Изменение документа

Модификации DOM – это ключ к созданию «живых» страниц.

document.createElement(tag) - Создаёт новый элемент с заданным тегом:

```
let div = document.createElement('div');
div.className = "alert";
div.innerHTML = "<strong>Всем привет!</strong> Вы прочитали важное сообщение.";
```

document.createTextNode(text) - Создаёт новый текстовый узел с заданным текстом:

```
let textNode = document.createTextNode('А вот и я');
```

Методы вставки

Вот методы для различных вариантов вставки:

node.append(...nodes or strings) – добавляет узлы или строки в конец node,

```
document.body.append(div);
```

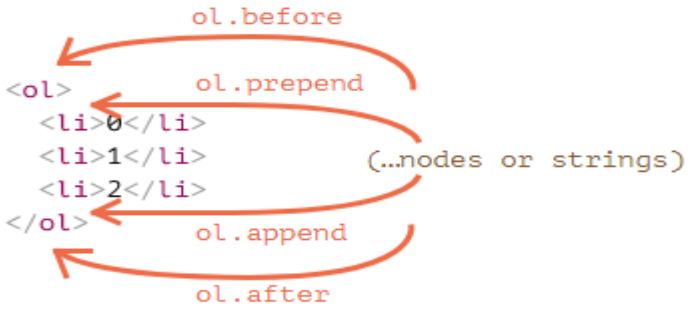
node.prepend(...nodes or strings) – вставляет узлы или строки в начало node,

node.before(...nodes or strings) –- вставляет узлы или строки до node,

node.after(...nodes or strings) –- вставляет узлы или строки после node,

node.replaceWith(...nodes or strings) –- заменяет node заданными узлами или строками.

```
before
<ol id="ol">
  <li>prepend</li>
  <li>0</li>
  <li>1</li>
  <li>2</li>
  <li>append</li>
</ol>
after
```



Эти методы могут использоваться только для вставки DOM-узлов или текстовых фрагментов.

А что, если мы хотим вставить HTML именно «как html», со всеми тегами и прочим, как делает это elem.innerHTML?

insertAdjacentHTML/Text/Element

С этим может помочь другой, довольно универсальный метод:

elem.insertAdjacentHTML(where, html).

Первый параметр – это специальное слово, указывающее, куда по отношению к elem производить вставку.

Значение должно быть одним из следующих:

"**beforebegin**" – вставить html непосредственно перед elem,

"**afterbegin**" – вставить html в начало elem,

"**beforeend**" – вставить html в конец elem,

"**afterend**" – вставить html непосредственно после elem.

Второй параметр – это HTML-строка, которая будет вставлена именно «как HTML».

Например:

```
<div id="div"></div>
<script>
  div.insertAdjacentHTML('beforebegin', '<p>Привет</p>');
  div.insertAdjacentHTML('afterend', '<p>Пока</p>');
</script>
```

…Приведёт к:

```
<p>Привет</p>
```

```
<div id="div"></div>
<p>Пока</p>
```

У метода есть два брата:
elem.insertAdjacentText(where, text) – такой же синтаксис, но строка text вставляется «как текст», вместо HTML,
elem.insertAdjacentElement(where, elem) – такой же синтаксис, но вставляет элемент elem.
Они существуют, в основном, чтобы унифицировать синтаксис. На практике часто используется только insertAdjacentHTML. Потому что для элементов и текста у нас есть методы append/prepend/before/after – их быстрее написать, и они могут вставлять как узлы, так и текст.

Удаление узлов

Для удаления узла есть методы **node.remove()**

Если нам нужно переместить элемент в другое место – нет необходимости удалять его со старого.
Все методы вставки автоматически удаляют узлы со старых мест.

Клонирование узлов

Вызов **elem.cloneNode(true)** создаёт «глубокий» клон элемента – со всеми атрибутами и дочерними элементами.
Если мы вызовем **elem.cloneNode(false)**, тогда клон будет без дочерних элементов. **false по умолчанию.**

```
let div2 = div.cloneNode(true); // клонировать сообщение
div2.querySelector('strong').innerHTML = 'Всем пока!'; // изменить клонированный элемент
div.after(div2); // показать клонированный элемент после существующего div
```

При **true** смотреть, чтобы не дублировались **id!**

Шаблонные строки

```
var container = document.querySelector(".container");
var heading = "Title1";
var text = "Some text";
var templateString = `<h2>${heading}</h2><p>${text}</p>`;
container.insertAdjacentHTML("beforeend", templateString);
```

Устаревшие методы вставки/удаления

parentElem.appendChild(node) - Добавляет node в конец дочерних элементов parentElem.

parentElem.insertBefore(node, nextSibling) - Вставляет node перед nextSibling в parentElem.

parentElem.replaceChild(node, oldChild) - Заменяет oldChild на node среди дочерних элементов parentElem.

parentElem.removeChild(node) - Удаляет node из parentElem (предполагается, что он родитель node).

document.write.
Есть ещё один, очень древний метод добавления содержимого на веб-страницу: **document.write**.

```
<script>
  document.write('<b>Привет из JS</b>');
</script>
```

Вызов document.write(html) записывает html на страницу «прямо здесь и сейчас». Строка html может быть динамически сгенерирована, поэтому метод достаточно гибкий. Мы можем использовать JavaScript, чтобы создать полноценную веб-страницу и записать её в документ. В современных скриптах он редко встречается из-за следующего важного ограничения: Вызов document.write работает только во время загрузки страницы. Если вызвать его позже, то существующее содержимое документа затрётся.