**Мультивставка: insertAdjacentHTML и DocumentFragment**

Обычные методы вставки работают с одним узлом. Но есть и способы вставлять множество узлов одновременно.

**[Оптимизация вставки в документ](http://learn.javascript.ru/multi-insert" \l "optimizatsiya-vstavki-v-dokument)**

Рассмотрим задачу: сгенерировать список UL/LI.

Есть две возможных последовательности:

1. Сначала вставить UL в документ, а потом добавить к нему LI:

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

document.body.appendChild(ul); // сначала в документ

for (...) ul.appendChild(li); // потом узлы

1. Полностью создать список «вне DOM», а потом – вставить в документ:

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

for(...) ul.appendChild(li); // сначала вставить узлы

document.body.appendChild(ul); // затем в документ

Как ни странно, между этими последовательностями есть разница. В большинстве браузеров, второй вариант – быстрее.

Почему же? Иногда говорят: «потому что браузер перерисовывает каждый раз при добавлении элемента». Это не так. Дело вовсе не в перерисовке.

Браузер достаточно «умён», чтобы ничего не перерисовывать понапрасну. В большинстве случаев процессы перерисовки и сопутствующие вычисления будут отложены до окончания работы скрипта, и на тот момент уже совершенно без разницы, в какой последовательности были изменены узлы.

**Тем не менее, при вставке узла происходят разные внутренние события и обновления внутренних структур данных, скрытые от наших глаз.**

Что именно происходит – зависит от конкретной, внутренней браузерной реализации DOM, но это отнимает время. Конечно, браузеры развиваются и стараются свести лишние действия к минимуму.

**[Бенчмарк](http://learn.javascript.ru/multi-insert" \l "insert-bench-tbody)**

Чтобы легко проверить текущее состояние дел – вот два бенчмарка.

Оба они создают таблицу 20x20, наполняя TBODY элементами TR/TD.

При этом первый вставляет все в документ тут же, второй – задерживает вставку TBODY в документ до конца процесса.

Кликните, чтобы запустить.

|  |
| --- |
|  |

Код для тестов находится в файле [insert-bench.js](http://learn.javascript.ru/article/multi-insert/insert-bench.js).

**[Добавление множества узлов](http://learn.javascript.ru/multi-insert" \l "dobavlenie-mnozhestva-uzlov)**

Продолжим работать со вставкой узлов.

Рассмотрим случай, когда в документе *уже есть* большой список UL. И тут понадобилось срочно добавить еще 20 элементов LI.

Как это сделать?

Если новые элементы пришли в виде строки, то можно попробовать добавить их так:

ul.innerHTML += "<li>1</li><li>2</li>...";

Но операцию ul.innerHTML += "..." можно по-другому переписать как ul.innerHTML = ul.innerHTML + "...". Иначе говоря, она *не прибавляет, а заменяет* всё содержимое списка на дополненную строку. Это и нехорошо с точки зрения производительности, но и будут побочные эффекты. В частности, все внешние ресурсы (картинки) внутри перезаписываемого innerHTML будут загружены заново. Если в каких-то переменных были ссылки на элементы списка – они станут неверны, так как содержимое полностью заменяется. В общем, так лучше не делать.

А если нужно вставить в середину списка? Здесь innerHTML вообще не поможет.

Можно, конечно, вставить строку во временный DOM-элемент и перенести оттуда элементы, но есть и гораздо лучший вариант: метод insertAdjacentHTML!

**[insertAdjacent\*](http://learn.javascript.ru/multi-insert" \l "insertadjacent)**

Метод [insertAdjacentHTML](https://developer.mozilla.org/en/DOM/element.insertAdjacentHTML) позволяет вставлять произвольный HTML в любое место документа, в том числе *и между узлами*!

Синтаксис:

elem.insertAdjacentHTML(where, html);

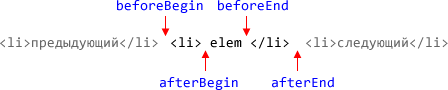
**html**

Строка HTML, которую нужно вставить

**where**

Куда по отношению к elem вставлять строку. Всего четыре варианта:

1. beforeBegin – перед elem.
2. afterBegin – внутрь elem, в самое начало.
3. beforeEnd – внутрь elem, в конец.
4. afterEnd – после elem.



Например, вставим пропущенные элементы списка *перед* <li>5</li>:

<ul>

<li>1</li>

<li>2</li>

<li>5</li>

</ul>

<script>

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

var li5 = ul.children[2];

li5.insertAdjacentHTML("beforeBegin", "<li>3</li><li>4</li>");

</script>

У этого метода есть «близнецы-братья»:

* [elem.insertAdjacentElement(where, newElem)](http://help.dottoro.com/ljbreokf.php) – вставляет в произвольное место не строку HTML, а элемент newElem.
* [elem.insertAdjacentText(where, text)](http://help.dottoro.com/ljrsluxu.php) – создаёт текстовый узел из строки text и вставляет его в указанное место относительно elem.

Синтаксис этих методов, за исключением последнего параметра, полностью совпадает с insertAdjacentHTML. Вместе они образуют «универсальный швейцарский нож» для вставки чего угодно куда угодно.

**[DocumentFragment](http://learn.javascript.ru/multi-insert" \l "documentfragment)**

**Важно для старых браузеров**

Оптимизация, о которой здесь идёт речь, важна в первую очередь для старых браузеров, включая IE9-. В современных браузерах эффект от нее, как правило, небольшой, а иногда может быть и отрицательным.

До этого мы говорили о вставке строки в DOM. А что делать в случае, когда надо в существующий UL вставить много *DOM-элементов*?

Можно вставлять их один за другим, вызовом insertBefore/appendChild, но при этом получится много операций с большим живым документом.

**Вставить пачку узлов единовременно поможет DocumentFragment. Это особенный *кросс-браузерный* DOM-объект, который похож на обычный DOM-узел, но им не является.**

Синтаксис для его создания:

var fragment = document.createDocumentFragment();

В него можно добавлять другие узлы.

fragment.appendChild(node);

Его можно клонировать:

fragment.cloneNode(true); // клонирование с подэлементами

**У DocumentFragment нет обычных свойств DOM-узлов, таких как innerHTML, tagName и т.п. Это не узел.**

Его «Фишка» заключается в том, что когда DocumentFragment вставляется в DOM – то он исчезает, а вместо него вставляются его дети. Это свойство является уникальной особенностью DocumentFragment.

Например, если добавить в него много LI, и потом вызвать ul.appendChild(fragment), то фрагмент растворится, и в DOM вставятся именно LI, причём в том же порядке, в котором были во фрагменте.

Псевдокод:

// хотим вставить в список UL много LI

// делаем вспомогательный DocumentFragment

var fragment = document.createDocumentFragment();

for (цикл по li) {

fragment.appendChild(list[i]); // вставить каждый LI в DocumentFragment

}

ul.appendChild(fragment); // вместо фрагмента вставятся элементы списка

В современных браузерах эффект от такой оптимизации может быть различным, а на небольших документах иногда и отрицательным.

Понять текущее положение вещей вы можете, запустив следующий [небольшой бенчмарк](https://plnkr.co/edit/86cG7KH7sWLoq7lNoQcU?p=preview).

**[append/prepend, before/after, replaceWith](http://learn.javascript.ru/multi-insert" \l "append-prepend-before-after-replacewith)**

Сравнительно недавно в [стандарте](https://dom.spec.whatwg.org/) появились методы, которые позволяют вставить что угодно и куда угодно.

Синтаксис:

* node.append(...nodes) – вставляет nodes в конец node,
* node.prepend(...nodes) – вставляет nodes в начало node,
* node.after(...nodes) – вставляет nodes после узла node,
* node.before(...nodes) – вставляет nodes перед узлом node,
* node.replaceWith(...nodes) – вставляет nodes вместо node.

Эти методы ничего не возвращают.

Во всех этих методах nodes – DOM-узлы или строки, в любом сочетании и количестве. Причём строки вставляются именно как текстовые узлы, в отличие от insertAdjacentHTML.

Пример (с полифиллом):

<html>

<head>

<meta charset="utf-8">

<script src="https://cdn.polyfill.io/v1/polyfill.js?features=Element.prototype.append,Element.prototype.after"></script>

</head>

<body>

<script>

// добавим элемент в конец <body>

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

document.body.append(p);

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

em.append('Мир!');

// вставить в параграф текстовый и обычный узлы

p.append("Привет, ", em);

// добавить элемент после <p>

p.after(document.createElement('hr'))

</script>

</body>

</html>

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

* Манипуляции, меняющие структуру DOM (вставка, удаление элементов), как правило, быстрее с отдельным маленьким узлом, чем с большим DOM, который находится в документе.

Конкретная разница зависит от внутренней реализации DOM в браузере.

* Семейство методов для вставки HTML/элемента/текста в произвольное место документа:
  + elem.insertAdjacentHTML(where, html)
  + elem.insertAdjacentElement(where, element)
  + elem.insertAdjacentText(where, text)
* DocumentFragment позволяет минимизировать количество вставок в большой живой DOM. Эта оптимизация особо эффективна в старых браузерах, в новых эффект от неё меньше или наоборот отрицательный.

Элементы сначала вставляются в него, а потом – он вставляется в DOM. При вставке DocumentFragment«растворяется», и вместо него вставляются содержащиеся в нём узлы.

DocumentFragment, в отличие от insertAdjacent\*, работает с коллекцией DOM-узлов.

* Современные методы, работают с любым количеством узлов и текста, желателен полифилл:
  + append/prepend – вставка в конец/начало.
  + before/after – вставка перед/после.
  + replaceWith – замена.