Библиотека jQuery позволяет легко манипулировать кодом HTML после его отображения в браузере. Она также предоставляет инструменты, которые помогут пользователю взаимодействовать с вашей страницей; инструменты, которые помогут вам создавать анимацию; и инструменты, которые позволяют вам общаться с сервером без перезагрузки страницы. Для начала рассмотрим некоторые основы jQuery, а также применим jQuery для его основной работы: получить какие-то элементы и что-то с ними сделать.

Библиотека jQuery предоставляет функцию jQuery, которая позволяет вам выбирать элементы с помощью селекторов CSS.

var listItems = jQuery( 'li' );

Конечно, если вы уже видели любой jQuery-код, то, вероятно, больше привыкли к чему-то вроде этого:

var listItems = $( 'li' );

Корректные имена в JavaScript могут быть достаточно произвольны, если только они не начинаются с цифры и не включают в себя дефис. Таким образом, $ в коде просто более короткое и удобное имя для функции jQuery. На деле в конце [исходного кода jQuery](https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.js) вы найдёте:

window.jQuery = window.$ = jQuery;

При вызове функции $() и передаче в неё селектора вы создаёте новый объект jQuery. Конечно, в JavaScript функции тоже объекты, это означает что у $ (и jQuery, естественно) также есть свойства и методы.

### $(document).ready()

Перед безопасным использованием jQuery для выполнения чего-либо на странице, вы должны убедиться, что страница находится в состоянии, когда она готова к манипуляциям. С jQuery мы должны поместить наш код в функцию, а затем эту функцию в $(document).ready(). Как вы можете видеть мы используем анонимную функцию.

$( document ).ready(function() {

console.log( 'готов!' );

});

Как только документ будет готов, выполнится функция, которую мы передали в .ready(). Что здесь происходит? Мы используем $(document) для создания jQuery-объекта document, а затем вызываем функцию .ready() для этого объекта, передавая ему функцию, которую мы хотим выполнить.

Поскольку такое часто где встречается, можете при желании использовать сокращённый метод — функция $() делает двойную работу как псевдоним для $(document).ready(), если вы передаёте функцию:

$(function() {

console.log( 'готов!' );

});

Дальше в данном руководстве будем подразумевать, что рассматриваемый код заключён в $(document).ready(function() { ... }), но для краткости эту часть не указываем.

### Получаем некоторые элементы

Самое простое, что мы можем делать с jQuery — это получить некоторые элементы и что-то ними выполнить. Если вы знакомы с селекторами CSS, получить произвольные элементы очень легко: вы просто передаёте соответствующий селектор в $().

$( '#header' ); // выбрать элемент с идентификатором header

$( 'li' ); // выбрать все пункты списков на странице

$( 'ul li' ); // выбрать все пункты маркированного списка

$( '.person' ); // выбрать все элементы с классом person

Важно понимать, что любая сделанная выборка будет содержать только те элементы, которые существовали на странице на момент выбора. Иными словами, если вы пишете var anchors = $( 'a' ), а затем позже добавите ещё один элемент <a> на свою страницу, то переменная anchors не будет содержать этот новый элемент.

### Другие способы создания объекта jQuery

Кроме передачи простого селектора в $(), вы можете создавать объекты jQuery и другими способами:

// создать jQuery-объект из элемента DOM

$( document.body.children[0] );

// создать jQuery-объект из списка элементов DOM

$( [ window, document ] );

// сделать выборку в контексте элемента DOM

var firstBodyChild = document.body.children[0];

$( 'li', firstBodyChild );

// сделать выборку из предыдущей

var paragraph = $( 'p' );

$( 'a', paragraph );

### Моя выборка содержит что-нибудь?

Иногда вам требуется что-то сделать, только когда вашему выбору соответствует несколько элементов. Поскольку функция $() всегда возвращает объект jQuery и он всегда истинный, вы должны проверить содержимое вашей выборки, чтобы определить, было ли что-то найдено.

**Внимание! Ошибочный код**

if ( $( '#nonexistent' ) ) {

// Неверно! Этот код будет выполняться всегда!

}

if ( $( '#nonexistent' ).length > 0 ) {

// Правильно! Этот код будет выполняться, только если на странице

// есть элемент с идентификатором nonexistent

}

Мы можем сократить проверку ещё больше, если вспомним, что 0 – это ложное значение:

if ( $( '#nonexistent' ).length ) {

// Этот код будет работать только, если имеется соответствующий элемент

}

### Получение отдельных элементов из выборки

Если вам нужно работать с исходным элементом DOM из выборки, то следует получить доступ к элементу из объекта jQuery. Например, если вы хотите напрямую получить доступ к свойству value элемента <input>, то должны работать с исходным элементом DOM.

var listItems = $( 'li' );

var rawListItem = listItems[0]; // или listItems.get( 0 )

var html = rawListItem.innerHTML;

Заметьте, что вы *не можете* вызвать методы jQuery для исходных элементов DOM. Поэтому следующий пример работать не будет:

var listItems = $( 'li' );

var rawListItem = listItems[0];

var html = rawListItem.html();

// Объект #<HTMLInputElement> не содержит метод html

Если вам требуется работать с одним элементом в выборке и вы хотите использовать методы jQuery для этого элемента, то можете получить новый объект jQuery, содержащий единственный элемент, через .eq(), передавая ему индекс элемента.

var listItems = $( 'li' );

var secondListItem = listItems.eq( 1 );

secondListItem.remove();

### Создание новых элементов

У функции $ есть также последняя роль: создание новых элементов. Если в $() передаётся фрагмент HTML, то это создаст новый элемент в памяти — иными словами, элемент будет создан, но не добавлен на страницу, пока вы этого не сделаете.

$( '<p>' ); // создание нового элемента <p> без содержимого

$( '<p>Привет!</p>' ); // создание нового элемента <p> с текстом

$( '<p class="greet">Привет!</p>' ); // создание нового элемента <p> с текстом и классом

Вы также можете создать элемент, передавая объект с информацией о его создании:

$( '<p>', {

html: 'Привет!',

'class': 'greet'

});

Обратите внимание, что мы должны указать свойство class в кавычках, поскольку class это зарезервированное слово в JavaScript и написание без кавычек вызовет ошибки в некоторых браузерах.

### Работа с выборкой

После создания объекта jQuery содержащего вашу выборку, вы, вероятно, хотите что-то с ней сделать. Но до этого рассмотрим несколько принципов, которые являются ключевыми для понимания того, что делает jQuery.

### Проверка выборки

Мы можем определить, соответствует ли выборка определённому критерию, с помощью метода .is(). Этот метод наиболее часто используется, чтобы предоставить селектор в качестве единственного аргумента. Он возвращает true или false в зависимости от того, соответствует ли выборка селектору:

$( 'li' ).eq( 0 ).is( '.special' ); // false

$( 'li' ).eq( 1 ).is( '.special' ); // true

Вы также можете передать в метод .is() объект jQuery, исходный элемент DOM или даже функцию, если вам нужна комплексная проверка.

### Получение, установка и неявный перебор

Есть много методов, которые вы можете вызвать после создания выборки. Эти методы, как правило, делятся на две категории: метод чтения (геттер) и устанавливающий метод (сеттер). Методы чтения извлекают фрагмент информации из выборки, а методы установки в некотором роде изменяют выборку. Практически во всех случаях методы чтения оперируют только первым элементом в выборке (.text() является известным исключением); методы установки же работают для всех элементов в выборке используя то, что известно как *неявный перебор*.

Неявный перебор означает, что jQuery автоматически перебирает все элементы в выборке, когда вы вызываете для выборки устанавливающий метод. Это означает, что если вы хотите сделать что-то для всех элементов в выборке, то не должны вызывать метод для каждого элемента в вашей выборке — просто вызываете метод для самой выборки и jQuery обработает элементы для вас.

Предположим, что вы хотите изменить HTML у всех пунктов списка на странице. Чтобы сделать это мы должны использовать метод .html(), который изменит HTML у всех выбранных пунктов списка.

$( 'li' ).html( 'Новый HTML' );

Вы также можете передать функцию в метод установки jQuery. Возвращаемое значение этой функции используется в качестве нового значения, и оно получает два аргумента: индекс элемента в выборке и старое значение, которое вы пытаетесь изменить. Это полезно, когда вам необходима информация о текущем состоянии элемента, чтобы правильно установить новое состояние.

$( 'li' ).html(function( index, oldHtml ) {

return oldHtml + '!!!'

});

### Явный перебор

Иногда задача, которую вы пытаетесь решить, не вписываются ни в один из существующих методов jQuery и тогда вы должны сделать явный перебор выборки с помощью метода .each(). В следующем коде в начало пункта списка мы добавляем тег <b> содержащий индекс элемента списка.

$( 'li' ).each(function( index, elem ) {

// this: текущий, исходный элемент DOM

// index: индекс текущего элемента в выборке

// elem: текущий, исходный элемент DOM (также как this)

$( elem ).prepend( '<b>' + index + ': </b>' );

});

Заметьте, что внутри функции, которую мы передаём в .each(), у нас есть два способа доступа к текущему исходному элементу DOM: через this и через elem. Как обсуждалось в разделе «Основы JavaScript», this — специальное ключевое слово в JavaScript, указатель на объект, который является текущим контекстом функции. В jQuery this почти всегда относится к исходному элементу DOM, с которым функция в данный момент работает. Так, в случае .each(), this относится к текущему элементу в наборе элементов которые мы перебираем.

### Цепочки

Одной из самых выгодных частей jQuery является способность «сцеплять» методы вместе. Это означает, что мы можем вызвать набор методов для выборки без её повторения или сохранения выборки в переменной. Мы даже можем сделать новые выборки на основе предыдущих и всё без разрыва цепочки.

$( 'li' )

.click(function() {

$( this ).addClass( 'clicked' );

})

.find( 'span' )

.attr( 'title', 'Наведи на меня' );

Цепочки возможны, потому что каждый устанавливающий метод в jQuery возвращает выборку, с которой он был вызван. Это чрезвычайно мощная особенность и многие библиотеки взяли её на вооружение. Тем не менее, она должна применяться с осторожностью. Большие цепочки могут сделать код *чрезвычайно* трудным для чтения, изменения и отладки. Нет жёстких правил, насколько длинной должна быть цепочка, но даже для простой цепочки выше, вероятно, следует провести рефакторинг для удобства чтения.

var listItems = $( 'li' );

var spans = listItems.find( 'span' );

listItems

.click(function() {

$( this ).addClass( 'clicked' );

});

spans.attr( 'title', 'Наведи на меня' );

### Обход и манипуляция

jQuery обеспечивает мощные инструменты для нахождения элемента или последующих элементов и работы с ними для получения желаемого результата. С помощью методов обхода и манипуляций можно браться за такие задачи, выполнение которых через родную манипуляцию DOM будет весьма болезненно. jQuery делает подобные задачи довольно простыми и интуитивно понятными.

Предварительно вы должны знать несколько важных терминов. Рассмотрим фрагмент HTML:

<ul>

<li>

<span>

<i>Foo</i>

</span>

</li>

<li>Bar</li>

</ul>

* Первый элемент списка является **дочерним** для маркированного списка.
* Маркированный список является **родителем** обоих элементов списка.
* <span> является **потомком** маркированного списка.
* Маркированный список является **предком** для всего внутри него.
* Два элемента списка являются **родственными**.

### Обход

jQuery позволяет нам делать «обход» или передвижение по HTML-элементам, которые составляют нашу страницу. Сперва мы делаем начальную выборку, а затем двигаемся через DOM относительно этой выборки. По мере продвижения через DOM мы меняем нашу оригинальную выборку; в некоторых случаях заменяем оригинальную выборку новой, в других случаях мы добавляем или убираем что-то из оригинальной выборки.

### Фильтрация выборки

Вы можете отфильтровать существующую выборку, чтобы она включала только те элементы, которые соответствуют заданному критерию. Например, вы можете фильтровать выборку следующими способами:

var listItems = $( 'li' );

// фильтровать выборку только для элементов с классом special

var special = listItems.filter( '.special' );

// фильтровать выборку только для элементов без класса special

var notSpecial = listItems.not( '.special' );

// фильтровать выборку только для элементов содержащих <span>

var hasSpans = listItems.has( 'span' );

Важно отметить, что .not() не является противоположностью .is(). Метод .is() возвращает логическое значение, в то время как метод .not() возвращает новый объект jQuery.

### Поиск элементов относительно выборки

Начальная выборка может служить «базой» для создания дополнительных выборок. К примеру, у вас может быть имеющаяся выборка, которая содержит отдельный пункт списка, а затем необходимо работать с его родственными элементами или родительским маркированным списком. Вы можете легко сделать новую выборку относительно существующей:

// Получить первый пункт списка на странице

var listItem = $( 'li' ).first(); // и .last()

// Получить родственные элементы этого пункта списка

var siblings = listItem.siblings();

// Получить следующего родственника этого пункта списка

var nextSibling = listItem.next(); // и .prev()

// Получить родителя этого пункта списка

var list = listItem.parent();

// Получить пункты списка являющихся непосредственными потомками list

var listItems = list.children();

// Получить ВСЕ пункты списка из list, включая вложенные

var allListItems = list.find( 'li' );

// Найти всех предков этого пункта списка с классом module

var modules = listItem.parents( '.module' );

// Найти ближайшего предка этого пункта списка с классом module

var module = listItem.closest( '.module' );

Вы также можете добавить что-то в существующую выборку с помощью метода .add(). Можно передать селектор, массив элементов, строку HTML или объект jQuery.

var list = $( '#my-unordered-list' );

// Сделать нечто со списком, а затем...

var listAndListItems = list.add( '#my-unordered-list li' );

### Возвращение к исходной выборке

Когда вы используете один из методов обхода для поиска некоторых элементов относительно исходной выборки, jQuery сохраняет на неё указатель для случая, если вы захотите вернуться к исходной выборке. Рассмотрим, например, ситуацию, когда вы выбираете маркированный список, вносите в его элементы некоторые изменения, а затем снова желаете работать со списком. Вы можете использовать метод jQuery .end(), чтобы вернуться к исходной выборке:

$( '#my-unordered-list' )

.find('li')

// Теперь мы работаем с элементами списка

.addClass('special')

.end()

// Теперь мы вернулись к работе со списком

.addClass('super-special');

Метод .end() позволят легко делать множество изменений в одном выражении. Такая практика мало что даёт для понимания кода, зато это коротко, словно рассказать историю, не переводя дыхания. Поэтому вы должны использовать этот метод редко. Чаще всего это приводит к коду, который трудно читать, поддерживать и отлаживать.

Лучшее решение может выглядеть следующим образом:

var list = $( '#my-unordered-list' );

var listItems = list.find('li');

listItems.addClass( 'special' );

list.addClass( 'super-special' );

jQuery также предлагает метод .addBack(), если вы хотите добавить свою исходную выборку в текущую. К примеру:

$( 'li.special' )

.siblings()

// Теперь мы работаем с родственниками исходной выборки

.removeClass( 'important' )

.addBack()

// Теперь мы работаем с исходным li И его родственниками

.addClass( 'urgent' );

Как и с .end() всё это может привести к коду, с которым тяжело разбираться. Метод .addBack(), несмотря на его пользу, может легко усложнить код. Вместо этого лучшим решением было бы использовать метод .add(), чтобы объединить две исходные выборки:

var specialListItems = $( 'li.special' );

var otherListItems = specialListItems.siblings();

otherListItems.removeClass( 'important' );

specialListItems.add( otherListItems ).addClass( 'urgent' );

### Манипуляция

Методы манипуляции jQuery позволяют вам менять DOM вашей страницы используя гораздо дружелюбный синтаксис, чем родные методы манипуляции DOM. Методы манипуляции возвращают объект jQuery, а значит вы можете соединять или объединять их с другими методами jQuery, которые рассматривались ранее.

### Изменение элементов

Есть множество способов для изменения элементов с помощью jQuery. Мы рассмотрим как решать некоторые наиболее распространённые задачи.

#### **Добавление и удаление классов**

Атрибут class может быть использован для добавления правил CSS, а также может быть полезен для выбора элементов через jQuery. Например, к элементу может быть добавлен класс hidden, с соответствующим правилом CSS, который определяет, что для этого класса установить display как none. Используя jQuery мы можем добавлять и удалять классы и тем самым влиять на отображение элементов.

$( 'li' ).addClass( 'hidden' );

$( 'li' ).eq( 1 ).removeClass( 'hidden' );

Если в вашем случае постоянно требуется добавлять и удалять класс, jQuery предлагает метод .toggleClass(). Следующий код добавляет класс hidden если его нет и удаляет его, если он присутствует.

$( 'li' ).eq( 1 ).toggleClass( 'hidden' );

#### **Изменение стиля**

При возможности каждый раз следует использовать классы в сочетании с правилами CSS, чтобы повлиять на оформление элементов, а jQuery применять только для добавления и удаления этих классов, как показано выше. В этом разделе мы увидим как изменить стиль элемента напрямую, но правила CSS в сочетании с классами для получения желаемого эффекта всегда предпочтительнее.

Если вы не можете получить результат через добавление и удаление классов, то jQuery предлагает метод .css(), который позволяет вам установить стиль элементов напрямую. Обычно это требуется, если вы хотите установить числовые значения вычисляемые на лету — например, информация о позиционировании. Метод .css() не должен применяться для задания простых стилей, таких как display: none — практически во всех случаях для этого предпочтительнее использовать классы и CSS.

Рассмотрим, к примеру, ситуацию, когда вы хотите задать стиль элемента на основе ширины родительского элемента. Для резинового макета заранее узнать ширину родительского элемента может быть трудно или невозможно. В этом случае мы можем обратиться к методу .css().

var list = $( '#my-unordered-list' );

var width = Math.floor( list.width() \* 0.1 );

list.find('li').each(function( index, elem ) {

var padding = width \* index;

$( elem ).css( 'padding-left', padding + 'px' );

});

Если вам нужно установить сразу несколько свойств, вы можете передать объект в метод .css() вместо имени свойства и его значения. Обратите внимание, что имена свойств с дефисом нужно брать в кавычки.

$( 'li' ).eq( 1 ).css({

'font-size': '20px',

'padding-left': '20px'

});

#### **Изменение значений форм**

jQuery предлагает метод .val() для изменения значения элементов формы, таких как <select> и <input>.

Для текстовых элементов <input> вы можете установить их содержание передавая строку в метод .val():

$( 'input[type="text"]' ).val( 'новое значение' );

Для элементов <select> вы также можете установить выбранный пункт с помощью .val():

$( 'select' ).val( '2' );

Для чекбоксов вы должны установить свойство checked для элемента используя метод .prop().

$( 'input[type="checkbox"]' ).prop( 'checked', 'checked' );

#### **Изменение других атрибутов**

Вы можете использовать метод .attr(), чтобы изменить другие атрибуты элементов. Например, вы можете с его помощью установить новый атрибут title для ссылки:

$( 'a' ).attr( 'title', 'Нажми меня!' );

При установке атрибута вы можете передать функцию в качестве второго аргумента. Подобно другим устанавливающим методам эта функция получает два аргумента: индекс элемента, с которым она работает, и исходное значение атрибута. Функция должна вернуть новое значение атрибута.

$( 'a' ).attr( 'href', function(index, value) {

return value + '?special=true';

});

Вы также можете удалить атрибуты полностью через .removeAttr().

### Получение информации из элементов

В разделе «Основы jQuery» мы обсудили понятия «устанавливающий метод» и «метод чтения». Все методы, которые могут быть использованы для изменения элементов, также могут применяться и для получения информации из элементов. Например, описанный выше метод .val() может быть как устанавливающим методом, так и методом чтения.

var input = $( 'input[type="text"]' );

input.val( 'новое значение' );

input.val(); // вернёт 'новое значение'

Кроме того, метод .css() может быть использован для получения значения отдельных свойств CSS, если для него указать только имя свойства, а не значение.

var listItemColor = $( 'li' ).css( 'color' );

Когда методы манипуляции используются в качестве установки, метод работает только с первым элементом в выборке, с одним исключением: метод .text(). В этом случае, если аргумент не передаётся в метод, то будет возвращён текст всех выбранных элементов.

### Размещение элементов в документе

Если вы выбрали элемент или создали новый элемент, то можете взять вашу выборку и поместить её в документ. Для этого есть два основных способа: путём вызова метода для элемента или элементов, которые вы хотите разместить, или путём вызова метода для элемента, относительно которого вы хотите разместить.

Рассмотрим, к примеру, ситуацию, когда вы хотите переместить первый элемент списка в его конец. Есть несколько способов для достижения этой цели.

Вы можете добавить пункт в список, вызывая .appendTo() для элемента списка:

var listItem = $( '#my-unordered-list li' ).first();

listItem.appendTo( '#my-unordered-list' );

Вы можете добавить пункт в список, вызывая .append() для списка:

var listItem = $( '#my-unordered-list li' ).first();

$( '#my-unordered-list' ).append( listItem );

Вы можете вставить элемент списка после последнего элемента с помощью вызова .insertAfter() для того элемента списка, который вы хотите переместить:

var listItems = $( '#my-unordered-list li' );

listItems.first().insertAfter( listItems.last() );

Вы также можете вставить элемент списка после последнего элемента с помощью вызова .after() для последнего пункта списка:

var listItems = $( '#my-unordered-list li' );

listItems.last().after( listItems.first() );

Существует много других методов для размещения элементов — вы можете разместить их вокруг, внутри, снаружи других элементов, всё зависит от ваших конкретных потребностей.

Наиболее эффективный способ основан на элементах, к которым у вас уже есть доступ. В приведённом выше примере вы можете выбрать как добавить элемент в маркированный список, если список был уже выбран для какой-то другой цели; или, если у вас уже имеется выборка, содержащая все элементы списка, то будет проще разместить первый элемент списка относительно последнего.

Когда выбираете как разместить элемент, вы должны учитывать не только самый простой способ, но и как его сопровождать. Остерегайтесь размещения, в котором делается предположение о точной структуре HTML на вашей страницы.

### Копирование элементов

Вы можете сделать копию элемента или набора элементов используя метод .clone(). Он позволяет делать копию элементов, но заметьте, что копия хранится только в памяти — вы должны самостоятельно поместить её в документ. Вы можете манипулировать клонированным элементом или элементами до их размещения в документе.

var clones = $( 'li' ).clone();

clones.html(function( index, oldHtml ) {

return oldHtml + '!!!';

});

$( '#my-unordered-list' ).append( clones );

jQuery не помешает вам клонировать элемент с идентификатором, поэтому вы должны убедиться, что изменили или удалили атрибут id перед установкой элемента в документ, поскольку в документе не должно быть несколько элементов с одинаковым id.

### Удаление элементов

Есть три способа для удаления элементов из документа: .remove(), .detach() и .replaceWith(). Каждый метод служит конкретной цели.

Метод .remove() следует использовать для удаления элементов навсегда, также он убирает любые обработчики событий связанных с элементами. Метод .remove() возвращает указатель на удалённые элементы, но если вы повторно их добавите, то у них больше не будет связанных с ними событий.

$( '#my-unordered-list li' ).click(function() {

alert( $( this ).text() );

});

var removedListItem = $( '#my-unordered-list li' ).first().remove();

removedListItem.appendTo( '#my-unordered-list' );

removedListItem.trigger( 'click' ); // alert не работает!

Метод .detach() применяется для временного удаления элементов из документа. К примеру, если вы собираетесь внести множество изменений в структуру вашей страницы через jQuery, то будет более эффективно использовать .detach() чтобы удалить выбранные элементы, внести в них необходимые изменения, а затем снова прикрепить элемент, используя один из методов вставки. Элементы, удалённые через .detach(), сохраняют свои обработчики событий; вы можете повторно добавить их в документ с помощью .appendTo() или другим методом.

$( '#my-unordered-list li' ).click(function() {

alert( $( this ).text() );

});

var detachedListItem = $( '#my-unordered-list li' ).first().detach();

// Сделать что-то комплексное с элементом списка

detachedListItem.appendTo( '#my-unordered-list' );

detachedListItem.trigger( 'click' ); // alert!

Наконец, метод .replaceWith() заменяет элемент или элементы на элемент или HTML, который передаётся в качестве аргумента .replaceWith(). Метод возвращает заменённые элементы, но как и с .remove(), убираются все обработчики событий связанные с ними.

$( '#my-unordered-list li' ).click(function() {

alert( $( this ).text() );

});

var replacedListItem = $( '#my-unordered-list li' ).first()

.replaceWith( '<li>новое!</li>' );

replacedListItem.appendTo( '#my-unordered-list' );

replacedListItem.trigger( 'click' ); // alert не работает!