

Лабораторная работа 1. Основные функции jQuery и работа с ними

jQuery – это JavaScript-библиотека, фокусирующаяся на взаимодействии JavaScript, HTML и CSS.

Основные функции jQuery:

- 1) Обращаться к любому элементу DOM (объектной модели документа) и не только обращаться, но и манипулировать ими.
- 2) Работать с событиями.
- 3) Легко осуществлять различные визуальные эффекты.
- 4) Работать с AJAX (очень полезная технология, позволяющая общаться с сервером без перезагрузки страницы)
- 5) Создавать элементы пользовательских интерфейсов.

Для работы с jQuery библиотеку нужно скачать [на сайте разработчиков](#) или [берем](#) из общей папки сети, при необходимости распаковать и перенести файл jquery-x.x.x.js в ту папку, из которой вы собираетесь его подключать.

Для подключения библиотеки jQuery к html-странице. Используется тег `<script>`, как и в ситуациях с подключением всех внешних файлов скриптов .

```
<html>
<head>
<title>jQuery</title>
<link rel="stylesheet" type="text/css"
href="style.css">
<script type="text/javascript" src="jquery-
x.x.x.js"></script>
<script type="text/javascript"
src="script.js"></script>
</head>
<body>
</body>
</html>
```

Обратите внимание на имя файла. Здесь используется библиотека jquery-х.х.х, без указания версии, но Вы должны указать версию, которой собираетесь воспользоваться.

Задание 1.Создайте в папке html-файл(index.html), стилевой файл (style.css), файл, который будет содержать js-функции (script.js) и копируем туда файл библиотеки jQuery (jquery-х.х.х.js).

Создадим первую страницу, на которой покажем работу эффектов jQuery .
Результат должен получиться как на рисунках 1 и 2.

Эффекты jQuery

Show() SlaiideDown() Animate()

Рисунок 1. Web-страница до нажатия на кнопки

Эффекты jQuery



Рисунок 2. Web-страница после нажатия на кнопки

На html-странице (рисунок 1) три прямоугольника(скрыты до начала работы скриптов) и три кнопки:

```

<html>
<head>
<title>jQuery</title>
<link rel="stylesheet" type="text/css"
href="style.css">
<script type="text/javascript" src="jquery-
x.x.x.js"></script>
<script type="text/javascript"
src="script.js"></script>
</head>
<body>
<h2>Эффекты jQuery</h2>
<div id="v1" >1</div>
<div id="v2" >2</div>
<div id="v3" >3</div>
<button onclick="addEffect1()">Эффект Show()</button>
<button onclick="addEffect2()">Эффект
SlideDown()</button>
<button onclick="addEffect3()">Эффект
Animate()</button>
</body>
</html>

```

Оформите внешний вид и сделайте прямоугольники невидимыми с помощью CSS самостоятельно.

На странице script.js пишем код функций, которые будут срабатывать при нажатии на кнопки:

```

function addEffect1() {
$("#kv1:hidden").show();
}
function addEffect2() {
$("#kv2:hidden").slideDown("slow");
}
function addEffect3() {
$("#kv3:hidden").show().animate( {
fontSize:"36px"
} , 3000);
}

```

Описание функций, использованных в данном задании.

Функция `addEffect1()` видит `$` (знак доллара) и понимает, что это jQuery, затем она видит `"#kv1:hidden"` и понимает, что ей нужно найти элемент с `id="kv1"`, обладающий свойством `hidden` (невидимый). Далее она видит `.show()` и понимает, что найденный элемент надо сделать видимым.

Функция `addEffect2()` видит `$` (знак доллара) и понимает, что это jQuery, затем она видит `"#kv2:hidden"` и понимает, что ей нужно найти элемент с `id="kv2"`, обладающий свойством `hidden` (невидимый). Далее она видит `.slideDown("slow")` и понимает, что найденный элемент надо медленно ("slow") отобразить сверху-вниз (`slideDown`).

Функция `addEffect3()` видит `$` (знак доллара) и понимает, что это jQuery, затем она видит `"#kv3:hidden"` и понимает, что ей нужно найти элемент с `id="kv3"`, обладающий свойством `hidden` (невидимый). Далее она видит `.show()` и понимает, что найденный элемент надо сделать видимым. Затем она видит `.animate({fontSize:"36px"} , 3000)` и понимает, что размер шрифта нужно за 3 секунды (3000) увеличить до 36 пикселей (`fontSize:"36px"`)

Основные понятия jQuery. Синтаксис оператора jQuery можно представить приблизительно следующим образом:

```
$("селектор:свойство селектора").действие("свойства действия");
```

Где:

- 1) **селектор** - элемент или элементы, с которыми мы будем что-либо делать
- 2) **действие** - что именно мы будем делать с выбранными элементами. Это может быть какой-либо эффект, css-стиль, изменение html-кода и т.д. Здесь же могут быть указаны какие-либо события.

3) свойства действия - если они предусмотрены действием.

Задание 1.Измените код так, чтобы прямоугольники

- 1) открывались на 50% своей высоты при наведении курсора
- 2) открывались по двойному клику

Лабораторная работа 2. JQuery селекторы

Задание селекторов

#id - выбирает единственный элемент с переданным идентификатором (id).

Пример:

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

Данный оператор выберет элемент с id="lok" и добавит ему CSS-стиль, в данном случае рамку, шириной в 1 пиксел, сплошную и красную.

.class - выбирает все элементы с переданным классом.

Пример:

```
$(".lok").css("border", "1px solid red");
```

Данный оператор выберет все элементы с class="lok" и добавит им CSS-стиль, в данном случае рамку, шириной в 1 пиксел, сплошную и красную.

elements - выбирает все элементы с выбранным именем(имеются ввиду названия тегов html).

Пример:

```
$("div").css("border", "1px solid red");
```

Данный оператор выберет все блоки и обведет их в рамки.

"*" - выбирает все элементы, включая head и body

Пример: Данный оператор добавит рамку ко всем элементам.

```
$("*").css("border", "1px solid red");
```

Данный оператор добавит рамку ко всем элементам в тегах body.

```
$("*", document.body).css("border", "1px solid red");
```

selector1, ..., selectorN - выбирает все элементы, переданные в селекторах.

Пример:

Данный оператор добавит рамку ко всем блок-ам, span-ам, и абзацам, имеющим класс lok. Таким образом, через запятую можно указать любые селекторы.

```
$("#div, span, p.lok").css("border", "1px solid red");
```

Задание 1: Создайте страницу по образцу (рисунок 3). При клике по кнопкам должны выполняться соответствующие действия(указанные на кнопках).

Обработчики событий нажатия задаем через функции.

Для этого можно использовать например, такой скрипт функции:

```
function addStyle1(){  
    $("#st1").css("background", "red");  
}
```

Остальные функции создайте самостоятельно.

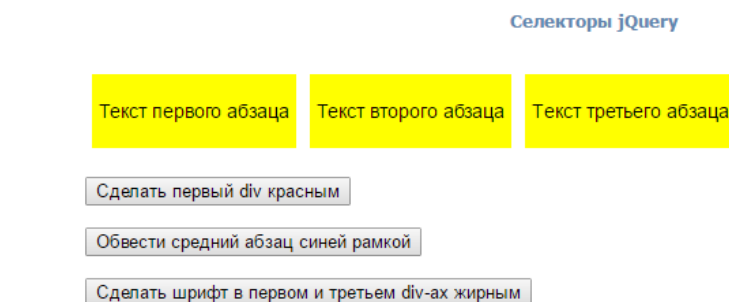


Рисунок 3. Исходная страница



Рисунок 4. Результат работы функций

Иерархические селекторы

У элементов страницы могут быть предки (`parent`) и потомки (`child`).

Например, все элементы, находящиеся на форме являются потомками формы, а форма, в свою очередь, является их предком.

В селекторах jQuery это используется следующим образом:

`parent child` - выбирает все элементы, являющиеся потомками для определенного элемента предка.

Пример:

```
$( function() {  
  $("form input ").css("border", "1px solid red")  
});
```

Данная инструкция добавит рамку всем формам, которые являются потомками элемента form.

`parent > child` - выбирает всех прямых наследников элемента-родителя.

Например, у нас есть html-страница такой структуры:

```
<body>  
  <h2>Селекторы jQuery</h2>  
  <div id="main">  
    <p></p>  
    <div></div>  
    <span></span>  
  </div>  
  <div id="content">  
    <p></p>  
    <div></div>  
    <span></span>  
  </div>  
</body>  
</html>
```

Тогда следующая инструкция:

```
$( function() {  
  $("#content > *").css("border", "1px solid red")  
});
```


добавит рамку всем элементам, которые находятся в блоке с `id="content"`, но не затронет одноименные элементы в блоке с `id="main"`.

`prev + next` - выбирает все элементы, которые являются следующими (`next`) за элементом `prev` (предыдущий).

Пример:

```
$( function(){  
    $("label + input ").css("border", "1px solid red")  
});
```

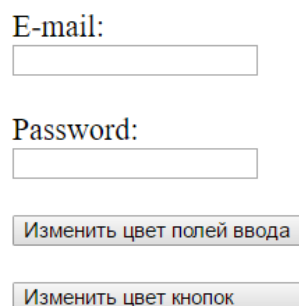
Данная инструкция добавит рамку только тем формам, которые следуют непосредственно за надписями (`label`).

Селекторы форм по атрибутам:

- `:input` - выбираются все элементы `input` .
- `:text` - выбираются все элементы `input` типа `text`.
- `:password` - выбираются все элементы `input` типа `password`.
- `:radio` - выбираются все элементы `input` типа `radio`.
- `:checkbox` - выбираются все элементы `input` типа `checkbox`.
- `:submit` - выбираются все элементы `input` типа `submit`.
- `:reset` - выбираются все элементы `input` типа `reset`.
- `:button` - выбираются все элементы `input` типа `button`.
- `:image` - выбираются все элементы `input` типа `image`.
- `:file` - выбираются все элементы `input` типа `file`.
- `:hidden` - выбираются все элементы `input` типа `hidden` или просто скрытые.

Задание 2. Сверстайте форму по образцу:

Селекторы форм jQuery



The screenshot shows a web form with the following elements:

- A label "E-mail:" followed by a text input field.
- A label "Password:" followed by a password input field.
- A button labeled "Изменить цвет полей ввода" (Change input field color).
- A button labeled "Изменить цвет кнопок" (Change button color).

Рисунок 5. Страница до работы скриптов

Сами функции будут обращаться к элементам по их атрибутам, в данном случае по типу элемента:

```
function addColor1(){
$:text, :password").css("background", "red");
}
function addColor2(){
$:button").css("background", "blue");
}
```

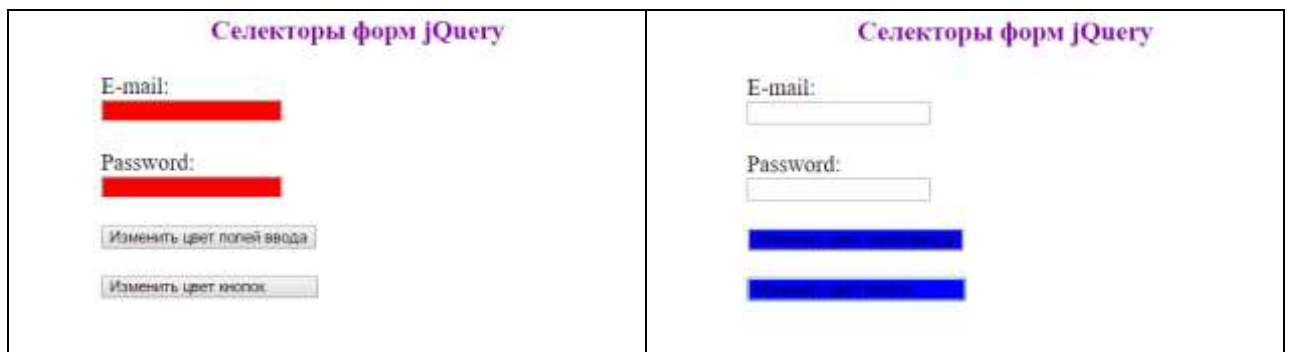


Рисунок 6. Результат работы скриптов

Напишите скрипт, который будет изменять цвет только у одной кнопки.

Лабораторная работа 3. Фильтры

:first - выбирает первый элемент соответствующего селектора.

Пример:

Данная инструкция напишет курсивом первую строку таблицы.

```
$("tr:first").css("font-style", "italic");
```

:last - выбирает последний элемент соответствующего селектора.

Пример:

Данная инструкция напишет курсивом последнюю строку таблицы.

```
$("tr:last").css("font-style", "italic");
```

:even - выбирает четные элементы, начиная с нуля.

Пример:

Данная инструкция напишет курсивом все четные строки таблицы (т.к. нумерация идет с нуля, то зрительно кажется, что нечетные строки).

```
$("tr:even").css("font-style", "italic");
```

:odd - выбирает нечетные элементы, начиная с нуля.

Пример:

Данная инструкция напишет курсивом все нечетные строки таблицы (т.к. нумерация идет с нуля, то зрительно кажется, что четные строки).

```
$("tr:odd").css("font-style", "italic");
```

:eq(index) - выбирает элемент по его индексу (начиная с нуля).

Пример:

Данная инструкция напишет курсивом текст второй ячейки таблицы (т.к. нумерация идет с нуля, то зрительно кажется, что третьей ячейки).

```
$("td:eq(2)").css("font-style", "italic");
```

:gt(index) - выбирает элементы с индексом больше указанного.

Пример:

```
$("td:gt(2)").css("font-style", "italic");
```

Данная инструкция напишет курсивом текст во всех ячейках таблицы, начиная со второй (т.к. нумерация идет с нуля, то зрительно кажется, что с третьей ячейки).

:lt(index) - выбирает элементы с индексом меньше указанного.

Пример:

```
$("td:lt(4)").css("font-style", "italic");
```

Данная инструкция напишет курсивом текст в первых пяти ячейках таблицы.

Задание 1.Создайте таблицу, у которой можно будет поменять цвет строк: например, четные строки будут серыми, а нечетные - синими, а цвет шрифта ячеек, начиная с 5 ячейки, будет желтым. Помним, что нумерация идет с нуля, поэтому внешне четные и нечетные строки меняются местами, а 5 ячейка зрительно выглядит, как шестая.

Фильтры jQuery

jQuery	четная строка
jQuery	нечетная строка
jQuery	четная строка
jQuery	нечетная строка
jQuery	четная строка
jQuery	нечетная строка

Рисунок 7. Исходная таблица



Рисунок 8. Изменение таблицы при помощи скрипта

Фильтры контента

:header - выбирает все элементы, которые являются заголовками (h1, h2...).

Пример:

```
$(":header").css("color", "red");
```

Данная инструкция сделает все заголовки красными.

:contains (text) - выбирает элементы, которые содержат переданный текст (text).

Пример:

```
$("div:contains('New')").css("text-decoration",  
"underline");
```

Данная инструкция подчеркнет все элементы блока, содержащие подстроку New.

:empty - выбирает все элементы, которые не содержат потомков (т.е. являются пустыми).

Пример:

```
$("td:empty").text("Пусто");
```

Данная инструкция найдет все пустые ячейки таблицы и вставит в них текст "Пусто".

:has(selector) - выбирает элементы, которые содержат хотя бы один элемент, указанный в селекторе.

Пример:

```
$("div:has(p)").css("font-style", "italic");
```

Данная инструкция найдет те блоки, в которых есть хотя бы один параграф (p) и сделает текст в них курсивом.

:parent - выбирает родительские элементы, т.е. те, у которых есть потомки.

Пример:

```
$("td:parent").css("font-style", "italic");
```

Данная инструкция найдет все ячейки таблицы, в которых есть текст или еще что-либо, и сделает их шрифт курсивом.

:animated - выбирает элементы, которые в данный момент являются анимированными.

Пример:

```
$("div:animated").css("border", "1px solid red");
```

Данная инструкция найдет анимированные блоки и обведет их рамкой.

Фильтры видимости

:hidden - выбирает все скрытые элементы и элементы input , имеющие тип "hidden".

Пример:

```
$("div:hidden").show();
```

Данная инструкция покажет все скрытые блоки.

:visible - выбирает все видимые элементы.

Пример:

```
$("div:visible").css("color", "red");
```

Данная инструкция сделает цвет текста в видимых блоках красным.

Фильтры атрибутов

[attribute] - выбирает все элементы с указанным атрибутом.

Пример:

```
$("div[id]").css("color", "red");
```

Данная инструкция сделает цвет текста красным во всех блоках, у которых есть идентификатор (id).

[attribute=value] - выбирает элементы с атрибутом, значение которого соответствует указанному.

Пример:

```
$("input [name='Main']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у формы с именем "Main".

[attribute!=value] - выбирает элементы с атрибутом, значение которого не соответствует указанному.

Пример:

```
$("input [name!='Main']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у всех input -ов, кроме input -а с именем "Main".

[attribute^=value] - выбирает все элементы, соответствующий атрибут которых начинается с указанного значения.

Пример:

```
$("input [name^='news']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у тех input-ов, атрибуты name которых начинаются с news.

`[attribute$=value]` - выбирает все элементы, соответствующий атрибут которых заканчивается указанным значением.

Пример:

```
$("input [name$='news']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у тех `input` -ов, атрибуты `name` которых заканчивается на `news`.

`[attribute*=value]` - выбирает все элементы, соответствующий атрибут которых содержит в качестве подстроки указанное значение.

Пример:

```
$("input [name*='news']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у тех форм, атрибуты `name` которых содержат подстроку `'news'`, неважно вначале, в конце или середине.

`[selector1]...[selectorN]` - выбирает все элементы, которые имеют указанные атрибуты и соответствующие значения.

Пример:

```
$("input [id][name*='news']").css("color", "red");
```

Данная инструкция сделает цвет текста красным у тех `input` -ов, у которых есть идентификатор и атрибуты `name` которых содержат подстроку `'news'`.

Фильтры форм

`:enabled` / `:disabled` - выбирает все элементы, имеющие активное (`enabled`) / запрещенное (`disabled`) состояние.

Пример:

```
$("input :enabled").css("color", "red");
```

Данная инструкция сделает цвет текста красным только у активных `input` -ов.

`:checked` - выбирает все элементы, которые отмечены.

Пример:

```
$("input :checked").length;
```

Данная инструкция сосчитает все отмеченные формы.

`:selected` - выбирает все выбранные элементы.

Пример:

```
$("select option:selected").length;
```

Данная инструкция сосчитает сколько выбрано элементов в списке.

Фильтры потомков

`:first-child` - выбирает элементы, которые являются первыми потомками своих родителей.

Пример:

```
$("div span:first-child").css("border", "1px solid blue");
```

Данная инструкция обведет рамкой первый span в каждом блоке.

`:last-child` - выбирает элементы, которые являются последними потомками своих родителей.

Пример:

```
$("div span:last-child").css("border", "1px solid blue");
```

Данная инструкция обведет рамкой последний span в каждом блоке.

`:only-child` - выбирает элементы, которые являются единственными потомками своих родителей.

Пример:

```
$("div button:only-child").css("border", "1px solid blue");
```

Данная инструкция обведет рамкой те блоки, которые имеют кнопку и при том только одну.

Задание 2. Создайте форму(например, для регистрации), в которой можно применить не менее шести фильтров, из данной лабораторной работы.

Лабораторная работа 4. Методы для работы с CSS-стилями

В jQuery имеется три категории методов работы с CSS:

- манипуляция с элементами, подходящими по шаблону;
- возвращение значения элемента,
- изменение самих элементов.

Чтобы добавить какому-либо элементу стиль, необходимо воспользоваться следующим методом:

`.css(name,value)`

Пример:

```
$("div").css("border", "1px solid blue");
```

В качестве параметров здесь используются названия и значения, применимые в CSS: background, border, font-style, color и т.д.

Если необходимо задать для элемента несколько CSS-правил, то лучше использовать следующую конструкцию:

`.css({properties})`

Пример:

```
$("div").css({  
border:"1px solid blue",  
fontWeight:"bolder",  
backgroundColor:"red"  
});
```

Обратите внимание, что для сложносоставных свойств CSS вроде `font-weight` и `background-color` используются эквиваленты из JS: `fontWeight`, `backgroundColor` и т.д.

Другие методы для работы со стилями:

`.addClass(class)`

Пример:

```
$("p:last").addClass("main");
```

Данная инструкция добавит класс `main` к последнему элементу параграфа.

.removeClass(class)

Пример:

```
$("p:even").removeClass("main");
```

Данная инструкция удалит класс `main` из всех четных параграфов.

.toggleClass(class)

Пример:

```
$("p").toggleClass("main");
```

Данная инструкция удалит класс `main` из всех параграфов, если он присутствует. И добавит этот класс, если он отсутствует.

.offset()

Пример:

```
var offDiv=$("div").offset();
```

Данная инструкция позволяет получить отступы слева и сверху для элемента. Чтобы получить значения конкретного свойства, нужно использовать следующие свойства: `offset.left` для отступа слева и `offset.top` - для отступа сверху.

.height(value)

Пример:

```
$("div").height();  
$("div").height(200);
```

Данная инструкция позволяет получить (первая строка) и задать (вторая строка) высоту элемента.

.width(value)

Пример:

```
$("div").width();  
$("div").width(200);
```

Данная инструкция позволяет получить (первая строка) и задать (вторая строка) ширину элемента.

Можно также применять переменную для определения стилей элементов:

```
var widDiv=$("#div").width();
$("#div.fir").width(300);
```

Первая строка запишет в переменную `widDiv` значение ширины первого блока. Вторая инструкция задаст блокам класса `fir` ширину в 300 пикселей. Это очень интересная особенность методов jQuery, они используются, как для задания параметров (когда принимаются 2 аргумента), так и для получения значений этих параметров (если передается один аргумент).

Задание 1. Создайте страницу по образцу, используя предлагаемые части кода или создайте полностью на свое усмотрение. При клике по вкладке, она меняет цвет и визуально становится частью страницы. Это один из приемов оформления меню сайта.

jQuery - Filtr_menu



Рисунок 9. Образец страницы с вкладками

```
<h3>jQuery - Filtr_menu</h3>
<div id="vkladki">
  <div class="vkl" id="vkladka1"
onClick="chang('#vkladka1');"> Вкладка 1</div>
  <div class="vkl" id="vkladka2"
onClick="chang('#vkladka2');"> Вкладка 2</div>
  <div class="vkl" id="vkladka3"
onClick="chang('#vkladka3');"> Вкладка 3</div>
  <div class="vkl" id="vkladka4"
onClick="chang('#vkladka4');"> Вкладка 4</div>
</div>
<div style="clear: both"></div>
<div class="content"></div>
<div style="clear: both"></div>
```

Все вкладки имеют единый стиль, определяемый классом `vk1`, а выделенная вкладка имеет класс `selected`.

Внешний вид определите вначале самостоятельно в файле `style.css`, а затем переопределите его при помощи jQuery.

Функция `chang()`:

```
function chang(s) {  
    $('.selected').removeClass('selected');  
    $(s).addClass('selected');  
}
```

Поясните, как работает данная функция.

Лабораторная работа 5. Методы для работы с html

Основные методы работы с html:

html (val) - добавляет html-код в выбранные элементы.

Пример:

```
$( "div.sp" ).html ( "<span>Привет</span>" );
```

Данная инструкция добавит во все блоки с классом sp, span-ы с текстом "Привет".

text () / val () - возвращает текстовое содержимое элемента / значение элемента.

Пример:

```
$( "p" ).text ();
```

Данная инструкция вернет текст из параграфа.

text (val) / val (val) - установит текст для элемента / значение для элемента.

Пример:

```
$( "p" ).text ( "Привет!" );
```

Данная инструкция в параграфе напишет слово "Привет!".

Задание 1. Создайте страницу такую, чтобы при нажатии по кнопке "Повторить", текст из первого абзаца дублировался во второй абзац.



Текст первого абзаца!

Повторить!

Рисунок 10. Исходная страница

Функция будет выглядеть так:

```
function addHtml () {  
var tp1=$( "#p1" ).text ();  
$( "#p2" ).text (tp1);  
}
```

```
}
```

Т.е. сначала в переменную `tr1` мы запишем текстовое содержимое первого абзаца (первая строка), а затем передадим его во второй абзац (вторая строка).

Измените код так, чтобы получилась следующая страница (рисунок 11)

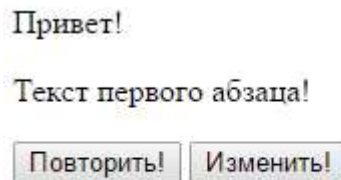


Рисунок 11. Измененная страница

Методы, позволяющие манипулировать вставляемыми элементами.

`append (content)` - добавляет `content` внутрь всех выбранных элементов *в конец* существующего контента.

Пример:

```
$ ("p") .append ("<b>Привет</b>") ;
```

Данная инструкция добавит в конец параграфа текст "Привет" жирным шрифтом.

`appendTo (content)` - обратная операция, добавляет в `content` все выбранные элементы.

Пример:

```
$ ("p") .appendTo ("#main") ;
```

Данная инструкция добавит параграф в конец элемента с идентификатором "main".

Иными словами:

`$ (A) .append (B)` - добавит B в конец A.

`$ (A) .appendTo (B)` - добавит A в конец B.

prepend (content) - добавляет content внутрь всех выбранных элементов в начало существующего контента.

Пример:

```
$ ("p").prepend("<b>Привет</b>");
```

Данная инструкция вставит в начало параграфа текст "Привет" жирным шрифтом.

prependTo (content) - обратная операция, добавляет в content все выбранные элементы.

Пример:

```
$ ("p").prependTo("#main");
```

Данная инструкция вставит параграф в начало элемента с идентификатором "main".

Иными словами:

$\$(A).prepend(B)$ - добавит B в начало A.

$\$(A).prependTo(B)$ - добавит A в начало B.

after (content) - добавляет content ПОСЛЕ всех выбранных элементов (заметьте ПОСЛЕ элемента, а не в конец элемента, как в случае с append).

Пример:

```
$ ("p").after("<b>Привет</b>");
```

Данная инструкция вставит после параграфа текст "Привет" жирным шрифтом.

befor (content) - добавляет content ДО всех выбранных элементов (заметьте ДО элемента, а не в начало элемента, как в случае с prepend).

Пример:

```
$ ("p").befor("<b>Привет</b>");
```

Данная инструкция вставит перед параграфом текст "Привет" жирным шрифтом.

insertAfter (content) - вставляет все выбранные элементы ПОСЛЕ content .

Пример:

```
$ ("p").insertAfter("#main");
```

Данная инструкция вставит параграф после элемента с идентификатором "main".

Иными словами:

`$ (A) .after (B)` - вставит B после A.

`$ (A) .insertAfter (B)` - вставит A после B.

insertBefore (content) - вставляет все выбранные элементы ПЕРЕД content .

Пример:

```
$ ("p").insertBefore("#main");
```

Данная инструкция вставит параграф перед элементом с идентификатором "main".

Иными словами:

`$ (A) .before (B)` - вставит B перед A.

`$ (A) .insertBefore (B)` - вставит A перед B.

Задание 2. Сделать так, чтобы при нажатии на кнопку "Добавить" в красном прямоугольнике появлялись желтые квадратики (один клик - один квадратик).

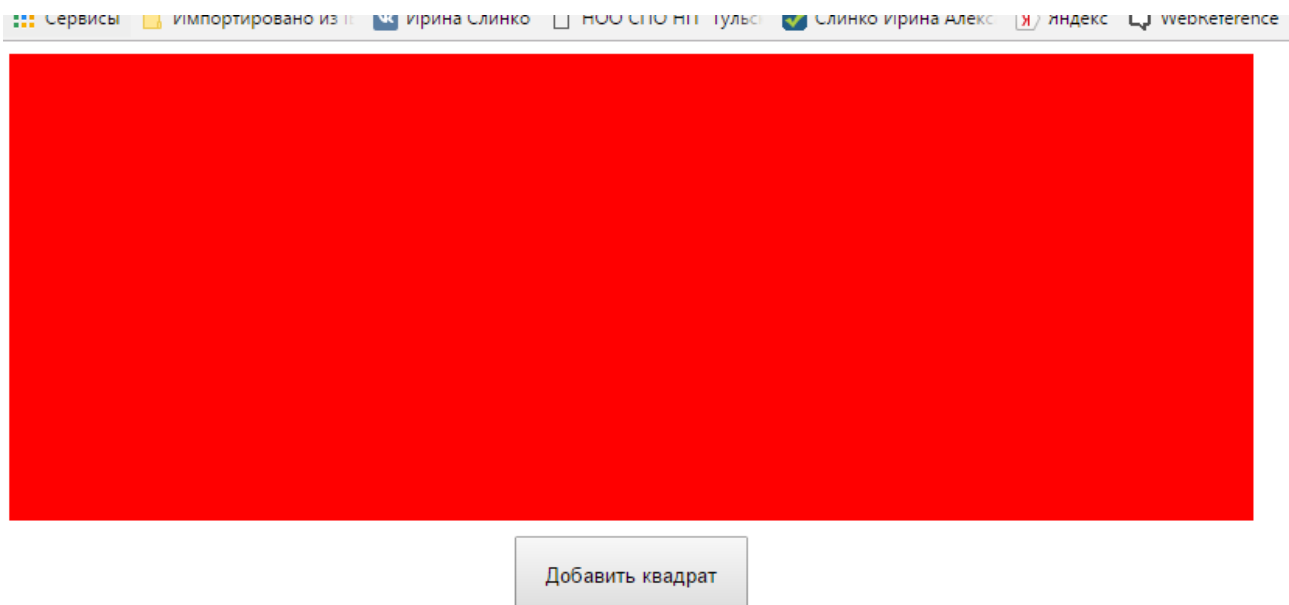


Рисунок 12. Исходная страница

Код html-страницы состоит из одного блока и одной кнопки, а желтые квадраты нужно формировать при помощи функции jQuery.

На странице style.css как всегда определяем стили для основного блока и желтых квадратов.

Код функции может быть, например таким:

```
function addYellowSquare() {  
    $(".class1").prepend("<div  
class='yellow_square'></div>");  
}
```



Рисунок 13. Результат работы скрипта

wrap (html) - оборачивает каждый элемент в элемент-обертку.

Пример:

```
$ ("p").wrap("<span></span>");
```

Данная инструкция обернет каждый параграф span-ом.

wrapAll (html) - оборачивает все элементы в ОДИН элемент обертку.

Пример:

```
$ ("p").wrapAll("<span></span>");
```

Данная инструкция обернет все параграфы одним span-ом.

wrapInner (html) - оборачивает внутренне содержание каждого элемента.

Пример:

```
$ ("p").wrapInner("<em></em>");
```

Данная инструкция выделит содержимое каждого параграфа курсивом.

replaceWith (content) - замещает одни элементы другими.

Пример:

```
$ ("span").replaceWith("<div></div>");
```

Данная инструкция заменит все span-ы блоками.

replaceAll(selector) - обратная операция, т.е. все selector-ы будут заменены на указанные элементы.

Пример:

```
$ ("span").replaceAll("<div></div>");
```

Данная инструкция заменит все блоки span-ами.

empty() - удаляет из элемента все узлы-потомки, включая текст.

Пример:

```
$ ("span").empty();
```

Данная инструкция удалит все содержимое span-ов.

remove() - удаляет сами элементы.

Пример:

```
$ ("span").remove();
```

Данная инструкция удалит все span-ы.

clone() - копирует элементы

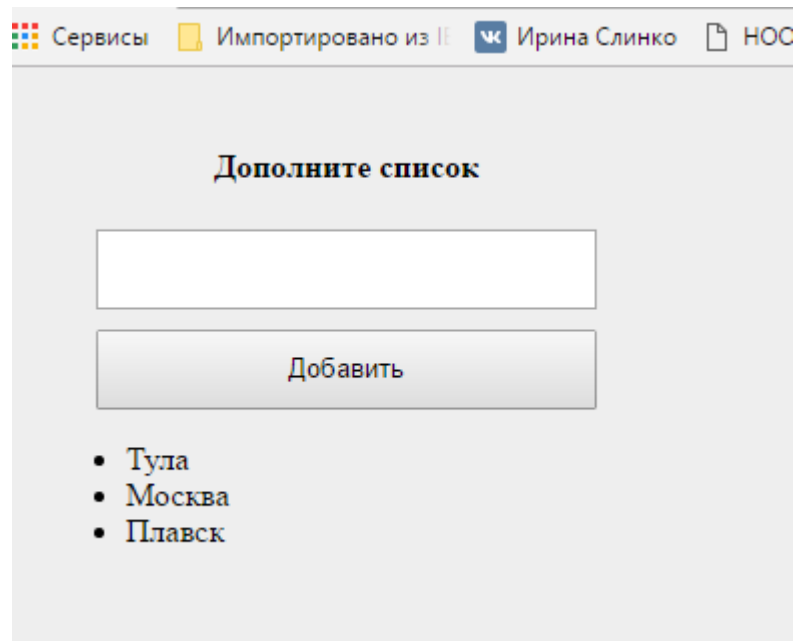
Пример:

```
$ ("b").clone().append("div");
```

Данная инструкция возьмет элементы, выделенные жирным и добавит их в блок.

Задание 3. Для предыдущего задания оберните красный блок и кнопку в общий блок «обертку» и сделайте шрифт на кнопке жирным при помощи jQuery.

Задание 4. Пусть у нас есть некоторый список(рисунок 14), и мы хотим разрешить пользователям добавлять в него пункты(рисунок 15).
Создайте страницу по образцу. Подключите рекомендуемый скрипт (можно внести свои правки) и поясните его работу в комментариях.



Сервисы Импортировано из IE Ирина Слинко НОС

Дополните список

- Тула
- Москва
- Плавск

Рисунок 14. Исходная страница

```
function addSpisok() {  
    var jq=$('#user_text').val();  
    $('#jq').append('<li>'+jq+'</li>');  
}
```

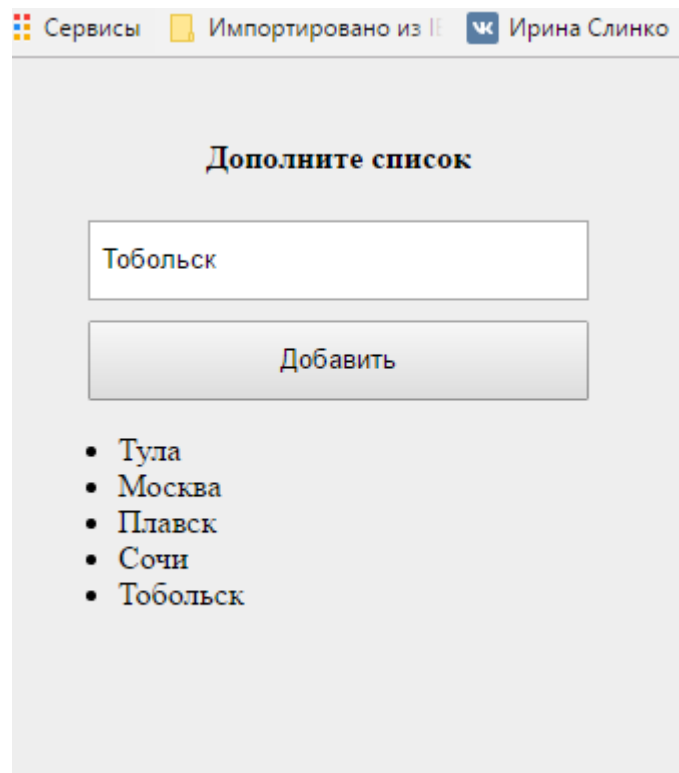


Рисунок 15. Результат работы

Задание 5. Самостоятельно, используя изученные методы, нарисуйте шахматную доску (8 x 8 клеток).