Основное назначение библиотеки jQuery

Данная библиотека позволяет изменять содержимое HTML- документов путем манипулирования объектами модели, создаваемой браузерами в процессе обработки HTML-кода (так называемые DOM-манипуляции).

Преимущества библиотеки jQuery:

1.Средства jQuery необычайно выразительны. Эта библиотека позволяет добиться гораздо большего при намного меньшем объеме кода, чем в случае использования программных DOM-интерфейсов браузеров.

2.Методы Query применимы к целым группам элементов. Предлагаемый в DOM- модели стандартный подход, основанный на шаблонной цепочке действий “выбрать-повторить-изменить”, больше не требуется. Следствием этого является уменьшение количества циклов for в коде, а значит, и снижение вероятности появления в нем ошибок.

3. Библиотека jQuery справляется с различиями в реализации DOM в различных браузерах (**проблемы кросс-браузерности**). Т.е. вы перестаете беспокоиться об особенностях поддержки того или иного средства, чем славится браузер Internet Explorer (IE).

Достаточно всего лишь сформулировать свои пожелания, и библиотека самостоятельно обеспечит совместимость с конкретным браузером.

4. Библиотека jQuery имеет **открытый исходный код**. Если принципы работы какого-либо средства для меня не совсем ясны или получаемый результат не совпадает с ожидаемым, можно обратиться непосредственно к коду библиотеки.

Установка библитеки jQuery

1. Скачать фреймворк с домашней странице проекта (http://jquery.com/) и подключить в коде

```
<head>
<script type="text/javascript" src="js/jquery-1.11.3.min.js"></script>
</head>
```

Данный способ хорош для работы в offline, или при медленном соединении синтернетом

2. С использованием CDN (*Content Delivery Network* или *Content Distribution Network*, CDN):

```
<head>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
</script>
</head>
```

Можно проверить, подгрузилась ли библиотека, и если нет, то подгрузить с собственного сайта. Для этого можно добавить:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
</script>
<script>window.jQuery ||

document.write('<script src="js/jquery-1.11.3.min.js"></script>')</script>
```

Доступ к возможностям

Доступ к возможностям библиотеки **jQuery** осуществляется с помощью функции библиотеки jQuery **$()**, которую мы будем называть просто функцией $(). Эта функция служит точкой входа в мир jQuery, а символ $ является удобным коротким псевдонимом пространства имен **jQuery**.

Т.е $() = jQuery ()

Вариант вызова

$(функция)	Позволяет указать функцию, которая должна быть выполнена по завершении построения DOM
$(селектор) $(селектор, контекст)	Осуществляет выбор группы элементов в документе с помощью селектора
$(HTMLElement) $(HTMLElement [])	Создает объект jQuery из объекта или массива объектов HTMLElement
$()	Создает пустой набор элементов
$(нтмл код), $(html, attributes)	Создает новые элементы из фрагмента HTML кода

Манипулирование объектами

Общее правило- манипулировать DOM объектами, можно только после того как объекты были загружены.

Т.е. если у вас есть фрагмент кода

<html>

<h1> 3D Printers shop </h1>

…..

</html>

То обратиться к заголовку h1 вы можете в конце документа или после строк, формирующих h1.

```
<html>
<head>
<script type="text/javascript" src="js/jquery-1.11.3.js"> </script>
</head>
<body>
<h1> 3D Printers shop </h1>
<script type="text/javascript">
$("h1").css("color","red");
 </script>
…….
</body>
</html>
```

```
<html>
<head>
<script type="text/javascript" src="js/jquery-1.11.3.js"> </script>
</head>
<body>
<h1> 3D Printers shop </h1>
……..
<script type="text/javascript">
$("h1").css("color","red");
 </script>
</body>
</html>
```

Отсрочка выполнения функции до момента готовности DOM

В предыдущем примере элемент script был помещен в конце документа, чтобы браузер успел создать все объекты в DOM до того, как начнет выполняться код JavaScript.

Библиотека jQuery предлагает способ решения этой проблемы.

Ожидание завершения построения DOM

```
<script type="text/javascript">
$(document).ready(function () {
// ...выполняемый код...
});
</script>
```

В этом сценарии мы передаем объект **document** функции **$()** в качестве аргумента и вызываем метод **ready(),** передавая ему функцию, которую хотим выполнить после окончания загрузки и готовности DOM к работе. Можете поместить этот элемент script в любое место документа, будучи уверенным в том, что jQuery не допустит преждевременного выполнения функции.

Добавьте код

```
<p class="descr"> Most Popular </p>
<p> Description</p>
<p id="pcl"> click me </p>
```

Использование альтернативной нотации

При желании можете передать свою функцию в качестве параметра непосредственно $-функции jQuery. При таком способе записи вызова результат будет тем же, что и в случае вызова $ (document) ready ().

```
<script type="text/javascript">
$(function()
    {$("#pcl").css("color", "blue");
    });
</script>
```

Выбор элементов В Javascript:

- getElementById(*id*) – поиск по id="…"
- getElementsByName(*name*) – поиск по name="name"
- getElementsByClassName(*class*) – поиск по class="class"
- getElementsByTagName(*tag*) – поиск по тегу
- querySelectorAll(*selector*) – поиск по произвольному CSS селектору

Для выбора элементов в jQuery используют селекторы

Селекторы в jQuery разделены на следующие категории:

- основные селекторы;
- иерархические селекторы;
- основные фильтры;
- фильтры для выбора дочерних элементов;
- фильтры для указания видимости элементов;
- фильтры по контенту;
- селекторы элементов форм;
- селекторы атрибутов.

Основные селекторы (Базовые селекторы)

1. Селектор *

Соответствует любому элементу.

Возвращает: Массив элементов.

Использование $("*")

var elementCount = $("*").length - возвращает число элементов

2. Селектор E

Выбирает все элементы с именем тега E.

Возвращает: Массив элементов.

<script>$("div").css("border", "9px solid red");</script>

3. Селектор .C

Выбирает все элементы с именем класса C.

Возвращает: Массив элементов.

<script>$(". descr ").css("border", "3px solid red");</script>

4. Селектор #id

Выбирает элемент с указанным идентификатором (атрибутом id).

<script>$("#myDiv").css("border","3px solid red");</script>

5. Множественный селектор ("selector1, selector2, selectorN")

<script>$("div, span, p.myClass").css("border", "3px solid red");</script>

Некоторые свойства и методы объекта jQuery

- **each(функция**) -Выполняет указанную функцию для каждого из выбранных элементов
- **get(индекс)-**Получает объект HTMLElement с указанным индексом
- **index(HTMLElement)-**Производит поиск указанного объекта HTMLElement среди набора выбранных элементов и возвращает его индекс,если находит его
- **index(jQuery)** -Аналогичен предыдущему методу, но возвращает индекс первого из элементов, содержащихся в указанном объекте jQuery
- **index(селектор**)-Возвращает индекс первого найденного элемента в объекте jQuery, вычисляемый относительно элементов соответствующих селектору
- **length** Возвращает число элементов в объекте jQuery

- **toArray()**Возвращает объекты HTMLElement, содержащиеся в объекте jQuery, в виде массива

Для примера, var fp=$("p").get(0) возвращает HTMLElement объект

Создание объектов jQuery из DOM-объектов

Объекты jQuery можно создавать, передавая объект или массив объектов HTMLElement функции $() в качестве аргумента.

```
$(document).ready(function() {
var elems = document.getElementsByTagName("p");
console.log($(elems));
$(elems).mouseenter(function(e) {
$(this).css("border","1px red solid" );
})
.mouseout(function(e) {
$(this).css("border", "none");
})

}
)
```

Итерирование функции по DOM-объектам

Метод each() позволяет определить функцию, которая будет выполнена для каждого из DOM-объектов, содержащихся в объекте jQuery

```
$(document).ready(function() {
        $('ul > li').each( function(index){
                        $(this).append('<span>'+index+'</span>');
                        })
});
```

- **2. Иерархические селекторы**
- Дочерний селектор (“parent > child”)

  Выбирает всех непосредственных потомков (детей) заданного родителя
- **Селектор потомков** (“ancestor descendant”)

  Выбирает все элементы, которые являются потомками данного родителя
- **Селектор (“prev + next”)**

  Выбирает все элементы next, которым непосредственно предшествует элементы prev
- **Селектор (“prev ~ siblings”)**

  Выбирает все братские элементы, следующие после заданного элемента prev на том же уровне вложенности

3. **Селекторы атрибутов**

Данная группа селекторов позволяет выбрать элементы по их атрибутам

- Селектор [name=”value”]

Выбирает все элементы с атрибутом с именем name, значение которого точно совпадает с заданным значение value

```
<!doctype html>
<head>
<script type="text/javascript" src="js/jquery-1.11.3.js"> </script>
<script>
$(function(){
 var inp=$("input[name='t1']");
 inp.on("click",function(event) {
 alert($(this).val());
 alert(event.target.value);}
 ); }
 )
</script>
</head>
<body><form>
<input type="text" name="t1" value="aaa"  />1 <br />
<input type="radio" name="numbers" value=1 checked />1 <br />
<input type="radio" name="numbers" value=2 />2 <br />
<input type="radio" name="numbers" value=3 />3 <br />
</form></body></html>
```

Селектор [name|=”value”]

Выбирает все элементы с заданным атрибутом, значение которого либо совпадает со значение value, либо со строкой, в которой начальная часть строки value, а затем следует дефис

```
<body>

<a href="example.html" hreflang="en">Some text</a>
<a href="example.html" hreflang="en-UK">Some other text</a>
<a href="example.html" hreflang="english">will not be outlined</a>

<script>
$( "a[hreflang|='en']" ).css( "border", "3px dotted green" );
</script>

</body>
```

- **Селектор [name^=”value”]**

Выбирает все элементы с заданным атрибутом, значение которого начинается с подстроки value.

- **Селектор [name*=”value”]**

Выбирает все элементы с заданным атрибутом, значение которого содержит подстроку value.

- **Селектор [name~=”value”]**

Выбирает все элементы с заданным атрибутом, значение которого содержит заданное слово, разделенное пробелами.

- **Селектор** [name$=”value”]

Выбирает все элементы с заданным атрибутом, значение которого заканчивается заданным словом.

- **Селектор** [name!=”value”]

Выбирает все элементы, которые либо не имеют заданного атрибута, либо значение его не совпадает с указанным значением.

- **Множественный [name=”value”][name2=”value2″]**

Множественный селектор

```
<input id="man-news" name="man-news">
<input name="milkman">
<input id="letterman"
  name="new-letterman">
<input name="newmilk">


<script>
$( "input[id][name$='man']" ).val( "only this
  one" );
</script>
```