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 №8

“Изучение библиотеки jQuery, добавление подсветки для наведённого поста и эффекта картинки-логотипа”

Выполнил:

студент 3 курса,

группы БФИ2001

Лушин Е. А.

Москва 2023

**Цель работы:** изучить основы работы с библиотекой jQuery.

**Задание:**

Изучение библиотеки jQuery, добавление подсветки для наведённого поста и эффекта для картинки-логотипа

* Сделайте так, чтобы при наведении на картинку-логотип она увеличивалась в размерах (ширина становилась больше на 20px, а высота увеличивалась пропорционально).

**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(‵<p>hey!</p>′)*

В данном случае, внутри переменной e.target хранится уже знакомый вам обычный узел DOM-a, 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. C 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**

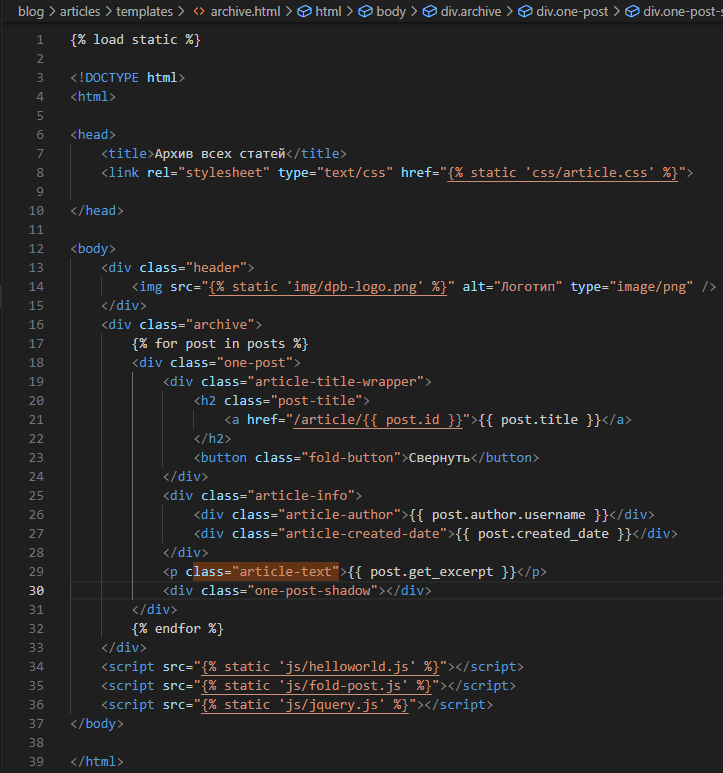
Скопируем предыдущий проект (лабораторная работа №7).

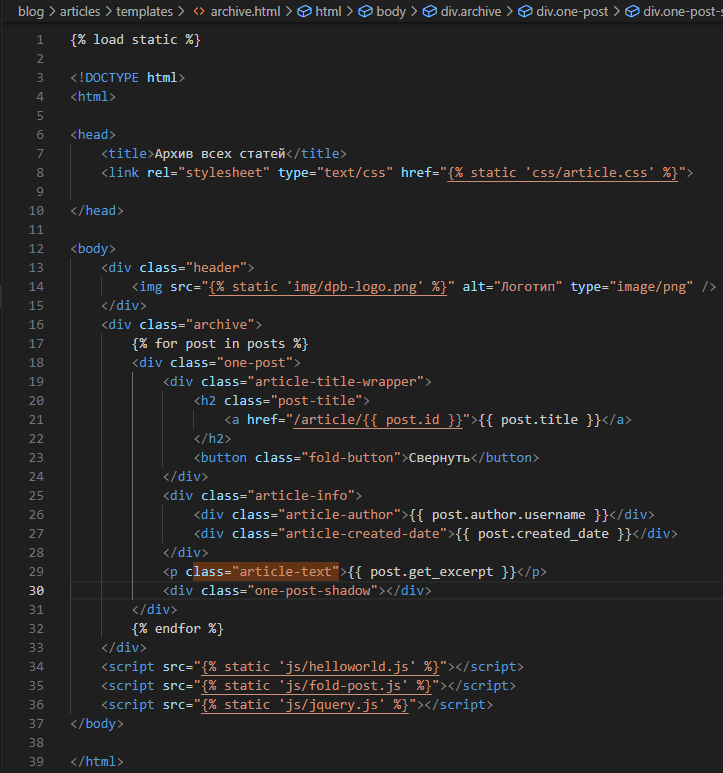
Подключим в наш проект jQuery: **<script src="{% static '/articles/js/jquery-3.6.0.min.js' %}"></script>**

Добавим в проект файл highlight-post.js в директории articles/static/js. Подключим в archive.html и этот файл.

Сначала каждому элементу класса “one-post” добавим в конце разметки блок с классом “one-post-shadow”.

Листинг 1. Содержимое файла archive.html





Теперь этому элементу зададим такой стиль, чтобы блок занимал всю высоту и всю ширину родительского элемента, имел черный цвет фона, и находился на заднем плане от содержимого.

Листинг 2. 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;

}

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-shadow {

    position: absolute;

    top: 0;

    left: 0;

    height: 100%;

    width: 100%;

    background: black;

    z-index: -1;

    /\* это свойство «отодвигает» элемент на задний план \*/

}

На рисунке 1 на странице 8 показан результат добавления темного фона.

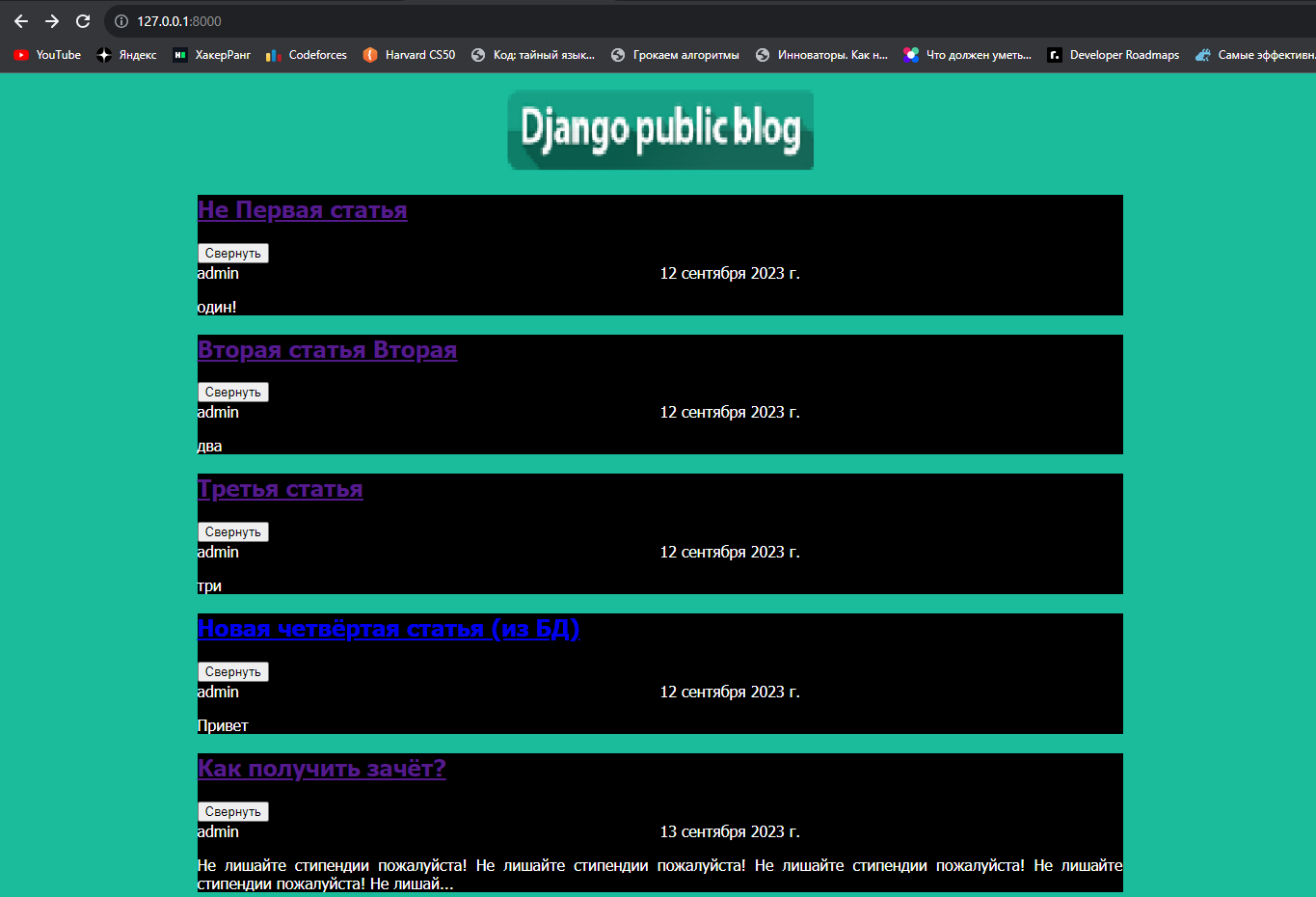
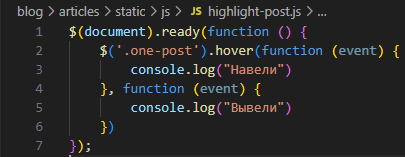


Рисунок 1 – Результат добавления тёмного фона

Добавим код в файл highlight-post.js.

Листинг 3. Содержимое файла highlight-post.js



Попробуем навести курсор на одну из статей. Далее уберем курсор с данного элемента. Как видим события корректно отрабатывают (результат выводится в консоль). Результат работы представлен на рисунке 2 на странице 9.

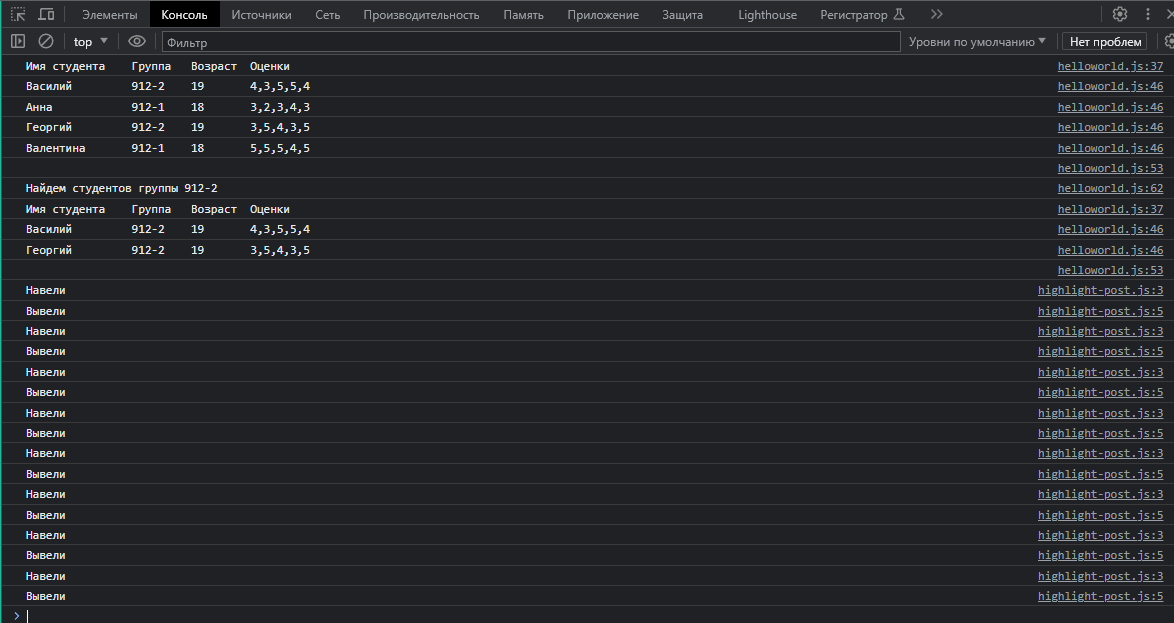
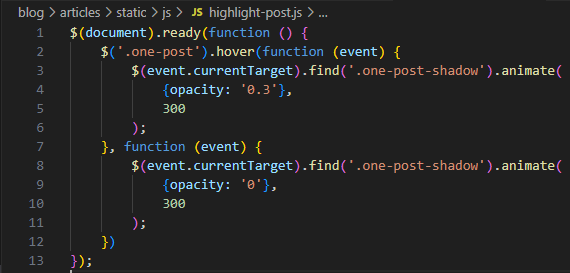


Рисунок 2 – Работа события “hover”

Изменим код данного события. Добавим изменение прозрачности тени при наведении. Также установим значение css-свойства opacity: ‘0’ блоку “one-post-shadow”, чтобы блок тени при первичной загрузке страницы не был виден.

Листинг 4. Содержимое файла highlight-post.js



На рисунке 3 представлен результат работы.

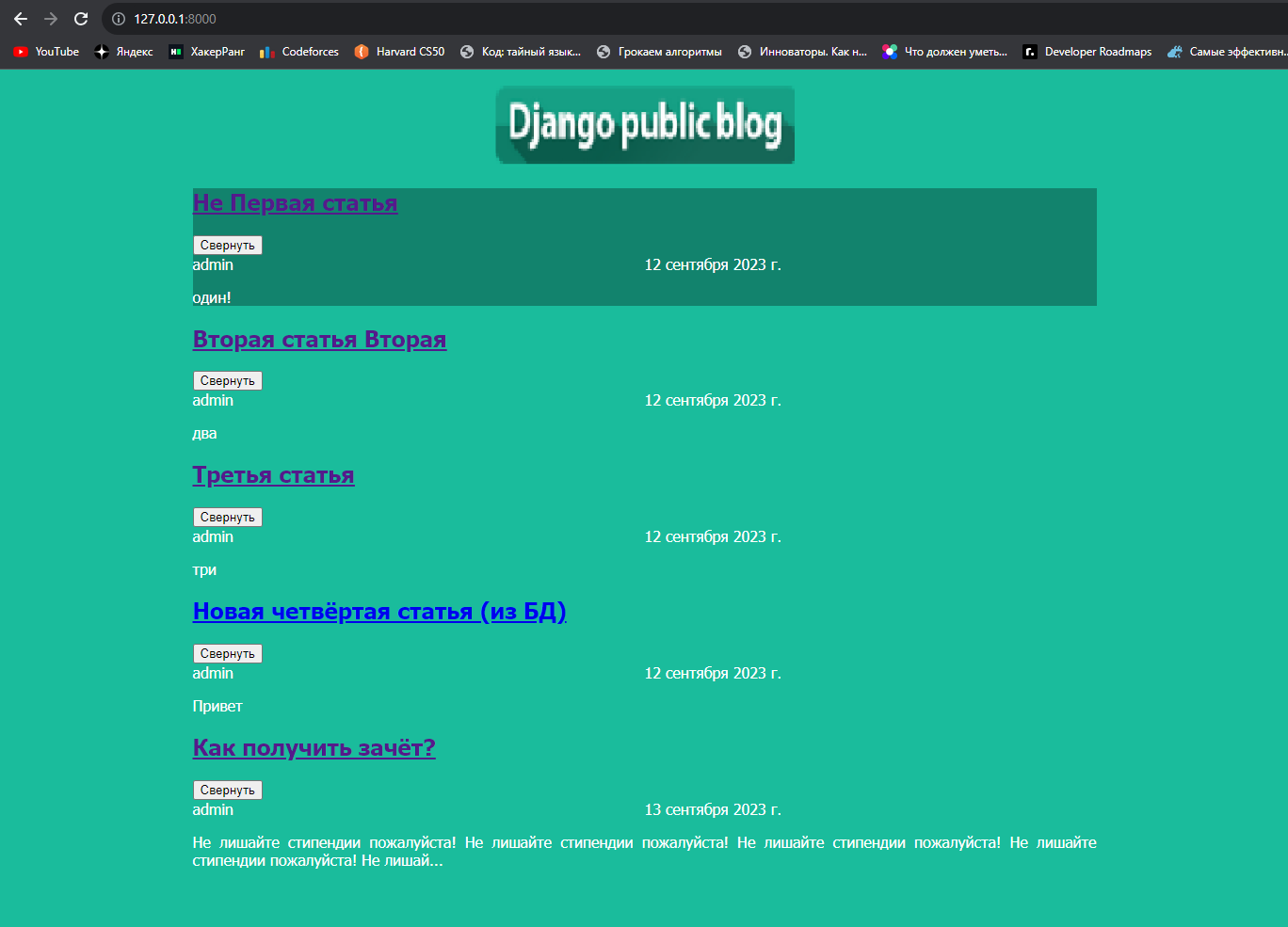


Рисунок 3 – Изменение прозрачности тени при наведении

Сделаем изменение размера картинки логотипа при наведении, используя вышеприведенные события.

Листинг 5. Содержимое файла highlight-post.js



Результат работы при наведении и обычном состоянии представлен на рисунках 4 и 5.



Рисунок 4 – Наведение курсора на логотип

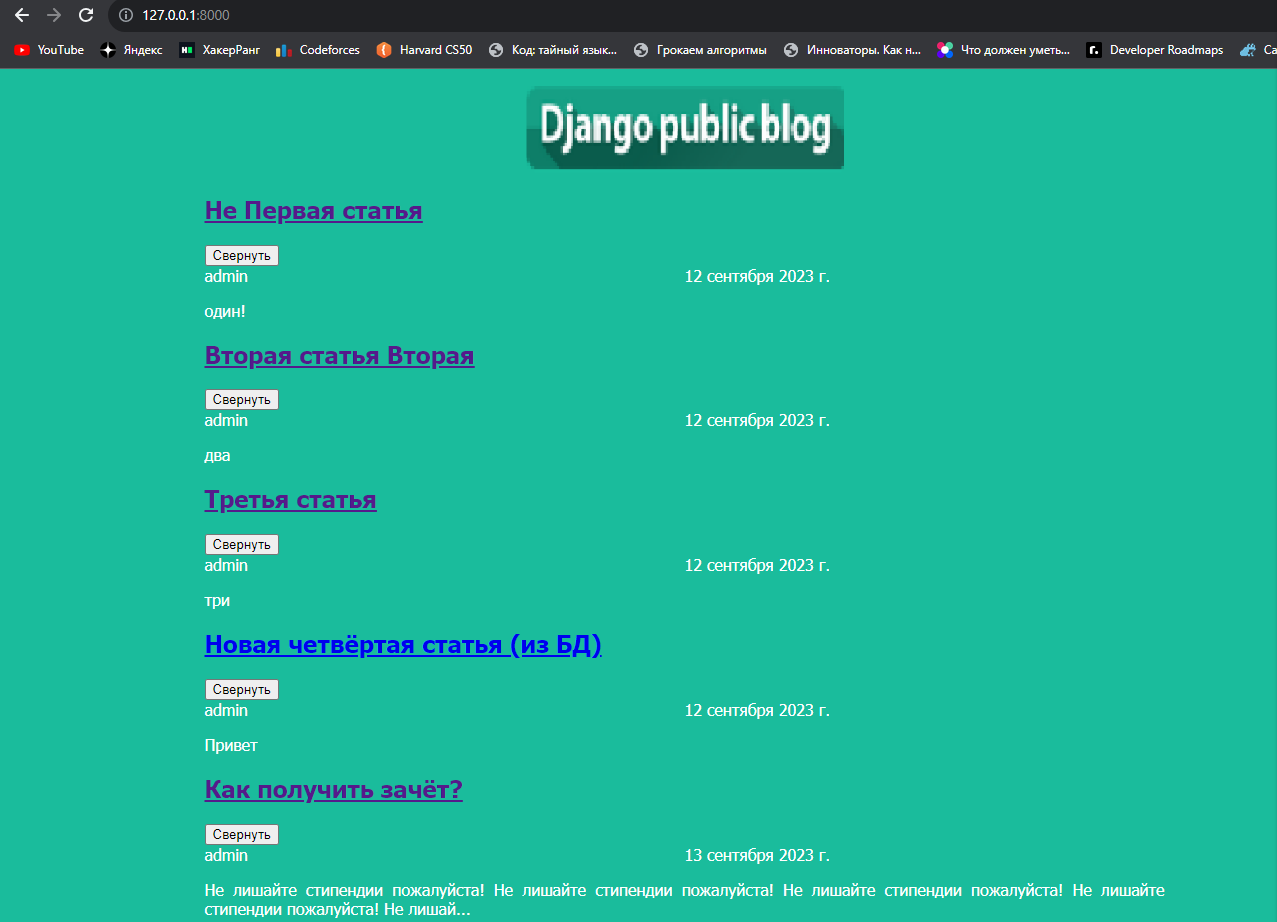


Рисунок 5 – Обычное состояние логотипа

**Вывод** В данной лабораторной работе были изучены основы работы с библиотекой jQuery.