# Лабораторная работа №5

Тема: Изучение библиотеки JQuery.

**Задание**

На основе лабораторной работы №4 упростить работу с DOM.

**Требования**

* Переписать несколько функций из лабораторной работы №4 с помощью методов JQuery, проанализировать изменения в коде;
* Создать анимацию с помощью метода .animate();
* Добавить слайдер с изображениями;
* Реализовать возможность скрывать и показывать блоки;
* Использовать функции из каждой категории методов (фильтрация набора, обход набора, работа с css, работа с атрибутами и свойствами, добавление и удаление содержимого, события)

# Справочная информация

## О JQuery

jQuery - это популярная библиотека JavaScript, созданная с целью облегчить разработчикам использование JavaScript на веб-сайтах. Это не отдельный язык программирования и работает в сочетании с JavaScript.

## Подключение

Подключить библиотеку можно двумя способами:

1. Скачать с официального сайта <https://jquery.com/> и прописать путь к файлу

<script src="path/to/jquery.js"></script>

1. Воспользоваться JQuery CDN: скопировать ссылку с сайта <https://code.jquery.com/>.

Ссылка выглядит примерно так:

<script src="https://code.jquery.com/jquery-3.4.1.min.js"

integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="

crossorigin="anonymous"></script>

Обратите внимание, что этот вариант требует интернет-подключения.

В обоих случаях библиотека подключается внутрь тега <head></head>

Примечание:

сompressed / production / minified version означает сжатую и оптимизированную версию файла, непригодную для чтения.

uncompressed, development version означает код в привычном виде, используемый для разработки.

## Основы JQuery

Стандартный синтаксис jQuery команд:

**$(селектор).метод();**

Знак $ сообщает, что символы идущие после него являются jQuery кодом.

Селектор позволяет выбрать элемент на странице.

Метод задает действие, которое необходимо совершить над выбранным элементом.

При вызове функции $() и передаче в неё селектора создается новый объект jQuery.

**Предотвращение преждевременного выполнения кода в jQuery**

Перед использованием jQuery для выполнения чего-либо на странице, необходимо убедиться, что страница находится в состоянии, когда она готова к изменению. Поэтому следует обернуть функции в $(document).ready((function(){}) или просто $(function (){}) .

$( function () {

//здесь будет ваш jQuery код

});

**Селекторы**

$(this); //выбрать текущий HTML элемент

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

$( 'p' ); // выбрать все параграфы

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

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

$('p.article' ); // выбрать все элементы p на странице с class="par".

Таким образом можно задать любые селекторы, которые были рассмотрены в разделе 3.

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

**Наборы jQuery**

Почти все методы jQuery возвращают набор, к которому этот метод был применен, что позволяет выстраивать цепочки методов любой длины.

Обратите внимание на переменную articles:

let articles = $('.article');

В нее запишется группа выбранных элементов, так называемый набор элементов jQuery.

Таким образом можно одновременно менять все элементы набора jQuery (в отличие от querySelectorAll никаких циклов не требуется).

**Получение отдельных элементов из выборки**

Для того, чтобы получить конкретный элемент из выборки, достаточно обратиться к нему через this:

$(".form-control").focus(function () {

$(this).css({background: "red"});

})

**Создание элементов**

С помощью функции $ также можно производить создание новых элементов. Если в $() передаётся фрагмент HTML, то это создаст новый элемент в памяти — иными словами, элемент будет создан, но не добавлен на страницу, пока вы этого не сделаете.

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

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

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

$( '<p>', {

html: 'Привет!',

'class': 'greet'

}); // создание нового элемента <p> с текстом и классом

Селекторам можно присвоить сразу несколько методов подобной записью:

$("div") // найдем все div-элементы

.css("color", "red") // установим красный цвет текста

.add(".article") // добавим в набор элементы с классом article

.fadeTo(0, 0.5); // установим прозрачность в 50% всем div'ам и элементам класса article

## Методы

### Фильтрация набора

**.filter()** – фильтрует набор выбранных элементов с помощью заданного селектора или функции.

$("div").filter(".article") //вернет div-элементы с классом article.

**.not()** – возвращает элементы, не соответствующие заданным условиям.

$("div").not(".article") //вернет div-элементы не имеющие класса article.

**.eq()** – возвращает элемент, идущий под заданным номером в наборе выбранных элементов.

$("div").eq(0) //вернет первый div-элемент на странице.

**.first()** – возвращает первый элемент в наборе. Метод не имеет параметров.

**.last()** – возвращает последний элемент в наборе.

**.has()** – фильтрует набор выбранных элементов, оставляя те, которые имеют потомков, соответствующих селектору.

$("div").has(".article") //вернет div-элементы, внутри которых есть элементы с классом article.

**.is()** – проверяет, содержится ли в наборе, хотя бы один элемент удовлетворяющий заданному селектору.

$(".article ").is("#cat") //проверит, имеется ли среди элементов с классом article, элементы с идентификатором cat.

**.slice()** – Возвращает элементы с индексами от start до end, если последний задан или до конца, если параметр end опущен. Элементы с индексом start включаются в результат, а end нет (т.е. .slice(3,5) вернет элементы, идущие под номером 3 и 4, элемент с номером 5 включен не будет). Кроме этого, параметры могут быть заданы в форме отрицательных чисел, в таком случае, отсчет элементов идет с конца набора: -1 – последний элемент, -2 – предпоследний элемент и т.д.

$("div").slice(3) вернет все div-элементы, начиная с четвертого (с индексами 3, 4, ...).

$("div").slice(3, 5) вернет div-элементы с индексами 3 и 4.

$("div").slice(-4, -2) вернет div-элементы, идущие четвертым и третьим с конца.

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

### Обход набора

**.each()** – вызывает заданную функцию для каждого элемента набора.

let heights = []; // переменная, которая будет хранить высоты элементов

$("div").each(function(index, element){

heights.push($(element).height());

}); // в переменную heights будут помещены значения высот всех div-элементов

**.map()** – вызывает заданную функцию для каждого элемента набора, и в итоге создает новый набор, составленный из значений, возвращенных этой функцией.

let classes = $("div").map(function(index, element){

return $(element).attr("class");

}); // переменной classes будет присвоен объект jQuery, содержащий список всех классов у div-элементов.

### Работа с CSS

**.css()** – возвращает/изменяет (в зависимости от числа входных параметров) CSS параметры элемента

*Пример*

$("body").css("color") //вернет цвет шрифта в body

$(".block ").css("top", "5px") //установит смещение по вертикали равным 5px у всех элементов с классом block

$(".block").css({"top":"5px", "left":"0"}) //установит смещения по вертикали и горизонтали равными 5px и 0px у всех элементов с классом block

**.height()** – возвращает/изменяет высоту элемента.

$(".content").height(30) //устанавливает значение высоты в 30 пикселей всем элементам с классом content.

**.width()** – возвращает/изменяет ширину элемента

**.position()** – возвращает/изменяет позицию элемента

### Работа с атрибутами и свойствами

**.attr()** – возвращает/изменяет (в зависимости от числа параметров) значение атрибута у элементов на странице

$("div").attr("class") вернет имя класса у первого div-элемента на странице

**.removeAttr()** – удаляет атрибут у элементов на странице

$("div").removeAttr("id") удалит идентификаторы у всех div-элементов на странице

**.prop()** – возвращает/изменяет (в зависимости от числа параметров) значение заданного свойства у элементов на странице

**.removeProp()** – удаляет заданное свойство у элементов на странице

**.addClass()** – добавляет класс элементам на странице

$("#changeTheme").addClass("red") //добавит класс red элементу с идентификатором changeTheme

**.removeClass()** – удаляет класс(ы) у элементов на странице

**.toggleClass()** – изменяет наличие класса у элементов на противоположное (добавляет/удаляет)

$("#content").toggleClass("clName1 clName2") "переключит" классы clName1 и clName2 у элемента с идентификатором content

**.hasClass(clName)** – проверяет наличие заданного класса хотя бы у одного из выбранных элементов

**.val()** – возвращает/изменяет (в зависимости от числа параметров) значение атрибута value у элементов на странице

$(".surname").val() вернет значение первого элемента с классом surname

### Добавление содержимого

**.html()** – возвращает/изменяет (в зависимости от числа параметров) html-содержимое элементов на странице

$(".article").html("<p>Новый текст</p>") изменит содержимое всех элементов с классом article на параграф с текстом "Новый текст!".

**.text()** – возвращает/изменяет (в зависимости от числа параметров) текст, находящийся в элементах на странице

$(".article").text() //вернет текстовое содержимое всех элементов с классом article.

**.append()** – добавляет заданное содержимое в конец элементов на странице

$(".list").append("<li class='item'>Тест</li>"); //элемент списка с текстом "Тест" будет добавлен в конец элементов с классом list.

**.prepend()** – добавляет заданное содержимое в начало элементов на странице.

**.after()** – добавляет заданное содержимое после элементов на странице

$(".list ").after("<li class='item'>Тест</li>"); //элемент списка с текстом "Тест" будет добавлен после элементов с классом list.

**.before()** – добавляет заданное содержимое перед элементами на странице

**.wrap()** – окружает элементы на странице заданными html-элементами.

$(".list").wrap("<div class='new'></div>");

### Удаление содержимого

**.remove()** – удаляет элементы на странице

$("div.content").remove() удалит все div-элементы с классом content.

**.empty()** – удаляет содержимое элементов на странице

$("div.content").empty() удалит содержимое всех div-элементов с классом content.

**.unwrap()** – удаляет родительские элементы, при этом их содержимое остается на месте

$("div.content").unwrap() удалит родительские элементы у всех div-элементов с классом content.

### Управление анимацией

**.animate(**properties, options**)** – выполняет анимацию, которая была создана пользователем.

properties — список CSS-свойств, участвующих в анимации и их конечных значений. Задается объектом, в формате css-свойство:значение.

options — дополнительные опции. Должны быть представлены объектом, в формате опция:значение.

Варианты опций:

duration — продолжительность выполнения анимации (см. выше).

easing — изменение скорости анимации (будет ли она замедляется к концу выполнения или наоборот ускориться). (см. описание ниже)

complete — функция, которая будет вызвана после завершения анимации.

step — функция, которая будет вызвана после каждого шага анимации.

queue — булево значение, указывающее, следует ли помещать текущую анимацию в очередь функций. В случае false, анимация будет запущена сразу же, не вставая в очередь.

specialEasing — позволяет установить разные значения easing, для разных CSS-параметров. Задается объектом, в формате параметр:значение.

$("#block").animate({

width: "70%", // ширина станет 70%

opacity: 0.4, // прозрачность будет 40%

marginLeft: "0.6in", // отступ от левого края элемента станет равным 6 дюймам

fontSize: "3em", // размер шрифта увеличится в 3 раза

borderWidth: "10px" // толщина рамки станет 10 пикселей

}, 1500); // анимация будет происходить 1,5 секунды

**.queue()** — предоставляет/изменяет (в зависимости от параметров) очередь функций.

**.clearQueue()** — очищает очередь функций.

.**dequeue()** — начинает выполнение следующей функции в очереди.

**.stop()** — останавливает выполнение текущей анимации.

**.delay()** — приостанавливает выполнение следующих анимаций на заданное время.

**.toggle()** — поочередно выполняет вызов одной из нескольких заданных функций.

**Стандартные анимации**

**.hide() .show()** – скрывает/показывает элементы на странице (за счет плавного изменения его размера и прозрачности).

$("#leftFit").hide(300) //в течении 300 милисекунд скроет элемент с идентификатором leftFit.

**.slideUp() .slideDown()** – разворачивает/сворачивает элементы на странице (за счет плавного изменения высоты элементов).

**.slideToggle()** – поочередно разворачивает/сворачивает элементы на странице, как это делают .slideUp() и .slideDown().

**.fadeIn() .fadeOut()** – скрывает/показывает элементы на странице за счет плавного изменения прозрачности.

.**fadeTo()** – плавно изменяет прозрачность элементов.

**.fadeToggle()** – поочередно скрывает/показывает элементы на странице, как это делают .fadeIn() и .fadeOut().

### События

Методы событий в jQuery пишутся как обычные DOM-события без приставок, например .click, .dbclick, .focus и т.д.