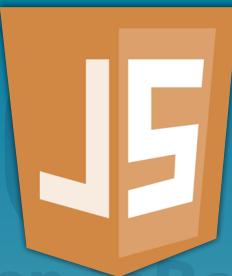


# Разработка клиентских сценариев с использованием JavaScript и библиотеки jQuery



# Урок № 11

## Введение в jQuery

### Содержание

<b>Использование jQuery .....</b>	<b>4</b>
Что такое jQuery.....	4
Подключение jQuery .....	7
Версии jQuery .....	10
Доступ к элементам страницы	
при помощи функции \$ .....	11
Понятие селектора.....	14
CSS селекторы.....	14
jQuery селекторы.....	16
Селекторы форм.....	16
Особенности селекторов в jQuery .....	17
Методы управления стилями jQuery .....	18
Traversing. Методы обхода DOM.....	20
Обработка событий в jQuery .....	22
Назначение одного обработчика	
для нескольких событий .....	28
Анимационные методы jQuery.....	29

Методы, изменяющие высоту/ширину элемента и его видимость (display).....	30
Методы, изменяющие непрозрачность элемента (свойство opacity).....	33
Метод animate() для создания произвольных эффектов.....	35
Делегирование событий.....	37
Отмена обработчиков событий.....	43
Обработка данных форм.....	46

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

# Использование jQuery

## Что такое jQuery

jQuery — это библиотека функций, написанная на JavaScript, позволяющая простыми методами манипулировать DOM-элементами, обрабатывать события, использовать анимацию, а также загружать или получать данные на основе AJAX.



Рисунок 1

Разработчиком jQuery является Джон Резиг (*John Resig*), который опубликовал первую версию этой библиотеки в 2006 году на компьютерной конференции «BarCamp» в Нью-Йорке. Она мгновенно стала популярной, т. к. решала несколько проблем, существенных для программистов на JavaScript:

- Позволяла легко обращаться к элементам, используя CSS-селекторы и различные фильтры.

- Была кросбраузерной, т.е. работала одинаково хорошо и в Opera, и Mozilla Firefox, и в Internet Explorer, который был достаточно популярен в то время.
- Давала возможно делать анимационные эффекты в то время, когда css-свойство `transition`, `animation` и `@keyframes` еще не существовали.

В свое время Джон Резиг написал: «jQuery — это Javascript-библиотека, в основе которой лежит девиз: Программировать на Javascript должно быть приятно. jQuery берёт частые, повторяющиеся задачи, извлекает всю ненужную разметку и делает их короткими, элегантными и понятными».

В первой версии jQuery еще не имела средств работы с AJAX, но они были очень скоро добавлены. В январе 2006 года был создан первый плагин для работы с JSON-данными. Он показал, что jQuery может расширять функциональность за счет плагинов. Поэтому в скором времени таких плагинов было создано множество, что стало одной из причин успеха jQuery на долгие годы.

Сейчас разработка jQuery ведётся командой добровольцев на пожертвования. Об истории версий jQuery вы можете прочитать на [официальном сайте](#).

На данный момент популярность jQuery идет вниз, т. к. современные стандарты уже поддерживают многое из того, что изначально принесло популярность этой библиотеке:

- Выбор элементов по css-селектору (методы `document.querySelector()` и `document.querySelectorAll()`);
- Переключения классов (`element.classList.add()`, `element.classList.remove()`, `element.classList.toggle()`);

- Альтернативные способы передачи данных (Fetch API, а не AJAX).

Очень вероятно, что к появлению этих возможностей в стандарте JavaScript подтолкнула именно библиотека jQuery.

Зачем вам учить jQuery, если вы знаете JavaScript? Во-первых, затем, что она упрощает понимание кода и позволяет в 3 строчки сделать то, что на нативном JS займет 10-15 строк, а иногда и больше. Во-вторых, затем, что на основе jQuery написано огромное количество плагинов, позволяющих решить какие угодно задачи, задействовав минимум кода и усилий. В-третьих, в сети существует масса сайтов, которые используют jQuery, и в случае добавления/изменения кода вам придется разбираться, что и как в них работает. В-четвертых, большинство современных CMS (Wordpress, Drupal, Joomla и другие) используют функциональность jQuery и для работы основных своих функций (меню, перетаскиваемые элементы, отправка AJAX-запросов и т. д.), так и для работы шаблонов (тем), плагинов и других расширений для этих CMS. В-пятых, “порог входления” в jQuery очень низкий, т. е. для того, чтобы разобраться, как и что работает в этой библиотеке, необходимо от нескольких часов до 2-3-х дней, даже, если вам никто об этом не рассказывает. Вряд ли вы столь же быстро разберетесь с React, Angular или Vue.js.

При этом вы должны понимать, что jQuery — это библиотека, написанная на JavaScript, то есть это не какая-то надстройка над JS, а переработанный и упрощенный подход к написанию кода, к тому же кроссбраузерный.

## Подключение jQuery

Официальный сайт вы найдете по адресу <https://jquery.com/>. На нем всегда можно скачать свежую версию jQuery. На данный момент — это версия 3.5.1.

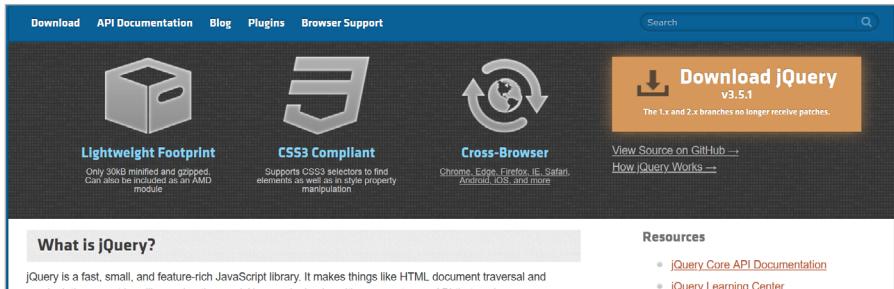


Рисунок 2

При клике на ссылку «Download jQuery» вы перейдете на страницу, которая позволит вам скачать последнюю версию библиотеки в нескольких вариантах:

- **Compressed, production jQuery** — сжатый, минифицированный файл, который стоит использовать для работы со страницами сайта. То есть весит этот файл вдвое меньше, чем несжатый, что ускоряет загрузку сайта, но он практически нечитабелен, а значит, код в нем вам будет непонятен.
- **Uncompressed, development jQuery** — несжатый файл для разработчиков плагинов с комментариями ко многим функциям. Это файл с читабельным кодом позволяет понять, как устроена jQuery изнутри и воспользоваться ее функциями для написания расширения.
- **Slim build** — «тонкая сборка», т. е. уменьшенная версия jQuery, откуда убраны все анимационные методы

и методы для работы с AJAX. Соответственно, использовать в коде вы их не сможете. Это вариант для тех, кто не собирается ничего загружать или отправлять с помощью AJAX-технологии, а вместо анимационных методов будет использовать CSS-свойства `@keyframes`, `animation` и `transition`.

#### jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed, production jQuery 3.5.1](#)

[Download the uncompressed, development jQuery 3.5.1](#)

[Download the map file for jQuery 3.5.1](#)

You can also use the `slim` build, which excludes the `ajax` and `effects` modules.

[Download the compressed, production jQuery 3.5.1 slim build](#)

[Download the uncompressed, development jQuery 3.5.1 slim build](#)

[Download the map file for the jQuery 3.5.1 slim build](#)

[jQuery 3.5.1 release notes](#)

Рисунок 3

Мы будем использовать для разбора HTML-страницы с примером compressed-версию jQuery. Именно ее и стоит скачать. Учтите, что на момент чтения вами этого урока номер версии jQuery может отличаться. Имеет смысл скачать именно последнюю версию, т.к. она является наиболее актуальной.

Для того чтобы загрузить выбранную версию jQuery кликните на ссылке, нажмите **CTRL+S** и сохраните файл в нужную вам папку со всеми файлами вашего сайта. Чаще всего ее сохраняют в папку `js`, в которой хранят и другие JS-файлы. Для подключения jQuery в файл необходимо указать в атрибуте `src` тега `<script>` путь к загруженному файлу с этой библиотекой, например, в блоке `<head>`:

```
<head>
    <meta charset="UTF-8">
    <title>Index</title>
    <link rel="stylesheet" href="css/style.css">
    <script src="js/jquery.min.js"></script>
    <script src="js/your-script.js"></script>
</head>
```

... или перед закрывающимся `</body>`:

```
<script src="js/jquery.min.js"></script>
<script src="js/your-script.js"></script>
</body>
</html>
```

Обратите внимание на строку

```
<script src="js/your-script.js"></script>
```

Она идет следом за подключением jQuery и содержит ссылку на файл скрипта, в котором вы будете писать код, используя функциональность jQuery.

Помимо подключения jQuery из папки вашего проекта, довольно распространенным является способ подключения этой библиотеки с CDN — сетей распределенного контента, которые позволяют загружать множество популярных библиотек, плагинов и фреймворков, указав ссылку на них. Для загрузки jQuery вы можете использовать такие CDN:

- [jQuery CDN](#);
- [Google CDN](#);
- [Microsoft CDN](#);

- [CDNJS CDN](#);
- [jsDelivr CDN](#).

В этом случае подключение jQuery будет выглядеть так:

```
<script src="https://ajax.googleapis.com/ajax/libs/
            jquery/3.5.1/jquery.min.js"></script>
<script src="js/your-script.js"></script>
```

Плюсом такого подключения является то, что вы можете писать код, не имея на компьютере скачанной версии jQuery, и он будет работать. Также есть вероятность, что вы или пользователь, который будет просматривать в браузере вашу html-страницу (сайт), уже загружал jQuery с этого ресурса, и она осталась в кэше браузера. Тогда не придется тратить время на загрузку этой библиотеки.

К минусам загрузки jQuery с CDN можно отнести то, что при отключенном Интернете вы не сможете писать код на jQuery, т. к. не будет описания функций, которые вы наверняка используете в коде. В этом случае ваш скрипт выдаст ошибку вида

```
Uncaught ReferenceError: $ is not defined
```

## Версии jQuery

На данный момент принято использовать jQuery версии 3\*, т.е. 3.0, 3.1, 3.2 и т.д. Они основаны на стандартах ES6, ES7, поддерживают новые возможности типа промисов, цикла `for..of`, нового API для работы с анимацией и [другие \(рус\)](#).

Для совместимости со всеми старыми браузерами стоит использовать версии линейки 1\*, т. к. они поддер-

живают совместимость с браузером Internet Explorer v6-8. Самой последней из них является версия 1.12.4.

В версиях 2.\* от поддержки старых IE отказались. Поэтому линейка версий 2.\* немного легче из-за вырезанного кода с поддержкой IE v.6-8. Она является промежуточной между версиями 1 и 3, и встречается в основном на тех сайтах, где была подключена во время ее выпуска.

## Доступ к элементам страницы при помощи функции \$

В jQuery к любому объекту необходимо обращаться, используя ключевое слово jQuery или его замену — \$. Для того, чтобы использовать эту библиотеку в заголовочной части страницы, когда весь контент еще не загрузился, необходимо записать весь код внутри такой функции:

```
jQuery(document).ready(function() {
    // ваш код
});

$(document).ready(function() {
    // ваш код
});

// используем стрелочную функцию
$(document).ready(()=>{
    $('.block').addClass('active');
    // ваш код
});

// укороченный вариант
$(function() {
    $('.nav-link').addClass('active');
    // ваш код
});
```

```
// используем стрелочную функцию
$(() => {
  $('.block').attr('data-text', 'Test Block');
  // ваш код
});
```

Функций много, но на самом деле все они выполняют одно и то же — отслеживают момент загрузки DOM-дерева документа, чтобы позволить скрипту с ним работать без ошибок. Какую выбрать вы решаете сами в зависимости от того, насколько вы поняли синтаксис стрелочных функций или функций-коллбэков.

В большинстве руководств по jQuery вы найдете вариант

```
$ (document).ready(function() {
  // ваш код
});
```

Можете использовать именно его.

В том случае, если вы подключаете свой скрипт перед закрывающимся тегом `</body>`, код в синтаксисе jQuery можно записывать без `$(document).ready()`.

Для того чтобы разобраться с возможностями jQuery, мы рассмотрим с вами, каким образом можно добавить интерактивности для страницы воображаемой онлайн-библиотеки, состоящей из шапки, нескольких разделов и подвала. Готовый пример с HTML-разметкой, CSS-стилями и кодом на jQuery вы найдете в папке `examples/books` этого урока. Файл `index.html` в этой папке предполагает подключение jQuery и плагинов с CDN, а `index-local.html` — из папки `books/js`.



About Us Testimonials Books Contact Us

**WELCOME TO BOOKS ONLINE LIBRARY!**

Epos provide how at this mistaken idea of denouncing pleasure and痛斥以行乐为耻的错误观念。该句出自古罗马帝国元老院的一位演说家之口，他痛斥行乐为耻的错误观念。该句出自古罗马帝国元老院的一位演说家之口，他痛斥行乐为耻的错误观念。

We are the best online library at 2021

- Our books
- Our mission
- Our message
- Our Vision
- Our Library
- Our Book API

**OUR CUSTOMERS TESTIMONIALS**



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Ut labore et dolore magna aliquyam erat, sed diam voluptua. Ut labore et dolore magna aliquyam erat, sed diam voluptua.

John Doe



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Ut labore et dolore magna aliquyam erat, sed diam voluptua. Ut labore et dolore magna aliquyam erat, sed diam voluptua.

Jane Smith

**BOOKS**

Horror, Detective, Romance, Poetry, Classics

The Hound of the Baskervilles	Poems	Adventures of Sherlock Holmes	Dr Jekyll and Mr. Hyde
Author: Arthur Conan Doyle	Author: Edgar Allan Poe	Author: Arthur Conan Doyle	Author: Robert Louis Stevenson
			
The Invisible Man	The Memoirs of Sherlock Holmes	Memoirs of Sherlock Holmes (II)	The ABC Murders
Author: H.G. Wells	Author: Arthur Conan Doyle	Author: Arthur Conan Doyle	Author: Agatha Christie
			
The Case-Book of Sherlock Holmes	After the Funeral	Memoirs of Sherlock Holmes (III)	The Maltese Falcon
Author: Arthur Conan Doyle	Author: Arthur Conan Doyle	Author: Arthur Conan Doyle	Author: Agatha Christie
			

**Subscribe to our newsletter**

Sign up for our newsletter to get the latest news and books.

Subscribe Now

**CONTACT US**

Write us a letter. We're in touch.

Your name\*  Your email\*   
 Subject  Your phone\*   
 Your message\*

\* All fields are required

Send a letter

Books Online Library | 2021. All rights reserved.

Рисунок 4

Поскольку на этой странице существует масса html-элементов, первое, с чем нам необходимо разобраться — это как выбирать из всего множества нужные элементы с помощью jQuery. Для этой цели мы будем использовать селекторы jQuery. Сначала познакомимся с теорией.

## Понятие селектора

Понятие селектора должно быть вам знакомо из CSS и JavaScript, если вы использовали методы `document.querySelector()` или `document.querySelectorAll()`. В jQuery селектором является строка, в которой задаются условия поиска DOM-элементов на странице. В качестве такой строки можно использовать CSS-селекторы или специальные jQuery-селекторы, которые еще называют фильтрами.

## CSS селекторы

Общий вид записи CSS-селектора выглядит так:

```
$( 'селектор' )
```

Например, `$('#date-now')` предоставит вам доступ к единственному элементу на странице, который имеет атрибут `id="date-now"`. Селектор `$(".accordion-header")` позволит обращаться сразу к нескольким элементам с классом `.accordion-header`, а селектор  `$('[href^="#"]')` даст доступ ко всем ссылкам, у которых атрибут `href` начинается с `#`. Такой селектор, как `$(".block:nth-child(3n)")` позволит выбрать каждый 3-й элемент с классом `.block`. То есть выбор элементов осуществляется так же, как это работа-

ет с CSS-селекторами. Только в случае jQuery мы будем вызывать для этих селекторов различные методы jQuery и назначать обработчики событий, а не только задавать CSS-свойства.

CSS-селекторы в jQuery фактически опираются на правила записи селекторов в CSS, поэтому, если вы хорошо разбираетесь в CSS, то обращение к элементам для вас не будет представлять труда.

В таблице 1 приложения вы найдете различные CSS-селекторы, используемые в jQuery.

**Практическое задание:** Необходимо сделать так, чтобы год в подвале страницы всегда соответствовал текущему. В разметке год находится в пустом элементе `<span id="date-now"></span>`:

```
<p>Books Online Library | <span id="date-now"></span>
| All rights reserved.</p>
```

Мы обратимся к этому элементу с помощью CSS-селектора и установим текст в нем, исходя из текущей даты и года, который можем получить с помощью метода `getFullYear()` объекта Date:

```
$( '#date-now' ).text( new Date().getFullYear() );
```

Метод `text("some text")` в этой строке дает нам возможность записать любой текст в указанном в селекторе элементе. Если вы используете метод `text()` без параметров, то получите тот текст, который в данный момент записан внутри элемента (аналог JS свойства `textContent`).



Рисунок 5

## jQuery селекторы

jQuery-селекторы записываются так:

```
$( ':селектор' )
```

Например, `$("section:even").css("background", "gray")` позволит вам залить все четные элементы `<section>` серым цветом фона. Селектор `$(":header")` даст возможность управлять всеми заголовками на странице, вне зависимости от того, какой это тег — `h1`, `h2`, `h3` или `h4`. Селектор `$(".block:hidden")` позволит выбрать элементы с классом `.block`, которые были скрыты с помощью CSS или JavaScript (то есть имеют свойство `display: none`, заданное для класса или для отдельного элемента, или атрибут `hidden`, или `type=hidden`).

Как правило jQuery-селекторы позволяют найти либо дочерние элементы по определенному признаку, либо элементы с определенными характеристиками (скрытые, анимированные и т. п.), либо определенные элементы форм. Их варианты представлены в 2-х таблицах приложения.

## Селекторы форм

Эти селекторы используются для управления элементами форм: полями ввода, переключателями, флагками, выпадающими списками и кнопками.

Предположим, что вам нужно сделать недоступной кнопку в форме, пока пользователь не согласится с условиями предоставления услуг. Для этого подойдет строка

```
$('.button').prop('disabled', true);
```

В случае, если нужно, наоборот, эту кнопку сделать доступной, сработает селектор

```
button:disabled').prop('disabled', false);
```

Если поле с паролем нужно выделить рамкой (в случае неправильного заполнения, например), подойдет такой селектор:

```
':password').css('border', '2px solid red');
```

Множество других селекторов с примерами вы найдете в таблице 3 приложения.

## Особенности селекторов в jQuery

Особенностью использования селекторов в jQuery является то, что функция jQuery, представленная обычно в виде \$ всегда возвращает объект, а не ошибку, если вы указали селектор неверно или таких селекторов на странице нет. Это имеет свои плюсы и минусы. К плюсам относится то, что код, написанный на jQuery выполнится и тогда, когда в обычном JavaScript будет ошибка типа «Cannot set property 'onclick' of null», и выполнение дальнейшего кода будет невозможно из-за нее. Минусом является то, что отследить, что вы сделали неверно, будет сложно, т.к. ошибок в консоли браузера нет, но и код тоже не работает.

Поэтому при использовании любого селектора вы можете проверить, есть ли такой объект, используя свойство `length`, которое всегда присутствует в объекте jQuery, и может быть выведено или отображено в консоли. Если `length` равна `0`, то элемента, соответствующего вашему селектору, в DOM-дереве страницы нет.

Вид элементов, доступных на странице, выведенных в `console.log()`.

```
S.fn.init(4) [div.column.bg-blue, div.column.bg-blue,
▼ div.column.bg-blue, div.column.bg-blue, prevObject: S.fn.init
(1) [ i
  ▶ 0: div.column.bg-blue
  ▶ 1: div.column.bg-blue
  ▶ 2: div.column.bg-blue
  ▶ 3: div.column.bg-blue
  length: 4
  ▶ prevObject: S.fn.init [document]
  ▶ __proto__: Object(0)
```

Рисунок 6

Вид элементов, не существующих на странице, выведенных в `console.log()`.

```
▼ S.fn.init(0) [ i
  length: 0
  ▶ prevObject: S.fn.init [document]
  ▶ __proto__: Object(0)
```

Рисунок 7

## Методы управления стилями jQuery

Наверняка вы уже управляли стилями элементов на странице с помощью JavaScript. jQuery позволяет также

это сделать с помощью нескольких методов, которые либо добавляют атрибут `style` к указанному селектору (метод `css()`), или управляют добавлением/удалением атрибута `class` или значений внутри него.

Например, код `$('#block').css('color', 'red')` установит красный цвет текста для элемента с `id="block"` аналогично такой записи:

```
<div id="block" style="color: red">
    Lorem, ipsum dolor
</div>
```

Если необходимо задать несколько стилевых свойств, то синтаксис несколько меняется:

```
$('.example').css({
    color: 'red',
    'background-color': 'pink',
    fontSize: '18px'});
```

В этом случае вне зависимости от того, в каком формате — CSS или JS — были заданы свойства, они будут записаны в виде атрибута `style` сразу в нескольких элементах с классом `example`.

```
<div class="example" style="color: red;
    background-color: pink; font-size: 18px">
    Lorem ipsum dolor sit.
</div>
<div class="example" style="color: red;
    background-color: pink; font-size: 18px">
    Dolores numquam architecto velit.
</div>
```

```
<div class="example" style="color: red;  
background-color: pink; font-size: 18px">  
    Tempora excepturi tempore corporis.  
</div>
```

Когда вы работаете с классами, то обычно приходится добавлять или удалять какой-либо класс для одного или нескольких элементов. В примере ниже мы сначала удаляем класс '*active*' для ссылки, которая его имела, а затем назначаем для ссылок, которые находятся внутри *li*-элементов, расположенных в элементе с классом *.menu*.

```
$( 'a.active' ).removeClass('active');  
$( '.menu li a' ).addClass('active');
```

Методы для управления стилями и классами в jQuery вы найдете в таблице 4 приложения к данному уроку.

## Traversing. Методы обхода DOM

Селекторы в jQuery позволяют обратиться к одному или сразу к нескольким элементам. Причем после этого можно без дополнительного обхода всех элементов в цикле сразу назначать обработчики событий или вызывать нужные методы. Однако в процессе работы с элементами, помимо селекторов, вам понадобится обратиться ко вложенным или родительским элементам селектора, чтобы выполнить ряд действий. Для этого в jQuery существует ряд методов для обхода DOM-элементов (*traversing*). Они представлены в таблице 5 приложения.

В качестве примеров мы будем использовать поиск родительских элементов при клике на кнопке, закрывающей всплывающее окно. В этом случае мы используем метод `closest()`:

```
$('.popup-close').closest('.popup-wrapper') .  
fadeOut(400);
```

Для управления скрытым разделом с рекламой самой читаемой книги мы используем метод `parents(selector)`, который ищет указанный селектор среди всех родительских селекторов от текущего.

```
let section = $(this).parents('section');
```

`$(this)` в этом коде — это замена `this` в нативном JavaScript.

При клике на одном из пунктов меню, нам необходимо будет найти в родительском элементе ссылку, у которой есть класс `active` и удалить этот класс. Обращение к родительскому элементу с классом `menu` будет выполнено также с помощью метода `.closest('.menu')`, а поиск нужного элемента — с помощью метода `.find('.active')`. Оба метода принимают в качестве параметра селектор:

```
$(this).closest('.menu') .find('.active')  
.removeClass('active');
```

Когда мы будем изменять видимость элемента, следующего за активным заголовком, для доступа к этому элементу мы используем метод `next()`:

```
activeHeader.next() .slideUp(200);
```

Методы обхода DOM, кроме того, позволяют оберачивать элементы в другие элементы, перебирать элементы набора, назначая индивидуальные свойства, или обращаться к первому/последнему элементу в наборе.

При разборе примера вы еще увидите их использование.

## Обработка событий в jQuery

Как правило, изменения в стилях/классах для html-элементов необходимо назначать при обработке каких-либо событий. Способы обработки событий в jQuery похожи на известные вам из JavaScript, но имеют укороченный синтаксис.

Для того чтобы назначить обработчик события, используйте метод **on**:

```
$('селектор').on('событие', function() {  
    // ваш код  
});
```

Данный синтаксис предполагает, что в качестве события вы указываете любое из известных вам без приставки '**on**', как для метода **addEventListener()**, например:

```
$(‘a.active’).on(‘click’, function() {  
    $(this).removeClass(‘active’);  
})
```

Вместо ключевого слова **this**, которое в нативном JS указывало на объект, для которого назначено событие,

в jQuery используется аналог — `$(this)`, указывающий на jQuery-объект, соответствующий селектору, но при этом именно тот из набора, по которому был выполнен клик или для которого наступило указанное событие.

Например, для того чтобы получить высоту спрятанного элемента с рекламой наиболее читаемой книги, мы будем обрабатывать событие `resize` для объекта `window` (`$(window)` в jQuery).

```
let sectionMore = $('#see-more'),
    topCoord = $(window).height() -
                sectionMore.outerHeight();

$(window).on('resize', function(){
    topCoord = $(window).height() -
                sectionMore.outerHeight();
    if (!sectionMore.hasClass('closed')) {
        sectionMore.css('top', topCoord);
    }
})
```

Дело в том, что нужный нам элемент — это

```
<section id="see-more" class="closed">
```

Этот раздел находится внизу экрана за счет фиксированного позиционирования и скрыт при загрузке страницы за счет координаты `top: 97vh`.

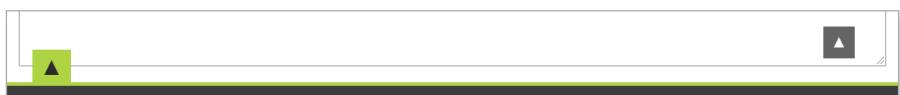


Рисунок 8

При клике на зеленой кнопке со стрелкой мы будем его показывать на всю высоту, которая меняется в зависимости от ширины экрана. За счет обработки события `resize` мы будем динамически получать высоту этого элемента (включая `padding`) методом `sectionMore.outerHeight()` и вычитать ее из высоты окна браузера `$(window).height()`, а затем назначать результат вычислений для `section` в качестве значения css-свойства `top`, если у `section` отсутствует класс `close`, т. е. она находится в развернутом состоянии, как на скриншоте ниже. В этом кусочке кода нам понадобились и методы для управления css-стилями, и проверка на существования определенного класса для элемента `section`.

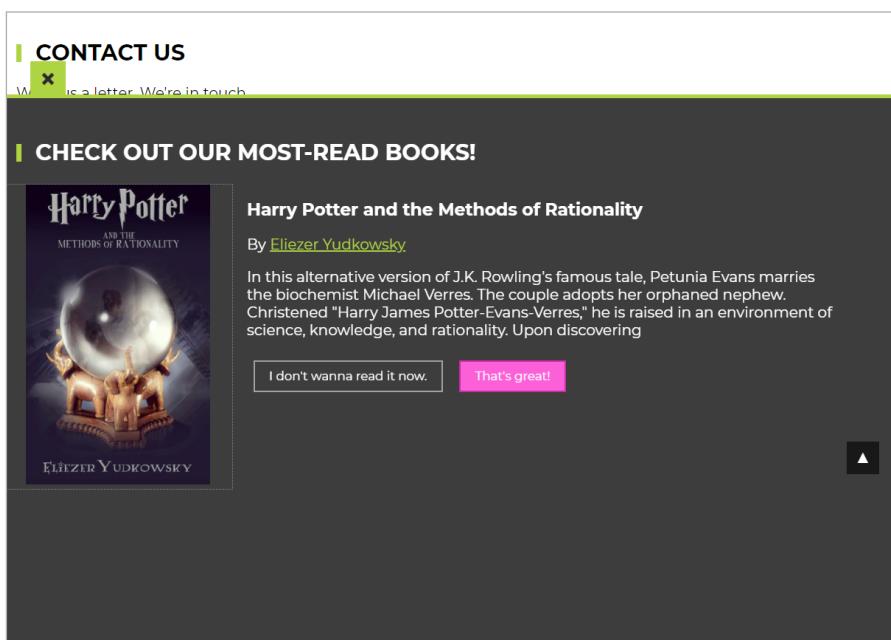


Рисунок 9

В jQuery есть возможность использовать краткую форму обработчика события. Вместо

```
$( 'селектор' ).on( 'mousedown', function() {     } );
```

вы можете написать

```
$( 'селектор' ).mousedown( function() {     } );
```

Естественно, вместо `mousedown` можно использовать события `mouseover`, `mouseout`, `click`, `keyup` или `input`, но без приставки `on`.

Например, для изменения внешнего вида верхнего меню мы можем записать такой код:

```
$('.menu li a').click(function(){
    $(this).closest('.menu').find('.active').
        removeClass('active');
    $(this).addClass('active');

})
```

Что здесь происходит? При загрузке страницы активным пунктом меню является **About Us**, о чем свидетельствует черная стрелка над текстом ссылки и наличие класса `active` в html-разметке.

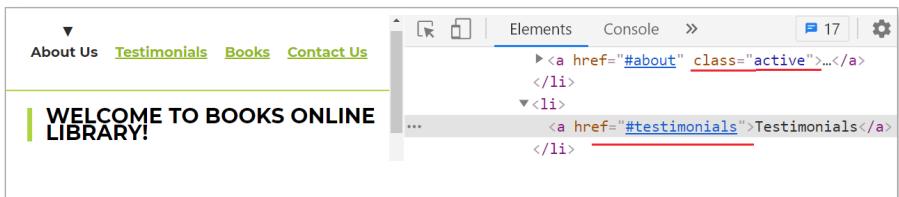


Рисунок 10

Но при клике на любом другом пункте меню, например, на **Testimonials**, черная стрелка переходит к этому пункту вместе с классом **active**, а для **About Us** атрибут **class** становится пустым.

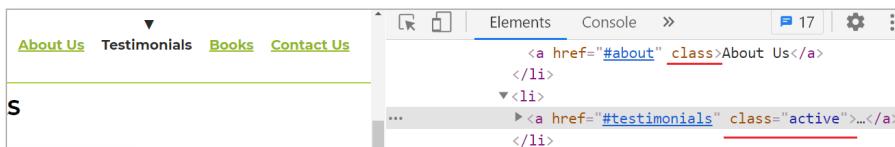


Рисунок 11

В этом коде еще представляет интерес строка

```
$(this).closest('.menu').find('.active').
    removeClass('active');
```

В ней мы отталкиваемся от того элемента, по которому был сделан клик (**\$(this)**), причем неважно, какой именно это пункт меню. Затем обращаемся к ближайшему родительскому элементу с классом **.menu (closest('.menu'))**, находим в нем вложенный элемент с классом **.active (find('.active'))**, и только потом удаляем у найденного таким сложным способом элемента класс **.active**.

Зачем такие сложности? Почему бы сразу не написать

```
$('.active').removeClass('active');
```

Это можно было бы сделать, если бы у нас не было класса **.active** в других элементах. Однако такой класс присутствует в 2-х элементах в **<div class="accordion">**, поэтому мы можем такой строкой «поломать» работоспособность другого элемента.



Рисунок 12

Назначать обработчик события можно в виде именованной функции, описанной отдельно, таким образом:

```

$('.menu li a').on('click', toggleMenu);

$('.menu li a').click(toggleMenu);

```

Это имеет смысл делать тогда, когда такая функция вызывается для нескольких элементов в разных блоках кода или не только по событию, но и при загрузке страницы для осуществления какой-либо проверки.

Например, на странице онлайн-библиотеки функция `checkToTop()` проверяет положение полосы прокрутки документа в момент загрузки страницы и по событию `scroll` для `window`, отображая ссылку «To top» только в том случае, если прокрученено более `100px` страницы вниз.

```

function checkToTop() {
  $('#totop').css({
    bottom: ($(window).scrollTop() > 100 ? '7%' : '-100px')
  });
}

checkToTop();
$(window).scroll(checkToTop);

```

Разница показана на рисунке 13.

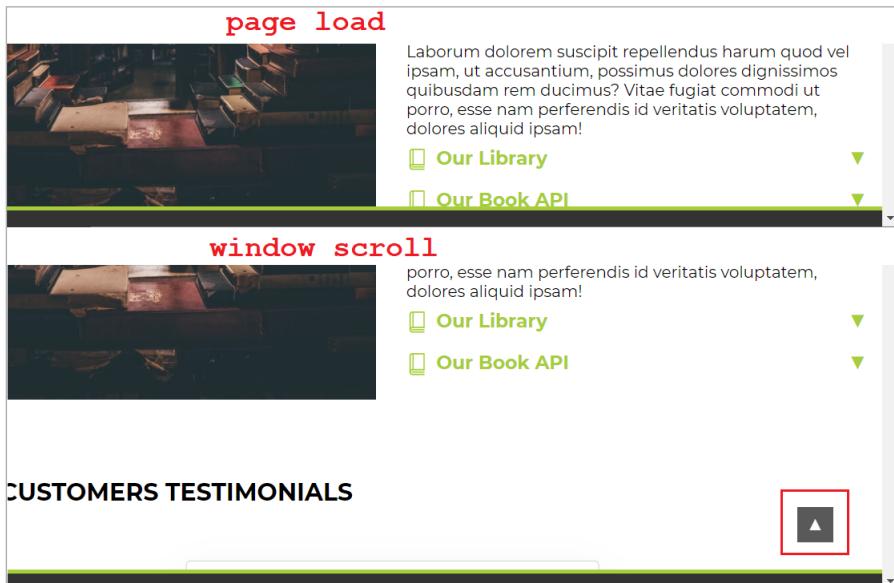


Рисунок 13

## Назначение одного обработчика для нескольких событий

Метод `on()` достаточно универсален, и позволяет назначить одну функцию для обработки сразу нескольких событий. Например, при вводе символов в поле ввода можно сразу обработать такие события:

```
$( 'input:text' ).on( 'keydown input paste', function() {
    console.log( $(this) .val() );
});
```

Попробуйте набрать текст в первом поле ввода в разделе [Contact Us](#). Вы увидите лесенку из символов при

ручном наборе и добавление слов(а) при вставке скопированного текста.

`$(this).val()` — это значение `value` для поля ввода.

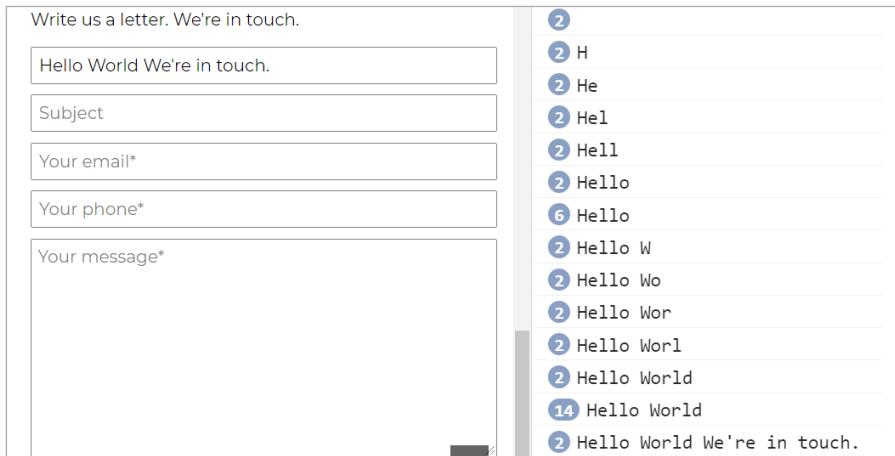


Рисунок 14

## Анимационные методы jQuery

В свое время jQuery стала очень популярной не в последнюю очередь из-за того, что позволяла простым способом задавать анимационные эффекты для различных элементов, причем еще в то время, когда CSS-анимаций с помощью свойств `transition` и `@keyframes` не существовало.

В jQuery анимационные эффекты задаются с помощью нескольких методов. Поскольку для любой анимации необходимо указывать время, то в качестве параметров этих методов используются числа в миллисекундах (`$('.block').hide(500)`) или ключевые слова `'slow'`, `'normal'`, `'fast'`.

К анимационным методам jQuery относятся:

*Методы, изменяющие высоту/ширину элемента и его видимость (display)*

- `show([time])/hide([time])` — отображение/скрытие элементов с изменением их высоты и ширины во времени. В конце анимации элементу, для которого она вызывалась, добавляется стилевое свойство `display: block` или `display: none`. Параметр `time` является необязательным и позволяет указать время в миллисекундах, в течение которого будет выполняться анимация. Если этот параметр отсутствует, то анимация не выполняется. Происходит только отображение/скрытие элемента.
- `toggle([time])` — метод-переключатель видимости/невидимости объекта. Проверяет, является ли элемент скрытым, и в этом случае показывает его, как метод `show()`. Если элемент отображается, то скрывает его аналогично методу `hide()`.
- `slideUp()/slideDown()` — анимационное скрытие/раскрытие элемента с изменением его высоты.
- `slideToggle()` — метод-переключатель, который скрывает открытый элемент и показывает скрытый элемент с анимационным изменением его высоты. В конце анимации элементу добавляется стилевое свойство `display: block` или `display: none`.

Часть вышеперечисленных методов мы используем для создания аккордеона — элемента, в котором при клике на заголовке отображается спрятанный ниже его текст.

```
$('.accordion-header').on('click', function () {
    let activeHeader = $(this).parent().
        find('.accordion-header.active');
    if (!$(this).hasClass('active')) {
        activeHeader.next().slideUp(200);
        activeHeader.removeClass('active');
        $(this).addClass('active').next().slideToggle(400);
    }
});
```

Особенностью данного аккордеона является то, что при клике на заголовке меняются стили псевдоэлементов `::before` и `::after` (см. рисунок 15), т. к. к заголовку добавляется класс `active`.

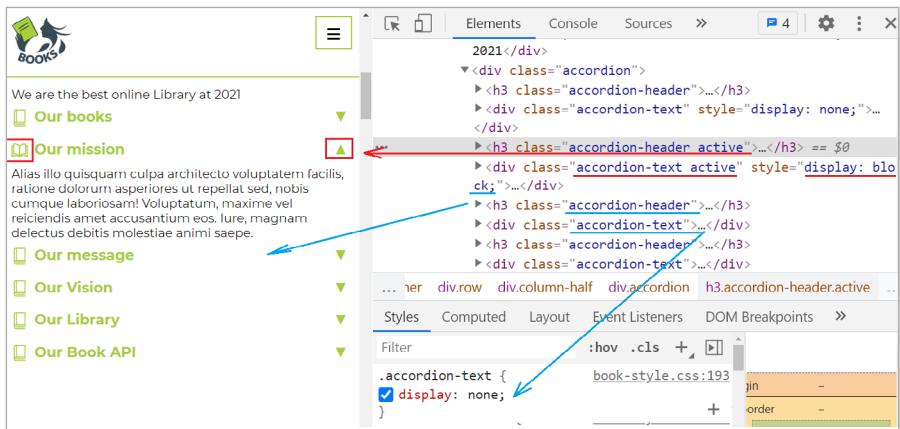


Рисунок 15

Кроме того, при разворачивании одного блока с текстом, открытый ранее блок должен свернуться, так что класс `active` может быть назначен только одному заголовку. Поэтому в коде мы ищем активный заголовок,

используя строку `$(this).parent().find('.accordion-header.active')`, которая выполняет поиск селектора `.accordion-header.active` методом `find()` в родительском по отношению к текущему элементе, которым является `<div class="accordion">`.

Переключать видимость элементов мы будем только для тех, которые на данный момент спрятаны — для этого нам нужна строка `!$(this).hasClass('active')`. Причем нам нужно сначала свернуть раскрытый блок текста, а затем показать новый.

Поскольку тот элемент, который нам нужно скрыть, находится после активного на данный момент заголовка, с точки зрения DOM-модели по отношению к заголовку он будет следующим узлом-элементом, т.е. `nextElementSibling`. С точки зрения jQuery обратиться к следующему узлу можно с помощью метода обхода DOM `next()`:

```
activeHeader.next().slideUp(200);
```

Для разворачивания нового блока текста после другого заголовка, на котором пользователь кликнул, мы используем строку

```
$(this).addClass('active').next().slideToggle(400);
```

За счет того, что при вызове различных методов jQuery возвращается jQuery-объект, мы можем последовательно вызвать метод, добавляющий класс `active` к заголовку, по которому щелкнули, а затем обратиться к следующему за ним элементу и раскрыть его анимационным методом `slideToggle()`.

## Методы, изменяющие непрозрачность элемента (свойство opacity)

- **fadeIn(time)** — изменяет значение свойства **opacity** от 0 до 1, как бы «проявляя» элемент;
- **fadeOut(time)** — изменяет значение свойства **opacity** от 1 до 0, постепенно скрывая элемент;
- **fadeTo(time, opacityValue)** — изменяет значение свойства **opacity** до требуемого значения (от 0 до 1).

Например, с помощью метода **fadeIn()** через 1,5 секунды после загрузки страницы появляется окно с предложением оформить подписку на сервис.

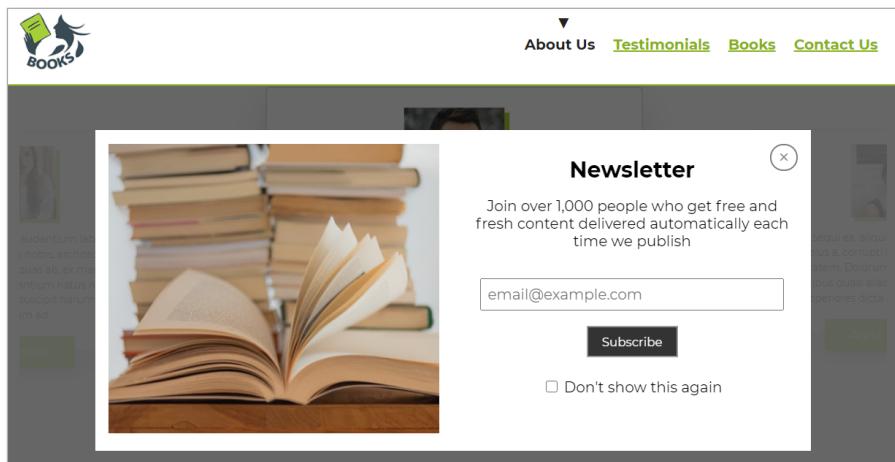


Рисунок 16

Код здесь таков:

```
setTimeout(function () {
    $('#newsletter').fadeIn(500);
}, 1500);
```

Как вы видите, мы вполне успешно сочетаем здесь нативный JavaScript в виде вызова метода `setTimeout()` и анимацию на jQuery.

Для того чтобы закрыть это всплывающее окно, мы должны обработать клик по элементу с классом `.popup-close` и использовать метод `fadeOut()`, который скроет не только окно, но и фон, закрывающий текст страницы.

```
$('.popup-close').click(function () {
  $(this).closest('.popup-wrapper').fadeOut(400);
});
```

За фон отвечает элемент с классом `.popup-wrapper`, который является родителем родительского для `.popup-close` элемента. Поэтому, для того чтобы «добраться» до него, мы используем метод `closest(selector)`, который как раз и возвращает ближайший родительский элемент или текущий элемент с указанным классом.

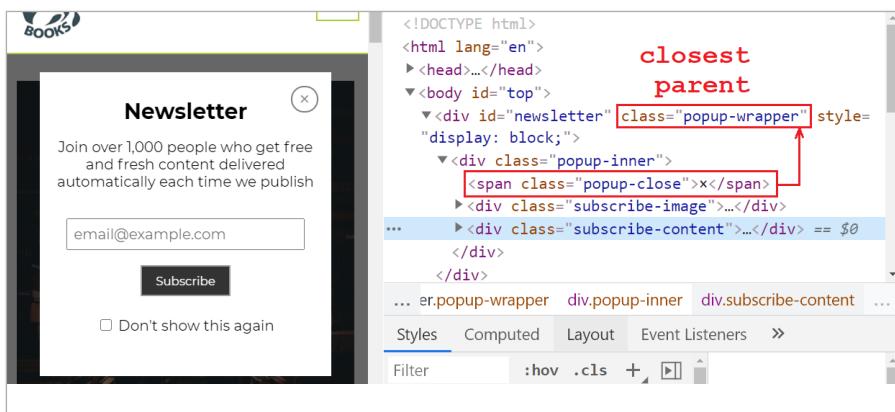


Рисунок 17

## Метод `animate()` для создания произвольных эффектов

- `animate({prop1: value, prop2: value}, time)` — позволяет указать объект, содержащий свойства и значения любых свойств, которые можно изменить количественно, то есть `border-radius`, `width`, `height`, `left`, `top`, `padding`, `margin` и т. п. и количество времени на анимацию. Вы не можете с помощью этого метода анимировать такие свойства, как `visibility` или `display`, а также `background-color` или `color`. Для анимации последних существует плагин [jQuery.Color](#).

Рассмотрим примеры с нашей страницы, которые используют метод `animate()` при обработке событий. Например, клик по зеленой кнопке-стрелке внизу экрана отобразит нам блок с рекламой самой читаемой книги:

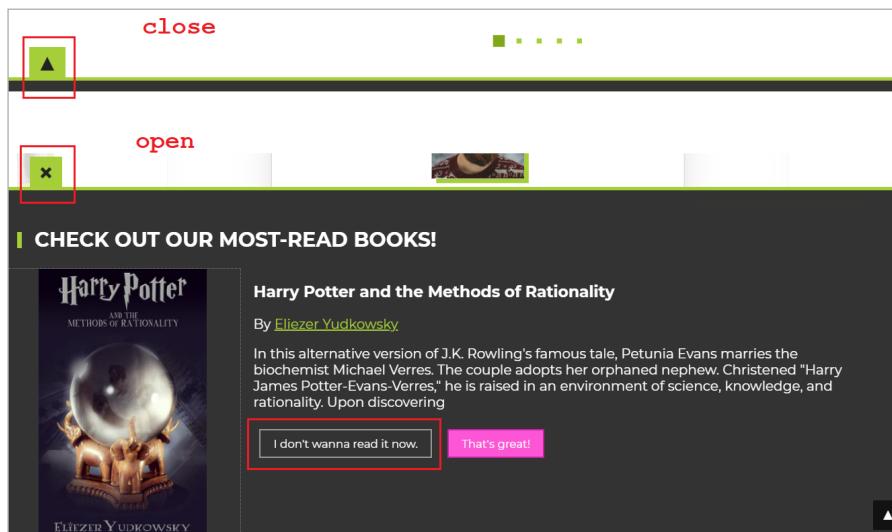


Рисунок 18

Эта кнопка имеет разный вид, когда раздел страницы с ней скрыт и полностью виден. Кроме того, скрывать его мы будем еще и при клике на кнопке с `id="not-now"` и текстом *I don't wanna read it now*. Поэтому обработчик события запишем для 2-х селекторов сразу, а в функции определим переменную `section` для обращения к родительскому элементу с соответствующим селектором тега, а переменную `target` — для ссылки на зеленую кнопку. Метод `animate()` нам необходим для того, чтобы красиво передвинуть верх фиксированной `section` в рассчитанную ранее координату или опустить ее в самый низ. Кроме того, для изменения вида зеленой кнопки мы используем метод `jQuery html()`, передавая в него спецсимволы, обозначающие стрелку или крестик в зависимости от того, открыта или нет наша `section`. За это отвечает наличие / отсутствие для раздела с рекламой класса `closed`, который добавляется / удаляется в коде строкой `section.toggleClass('closed')`.

```
$('.close-green, #not-now').click(function () {
    let section = $(this).parents('section');
    let target = $('.close-green');
    if (section.hasClass('closed')) {
        section.animate({
            top: topCoord
        }, 400);
        target.html('&times;');
    } else {
        section.animate({
            top: '97vh'
        }, 400);
    }
});
```

```

        target.html('&#9650;');
    }

    section.toggleClass('closed');
}) ;

```

Еще один вариант использования метода `animate()` предполагает анимированную прокрутку до нужного раздела страницы или на самый верх при клике на ссылке с хешем (#):

```

 $('[href^="#"]').on('click', function (event) {
  if ($(this).attr('hash') !== "") {
    event.preventDefault();
    let hash = $(this).prop('hash');
    $('html, body').animate({
      scrollTop: $(hash).offset().top - 60
    }, 800, function () {});
  }
}) ;

```

Здесь мы анимируем для `html` и `body` такое свойство, как `scrollTop` элемента, получая его текущее положение элемента относительно верха страницы.

## Делегирование событий

Делегирование событий подразумевает, что обработчик события назначается для родительского элемента, а само событие обрабатывается для вложенных (дочерних) элементов. Этот вариант бывает необходим тогда, когда вложенных элементов очень много или эти элементы были добавлены в процессе работы JS или jQuery-кода.

Рассмотрим пример, в котором при наведении на элементы списка мы должны получить в блоке, расположенному справа от них, какую-то английскую поговорку. Текст в элементах списка нам дает только примерное представление о сути поговорки. Сама поговорка скрыта в атрибуте `data-text` внутри элемента `<li>`:

```
<ul id="proverbs">
  <li data-text="When in Rome, do as the Romans.">
    How to behave...
  </li>

  <li data-text="Hope for the best,
    but prepare for the worst.">
    About hope
  </li>

  <li data-text="Fortune favors the bold.">
    About fortune
  </li>

  <li data-text="Birds of a feather flock together.">
    About people
  </li>

  <li data-text="Keep your friends close
    and your enemies closer.">
    About friends and enemies
  </li>
</ul>
```

Мы будем обрабатывать 2 события — `mouseenter` и `mouseleave` для родительского элемента `<ul>`, но при этом получать информацию о поговорках будем из элементов `<li>`, используя свойство `event.target` (селектор `$(event.target)` в jQuery) объекта `event`, который всегда

передается в качестве параметра в функцию-обработчик события. Сделаем мы это потому, что используем в этом скрипте синтаксис стрелочных функций из стандарта ES6(ECmaScript2015).

```
$(()=>{
    const output = $('#output');
    $('ul#proverbs').on('mouseenter', 'li', (event)=>{
        console.log($(event.target));
        output.text($(event.target).data('text'));
    }).on('mouseleave', 'li', (event)=>{
        output.text('');
    });
})
```

Код предполагает, что при обработке события **mouseenter** мы выводим в пустом `<div id="output">` поговорку из `<li>`, используя метод `data('text')` для доступа к атрибуту `data-text`, а при обработке события **mouseleave** убираем текст, помещая в `#output` пустую строку.

В строке `console.log($(event.target));` выводится информация о том, что мы действительно обращаемся к элементу `<li>`. Пример вы можете найти в папке *examples* в файле *delegate.html*.

**English proverbs**

- How to behave...
- About hope
- About fortune
- About people
- About friends and enemies

When in Rome, do as the Romans.

Рисунок 19

В jQuery делегирование событий осуществляется методом `on()`. Общий вид таков:

```
$('родительский селектор').  
  on('событие', 'селектор дочерних элементов',  
    function() { ... });
```

Например, для элементов типа `<button class = "btn-plus">`, которые могут добавляться в форму путем клонации обработка события клика будет такой:

```
$('form').on('click', '.btn-plus', function() {  
  console.log('button clicked');  
});
```

В нашем примере такое действие происходит для открытия всплывающих окон. Таких в разметке страницы 2: для подписки на новости

```
<div id="newsletter" class="popup-wrapper">  
  ...  
</div>
```

и для вывода сообщения при заполнении формы контактов

```
<div id="thanks-contact" class="popup-wrapper">  
  ...  
</div>
```

Всплывающее окно с подпиской выводится либо через 1,5 секунды после загрузки страницы (мы разбирали этот пункт выше), либо при клике на кнопку в элементе `<section id="subscribe">` (рис. 20).



*Рисунок 20*

Код кнопки:

```
<button class="btn btn-info btn-large"
        data-popup="#newsletter">
    Subscribe Now
</button>
```

Как видно, у этой кнопки есть **data-атрибут `data-popup="#newsletter"`**, который мы и примем за основу при назначении обработчика события в виде функции `popupShow()`;

```
$( '[data-popup]' ).on('click', popupShow);

function popupShow(evt) {
    evt.preventDefault();
    let id = $($this).data('popup');
    id.fadeIn(600);
}
```

В функции мы получаем **id** элемента, указанного в атрибуте **data-popup** и с помощью анимационного метода `fadeIn(600)` показываем соответствующее всплывающее окно:

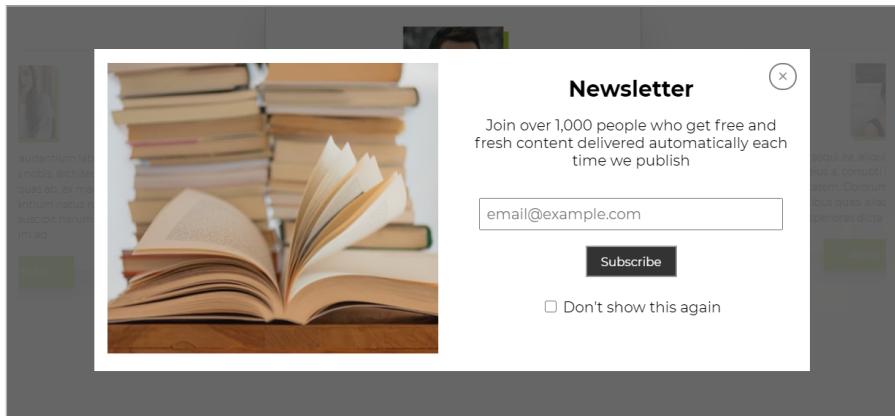


Рисунок 21

Где же здесь делегирование события? Его действительно пока нет. Дело в том, что второе всплывающее окно мы вызываем после проверки правильности заполнения полей формы, причем атрибут `data-popup="#thanks-contact"` добавляется в скрипте после проверки всех полей (см. описание процесса ниже). А это говорит о том, что на момент загрузки страницы кнопка с кодом

```
<button type="submit" id="contact-btn"
        class="btn btn-dark">
    Send a letter
</button>
```

не имела назначенного обработчика события, т. к. в ее разметке отсутствует нужный `data-атрибут`, поэтому функция `popupShow()`, как обработчик события, просто не будет вызвана.

Как раз для этой ситуации и нужно воспользоваться делегированием событий. Мы назначим обработчик со-

бытия для элемента `body`, однако отслеживать его будем только для элементов с атрибутом `data-popup`. Записать это нужно таким образом:

```
$( 'body' ).on( 'click', '[data-popup]', popupShow );
```

вместо строки

```
// $('[data-popup]').on('click', popupShow);
```

Пользователь, заполнивший поля формы и нажавший на кнопку с текстом «Send a letter», увидит примерно такое сообщение:

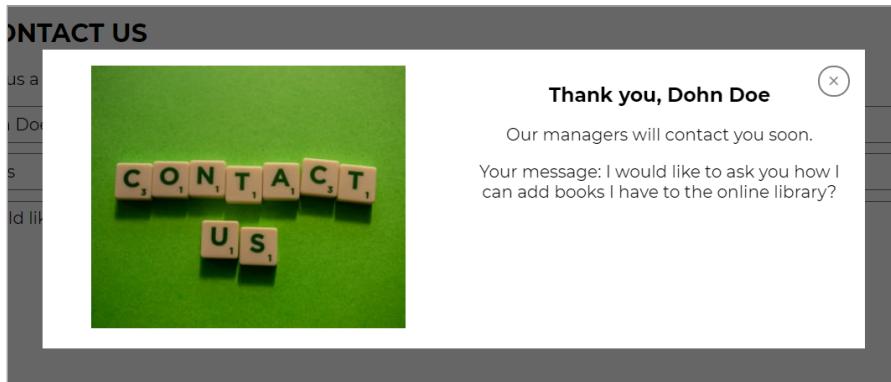


Рисунок 22

## Отмена обработчиков событий

В том случае, когда нам в какой-то момент нужно, чтобы обработчик события был удален, применяется метод `off()`. При использовании этого метода важно, чтобы при назначении обработчика события использовалась та же функция, что и для его отмены, как для методов

`addEventListener()` и `removeEventListener()` в нативном JavaScript. Общий вид записи:

```
// назначаем обработчик события  
$( 'селектор' ).on( 'событие', someFunc );  
  
// отменяем обработчик события  
$( 'селектор' ).off( 'событие', someFunc );
```

Например, такой подход мы используем для маленького рекламного блока в `header` нашей страницы. По центру вы можете видеть зеленый восклицательный знак — это кнопка с `id="start-stop"`. Рядом с ней в разметке расположен `<span id="text-change"></span>`, куда при нажатии на эту кнопку будет выводится несколько строк из массива с помощью jQuery. Внешний вид кнопки будет меняться при этом на крестик, и повторное нажатие на кнопку приведет к остановке смены текста и его удалению из `header-a`. То есть кнопка будет переключателем работы 2-х функций.



Рисунок 23

В коде это будет выглядеть так:

```
let textAnim = ['Read our books!', 'Subscribe to news',
    'Learn about new books',
    'Add books to our library'],
    textId=null, textIndex=0;
$("#start-stop").on('click', showAnimatedText);

function showAnimatedText(){
    $(this).next().addClass('fromTopToDown');
    textId = setInterval(function(){
        $('#text-change').text(textAnim[textIndex]);
        textIndex == textAnim.length-1 ?
            textIndex = 0: textIndex++;
    }, 2000);
    $("#start-stop").html('&times;');
    .off('click',showAnimatedText)
    .on('click',stopAnimatedText);
}

function stopAnimatedText(){
    $('#text-change').text('');
    $(this).next().removeClass('fromTopToDown');
    clearInterval(textId);
    $("#start-stop").text('!')
    .off('click',stopAnimatedText)
    .on('click', showAnimatedText);
}
```

Анимацию смены текста в данном случае мы реализуем за счет добавления/удаления к текстовому блоку класса `fromTopToDown` с такими CSS-свойствами:

```
.fromTopToDown {
    animation: fromTopToDown 2s ease-out infinite;
}
```

Метод `off()` также позволяет отменить события, назначенные с использованием делегирования события. В этом случае запись будет такая:

```
$( 'родительский селектор' ).off( 'событие' ,  
    'селектор дочерних элементов' , someFunc );
```

## Обработка данных форм

Для форм в арсенале jQuery есть свои селекторы (таблица 3 приложения). В основном, они помогают найти однотипные селекторы для различных вариантов полей `input` или `button` (`$('.submit')`, `$('.input')`), а также обращаться к выделенным флагкам или переключателям (`$('.checked')`) или выбранным пунктам выпадающих списков (`$('.option:selected')`).

Флажок (`checkbox`) встречается нам на странице 1 раз во всплывающем окне с предложением оформить подписку. Он имеет в коде `id="forgetMe"`. Клик по этому элементу предполагает, что при следующем заходе на страницу мы уже не увидим данное окно.

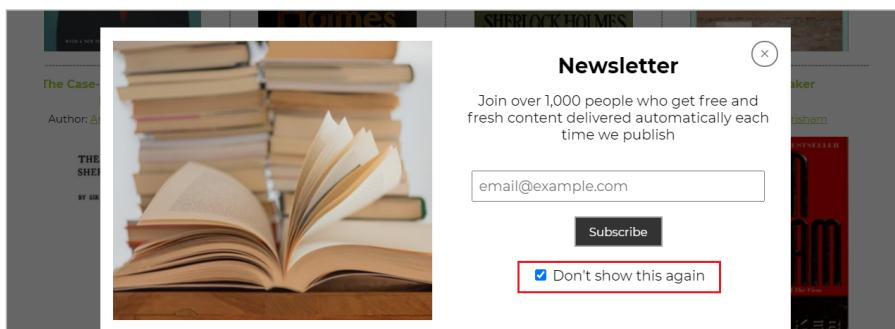


Рисунок 24

Поэтому обрабатываем для флажка событие `click`, сохраняя в `localStorage` браузера информацию о том, что пользователь выделил флажок. Объекте `localStorage` хорош тем, что он сохраняет нужные данные даже после того, как пользователь закрыл страницу в браузере, так что при следующем заходе на это сайт код на JavaScript (jQuery) может проверить эти данные и отображать информацию для этого конкретного пользователя, основываясь на его предпочтениях.

Для этого действия нам понадобится такое свойство флажка, как `checked`, которое при обработке события мы можем получить строкой `$(this).prop('checked')`. Если оно возвращает `true`, то в `localStorage` в виде значения '`newsletter`' мы помещаем `1`, если же пользователь снял такую отметку и свойство стало равно `false`, то сохраняем в `newsletter 0`.

```
$('#forgetMe').click(function () {
  console.log($(this).prop('checked'));
  localStorage.setItem('newsletter',
    $(this).prop('checked') ? 1 : 0);
});
```

Кроме того, мы теперь несколько изменим код появления самого всплывающего окна: перед вызовом `setTimeout()` сначала идет проверка на наличие в `localStorage` данных с именем `newsletter`, и только в случае, если таких данных там нет или там находится `0`, окно будет показано через `1,5` секунды после загрузки страницы.

```
let isNewsletterPopup = +localStorage.getItem('newsletter');
```

```
if (!isNewsletterPopup) {  
    setTimeout(function () {  
        $('#newsletter').fadeIn(500);  
    }, 1500);  
}
```

В нижней части страницы у нас есть контактная форма. Для нее мы будем обрабатывать событие `submit`. Однако, до того как пользователь будет отправлять форму, мы используем плагин [jquery.maskedinput](#) для того, чтобы пользователь вводил в поле для телефона только цифры и в определенной последовательности:

```
let userPhone = $('#user-phone');  
userPhone.mask("(999) 999-9999");
```

Чтобы плагин сработал, вам необходимо его подключить после jQuery, но до вашего файла с основным скриптом для страницы.

```
<script src="https://ajax.googleapis.com/ajax/libs/  
          jquery/3.5.1/jquery.min.js">  
</script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/  
          jquery.maskedinput/1.4.1/jquery.  
          maskedinput.min.js">  
</script>  
<script src="js/index.js"></script>
```

Этот код приведет к тому, что при попадании фокуса в поле с текстом «Your phone» пользователь увидит подсказку в виде подчеркиваний в месте расположения цифр.

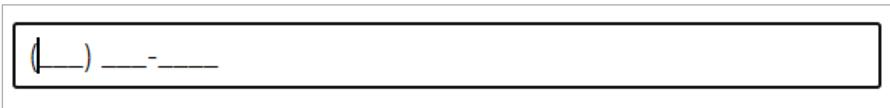


Рисунок 25

Как вы видите, использование этого плагина очень простое. Больше вариантов оформления маски вы найдете на [странице плагина](#).

Для остальных полей перед отправкой формы мы выполним элементарную проверку на соответствие определенному количеству символов. Если символов недостаточно, мы будем подсвечивать красной рамкой соответствующее поле ввода (за это отвечает класс `invalid` в CSS-стилях). Если же пользователь исправил свои ошибки, то этот класс мы будем убирать.

Основой для проверки служит вызов метода `.val()`, который в jQuery служит заменой свойству `value` для элементов форм в нативном JavaScript.

```
$('#contact-form').submit(function (e) {
    e.preventDefault();
    let userName = $('#user-name'),
        userEmail = $('#user-email'),
        userMessage = $('#user-message');

    // проверяем заполнение полей формы
    if (userName.val().length < 2) {
        userName.addClass('invalid');
        return false;
    } else userName.removeClass('invalid');

    if (userEmail.val().length < 7) {
        userEmail.addClass('invalid');
```

```
        return false;
    } else userEmail.removeClass('invalid');

if (userPhone.val().length < 7) {
    userPhone.addClass('invalid');
    return false;
} else userPhone.removeClass('invalid');

if (userMessage.val().length < 7) {
    userMessage.addClass('invalid');
    return false;
} else userMessage.removeClass('invalid');

// помещаем текст с сообщением в попур-окно
// и показываем его пользователю

$('#thanks-contact h3').text('Thank you,
    ${userName.val()}');
$('#thanks-contact .content').
    text('Your message: ${userMessage.val()}');
$('#contact-btn').attr('data-popup',
    "#thanks-contact");
$('#contact-btn').click();

})
```

После того, как пользователь все заполнил верно и щелкнул на кнопку «Send a letter», мы выводим его имя и текст сообщения в модальное окно с `id="thanks-contact"` с помощью метода `jQuery.text()`. Кроме того, добавляем `attr()` для кнопки `submit` атрибут `data-popup`, который отвечает за запуск всплывающего окна, со значением `«#thanks-contact»`, а потом еще и генерируем клик по этой кнопке, чтобы увидеть окно с текстом.

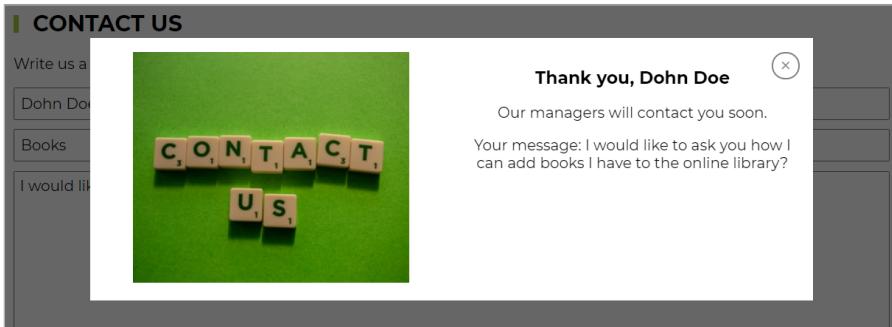


Рисунок 26



Unit 11.

## Введение в jQuery

© Компьютерная Академия «Шаг», [www.itstep.org](http://www.itstep.org)  
© Елена Слуцкая.

Все права на охраняемые авторским правом фото-, аудио- и видео-произведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объеме, оправданном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объем и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.