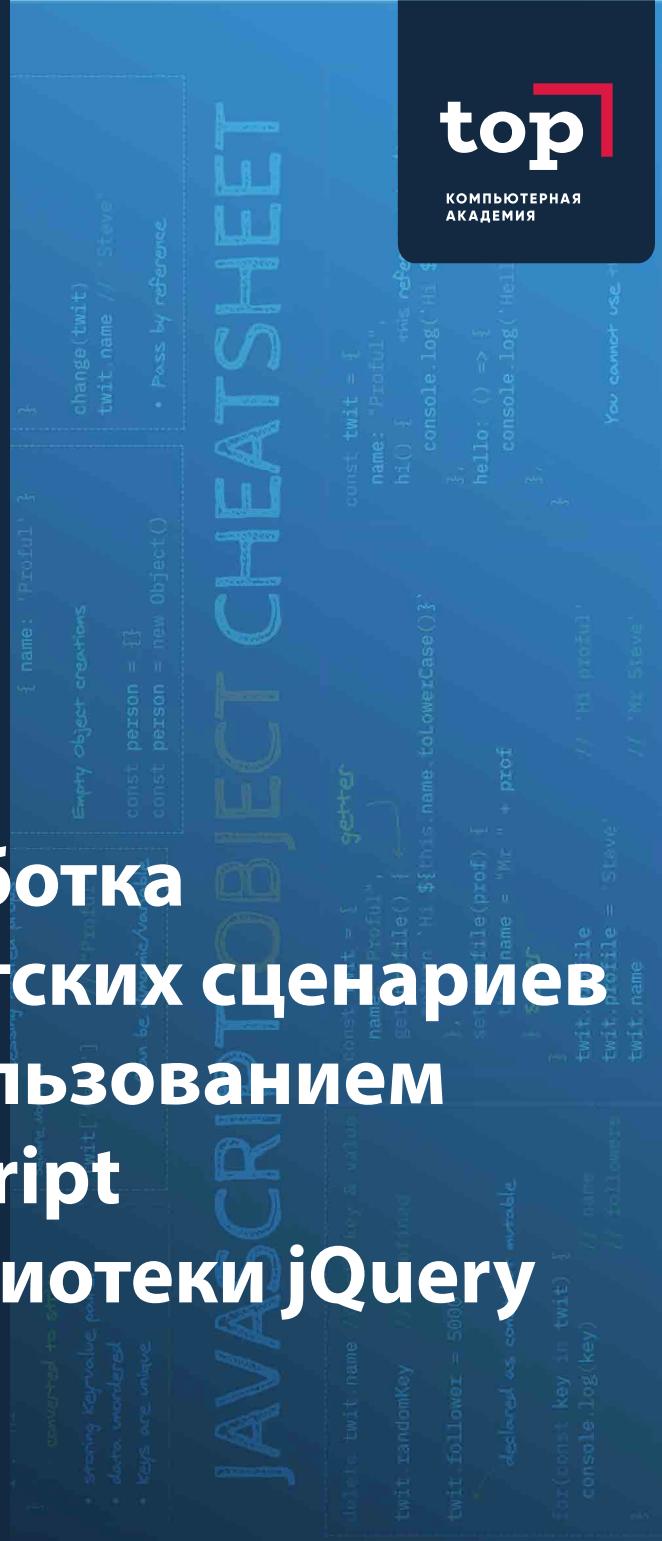


Разработка клиентских сценариев с использованием JavaScript и библиотеки jQuery



Урок № 12

jQuery- плагины

Содержание

jQuery-плагины	3
jQuery и AJAX	10
Обработка ошибок при AJAX-запросе	20
Домашнее задание.....	23
Задание 1	23
Задание 2	24
Задание 3	25
Приложение	27

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе [Adobe Acrobat Reader](#).

jQuery-плагины

Информация о jQuery была бы неполной, если не упомянуть о различных плагинах, которые были написаны сообществом разработчиков и предлагаются для бесплатного скачивания. С одним таким плагином — jquery.maskedinput — мы уже познакомились при обработке элементов форм. Плагины под jQuery решают различные задачи — от построения различных анимированных графиков, создания модальных окон до прокрутки блоков экранами. Больше всего плагинов, пожалуй, создано для фотогалерей и слайдеров. Именно из этой группы мы и будем рассматривать слайдер для формирования отзывов пользователей. Называется он «Owl Carousel» и доступен для скачивания на [официальном сайте](#). Он хорош тем, что позволяет настроить отображение элементов под различные экраны, поддерживает события перетаскивания для различных браузеров (современных и не очень) и весит порядка 50кб. Плюс этот плагин можно не только скачать, но и подключить к вашей странице через CDN, что мы и сделаем.

Общая схема работы с jQuery-плагинами обычно укладывается в такую последовательность:

1. Подключаем в блоке `<head>` стилевой файл плагина с помощью тега `<link>`. Например, для Owl Carousel:

```
<!-- Из локальной папки css -->
<link rel="stylesheet"
      href="css/owl.carousel.min.css">
```

```
<!-- либо с CDN -->
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/
          OwlCarousel2/2.3.4/assets/owl.carousel.css">
```

2. Добавляем определенную разметку в html-файл.

```
<div id="customers-testimonials" class="owl-carousel">
    <div class="item">
        <div class="shadow-effect">
            
            <p>Lorem ipsum dolor sit... animi.</p>
        </div>

        <div class="testimonial-name">
            John Doe
        </div>
    </div>

    <div class="item">
        <div class="shadow-effect">
            
            <p>Quasi sequi ea, aliquid ... repellat.</p>
        </div>

        <div class="testimonial-name">
            Anna Traube
        </div>
    </div>
    ...
</div>
```

3. В нижней части <body> подключаем js-файл плагина после jQuery:

```
<!-- Сначала подключаем js-файл с jQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/
    jquery/3.5.1/jquery.min.js">
</script>

<!-- owl carousel из локальной папки js -->
<script src="js/owl.carousel.min.js"></script>

<!-- либо с CDN -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/
    OwlCarousel2/2.3.4/owl.carousel.min.js">
</script>
```

4. Подключаем скрипт js-файла, в котором будет вызвана основная функция плагина:

```
<script src="js/index.js"></script>
```

5. Вызываем функцию плагина с настройками:

```
$('#customers-testimonials').owlCarousel({
  loop: true,
  center: true,
  items: 3,
  margin: 0,
  // dots: true,
  // autoplay: true,
  // autoplayTimeout: 8500,
  smartSpeed: 450,
  responsive: {
    0: {
      items: 1
    },
  }
});
```

```
    800: {
      items: 3
    }
  );
});
```

Настройки Owl Carousel вы можете найти в [документации на официальном сайте](#). Здесь приведем описание тех настроек, которые были использованы в примере:

- **loop** — зацикливание карусели. Происходит дублирование первого и последнего элементов, чтобы получить иллюзию постоянно прокручиваемой карусели;
- **center** — центрирование элементов. Опция хорошо работает как с четным, так и с нечетным количеством элементов;
- **items** — количество элементов, отображаемых в карусели;
- **margin** — свойство `margin-right` (в `px`) для каждого элемента;
- **dots: true** — показывает навигацию с помощью точек внизу карусели (значение по умолчанию);
- **autoplay: true** — позволяет прокручивать карусель автоматически через определенные промежутки времени. В примере эта опция закомментирована, но вы можете удалить комментарий и посмотреть, как изменится работа карусели.
- **autoplayTimeout: 8500** — время в миллисекундах, через которое происходит автоматическая смена (прокручи-

вание) слайдов в карусели. Раскомментируйте опцию в примере, если хотите, чтобы карусель прокручивалась автоматически через 8,5 секунд.

- **smartSpeed** — скорость смены слайдов в миллисекундах при клике навигационные точки.
- **responsive** — возможность задать гибкие (адаптивные) настройки для различных разрешений экрана в виде объекта, для которого разработчики выдали [такие рекомендации](#):
 - ▷ Каждый ключ точки останова может быть числовым значением (480) или строкой: '480'.
 - ▷ Owl имеет встроенную опцию сортировки, но лучше всего установить от самых маленьких экранов до самых широких.
 - ▷ Адаптивные параметры всегда перезаписывают настройки верхнего уровня.
 - ▷ По умолчанию для параметра **responsive** установлено значение **true**, поэтому карусель всегда пытается соответствовать оболочке (даже если медиа-запросы не поддерживают IE7 / IE8 и т. д.).
 - ▷ Если у вас негибкий макет, установите **responsive: false**.

В нашем примере мы изменяем для этой настройки количество отображаемых слайдов с 3 для больших экранов до 1 для экранов планшетов и смартфонов. Остальные настройки вы можете разобрать самостоятельно.

Имейте в виду, что для оформления карусели был добавлен css-код в блоке с комментарием /*-- Testimonials --*/:

```
.shadow-effect {  
    background: #fff;  
    padding: 20px;  
    border-radius: 4px;  
    text-align: center;  
    border: 1px solid #d6d6d6;  
    box-shadow: 0 19px 38px rgba(0, 0, 0, 0.10),  
                0 15px 12px rgba(0, 0, 0, 0.02);  
}  
  
.shadow-effect p {  
    font-size: 14px;  
    line-height: 1.5;  
    margin: 0 0 17px 0;  
    font-weight: 300;  
}  
  
.testimonial-img {  
    box-shadow: 5px 5px 0 #a6ce39;  
    max-width: 100px;  
    margin: 0 auto 17px;  
}  
  
.testimonial-name {  
    margin: -20px auto 0;  
    display: table;  
    width: auto;  
    background: #a6ce39;  
    padding: 12px 35px;  
    text-align: center;  
    color: #fff;  
    box-shadow: 0 9px 18px rgba(0, 0, 0, 0.12),  
                0 5px 7px rgba(0, 0, 0, 0.05);  
}  
  
#customers-testimonials .item {  
    text-align: center;
```

```
padding: 50px;
margin-bottom: 80px;
opacity: .2;
transform: scale(0.8);
transition: all 0.3s ease-in-out;
}

#customers-testimonials .owl-item.active.center .item
{
    opacity: 1;
    transform: scale(1.10);
}

#customers-testimonials.owl-carousel .owl-dots
    .owl-dot.active span,
#customers-testimonials.owl-carousel .owl-dots
    .owl-dot:hover span {
background: #82a819;
    transform: translateY(-50%) scale(0.7);
}

#customers-testimonials.owl-carousel .owl-dots {
display: inline-block;
width: 100%;
text-align: center;
}

#customers-testimonials.owl-carousel .owl-dots .owl-dot {
display: inline-block;
outline: none;
}

#customers-testimonials.owl-carousel .owl-dots
    .owl-dot span {
background: #a6ce39;
display: inline-block;
height: 20px;
```

```
margin: 0 2px 5px;  
transform: translateY(-50%) scale(0.3);  
transition: all 250ms ease-out;  
width: 20px;  
}
```

jQuery и AJAX

jQuery предоставляет очень простые методы для работы с технологией AJAX. Эта технология предназначена для асинхронной загрузки и передачи данных между сервером и html-страницей посредством JavaScript. То есть данные, которые мы можем отображать на странице, изначально на ней не присутствовали, однако AJAX позволяет их получить из внешнего файла. Либо мы можем отправить данные любой формы и без перезагрузки страницы получить ответ от сервера, например, о том, что на почту пользователю отправлено письмо с запрошенной информацией.

Мы будем загружать с помощью AJAX и jQuery список книг из библиотеки, предоставляющей информацию о книгах в виде открытого API, который к тому же не требует регистрации.

API (*Application Programming Interface*), или программный интерфейс приложения, позволяет пользователям какого-либо сайта получать информацию о продуктах/услугах, представляемых сайтом, или возможность совершать какие-то действия, например, проводить платежи.

Для нашей страницы нужна информация о книгах, которая представлена на сайте <https://openlibrary.org/>

[developers/api](#) в виде различных вариантов: [Books](#), [Covers](#), [Lists](#), [Read](#), [Recent Changes](#), [Search](#), [Search inside](#) и [Subjects](#). Мы используем возможность загружать книги по такому параметру, как Subject (Рубрика — <https://openlibrary.org/dev/docs/api/subjects>), и выберем в качестве рубрики [Mystery and detective stories](#). Все рубрики OpenLibrary вы найдете по адресу <https://openlibrary.org/subjects>.

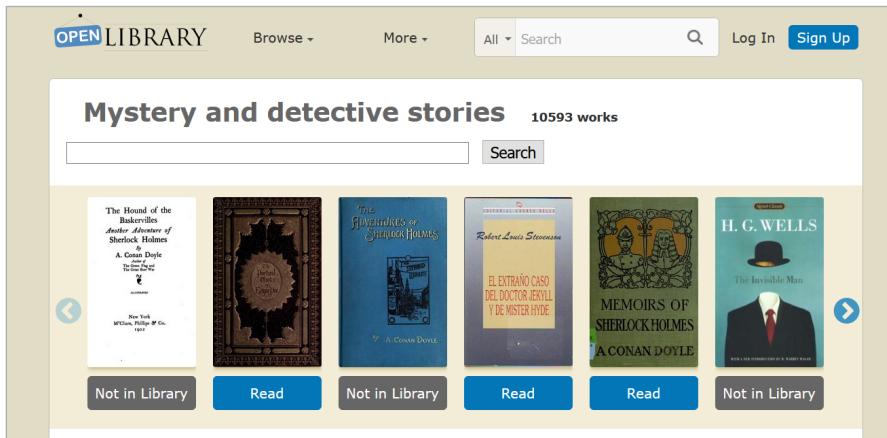


Рисунок 1

API OpenLibrary Subject подразумевает, что вы можете загрузить JSON-файл, в котором будет находиться данные о 12 (или меньшем количестве) книгах в указанной рубрике. Нужный нам этот файл будет находиться по адресу https://openlibrary.org/subjects/mystery_and_detective_stories.json.

Теперь нам нужно разобраться с тем, как с помощью методов jQuery выполнить асинхронную загрузку книг с помощью технологии AJAX. Для этого нам понадобится метод [\\$.ajax\(\)](#), который предполагает такую запись:

```

$.ajax({
    url: 'somefile.ext',
    dataType: "json",
}).done(function(data) {
    // код, который сработает в случае успешной
    // отправки формы на сервер
    console.log("Ok! " + data);
}).fail(function() {
    // код, который будет выводится в случае ошибки
    console.log("Error from mail!!!!" + data);
});

```

В самом методе `$.ajax()` можно указать довольно много опций, но параметр `url` является обязательным, а `dataType: "json"` важен для загрузки данных в соответствующем формате. Поскольку мы будем запрашивать данные методом `GET`, то еще один важный параметр `method` мы не указываем, т. к. по умолчанию он имеет значение '`GET`'. Если же мы будем отправлять данные, то тогда нам понадобится явно задать параметр `method: 'POST'`.

Метод `$.ajax()` посылает запрос на указанный в `url` адрес и получает ответ в виде данных, которые за счет параметра `dataType: "json"` представлены не в виде текста, а в виде объекта или массива объектов. В случае успешной передачи данных возвращается ответ (`response`) и обрабатывается в функции `done()`. Если запрос завершился ошибкой, то обработка ошибки выполняется функцией `fail()`.

Пока что мы поработаем с функцией `done()` и выведем в `console.log()` все данные, полученные с сервера `OpenLibrary` в переменную `data`:

```

var category_url = "https://openlibrary.org/subjects/
                    mystery_and_detective_stories";
let bookDiv = '';
$.ajax({
    url: category_url,
    dataType: "json"
}).done(function (data) {
    console.log(data);
})

```

index.js:176

```

{key: "/subjects/mystery_and_detective_stories", name: "mystery and detective stories", subject_type: "subject", work_count: 10593, works: Array(12), ...} ⓘ
  ebook_count: 10593
  key: "/subjects/mystery_and_detective_stories"
  name: "mystery and detective stories"
  subject_type: "subject"
  work_count: 10593
  ▼works: Array(12)
    ► 0: {key: "/works/OL262454W", title: "The Hound of the Baskervilles"..., edition_count: 1, cov...
    ► 1: {key: "/works/OL40992W", title: "Poems", edition_count: 248, cov...
    ► 2: {key: "/works/OL262421W", title: "The Adventures of Sherlock Hol...", edition_count: 1, cov...
    ► 3: {key: "/works/OL24160W", title: "Dr Jekyll and Mr. Hyde", edition_c...
    ► 4: {key: "/works/OL262463W", title: "Memoirs of Sherlock Holmes (11)", edition_count: 1, cov...
    ► 5: {key: "/works/OL52266W", title: "The invisible man", edition_count: 1, cov...
    ► 6: {key: "/works/OL262480W", title: "The Return of Sherlock Holmes"..., edition_count: 1, cov...
    ► 7: {key: "/works/OL472693W", title: "The A.B.C. Murders", edition_count: 1, cov...
    ► 8: {key: "/works/OL262580W", title: "The Case-Book of Sherlock Hol...", edition_count: 1, cov...
    ► 9: {key: "/works/OL472420W", title: "After the Funeral", edition_count: 1, cov...
    ► 10: {key: "/works/OL262554W", title: "Works (Adventures of Sherlock Hol...", edition_count: 1, cov...
    ► 11: {key: "/works/OL77021W", title: "The Rainmaker", edition_count: 1, cov...
    length: 12
    ► __proto__: Array(0)

```

Рисунок 2

После обновления страницы вы увидите в консоли объект с множеством полей, из которых для нас будут представлять интерес `ebook_count` и `work_count`, которые содержат число, указывающее на количество книг в данной рубрике (важно, чтобы было больше 0), а также

поле **works**, в котором мы видим массив из 12 элементов (рис. 2). Это то, что нам нужно — информация о книгах.

Поскольку поле (свойство) **works** — это массив, то для вывода книг нам нужно его перебрать и на основе данных в каждом элементе массива создать **div**, отвечающий за внешний вид блока с информацией о книге. Перебирать массив в JavaScript удобно с помощью метода **forEach()**, в jQuery для этого служит специальный метод **\$.each()**, который имеет такой синтаксис:

```
$.each(arr, function(index, item){  
    console.log(index, item); })
```

где:

arr — это массив, который мы перебираем;

index — индекс элемента;

item — значение элемента.

Нам еще понадобится переменная **bookDiv**, которая сначала будет пустой строкой и в которую мы потом запишем всю разметку для книг.

Перепишем наш код для массива **data.works**:

```
let bookDiv = '';  
$.ajax({  
    url: category_url,  
    dataType: "json",  
    beforeSend: function () {  
        $('.loader').show();  
    }  
}).done(function (data) {  
    // console.log(data);  
    $('.loader').hide();
```

```
$.each(data.works, function (index, bookInfo) {
    console.log(bookInfo);

    if (bookInfo.authors.length == 1)
        var authorsInfo = 'Author: <a href="https://openlibrary.org/${bookInfo.authors[0].key}"/>${bookInfo.authors[0].name}</a>';
    else {
        var authorsInfo = 'Authors: ';
        $.each(bookInfo.authors, function (i, author) {
            authorsInfo += '<a href="https://openlibrary.org/${author.key}"/>${author.name}</a>';
        });
        authorsInfo = authorsInfo.slice(0, -2);
    }

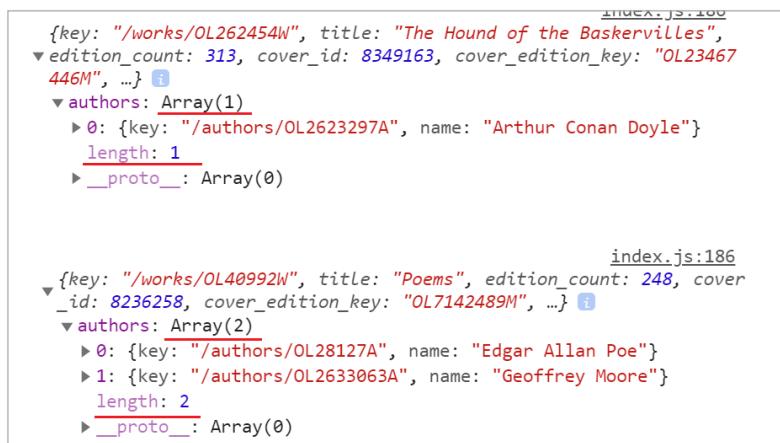
    bookDiv += '<div class="book">
        <div class="book-title">
            ${bookInfo.title.length < 50 ?
                bookInfo.title :
                bookInfo.title.slice(0, 60) + "..."}
        </div>
        <div class="book-author">
            ${authorsInfo}
        </div>
        <div class="book-cover">
            
        </div>
        <div class="book-hidden">
            <a href="https://openlibrary.org${bookInfo.key}" class="btn btn-info" target="_blank">
                More Info
            </a>
        </div>
    </div>'
```

```

    ${bookInfo.availability.is_readable ?
      '<a href="https://openlibrary.org/borrow/
      ia/${bookInfo.ia}?ref=ol"
      class="btn btn-read" target="_blank">
      Read Book</a>' : ''}
    </div>
  </div>';
});
$('.row.books').html(bookDiv)
})

```

В этом коде мы выбираем в переменную `authorsInfo` информацию об авторах. Дело в том, что у книги может быть один автор, а может быть несколько, поэтому параметр `authors` представляет собой массив, который нам тоже придется перебрать. Плюс данные в этом массиве содержат не только имя автора, но и ссылку на страницу с информацией о нем, поэтому мы формируем соответствующую разметку (рис. 3).



The screenshot shows two book objects from the Open Library API. The first book has one author, Arthur Conan Doyle. The second book has two authors, Edgar Allan Poe and Geoffrey Moore.

```

{
  key: "/works/OL262454W",
  title: "The Hound of the Baskervilles",
  ▼edition_count: 313, cover_id: 8349163, cover_edition_key: "OL23467446M", ...
  ▼authors: Array(1)
    ▶ 0: {key: "/authors/OL2623297A", name: "Arthur Conan Doyle"}
    length: 1
  ▶ __proto__: Array(0)

  ▲index.js:186
  ▼{key: "/works/OL40992W", title: "Poems", edition_count: 248, cover_id: 8236258, cover_edition_key: "OL7142489M", ...} ⓘ
  ▼authors: Array(2)
    ▶ 0: {key: "/authors/OL28127A", name: "Edgar Allan Poe"}
    ▶ 1: {key: "/authors/OL2633063A", name: "Geoffrey Moore"}
    length: 2
  ▶ __proto__: Array(0)
}

```

Рисунок 3

При формировании заголовка книги нам придется учесть, что помимо информации нам важен внешний вид блоков, поэтому слишком длинные названия мы будем укорачивать с помощью методов строк, чтобы заголовок уложился в 2 строки или в 60 символов.

```
<div class="book-title">
    ${bookInfo.title.length < 50 ? bookInfo.title :
        bookInfo.title.slice(0, 60) + '...'}
</div>
```

Вот, как это будет выглядеть:



Рисунок 4

Для каждого такого блока были написаны стили, поэтому разметка содержит ряд классов, описание которых вы найдете в css-файле. Один из блоков `<div class = "book-`

hidden"> содержит информацию, которая появляется при наведении на блок с книгой. Там будут размещены кнопки ссылки на страницу с информацией о книге и на страницу, где эту книгу можно почитать онлайн. Однако, почитать можно не все книги, поэтому нам нужно проверить значение свойства `availability.is_readable` и в зависимости от этого выводить или же не выводить вторую кнопку-ссылку.

```

{
  "key": "/works/OL262554W",
  "title": "Works (Adventures of Sherlock Holmes / Case-Book of Sherlock...",
  "author": "Arthur Conan Doyle"
}



More Info
Read Book



{
  "key": "/works/OL77021W",
  "title": "The Rainmaker",
  "author": "John Grisham"
}



More Info


```

```

  {key: "/works/OL262554W", title: "Works (Adventures of Sherlock Holmes / Case-Book o.Sig",  

  ▼ n of Four / Study in Scarlet / Valley of Fear)", edition_count: 70, cover_id: 8351939, c  

  over_edition_key: "OL26670244M", ...} ⓘ  

  ▶ authors: [...]  

  ▶ availability:  

    available_to_borrow: false  

    available_to_browse: false  

    available_to_waitlist: false  

    collection: "digitallibraryindia,JaiGya..."  

    identifier: "in.ernet.dli.2015.150489"  

    is_browsable: false  

    is_lendable: false  

    is_printdisabled: false  

is_readable: true  

    is_restricted: false
  
```

```

  {key: "/works/OL77021W", title: "The Rainmaker",  

  ▼ er", edition_count: 63, cover_id: 9322929, c  

  over_edition_key: "OL27112865M", ...} ⓘ  

  ▶ authors: [...]  

  ▶ availability:  

    error_message: "not found"  

    identifier: "rainmakergrish00gris"  

    is_browsable: false  

    is_lendable: false  

    is_printdisabled: false  

is_readable: false  

    is_restricted: true
  
```

Рисунок 5

Последняя строка в этой функции выводит всю сформированную в переменной `bookDiv` разметку в подготовленный для этого пустой `<div class="row books"></div>`.

```
$('.row.books').html(bookDiv);
```

В коде присутствует еще один параметр `beforeSend` (перед отправкой) в виде функции, которая, как правило, используется для отображения предзагрузчика в виде `<div class = "loader">` методом `show()`, пока браузер загружает файл, указанный в параметре `url`:

```
$.ajax({
  url: category_url,
  dataType: "json",
  beforeSend: function () {
    $('.loader').show();
  }
}).done(function (data) {
  $('.loader').hide();
  ...
})
```

В функции `done()`, которая обрабатывает загруженную информацию, мы предзагрузчик прячем методом `hide()`.



Рисунок 6

Для того чтобы посмотреть на то, что возвращает запрос на сервер, вы можете перейти на вкладку Network (Сеть)

в браузере и внизу кликнуть по названию загружаемого файла. Во вкладке **Preview** вы увидите ту же информацию, что и в параметре **data** функции **done()**.

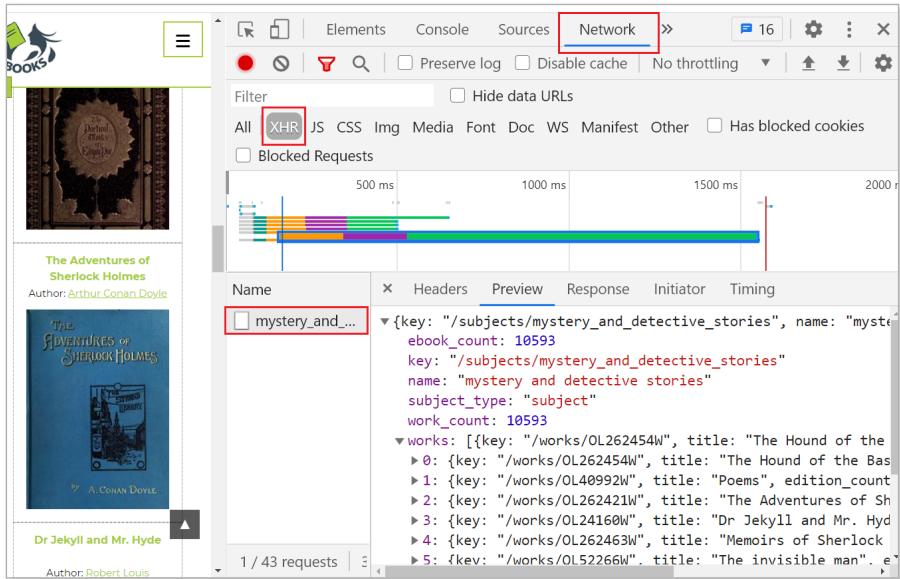


Рисунок 7

Обработка ошибок при AJAX-запросе

В том случае, если мы сделаем ошибку в адресе, ведущему к json-файлу, добавив, например, цифру 1 к адресу:

```
var category_url = "https://openlibrary.org/subjects/mystery_and_detective_stories1.json";
```

API OpenLibrary все равно вернет нам данные в виде объекта, но в его свойстве **ebook_count** будет указан **0**. Эту ситуацию мы можем предусмотреть, добавив в функции

done(), обрабатывающей успешный запрос на сервер, такое условие:

```
if (data.ebook_count == 0) {
    $('.row.books').addClass('justify-content-center my-40')
    .html('<h3>No books found</h3>');
    return false;
}
```

Тогда на странице мы увидим сообщение о том, что книги не найдены, а во вкладке Network — ответ от сервера.

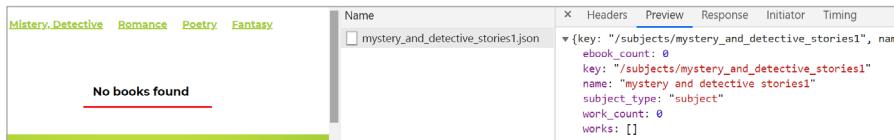


Рисунок 8

Однако ошибка может быть такой, что сервер вернет нам сообщение «Failed to load response data», т. е. «Не удалось загрузить данные ответа». Это может произойти, если в переменной **category_url** убрать расширение файла **.json**:

```
var category_url = "https://openlibrary.org/subjects/
    mystery_and_detective_stories";
```

В этом случае нам нужно будет написать код для функции **fail()**, которая как раз предназначена для обработки различных ошибок:

```
fail(function () {
    $('.loader').hide();
```

```
$('.row.books').addClass('justify-content-center  
error my-3').html('<h3 class="text-center">  
Loading of books failed...<br>An error has  
occurred.</h3>')  
});
```

Сообщение из этой функции выводится под нашим меню рубрик.

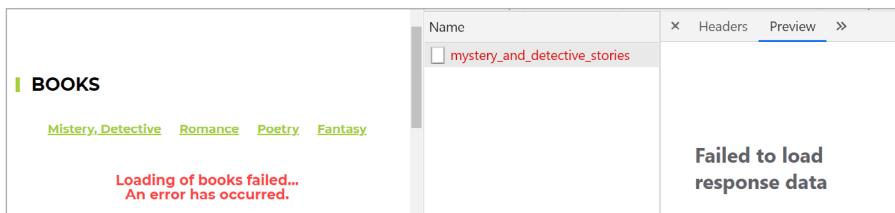


Рисунок 9

Домашнее задание

Ваше домашнее задание будет основано на файлах примера.

Задание 1

При уменьшении размера экрана изменяется вид меню. Вам необходимо будет сделать так, чтобы при клике на кнопку-гамбургер оно разворачивалось и сворачивалось с помощью одного из анимационных методов jQuery. Кроме того, внешний вид кнопки должен меняться: вместо 3-х полосок гамбургера при открытом меню должен появиться крестик и наоборот. Можно использовать для изменения кнопки html-спецсимволы × и ≡.

Подсказка: можете использовать какой-нибудь класс, например 'open' для открытого меню.

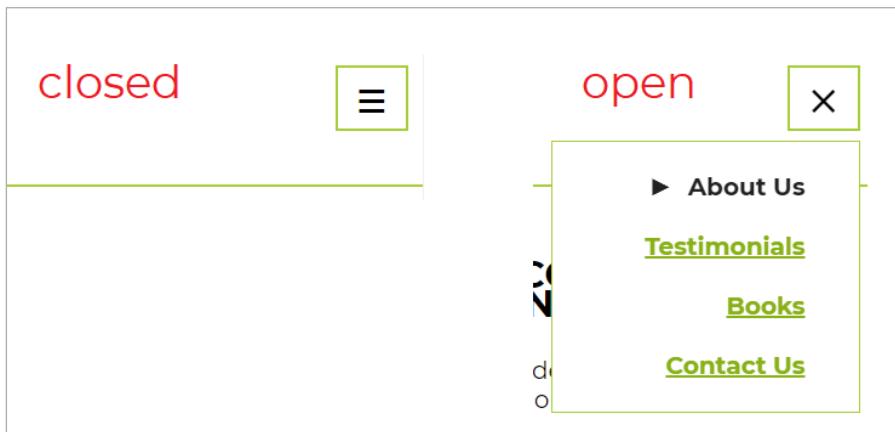


Рисунок 10

Задание 2

Обработайте клик на кнопке с текстом «That's great!» в разделе `#see-more`, выведя рядом с ней текст 'You are 1025 reader now.', но получить цифру `1025` вы должны из атрибута `data-num="1024"`, увеличив его значение на `1`. Используйте для этого метод `data()`, позволяющий работать с data-атрибутами, появившимися в HTML5, в jQuery, или метод `attr()`.

```
<button class="btn btn-read"
       id="readers-plus"
       data-num="1024">
    That's great!
</button>
<span id="num-of-readers"></span>
```

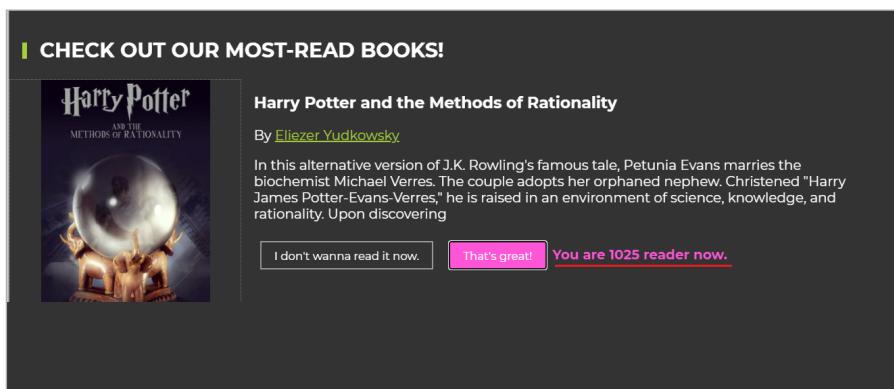


Рисунок 11

Через 1,5 секунды после появления этого сообщения перенаправьте пользователя с помощью `location.href` на страницу с адресом, указанным в ссылке с классом

`btn-read` внутри `<div class="book">` этого раздела. Получить ссылку нужно с помощью методов обхода дерева DOM, а не вставить ее в виде текста. Вы можете использовать метод `attr()` для того, чтобы добраться до значения атрибута `href` ссылки.

```
<a href="https://openlibrary.org/borrow/ia/
    hpmor?ref=ol" class="btn btn-read"
    target="_blank">Read Book</a>
```

Задание 3

Преобразуйте код AJAX-загрузки книг одной рубрики в функцию и используйте ее для загрузки книг других рубрик при клике на ссылке в верхней части раздела `Books`. Используйте значение атрибута `href` каждой из этих ссылок, чтобы получить путь к JSON-файлу рубрики на OpenLibrary. Для этого вам нужно будет добавить к ним расширение `.json`.

```
<div class="categories text-center">
    <a href="https://openlibrary.org/subjects/mystery_and_detective_stories" class="c
    <a href="https://openlibrary.org/subjects/romance" class="cat-link">Romance</a>
    <a href="https://openlibrary.org/subjects/poetry" class="cat-link">Poetry</a>
    <a href="https://openlibrary.org/subjects/fantasy" class="cat-link">Fantasy</a>
</div>
```

Рисунок 12

Для активного элемента меню должен быть назначен класс `active`, чтобы его внешний вид поменялся (подчеркнуто на рисунках 13-14).



Рисунок 13

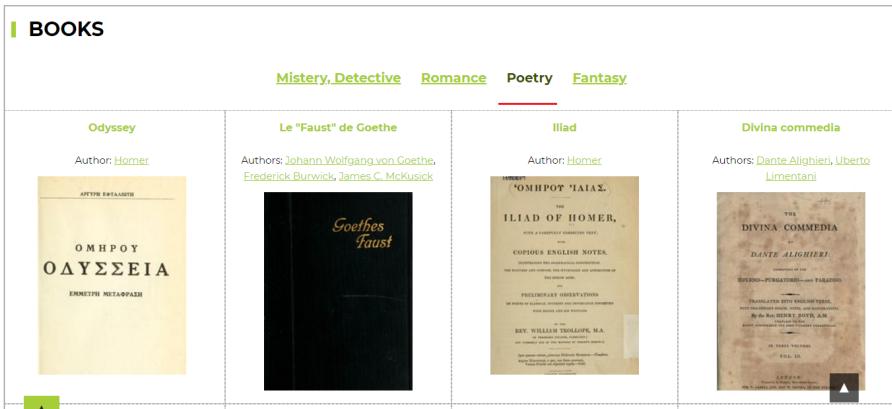


Рисунок 14

Приложение

Таблица 1. CSS-селекторы jQuery

Селектор	Описание	Пример
ПРОСТЫЕ СЕЛЕКТОРЫ		
<code>*</code>	все элементы	<code>\$("*")</code>
<code>.className</code>	элементы с классом className	<code>\$(".bg-color")</code>
<code>#idName</code>	элемент (один!) с идентификатором idName	<code>\$("#test")</code>
<code>tagName</code>	элементы с заданным именем тега	<code> \$("h2")</code>
КОМБИНИРОВАННЫЕ СЕЛЕКТОРЫ		
<code>first, second, ...</code>	элементы удовлетворяющие любому из селекторов first, second, ...	<code> \$("h2, p")</code>
<code>outer inner</code>	элементы из inner, которые являются потомками (т.е. лежат внутри) элементов из outer	<code> \$(".test p")</code>
<code>parent > child</code>	элементы из child, которые являются прямыми потомками элементов из parent	<code> \$("#wrap > div")</code>
<code>prev + next</code>	элементы из next, которые следуют непосредственно за элементами из prev	<code> \$("label + input")</code>
<code>prev ~ next</code>	элементы из next, которые следуют за элементами из prev	<code> \$(".test ~ .block")</code>
СЕЛЕКТОРЫ АТРИБУТОВ		
<code>[name]</code>	элементы, содержащие атрибут name	<code> \$('[data-url]')</code> <code> \$('[data-dismiss]')</code>
<code>[name = value]</code>	элементы, у которых значение атрибута name совпадает с value	<code> \$('[name='radio-bg'])'</code> <code> \$('[data-dismiss='true'])'</code>

Таблица 1 (продолжение)

Селектор	Описание	Пример
[name != value]	элементы, у которых значение атрибута name не совпадает с value	<code>\$("[class!='test']")</code>
[name ^= value]	элементы, у которых значение атрибута name начинается с value	<code>\$("a[href^='https://'])")</code>
[name \$= value]	элементы, у которых значение атрибута name заканчивается на value	<code>\$("[class\$='-bg']")</code>
[name*="value"]	элементы, у которых значение атрибута name содержит value	
[first][second][...]	элементы, соответствующие всем заданным условиям на атрибуты одновременно	<code>\$("[name='bg'][type='radio'])")</code>
ФИЛЬТРЫ ДОЧЕРНИХ ЭЛЕМЕНТОВ		
:first-child	элементы, которые идут первыми в своих непосредственных предках	<code>\$(".block:first-child")</code>
:last-child	элементы, которые идут последними в своих непосредственных предках	<code>\$(".block:last-child")</code>
:nth-child()	элементы, которые расположены в своих непосредственных предках по определенным условиям	<code>\$(".block:nth-child(3n)")</code> ;
:only-child	элементы, которые являются единственными в своих непосредственных предках	<code>\$(".block img:only-child")</code>
:only-of-type	элементы, являющиеся единственными, потомками в своих родительских элементах	<code>\$("button:only-of-type")</code>
:first-of-type	те из выбранных элементов, которые первыми встречаются в своих родительских элементах	<code>\$("span:first-of-type")</code>

Таблица 1 (продолжение)

Селектор	Описание	Пример
:last-of-type	те из выбранных элементов, которые последними встречаются в своих родительских элементах	<code>\$('text:last-of-type')</code>
:nth-last-of-type()	те из выбранных элементов, которые являются определенными по счету относительно последнего элемента в своем элементе-родителе	<code>\$("ul li:nth-last-of-type(2)")</code>
:nth-of-type()	дочерние элементы определенного типа в определенной последовательности внутри родительского элемента	<code>\$(".column:nth-of-type(3n)")</code>

Таблица 2. jQuery-селекторы

Селектор	Описание	Пример
ПРОСТИЕ ФИЛЬТРЫ		
:first	первый найденный элемент	<code>\$(".block:first")</code>
:last	последний найденный элемент	<code>\$(".block:last")</code>
:eq(n)	элемент идущий под заданным номером среди выбранных	<code>\$(".block:eq(3)")</code>
:not(selector)	все найденные элементы, кроме указанных в selector	<code>\$(":not(.color)")</code>
:even	элементы с четными номерами позиций, в наборе выбранных элементов	<code>\$(".block:even")</code>
:odd	элементы с нечетными номерами позиций, в наборе выбранных элементов	<code>\$(".block:odd")</code>
:gt()	элементы с индексом превышающим n	<code>\$(".block:gt(2)")</code>
:lt()	элементы с индексом меньшим, чем n	<code>\$(".block:lt(3)")</code>

Таблица 2 (продолжение)

Селектор	Описание	Пример
:header	элементы, являющиеся заголовками (с тегами h1, h2 и т. д.)	<code>\$(":header")</code>
:animated	элементы, которые в данный момент задействованы в анимации	<code>\$(".block:animated")</code>
:hidden	невидимые элементы страницы	<code>\$("form:hidden")</code>
:visible	видимые элементы страницы	<code>\$("form:visible")</code>
:lang(language)	элементы, в которых указаны языки содержимого	<code>\$("html:lang(ru)")</code>
:root	элемент, который является корневым в документе.	<code>\$(":root")</code>

Таблица 3. Селекторы форм

Селектор	Описание	Пример
:button	элементы с тегом <code><button></code> или типом <code>button</code> (<code><input type="button"></code>)	<code>\$(':button').addClass('shadow');</code>
:checkbox	элементы с атрибутом <code>type="checkbox"</code>	<code>\$(':checkbox').addClass('shadow');</code>
:checked	выбранные элементы (со статусом/атрибутом <code>checked</code>). Это могут быть элементы с атрибутом <code>type="checkbox"</code> или <code>type="radio"</code> .	<code>\$(':checked').addClass('shadow');</code>
:disabled	неактивные элементы формы (со статусом <code>disabled</code>)	<code>\$(':disabled').prop('disabled', false);</code>
:enabled	активные элементы формы (со статусом <code>enabled</code> или без атрибута <code>disabled</code>)	<code>\$(':enabled').addClass('bg-gray');</code>
:focus	элементы формы, находящиеся в фокусе	<code>\$(':focus').addClass('bg-red');</code>
:file	поля загрузки файлов (<code><input type="file"></code>)	<code>\$(':file).addClass('red-border');</code>
:image	элементы, являющиеся изображениями для отправки формы (<code><input type="image"></code>)	<code>\$(':image').prop('disabled', true);</code>

Таблица 3 (продолжение)

Селектор	Описание	Пример
<code>:input</code>	элементы, являющиеся элементами формы (с тегами <code>input</code> , <code>textarea</code> или <code>button</code>)	<code>\$('.input').addClass('bg-green shadow')</code>
<code>:password</code>	поля для ввода пароля (<code><input type="password"></code>)	<code>\$('.password').addClass('bg-red')</code>
<code>:radio</code>	элементы, являющиеся переключателями, т.е. имеющие <code>type="radio"</code>	<code>\$('.radio').addClass('shadow')</code>
<code>:reset</code>	кнопки для очистки формы (<code><input></code> или <code><button></code>)	<code>\$('.reset').addClass('shadow bg-gray');</code>
<code>:selected</code>	выбранные элементы (со статусом <code>selected</code>), как правило, элементы типа <code><option></code>	<code>\$('.selected').addClass('gray-border');</code>
<code>:submit</code>	кнопки для отправки формы (<code><input></code> или <code><button></code>)	<code>\$('.submit').addClass('shadow');</code>
<code>:text</code>	любые текстовые поля	<code>\$('.text').addClass('bg-blue');</code>

Примеры выбора элементов страницы в зависимости от используемых селекторов вы найдете на ресурсе [codepen.io](#) или в файле `examples/jquery-selectors.html`.

Для того чтобы увидеть, как действует селектор, вам необходимо будет сделать клик по любой радио-кнопке и обратить внимание на изменившиеся стили. В некоторых случаях над выбранным элементом будет появляться стрелка, указывающая на то, что изменилось.

See jQuery selectors [here](#)

Рисунок 15

Чтобы понять, какие именно селекторы были использованы, придется заглянуть в js-код.

The screenshot shows a computer screen with a dark-themed code editor. The code in the editor is:

```

13 return prev.selector();
14
15 $(':header').andSelf()
16 const children = $(this).children();
17 if (!children.length) {
18   $(header).append(this);
19   $(children);
20 }
21
22 return header;
  
```

Below the code editor, there is a title "jQuery Selectors" followed by a grid of checkboxes for various CSS and jQuery selector types. The grid is organized into several columns:

- CSS-selectors** (Columns 1-2):
 - *
 - #id
 - .className
 - tagName
 - first, second, ...
 - outer inner
- jQuery-selectors(filters)** (Column 3):
 - parent > child
 - prev + next
 - [name ^= value]
 - [name \$= value]
 - [name *= value]
 - [name != value]
 - :only-child
 - :first-of-type
 - :last-of-type
 - :nth-child(2n+1)
 - :eq(n)
- jQuery Form elements selectors (filters)** (Column 4):
 - :first
 - :last
 - :not(selector)
 - :even
 - :odd
 - :header
 - :hidden
 - :visible
 - :root
- jQuery Form elements filters** (Column 5):
 - :button
 - :checkbox
 - :checked
 - :enabled
 - :disabled
 - :radio
 - :submit
 - :reset
 - :text

Below the grid, there are five columns of placeholder text and images:

Column 1	Column 2	Column 3	Column 4	Column 5
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam sequi eligendi iste magni.</p> <p>Dolorum illo nostrum vel. Explicabo alias vel placeat lorem querat quis praesentium!</p> <p>Debitis assumenda magna, ad mollitia repellat consequuntur eveniet molestiae nisi similique!</p>	<p>Dobrom illo nostrum vel. Explicabo alias vel placeat lorem querat quis praesentium!</p> <p>Ab rem at deflectus, praesentium minus harum modi incident autem eaque quis velit accusamus?</p> <p>Paratur cupiditate doloremque commodi omnis ipsam modi incident, explicabo provident.</p>	<p>Obsecrati voluntibus amet esse at. Veniam accusamus, fugiat eligendi esse ullam architecto!</p> <p>Debitis assumenda magna, ad mollitia repellat consequuntur eveniet molestiae nisi similique!</p>	<p>Beariae harum quasi dolore voluplates amet veritatis minima cumque quos nemo? Consequuntur.</p> <p>Illo nostrum vel. Explicabo alias consecetur vel placeat minima cumque quos nemo? Consequuntur.</p> <p>Consequuntur.</p>	<p>Illo nostrum vel. Explicabo alias consecetur vel placeat minima cumque quos nemo? Consequuntur.</p> <p>Error facere adipisci voluplates qua facilis, delectili, aliquam a totam suscipit consequatur?</p> <p>Cum dolorum perferendis iure possimus voluplates similique mollitia repellat, ab omnis! Porro.</p>

Contact Form

Form fields:

- Consultation
- Order
- Complaint

Text input fields:

- Your Name
- Your Phone
- Subject
- Your Email
- Your Message

Buttons:

- Submit
- Reset

Text at the bottom: See jQuery selectors [here](#)

Рисунок 16

Таблица 4. Методы управления стилями и классами jQuery

Метод	Описание	Пример
МЕТОДЫ УПРАВЛЕНИЯ СТИЛЯМИ JQUERY		
<code>\$(‘селектор’).css(‘property’)</code>	получить значение указанного css-свойства	<code>\$('#block').css('display');</code>
<code>\$(‘селектор’).css(‘property’, ‘value’);</code>	установить 1 значение	<code>\$('#block').css('color', 'red');</code>
<code>\$(‘селектор’).css({prop1: ‘value1’, ‘prop2’: ‘value2’, prop3: ‘value3’});</code>	установить несколько значений. CSS-стили можно записывать в кавычках в нотации CSS или без кавычек в нотации JavaScript с помощью camelCase	<code>\$('.example').css({color: 'red', 'background-color': 'pink', fontSize: '18px'});</code>
МЕТОДЫ УПРАВЛЕНИЯ КЛАССАМИ JQUERY		
<code>\$(‘селектор’).addClass(‘имя_класса’)</code>	Добавляет класс к списку классов элемента. Можно добавить несколько через пробел	<code>\$(‘a’).addClass(‘read-more’); \$(‘.btn’).addClass(‘btn-outline btn-secondary’);</code>
<code>\$(‘селектор’).removeClass(‘имя_класса’)</code>	Удаляет класс из списка классов элемента. Можно удалить несколько через пробел	<code>\$(‘.nav-links.active’).removeClass(‘active’); \$(‘.products’).removeClass(‘col-md-4 col-sm-6’);</code>
<code>\$(‘селектор’).toggleClass(‘имя_класса’)</code>	Добавляет/удаляет класс(-ы) из списка классов элемента в зависимости от того, назначен ли соответствующий класс(-ы) элементу.	<code>\$(‘.nav-links’).toggleClass(‘active’); \$(‘.products’).toggleClass(‘col-md-4 col-sm-6’);</code>
<code>\$(‘селектор’).hasClass(‘имя_класса’)</code>	Проверяет, назначен ли указанный класс элементу. Если да, то возвращает true, в противном случае — false	<code>if(\$('.nav-links').hasClass(‘active’)) { \$('.nav-links’).parent().addClass(‘menu-active’); }</code>

Таблица 5. Traversing. Методы обхода DOM

Метод	Описание	Пример
<u>add()</u>	Добавление элементов в набор выбранных элементов	<code>\$('.column h3').add('p').addClass('red-border');</code>
<u>addBack()</u>	После поиска дочерних элементов, добавляет предыдущий набор элементов к текущему для назначения стилей или обработчиков событий	<code>\$('.column').find('p').addBack().addClass('red-border');</code>
<u>children()</u>	Возвращает все прямые дочерние элементы выбранного элемента	<code>\$('.column-selected').children().addClass('red-border');</code>
<u>closest()</u>	Возвращает первый предок выбранного элемента	<code>\$('.column').closest('.container').addClass('gray-border');</code>
<u>contents()</u>	Возвращает все прямые дочерние элементы выбранного элемента (включая узлы текста и комментария)	<code>\$('.column-selected').contents().addClass('shadow');</code>
<u>each()</u>	Выполняет функцию для каждого выбранного элемента	<code>\$('.column').each(function(ind, elem){ let columnText = \$(elem).find('p'); columnText.html(ind+1) + columnText.text();});</code>
<u>end()</u>	Завершает последнюю операцию фильтрации (поиска) вложенных элементов в текущей цепочке и возвращает набор элементов в предыдущее состояние	<code>\$('.column').find('.text').addClass('shadow').end().find('img').addClass('shadow');</code>
<u>eq()</u>	Возвращает элемент с указанным индексом в списке выбранных элементов. Индексы считаются с 0.	<code>\$('.column').eq(1).addClass('bg-blue');</code>

Таблица 5 (продолжение)

Метод	Описание	Пример
<u>filter()</u>	Возвращает набор совпадающих элементов, которые соответствуют указанному селектору или функции	<code>\$('.column img').filter('.snow').addClass('gray-border shadow');</code>
<u>find()</u>	Возвращает элементы-потомки выбранного элемента по указанному селектору	<code>\$('.column').find('.text').addClass('shadow');</code>
<u>first()</u>	Возвращает первый элемент среди выбранных	<code>\$('.column').first().addClass('bg-red');</code>
<u>has()</u>	Возвращает все элементы, которые имеют один или несколько элементов внутри них с указанным селектором	<code>\$('.column').has('img.snow').addClass('bg-blue');</code>
<u>is()</u>	Проверяет набор совпадающих элементов на объект Selector/элемент/jQuery и возвращает значение true, если хотя бы один из этих элементов соответствует заданным аргументам	<code>\$('.column').each(function(i, elem){ if(\$(this).is(':nth-child(2n)')) \$(this).addClass('bg-red');});</code>
<u>last()</u>	Возвращает последний элемент из выбранных элементов	<code>\$('.column').last().addClass('bg-green');</code>
<u>map()</u>	Передает каждый элемент из набора с помощью функции, производя новый объект jQuery, содержащий возвращаемые значения	<code>\$('.container: eq(1)').prepend('<p class="text-center">Photo titles: '+ \$('.column img').map(function(ind, elem){ return \$(elem).attr('alt'); }).get().join(' ')+'</p>');</code>
<u>next()</u>	Возвращает следующий родственный элемент выбранного элемента	<code>\$('.column-selected').next().addClass('bg-green');</code>
<u>nextAll()</u>	Возвращает все соседние элементы выбранного элемента	<code>\$('.column-selected').nextAll().addClass('bg-red');</code>

Таблица 5 (продолжение)

Метод	Описание	Пример
nextUntil()	Возвращает все соседние элементы одного уровня между двумя заданными аргументами	<code>\$('.column p').nextUntil('img').addClass('shadow');</code>
not()	Возвращает элементы, не совпадающие с определенными критериями	<code>\$('.column').not(':nth-child(2)').addClass('bg-green');</code>
offsetParent()	Возвращает первый позиционный родительский элемент	<code>\$('.column-selected').offsetParent().addClass('gray-border');</code>
parent()	Возвращает прямой родительский элемент для выбранного элемента	<code>\$('.column').parent().addClass('red-border');</code>
parents()	Возвращает все элементы-предки выбранного элемента	<code>\$('.column').parents('article').addClass('bg-red shadow');</code>
parentsUntil()	Возвращает все элементы-предки заданного элемента — не только непосредственного, но и прародителя, прапрародителя и так далее до корневого элемента (то есть до тега html).	<code>\$('.column').parentsUntil('.container').addClass('bg-blue');</code>
prev()	Возвращает предыдущий родственный элемент выбранного элемента (-ов)	<code>\$('.column-selected').prev().addClass('bg-blue');</code>
prevAll()	Возвращает все предыдущие родственные элементы выбранного элемента	<code>\$('.column-selected').prevAll().addClass('bg-red shadow');</code>
prevUntil()	Возвращает все предыдущие элементы одного уровня между двумя заданными аргументами	<code>\$('.column p').prevUntil('img').addClass('shadow');</code>
siblings()	Возвращает все родственные элементы выбранного элемента	<code>\$('.column p:first-of-type').siblings().addClass('gray-border');</code>

Таблица 5 (продолжение)

Метод	Описание	Пример
slice()	Выбирает из набора элементов только те, чьи индексы указаны в диапазоне в скобках	<code>\$('.column').slice(1,4).addClass('bg-red shadow');</code>
wrap()	Помещает выбранные элементы внутрь заданного элемента	<code>\$('.column').wrap('<div class="bg-green shadow"></div>');</code>
wrapAll()	Оборачивает все выбранные элементы в общий элемент-обертку	<code>\$('.column').wrapAll('<div class="bg-red shadow"></div>');</code>

В примере, который находится в папке *examples/jQuery-traversing-methods.html* или на ресурсе codepen.io, вы можете пощелкать по радио-переключателям, чтобы посмотреть работу всех перечисленных методов (рис. 17).

jQuery Traversing

add()
 end()
 next()
 first()
 each()
 Reset styles

addBack()
 closest()
 nextAll()
 last()
 map()

children()
 parent()
 nextUntil()
 siblings()
 eq()

contents()
 parents()
 prev()
 slice()
 not()

filter()
 parentsUntil()
 prevAll()
 has()
 wrap()

find()
 offsetParent()
 prevUntil()
 is()
 wrapAll()

Column 1 Lorem ipsum dolor sit, amet consectetur adipisciing elit. Esse sequi eligendi iste magni. Dolorem illo nostrum vel. Explicabo alias vel placeat lorem quaerat quis praesentium!	Column 2 Dolorem illo nostrum vel. Explicabo alias vel placeat lorem quaerat quis praesentium! Ab rem at delectus, praesentium minus harum modi incidunt autem eaque quis velit accusamus?	Column 3 Obcaecati voluptatibus amet esse at. Veniam accusamus, fugiat eligendi esse ullam architecto! Paratur cupiditate doloremque commodi omnis ipsam modi incidunt, explicabo provident.	Column 4 Beatae harum quasi dolore voluptates amet veritatis minima cumque quos nemo? Consequuntur. Illo nostrum vel. Explicabo alias consequetur vel placeat minima cumque quos nemo? Consequuntur.	Column 5 Illo nostrum vel. Explicabo alias consequetur vel placeat minima cumque quos nemo? Consequuntur. Error facere adipisci voluptates quia facilis, deleniti, aliquam a totam suscipit consequatur?
				

Рисунок 17

В коде добавляются различные классы со стилями, выделяющими какие-либо элементы в зависимости от выбранного метода.

По центру выделена колонка, относительно которой определяются элементы такими методами, как `prev()`, `prevAll()`, `next()` и др.

Код из этого файла использован в таблице с описанием методов в виде примеров.

Обратите внимание на то, как визуально изменяется разметка примера при использовании методов `wrap()` и `wrapAll()` (рис. 18).

Стоит заглянуть в Инспектор свойств браузера, чтобы увидеть, почему так происходит.

```
... ▼<article> == $0
  <h1 class="text-center">jQuery Traversing</h1>
  ▶<div class="container">...
  ▼<div class="container related">
    ▼<div class="row column-content">
      ▶<div class="column">...
      ▶<div class="column">...
      ▶<div class="column column-selected">...
      ▶<div class="column">...
      ▶<div class="column">...
    </div>
  </div>
</article>
```

start markup

Рисунок 18

Метод `wrap()` обернул в `<div class="bg-green shadow">` каждый из `<div class="column"></div>` (рис. 19).

Метод `wrapAll()` обернул в `<div class="bg-red shadow">` все `<div class="column"></div>` (рис. 20).

```

▼<article>
  <h1 class="text-center">jQuery Traversing</h1>
  ▶<div class="container">...</div>
  ▼<div class="container related">
    ▼<div class="row column-content">
      ▼<div class="bg-green shadow"> == $0
        ▶<div class="column">...</div>
      </div>
      ▼<div class="bg-green shadow">
        ▶<div class="column">...</div>
      </div>
      ▼<div class="bg-green shadow">
        ▶<div class="column column-selected">...</div>
      </div>
      ▼<div class="bg-green shadow">
        ▶<div class="column">...</div>
      </div>
      ▼<div class="bg-green shadow">
        ▶<div class="column">...</div>
      </div>
    </div>
    after wrap()
  </article>

```

Рисунок 19

```

▼<article>
  <h1 class="text-center">jQuery Traversing</h1>
  ▶<div class="container">...</div>
  ▼<div class="container related">
    ▼<div class="row column-content">
      ...
      ▼<div class="bg-red shadow"> == $0
        ▶<div class="column">...</div>
        ▶<div class="column">...</div>
        ▶<div class="column column-selected">...</div>
        ▶<div class="column">...</div>
        ▶<div class="column">...</div>
      </div>
    </div>
    after wrapAll()
  </article>

```

Рисунок 20