

Ф.А. Перепелица

Разработка интерактивных сайтов с
использованием jQuery

Учебное пособие



Санкт-Петербург
2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Ф.А. Перепелица

**РАЗРАБОТКА ИНТЕРАКТИВНЫХ САЙТОВ С
ИСПОЛЬЗОВАНИЕМ JQUERY**

Учебное пособие



Санкт-Петербург
2015

Перепелица Ф.А. Разработка интерактивных сайтов с использованием jQuery. – СПб: Университет ИТМО, 2015.– 142 с.

Учебное пособие предназначено как для начинающих разработчиков, так и для специалистов, уже занимающихся разработкой Web-сайтов.

jQuery – бесплатный JavaScript-фреймворк позволяющий расширить функциональность и добавить различные эффекты к пользовательскому интерфейсу веб-приложения.

В курсе будет продемонстрированы примеры добавления интерактивных и динамических компонентов страницы, различных элементов управления, эффектов и плагинов.

Учебное пособие используется при изучении дисциплины «Языки разработки приложений для Web» магистерских программ «Web-технологии» и «Компьютерная графика и Web-дизайн» в рамках направления подготовки 09.04.02 «Информационные системы и технологии».

Пособие предназначено для тех учащихся, которые не имеют доступ к специальным учебным файлам (ресурсам), размещенным в аудиториях Университета ИТМО или в центре дистанционного обучения Академии методов и техники управления (ЛИМТУ) — de.ifmo-online.ru

Рекомендовано к печати Ученым советом ЛИМТУ, 27 мая 2015, протокол №5/2015



Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

© Университет ИТМО, 2015

© Перепелица Ф.А., 2015

| | |
|---|-----------|
| Тема 1. Введение в jQuery..... | 11 |
| Библиотеки JavaScript..... | 11 |
| Библиотека jQuery | 11 |
| Начало работы с jQuery | 12 |
| Тема 2. Выбор элементов с помощью селекторов и фильтров..... | 14 |
| Селекторы..... | 14 |
| Базовые селекторы выбора | 14 |
| Универсальный селектор * | 14 |
| Селектор выбора по классу | 14 |
| Селектор выбора по идентификатору..... | 15 |
| Селектор выбора по имени тега..... | 15 |
| Выбор элементов с помощью комбинирования селекторов..... | 15 |
| Выбор нескольких элементов | 15 |
| Выбор потомков элемента | 15 |
| Выбор прямых потомков элемента..... | 16 |
| Выбор следующего селектора..... | 16 |
| Выбор всех следующих селекторов | 16 |
| Выбор элементов по атрибутам | 16 |
| Выбор элемента по имени атрибута..... | 16 |
| Выбор элемента по имени и значению атрибута | 16 |
| Выбор элемента по несовпадению с атрибутом..... | 16 |
| Выбор элемента по началу названия атрибута..... | 17 |
| Выбор элемента по концу названия атрибута | 17 |
| Выбор элемента по части названия атрибута | 17 |
| Выбор элемента по слову в атрибуте..... | 17 |
| Выбор элементов по нескольким атрибутам | 17 |
| Выбор элемента по языку..... | 17 |
| Фильтры | 17 |
| Базовые фильтры. | 18 |
| Выбор элемента по индексу..... | 18 |
| Выбор четных элементов..... | 18 |
| Выбор нечетных элементов..... | 18 |
| Выбор первого дочернего элемента..... | 19 |
| Выбор последнего дочернего элемента..... | 19 |
| Выбор элемента в фокусе | 19 |
| Выбор элементов, идущих после указанного | 19 |

| | |
|---|----|
| Выбор элементов, идущих до указанного..... | 19 |
| Выбор заголовков..... | 20 |
| Выбор элементов, которые в настоящий момент участвуют в анимации | 20 |
| Инвертирование выделения элементов | 20 |
| Фильтры видимости объектов. | 20 |
| Выделение невидимых элементов | 20 |
| Выделение видимых элементов | 21 |
| Фильтры дочерних элементов | 21 |
| Выбор первых вложенных элементов | 21 |
| Выбор последних вложенных элементов..... | 21 |
| Выбор первого указанного элемента | 22 |
| Выбор последнего указанного элемента | 22 |
| Выбор указанного элемента в своих родителях | 22 |
| Выбор указанного элемента, считая от последнего дочернего элемента..... | 23 |
| Выбор определенного элемента в своих родителях..... | 23 |
| Выбор определенного элемента, считая от последнего дочернего элемента.... | 24 |
| Выбор единственного дочернего элемента..... | 24 |
| Выбор единственного указанного дочернего элемента | 25 |
| Фильтры по содержимому | 25 |
| Выбор элемента по тексту | 25 |
| Выбор всех непустых элементов | 25 |
| Выбор пустых элементов | 25 |
| Выбор элементов с указанным тегом..... | 26 |
| Фильтры элементов форм..... | 26 |
| Выбор всех элементов форм | 26 |
| Выбор кнопок | 26 |
| Выбор кнопок «Отправить»..... | 26 |
| Выбор кнопок «Очистить» | 26 |
| Выбор «Флажков» | 27 |
| Выбор «Переключателей» | 27 |
| Выбор текстовых полей | 27 |
| Выбор полей для ввода паролей | 27 |
| Выбор полей для загрузки файлов | 28 |
| Выбор кнопки отправки формы в виде изображения..... | 28 |
| Выбор выпадающего меню. | 28 |
| Выделение выбранных элементов | 28 |
| Выделение активных элементов..... | 28 |
| Выделение заблокированных элементов | 28 |

| | |
|---|-----------|
| Практические задания к теме 2 | 29 |
| Практическое задание 2.1 Выбор элементов по классу | 29 |
| Практическое задание 2.2 Выбор элементов по идентификатору и имени тега | 30 |
| Практическое задание 2.3 Выбор смежных элементов | 31 |
| Практическое задание 2.4 Выбор элементов по атрибуту и значению | 31 |
| Практическое задание 2.5 Выбор элемента по нескольким атрибутам | 32 |
| Практическое задание 2.6 Выбор четных и нечетных элементов | 34 |
| Практическое задание 2.7 Выбор элементов до и после указанного | 35 |
| Практическое задание 2.8 Выбор первого и последнего элемента | 36 |
| Практическое задание 2.9 Инвертирование выделения элементов | 37 |
| Практическое задание 2.10 Выбор первого и последнего дочернего элемента | 38 |
| Практическое задание 2.11 Выбор элемента по содержимому | 39 |
| Практическое задание 2.12 Выбор кнопок формы | 40 |
| Практическое задание 2.13 Выбор элементов формы | 41 |
| Самостоятельная работа к теме 2 | 43 |
| Самостоятельная работа 2.1 Выбор элементов | 43 |
| Тема 3. Манипуляции с элементами, их свойствами и содержимым. | 44 |
| Манипуляции со стилями | 44 |
| Функция CSS. | 44 |
| Определение наличия класса у элемента. | 44 |
| Добавление класса к элементу | 44 |
| Удаление класса из элемента | 45 |
| Управление высотой элемента | 45 |
| Управление шириной элемента | 46 |
| Определение позиции элемента | 47 |
| Определение положения полосы прокрутки | 48 |
| Добавление или удаление класса | 48 |
| Управление атрибутами | 48 |
| Возвращение и изменение атрибута элемента | 48 |
| Удаление атрибута элемента | 49 |
| Изменение свойств атрибутов элемента | 49 |
| Удаление свойства элемента | 49 |
| Замена и клонирование элементов | 49 |
| Замена элементов и содержимого | 49 |
| Клонирование элементов и их содержимого | 50 |
| Удаление элемента и его содержимого | 50 |
| Удаление элементов | 50 |

| | |
|---|-----------|
| Удаление содержимого элемента | 50 |
| Удаление родительских элементов | 50 |
| Добавление содержимого элемента | 51 |
| Работа с html-содержимым элемента | 51 |
| Изменение текстового содержимого элемента..... | 51 |
| Изменение содержимого атрибута value | 51 |
| Добавление содержимого в конец элемента | 51 |
| Добавление содержимого в начало элемента | 52 |
| Добавление содержимого после элемента | 52 |
| Добавление содержимого до элемента..... | 52 |
| Обертывание элементов..... | 53 |
| Обертывание содержимого элемента..... | 53 |
| Практические задания к теме 3 | 54 |
| Практическое задание 3.1 Использование функции css | 54 |
| Практическое задание 3.2 Добавление и удаление класса к элементу..... | 55 |
| Практическое задание 3.3 Добавление класса к элементу..... | 56 |
| Практическое задание 3.4 Изменение положения на странице | 57 |
| Практическое задание 3.5 Изменение положения прокрутки на странице | 58 |
| Практическое задание 3.6 Изменение класса элемента..... | 59 |
| Практическое задание 3.7 Добавление и удаление атрибута | 60 |
| Практическое задание 3.8 Добавление и удаление значений атрибута | 62 |
| Практическое задание 3.9 Замена и клонирование элемента | 63 |
| Практическое задание 3.10 Удаление элементов и их содержимого..... | 64 |
| Практическое задание 3.11 Изменение содержимого элемента..... | 66 |
| Практическое задание 3.12 Добавление содержимого в начало и конец элемента | 67 |
| Практическое задание 3.13 Добавление элемента до и после выбранного элемента | 68 |
| Практическое задание 3.14 Обёртывание элемента и его содержимого..... | 69 |
| Самостоятельная работа к теме 3..... | 72 |
| Самостоятельная работа 3.1 Выбор элементов | 72 |
| Тема 4. События | 73 |
| Объект event | 73 |
| Определение типа события..... | 73 |
| Время запуска события..... | 73 |
| Пространство имен | 74 |
| Прямая и делегированная обработка (параметр selector) | 74 |
| Обработчики событий..... | 75 |

| | |
|---|----|
| Дополнительные данные в обработчике событий | 76 |
| Определение источника события | 77 |
| Определение положения курсора мыши. | 77 |
| Базовые события. | 77 |
| События на выбранных элементах..... | 77 |
| Удаление обработчиков событий | 78 |
| Событие на выбранных элементах | 78 |
| Удаление обработчика событий | 78 |
| События мыши | 78 |
| Обработчик события click | 79 |
| Обработчик события dblclick | 79 |
| Обработчик события hover | 79 |
| Нажатие клавиши мыши | 79 |
| Отпуск клавиши мыши | 79 |
| Наведение курсора мыши на элемент..... | 80 |
| Выход курсора мыши с элемента..... | 80 |
| Движение мышью | 81 |
| Поочерёдное выполнение нескольких функций. | 81 |
| События загрузки страницы | 82 |
| Обработка загрузки DOM..... | 82 |
| Обработка загрузки страницы | 82 |
| Обработка выгрузки страницы. | 82 |
| События браузера | 82 |
| Обработчик ошибки. | 83 |
| Изменение окна браузера | 83 |
| Прокрутка документа | 83 |
| События клавиатуры | 83 |
| Нажатие клавиши клавиатуры..... | 85 |
| Отпускание клавиши клавиатуры..... | 85 |
| Определение введенного символа..... | 85 |
| События формы..... | 85 |
| Получение фокуса элементом. | 85 |
| Потеря фокуса элементом. | 86 |
| Выделение текста. Событие Select. | 86 |
| Изменение элемента формы. Событие Change. | 86 |
| Отправка формы. Событие Submit | 86 |
| Практические задания к теме 4 | 87 |

| | |
|--|------------|
| Практическое задание 4.1 Всплытие событий | 87 |
| Практическое задание 4.2 Поля метода Event | 88 |
| Практическое задание 4.3 Делегирование событий | 90 |
| Практическое задание 4.4 Извлечение данных, переданных событием..... | 91 |
| Практическое задание 4.5 Событие on и удаление события | 92 |
| Практическое задание 4.6 Событие one и его удаление..... | 94 |
| Практическое задание 4.7 События мыши | 95 |
| Практическое задание 4.8 События мыши | 97 |
| Практическое задание 4.9 События мыши | 99 |
| Практическое задание 4.10 Использование метода toggle..... | 101 |
| Практическое задание 4.11 События браузера..... | 102 |
| Практическое задание 4.12 События клавиатуры..... | 104 |
| Самостоятельная работа к теме 4..... | 105 |
| Самостоятельная работа 4.1 Определение положения курсора мыши..... | 105 |
| Самостоятельная работа 4.2 Определение времени события | 105 |
| Тема 5. Методы работы с наборами элементов | 106 |
| Перемещение по иерархии DOM. | 106 |
| Поиск заданных дочерних элементов. | 106 |
| Поиск родительских элементов. | 106 |
| Поиск дочерних элементов по заданному селектору. | 106 |
| Выбор элементов, лежащих после указанного..... | 107 |
| Выбор элементов, лежащих перед указанным..... | 107 |
| Выбор позиционированного родителя. | 108 |
| Выбор лежащих рядом элементов. | 108 |
| Фильтрация элементов набора..... | 108 |
| Метод filter..... | 108 |
| Метод is..... | 108 |
| Поиск элементов по индексам..... | 108 |
| Дополнительные методы..... | 108 |
| Добавление элементов в набор..... | 108 |
| Возвращение к предыдущему набору элементов..... | 109 |
| Объединение выбранных наборов..... | 109 |
| Содержимое элемента..... | 109 |
| Функция для элемента в наборе элемента. | 109 |
| Элементы набора элементов. | 110 |
| Преобразование набора элементов в массив..... | 110 |
| Индекс элемента в массиве..... | 110 |

| | |
|--|------------|
| Размер набора элементов | 110 |
| Пользовательские данные элемента | 110 |
| Практические задания к теме 5 | 112 |
| Практическое задание 5.1 Поиск родительских элементов..... | 112 |
| Практическое задание 5.2 Выбор элементов(,) лежащих после указанного | 113 |
| Практическое задание 5.3 Выбор элементов, лежащих перед указанным | 115 |
| Практическое задание 5.4 Фильтрация элементов набора | 117 |
| Практическое задание 5.5 Объединение выбранных наборов | 118 |
| Практическое задание 5.6 Содержимое элемента | 119 |
| Практическое задание 5.7 Функция для элемента в наборе элемента..... | 121 |
| Самостоятельная работа к теме 5..... | 123 |
| Самостоятельная работа 5.1 Выбор позиционированного родителя | 123 |
| Самостоятельная работа 5.2 Содержимое элемента | 123 |
| Самостоятельная работа 5.3 Массив элементов..... | 123 |
| Тема 6. Анимация и эффекты jQuery | 124 |
| Встроенные функции анимации. | 124 |
| Появление и скрытие элемента. | 124 |
| Сворачивание и разворачивание элемента | 124 |
| Поочередное разворачивание и сворачивание элементов..... | 125 |
| Изменение прозрачности элемента | 125 |
| Изменение уровня прозрачности элемента..... | 125 |
| Поочередное появление и скрытие элементов | 125 |
| Управление анимацией..... | 126 |
| Создание пользовательской анимации..... | 126 |
| Изменение очереди анимации | 126 |
| Очистка очереди функций | 127 |
| Приостановка анимации..... | 128 |
| Завершение и остановка анимации..... | 128 |
| Изменение скорости анимации | 128 |
| Отмена всех анимации | 129 |
| Выбор анимированного элемента | 129 |
| Практические задания к теме 6 | 130 |
| Практическое задание 6.1 Появление и скрытие элемента..... | 130 |
| Практическое задание 6.2 Использование прозрачности..... | 132 |
| Практическое задание 6.3 Одновременная и последовательная анимация | 134 |
| Практическое задание 6.4 Вставка функции в очередь..... | 136 |
| Практическое задание 6.5 Остановка анимации | 138 |

| | |
|---|-----|
| Самостоятельная работа к теме 6..... | 141 |
| Самостоятельная работа 6.1 Сворачивание и разворачивание элемента..... | 141 |
| Самостоятельная работа 6.2 Последовательное изменение прозрачности элемента | 141 |

Тема 1. Введение в jQuery

Библиотеки JavaScript

Библиотеки JavaScript получили заслуженное уважение в среде веб-разработчиков и используются во множестве проектов. Основные причины использования библиотек следующие:

- синтаксис библиотек более простой по сравнению с javascript
- библиотеки кроссбраузерны, что существенно упрощает разработку сайтов для разных браузеров и не требует тестирования кода для разных браузеров
- в библиотеках реализованы популярные функции, что существенно экономит время при написании кода
- для библиотек написано множество плагинов
- библиотеки поддерживают сложные визуальные эффекты

Помимо jQuery к наиболее распространенным библиотекам можно отнести следующие:

- Dojo - разработана с целью упростить ускоренную разработку основанных на JavaScript или AJAX приложений и сайтов.
- MooTools - является модульной, объектно-ориентированной библиотекой, созданной для помощи разработчикам JavaScript. MooTools совместим и протестирован с браузерами: Safari 2+, Internet Explorer 6+, Firefox 2+ (и другими, основанными на движке Gecko), Opera 9+. Подходит для написания сложных приложений
- Yahoo! UI Library (YUI) — библиотека JavaScript для создания богатых интерактивными возможностями приложений или/и пользовательского интерфейса. Использует AJAX, анимацию, надстройки над XMLHttpRequest и DOM, «drag-and-drop», слайдеры, слайды, календари, деревья, табы и другие новинки, составляющие понятие «Веб 2.0»
- Prototype — библиотека JavaScript, упрощающая работу с Ajax и некоторыми другими функциями. Несмотря на его доступность в виде отдельной библиотеки, он обычно используется программистами вместе с более сложными языками программирования.

Сравнение возможностей библиотеки можно посмотреть по ссылке - http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

Библиотека jQuery

«jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими» - wikipedia.

В чем преимущества jQuery и почему это библиотека так распространена:

- самый простой синтаксис и самый компактный код;

- простое и понятное обращение и манипулирование элементами веб-страницы;
- множество готовых плагинов для решения разных задач;
- доступная документация
- библиотека постоянно развивается добавляя новые возможности и упрощая работу с кодом

Официальная документация jQuery располагается на сайте <http://jquery.com/>.

Наиболее полная документация на русском языке на сайте <http://jquery.page2page.ru>.

Начало работы с jQuery

jQuery является внешней библиотекой и включается в веб-страницу, как один внешний JavaScript-файл. Вы можете скачать последнюю версию jQuery с официального сайта <http://jquery.com/download/> и использовать, локальный файл библиотеки.

```
<script type="text/javascript" src="jquery-1.9.1.js"></script>
```

Так же можно дать прямую ссылку на файл с библиотекой:

```
<script type="text/javascript" src="http://code.jquery.com/jquery-1.9.1.js"></script>
```

Можно использовать сжатую версию библиотеки, тогда в названии файла после версии будет написано «min» - jquery-1.9.1.min.js, или не сжатую, тогда в названии файла не будет окончания «min» - jquery-1.9.1.js. Разница между сжатой и несжатой версией файла заключается в том, что в сжатой версии удалены все ненужные пробелы и переносы строк, что приводит к уменьшению размера файла и соответственно времени загрузки. Размер файла несжатой версии библиотеки равен 264 кб, а сжатой 92 кб.

Основной функцией jQuery является функция jQuery() или \$() с нее начинается любая запись, использующая библиотеку. Например, для выбора всех div-элементов на странице с помощью jQuery следует ввести следующий код:

```
$("#div")
```

Так же следует помнить, что jQuery библиотека javascript, так что код следует писать внутри тега javascript.

```
<script type="text/javascript">
$("#div")
</script>
```

Так же можно совмещать код jQuery и стандартного javascript. В следующем примере в первой строке стандартными средствами JavaScript выбран элемент, а во второй строке с помощью jQuery и метода css() задана граница выбранному элементу:

```
var elem = document.getElementById("element");
$(elem).css("border", "1px solid red")
```

Но можно использовать и один jQuery, что значительно короче:

```
$("#element").css("border", "1px solid red")
```

В следующих темах рассмотрены основные возможности библиотеки jQuery.

Тема 2. Выбор элементов с помощью селекторов и фильтров

Селекторы

Селекторами называют строчные выражения, которые используются для поиска и выбора DOM элементов на веб-странице. Синтаксис селекторов во многом схож с синтаксисом CSS. Селекторы позволяют находить элементы по различным свойствам: по классам, идентификаторам, тегам, наличию атрибутов, значению атрибутов, по иерархическому порядку расположения элементов и даже по содержимому элементов.

Базовые селекторы выбора

Универсальный селектор *

Универсальный селектор позволяет выбирать все элементы на странице, включая теги разделов (body, head) и теги script и link. Этот селектор замедляет работу скрипта, так что если это возможно лучше использовать другой селектор.

Для выбора всех тегов на странице используется следующая запись:

```
jQuery("*").css('border', '1px solid red')
```

Для того чтобы выбрать все элементы внутри отдельного тега используется следующая запись:

```
jQuery("body *").css("border", "3px solid red")
```

Для того чтобы выбрать все элементы внутри класса используется следующая запись:

```
jQuery(".header *").css("border", "3px solid red")
```

Селектор выбора по классу

Селектор выбора по классу позволяет находить элементы, используя классы. Данный селектор является альтернативой `getElementsByClassName()` в JavaScript. Если на странице несколько элементов с одним и тем же классом, то возвращается массив объектов. К одному из элементов массива элементов, выбранных по классу, можно обратиться через индекс.

Для выбора элемента по классу используется следующая запись:

```
jQuery(".header")
```

Так же можно ограничить выбор класса определенным тегом и применить к нему свойство CSS:

```
jQuery('div.header').css('backgroundColor', 'red')
```

Для выбора по двум классам, примененным к одному элементу, используется следующая запись:

```
jQuery(".header.logo")
```

```
| <div class='header logo'>
```

Для выбора элемента по классу и его индексу в массиве классов используется следующая запись:

```
| $(".header")[0]
```

Селектор выбора по идентификатору

Селектор выбора по идентификатору(id) позволяет находить элементы, используя идентификатор элемента. Данный селектор является альтернативой getElementById() в JavaScript. При выборе по идентификатору создается массив элементов, даже если элемент один в выборке.

Идентификатор должен использоваться только один раз на странице. Если используется один и тот же идентификатор более одного раза на странице, то при выборе с помощью селектора по идентификатору будет выбран только первый элемент с данным идентификатором.

Для выбора тега по идентификатору используется следующая запись:

```
| $("#header")
```

Селектор выбора по имени тега

Селектор выбора по имени тега позволяет находить элементы, используя имена тегов. Данный селектор является альтернативой getElementsByTagName() в JavaScript.

Для выбора тега по имени используется следующая запись:

```
| $("div")
```

Выбор элементов с помощью комбинирования селекторов

Комбинирование селекторов позволяет создавать более сложные выборочные запросы. Написание комбинированных запросов схоже с синтаксисом CSS.

Выбор нескольких элементов

Выбирает все элементы, указанные через запятую.

Для выбора одновременно нескольких элементов используется следующая запись:

```
| $("div, .header, #logo")
```

Выбор потомков элемента

Для выбора потомка элемента следует поставить пробел между обращением к элементу и выбираемым элементом.

В следующей записи будет выбран элемент с классом logo, который находится внутри тега div.

```
| $("div .logo")
```

Тема 2. Выбор элементов с помощью селекторов и фильтров

В следующей записи будет выбран тег p, который находится внутри тега div с классом header.

```
$("#div.header p ")
```

Выбор прямых потомков элемента

Для выбора прямых потомков элемента используется следующая записи:

```
$("#div>p")  
$("#div.header>span ")
```

Выбор следующего селектора

В следующей записи будет выбран тег p, который располагается после тега div.

```
$("#div+p")
```

Выбор всех следующих селекторов

Для выбора всех последующих после указанного селектора используется следующая записи:

```
$("#div~p")
```

Выбор элементов по атрибутам

Выбор элемента по имени атрибута

Для выбора элементов с атрибутом href используется следующая записи:

```
$("#[href]")
```

Для выбора только тега p с атрибутом align используется следующая записи:

```
$("#p[align]")
```

Для выбора всех элементов с атрибутом href, находящихся внутри тега div, используется следующая записи:

```
$("#div [href]")
```

Выбор элемента по имени и значению атрибута

Для выбора элементов с атрибутом href и значением http://www.yandex.ru/ используется следующая записи:

```
$("#[href= http://www.yandex.ru] ")
```

Если значение атрибута состоит из двух и более слов, то их следует заключить в кавычки.

```
$("#[alt= 'Бегущий пес'] ")
```

Выбор элемента по несовпадению с атрибутом

Для выбора элемента, который либо не имеют указанный атрибут, либо имеют

указанный атрибут, но с другим значением используется следующая запись:

```
jQuery("[href != http://www.yandex.ru] ")
```

```
jQuery("[alt != 'Бегущий пес']")
```

Выбор элемента по началу названия атрибута

Для выбора всех элементов, чье значение атрибута title начинается с news, используется следующая запись:

```
jQuery("[title ^= news]")
```

Выбор элемента по концу названия атрибута

Для выбора всех элементов, чье значение атрибута title заканчивается на '2012', используется следующая запись:

```
jQuery("[title $= 2012]")
```

Выбор элемента по части названия атрибута

Для выбора всех элементов, чье значение атрибута title содержит 'new', используется следующая запись:

```
jQuery("[title *= newweek]")
```

Выбор элемента по слову в атрибуте

Для выбора всех элементов, у которых в значении атрибута title встречается слово "new" используется следующая запись:

```
jQuery("[title ~= new]")
```

Выбор элементов по нескольким атрибутам

Для выбора элементов по нескольким атрибутам используется следующая запись:

```
jQuery("[type=radio][name = 'gender'][value != 'male']")
```

В данном примере будет выбран элемент с атрибутом type равным radio, атрибутом name равным gender и атрибутом value не равным male.

Выбор элемента по языку

Выбор элемента по коду языка осуществляется с помощью селектора lang.

```
jQuery("div:lang(en-us) ")
```

Фильтры

Фильтры предназначены для уточнения поиска объектов. Фильтры схожи по назначению и синтаксису с псевдоклассами css. Написание фильтров в отличие от селекторов начинается всегда с двоеточия.

Базовые фильтры.

Выбор элемента по индексу.

Все элементы на странице можно представить в виде массива. У каждого элемента свой индекс, по которому его можно выбрать, используя фильтр `eq()`. В скобках фильтра располагается индекс массива, то есть для того чтобы выбрать первый элемент необходимо указать значение 0. Отрицательный индекс не допустим.

Синтаксис этого фильтра следующий:

```
| $(":eq(index) ")
```

В следующей записи мы выбираем первый элемент на странице и присваиваем ему фоновый цвет. Первый элемент на странице тег `html`.

```
| $(":eq(0) ").css('backgroundColor','green')
```

Если необходимо выбрать первый элемент в теле документа следует использовать уточнение в виде тега `body`. Например, выберем первый тег `div` внутри тега `body`.

```
| $('body div:eq(0)').css('backgroundColor','green')
```

В следующем показан еще более точный способ выбора объекта. В этом примере будет выбран второй `div` с идентификатором `header`, находящийся в теге `body`.

```
| $('body div#header:eq(1)').css('backgroundColor','green')
```

Выбор четных элементов

С помощью фильтра `even` можно выбрать четные элементы на всей странице или в заданном теге. Особенно удобно выбирать четные строчки в таблицы для их оформления.

Синтаксис этого фильтра следующий:

```
| $(":even")
```

Для выбора четных ячеек в таблице с идентификатором `mail` следует использовать следующую запись:

```
| $("#mail tr:even").css('backgroundColor','#cccccc')
```

Выбор нечетных элементов

С помощью фильтра `odd` можно выбрать нечетные элементы на всей странице или в заданном теге. Особенно удобно выбирать нечетные строчки в таблицы для их оформления.

Синтаксис этого фильтра следующий:

```
| $(":odd")
```

Для выбора нечетных ячеек в таблице с идентификатором `mail` следует использовать следующую запись:

```
| $("#mail tr:odd").css('backgroundColor','#E6E6E6')
```

Выбор первого дочернего элемента

С помощью фильтра `first` можно выбрать первый элемент на странице или в указанном теге. Этот фильтр действует аналогично фильтру `:eq` со значением 0 (`:eq(0)`).

Синтаксис этого фильтра следующий:

```
| $(":first")
```

В следующем примере выбрана первая строка в таблице с идентификатором `mail`

```
| $('#mail tr:first').css('backgroundColor','#95AFC1')
```

Выбор последнего дочернего элемента

С помощью фильтра `last` можно выбрать последний элемент на странице или в указанном теге.

Синтаксис этого фильтра следующий:

```
| $(":last")
```

В следующем примере выбрана последняя строка в таблице с идентификатором `mail`

```
| $('#mail tr:last').css('backgroundColor','#95AFC1')
```

Выбор элемента в фокусе

С помощью фильтра `focus` можно выбрать выделенный элемент на странице.

Синтаксис этого фильтра следующий:

```
| $(".focus")
```

В следующем примере выбрано поле ввода текста при выделении.

```
| $("input:focus")
```

Выбор элементов, идущих после указанного

С помощью фильтра `gt` можно выбрать все элементы после указанного. Для выбора всех элементов необходимо указать **индекс** элемента, после которого следует выбирать объекты.

Синтаксис этого фильтра следующий:

```
| $(".gt(index)")
```

В следующем примере выбраны все строки, идущие после третьей строки объекта с идентификатором `mail`.

```
| $(".gt(2)").css('backgroundColor','red')
```

Выбор элементов, идущих до указанного

С помощью фильтра `lt` можно выбрать все элементы до указанного. Для выбора всех элементов необходимо указать **индекс** элемента, до которого следует выбирать объекты.

Синтаксис этого фильтра следующий:

```
$("#:lt(index)")
```

В следующем примере выбраны все строки, идущие до третьей строки объекта с идентификатором mail.

```
$("##mail tr:lt(2)").css('backgroundColor','red')
```

Выбор заголовков

С помощью фильтра header можно выбрать заголовки всех уровней (h1, h2, h3, h4, h5, h6).

Синтаксис этого фильтра следующий:

```
$("#:header")
```

В следующем примере будут выбраны все заголовки и им будет назначен фоновый цвет:

```
$("#:header").css('backgroundColor','#95AFC1')
```

В следующем примере будет найден заголовок внутри класса content и назначен ему размер шрифта. На заголовки вне класса content это правило действовать не будет.

```
$('.content :header').css('fontSize','12px')
```

Выбор элементов, которые в настоящий момент участвуют в анимации

С помощью фильтра animated можно выбрать элементы, участвующие в данный момент в анимации. Анимация и взаимодействие с анимированными элементами будут рассмотрены в последующих главах.

Синтаксис этого фильтра следующий:

```
$("#:animated")
```

Инвертирование выделения элементов

С помощью фильтра not можно выбрать все элементы, которые не удовлетворяют условию, то есть инвертировать выбор.

Синтаксис этого фильтра следующий:

```
$("#:not(selector)")
```

В следующем примере будут все заголовки, кроме находящихся внутри класса content

```
$("#:header:not(.content :header)").css('backgroundColor','#95AFC1')
```

Фильтры видимости объектов.

Выделение невидимых элементов

С помощью фильтра hidden можно выбрать элементы, которые в данный момент

невидимы на странице. Невидимым элементом считается элемент, удовлетворяющий одному из следующих правил:

- css-свойство `display` равно `none`
- элементом формы с `type="hidden"`
- Высота или ширина элемента равна 0
- Элемент находится внутри невидимого элемента и поэтому тоже невидим на странице.

Элементы с css-свойством `visibility` равным `hidden`, а так же элементы с нулевой прозрачностью считаются видимыми, поскольку они продолжают занимать место на странице.

Синтаксис этого фильтра следующий:

```
| $(":hidden")
```

В следующем примере будут выделены все скрытые строки внутри элемента с идентификатором `mail`.

```
| $("#mail tr:hidden")
```

Выделение видимых элементов

С помощью фильтра `visible` можно выбрать элементы, которые в данный момент видимы на странице.

Синтаксис этого фильтра следующий:

```
| $(":visible ")
```

В следующем примере будут выделены все видимые строки внутри элемента с идентификатором `mail`.

```
| $("#mail tr:visible")
```

Фильтры дочерних элементов

Выбор первых вложенных элементов

С помощью фильтра `first-child` можно выбрать элементы, которые располагаются первыми внутри указанных элементов.

Синтаксис этого фильтра следующий:

```
| $(":first-child ")
```

В следующем примере будет выбран первый элемент в теге `body`.

```
| $('body :first-child').css('border','2px solid green')
```

В следующем примере выбраны все первые элементы внутри `div`.

```
| $('div :first-child').css('border','2px solid green')
```

Выбор последних вложенных элементов

С помощью фильтра `last-child` можно выбрать элементы, которые располагаются

последними внутри указанных элементов.

Синтаксис этого фильтра следующий:

```
$("#:last-child ")
```

В следующем примере будет выбран последний элемент в теге body.

```
$('#body :last-child').css('border','2px solid red')
```

В следующем примере выбраны все последние элементы внутри div.

```
$('#div :last-child').css('border','2px solid red')
```

Выбор первого указанного элемента

С помощью фильтра first-of-type можно выбрать все указанные первые вложенные элементы, даже если они не являются первым дочерним элементом.

Синтаксис этого фильтра следующий:

```
$("#:first-of-type")
```

В следующем примере будут выбраны все первые дочерние теги p.

```
$("#p:first-of-type")
```

Выбор последнего указанного элемента

С помощью фильтра last-of-type можно выбрать все указанные последние вложенные элементы, даже если они не являются последними дочерними элементами.

Синтаксис этого фильтра следующий:

```
$("#:last-of-type")
```

В следующем примере будут выбраны все последние дочерние теги p.

```
$("#p:last-of-type")
```

Выбор указанного элемента в своих родителях

С помощью фильтра nth-child можно выбрать элементы с определенными условиями внутри непосредственных предков. При указании числовой позиции отсчет ведется от 1, а не от 0, как при указании индекса.

Синтаксис этого фильтра следующий:

```
$("#:nth-child(number|even|odd|equation)")
```

В следующем примере будут выбраны все span элементы, находящиеся в своих родителях пятыми.

```
$("#span:nth-child(5)")
```

В следующем примере будут выбраны все нечетные span элементы, находящиеся в своих родителях.

```
$("#span:nth-child(odd)")
```

В следующем примере будут выбраны все четные span элементы, находящиеся в своих родителях.

```
| $("span:nth-child(even)")
```

В следующем примере будут выбраны все span элементы(,) находящиеся в своих родителях, удовлетворяющих следующему выражению $3n+2$ (5, 7, 9...).

```
| $("span:nth-child(3n+2)")
```

Выбор указанного элемента, считая от последнего дочернего элемента

С помощью фильтра nth-last-child можно выбрать элементы с определенными условиями внутри непосредственных предков, считая от последнего элемента. При указании числовой позиции отсчет ведется от 1, а не от 0, как при указании индекса.

Синтаксис этого фильтра следующий:

```
| $("nth-last-child (number|even|odd|equation)")
```

В следующем примере будут выбраны все span элементы, находящиеся в своих родителях пятыми, считая от последнего дочернего элемента.

```
| $("span:nth-last-child (5)")
```

В следующем примере будут выбраны все нечетные span элементы, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
| $("span:nth-last-child (odd)")
```

В следующем примере будут выбраны все четные span элементы, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
| $("span:nth-last-child (even)")
```

В следующем примере будут выбраны все span элементы, находящиеся в своих родителях, удовлетворяющих следующему выражению $3n+2$ (5, 7, 9...), считая от последнего дочернего элемента.

```
| $("span:nth-last-child (3n+2)")
```

Выбор определенного элемента в своих родителях

С помощью фильтра nth-of-type можно выбрать определенные элементы с определенными условиями внутри непосредственных предков, считая от первого вложенного элемента. При указании числовой позиции отсчет ведется от 1, а не от 0, как при указании индекса.

Синтаксис этого фильтра следующий:

```
| $("nth-of-type (number|even|odd|equation)")
```

В следующем примере будут выбраны все пункты списка, находящиеся в своих родителях вторыми, считая от последнего дочернего элемента.

```
| $("ul li:nth-of-type (2) ")
```

Тема 2. Выбор элементов с помощью селекторов и фильтров

В следующем примере будут выбраны все нечетные пункты списка, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
$("ul li:nth-of-type(odd)")
```

В следующем примере будут выбраны все четные пункты списка, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
$("ul li:nth-of-type(even)")
```

В следующем примере будут выбраны все пункты списка, удовлетворяющие следующему выражению $3n+2$ (5, 7, 9...), считая от последнего дочернего элемента.

```
$("ul li:nth-of-type(3n+2)")
```

Выбор определенного элемента, считая от последнего дочернего элемента

С помощью фильтра `nth-last-of-type` можно выбрать определенные элементы с определенными условиями внутри непосредственных предков, считая от последнего вложенного элемента. При указании числовой позиции отсчет ведется от 1, а не от 0, как при указании индекса.

Синтаксис этого фильтра следующий:

```
$(".nth-last-of-type (number|even|odd|equation)")
```

В следующем примере будут выбраны все пункты списка, находящиеся в своих родителях вторыми, считая от последнего дочернего элемента.

```
$("ul li:nth-last-of-type(2) ")
```

В следующем примере будут выбраны все нечетные пункты списка, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
$("ul li:nth-last-of-type (odd)")
```

В следующем примере будут выбраны все четные пункты списка, находящиеся в своих родителях, считая от последнего дочернего элемента.

```
$("ul li:nth-last-of-type (even)")
```

В следующем примере будут выбраны все пункты списка, удовлетворяющие следующему выражению $3n+2$ (5, 7, 9...), считая от последнего дочернего элемента.

```
$("ul li:nth-last-of-type (3n+2)")
```

Выбор единственного дочернего элемента

С помощью фильтра `only-child` можно выбрать элементы, являющиеся единственными дочерними элементами.

Синтаксис этого фильтра следующий:

```
$(".only-child")
```

В следующем примере будут выбраны все `span` элементы, являющиеся

единственными дочерними элементами в своих родителях.

```
$("#span:only-child")
```

Выбор единственного указанного дочернего элемента

С помощью фильтра `only-of-type` можно выбрать указанные элементы, являющиеся единственными дочерними элементами.

Синтаксис этого фильтра следующий:

```
$("#:only-of-type")
```

В следующем примере будут выбраны только `span` элементы, являющиеся единственными дочерними элементами в своих родителях.

```
$("#span:only-of-type ")
```

Фильтры по содержимому

Выбор элемента по тексту

С помощью фильтра `contains` можно выбрать элементы, содержащие указанную строку текста. Нужно иметь ввиду, что выделен будет не только элемент, в котором непосредственно находится текст, но и все родительские элементы этого элемента.

Синтаксис этого фильтра следующий:

```
$("#:contains (string)")
```

В следующем примере будут выбраны элементы, содержащие текст «Ноль».

```
$("#:contains('Ноль').css('border','2px solid green')
```

В следующем примере будут выделены все объекты, находящиеся внутри класса `one` и в которых не содержится текст «Ноль».

```
$("#.one :not(:contains('Ноль'))).css('border','2px solid green')
```

Выбор всех непустых элементов

С помощью фильтра `parent` можно выбрать элементы, которые имеют содержимое как текст, так и вложенные элементы. Пробел и перенос строки воспринимается, как вложенное содержимое.

Синтаксис этого фильтра следующий:

```
$("#:parent ")
```

В следующем примере будут выбраны все элементы, содержащие текст или вложенные элементы, находящиеся внутри класса `two`.

```
$("#.two :parent").css('backgroundColor','red')
```

Выбор пустых элементов

С помощью фильтра `empty` можно выбрать элементы, которые не имеют содержимое как текста, так и вложенные элементы. Пробел и перенос строки

воспринимается, как вложенное содержимое.

Синтаксис этого фильтра следующий:

```
$("#:empty")
```

В следующем примере будут выбраны все пустые элементы внутри класса two.

```
$(".two :empty").css('backgroundColor','red')
```

Выбор элементов с указанным тегом

С помощью фильтра `has` можно выбрать элементы, внутри которых содержится указанный тег. Синтаксис этого фильтра следующий:

```
$("#:has(selector)")
```

В следующем примере будут выбраны все вложенные теги `i` внутри класса `two`.

```
$(".two :has(i)").css('border','2px solid green')
```

Фильтры элементов форм

Выбор всех элементов форм

С помощью фильтра `:input` можно выбрать все элементы формы, включая `textarea`, `select` и кнопки.

```
$("#:input")
```

Выбор кнопок

С помощью фильтра `button` можно выбрать элементы формы, которые являются кнопками. К кнопкам относятся элементы `input` со значением атрибута `type` равным `button`.

Синтаксис этого фильтра следующий:

```
$("#:button")
```

Выбор кнопок «Отправить»

С помощью фильтра `submit` можно выбрать элементы формы, которые являются кнопками отправить. К кнопкам относятся элементы `input` со значением атрибута `type` равным `submit`.

Синтаксис этого фильтра следующий:

```
$("#:submit ")
```

Выбор кнопок «Очистить»

С помощью фильтра `reset` можно выбрать элементы формы, которые являются кнопками «Очистить». К кнопкам относятся элементы `input` со значением атрибута `type` равным `reset`. Синтаксис этого фильтра следующий:

```
$("#:reset")
```

Выбор «Флажков»

С помощью фильтра `checkbox` можно выбрать элементы формы, которые являются «Флажками». К флажкам относятся элементы `input` со значением атрибута `type` равным `checkbox`. Синтаксис этого фильтра следующий:

```
| $("checkbox ")
```

В следующем примере выбираются все флажки на странице, и для них устанавливается атрибут выделения `checked`.

```
| $("input:checkbox").attr('checked','checked')
```

Выбор «Переключателей»

С помощью фильтра `radio` можно выбрать элементы формы, которые являются кнопками «Переключателями». К флажкам относятся элементы `input` со значением атрибута `type` равным `radio`.

Синтаксис этого фильтра следующий:

```
| $("radio ")
```

В следующем примере выбираются все переключатели на странице, и для них устанавливается атрибут блокировки `disabled`.

```
| $("input:radio").attr('disabled','disabled')
```

Выбор текстовых полей

С помощью фильтра `text` можно выбрать элементы формы, которые являются полями ввода текста. К полям ввода текста относятся элементы `input` со значением атрибута `type` равным `text`.

Синтаксис этого фильтра следующий:

```
| $("text")
```

В следующем примере выбирается поле ввода текста и присваивается содержимое в виде текста, и текст оформляется красным цветом.

```
| $("input:text").val("Введите ваше имя").css('color','red')
```

Выбор полей для ввода паролей

С помощью фильтра `password` можно выбрать элементы формы, которые являются полями ввода пароля. К полям ввода пароля относятся элементы `input` со значением атрибута `type` равным `password`.

Синтаксис этого фильтра следующий:

```
| $("password ")
```

В следующем примере выбирается поле ввода пароля, присваивается содержимое в виде текста и применяются рамка красного цвета.

```
| $("input:password").val("Введите ваш пароль").css('border','2px solid red')
```

Выбор полей для загрузки файлов

С помощью фильтра `file` можно выбрать элементы формы, которые являются полями для загрузки файлов. К полям для загрузки файлов относятся элементы `input` со значением атрибута `type` равным `file`.

Синтаксис этого фильтра следующий:

```
| $("file")
```

В следующем примере выбирается поле для загрузки файлов, и к нему применяется рамка красного цвета.

```
| $("input:file").css('border','2px solid red')
```

Выбор кнопки отправки формы в виде изображения

С помощью фильтра `image` можно выбрать элементы формы, которые являются кнопками в виде изображений. К кнопкам в виде изображений относятся элементы `input` со значением атрибута `type` равным `image`.

Синтаксис этого фильтра следующий:

```
| $("image")
```

В следующем примере выбирается кнопка в виде изображений, и к ней применяется рамка красного цвета.

```
| $("input:image ").css('border','2px solid red')
```

Выбор выпадающего меню.

С помощью фильтра `selected` можно выбрать элементы формы, которые являются выпадающими списками. К спискам относятся элементы `select`.

Синтаксис этого фильтра следующий:

```
| $("selected ")
```

Выделение выбранных элементов

С помощью фильтра `checked` можно выбрать элементы формы, которые были выбраны. К таким элементам относятся флажки и переключатели, которые были активизированы. Синтаксис этого фильтра следующий:

```
| $("checked ")
```

Выделение активных элементов

С помощью фильтра `enabled` можно выбрать незаблокированные элементы формы. Синтаксис этого фильтра следующий:

```
| $("enabled")
```

Выделение заблокированных элементов

С помощью фильтра `disabled` можно выбрать заблокированные элементы формы. Синтаксис этого фильтра следующий:

```
$("#disabled ")
```

Практические задания к теме 2

Практическое задание 2.1 Выбор элементов по классу

В этом задании показано, как можно выбрать элементы с помощью универсального селектора и класса. Создайте новый html файл и сохраните его под именем **practicejQuery2_01.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('body *').css('border','2px solid green')
$('.header').css('backgroundColor','red')
$('.header.logo').css('color','white').css('fontWeight','bold')
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; float:left;
}
</style>
</head>
<body>
<div class='header'>1</div><div class='header logo'>2</div><div>3</div><div
class='header'>4</div>
<div>5</div>
</body>
```

```
| </html>
```

Практическое задание 2.2 Выбор элементов по идентификатору и имени тега

В этом задании показано, как можно выбрать элементы с помощью идентификатора и имени тега.

Создайте новый html файл и сохраните его под именем **practicejQuery2_02.html**.

Введите следующий код:

```
| <!doctype html>
| <html>
| <head>
| <meta charset="windows-1251">
|     <title>Выбор элементов по классу</title>
|     <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
|     <script type="text/javascript">
|         window.onload=function()
|         {
|             $('body *').css('border','2px solid green')
|             $('#header').css('backgroundColor','green')
|             $('div').css('color','red').css('fontWeight','bold')
|             $('span').css('color','#408080').css('fontStyle','italic')
|         }
|     </script>
|     <style type="text/css">
|         div, span
|         {
|             display:block; padding:25px; margin:10px; float:left;
|         }
|     </style>
| </head>
|
| <body>
|     <div
| id='header'>1</div><span>2</span><div>3</div><div>4</div><div>5</div>
| </body>
```

```
| </html>
```

Практическое задание 2.3 Выбор смежных элементов

В этом задании показано, как можно выбрать смежные элементы.

Создайте новый html файл и сохраните его под именем **practicejQuery2_03.html**.

Введите следующий код:

```
| <!doctype html>
| <html>
| <head>
| <meta charset="windows-1251">
| <title>Выбор элементов по классу</title>
| <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
| <script type="text/javascript">
| window.onload=function()
| {
| $('body *').css('border','2px solid green')
| $('div+span').css('backgroundColor','green')
| $('span~div').css('color','red').css('fontWeight','bold')
| }
| </script>
| <style type="text/css">
| div, span
| {
|     display:block; padding:25px; margin:10px; float:left;
| }
| </style>
| </head>
| <body>
| <div id='header'>1</div><span>2</span><div>3</div><div>4</div><div>5</div>
| </body>
| </html>
```

Практическое задание 2.4 Выбор элементов по атрибуту и значению

В этом задании показано, как можно выбрать элементы по атрибуту и его значению.

Создайте новый html файл и сохраните его под именем **practicejQuery2_04.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
    window.onload=function()
    {
        $('[title='Яндекс
почта']').css('backgroundColor','green').css('color','white')
        $('body [title!='Яндекс почта']').css('backgroundColor','#7CBEBE')
        $('[title $=почта]').css('textDecoration','none').css('fontWeight','bold')
    }
</script>
<style type="text/css">
a
{
    display:block;padding:5px;margin:10px;    float:left; width:200px;
}
</style>
</head>
<body>
    <a href="http://www.yandex.ru" title="Яндекс почта">Яндекс</a>
    <a href="http://www.google.ru/" title="Google почта">Google</a>
    <a href="http://www.mail.ru/" title="Mail почта">Mail</a>
    <a href="http://www.yahoo.com/" title="Yahoo поисковая
система">Yahoo</a>
</body>
</html>
```

Практическое задание 2.5 Выбор элемента по нескольким атрибутам

В этом задании показано, как можно выбрать элементы по нескольким атрибутам и их значениям.

Создайте новый html файл и сохраните его под именем **practicejQuery2_05.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("[type=checkbox][name!=male]").attr("checked","checked")
}
</script>
<style type="text/css">
a
{
display:block;padding:5px;margin:10px; float:left; width:200px;
}
</style>
</head>

<body>
<label><input type="checkbox" name="male">Мужской</label>
<br>
<label><input type="checkbox" name="female">Женский</label>
<br>
<label><input type="checkbox" name="unknown"> Не указывать</label>
<br>
</body>
</html>
```

Практическое задание 2.6 Выбор четных и нечетных элементов

В этом задании показано, как можно выбрать четные и нечетные элементы. Создайте новый html файл и сохраните его под именем **practicejQuery2_06.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
    window.onload=function()
    {
        $('body tr#header:eq(0)').css('fontSize','20px').css('fontWeight','bold')
        $('#mail tr:even').css('backgroundColor','#cccccc')
        $('#mail tr:odd').css('backgroundColor','#E6E6E6')
    }
</script>
</head>
<body>
<table width="297" border="0" id="mail">
  <tr id="header">
    <td width="35"><input type="checkbox" onClick="sel()"/></td><td
width="109">Тема </td>
    <td width="34">Дата</td>
  </tr>
  <tr>
    <td><input type="checkbox" /></td><td>Привет</td><td>20 марта</td>
  </tr>
  <tr>
    <td><input type="checkbox" checked="checked"/></td><td>Как дела</td><td>19
марта</td>
  </tr>
  <tr>
    <td></td></tr>
  </tr>
</table>
```

```

    <td><input type="checkbox" /></td><td>Встреча</td><td>18 марта</td>
  </tr>
  <tr>
    <td><input type="checkbox" /></td> <td>Пока</td><td>17 марта</td>
  </tr>
</table>
</body>
</html>

```

Практическое задание 2.7 Выбор элементов до и после указанного

В этом задании показано, как можно выбрать элементы до и после указанного. Создайте новый html файл и сохраните его под именем **practicejQuery2_07.html**. Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
    {
        $("#mail tr:gt(2)").css('backgroundColor','red')
        $("#mail tr:lt(2)").css('backgroundColor','green')
    }
</script>
</head>
<body>
<table width="297" border="0" id="mail">
  <tr id="header">
    <td width="35"><input type="checkbox" onClick="sel()"/></td><td
width="109">Тема </td>
    <td width="34">Дата</td>
  </tr>

```

```
<tr>
  <td><input type="checkbox" /></td><td>Привет</td><td>20 марта</td>
</tr>
<tr>
  <td><input type="checkbox" checked="checked"/></td><td>Как дела</td><td>19
марта</td>
</tr>
<tr>
  <td><input type="checkbox" /></td><td>Встреча</td><td>18 марта</td>
</tr>
<tr>
  <td><input type="checkbox" /></td> <td>Пока</td><td>17 марта</td>
</tr>
</table>
</body>
</html>
```

Практическое задание 2.8 Выбор первого и последнего элемента

В этом задании показано, как можно выбрать первый и последний элементы. Создайте новый html файл и сохраните его под именем **practicejQuery2_08.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#mail tr:last').css('backgroundColor','#95AFC1')
$('#mail tr:first').css('backgroundColor','#95AFC1')
}
</script>
```

```

</head>
<body>
<table width="297" border="0" id="mail">
  <tr id="header">
    <td width="35">
      <input type="checkbox" onClick="sel()"/>
    </td>
    <td width="109">Тема </td><td width="34">Дата</td>
  </tr>
  <tr>
    <td><input type="checkbox" /></td><td>Привет</td><td>20 марта</td>
  </tr>
  <tr>
    <td><input type="checkbox" checked="checked" /></td><td>Как дела</td><td>19
марта</td>
  </tr>
  <tr>
    <td><input type="checkbox" /></td><td>Встреча</td><td>18 марта</td>
  </tr>
  <tr>
    <td><input type="checkbox" /></td>
    <td>Пока</td>
    <td>17 марта</td>
  </tr>
</table>
</body>
</html>

```

Практическое задание 2.9 Инвертирование выделения элементов

В этом задании показано, как можно инвертировать элементы.

Создайте новый html файл и сохраните его под именем **practicejQuery2_09.html**.

Введите следующий код:

```

<!doctype html>
<html>

```

```
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
    {
        $(':header:not(.content :header)').css('backgroundColor','#95AFC1')
        $(':header:not(body>:header)').css('backgroundColor','red')
    }
</script>
</head>
<body>
<h1>Заголовок первого уровня</h1>
<div class="content">
    <h2>Заголовок второго уровня</h2>
    <h3>Заголовок третьего уровня</h2>
</div>
<h4>Заголовок четвертого уровня</h4>
</body>
</html>
```

Практическое задание 2.10 Выбор первого и последнего дочернего элемента

В этом задании показано, как можно выбрать первый и последний дочерний элемент.

Создайте новый html файл и сохраните его под именем **practicejQuery2_10.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
```

```

window.onload=function()
{
$('div :first-child').css('border','2px solid green')
$('div :last-child').css('border','2px solid red')
}
</script>
<style type="text/css">
div span { padding:25px; margin:10px; float:left; display:block;}
.two { clear:both;}
</style>
</head>
<body>
<div class='one'>

<span>1</span><span>2</span><span>3</span><span>4</span><span>5</span>
</div>
<div class='two'>

<span>6</span><span>7</span><span>8</span><span>9</span><span>0</span>
</div>
</body>
</html>

```

Практическое задание 2.11 Выбор элемента по содержимому

В этом задании показано, как можно выбрать элемент по содержимому.

Создайте новый html файл и сохраните его под именем **practicejQuery2_11.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">

```

```
window.onload=function()
    {
        $(".one :contains('Ноль')).css('border','2px solid red')
        $(".one :not(:contains('Ноль'))").css('border','2px solid green')
        $(".two :parent").css('backgroundColor','red')
        $(".two :empty").css('backgroundColor','green')
        $(".two :has(i)").css('border','2px solid green')
    }
</script>
<style type="text/css">
div span { padding:25px; margin:10px; float:left; display:block;}
.two { clear:both;}
</style>
</head>
<body>
<div class='one'>
<span>Ноль(0)</span><span>Один</span><span>Два</span><span>Три</span><span>Четыре</span>
</div>
<div
class='two'><span>Пять</span><span><i></i></span><span></span><span>Восемь
</span><span>
Девять</span>
</div>
</body>
</html>
```

Практическое задание 2.12 Выбор кнопок формы

В этом задании показано, как можно выбрать кнопки формы.

Создайте новый html файл и сохраните его под именем **practicejQuery2_12.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
```

```

<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$(".button").css('border','2px solid red')
$(".submit").css('border','2px solid green')
$(".reset").css('border','2px solid yellow')
}
</script>
</head>
<body>
<input type="button" value="Кнопка"><input type="submit" value="Кнопка
отправить"><input type="reset" value="Кнопка очистить">
</body>
</html>

```

Практическое задание 2.13 Выбор элементов формы

В этом задании показано, как можно выбрать элементы формы.

Создайте новый html файл и сохраните его под именем **practicejQuery2_13.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Выбор элементов по классу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$(".button").css('border','2px solid red')
$(".submit").css('border','2px solid green')
$(".reset").css('border','2px solid yellow')
}

```

```
$("input:checkbox").attr('checked','checked')
$("input:radio").attr('disabled','disabled')
$("input:text").val("Введите ваше имя").css('color','red')
$("input:password").val("Введите ваш пароль").css('border','2px solid red')
$("input:file").css('border','2px solid red')
}
</script>
</head>
<body>
<input type="text"><br>
<input type="password"><br>
<input type="checkbox" value="sport">Спорт<br>
<input type="checkbox" value="books">Чтение<br>
<input type="checkbox" value="cinema">Кинопросмотры<br>
<input type="radio" name="gender" value="male">Мужской<br>
<input type="radio" name="gender" value="female " id="gender_1">Женский<br>
<input type="radio" name="gender" value="unknown" id="gender_2">Не
указывать<br>
<input name="screen" type="file"><br>
<input type="button" value="Кнопка">
<input type="submit" value="Кнопка отправить">
<input type="reset" value="Кнопка очистить">
</body>
</html>
```

Самостоятельная работа к теме 2

Самостоятельная работа 2.1 Выбор элементов

Выберите элемент 7 разными способами.

Тема 3. Манипуляции с элементами, их свойствами и содержимым.

Манипуляции со стилями.

Функция CSS.

Метод `css` возвращает или изменяет значения `css`-величин у выбранных элементов веб-страницы. Функция имеет четыре варианта использования.

В следующем примере возвращается значение свойства `background-color` **первого** `div` элемента на странице

```
$('#div').css('backgroundColor')
```

В следующем примере элементу с классом `header` назначается `css`-свойство `border`.

```
$('.header').css('border', '3px solid green')
```

В следующем примере присваивается несколько `css`-свойств элементу с классом `logo`

```
$('.logo').css({'border':'3px solid yellow', 'color':'white'})
```

В следующем примере при наступлении события `click` (в JavaScript `onclick`) на `div`-элементе его фоновый цвет меняется на голубой. Давайте подробнее рассмотрим этот пример. В начале мы создаем массив из всех `div`-элементов на веб-странице (`$("#div")`). Далее на эти `div`-элементы устанавливается обработчик события (`.on`), который позволяет применять событие несколько раз событие рассматривается в следующей главе). Далее объявляется событие `click`, при котором запускается неназванная функция (`"click", function()`). При наступлении события на одном из `div`-элементов этот элемент захватывается с помощью оператора `this` (`$(this)`). И в результате к `div`-элементу, на котором был произведен щелчок мышью, присваивается голубой цвет фона.

```
$("#div").on( "click", function() {  
    $(this).css( "backgroundColor", "blue" );  
});
```

Определение наличия класса у элемента.

С помощью метода `hasClass` можно определить присвоен ли указанный класс выбранным элементам. Метод возвращает `true` при нахождении класс и `false` при его отсутствии.

В следующем примере определяется наличие класса `logo` в `div`-элементах на веб-странице.

```
$('#div').hasClass('logo')
```

Добавление класса к элементу

С помощью метода `addClass` можно добавить класс к выбранным элементам. При

добавлении нового к класса к элементу с уже присвоенным классом, старый класс не удаляется.

В следующем примере ко всем div-элементам на странице присваивается класс square.

```
$('#div').addClass('square')
```

Так же можно добавлять несколько классов одновременно. В следующем примере к четным div-элементам на странице применяются два класса font и border.

```
$('#div:even').addClass('font border')
```

Удаление класса из элемента

С помощью метода removeClass можно удалить класс из выбранных элементов.

В следующем примере будет удален класс square у последнего div-элемента.

```
$('#div:last').removeClass('square')
```

Управление высотой элемента

Для работы с высотой элемента в jQuery существуют три функции; height(), innerHeight() и innerHeight(). Все эти функции возвращают значение высоты элемента, если элементов было выбрано несколько. На рисунке 3.1 наглядно показана область действия этих функций.

Первая функция height() возвращает и устанавливает высоту элемента без учета отступов (margin и padding) и толщины рамки (border). В следующем примере возвращается высота div-элемента с идентификатором header и назначается новая. Возвращается значение без единиц измерения и назначается так же без единиц измерения.

```
alert($("#header").height())
$("#header").height('40')
```

Вторая innerHeight() возвращает высоту элемента с учетом размера внутренних отступов (padding). В следующем примере возвращается значение высоты и внутреннего отступа.

```
alert($(".logo").innerHeight() )
```

Третья функция outerHeight(margin) — высота элемента с учетом внутренних отступов, рамки (border) и при необходимости внешних отступов (margin). В первом примере вернется значение высоты с внутренними отступами и границами. Во втором примере вернется значение высоты с внутренними отступами, границами и внешними отступами.

```
alert($(".header").outerHeight())
alert($(".header").outerHeight(true) ) внутренними
```

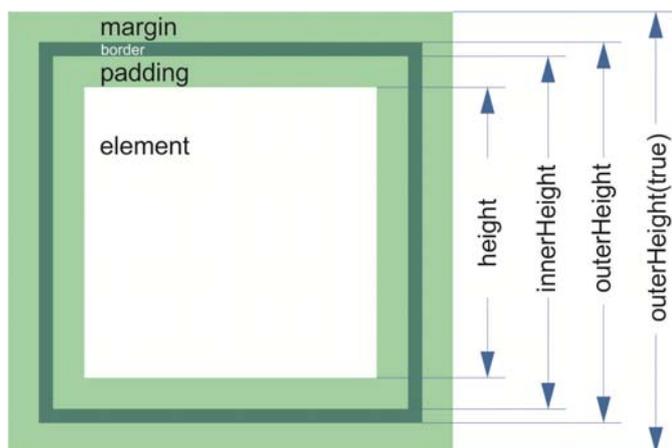


Рисунок 3.1 Функции высоты элемента.

Управление шириной элемента

Для работы с шириной элемента в jQuery существуют три функции; `width()`, `innerWidth()` и `innerWidth()`. Все эти функции возвращают значение ширины первого элемента, если элементов было выбрано несколько. На рисунке 3.2 наглядно показано область действия этих функций.

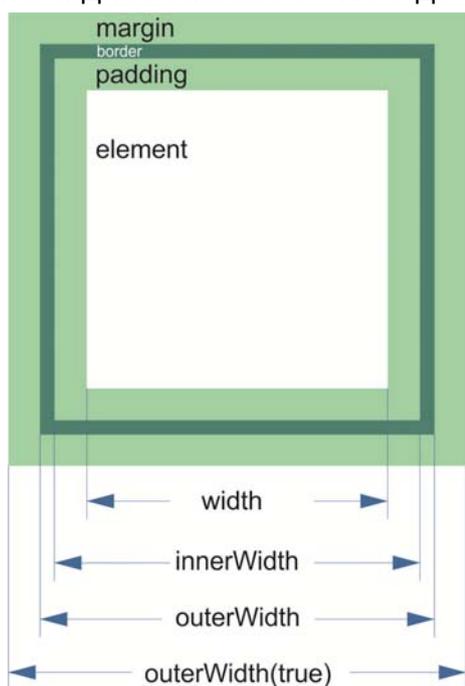


Рисунок 3.2 Функции ширины элемента.

Первая функция `width()` возвращает и устанавливает ширину элемента без учета отступов (`margin` и `padding`) и толщины рамки (`border`). В следующем примере возвращается ширина `div`-элемента с идентификатором `header` и назначается новая. Возвращается значение без единиц измерения и назначается так же без единиц измерения.

```
alert($("#header").width())
$("#header").width('40')
```

Вторая `innerWidth()` возвращает ширину элемента с учетом размера внутренних

отступов (padding). В следующем примере возвращается значение высоты и внутреннего отступа.

```
alert($(".logo").innerWidth() )
```

Третья функция `outerWidth(margin)` — высота элемента с учетом внутренних отступов, рамки (border) и при необходимости внешних отступов (margin). В первом примере вернется значение высоты с внутренними отступами и границами. Во втором примере вернется значение высоты с внутренними отступами, границами и внешними отступами.

```
alert($(".header").outerWidth ())
alert($(".header").outerWidth (true) )
```

Определение позиции элемента

Для определения позиции элемента на странице в jQuery существует две функции `offset` и `position`. Функции могут возвращать позицию элемента от верхнего и от левого края. Функция `offset` возвращает координаты относительно начала страницы, а `position` относительно ближайшего родителя, у которого задан тип позиционирования (css-свойство `position` равно `relative` или `absolute` или `fixed`).

Так же при помощи функции `offset` можно изменить положение элемента на веб-странице.

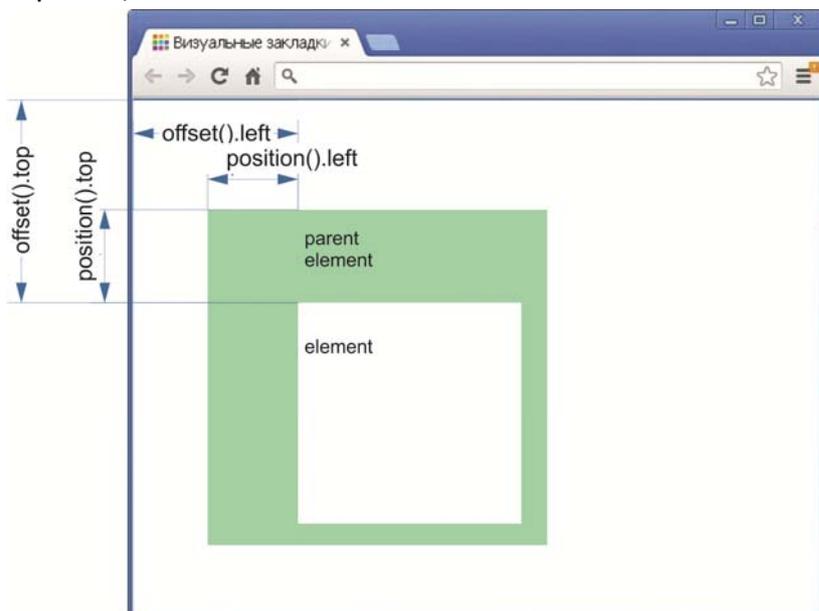


Рисунок 3.3 Положение элемента на странице

В следующем примере возвратится значение положения объекта относительно верхнего края родителя.

```
$('.header').position.left
```

В следующем примере элемент с классом `header` переместится относительно родителя сверху на 120 пикселей и слева на 100 пикселей.

```
$('.header').offset({top:120, left:100})
```

Определение положения полосы прокрутки

Для определения и установки значения вертикального и горизонтального скроллинга используются методы `scrollTop` и `scrollLeft`.

В следующем примере для элемента с идентификатором `header` будет установлена вертикальная прокрутка 200 пикселей, а затем выведено значение вертикальной прокрутки с помощью метода `alert`.

```
$("#header").scrollTop('200')  
alert($("#header").scrollTop())
```

В следующем примере для элемента с идентификатором `header` будет установлена горизонтальная прокрутка 300 пикселей, а затем выведено значение горизонтальной прокрутки с помощью метода `alert`.

```
$("#header").scrollLeft('300')  
alert($("#header").scrollLeft())
```

Добавление или удаление класса.

Для добавления и удаления класса можно использовать метод `toggleClass()`.

В следующем примере при щелчке мыши на `div`-элементе существующий класс будет заменен на класс `blue`. При повторном щелчке класс `blue` будет удален.

```
$("#div").click(function () {  
    $(this).toggleClass("blue");  
})
```

В следующем примере при щелчке мыши на `div`-элементе существующий класс будет заменен на класс `blue`. При повторном щелчке класс `blue` удален не будет за счет установленного булевого значения `true`.

```
$("#div").click(function () {  
    $(this).toggleClass("blue", true);  
})
```

Управление атрибутами.

Возвращение и изменение атрибута элемента

С помощью метода `attr` можно узнать значение атрибута, а также установить новый атрибут для элемента. Если выбрано несколько элементов, то изменения будут осуществляться с первым из выбранных.

В следующем примере вернётся значение атрибута `title` у первой ссылки в выборке.

```
alert($('a').attr('title'))
```

В следующем примере к первому div-элементу будет назначен атрибут class со значением blue.

```
$( 'div:eq(0)' ).attr( 'class', 'blue' )
```

В следующем примере будет выбрана последняя ссылка на веб-странице и ей будет назначено два атрибута со значениями.

```
$( "a:eq(-1)" ).attr( { "title": "Ссылка два", "class": "greenWhite" } )
```

Удаление атрибута элемента

С помощью метода removeAttr можно удалить атрибут у выбранного элемента. В следующем примере у div-элемента с индексом 3 будет удален атрибут class.

```
$( 'div:eq(3)' ).removeAttr( 'class' )
```

Изменение свойств атрибутов элемента

С помощью метода prop можно вернуть или назначить значение атрибута. Этот метод рекомендуется использовать для атрибутов, не имеющих значения (checked, disabled), в других случаях лучше использовать метод attr.

В следующем примере вернется значение атрибута value третьего тега input на веб-странице.

```
alert( $( 'input:eq(2)' ).prop( 'value' ) )
```

В следующем примере третий тег input на странице станет неактивным.

```
$( 'input:eq(2)' ).prop( 'disabled', 'true' )
```

В следующем примере для первого input-элемента присваивается два значения disabled (блокировка элемента) и checked (выделение элемента)

```
$( "input:first" ).prop( { 'disabled': true, 'checked': true } );
```

Удаление свойства элемента

С помощью метода removeProp можно удалить значение атрибута. Этот метод не рекомендуется использовать для атрибутов, которые были изначально присвоены элементу.

В следующем примере будет удалено значение атрибута value у первого input-элемента.

```
$( "input:first" ).removeProp( 'value' )
```

Замена и клонирование элементов

Замена элементов и содержимого

Для замены элемента и его содержимого используются два метода replaceWith иreplaceAll. Отличаем этих методов в очередности написания вставляемого элемента и элемента для вставки.

В следующем примере используется метод `replaceWith` для замены элементов. В начале примера указывается, какой элемент будет заменен, это последний `div`-элемент на веб-странице, а затем с помощью метода `replaceWith` он заменяется на пункт списка.

```
$("#div:last").replaceWith("<li>список</li>");
```

В следующем примере для замены элемента используется метод `replaceAll`. В отличие от предыдущего примера в этом примере сначала указывается элемент, на который будет заменен указанный в конце элемент.

```
$("#<li>список</li>").replaceAll("div:first")
```

Клонирование элементов и их содержимого

Для клонирования элемента и его содержимого используются метод `clone`. Элемент копируется со всеми вложенными элементами.

В следующем примере будет клонирован второй `div`-элемент на странице и с помощью метода `appendTo` добавлен в конец тега `body`. Метод `appendTo` будет рассмотрен в следующем разделе.

```
$("#div:eq(1)").clone().appendTo("body")
```

Удаление элемента и его содержимого

Удаление элементов

Для удаления элементов используются методы `remove` и `detach`. Отличие этих методов заключается в том, что метод `detach` удаляет элементы с возможностью их восстановления.

В следующем примере удаляется последний `div`-элемент на веб-странице с помощью метода `remove`.

```
$("#div:last").remove();
```

В следующем примере удаляется последний `div`-элемент на веб-странице с помощью метода `detach`.

```
$("#div:last").detach();
```

Удаление содержимого элемента

Для удаления содержимого элемента используется метод `empty`.

В следующем примере будет удалено содержимое у третьего `div`-элемента на веб-странице.

```
$('#div:eq(2)').empty()
```

Удаление родительских элементов

С помощью метода `unwrap` можно удалить родительский элемент указанного элемента. Содержимое удаленного элемента останется нетронутым.

В следующем примере будет удален родительский элемент i-элемента.

```
$(i).unwrap()
```

Добавление содержимого элемента

Работа с html-содержимым элемента

С помощью функции html() можно вернуть или изменить содержимое элемента.

В следующем примере возвращается содержимое первого div-элемента.

```
alert($("#div:first").html())
```

В следующем примере будет удалено содержимое второго div-элемента и добавлено содержимое в виде i-элемента и текста "Новый элемент".

```
$("#div:eq(1)").html('<i>Новый элемент</i>')
```

Изменение текстового содержимого элемента

С помощью метода text можно вернуть значение текстового содержимого или заменить его. Отличие от метода html в том, что если попробовать вставить сам элемент, то теги вставятся, как текст.

С помощью следующего примера можно вернуть текстовое содержимое второго div-элемента на странице.

```
alert($("#div:eq(1)").text())
```

В следующем примере будет заменено текстовое содержимое второго div-элемента на новое.

```
$("#div:eq(1)").text('Новое содержимое')
```

Изменение содержимого атрибута value

С помощью метода val можно вернуть или изменить содержимое элементов форм.

```
$("#input[type=text]").val()
```

Добавление содержимого в конец элемента

С помощью методов append и appendTo можно добавить содержимое непосредственно в конец элемента. Отличие этих методов в очередности написания вставляемого элемента и элемента для вставки.

В следующем примере используется метод append для вставки элементов. В начале примера указывается куда элемент будет вставлен, это первый div-элемент на веб-странице, а затем с помощью метода append в конец этого элемента вставляется пункт списка.

```
$("#div:first").append("<li>Тест</li>")
```

В следующем примере используется метод appendTo для вставки элементов. В начале примера определяется элемент для вставки, это пункт списка, а далее с

помощью метода `appendTo` определяется в какой элемент он будет вставлен.

```
$("#<li>appendTo</li>").appendTo($("#div:last"))
```

Добавление содержимого в начало элемента

С помощью методов `prepend` и `prependTo` можно добавить содержимое непосредственно в начало элемента. Отличие этих методов в очередности написания вставляемого элемента и элемента для вставки.

В следующем примере используется метод `prepend` для вставки элементов. В начале примера, в какой элемент будет вставлен элемент, это второй `div`-элемент на веб-странице, а затем с помощью метода `prepend` в начало этого элемента вставляется пункт списка.

```
$("#div:eq(1)").prepend("<li>prepend</li>")
```

В следующем примере используется метод `prependTo` для вставки элементов. В начале примера определяется элемент для вставки, это пункт списка, а далее с помощью метода `prependTo` определяется в какой элемент он будет вставлен.

```
$("#<li>prependTo </li>").prependTo($("#div: eq(2)"))
```

Добавление содержимого после элемента

С помощью методов `after` и `insertAfter` можно вставить элемент после выбранного элемента. Отличие этих методов в очередности написания вставляемого элемента и элемента для вставки.

В следующем примере используется метод `after` для вставки элементов. В начале примера указывается после какого элемента будет вставлен элемент, это первый `div`-элемент на веб-странице, а затем с помощью метода `after` после этого элемента вставляется пункт списка.

```
$("#div:first").after("<li>Текст</li>");
```

В следующем примере используется метод `insertAfter` для вставки элементов. В начале примера определяется элемент для вставки, это пункт списка, а далее с помощью метода `insertAfter` определяется после какого элемента он будет вставлен.

```
$("#<li>Текст</li>").insertAfter($("#div:last"))
```

Добавление содержимого до элемента

С помощью методов `before` и `insertBefore` можно вставить элемент до выбранного элемента. Отличие этих методов в очередности написания вставляемого элемента и элемента для вставки.

В следующем примере используется метод `before` для вставки элементов. В начале примера указывается до какого элемента будет вставлен элемент, это первый `div`-элемент на веб-странице, а затем с помощью метода `before` до этого элемента вставляется пункт списка.

```
$("#div:first").before("<li>Тест</li>");
```

В следующем примере используется метод `insertBefore` для вставки элементов. В начале примера определяется элемент для вставки, это пункт списка, а далее с помощью метода `insertBefore` определяется до какого элемента он будет вставлен.

```
$("#<li>Текст</li>").insertBefore($("#div:last"))
```

Обертывание элементов.

С помощью методов `wrap` и `wrapAll` можно обернуть элемент в другой. Различие методов заключается в том, что `wrap` оборачивает каждый выбранный элемент по отдельности, а `wrapAll` оборачивает все элементы целиком.

В следующем примере все `div`-элементы будут размещены внутри `li`-элементов.

```
$("#div").wrap("<li></li>");
```

В следующем примере все `li`-элементы будут обернуты в один `div`-элемент.

```
$("#li").wrapAll("<div class='top'></div>")
```

Обертывание содержимого элемента

С помощью метода `wrapInner` можно обернуть содержимое указанного тега.

В следующем примере содержимое всех `li`-элементов на веб-странице будет обернуто в тег `i`.

```
$("#li").wrapInner("<i></i>");
```

Практические задания к теме 3

Практическое задание 3.1 Использование функции css

В этом задании показано, как можно использовать функцию css для изменения внешнего вида объекта.

Создайте новый html файл и сохраните его под именем **practicejQuery3_01.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.1 Использование функции css </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
alert($('div').css('backgroundColor'))
$('.header').css('border', '3px solid green')
$('.logo').css({'border':'3px solid yellow', 'color':'white'})
$("div").on( "click", function()
{
(this).css( "backgroundColor", "blue" );
});
}
</script>
<style type="text/css">
div
{padding:25px; margin:10px; float:left; background-color:red;
}
</style>
</head>
<body>
<div class='header'>1</div><div class='logo'>2</div><div>3</div><div
```

```
class='header'>4</div><div>5</div>
</body>
</html>
```

Практическое задание 3.2 Добавление и удаление класса к элементу

В этом задании показано, как можно определить назначенный элементу класс, добавить новый и удалить класс.

Создайте новый html файл и сохраните его под именем **practicejQuery3_02.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.2 Добавление и удаление класса к элементу
</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
alert($('div').hasClass('logo'))
$('div').addClass('square')
$('div:even').addClass('font border')
$('div:last').removeClass('square')
$('div:last').addClass('squareGreen')
}
</script>
<style type="text/css">
.square
{
padding:25px; margin:10px; float:left; background-color:red;
}
.squareGreen
{
```

```
padding:25px; margin:10px; float:left; background-color:green;
}
.border
{
border:3px dotted #00CC99;
}
.font
{
font-weight:bold; color:white;
}
</style>
</head>
<body>
<div id='header'>1</div><div class='logo'>2</div><div>3</div><div
class='header'>4</div><div>5</div>
</body>
</html>
```

Практическое задание 3.3 Добавление класса к элементу

В этом задании показано, как можно определить высоту и ширину элемента и назначить новую.

Создайте новый html файл и сохраните его под именем **practicejQuery3_03.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.3 Добавление класса к элементу</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
/*Определение высоты элемента*/
alert('Высота элемента: '+ $("#header").height())
```

```

$("#header").height('40')
alert('Высота элемента с внутренними полями: '+$("#.logo").innerHeight() )
alert('Высота элемента с внутренними полями и границами:
'+$("#.header").outerHeight() )
alert('Высота элемента с внутренними полями, границами и отступами:
'+$("#.header").outerHeight(true) )
/*Определение ширины элемента*/
alert('Ширина элемента: '+ $("#header").width())
$("#header").width('40')
alert('Ширина элемента с внутренними полями: '+$("#.logo").innerWidth() )
alert('Ширина элемента с внутренними полями и границами:
'+$("#.header").outerWidth() )
alert('Ширина элемента с внутренними полями, границами и отступами:
'+$("#.header").outerWidth(true) )
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; border:2px solid #0C6; float:left; background-
color:red;
}
</style>
</head>
<body>
<div id='header'>1</div><div class='logo'>2</div><div>3</div><div
class='header'>4</div><div>5</div>
</body>
</html>

```

Практическое задание 3.4 Изменение положения на странице

В этом задании показано, как можно изменить положение элемента на странице. Создайте новый html файл и сохраните его под именем **practicejQuery3_04.html**. Введите следующий код:

```
<!doctype html>
```

```
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.4 Изменение положения на странице </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('.header').offset({top:120, left:100})
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; border:2px solid #0C6; float:left; background-color:red;
}
</style>
</head>
<body>
<div id='header'>1</div><div class='logo'>2</div><div>3</div><div
class='header'>4</div><div>5</div>
</body>
</html>
```

Практическое задание 3.5 Изменение положения прокрутки на странице

В этом задании показано, как можно изменить положение прокрутки элемента, как по вертикали, так и по горизонтали.

Создайте новый html файл и сохраните его под именем **practicejQuery3_05.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.5 Изменение положения прокрутки на странице
```

```

</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("#header").scrollTop('200')
alert($("#header").scrollTop())
$("#header").scrollLeft('300')
alert($("#header").scrollLeft())

}
</script>
<style type="text/css">
div
{
background:#CCCCCC none repeat scroll 0 0; border:3px solid #666666; margin:5px;
padding:5px; position:relative; width:200px; height:100px; overflow:auto;
}
p { margin:10px;padding:5px;border:2px solid #666;width:1000px;height:1000px; }
</style>
</head>
<body>
<div id='header'>
    <p>1</p>
</div>
</body>
</html>

```

Практическое задание 3.6 Изменение класса элемента

В этом задании показано, как можно изменить класс элемента.

Создайте новый html файл и сохраните его под именем **practicejQuery3_06.html**.

Введите следующий код:

```
<!doctype html>
```

```
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.6 Изменение класс элемента </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("div").click(function () {
    $(this).toggleClass("blue");
    })
}
</script>
<style type="text/css">
div
{
    padding:25px; margin:10px; float:left; background-color:red;
}
.blue
{ background-color:blue;}
</style>
</head>
<body>
<div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></body>
</html>
```

Практическое задание 3.7 Добавление и удаление атрибута

В этом задании показано, как можно добавить и удалить атрибут.

Создайте новый html файл и сохраните его под именем **practicejQuery3_07.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
```

```
<meta charset="windows-1251">
<title> Практическое задание 3.7 Добавление и удаление атрибута </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
alert($('a').attr('title') )
$('div:eq(0)').attr('class','blue')
$("a:eq(1)").attr({"title":"Ссылка два", "class":"greenWhite"})
$('div:eq(3)').removeAttr('class')
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; float:left; background-color:red;
}
.blue
{
background-color:#3CF;
}
.greenWhite
{
border:2px solid #FFF;
}
</style>
</head>
<body>
<div>
<a href="#" title="Первая ссылка">1</a>
</div>
<div>
```

```
<a href="#" title="Вторая ссылка ">2</a>
</div>
<div>
3
</div>
<div class="blue">
4
</div>
<div>
5
</div>
</body>
</html>
```

Практическое задание 3.8 Добавление и удаление значений атрибута

В этом задании показано, как можно добавить и удалить значение атрибута. Создайте новый html файл и сохраните его под именем **practicejQuery3_08.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.8 Добавление и удаление значений
атрибута</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
alert($('input:eq(2)').prop('value'));
$('input:eq(2)').prop('disabled', 'true');
$("input:first").prop({'disabled':true, 'checked':true});
$("input:first").removeProp('value')
}
}
```

```

</script>
</head>
<body>
<form name="form1" method="post" action="">
  <label>
    <input type="checkbox" name="hobby" value="Sport" >
    Спорт</label>
  <br>
  <label>
    <input type="checkbox" name="hobby" value="Reading" >
    Чтение </label>
  <br>
  <label>
    <input name="hobby" type="checkbox" value="Music" checked >
    Музыка</label>
  <br>
  <label>
    <input type="checkbox" name="hobby" value="Move" >
    Кино</label>
  <br>
</form>
</body>
</html>

```

Практическое задание 3.9 Замена и клонирование элемента

В этом задании показано, как можно добавить и удалить значение атрибута.

Создайте новый html файл и сохраните его под именем **practicejQuery3_09.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 3.9 Замена и клонирование элемента</title>

```

```
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("div:last").replaceWith("<li>список</li>");
    $("<li>список</li>").replaceAll("div:first")
    $("div:eq(1)").clone().appendTo("body")
}
</script>
<style type="text/css">
div
{
    padding:25px; margin:10px; float:left; background-color:red;
}
li
{
    padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
<body>
<div><a href="#" title="Первая ссылка">1</a></div>
<div><a href="#" title="Вторая ссылка ">2</a></div>
<div>3</div>
<div class="blue">4</div>
<div>5</div>
</body>
</html>
```

Практическое задание 3.10 Удаление элементов и их содержимого

В этом задании показано, как можно удалить элементы и их содержимое.

Создайте новый html файл и сохраните его под именем **practicejQuery3_10.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 3.10 Удаление элементов и их содержимого</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
divLast=$('#div:last')
divFirst=$('#div:first')
$(divLast).remove();
$(divFirst).detach();
divLast.prependTo('body')
divFirst.appendTo('body')
$('#div:eq(2)').empty()
$('i').unwrap()
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; float:left; background-color:red;
}
li
{
padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
```

```
<body>
<div><a href='#' title="Первая ссылка">1</a></div>
<div><a href='#' title="Вторая ссылка ">2</a></div>
<div>3</div>
<div class="blue"><i>4</i></div>
<div>5</div>
</body>
</html>
```

Практическое задание 3.11 Изменение содержимого элемента

В этом задании показано, как можно изменить содержимое элемента.

Создайте новый html файл и сохраните его под именем **practicejQuery3_11.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 3.11 Изменение содержимого элемента</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
alert($("#div:first").html())
$("#div:eq(0)").html('<i>Новый элемент</i>')
alert($("#div:eq(1)").text())
$("#div:eq(1)").text('Новое содержимое')
}
</script>
<style type="text/css">
div
{
padding:25px; margin:10px; float:left; background-color:red;
}
li
```

```

{
    padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
<body>
<div><a href='#' title="Первая ссылка">1</a></div>
<div><a href='#' title="Вторая ссылка">2</a></div>
<div>3</div>
<div class="blue"><i>4</i></div>
<div>5</div>
</body>
</html>

```

Практическое задание 3.12 Добавление содержимого в начало и конец элемента

В этом задании показано, как можно добавить содержимое в начало или конец документа.

Создайте новый html файл и сохраните его под именем **practicejQuery3_12.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.12 Добавление содержимого в начало и конец
элемента </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("div:first").append("<li>append</li>");
    $("<li>appendTo</li>").appendTo($("<div:last>"))
    $("div:eq(1)").prepend("<li>prepend</li>")
    $("<li> prependTo </li>").prependTo($("<div:eq(2)>"))
}

```

```
}
</script>
<style type="text/css">
div
{
    padding:25px; margin:10px; float:left; background-color:red;
}
li
{
    padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
<body>
<div><a href="#" title="Первая ссылка">1</a></div>
<div><a href="#" title="Вторая ссылка ">2</a></div>
<div>3</div>
<div class="blue"><i>4</i></div>
<div>5</div>
</body>
</html>
```

Практическое задание 3.13 Добавление элемента до и после выбранного элемента

В этом задании показано, как можно добавить элемент до и после выбранного. Создайте новый html файл и сохраните его под именем **practicejQuery3_13.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title> Практическое задание 3.13 Добавление элемента до и после выбранного
элемента </title>
```

```

<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
  $("div:first").after("<li>1.a</li>");
  $("<li>5.a</li>").insertAfter($("div:last"))
  $("div:first").before("<li>0</li>");
  $("<li>4.a</li>").insertBefore ($("div:last"))
}
</script>
<style type="text/css">
div
{
  padding:25px; margin:10px; float:left; background-color:red;
}
li
{
  padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
<body>
<div><a href="#" title="Первая ссылка">1</a></div>
<div><a href="#" title="Вторая ссылка ">2</a></div>
<div>3</div>
<div class="blue"><i>4</i></div>
<div>5</div>
</body>

```

Практическое задание 3.14 Обёртывание элемента и его содержимого

В этом задании показано, как можно обернуть элемент и его содержимое.

Создайте новый html файл и сохраните его под именем **practicejQuery3_14.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 3.14 Обертывание элемента и его
содержимого</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("div").wrap("<li></li>");
    $("li").wrapAll("<div class='top'></div>")
    $("li").wrapInner("<i></i>");
}
</script>
<style type="text/css">
div
{
    padding:25px;margin:10px;float:left;background-color:red;
}
li
{
    padding:25px;margin:10px;float:left;background-color:#3CC;display:block;
}
.top
{
    padding:25px;    margin:10px; float:left; background-color:#3C3;
display:block;
}
</style>
```

```
</head>
<body>
<div><a href="#" title="Первая ссылка">1</a></div>
<div><a href="#" title="Вторая ссылка ">2</a></div>
<div>3</div>
<div class="blue"><i>4</i></div>
<div>5</div>
</body>
</html>
```

Самостоятельная работа к теме 3

Самостоятельная работа 3.1 Выбор элементов

Создайте таблицу из 4-х строк и 3-х столбцов. Добавьте в первый столбец флажки, во второй и третий текст. С помощью события `click` (практика 3.6) запустите функцию, добавляющую еще одну строку в конце таблицы.

Тема 4. События

События играют важную роль в разработке веб-страницы. Часто необходимо реагировать на действие пользователя, а не просто выполнять скрипт автоматически при его загрузке. События jQuery во многом сходны с событиями JavaScript, но все события jQuery действуют одинаково во всех браузерах, то есть они кроссбраузерны.

Для того, что бы разобраться в событиях jQuery вспомним, как вообще действуют события.

Веб-страница обладает вложенной иерархической структурой. Элементы вкладываются друг в друга вплоть до элемента body, который в свою очередь вложен в элемент html.

При возникновении события на вложенном элементе, далее то же событие возникает на его родительских элементах и так до верхнего элемента. Такое поведение событий называется всплывающим, так как событие поднимается вверх по структуре веб-страницы. В практическом задании 4.1 продемонстрировано всплытие событий.

Объект event

Объект event содержит данные о произошедшем событии. Передается во все обработчики событий, которые были установлены средствами jQuery. Объект event отличается от стандартных объектов событий, которые получают обработчики, установленные обычными средствами javascript. Некоторые поля подвергаются изменению для обеспечения кроссбраузерности. В jQuery в event добавлены дополнительные поля и методы.

Следующие поля присутствуют в каждом объекте: altKey, attrChange, attrName, bubbles, button, cancelable, charCode, clientX, clientY, ctrlKey, currentTarget, data, detail, eventPhase, fromElement, handler, keyCode, layerX, layerY, metaKey, namespace (только с jQuery-1.4.3 и старше), newValue, offsetX, offsetY, originalTarget, pageX, pageY, prevValue, relatedNode, relatedTarget, screenX, screenY, shiftKey, srcElement, target, toElement, view, wheelDelta, which

Для того чтобы увидеть эти поля выполните практическое задание 4.2.

Основные свойства объекта event описаны ниже.

Так же можно создавать пользовательские события.

Определение типа события

Для определения типа событий используется свойство type объекта event.

```
$("#a").click(function(event) {  
    alert(event.type); // "click"  
});
```

Время запуска события

Для определения времени, когда произошло событие, используется свойство

Тема 4. События

timeStamp объекта event. Время события определяется в миллисекундах от 1 января 1970 года.

```
event.timeStamp
```

Пространство имен

Тип событий может быть задан с указанием пространства имен, например: `on('click.namespasing', function)`. Здесь `namespasing` это пространство имен, а `click` — тип события. Пространство имен позволяет разделить обработчики одних и тех же событий на подгруппы, которые, впоследствии, будет легко отдельно вызывать методом `trigger()` и удалять методом `off()`.

Можно указывать сразу несколько пространств имен: `on('click.namespasing1.namespasing2',function)`. Чтобы обратиться к такому событию, достаточно указать любое из них (или не указывать вовсе).

В следующем примере устанавливается событие `click` на все `li`-элементы

```
$( 'li' ).on( 'click', function1 );
```

Далее создается другой обработчик событий `click` с пространством имен `list1`

```
$( 'li' ).on( 'click.list1', function2 );
```

Далее создается другой обработчик событий `click` с пространством имен `list1`

```
$( 'li' ).on( 'click.list2', function3 );
```

В следующем примере вызываются все обработчики события `click` и вызываются `function1`, `function2` и `function3`

```
$( 'li' ).trigger( 'click' );
```

В следующем примере вызываются обработчики события `click` только с пространством имен `list1`

```
$( 'li' ).trigger( 'click.list1' );
```

В следующем примере удаляются обработчики `click` только с пространством имен `list1`

```
$( 'div' ).off( 'click.list1' );
```

Определить используемое пространство имён можно, используя свойство `namespace` объекта `event`.

Прямая и делегированная обработка (параметр selector)

Если этот параметр не задан или равен `null`, то обработчик события будет установлен на выбранные элементы, и будет срабатывать в тех случаях, когда событие возникло непосредственно на этих элементах или «поднялось» от их потомков. Такую обработку событий называют прямой.

Предположим на странице есть `div`-элемент, внутри которого располагаются `span`-элементы, которые должны реагировать на какое-то событие.

Если установить обработчики непосредственно на существующие `span`-элементы,

то на новых span-элементах, добавленных с помощью jQuery, эти обработчики действовать не будут. Чтобы избежать ошибок, которые могут появиться в результате «прямой» обработки событий, следует использовать так называемую «делегированную обработку». Делегированная обработка подразумевает установку обработчика событий на объект, внутри содержащий элементы. Тогда при возникновении событий на вновь созданных элементах события так же будут обрабатываться (рисунок 4.1).

**«Прямая»
обработка событий**



**«Делегированная»
обработка событий**



Рисунок 4.1 Прямая и делегированная обработка событий

Вот так выглядит запись прямой обработки событий:

```
$( 'div span' ).on( 'click', function() { обработчик событий } )
```

А так делегированной обработки событий:

```
$( 'div' ).on( 'click', 'span', function() { обработчик событий } )
```

Для делегирования события не следует использовать события focus, blur, load, paste и reset. Так как они не поддерживают всплывающие события.

Так же делегирование событий следует использовать, как можно ближе к необходимым элементам.

Обработчики событий.

Обработчиком событий, как правило, является функция, реже false.

Вызов обработчика событий в виде анонимной функции:

```
$( 'div' ).on( 'click', function() {
    alert( 'Вы нажали на элемент div' );
} );
```

Вызов функции с заданным именем:

```
function myF () {
    alert( 'Вы нажали на элемент div' );
}
```

```
});  
$('div').on('click', myF)
```

При возникновении события, jQuery передает в вызываемый обработчик объект event, в котором содержится вся информация о произошедшем событии.

После возникновения, событие всегда всплывает по структуре DOM до объекта document. Для остановки всплытия события используется метод event.stopPropagation(), а для проверки вызывался ли сам метод event.stopPropagation() используется метод event.stopImmediatePropagation().

Некоторые события порождают выполнение связанных с ними действий. Например, нажатие на кнопке с типом submit приводит к отправки формы. Для отмены связанных действий используется метод event.preventDefault().

Использование false, как обработчика событий, приводит к отмене связанных событий и событие не будет всплывать по иерархии DOM, что равнозначно событиям event.preventDefault() и event.stopPropagation() соответственно.

Например:

```
$('#input[type='submit'] ").on("click", false);
```

Для проверки запуска метода preventDefault() используется метод isDefaultPrevented(), который возвращает true, если метод preventDefault() был запущен для данного элемента и false, если запущен не был.

```
$("#a").click(function(event){  
    alert( event.isDefaultPrevented() ); // false  
    event.preventDefault();  
    alert( event.isDefaultPrevented() ); // true  
});
```

Для проверки был ли запущен метод stopPropagation() используется метод isPropagationStopped(), который возвращает true, если метод stopPropagation() был запущен для данного элемента и false, если запущен не был.

Дополнительные данные в обработчике событий

Через параметр data можно передавать данные любого типа. Извлекать данные можно с помощью event.data.

В следующем примере задаются дополнительные данные с именем value со значением «Ссылка».

```
$("#a").on("click", { value: "Ссылка" }, myF);
```

В следующем примере извлекаются данные с именем «value» со значением «Ссылка»

```
function myF (event){  
    alert("Это "+event.data.value);
```

```
});
```

Определение источника события

С помощью свойства `event.currentTarget` и `event.target` можно определить, какой элемент является источником события. `CurrentTarget` совпадает с переменной `this`. Но не всегда элемент может являться источником события, поскольку оно могло быть передано от дочернего элемента, в результате всплытия события, вверх по иерархии. Первоначальный источник события содержится в `event.delegateTarget`.

Определение положения курсора мыши.

Для определения положения курсора мыши используется два свойства объекта `event` `pageX` и `pageY`. Свойство `pageX` определяет положение курсора мыши от левого края документа, а свойство `pageY` определяет положение курсора от верхней части документа.

Базовые события.

События на выбранных элементах

Событие `on` устанавливает обработчики событий на выбранных элементах веб-страницы. Существует два способа использования события `on`.

Первый способ имеет следующую запись:

```
.on(events, [selector], [data], function)
```

`events` – тип обрабатываемого события, например `click`, `dblclick` и другие события, которые будут рассмотрены ниже. Для того чтобы применить несколько событий, их следует писать через пробел – `'click dblclick'`.

`selector` – определяет селектор-фильтр, по которому будут фильтроваться элементы, лежащие внутри найденных, что приведет к тому что обработчик будет срабатывать, если событие «поднялось» от одного из отфильтрованных элементов.

`data` – данные, которые будут передаваться обработчику событий и которые будут доступны с помощью объекта `events` (`events.data`)

`function` – функция или имя определенной ранее функции, которая будет установлена в качестве обработчика

С помощью второго способа можно установить несколько событий и их обработчиков событий на один объект. Второй способ имеет следующую запись:

```
.on(events-map, [selector], [data])
```

`events-map` - объект, в котором перечисляются типы обрабатываемых событий (`event`) и соответствующие им обработчики. Задается в формате `-1:function-1, events-2:function-2,`

`selector` – определяет селектор-фильтр, по которому будут фильтроваться элементы, лежащие внутри найденных, что приведет к тому что обработчик

будет срабатывать, если событие «поднялось» от одного из отфильтрованных элементов.

data – данные, которые будут передаваться обработчику событий и которые будут доступны с помощью объекта events (events.data)

Убрать установленный обработчик можно с помощью метода off(), который описан ниже.

Метод one заменил устаревший метод .bind(), .delegate(), .live(,) начиная с 1.7 версии jQuery.

Удаление обработчиков событий

С помощью метода off можно удалить обработчик событий, установленный методом on.

.off (events, [selector], function)

events – тип обрабатываемого события, например click, dblclick.

selector – селектор, указанный (если он был указан) при установке события в методе .on().

function – функция, которую следует удалить

Событие на выбранных элементах

Событие one устанавливает обработчики событий на выбранных элементах веб-страницы. В отличие от события on событие one вызывается не более одного раза.

.one(event, [data], function(event))

events – тип обрабатываемого события, например, click, dblclick и другие события, которые будут рассмотрены ниже.

data – данные, которые будут передаваться обработчику событий и которые будут доступны с помощью объекта events (events.data)

function(event) — функция, которая будет установлена в качестве обработчика. При вызове она будет получать объект события event.

Убрать установленный обработчик события можно с помощью метода unbind().

Удаление обработчика событий

С помощью метода unbind можно удалить обработчик событий, установленный методом on и устаревшим методом bind.

unbind (event, function)

event – тип обрабатываемого события, например click, dblclick.

function – функция, которую следует удалить

События мыши

Обработчик события click

Событие щелчка мыши по элементу устанавливается с помощью обработчика событий click.

```
$("#div").click(data, function(event))
```

data – данные, которые будут передаваться обработчику событий и которые будут доступны с помощью объекта events (events.data)

function(event) — функция, которая будет установлена в качестве обработчика. При вызове она будет получать объект события event.

Удалить обработчик click можно с помощью метода unbind().

Обработчик события dblclick

Событие двойного щелчка мыши по элементу устанавливается с помощью обработчика событий click.

```
$("#div").dblclick(data, function(event))
```

data – данные, которые будут передаваться обработчику событий и которые будут доступны с помощью объекта events (events.data)

function(event) — функция, которая будет установлена в качестве обработчика. При вызове она будет получать объект события event.

Обработчик события hover

Устанавливает обработчик при наведении курсора мыши и вывода курсора с объекта.

```
.hover(functionIn(event), functionOut(event))
```

functionIn(event) — функция, которая будет установлена в качестве обработчика при наведении курсора мыши на объект. При вызове она будет получать объект события event.

functionOut(event) — функция, которая будет установлена в качестве обработчика при выводе курсора мыши с объекта. При вызове она будет получать объект события event.

Нажатие клавиши мыши

Событие mousedown происходит при нажатии любой клавиши мыши. Для определения нажатой клавиши мыши используется переменная which объекта event, где значения 1-левая, 2-средняя, 3-правая кнопка мыши.

```
.mousedown(function(event){})
```

Отпуск клавиши мыши

Событие mouseup происходит при возвращении клавиш мыши в ненажатое состояние. Для определения ненажатой клавиши мыши используется переменная which объекта event, где значения 1-левая, 2-средняя, 3-правая кнопка мыши.

Тема 4. События

```
.mouseup(function(event){})
```

Наведение курсора мыши на элемент

При наведении курсора мыши на элемент запускаются события `mouseenter` и `mouseover`. Эти два события во многом схожи, но так же имеют одно важное отличие. Событие `mouseover` всплывает по иерархии DOM, что может приводить к выполнению заданной в событии функции несколько раз. Событие `mouseenter` выполняется один раз, что (в большинстве случаев предпочтительней).

```
.mouseenter (function(event){})
```

```
.mouseover (function(event){})
```

Рассмотрим подробнее действие двух этих событий на примере рисунка 4.2

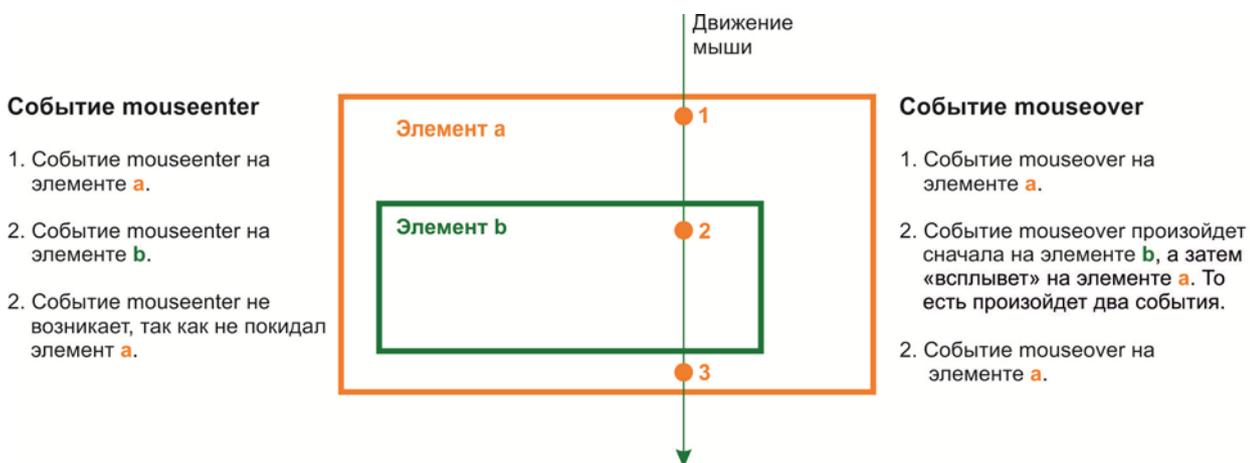


Рисунок 4.2 События `mouseenter` и `mouseover`

Выход курсора мыши с элемента

При выходе курсора мыши с элемента запускаются события `mouseleave` и `mouseout`. Эти два события во многом схожи, но так же имеют одно важное отличие. Событие `mouseout` всплывает по иерархии DOM, что может приводить к выполнению заданной в событии функции несколько раз. Событие `mouseleave` выполняется один раз, что в большинстве случаев предпочтительней.

```
.mouseleave (function(event){})
```

```
.mouseout (function(event){})
```

Рассмотрим подробнее действие двух этих событий на примере рисунка 4.3

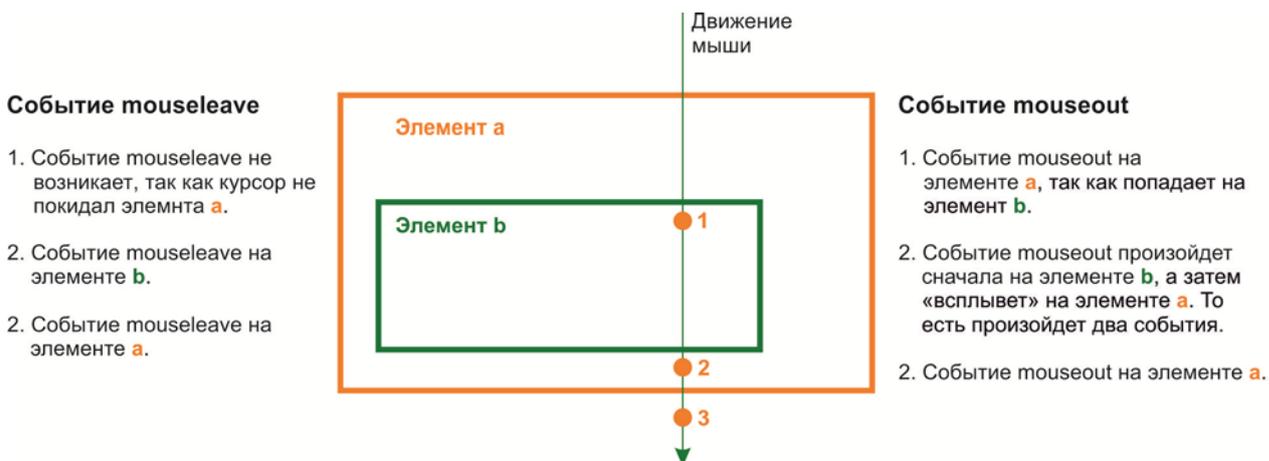


Рисунок 4.3 Событие mouseleave и mouseout

Свойство event.relatedTarget возвращает, куда переместился курсор с объекта

```
event.relatedTarget.nodeName
```

Движение мышью

Событие mousemove происходит при возвращении движения мыши.

```
.mousemove (function(event){})
```

Поочерёдное выполнение нескольких функций.

С помощью метода toggle() можно последовательно запускать несколько функций и управлять видимостью элемента.

Рассмотрим первое применение метода toggle - вызов последовательно нескольких функций. В ниже приведенном коде будут последовательно вызваны три функции при наступлении события click. После вызова последней функции будет вызвана первая и так далее.

```
.toggle (function(){alert("Вызов первой функции")},
        function(){ alert("Вызов второй функции")},
        function(){alert("Вызов третьей функции")}
    )
```

Этот способ применения считается устаревшим и, начиная с версии 19.0 не поддерживается.

Второй способ применения этого метода – изменение видимости элемента. При использовании метода toggle для скрытия/отображения можно указать время продолжительности анимации скрытия в миллисекундах или с помощью ключевых слов fast (200 миллисекунд) или slow (600 миллисекунд). Так же при необходимости можно указать функции, которые запустятся после завершения анимации (более подробно анимация рассмотрена в задании 4.10).

```
.toggle (animation, function(){})
```

Так же элементы можно скрывать/отображать без использования анимации для этого следует использовать аргумент showOrHide со значением true (виден) или

false (скрыт). Пример скрытия объекта демонстрирует следующая запись:

```
.toggle (showOrHide=false)
```

События загрузки страницы

Обработка загрузки DOM

Событие ready происходит после построения дерева документа, когда загружена только структура документа, но еще не загружено содержимое документа (текст, изображения, мультимедийные элементы). Это событие удобно для загрузки вспомогательного кода, которое не требует немедленного исполнения или работает со структурой строящегося документа. Похожее событие javascript onload срабатывает только тогда, когда страница полностью сформирована и загружены все ее составляющие, что не всегда удобно. Одновременное применение события ready и <body onload="" "> приводит к ошибке, как альтернативу onload можно использовать событие jQuery load.

Событие ready привязывается к объекту document и запускает обработчик события в виде функции.

Запуск анонимной функции:

```
$(document).ready(function(){ });
```

Запуск пользовательской функции:

```
$(document).ready(loadPage);  
function loadPage(){
```

Обработка загрузки страницы

Событие load происходит, когда элемент и все его дочерние элементы загружены. Этот элемент можно применять как к объекту window, так и к элементам документа, имеющим поле URL (фреймы, скрипты, стили, изображения).

Событие load можно применить к объекту windows:

```
$(window).load(function () {});
```

или к объекту с полем url:

```
  
$('#dog').load(function() {})
```

Обработка выгрузки страницы.

Событие unload происходит, когда пользователь переходит по ссылке, изменяет адрес страницы вручную в адресной строке, при использовании кнопок «Вперед» и «Назад», а так же при закрытии окна браузера.

```
$(window).unload(function({})
```

События браузера

Обработчик ошибки.

Метод `error()` устанавливает обработчик ошибки при загрузке элементов веб-страницы.

В следующем примере обрабатывается ошибка загрузки изображения.

```
$( 'img' ).error( function() { alert( "Изображение не загружено" ); } );
```

Изменение окна браузера

Метод `resize()` устанавливает обработчик на событие изменения размеров окна браузера. Событие `resize` в разных браузерах вызывается разное количество раз. В частности, в браузере Internet Explorer событие выполняется два раза, а в Opera и FireFox один раз.

```
$( window ).resize( function() {} )
```

Прокрутка документа

Метод `scroll()` устанавливает обработчик событий на прокрутку окна браузера. Метод можно применять к окну браузера и элементам страницы, имеющим прокрутку.

```
$( window ).scroll( function() {} )
```

События клавиатуры

Для определения нажатий клавиш клавиатуры используется три метода `keydown()`, `keyup()` и `keypress()`. Любой из этих методов используется для того, чтобы узнать какая именно клавиша была нажата(,) используется переменная значения переменной `which` объекта `Event`. Вот как выглядит определение нажатой клавиши события `keydown`:

```
$( 'input' ).keydown( function( ev ) { alert( ev.which ); } )
```

При использовании методов `keydown()` и `keyup()` мы можем узнать только **код** нажатой клавиши для того, чтобы узнать вводимый символ можно воспользоваться таблицей 4.1. Так же эти методы не различают коды символов латинского и кириллического алфавита, и прописные и строчные символы.

Таблица 4.1 Коды клавиш клавиатуры

| Код | Символ | Код | Сим | Код | Символ | Код | Символ | Код | Символ |
|-----|-----------|-----|-----|-----|--------|-----|----------|-----|------------|
| 8 | Backspace | 48 | 0 | 75 | К | 97 | NumPad 1 | 118 | F7 |
| 9 | Tab | 49 | 1 | 76 | Л | 98 | NumPad 2 | 119 | F8 |
| 13 | Enter | 50 | 2 | 77 | М | 99 | NumPad 3 | 120 | F9 |
| 16 | Shift | 51 | 3 | 78 | Н | 100 | NumPad 4 | 121 | F10 |
| 17 | Ctrl | 52 | 4 | 79 | О | 101 | NumPad 5 | 122 | F11 |
| 18 | Alt | 53 | 5 | 80 | Р | 102 | NumPad 6 | 123 | F12 |
| 19 | Pause | 54 | 6 | 81 | Q | 103 | NumPad 7 | 144 | NumLock |
| 20 | CapsLock | 55 | 7 | 82 | Р | 104 | NumPad 8 | 145 | ScrollLock |

Тема 4. События

| | | | | | | | | | |
|----|----------|----|---|----|-------------|-----|----------|-----|-------------|
| 27 | Esc | 56 | 8 | 83 | S | 105 | NumPad 9 | 154 | PrintScreen |
| 32 | Пробел | 57 | 9 | 84 | T | 106 | NumPad * | 157 | Meta |
| 33 | PageUp | 65 | A | 85 | U | 107 | NumPad+ | 186 | ; |
| 34 | PageDown | 66 | B | 86 | V | 109 | NumPad - | 187 | = |
| 35 | End | 67 | C | 87 | W | 110 | NumPad . | 188 | , |
| 36 | Home | 68 | D | 88 | X | 111 | NumPad / | 189 | - |
| 37 | влево | 69 | E | 89 | Y | 112 | F1 | 190 | . |
| 38 | вверх | 70 | F | 90 | Z | 113 | F2 | 191 | / |
| 39 | вправо | 71 | G | 91 | лв window | 114 | F3 | 192 | ~ |
| 40 | вниз | 72 | H | 92 | пр window | 115 | F4 | 219 | [|
| 45 | Insert | 73 | I | 93 | Application | 116 | F5 | 220 |] |
| 46 | Delete | 74 | J | 96 | NumPad 0 | 117 | F6 | 221 | \ |

Метод `keypress()` позволяет узнать номер вводимого символа, а не код клавиши. Для определения вводимого символа можно воспользоваться следующей записью:

```
$( 'input' ).keypress( function( ev ) { alert( ev.which ); } )
```

Номер символа можно сопоставить с таблицей 4.2 и определить нажатую клавишу.

Таблица 4.2 Коды символов клавиатуры

| Код | Символ |
|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|
| 32 | пробел | 48 | 0 | 64 | @ | 96 | ` | 112 | p |
| 33 | ! | 49 | 1 | 65 | A | 97 | a | 113 | q |
| 34 | " | 50 | 2 | 66 | B | 98 | b | 114 | r |
| 35 | # | 51 | 3 | 67 | C | 99 | c | 115 | s |
| 36 | \$ | 52 | 4 | 68 | D | 100 | d | 116 | t |
| 37 | % | 53 | 5 | 69 | E | 101 | e | 117 | u |
| 38 | & | 54 | 6 | 70 | F | 102 | f | 118 | v |
| 39 | ' | 55 | 7 | 71 | G | 103 | g | 119 | w |
| 40 | (| 56 | 8 | 72 | H | 104 | h | 120 | x |
| 41 |) | 57 | 9 | 73 | I | 105 | i | 121 | y |
| 42 | * | 58 | : | 74 | J | 106 | j | 122 | z |
| 43 | + | 59 | ; | 75 | K | 107 | k | 123 | { |
| 44 | ` | 60 | < | 76 | L | 108 | l | 124 | |
| 45 | - | 61 | = | 77 | M | 109 | m | 125 | } |
| 46 | . | 62 | > | 78 | N | 110 | n | 126 | ~ |
| 47 | / | 63 | ? | 79 | O | 111 | o | 127 | del |

Таблица 4.3 Коды русских символов

| Код | Сим |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|

| | | | | | | | | | | | |
|------|---|------|---|------|---|------|---|------|---|------|---|
| 1040 | А | 1051 | Л | 1062 | Ц | 1072 | а | 1083 | л | 1094 | ц |
| 041 | Б | 1052 | М | 1063 | Ч | 1073 | б | 1084 | м | 1095 | ч |
| 1042 | В | 1053 | Н | 1064 | Ш | 1074 | в | 1085 | н | 1096 | ш |
| 1043 | Г | 1054 | О | 1065 | Щ | 1075 | г | 1086 | о | 1097 | щ |
| 1044 | Д | 1055 | П | 1066 | Ъ | 1076 | д | 1087 | п | 1098 | ъ |
| 1045 | Е | 1056 | Р | 1067 | Ы | 1077 | е | 1088 | р | 1099 | ы |
| 1046 | Ж | 1057 | С | 1068 | Ь | 1078 | ж | 1089 | с | 1100 | ь |
| 1047 | З | 1058 | Т | 1069 | Э | 1079 | з | 1090 | т | 1101 | э |
| 1048 | И | 1059 | У | 1070 | Ю | 1080 | и | 1091 | у | 1102 | ю |
| 1049 | Й | 1060 | Ф | 1071 | Я | 1081 | й | 1092 | ф | 1103 | я |
| 1050 | К | 1061 | Х | 1025 | Ё | 1082 | к | 1093 | х | 1105 | ё |

Нажатие клавиши клавиатуры

Метод `keydown()` устанавливает обработчик событий нажатие клавиши клавиатуры. Обработчик можно установить на объект `document` и на любой другой элемент страницы. Событие происходит только на элементе, находящемся в фокусе.

```
$(document).keydown(function(ev){alert(ev.which)})
```

Отпускание клавиши клавиатуры

Метод `keyup()` устанавливает обработчик событий отпуск нажатой клавиши клавиатуры. Обработчик можно установить на объект `document` и на любой другой элемент страницы. Событие происходит только на элементе, находящемся в фокусе.

```
$(document).keyup(function(ev){alert(ev.which)})
```

Определение введенного символа

Метод `keypress()` устанавливает обработчик событий нажатия и отпуск нажатой клавиши клавиатуры. Обработчик можно установить на объект `document` и на любой другой элемент страницы. Событие происходит только на элементе, находящемся в фокусе и возвращает номер введенного символа.

```
$(document).keypress(function(ev){alert(ev.which)})
```

События формы

Получение фокуса элементом.

Событие выделения элемента или получение им фокуса может вызвать два метода `focus()` и `focusin()`. Отличие этих двух методов заключается в том, что метод `focusin()` запускается, если выбран сам элемент или один из его дочерних

Тема 4. События

элементов. Метод `focus()` запускается только при выборе самого элемента.

В следующем примере функция будет запущена при получении фокуса любым элементом формы.

```
$('#form').focusin(function(){alert()})
```

А в этом примере функция запустится при выборе только текстового поля.

```
$("#[type=text]").focusin(function(){alert()})
```

Потеря фокуса элементом.

Для установки обработчика потери фокуса используется два метода `focusout()` и `blur()`. Метод `focusout()` запускается при потере фокуса, как самим элементом, так и его дочерними. Метод `blur()` запускается при потере фокуса самим элементом.

В следующем примере функция будет запущена при потере фокуса любым элементом формы.

```
$('#form').focusout(function(){alert()})
```

А в этом примере функция запустится при потере фокуса только текстовым полем.

```
$("#[type=text]").blur(function(){alert()})
```

Выделение текста. Событие `Select`.

Событие `select` происходит при выборе текста в элементах `<textarea>` и `<input type="text">` и запускает метод `select()`.

```
$("#[type=text]").select(function(){alert()})
```

При использовании мыши событие `select` возникает после завершения выделения. При использовании клавиатуры (клавиши стрелки с зажатой клавишей `Shift`) событие `select` возникает после каждого нажатия клавиши стрелки.

Для выделения текста, как в элементах формы, так и простого текста используются методы `document.getSelection()` в Firefox и Chrome и `document.selection.createRange()` в IE и Opera.

Изменение элемента формы. Событие `Change`.

При изменении элемента формы возникает событие `change`, которое запускает метод `change()`. Метод `change()` запускается не в момент изменения элемента, а когда он теряет фокус.

```
$("#[type=text]").change(function(){alert()})
```

Отправка формы. Событие `Submit`

Событие `submit` возникает перед отправкой формы на сервер.

```
$("#a").submit(function(){})
```

Практические задания к теме 4

Практическое задание 4.1 Всплытие событий

В этом задании показано, как работают события в JavaScript.

Создайте новый html файл и сохраните его под именем **practicejQuery4_01.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.1 Всплытие событий </title>
<style type="text/css">
#levelOne
{
    padding:25px; height:200px; width:200px; background-color:#C30;
}
#levelTwo
{
    padding:25px; height:150px; width:150px; background-color:#39C;
}
#levelThree
{
    padding:25px; height:100px; width:100px; background-color:#396; font-
size:20px;
}
</style>
</head>
<body>
<div id="levelOne" onclick="alert('Верхний div')">
    <div id="levelTwo" onclick="alert('Средний div')">
        <div id="levelThree" onclick="alert('Нижний div')">
            <a href="#" title="Ссылка " onclick="alert('Ссылка')">Ссылка</a>
        </div>
```

```
</div>
</div>
</body>
</html>
```

Практическое задание 4.2 Поля метода Event

В этом задании показаны поля метода event с помощью консоли firebug.

Создайте новый html файл и сохраните его под именем **practicejQuery4_02.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.2 Поля метода Event </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
document.getElementById('levelOne').onclick=function(e)
    {
        console.log(e)
    }
}
</script>
<style type="text/css">
#levelOne
{
padding:25px; height:200px; width:200px; background-color:#C30;
}
#levelTwo
{
padding:25px; height:150px; width:150px; background-color:#39C;
}
#levelThree
{
```

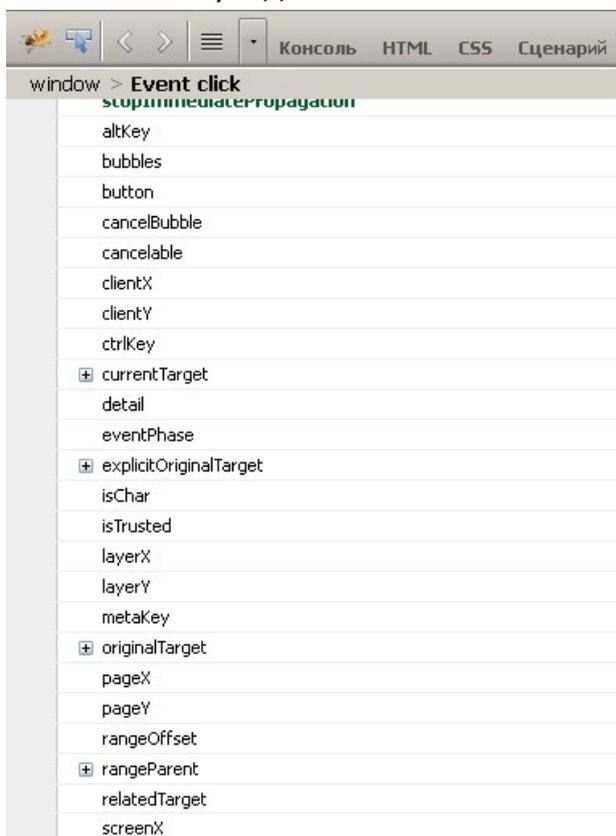
```

padding:25px; height:100px; width:100px; background-color:#396; font-
size:20px;
}
</style>
</head>
<body>
<div id="levelOne">
    <div id="levelTwo">
        <div id="levelThree">
            <a href="#" title="Ссылка">Ссылка</a>
        </div>
    </div>
</div><br>
</body>
</html>

```

После того как вы ввели код, откройте страницу в браузере firefox, щелкните по div-элементу и откройте панель firebug с помощью клавиши F12 или нажав на пиктограмму  в верхнем правом углу браузера.

В консоли вы увидите поля объекта event.



Практическое задание 4.3 Делегирование событий

В этом задании показано, как действует делегирование событий.

Создайте новый html файл и сохраните его под именем **practicejQuery4_03.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.3 Делегирование событий</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#One span').on('click', function(){
    var text = $(this).text();
    alert('Вы нажали на элемент с текстом - «' + text + '»');
    $("<span>Тест</span>").insertAfter($("#One span:eq(2)"));
});
$('#Two').on('click','span', function(){
    var text = $(this).text();
    alert('Вы нажали на элемент с текстом - «' + text + '»');
    $("<span>Тест</span>").insertAfter($("#Two span:eq(2)"));
});
}
</script>
<style type="text/css">
#One
{
    height:auto; width:200px; background-color:#CCE6E3;
    border:2px solid #09F; font-size:16px; float:left;
}
#Two
{
```

```

        height: auto; width:200px; background-color:#F8A998; border:2px solid #C00;
font-size:16px;
        float: left; margin-left:10px;

    }
span, h2
{
    display: block; margin:10px;
}
</style>
</head>
<body>
<div id="One">
<h2>Элемент 1</h2>
<span>Элемент 1 в элементе 1</span>
<span>Элемент 2 в элементе 1</span>
<span>Элемент 3 в элементе 1</span>
</div>
<div id="Two">
<h2>Элемент 2</h2>
<span>Элемент 1 в элементе 2</span>
<span>Элемент 2 в элементе 2</span>
<span>Элемент 3 в элементе 2</span>
</div>
</body>
</html>

```

В этом задании рассматривается делегирование событий. Для обоих div-элементов была создана анонимная функция, которая запускается с помощью события click. В первой функции событие установлено непосредственно к span-элементам и на вновь созданных span-элементах не действует. Во второй функции используется делегированный обработчик события, что приводит к тому, что событие происходит и на вновь созданных элементах.

Практическое задание 4.4 Извлечение данных, переданных событием

В этом задании показано, как задавать и извлекать данные, переданные с

Тема 4. События

помощью обработчика событий.

Создайте новый html файл и сохраните его под именем **practicejQuery4_04.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.4 Извлечение данных, переданных
событием</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("a").on("click", { value: "Ссылка" }, myF);
function myF (event){
alert("Это "+event.data.value);
alert("Положение курсора по X: "+event.pageX+" \nПоложение курсора по Y:
"+event.pageY)
};
}
</script>
</head>
<body>
<a href="#">Ссылка</a>
</body>
</html>
```

Практическое задание 4.5 Событие on и удаление события

В этом задании показано, как работает событие on и как оно удаляется.

Создайте новый html файл и сохраните его под именем **practicejQuery4_05.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
```

```

<title>Практическое задание 4.2 Событие on</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $('#levelOne').on('click dblclick', '#levelThree', function(){
        alert('Вы нажали на элемент с id='+ $(this).attr("id"));
    });
    $('#oFF').on('click', function(){
        $('#levelOne').off('click', '#levelThree')
        $('#oFF').val('Событие click удалено')
        $('#levelThree').text('Щелкните два раза')
    });
}
</script>
<style type="text/css">
#levelOne
{
    padding:25px; height:200px; width:200px; background-color:#C30;
}
#levelTwo
{
    padding:25px; height:150px; width:150px; background-color:#39C;
}
#levelThree
{
    padding:25px; height:100px; width:100px; background-color:#396; font-
size:20px;
}
</style>
</head>
<body>
<div id="levelOne">

```

Тема 4. События

```
        <div id="levelTwo">
            <div id="levelThree">
                Щелкните один раз
            </div>
        </div>
    </div>
</div><br>
<input type="button" value="Удалить событие click" id='oFF'>
</body>
</html>
```

Практическое задание 4.6 Событие `one` и его удаление

В этом задании показано, как работает событие `one` и как оно удаляется.

Создайте новый `html` файл и сохраните его под именем **practicejQuery4_06.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.2 Событие on</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
var a='Щелчок'
window.onload=function()
{
    $('#levelOne').one('click',{value:Щелчок}), function(event){
        $('#levelThree').text(event.data.value);
    });
    $('#oFF').on('click', function(){
        $('#levelOne').unbind('click')
        $('#oFF').val('Событие click удалено')
    });
}
</script>
<style type="text/css">
```

```

#levelOne
{
    padding:25px; height:200px; width:200px; background-color:#C30;
}
#levelTwo
{
    padding:25px; height:150px; width:150px; background-color:#39C;
}
#levelThree
{
    padding:25px; height:100px; width:100px; background-color:#396; font-
size:20px;
}
</style>
</head>
<body>
<div id="levelOne">
    <div id="levelTwo">
        <div id="levelThree">
            <a href="#" title="Ссылка">Ссылка</a>
        </div>
    </div>
</div>
</div>
<br>
<input type="button" value="Удалить событие click" id='oFF'>
</body>
</html>

```

Практическое задание 4.7 События мыши

В этом задании показано, как работает событие click, dblclick и hover.

Создайте новый html файл и сохраните его под именем **practicejQuery4_07.html**.

Введите следующий код:

```

<!doctype html>
<html>

```

Тема 4. События

```
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.7 События мыши</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("#One").dblclick(function(event){
$("#eventText").text('Вы два раза щелкнули на элемент')
})
$("#One").click(function(event){
$("#eventText").text('Вы один раз щелкнули на элемент')
})
$("#One").hover(
function(event){
    $("#eventText").text('Вы навели мышь на элемент')
    },
function(event){
    $("#eventText").text('Вы убрали мышь с элемента')

    }
)
}
</script>
<style type="text/css">
#One
{
    height:auto; width:200px; background-color:#CCE6E3;
    border:2px solid #09F; font-size:16px;
    float:left;
}
#Two
{
```

```
        height:auto; width:200px; background-color:#F8A998;
        border:2px solid #C00; font-size:16px; float:left; margin-left:10px;
    }
    span, h2
    {
        display:block; margin:10px;
    }
</style>
</head>
<body>
<div id="One">
<h2>Элемент 1</h2>
<span>Элемент 1 в элементе 1</span>
<span>Элемент 2 в элементе 1</span>
<span>Элемент 3 в элементе 1</span>
</div>
<div id="Two">
<h2>Элемент 2</h2>
<span>Элемент 1 в элементе 2</span>
<span>Элемент 2 в элементе 2</span>
<span>Элемент 3 в элементе 2</span>
</div>
<div id="eventText"></div>
</body>
</html>
```

Практическое задание 4.8 События мыши

В этом задании показано, как работает событие `mousedown` и `mouseup`.

Создайте новый html файл и сохраните его под именем **practicejQuery4_08.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
```

```
<meta charset="windows-1251">
<title>Практическое задание 4.8 События мыши</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#One').mousedown(function(event){
$('#eventText').css('color','red').text('Вы нажали '+event.which+' кнопку. В
координатах X:'+event.pageX+' и Y: '+event.pageY)
})
$('#One').mouseup(function(event){
$('#eventText').css('color','green').text('Вы отпустили '+event.which+' кнопку. В
координатах X:'+event.pageX+' и Y: '+event.pageY)
})
}
</script>
<style type="text/css">
#One
{height:auto; width:200px; background-color:#CCE6E3; border:2px solid #09F; font-
size:16px; float:left;
}
span, h2
{display:block; margin:10px;
}
div
{clear:both;}
</style>
</head>
<body>
<div id="One">
<h2>Элемент 1</h2>
<span>Элемент 1 в элементе 1</span>
<span>Элемент 2 в элементе 1</span>
```

```

<span>Элемент 3 в элементе 1</span>
</div>
<div id="eventText"></div>
</body>
</html>

```

Практическое задание 4.9 События мыши

В этом задании показано, как работает событие `mouseenter` и `mouseover`.
Создайте новый html файл и сохраните его под именем **practicejQuery4_09.html**.
Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 4.9 События мыши</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    var mouseover_o=0;
var mouseenter_o=0;
    $("#outer").mouseover(function(){
        a=++mouseover_o
        $(".outmouseover",this).text("Событие mouseover "+a);
    }).mouseenter(function(){
        b=++mouseenter_o
        $(".outmouseenter",this).text("Событие mouseenter "+b);
    });

    var mouseover_i=0;
    var mouseenter_i=0;
    $("#inner").mouseleave(function(){
        a=++mouseover_i

```

Тема 4. События

```
    $(".inmouseover",this).text("Событие mouseleave "+a);
  }).mouseout(function(){
    b=++mouseenter_i
    $(".inmouseenter",this).text("Событие mouseout "+b);
  });
}
</script>
<style type="text/css">
#outer
{margin:40px; height:250px; width:400px;border:3px solid #EF7F1A; font-
size:16px;}
#inner
{height:150px; width:300px; margin-left:30px;border:3px solid #61A375; font-
size:16px;}
span
{display:block; padding:10px; float:left;width:160px;}
div
{clear:both;}
</style>
</head>
<body>
<div id="outer">
  <span class='outmouseover'></span>
  <span class='outmouseenter'></span>
  <div id="inner">
    <span class='inmouseover'></span>
    <span class='inmouseenter'></span>
  </div>
</div>
<div id="eventText"></div>
</body>
</html>
```

Практическое задание 4.10 Использование метода toggle

В этом задании показано, как работает метод toggle.

Создайте новый html файл и сохраните его под именем **practicejQuery4_10.html**.

Обратите внимание на версию библиотеки jQuery, начиная с версии 1.9 toggle для вызова нескольких последовательных функций не работает.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 4_10.Использование метода toggle</title>
<script type="text/javascript" src="http://code.jquery.com/jquery-1.8.0.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#header').toggle(
    function(){alert("Вызов первой функции")},
    function(){alert("Вызов второй функции")},
    function(){alert("Вызов третьей функции")}
);
$('div:gt(0)').click(function(){
    $(this).toggle(1000,function(){
        alert("Элемент "+$(this).text()+" исчез");
        $(this).toggle(true);
        alert("Элемент "+$(this).text()+" и снова появился")
    }
    )
    }
    )
$('.header').toggle(showOrHide=false)
}
</script>
<style type="text/css">
```

```
div
{padding:25px; margin:10px; border:2px solid #0C6; float:left; background-
color:#FF9831;
}
</style>
</head>
<body>
<div id='header'>1</div>
<div class='logo'>2</div>
<div>3</div>
<div class='header'>4</div>
<div>5</div>
</body>
</html>
```

Практическое задание 4.11 События браузера

В этом задании показано, как работают события браузера.

Создайте новый html файл и сохраните его под именем **practicejQuery4_11.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 4.11. События браузера </title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
//Обработка ошибки при загрузки изображения
$('#dog')
.error(function() { alert('Изображение не загрузилось')
})
.attr("src", "dog.jpg");
//Определение размеров окна браузера с помощью метода resize
```



```
</body>  
</html>
```

Практическое задание 4.12 События клавиатуры

В этом задании показано, как работают события браузера.

Создайте новый html файл и сохраните его под именем **practicejQuery4_12.html**.

Введите следующий код:

```
<!doctype html>  
<html>  
<head>  
<meta charset="windows-1251">  
<title>Задание 4.12 События клавиатуры</title>  
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>  
<script type="text/javascript">  
window.onload=function()  
{  
  $('input').keypress(function(ev){alert("Вы нажали клавишу"+ev.which)});  
  $(document).keydown(function(ev){alert("Вы нажали клавишу"+ev.which)});  
  $(document).keyup(function(ev){alert("Вы нажали клавишу"+ev.which)});  
}  
</script>  
<style type="text/css">  
div  
{ padding:25px; margin:25px; border:2px solid #0C6; float:left; background-  
color:#FF9831;  
  
}  
</style>  
</head>  
<body>  
<input type='text'>  
</body>  
</html>
```

Самостоятельная работа к теме 4

Самостоятельная работа 4.1 Определение положения курсора мыши

Используя метод `one()` и событие `mousemove`, выведите значения свойств `pageX` и `pageY` объекта `event`, как текст страницы.

Самостоятельная работа 4.2 Определение времени события

Выведите в код страницы время выполнения события (например, `click`), используя свойство `timeStamp` объекта `event`.

Тема 5. Методы работы с наборами элементов

Началом работы с элементами страницы является обёртка их в набор элемента jQuery (например `$("#div")`). В теме «Выбор элементов с помощью селекторов и фильтров» были подробно описаны способы обертки элементов, а в теме «Манипуляции с элементами, их свойствами и содержимым» рассмотрено изменение свойств обернутого набора. В этой главе рассматриваются методы фильтрации уже готового набора элементов.

Перемещение по иерархии DOM.

В этом разделе рассматриваются манипуляции с набором элементов jQuery на основе их родительских и дочерних связей.

Поиск заданных дочерних элементов.

Для выбора заданных дочерних элементов используется метод `children()`. Метод `children()` выбирает элементы, непосредственно вложенные в указанный элемент.

В следующем примере будут выбраны все «прямые потомки», находящиеся в `div`-элементе.

```
$("#div").children().css("background-color", "red");
```

В следующем примере будут выбраны все `span`-элементы, находящиеся в `div`-элементе.

```
$("#div").children(span).css("background-color", "red");
```

Поиск родительских элементов.

Для выбора родительских элементов используются следующие методы: метод `closest()`, метод `parents()`, метод `parentsUntil()` и метод `parent()`.

Метод `closest()` ищет ближайшего (**первого**) родителя или прародителя указанного элемента, включая сам элемент. В отличие от метода `closest()` метод `parents()` вбирает **всех** ближайших (то есть всех, удовлетворяющих условиям элементов) родителей и прародителей указанного элемента, включая сам элемент. Метод `parent()` ищет прямых родителей заданных элементов. Метод `parentsUntil()` выбирает всех родителей указанного элемента, в том числе и прародителя.

В следующем примере будет выбран первый родительский `div`-элемент элемента с `id` равным `top`.

```
$("#top").closest("div").css("background-color", "red");
```

В следующем примере будут выбраны все родительские `div`-элементы элемента с `id` равным `top`.

```
$("#top").parents("div").css("background-color", "red");
```

Действие этих методов можно посмотреть в практическом задании 5.1.

Поиск дочерних элементов по заданному селектору.

С помощью метода `find()` осуществляется поиск внутри заданных элементов. Метод `find()` в отличие от метода `children()` ищет элементы на всех уровнях вложенности.

В следующем примере будут выбраны все элементы `i`, находящиеся внутри `p`.

```
$("#p").find("i").css("background-color", "red");
```

Так же удобно использовать метод `find` при выборе в уже выбранном наборе элементов. В следующем примере создается переменная `divElement`, в которую помещается набор `div`-элементов. Далее с помощью метода `find` из набора `div`-элементов выбираются все `span`-элементы.

```
var $divElement=$("#div");
```

```
$divElement.find("span");
```

Выбор элементов, лежащих после указанного.

Для выбора элементов после указанных, используются три метода: метод `next()`, метод `nextAll()` и метод `nextUntil()`. Метод `next()` позволяет выбрать один следующий элемент, идущий за указанным. В следующем примере будет выбран первый `div`-элемент, идущий за первым `div`-элементом на странице.

```
$("#div").eq(0).next("div").css("background-color","yellow");
```

Метод `nextAll()` выбирает все указанные элементы, идущие за указанным. В следующем примере будут выбраны все `div`-элементы, следующие за первым `div`-элементом на странице.

```
$("#div").eq(0).nextAll("div").css("background-color","blue");
```

Метод `nextUntil()` выбирает все соседние элементы между заданными, сами заданные элементы не выбираются. В следующем примере будут выбраны все соседние элементы между элементами с классом `top` и классом `bottom`.

```
$(".top").nextUntil(".bottom").css("background-color","green");
```

Действие этих методов можно посмотреть в практическом задании 5.2.

Выбор элементов, лежащих перед указанным.

Для выбора элементов, лежащих перед указанным набором, используются три метода: метод `prev()`, метод `prevAll()` и метод `prevUntil()`.

Метод `prev()` выбирает первый соседний элемент вверх по иерархии документа. В следующем примере будет выбран элемент, идущий за вторым `div`-элементом на странице.

```
$("#div").eq(1).prev("div").css("background-color","yellow");
```

Метод `prevAll()` выбирает все элементы, идущие перед указанным. В следующем примере будут выбраны все элементы, идущие перед элементом с классом `bottom`

```
$(".bottom").prevAll().css("background-color","blue");
```

Метод `prevUntil()` выбирает все верхние соседние элементы между заданными, сами заданные элементы не выбираются. В следующем примере будут выбраны

все верхние соседние от элемента с классом bottom до элемента с классом top.

```
$("#bottom").prevUntil(".top").css("background-color", "green");
```

Выбор позиционированного родителя.

С помощью метода `offsetParent()` выбирается родитель указанного элемента с позиционированием отличным от `static`.

Выбор лежащих рядом элементов.

Метод `siblings()` позволяет выбрать лежащие на одном уровне иерархии документа элементы, как вверх, так и вниз по иерархии.

Фильтрация элементов набора.

Для фильтрации набора элементов используют методы `filter()`, `is()` и `slice()`.

Метод filter.

С помощью метода `filter` можно извлечь из созданного набора элементы, удовлетворяющие заданному условию. В следующем примере из всех выбранных `div`-элементов на странице метод `filter` извлечёт элементы с классом `filt`.

```
$("#div").filter(".filt")
```

Действие этого метода можно посмотреть в практическом задании 5.3.

Метод is.

Метод `is()` проверяет присутствует ли указанный элемент в наборе элементов. Этот метод возвращает булево значение.

```
$("#div").is(".filt")
```

Действие этого метода можно посмотреть в практическом задании 5.4.

Поиск элементов по индексам.

С помощью метода `slice()` можно выбрать элементы по индексам из заданного набора. Данный метод можно использовать следующими способами:

```
$("#div").slice(2) - вернет все div-элементы, начиная с третьего (с индексами 2, 3, ...).
```

```
$("#div").slice(2, 5) - вернет div-элементы с индексами 2 и 4.
```

```
$("#div").slice(-3, -1) - вернет div-элементы, идущие третьими и вторыми с конца.
```

```
$("#div").slice(-2) - вернет предпоследний и последний div-элементы на странице.
```

Действие этого метода можно посмотреть в практическом задании 5.4.

Дополнительные методы.

Добавление элементов в набор.

С помощью метода `add()` можно добавить элементы к уже существующему набору элементов. В следующем примере выбраны все элементы с классом `one`, а далее с помощью метода `add()` к набору элементов добавлены элементы с классом `two`.

```
$(".one")  
  .css("background-color","red")  
  .add(".two")  
  .css("background-color","red")
```

Возвращение к предыдущему набору элементов.

С помощью метода `end()` можно вернуться по цепочке методов к предыдущему набору элементов. В следующем примере выбраны все `div`-элементы, затем с помощью метода `add()` к набору элементов добавлены элементы с классом `two`, а далее метод `end()` возвращает предыдущий набор элементов без элементов с классом `two`.

```
$(".div")  
  .css("background-color","red")  
  .add(".two")  
  .css("background-color","red")  
  .end()
```

Объединение выбранных наборов.

С помощью метода `addBack()` можно обратиться одновременно к текущему набору и предыдущему. В следующем примере синяя граница одновременно будет назначена и тегу `p` и `div`-элементу с классом `three`.

```
$(".div.three").find("p").addBack().css("border","2px solid blue");
```

Действие этого метода можно посмотреть в практическом задании 5.5.

Содержимое элемента.

С помощью метода `contents()` можно обратиться к дочерним элементам и их текстовому содержимому.

Действие этого метода можно посмотреть в практическом задании 5.6.

Функция для элемента в наборе элементов.

Метод `each()` выполняет заданную функцию для каждого элемента в наборе элементов в отдельности. Метод `each()` выполняет функцию `function` для каждого из выбранных элементов в качестве параметров передается индекс элемента в наборе и сам элемент `DOM`.

Синтаксис метода следующий:

```
.each(function(index, elementDOM){})
```

Действие этого метода можно посмотреть в практическом задании 5.7.

Элементы набора элементов.

Для выбора элементов или элемента из набора элементов используется метод `get()`. При указании параметра `index` возвращается элемент под указанным индексом из набора элементов. Если параметр `index` не указан, то возвращается массив элементов набора.

```
| .get(index)
```

Действие этого метода можно посмотреть в практическом задании 5.7.

Преобразование набора элементов в массив.

Метод `.toArray()` преобразует набор элементов в массив

```
| .toArray(index)
```

Индекс элемента в массиве

Метод `index()` возвращает индекс заданного элемента в наборе.

```
| $('#one').index()
```

Размер набора элементов

Метод `size()` возвращает количество элементов в наборе элементов

```
| $('div').size()
```

Пользовательские данные элемента

К любому обернутому в jQuery элементу можно добавить пользовательские переменные с помощью метода `data()`.

К элементу можно задать одну переменную:

```
| .data(nameVar, value )
```

К элементу можно задать группу переменных

```
| .data({nameVar1: value, nameVar2:value, ...} )
```

Для возвращения одной переменной, установленной с помощью метода `data()` используется следующая запись:

```
| .data('nameVar1')
```

Для возвращения всех переменных, установленных с помощью метода `data()` используется следующая запись:

```
| .data()
```

Для удаления установленной переменной методом `data()`, используется метод `removeData()`

```
| .removeData ('nameVar1')
```

Для проверки наличия у элемента переменной, установленной методом `data()`, используется метод `hasData()`

Практические задания к теме 5

Практическое задание 5.1 Поиск родительских элементов

В этом задании показано, как осуществляется выбор родительских элементов. Создайте новый html файл и сохраните его под именем **practicejQuery5_01.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 5.1 Поиск родительских элементов</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("#closest").click(function () {
        $("#levelThree").closest("div").css("background-color","yellow");
    })
    $("#parents").click(function () {
        $("#levelThree").parents("div").css("background-color","red");
    })
}
</script>
<style type="text/css">
#levelOne
{
    padding:25px;height:200px; width:200px; background-color:#C30;
}
#levelTwo
{
    padding:25px; height:150px; width:150px; background-color:#39C;
}
```

```

#levelThree
{
    padding:25px; height:100px; width:100px; background-color:#396; font-
size:20px;
    display:block;
}
</style>
</head>
<body>
<div id="levelOne">
    <div id="levelTwo" >
        <span id="levelThree" >
            <a href="#" title="Ссылка">Ссылка</a>
        </div>
    </div>
</div>
<input type="button" id='closest' value="Метод closest()">
<input type="button" id='parents' value="Метод parents()">
</body>
</html>

```

Практическое задание 5.2 Выбор элементов(,) лежащих после указанного

В этом задании показано, как осуществляется выбор родительских элементов. Создайте новый html файл и сохраните его под именем **practicejQuery5_02.html**. Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 5.2 Выбор элементов(,) лежащих после
указанного</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()

```

```
{
    $("#next").click(function () {
        $("div").eq(0).next("div").css("background-color","yellow");
    })
    $("#nextAll").click(function () {
        $("div").eq(0).nextAll("div").css("background-color","blue");
    })
    $("#nextUntil").click(function () {
        $(".top").nextUntil(".bottom").css("background-
color","green");
    })
}
</script>
<style type="text/css">
div
{
    padding:25px;margin:10px;float:left;background-color:red;
}
</style>
</head>
<body>
    <div class='top'>
        <a href='#' title="Первая ссылка">1</a>
    </div>
    <div>
        <a href='#' title="Вторая ссылка ">2</a>
    </div>
    <div>
        3
    </div>
    <div >
        <i>4</i>
    </div>
</div>
```

```

<div class="bottom">
  5
</div>
<input type="button" id='next' value="Метод next()">
<input type="button" id='nextAll' value="Метод nextAll()">
<input type="button" id='nextUntil' value="Метод nextUntil()">
</body>
</html>

```

Практическое задание 5.3 Выбор элементов, лежащих перед указанным

В этом задании показано, как осуществляется выбор родительских элементов.

Создайте новый html файл и сохраните его под именем **practicejQuery5_03.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Практическое задание 5.2 Выбор элементов, лежащих после
указанного</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("#prev").click(function () {
        $("div").eq(1).prev("div").css("background-color","yellow");
    })
    $("#prevAll").click(function () {
        $(".bottom").prevAll().css("background-color","blue");
    })
    $("#prevUntil").click(function () {
        $(".bottom").prevUntil(".top").css("background-
color","green");
    })
}
}

```

```
</script>
<style type="text/css">
div
{
    padding:25px; margin:10px; float:left; background-color:red;
}
li
{
    padding:25px; margin:10px; float:left; background-color:#3CC; display:block;
}
</style>
</head>
<body>
    <li class='top'>
        <a href='#' title="Первая ссылка">1</a>
    </li>
    <div>
        <a href='#' title="Вторая ссылка ">2</a>
    </div>
    <div>
        3
    </div>
    <div >
        <i>4</i>
    </div>
    <div class="bottom">
        5
    </div><br>
    <input type="button" id='prev' value="Метод prev()">
    <input type="button" id='prevAll' value="Метод prevAll()">
    <input type="button" id='prevUntil' value="Метод prevUntil()">
</body>
```

```
| </html>
```

Практическое задание 5.4 Фильтрация элементов набора

В этом задании показано, как осуществляется фильтрация элементов набора.

Создайте новый html файл и сохраните его под именем **practicejQuery5_04.html**.

Введите следующий код:

```
| <!doctype html>
| <html>
| <head>
| <meta charset="windows-1251">
| <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
| <script type="text/javascript">
| window.onload=function()
| {
|     $("div").css("border","2px solid green")
|     .filter(".filt")
|     .css("border","2px solid yellow");
|     $("div").click(function(){
|         if($(this).is(".filt"))
|             {
|                 alert("div имеет класс test");
|             }
|         else
|             {
|                 alert("div не имеет класс filt");
|             }
|     });
|     $("div").filter(".filt").slice(1).css("background-color","green")
| }
| </script>
| <style type="text/css">
| div
```

```
{
    padding:25px; margin:10px; float:left; background-color:red;
}
</style>
</head>
<body>
<div></div>
<div class="filt"></div>
<div class="filt"></div>
<div></div>
<div class="filt"></div>
<div></div>
</body>
</html>
```

Практическое задание 5.5 Объединение выбранных наборов

В этом задании показано, как осуществляется объединение выбранных наборов. Создайте новый html файл и сохраните его под именем **practicejQuery5_05.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$("div.one").find("p").css("border","2px solid green");
$("div.two").find("p").css("border","2px solid yellow");
$("div.three").find("p").addBack().css("border","2px solid blue");
}
</script>
<style type="text/css">
div
```

```

{
    padding:25px; margin:10px; float:left; background-color:red;
}
</style>
</head>
<body>
<div class="one"><p>Первый</p></div>
<div class="two"><p>Второй</p></div>
<div class='three'><p>Третий</p></div>
</body>
</html>

```

Практическое задание 5.6 Содержимое элемента

В этом задании показано, как осуществляется выбор содержимого элемента.

Создайте новый html файл и сохраните его под именем **practicejQuery5_06.html**.

Введите следующий код:

```

<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("#children").click(function ()
    {
        var $a=$("#one").children()
        alert($a.text());
    })
    $("#contents").click(function ()
    {
        var $a=$("#one").contents()
        alert($a.text());
    })
}

```

```
    $("#jQ").click(function()
    {
        $('i').contents().filter(function() {
            return this.nodeType == 3;
        })
        .wrap('<strong></strong>')
    })
}
</script>
<style type="text/css">
div
{
    padding:25px; margin:10px; background-color:red; width:140px;
}
</style>
</head>
<body>
<div class="one">
    Текстовый узел 1
    <p>Текстовый узел 2</p>
    Текстовый узел 3
</div>
<input type="button" id='children' value="Метод children()">
<input type="button" id='contents' value="Метод contents()">
<p class='jQ'>
<i>jQuery</i> — библиотека <i>JavaScript</i>, фокусирующаяся на
взаимодействии <i>JavaScript</i> и <i>HTML</i>. Библиотека <i>jQuery</i>
помогает легко получать доступ к любому элементу <i>DOM</i>, обращаться к
атрибутам и содержимому элементов <i>DOM</i>, манипулировать ими. Также
библиотека <i>jQuery</i> предоставляет удобный API по работе с
<i>AJAX</i>.</p>
<input type="button" id='jQ' value="Заменить курсив полужирным начертанием">
</body>
</html>
```

Практическое задание 5.7 Функция для элемента в наборе элемента

В этом задании показано, как осуществляется элементы в наборе элементов.

Создайте новый html файл и сохраните его под именем **practicejQuery5_07.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
    $("#get").click(function() {
        alert($("#div").get(1).tagName)
        alert($("#div").get(1).className)
        alert($("#div").get())
    })
    $("#start").click(function () {
    $("#div").each(function (index,elementDOM)
        {
            $(elementDOM).css( "backgroundColor", "#95AFC1" ).text("DIV " + index);
        });
    });
}
</script>
<style type="text/css">
.red
{
    padding:25px; margin:10px; background-color:red; float:left;
}
</style>
</head>
<body>
```

Тема 5. Методы работы с наборами элементов

```
<div class="red"></div><div class="red"></div> <p class="red"></p><div  
class="red"></div>  
<div class="red"></div><div class="red"></div>  
<input type="button" id='start' value="Пуск!">  
<input type="button" id='get' value="Метод get()">  
</body>  
</html>
```

Самостоятельная работа к теме 5

Самостоятельная работа 5.1 Выбор позиционированного родителя

Создайте два div-элемента. Один из элементов должен быть вложен в другой. Выберите родительский div-элемент, используя метод `offsetParent()` и переместите его на 100px влево от положения по умолчанию.

Самостоятельная работа 5.2 Содержимое элемента

Отредактируйте практическое задание 5.6 так, чтобы при выделении текста полужирным начертанием курсив удалялся.

Самостоятельная работа 5.3 Массив элементов

В практическом задании 5.7 извлеките параграф, используя метод `toArray()`.

Тема 6. Анимация и эффекты jQuery.

Используя функции jQuery, можно анимировать элементы веб-страницы. Все эффекты jQuery основываются на изменении CSS свойств. Можно использовать уже готовые эффекты, такие как, скрывание, появление, движение элементов или создавать свои более сложные, основанные на изменении нескольких свойств элемента.

Встроенные функции анимации.

В этом разделе рассматриваются функции, скрывающие элементы при помощи изменения видимости, размеров и прозрачности.

Появление и скрывание элемента.

Функция `.show()` позволяет показывать скрытый элемент. Элемент можно показать мгновенно, для этого следует использовать следующую запись:

```
$("#div").show();
```

Функция `.hide()` позволяет скрывать элемент. Элемент можно скрыть мгновенно, для этого следует использовать следующую запись:

```
$("#div").hide();
```

При скрывании элемента ему присваивается свойство `display` со значением `none`, при появлении возвращается исходное значение свойства `display`.

Используя дополнительные значения этих функций, можно плавно скрывать/показывать элемент, для этого следует добавить параметр продолжительности действия анимации (`duration`). В следующем примере показано, как установить скорость скрывания 400 миллисекунд.

```
$("#div").hide(400)
```

Значения `duration` задается в миллисекундах или с помощью строкового значения «fast» (200 миллисекунд) и «slow» (600 миллисекунд).

При завершении действия скрывать/показывать элемент, можно запустить функцию, внутри которой переменная `this` будет содержать анимированный элемент.

```
$("#div").hide(400, function () {})
```

И последний параметр `easing`, который определяет изменение скорости анимации. В стандартном jQuery доступны две такие функции: «linear» и «swing». Функция «swing» применяется по умолчанию и ускоряет анимации к ее завершению, а функция «linear» делает анимацию равномерной.

```
$("#div").hide(400, 'linear', function () {})
```

Действие этих функций можно посмотреть в практическом задании 6.1.

Сворачивание и разворачивание элемента

Для плавного сворачивания/разворачивания элемента используются функции `.slideDown()` и `.slideUp()`. Эти функции похожи на предыдущие функции скрытия/показа элементов и обладают теми же параметрами.

```
$("#div").slideDown(400, function () {})  
$("#div").slideUp(400, 'linear', function () {})
```

При скрытии элемента ему присваивается свойство `display` со значением `none`, при появлении возвращается исходное значение свойства `display`.

Поочередное разворачивание и сворачивание элементов

Функция `.slideToggle()` последовательно сворачивает элемент или, если элемент свернут, разворачивает его. Эта функция имеет параметры, которыми обладают предыдущие функции.

```
$("#div").slideToggle (400, 'linear', function () {})
```

Изменение прозрачности элемента

С помощью функций `.fadeIn()` и `.fadeOut()` можно скрывать/показывать элементы, за счет плавного изменения прозрачности.

```
$("#div").fadeIn(400, function () {})  
$("#div").fadeOut (400, 'linear', function () {})
```

При скрытии элемента ему присваивается свойство `display` со значением `none`, при появлении возвращается исходное значение свойства `display`.

Изменение уровня прозрачности элемента

Функция `.fadeTo()` позволяет изменять прозрачность элемента до заданного уровня. Помимо основных параметров (продолжительности действия анимации (`duration`), запуска функции по завершении анимации и параметра `easing`) для этой функции добавлен параметр **opacity**, который определяет степень прозрачности элемента от 1 (не прозрачен) до 0 (прозрачен).

```
$("#div").fadeTo(400, 0.4)
```

Действие функций `.fadeIn()`, `.fadeOut()` и `.fadeTo()` можно посмотреть в практическом задании 6.2.

Поочередное появление и скрытие элементов

Функция `fadeToggle()` позволяет последовательно показывать и скрывать элемент, так как это делают методы `.fadeIn()` и `.fadeOut()`, то есть плавно изменяя прозрачность. Эта функция поддерживает те же параметры, что и предыдущие функции.

```
$("#div").fadeToggle(400, function () {})
```

При скрытии элемента ему присваивается свойство `display` со значением `none`, при появлении возвращается исходное значение свойства `display`.

Управление анимацией

Создание пользовательской анимации

Пользовательская анимация создается с помощью функции `.animate()` и осуществляется за счет изменения указанных свойств CSS элемента.

Функция `.animate()` обладает следующими необязательными параметрами (знакомыми нам по предыдущим функциям анимации): продолжительности действия анимации (`duration`), запуска функции по завершении анимации и параметра `easing`. Дополнительный параметр, добавляемый первым, определяет изменяемые CSS свойства и называется **properties**. В следующем примере показан синтаксис функции `.animate()`.

```
$("#div").animate(properties, duration, easing, function)
```

Параметр `properties` определяет конечные значения изменяемых CSS свойств, то есть если высота элемента по умолчанию 500px, а в качестве значения параметра `properties` вы укажете 200px, то размер элемента изменится с 500px до 200px. Функция `animate()` может изменять только CSS свойства, значениями которых являются числа, строковые значения или шестнадцатеричные значения изменены быть не могут. Помимо абсолютных единиц могут применяться относительные значения, например `width:"-+=100"`, что изменит ширину элемента на 100px. Кроме того, можно задавать стандартные значения "hide", "show", "toggle", которые скроют, покажут или изменят видимость элемента на противоположную.

```
$('#div').animate({ width: "+=50", height: "200px"},5000);
```

Можно последовательно выполнять несколько анимационных эффектов заданных, как функцией `animate`, так и другими функциями анимации. В следующем примере к элементу с классом `animeConsecutivelyEl` последовательно применяется две функции `animate()` и функция `fadeTo()`. То есть в этом примере показана **очередь** из трех функций.

```
$(".animeConsecutivelyEl").animate({ width:"-+=50", borderWidth:"7px"},5000)
    .animate({ height: "200px"},5000)
    .fadeTo(400,0.2)
```

В практическом задании 6.3 показано использование функции `.animate()`. Обратите внимание на различия между последовательной и одновременной анимациями.

Изменение очереди анимации

Добавляя последовательно анимации к объекту, мы создаем очередь анимации, которая будет выполняться вне зависимости от промежуточных функций.

Давайте рассмотрим следующий пример.

```
$('#animeConsecutively').click(function()
{
    $(".animeConsecutivelyEl").animate({ width:"-+=50",
```

```
borderWidth:"7px"},5000);
    $(".animeConsecutivelyEl").animate({ height: "200px"},5000);
    $(".animeConsecutivelyEl").css("background-color",'red');
    $(".animeConsecutivelyEl").fadeTo(400,0.2);
  });
```

Для элемента с классом animeConsecutivelyEl назначена очередь из различных функций и три из них функции анимации, а одна меняет фоновый цвет. При выполнении данного кода три функции анимации будут помещены в свою очередь и будут выполняться последовательно, а функция изменения фонового цвета будет помещена в свою очередь и выполнится одновременно с первой функцией animate().

Для того чтобы последовательно выполнять не только функции анимации, но и другие функции используется метод .queue().

Изменим предыдущий пример и определим в очередь изменение фонового цвета.

```
$('#animeConsecutively').click(function()
  {
    $(".animeConsecutivelyEl").animate({ width:"-=50",
borderWidth:"7px"},5000);
    $(".animeConsecutivelyEl").animate({ height: "200px"},5000);
    $(".animeConsecutivelyEl").queue(function () {
    $(this).css("background-color",'red');
    $(this).dequeue();
  });
    $(".animeConsecutivelyEl").fadeTo(400,0.2);
  });
```

Там где необходимо в очередь анимации вставить изменение фонового цвета добавляем функцию queue(), внутри которой определяем функцию с необходимым изменением фона. Теперь после второй анимации будет изменен цвет фона. Для того чтобы выйти из вставленной функции и вернуться в очередь анимации следует добавить метод .dequeue() или для вызываемой функции добавить параметр next (.queue(function (next) {})).

Очистка очереди функций

Для очистки очереди функций у указанных элементов используется метод .clearQueue(). В следующем примере выполнится только первая анимация все последующие, стоящие в очереди, будут удалены.

```
$(".animeConsecutivelyEl")
```

```
.animate({ width:"-50", borderWidth:"7px"},5000);  
.animate({ height: "200px"},5000)  
.fadeTo(400,0.2);  
$(".animeConsecutivelyEl").clearQueue();
```

Приостановка анимации

По умолчанию следующая в очереди анимация начинается сразу после завершения предыдущей. С помощью метода `.delay()` можно установить задержку в миллисекундах между анимациями. В следующем примере между анимациями установлен промежуток равный одной секунде.

```
$(".animeConsecutivelyEl")  
.animate({ width:"-50", borderWidth:"7px"},5000);  
.delay(1000)  
.animate({ height: "200px"},5000)
```

Завершение и остановка анимации

Для мгновенного завершения анимации используется метод `.finish()`. При использовании этого метода анимация мгновенно переходит к конечному состоянию, указанному в настройках анимации. Метод `.finish()` появился в версии 1.9 и не работает в предыдущих версиях.

Для остановки анимации в данный момент используется метод `.stop()`. Этот метод имеет несколько способов использования в зависимости от наличия и количества установленных параметров.

В следующем примере метод `stop()` прекратит действие текущей анимации и перейдет к следующей.

```
$(".animeEl").stop()
```

Для остановки всей очереди анимации, следует установить для параметра `queue` значение `true`, как показано в следующем примере.

```
$(".animeEl").stop(true)
```

Параметр `jumpToEnd` определяет оставаться элементу в том состоянии, при котором был запущен метод `stop()` или принять состояние, заданное по умолчанию для данной анимации, как показано в следующем примере.

```
$(".animeEl").stop(true, true)
```

Действие методов `finish()` и `stop()` можно посмотреть в практическом задании 6.5.

Изменение скорости анимации

По умолчанию скорость анимации jQuery равняется 13 миллисекундам, но это значение можно изменить, используя свойство `$.fx.interval`. Для указания временного промежутка между кадрами анимации следует указать значение этого

свойства, как показано в примере:

```
| $.fx.interval=500
```

В свойстве \$.fx.interval fx является названием очереди, принятом по умолчанию jQuery.

Отмена всех анимации

Для отмены всех анимации на странице используется свойство jQuery.fx.off с установленным значением true, соответственно для запуска остановленной анимации значение свойства jQuery.fx.off следует установить, как false.

```
| jQuery.fx.off= true
```

Выбор анимированного элемента

Для выбора анимированного в данный момент элемента используется селектор :animated. В следующем примере для div-элемента с действующей анимации будет изменен фоновый цвет.

```
| $( "div:animated" ).css( "backgroundColor","red" );
```

Практические задания к теме 6

Практическое задание 6.1 Появление и скрытие элемента

В этом задании показано действие функций `show()` и `hide()`, которые позволяют скрывать/показывать элемент.

Создайте новый html файл и сохраните его под именем **practicejQuery6_01.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 6_01. Появление и скрытие элемента</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#speedily').click(function()
                                {$(".hiddenSpeed").hide()}
                                );
$('#speedily').dblclick(function()
                                {$(".hiddenSpeed").show()}
                                );

$('#duration').click(function()
                                {$(".hiddenDuration").hide(400)}
                                );
$('#duration').dblclick(function()
                                {$(".hiddenDuration").show("slow","linear", function()
{
                                $(this).css("background-color","#B6DADA")
                                }
                                )})
                                );
}
```

```

}
</script>
<style type="text/css">
#speedily, .hiddenSpeed, #duration, .hiddenDuration
{
    padding:25px;
    margin:10px;
    border:2px solid #0C6;
    background-color:#FF9831;
    width:200px;
}
#cont
{
    padding:25px;
    margin:10px;
    background-color:#B6DADA;
    width:270px;
    float:left;
}
</style>
</head>

<body>
<div id='cont'> Без анимации
    <div id='speedily'>
        Скрыть быстро - один щелчок<br>
        Показать быстро - два щелчка
    </div>
    <div class='hiddenSpeed'>
        Скрываемый/показываемый <br>
        элемент
    </div>

```

```
</div>
<div id='cont'> С анимацией
  <div id='duration'>
    Скрыть - один щелчок<br>
    Показать - два щелчка
  </div>
  <div class='hiddenDuration'>
    Скрываемый/показываемый <br>
    элемент
  </div>
</div>
</body>
</html>
```

Практическое задание 6.2 Использование прозрачности

В этом задании показано действие функций `.fadeIn()`, `.fadeOut()` и `.fadeTo()`, которые позволяют скрывать/показывать элемент, изменяя прозрачность элемента.

Создайте новый html файл и сохраните его под именем **practicejQuery6_02.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 6_02. Изменение прозрачности элемента</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#opacity').click(function()
    $('#opacityEl').fadeOut(500)}
);
$('#opacity').dblclick(function()
    $('#opacityEl').fadeIn("slow","linear")}
```

```

        );
    $('#opacityTo').click(function()
        {$(".hiddenDuration").fadeOut(400,0.2)}
        );
    $('#opacityTo').dblclick(function()
        {$(".hiddenDuration").fadeOut("slow",0.9)}
        );
    }
</script>
<style type="text/css">
#opacity, .opacityEl, #opacityTo, .hiddenDuration
{
    padding:25px;
    margin:10px;
    border:2px solid #0C6;
    background-color:#FF9831;
    width:200px;
}
#cont
{
    padding:25px;
    margin:10px;
    background-color:#B6DADA;
    width:270px;
    float:left;
}
</style>
</head>
<body>
<div id='cont'> Скрытие элемента
    <div id='opacity'>
        Скрыть - один щелчок<br>

```

```
    Показать - два щелчка  
</div>  
<div class='opacityEl'>  
Скрываемый/показываемый <br>  
элемент  
</div>  
</div>  
<div id='cont'> С анимацией  
<div id='opacityTo'>  
    Скрыть - один щелчок<br>  
    Показать - два щелчка  
</div>  
<div class='hiddenDuration'>  
    Скрываемый/показываемый <br>  
    элемент  
</div>  
</div>  
</body>  
</html>
```

Практическое задание 6.3 Одновременная и последовательная анимация

В этом задании показано действие функций `.animate`, которые позволяют анимировать изменения элемента.

Создайте новый html файл и сохраните его под именем **practicejQuery6_03.html**.

Введите следующий код:

```
<!doctype html>  
<html>  
<head>  
<meta charset="windows-1251">  
<title>Задание 6_03. Одновременная и последовательная анимация</title>  
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>  
<script type="text/javascript">  
window.onload=function()  
{
```

```

$('#anime').click(function()
    {$(".animeEl").animate({ width:"-50", height: "200px",
borderWidth:"7px"},5000)}
    );
$('#animeConsecutively').click(function()
    {
    $(".animeConsecutivelyEl").animate({ width:"-50",
borderWidth:"7px"},5000)
    .animate({ height: "200px"},5000)
    .fadeTo(400,0.2)
    });
}
</script>
<style type="text/css">
#anime, .animeEl, #animeConsecutively, .animeConsecutivelyEl
{
    padding:25px;
    margin:10px;
    border:2px solid #0C6;
    background-color:#FF9831;
    width:200px;
}
#cont
{
    padding:25px;
    margin:10px;
    background-color:#B6DADA;
    width:270px;
    float:left;
}
</style>
</head>
<body>

```

```
<div id='cont'>
  <div id='anime'>
    Одновременная анимация
  </div>
  <div class='animeEl'>
  </div>
</div>
<div id='cont'>
  <div id='animeConsecutively'>
    Последовательная анимация
  </div>
  <div class='animeConsecutivelyEl'>
  </div>
</div>
</body>
</html>
```

Практическое задание 6.4 Вставка функции в очередь

В этом задании показано, как следует вставлять функции в очередь анимации. Создайте новый html файл и сохраните его под именем **practicejQuery6_04.html**. Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 6_03. Одновременная и последовательная анимация</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#anime').click(function()
    {
        $(".animeEl").animate({ width:"-50", borderWidth:"7px"},5000)
        $(".animeEl").animate({ height: "200px"},5000)
```

```

        $(".animeEl").css("background-color",'red')
        $(".animeEl").fadeTo(400,0.2)
    });
    $('#animeConsecutively').click(function()
    {
        $(".animeConsecutivelyEl").animate({ width:"-50",
borderWidth:"7px"},5000)
        $(".animeConsecutivelyEl").animate({ height: "200px"},5000)
        $(".animeConsecutivelyEl").queue(function () {
    $(this).css("background-color",'red');
        });
        $(".animeConsecutivelyEl").fadeTo(400,0.2)
        });
    }
</script>
<style type="text/css">
#anime, .animeEl, #animeConsecutively, .animeConsecutivelyEl
{
    padding:25px;
    margin:10px;
    border:2px solid #0C6;
    background-color:#FF9831;
    width:200px;
}
#cont
{
    padding:25px;
    margin:10px;
    background-color:#B6DADA;
    width:270px;
    float:left;
}
</style>

```

```
</head>
<body>
<div id='cont'>
  <div id='anime'>
    Одновременная анимация
  </div>
  <div class='animeEl'>
  </div>
</div>
<div id='cont'>
  <div id='animeConsecutively'>
    Последовательная анимация
  </div>
  <div class='animeConsecutivelyEl'>
  </div>
</div>
</body>
</html>
```

Практическое задание 6.5 Остановка анимации

В этом задании показано, как можно останавливать анимацию.

Создайте новый html файл и сохраните его под именем **practicejQuery6_05.html**.

Введите следующий код:

```
<!doctype html>
<html>
<head>
<meta charset="windows-1251">
<title>Задание 6_03. Одновременная и последовательная анимация</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script type="text/javascript">
window.onload=function()
{
$('#anime').click(function()
{
```

```

        $(".animeEl").animate({ width:"100", borderWidth:"7px"},5000)
        $(".animeEl").animate({ height: "200px"},5000)
        $(".animeEl").fadeTo(400,0.2)
    });
    $("#finish").click(function()
    {
        $(".animeEl").finish()
    });
    $("#stop").click(function()
    {
        $(".animeEl").stop()
    });
    $("#stopTrue").click(function()
    {
        $(".animeEl").stop(true)
    });
    $("#stopTrueTrue").click(function()
    {
        $(".animeEl").stop(true, true)
    });
}
</script>
<style type="text/css">
#anime, .animeEl
{
    padding:25px;
    margin:10px;
    border:2px solid #0C6;
    background-color:#FF9831;
    width:200px;
}
#cont

```

```
{
    padding:25px;
    margin:10px;
    background-color:#B6DADA;
    width:270px;
    float:left;
}
</style>
</head>
<body>
<button id="finish">.finish()</button>
<button id="stop">.stop()</button>
<button id="stopTrue">.stop(true)</button>
<button id="stopTrueTrue">.stop(true, true)</button>
<br>
<div id='cont'>
    <div id='anime'>
        Анимация
    </div>
    <div class='animeEl'>
    </div>
</div>
</body>
</html>
```

Самостоятельная работа к теме 6

Самостоятельная работа 6.1 Сворачивание и разворачивание элемента

Используя функции `.slideDown()` и `.slideUp()`, скройте/покажите элементы на странице.

Используя функцию `.slideToggle()`, сверните и разверните элемент.

Самостоятельная работа 6.2 Последовательное изменение прозрачности элемента

Используя функцию `fadeToggle()`, скройте и покажите элемент.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА КОМПЬЮТЕРНОГО ПРОЕКТИРОВАНИЯ И ДИЗАЙНА

Кафедра компьютерного проектирования и дизайна входит в состав Академия методов и техники управления (ЛИМТУ). Академия является структурным подразделением Университета ИТМО.

На кафедре проводится обучение по следующим направлениям:

- **Подготовка магистров** по направлению 230400 «Информационные системы и технологии», магистерская программа – «Компьютерная графика и Web-дизайн».
- **Переподготовка и повышение квалификации специалистов** с широким спектром образовательных программ по двум направлениям:
Компьютерная графика применительно к полиграфии, разработке Web-узлов и дизайну интерьера.
Компьютерное проектирование применительно к машиностроению, строительству, архитектуре и геоинформационным технологиям.
- **Сертифицированное обучение в авторизованном учебном центре (АТС) Autodesk.**

На кафедре реализована система дистанционного обучения (ДО) по основным программам переподготовки и повышения квалификации специалистов.

Дистанционное обучение обеспечивает возможность пройти переподготовку с использованием Internet в любом месте в любое время. Дистанционное обучение дешевле традиционных форм обучения. Дисциплины в формате ДО специально структурированы, имеют большее информационное насыщение и значительный объем тестов для самопроверки и контроля

При кафедре действует рисовальный класс для взрослых, где желающие постигают искусство рисования и живописи.

Более подробную информацию о других программах и деятельности кафедры можно посмотреть на сайте www.limtu.ru

Перепелица Филипп Александрович

**Разработка интерактивных сайтов с использованием
jQuery**

Учебное пособие

В авторской редакции

Редакционно-издательский отдел Университета ИТМО

Зав. РИО

Н.Ф. Гусарова

Подписано к печати

Заказ №

Тираж

Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49