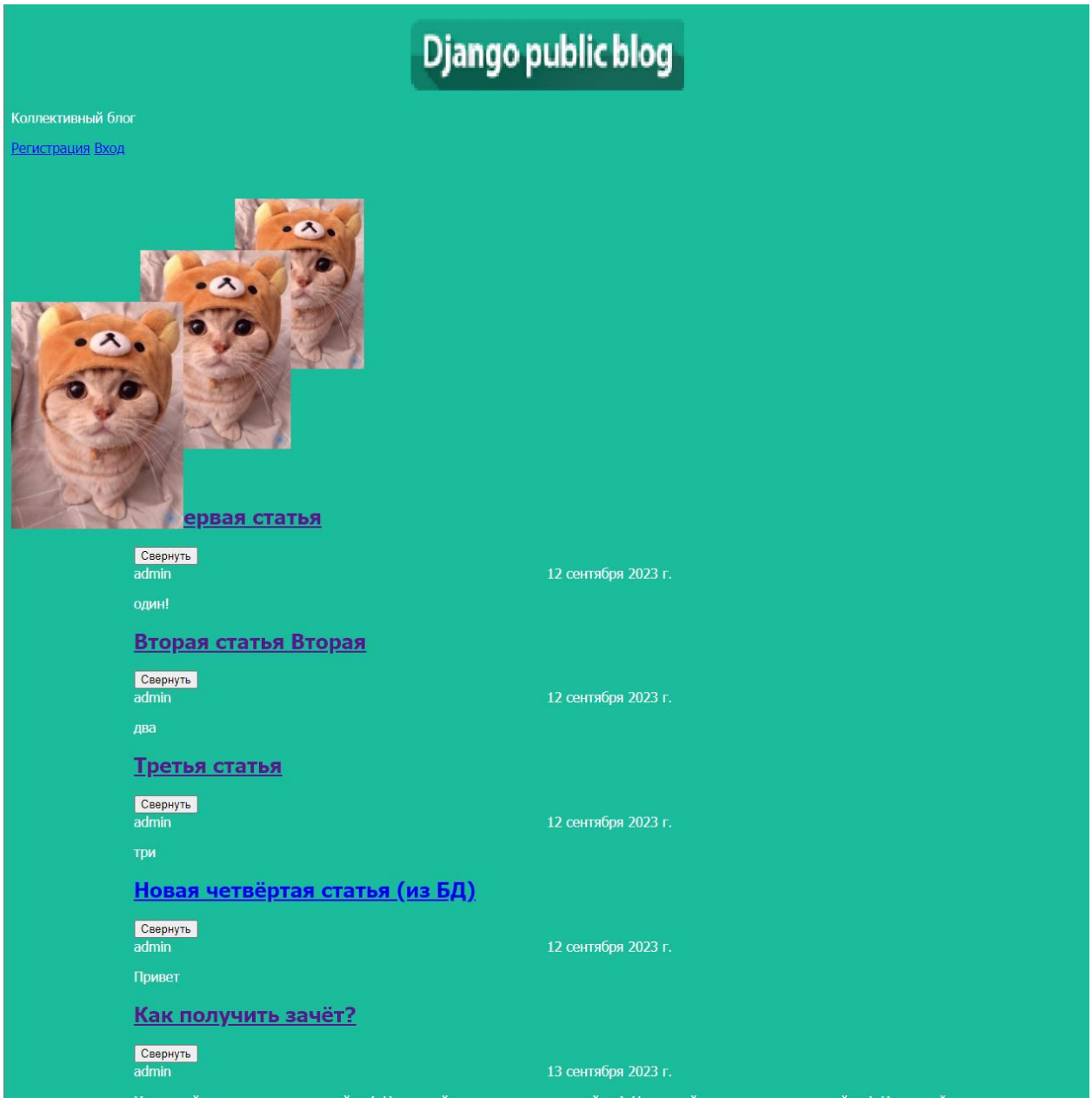

Рисунок 1 – Параллакс изображений (Начальное состояние)

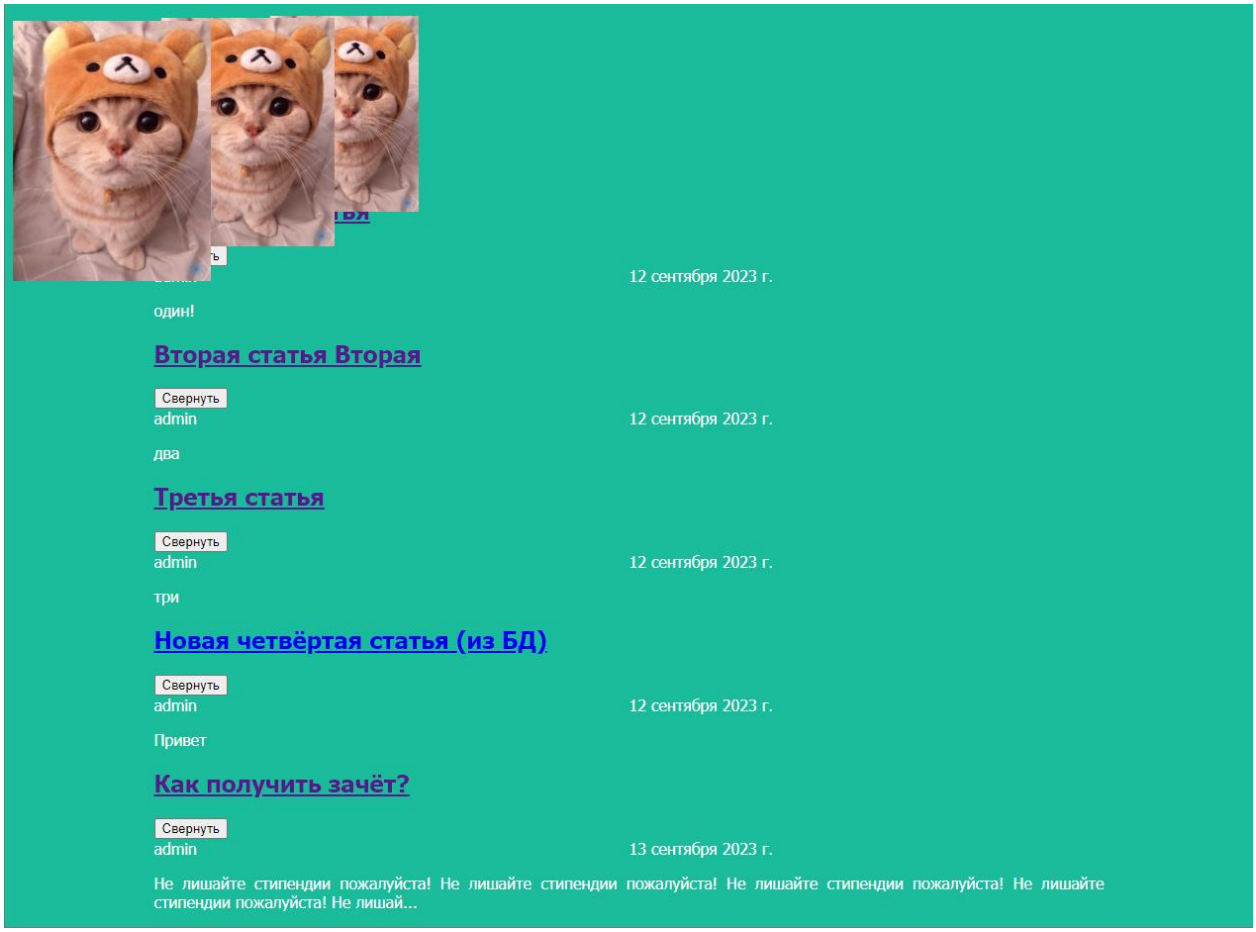

Рисунок 2 – Параллакс изображений (Произведена небольшая прокрутка страницы)

Аналогично разработаем эффект параллакса для логотипа. Для этого добавим блок «logo» и еще одно изображение логотипа.

Листинг 10. Содержимое файла archive.html

blog > articles > templates > archive.html > html

```
{% load static %}

<!DOCTYPE html>
<html>

<head>
    <title>Архив всех статей</title>
    <link rel="stylesheet" type="text/css" href="{% static 'css/article.css' %}">

</head>

<body>
    <div class="header">
        <img class="logo" src="{% static 'img/dpb-logo.png' %}" alt="" type="image/png" />
        <div class="logo">
            <img src="{% static 'img/django-logo.png' %}" alt="" type="image/png" />
            <img src="{% static 'img/python-logo.png' %}" alt="" type="image/png" />
        </div>
        <div class="header-text-area"></div>
        <p class="service-title">Коллективный блог</p>
        <a href="/register/" class="register-link">Регистрация</a>
        <a href="/login/" class="register-link">Вход</a>
    </div>
```

```
    <div class="icons-for-parallax">
        <img class="icon-for-parallax-first" src="{% static 'img/article-icon-for-parallax.png' %}" alt="" type="image/png" />
        <img class="icon-for-parallax-second" src="{% static 'img/article-icon-for-parallax.png' %}" alt="" type="image/png" />
        <img class="icon-for-parallax-third" src="{% static 'img/article-icon-for-parallax.png' %}" alt="" type="image/png" />
    </div>
    <div class="archive">
        {% for post in posts %}
        <div class="one-post">
            <div class="article-title-wrapper">
                <h2 class="post-title">
                    <a href="http://127.0.0.1:8000/article/{{post.id}}">{{post.title}}</a>
                </h2>
                <button class="fold-button">Свернуть</button>
            </div>
            <div class="article-info">
                <div class="article-author">{{ post.author.username }}</div>
                <div class="article-createddate">{{post.created_date}}</div>
            </div>
            <p class="article-text">{{ post.get_excerpt }}</p>
            <div class="one-post-shadow"></div>
        </div>
        {% endfor %}
    </div>
    <script src="{% static 'js/helloworld.js' %}"></script>
    <script src="{% static 'js/fold-post.js' %}"></script>
    <script src="{% static 'js/jquery.js' %}"></script>
    <script src="{% static 'js/highlight-post.js' %}"></script>
    <script src="{% static 'js/parallax.js' %}"></script>
</body>

</html>
```

Добавим стили для двух изображений логотипа.

Листинг 11. Содержимое файла article.css

```
body {
    background: #1abc9c;
    font-family: Tahoma, Arial, sans-serif;
    color: #ffffff
}

img {
    display: block;
    width: 318px;
    margin-left: auto;
    margin-right: auto;
}

.archive {
    width: 960px;
    margin-left: auto;
    margin-right: auto;
}

post-title a {
    color: #ffffff;
}

.article-author {
    width: 50%;
    float: Left;
}
```

```css
.article-created-date {
    text-align: right;
}

.article-image {
    display: block;
    width: 318px;
    margin-left: 0;
}

.link {
    color: white;
    font-weight: bold;
    position: absolute;
    right: 470px;
    top: 180px;
}

.article-border p {
    text-align: right;
}

.article-text {
    width: 960px;
    text-align: justify;
}

.article-created-data {
    text-align: right;
}

.content {
    text-align: center;
    padding-top: 70px;
}

input[name="title"] {
    padding: 5px;
    margin-bottom: 10px;
    border: 1px solid #888;
    outline: none;
    -moz-appearance: none;
    width: 200px;
    text-align: center;
    border-radius: 40px;
}

textarea[name="text"] {
    padding: 25px;
    margin-bottom: 10px;
    border: 1px solid #888;
```

```
    outline: none;
    -moz-appearance: none;
    width: 650px;
    height: 350px;
    resize: none;
    border-radius: 40px;
    scrollbar-width: thin;
}

.create {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.save_button {
    padding: 10px;
    width: 150px;
    background-color: white;
    border: none;
    border-radius: 40px;
    color: #1abc9c;
    font-weight: bold;
    letter-spacing: 0.06em;
    margin-top: 10px;
}

.save_button:hover {
    color: white;
    background-color: #1abc9c;
    box-shadow: 1px 1px 10px 10px;
    transition-duration: 0.3s;
}

.one-post {
    position: relative;
    /* Добавьте эту строку, чтобы создать контекст для z-index */
}

.one-post-shadow {
    position: absolute;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    background: black;
    z-index: -1;
    /* это свойство «отодвигает» элемент на задний план */
}

.icons-for-parallax {
```

```
    margin-top: 50px;
    height: 335px;
    position: relative;
}

.icons-for-parallax img {
    /* здесь находятся общие для всех картинок стили */
    position: absolute;
    top: 0;
}

.icon-for-parallax-first {
    width: 200px;
    margin-top: 120px;
    z-index: 3;
    /* эта картинка на переднем плане, у неё больший z-index*/
}

.icon-for-parallax-second {
    width: 175px;
    /* уменьшить ширину для картинки, чтобы визуально */
    /* находилась дальше для пользователя */
    margin-top: 60px;
    /* эта картинка слегка смещена вправо и вниз */
    left: 150px;
    /* за счет левой и верхней границ */
    z-index: 2;
}

.icon-for-parallax-third {
    width: 150px;
    left: 260px;
    z-index: 1;
    /* эта картинка на заднем плане, у неё меньший z-index */
}

.logo {
    position: relative;
    height: 160px;
}

.logo img {
    position: absolute;
    left: calc(50% - 160px);
}

.logo img:nth-child(2) {
    position: absolute;
    left: calc(50% - 350px);
    margin-top: 50px;
}
```

Добавим в parallax.js изменение положения изображений логотипа при прокрутке страницы.

Листинг 12. Содержимое файла parallax.js

blog > articles > static > js > JS parallax.js > ...

```
$(document).ready(function () {
    var yPosition;
    var scrolled = 0;
    var $parallaxElements = $('.icons-for-parallax img');
    var $logoElements = $('.logo img');
    var $header = $('.header')
    var initHeight = $header.height()

    $(window).scroll(function () {
        scrolled = $(window).scrollTop();
        for (var i = 0; i < $parallaxElements.length; i++) {
            yPosition = (scrolled * 0.15 * (i + 1));
            $parallaxElements.eq(i).css({top: yPosition});
        }

        for (var i = 0; i < $logoElements.length; i++) {
            yPosition = (scrolled * 0.15 * ($logoElements.length - (i * 5 + 1)));
            $logoElements.eq(i).css({top: yPosition});
        }

        $('.logo').css({marginTop: Math.min(scrolled, 30)})

        $header.height(initHeight + scrolled * 0.32)
    });
});
```

Конечный результат работы представлен на рисунках 3 и 4 на странице 18.

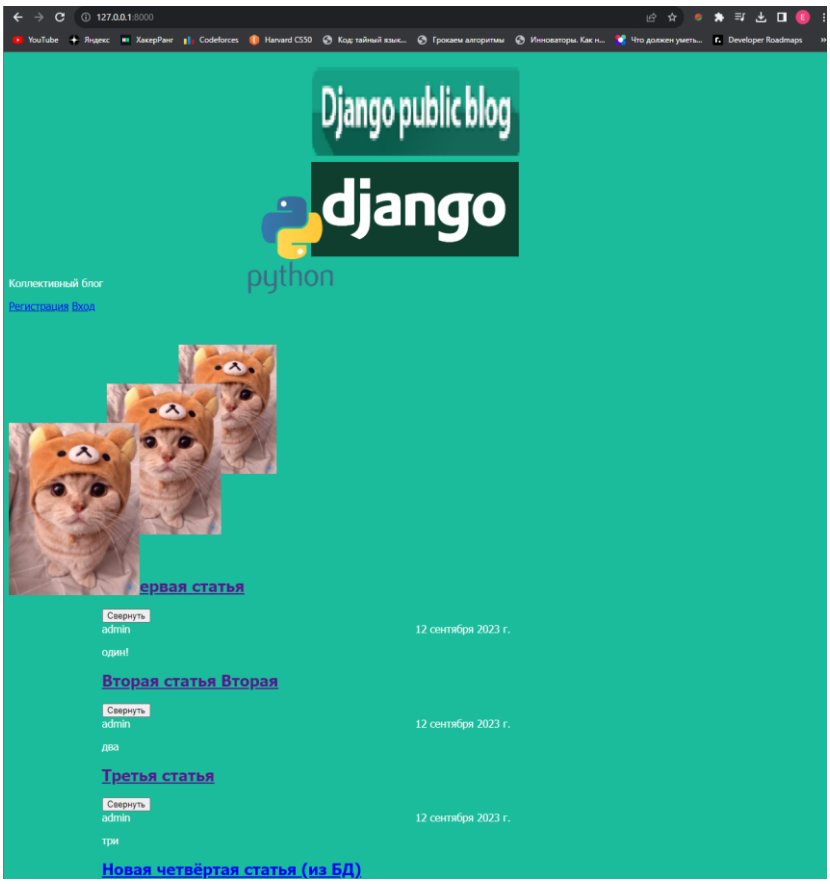

Рисунок 3 – Параллакс изображений логотипа (Начальное состояние)

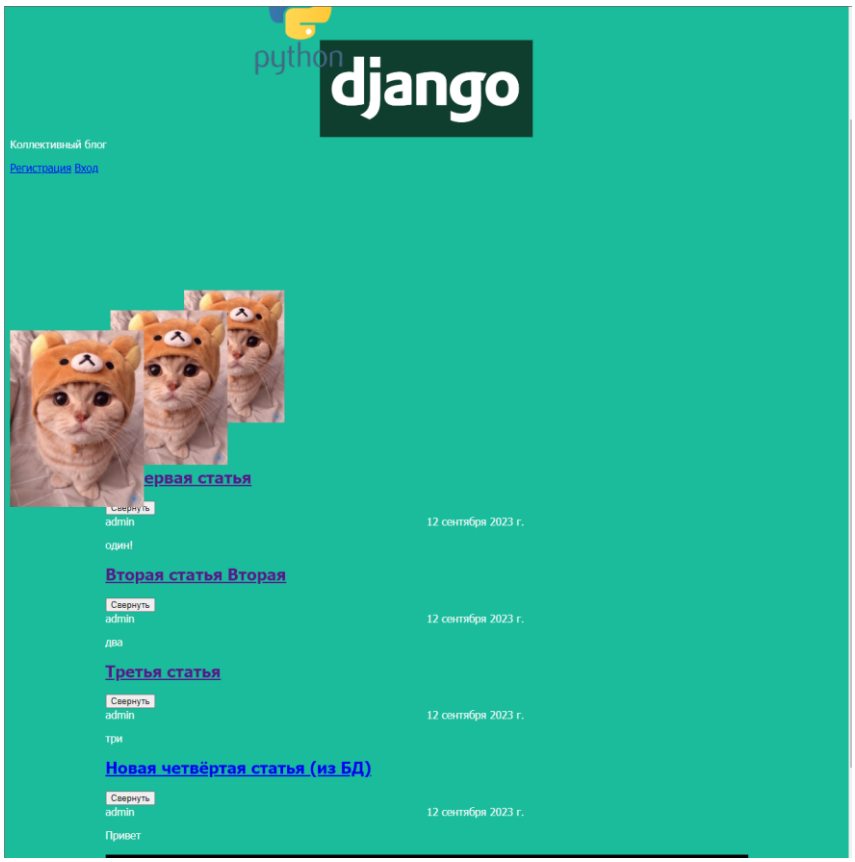

Рисунок 4 – Параллакс изображений логотипа (Произведена небольшая прокрутка страницы)

Вывод: В данной лабораторной работе мы научились научиться разрабатывать эффект параллакса с помощью библиотеки jQuery.