

DOM-манипуляции

Создание новых элементов

Создание элементов с использованием функции \$()

Одним из способов создания элементов является передача строки, содержащей HTML-фрагмент, функции \$ (), которая выполнит синтаксический анализ строки и создаст соответствующие DOM-объекты.

```
var newElem=$('<li class="dcell"></div>');
```

```
var tr1= $('<tr><td>aaaaaaaa</td><td>bbbbbbbb</td></tr>'); создание новой строки
```

Создание новых элементов путем клонирования существующих

Метод clone () позволяет создавать новые элементы на основе существующих. Вызов этого метода приводит к **клонированию каждого из элементов**, содержащихся в объекте jQuery, вместе со всеми их элементами-потомками.

```
var cloneCell = $('li.dcell').clone();
```

```
cloneCell.each(function (index, elem) {  
  console.log ("Новый элемент: " + elem.tagName + " " +  
    elem.className);  
});  
cloneCell.children('img').each(function(index, elem) {  
  console.log("Дочерний элемент: " + elem.tagName +  
" " + elem.src);  
});  
//$('#ul#forclone').append(cloneCell);
```

В этом сценарии **выбираются и клонируются все элементы li принадлежащие классу dcell**. Для демонстрации того, что при этом клонируются также элементы-потомки, в сценарии вызывается метод children() с селектором, обеспечивающим получение клонированных элементов img. Информация об элементах div и img выводится на консоль.

Вставка дочерних элементов и элементов-потомков

Методы для вставки дочерних элементов и элементов-потомков

append(HTML) append(jQuery) , append(HTMLElement[]) -
Вставляет указанные элементы в качестве последних дочерних элементов во все выбранные элементы

prepend(HTML) prepend(jQuery) prepend(HTMLElement[]) –
Вставляет указанные элементы в качестве первых дочерних элементов во все выбранные элементы

appendTo(jQuery) appendTo(HTMLElement[]) - Вставляет элементы, содержащиеся в объекте jQuery, в качестве последних дочерних элементов в элементы, заданные аргументом.

prependTo(HTML) prependTo(jQuery),prependTo(HTMLElement[])
Вставляет элементы, содержащиеся в объекте jQuery, в качестве первых дочерних элементов в элементы, заданные аргументом

Пример Our users:

```
<ul class="myUsers">
```

```
<li>Ivanov Ivan</li>
```

```
<li>Sokolov Dmitrii</li>
```

```
<li>Kolesnokov Semen</li>
```

```
</ul>
```

```
var newUser1=$('<li>Constantinov Vladimir</li>');
```

```
//создаст элемент
```

```
$('ul.myUsers').append(newUser1);
```

```
// добавит в конец списка
```

```
$('ul.myUsers').prepend('<li>Staameskin  
Konstantin</li>');
```

```
//добавит в начало списка.
```

В случае использования методов **appendTo ()** и **prependTo ()** все меняется местами: элементы, содержащиеся в объекте jQuery, вставляются в качестве дочерних элементов внутрь элементов, заданных аргументом

```
var span=$('<span>!! </span>');
span.prependTo($('ul.myUsers li'));
$('<input type="checkbox" name="checkUser"
 />').prependTo($('ul.myUsers li>span'));
$('ul.myUsers li>span').each(function(index,elem){
var newch=$('<input type="checkbox" name="checkUser"
 value='+index+' />');
newch.prependTo($(elem));
});
```

```
var row3= $('<ul class="drow" id="row3"> </ul>');
var cell=$('<li class="dcell"> </li>')
.append('')
.append('<div class="bot_part"><label
  for="pr_7">3D принтер Leapfrog Creatr
  2</label> <input type="text" name="pr_7"
  value="0" required /></div>');
cell.appendTo(row3);
row3.appendTo('#printers');
```

append(функция) prepend(функция)

Добавляет результат, возвращаемый функцией, в окончание или начало содержимого каждого из элементов, содержащихся в объекте jQuery

Элементы, которые передаются перечисленным выше методам в виде аргументов, вставляются в качестве дочерних элементов в каждый выбранный элемент, содержащийся в объекте jQuery. В связи с этим очень важно правильно формировать наборы элементов.

```
$('#ul.myUsers li>span').append( function(index){  
  var newch=$('#<input type="checkbox" name="checkUser"  
    value='+index+' />');  
  return newch;  
})
```

Вставка родительских элементов и элементов-предков

Библиотека jQuery предоставляет набор методов, обеспечивающих вставку элементов в документ как родительских или элементов-предков по отношению к существующим элементам. Такого рода операции называются обертыванием (wrapping), или внешней вставкой, (поскольку добавляемый элемент окружает собой уже существующие элементы).

wrap(HTML),wrap(jQuery),wrap(HTMLElement[]) -Обертывает указанные элементы вокруг каждого из элементов, содержащихся в объекте jQuery

wrapAll(HTML),wrapAll(jQuery),wrapAll(HTMLElement[]) Обертывает указанные элементы вокруг набора элементов, содержащихся в объекте jQuery (рассматриваемых как единая группа)

wrapInner(HTML),wrapInner(jQuery),wrapInner(HTMLElement[]) Обертывает указанные элементы вокруг содержимого каждого из элементов, содержащихся в объекте jQuery

wrap(функция),wrapInner(функция) Динамически обертывает элементы с использованием функции

При обертывании элементов эти методы могут принимать HTML-фрагмент в качестве аргумента, но всегда следует проверять, чтобы этот фрагмент содержал только один внутренний элемент. В противном случае jQuery не сможет определить, что именно необходимо сделать. Отсюда следует, что каждый элемент в аргументе метода может иметь не более одного родительского и не более одного дочернего элемента.

Пример использования метода wrap ()

```
<script type="text/javascript">
$(document).ready(function() {
var newElem = $("<div />").css("border", "thick solid red");
$('li.dcell> img').wrap(newElem);
});
</script>
```

В этом сценарии мы создаем новый элемент div и используем метод css () для установки значения CSS-свойства border. Затем элемент div добавляется в качестве родительского элемента по отношению ко всем элементам img.

Элементы, которые вы передаете в виде аргумента методу `wrap()`, вставляются между каждым из элементов, содержащихся в объекте `jQuery`, и его текущим родительским элементом.

Для того, чтобы вставить в документ элемент, который станет родительским сразу для нескольких элементов, следует использовать метод **`wrapAll ()`**

```
<script type="text/javascript">
$(document).ready(function() {
var newElem = $("<div />")
.css("border", "thick solid green");
$ ('ul.dtable,ul.myUsers') .wrapAll (newElem) ;
});
</script>
```

Этот сценарий отличается от предыдущего лишь тем, что в нем используется метод `wrapAll ()`.

Новый элемент добавляется как **общий родительский элемент** для всего набора выбранных элементов.

Будьте внимательны, используя этот метод. Если выбранные элементы не имеют общих родителей, то новый элемент будет добавлен как родительский для первого из них. После этого `jQuery` переместит все остальные элементы так, чтобы они стали сестринскими по отношению к первому элементу.

Обертывание содержимого элементов

Метод `wrapInner ()`

позволяет окружать элементами содержимое других элементов (а не сами элементы). Соответствующий пример приведен в листинге

```
<script type="text/javascript">
$(document).ready(function() {
var newElem = $("<div />").css("border", "thick solid red");
$('li.dcell').wrapInner(newElem);
});
</script>
```

Метод `wrapInner ()` вставляет новые элементы между каждым из содержащихся в объекте jQuery элементом и его дочерним элементом. В данном сценарии выбираются элементы, принадлежащие классу `dcell`, и содержимое каждого из них заключается в новый элемент `div`.

Обертывание элементов с использованием функции

Методам `wrap ()` и `wrapAll()` в качестве аргумента может быть передана функция, что обеспечивает возможность динамической генерации элементов. Единственным аргументом этой функции является индекс элемента в выбранном наборе. Внутри этой функции специальная переменная `this` ссылается на обрабатываемый элемент.

Вставка сестринских элементов

jQuery предоставляет также набор методов, обеспечивающих вставку элементов в документ как сестринских по отношению к существующим элементам.

after(HTML),after(jQuery),after(HTMLElement[])

Вставляет указанные элементы в качестве последних сестринских элементов во все выбранные элементы

before(HTML) before(jQuery) before(HTMLElement[])

Вставляет указанные элементы в качестве первых сестринских элементов во все выбранные элементы

insertAfter(HTML) insertAfter(jQuery) insertAfter(HTMLElement[])

Вставляет элементы, содержащиеся в объекте jQuery, в качестве последних сестринских элементов в элементы, заданные аргументом

insertBefore(HTML) insertBefore(jQuery) insertBefore(HTMLElement[])

Вставляет элементы, содержащиеся в объекте jQuery, в качестве первых сестринских элементов в элементы, заданные аргументом

after(функция) before(функция)

Добавляет результат, возвращаемый функцией, в окончание или начало содержимого

```
var newCell=$('<li class="dcell"> </li>')
.append('')
.append('<div class="bot_part"><label
  for="pr_7">3D принтер Leapfrog Creatr
  2</label> <input type="text" name="pr_7"
  value="0" required /></div>');
$('#row1 li.dcell#dc11').after(newCell);
или $('#row1 li.dcell#dc11').before(newCell);
```

Вставка сестринских элементов из объекта jQuery

Методы `insertAfter()` и `insertBefore()` вставляют элементы, содержащиеся в объекте jQuery, в качестве следующих или предшествующих сестринских элементов по отношению к элементам, переданным методу в качестве аргумента. Это та же функциональность, которую обеспечивают методы `after()` и `before()`, но отличающаяся тем, что объект jQuery и аргумент метода меняются ролями

```
newCell.insertAfter('#row1 #dc11');
```

Вставка сестринских элементов с использованием функции

Сестринские элементы можно вставлять динамически, передавая методам `after()` и `before()` функцию в качестве аргумента, как мы это уже делали при вставке родительских и дочерних элементов.

Замена элементов

С помощью методов, описанных далее можно заменить один набор элементов другим.

Методы замены элементов

**replaceWith(HTML), replaceWith (jQuery),
replaceWith(HTMLElement [])**

Заменяет элементы, содержащиеся в объекте jQuery, указанным содержимым

replaceAll (jQuery),replaceAll (HTMLElement [])

Заменяет элементы, заданные аргументом, элементами, содержащимися в объекте jQuery

replaceWith (функция)

Выполняет динамическую замену элементов, содержащихся в объекте jQuery, с использованием функции

Методы **replaceWith()** и **replaceAll ()** работают одинаковым образом, за исключением того, что объект jQuery и аргумент играют в них противоположные роли.

Пример использования обоих методов приведен в листинге

```
var newCell=$( '<li class="dcell"> </li>' )
.append( '' )
.append( '<div class="bot_part"> <label for="pr_7">3D сканер
Sense</label> <input type="text" name="sc_1" value="0" required
/></div>' );
$('#row4').children().first().replaceWith(newCell);
$(" <img src='img/3D_printer_CubeX.jpg' /> ").replaceAll('#row2 img')
.css("border", "thick solid red");
```

В этом сценарии сначала создается набор элементов, а затем в документе выполняется поиск элемента ul атрибут id которого равен row4, и его первый дочерний элемент заменяется новым содержимым с помощью метода replaceWith ().

Наконец, с помощью метода replaseAll () все элементы img, являющиеся потомками элемента, атрибут id которого равен row2, заменяются другими изображениями.

Удаление элементов

В дополнение к возможности вставки и замены элементов библиотека jQuery предлагает ряд методов для удаления элементов из DOM

detach(), detach(селектор)

Удаляет элементы из DOM. Данные, связанные с элементами, сохраняются

empty() Удаляет все дочерние узлы каждого из элементов, содержащихся в объекте jQuery

remove(),remove(селектор)

Удаляет элементы из DOM. По мере удаления элементов связанные с ними данные уничтожаются

unwrap() Удаляет родительские элементы каждого из элементов, содержащихся в объекте jQuery

```
<script type="text/javascript">
$(document).ready(function() {
$('img[src*=Cube_3], img[src*=Duplicator_4x]').parent().remove();
});
</script>
```

В этом сценарии мы выбираем элементы `img`, атрибуты `src` которых содержат `Cube_3` и `Duplicator`, получаем их родительские элементы, а затем удаляем их. Можно также отфильтровать удаляемые элементы, передав селектор методу `remove()`, как показано в листинге

Фильтрация удаляемых элементов с помощью селектора

```
<script type="text/javascript">
$(document).ready(function() {
$('li.dcell').remove(':has(img[src*=Cube_3], img[src*=Duplicator_4x])');
});
</script>
```

Оба сценария приводят к одному и тому же результату

Удаление элементов с сохранением данных

Метод `detach ()` работает аналогично методу `remove ()` с тем лишь отличием, что связанные с удаляемыми элементами данные сохраняются.

Если планируется последующая вставка удаленных элементов в другое место документа, то обычно предпочтение следует отдавать методу `detach ()`. Пример использования метода `detach ()` приведен в листинге

```
<script type="text/javascript">
$(document).ready(function() {
$('#row2').append($('img[src*=Duplicator_4x]').parent().detach());
});
</script>
```

В этом сценарии удаляется родительский элемент элемента `img`, атрибут `src` которого содержит `Duplicator_4x`. Затем элементы вновь вставляются в документ с помощью рассмотренного нами ранее метода `append ()`.

Лучше избегать такого подхода, поскольку использование метода `append ()` без вызова метода `detach ()` дает тот же эффект.

Инструкцию, являющуюся ключевой в листинге, можно переписать следующим образом:

```
$('#row2').append($('img[src*=Duplicator_4x]').parent());
```

Очистка элементов

Метод `empty()` удаляет все элементы-потомки и текст из элементов, содержащихся в объекте `jQuery`. Сами элементы остаются в документе.

Использование метода `empty()`

```
<script type="text/javascript">
$(document).ready(function() {
$('#row1').children().eq(1).empty().css("border", "thick solid red");
});
</script>
```

В этом сценарии из дочерних элементов элемента с атрибутом `id`, равным `row1`, выбирается элемент с индексом, равным `1`, и вызывается метод `empty()`. Чтобы сделать изменение более заметным, соответствующая позиция в документе заключена в красную рамку

Метод `unwrap()`

Метод `unwrap()` удаляет родительские элементы для элементов, содержащихся в объекте `jQuery`. Выбранные элементы становятся дочерними элементами родителей своих бывших родительских элементов. Пример использования метода `unwrap()`

```
<script type="text/javascript">
$(document).ready(function() {
$('div.bot_part > *').unwrap().css({'display':'block'});
});
</script>
```

В этом сценарии выбираются элементы являющиеся потомками элемента `div` класса `bot_part`, и вызывается метод `unwrap()`