**Добавление и удаление узлов**

Изменение DOM – ключ к созданию «живых» страниц.

В этой главе мы рассмотрим, как создавать новые элементы «на лету» и заполнять их данными.

**[Пример: показ сообщения](http://learn.javascript.ru/modifying-document" \l "primer-pokaz-soobscheniya)**

В качестве примера рассмотрим добавление сообщения на страницу, чтобы оно было оформленно красивее чем обычный alert.

HTML-код для сообщения:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<div class="alert">

<strong>Ура!</strong> Вы прочитали это важное сообщение.

</div>

**[Создание элемента](http://learn.javascript.ru/modifying-document" \l "sozdanie-elementa)**

Для создания элементов используются следующие методы:

**document.createElement(tag)**

Создает новый элемент с указанным тегом:

var div = document.createElement('div');

document.createTextNode(text)

Создает новый \*текстовый\* узел с данным текстом:

var textElem = document.createTextNode('Тут был я');

**[Создание сообщения](http://learn.javascript.ru/modifying-document" \l "sozdanie-soobscheniya)**

В нашем случае мы хотим сделать DOM-элемент div, дать ему классы и заполнить текстом:

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

После этого кода у нас есть готовый DOM-элемент. Пока что он присвоен в переменную div, но не виден, так как никак не связан со страницей.

**[Добавление элемента: appendChild, insertBefore](http://learn.javascript.ru/modifying-document" \l "dobavlenie-elementa-appendchild-insertbefore)**

Чтобы DOM-узел был показан на странице, его необходимо вставить в document.

Для этого первым делом нужно решить, куда мы будем его вставлять. Предположим, что мы решили, что вставлять будем в некий элемент parentElem, например var parentElem = document.body.

Для вставки внутрь parentElem есть следующие методы:

**parentElem.appendChild(elem)**

Добавляет elem в конец дочерних элементов parentElem.

Следующий пример добавляет новый элемент в конец <ol>:

<ol id="list">

<li>0</li>

<li>1</li>

<li>2</li>

</ol>

<script>

var newLi = document.createElement('li');

newLi.innerHTML = 'Привет, мир!';

list.appendChild(newLi);

</script>

**parentElem.insertBefore(elem, nextSibling)**

Вставляет elem в коллекцию детей parentElem, перед элементом nextSibling.

Следующий код вставляет новый элемент перед вторым <li>:

<ol id="list">

<li>0</li>

<li>1</li>

<li>2</li>

</ol>

<script>

var newLi = document.createElement('li');

newLi.innerHTML = 'Привет, мир!';

list.insertBefore(newLi, list.children[1]);

</script>

Для вставки элемента в начало достаточно указать, что вставлять будем перед первым потомком:

list.insertBefore(newLi, list.firstChild);

У читателя, который посмотрит на этот код внимательно, наверняка возникнет вопрос: «А что, если list вообще пустой, в этом случае ведь list.firstChild = null, произойдёт ли вставка?»

Ответ – да, произойдёт.

**Дело в том, что если вторым аргументом указать null, то insertBefore сработает как appendChild:**

parentElem.insertBefore(elem, null);

// то же, что и:

parentElem.appendChild(elem)

Так что insertBefore универсален.

**На заметку:**

Все методы вставки возвращают вставленный узел.

Например, parentElem.appendChild(elem) возвращает elem.

**[Пример использования](http://learn.javascript.ru/modifying-document" \l "primer-ispolzovaniya)**

Добавим сообщение в конец <body>:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.appendChild(div);

</script>

…А теперь – в начало <body>:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.insertBefore(div, document.body.firstChild);

</script>

**[Клонирование узлов: cloneNode](http://learn.javascript.ru/modifying-document" \l "klonirovanie-uzlov-clonenode)**

А как бы вставить второе похожее сообщение?

Конечно, можно сделать функцию для генерации сообщений и поместить туда этот код, но в ряде случаев гораздо эффективнее – *клонировать* существующий div, а потом изменить текст внутри. В частности, если элемент большой, то клонировать его будет гораздо быстрее, чем пересоздавать.

Вызов elem.cloneNode(true) создаст «глубокую» копию элемента – вместе с атрибутами, включая подэлементы. Если же вызвать с аргументом false, то копия будет сделана без дочерних элементов. Это нужно гораздо реже.

Пример со вставкой копии сообщения:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.insertBefore(div, document.body.firstChild);

// создать копию узла

var div2 = div.cloneNode(true);

// копию можно подправить

div2.querySelector('strong').innerHTML = 'Супер!';

// вставим её после текущего сообщения

div.parentNode.insertBefore(div2, div.nextSibling);

</script>

Обратите внимание на последнюю строку, которая вставляет div2 после div:

div.parentNode.insertBefore(div2, div.nextSibling);

1. Для вставки нам нужен будущий родитель. Мы, возможно, не знаем, где точно находится div (или не хотим зависеть от того, где он), но если нужно вставить рядом с div, то родителем определённо будет div.parentNode.
2. Мы хотели бы вставить *после* div, но метода insertAfter нет, есть только insertBefore, поэтому вставляем *перед* его правым соседом div.nextSibling.

**[Удаление узлов: removeChild](http://learn.javascript.ru/modifying-document" \l "udalenie-uzlov-removechild)**

Для удаления узла есть два метода:

**parentElem.removeChild(elem)**

Удаляет elem из списка детей parentElem.

**parentElem.replaceChild(newElem, elem)**

Среди детей parentElem удаляет elem и вставляет на его место newElem.

Оба этих метода возвращают удаленный узел, то есть elem. Если нужно, его можно вставить в другое место DOM тут же или в будущем.

**На заметку:**

Если вы хотите *переместить* элемент на новое место – не нужно его удалять со старого.

**Все методы вставки автоматически удаляют вставляемый элемент со старого места.**

Конечно же, это очень удобно.

Например, поменяем элементы местами:

<div>Первый</div>

<div>Второй</div>

<script>

var first = document.body.children[0];

var last = document.body.children[1];

// нет необходимости в предварительном removeChild(last)

document.body.insertBefore(last, first); // поменять местами

</script>

**Метод remove**

В современном стандарте есть также метод [elem.remove()](https://dom.spec.whatwg.org/" \l "dom-childnode-remove), который удаляет элемент напрямую, не требуя ссылки на родителя. Это зачастую удобнее, чем removeChild.

Он поддерживается во всех современных браузерах, кроме IE11-. Впрочем, легко подключить или даже сделать полифилл.

**[Удаление сообщения](http://learn.javascript.ru/modifying-document" \l "udalenie-soobscheniya)**

Сделаем так, что через секунду сообщение пропадёт:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<body>

<h3>Сообщение пропадёт через секунду</h3>

</body>

<script>

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.appendChild(div);

setTimeout(function() {

div.parentNode.removeChild(div);

}, 1000);

</script>

**[Текстовые узлы для вставки текста](http://learn.javascript.ru/modifying-document" \l "tekstovye-uzly-dlya-vstavki-teksta)**

При работе с сообщением мы использовали только узлы-элементы и innerHTML.

Но и текстовые узлы тоже имеют интересную область применения!

Если текст для сообщения нужно показать именно как текст, а не как HTML, то можно обернуть его в текстовый узел.

Например:

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6;

border-radius: 4px;

color: #3c763d;

background-color: #dff0d8;

}

</style>

<script>

var div = document.createElement('div');

div.className = "alert alert-success";

document.body.appendChild(div);

var text = prompt("Введите текст для сообщения", "Жили были <a> и <b>!");

// вставится именно как текст, без HTML-обработки

div.appendChild(document.createTextNode(text));

</script>

В современных браузерах (кроме IE8-) в качестве альтернативы можно использовать присвоение textContent.

**[Итого](http://learn.javascript.ru/modifying-document" \l "itogo)**

Методы для создания узлов:

* document.createElement(tag) – создает элемент
* document.createTextNode(value) – создает текстовый узел
* elem.cloneNode(deep) – клонирует элемент, если deep == true, то со всеми потомками, если false – без потомков.

Вставка и удаление узлов:

* parent.appendChild(elem)
* parent.insertBefore(elem, nextSibling)
* parent.removeChild(elem)
* parent.replaceChild(newElem, elem)

Все эти методы возвращают elem.

Методы для изменения DOM также описаны в спецификации [DOM Level 1](http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html).