Урок 11

Создание элемента

document.createElement(tag)

Создаёт новый элемент с заданным тегом:

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

Изменения, создание и удаление класса

className и classList

Изменение класса является одним из наиболее часто используемых действий в скриптах.

Когда-то давно в JavaScript существовало ограничение: зарезервированное слово типа "class" не могло быть свойством объекта. Это ограничение сейчас отсутствует, но в то время было невозможно иметь свойство elem.class.

Поэтому для классов было введено схожее свойство "**className**": elem.className соответствует атрибуту "class".

Например:

```
<body class="main block">
    <script>
        console.log(document.body.className); //main block
        document.body.className = 'main_block'; //изменили значение
    </script>
</body>
```

Если мы присваиваем что-то elem.className, то это заменяет всю строку с классами. Иногда это то, что нам нужно, но часто мы хотим добавить/удалить один класс.

Для этого есть другое свойство: elem.**classList**.

classList – это специальный объект с методами для добавления/удаления одного класса.

Например:

```
<body class="main">
    <script>
        document.body.classList.add('article'); //добавление класса
        console.log(document.body.className); //main article
    </script>
</body>
```

Так что мы можем работать как со строкой полного класса, используя className, так и с отдельными классами, используя classList. Выбираем тот вариант, который нам удобнее.

Методы classList:

- **elem.classList.add/remove("class")** – добавить/удалить класс.
- **elem.classList.toggle("class")** – добавить класс, если его нет, иначе удалить.
- **elem.classList.contains("class")** – проверка наличия класса, возвращает true/false.

Кроме того, **classList** является **перебираемым**, поэтому можно перечислить все классы при помощи **for...of**:

```
<body class="main page">
    <script>
      for (let name of document.body.classList) {
        console.log(name); // main, затем page
      }
    </script>
</body>
```

Создание сообщения

В нашем случае сообщение – это div с классом alert и HTML в нём:

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

Мы создали элемент, но пока он только в переменной. Мы не можем видеть его на странице, поскольку он не является частью документа.

Методы вставки

Чтобы наш div появился, нам нужно вставить его где-нибудь в document. Например, в document.body.

Для этого есть метод **append**, в нашем случае: document.body.append(div).

Вот пример:

```
let div = document.createElement('div');
div.classList.add("alert");
div.innerHTML = "<b>Всем привет!</b> Вы прочитали важное сообщение.";
document.body.append(div);
```

Вот методы для различных вариантов вставки:

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

Вот пример использования этих методов, чтобы добавить новые элементы в список и текст до/после него:

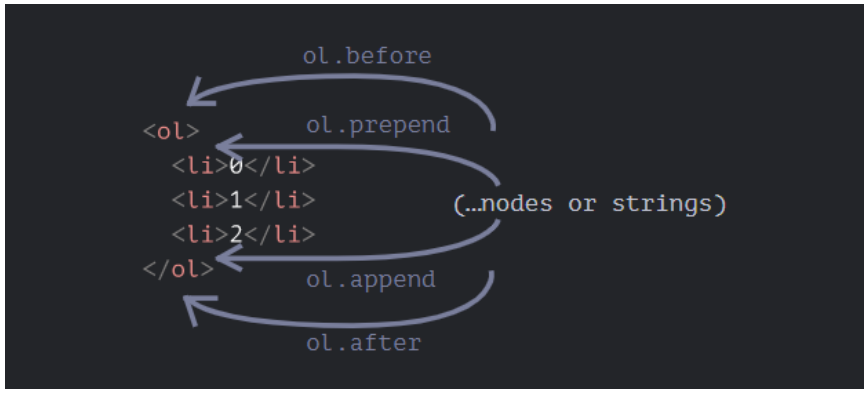
```
<ol id="ol">
  <li>0</li>
  <li>1</li>
  <li>2</li>
</ol>

<script>
  ol.before('before'); // вставить строку "before" перед <ol>
  ol.after('after'); // вставить строку "after" после <ol>

  let liFirst = document.createElement('li');
  liFirst.innerHTML = 'prepend';
  ol.prepend(liFirst); // вставить liFirst в начало <ol>

  let liLast = document.createElement('li');
  liLast.innerHTML = 'append';
  ol.append(liLast); // вставить liLast в конец <ol>
</script>
```

Наглядная иллюстрация того, куда эти методы вставляют:



Удаление узлов

Для удаления узла есть методы node.remove().

Например, сделаем так, чтобы наш первый блок удалился:

```
    <div class="block1">
        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti
fugiat voluptatem sint unde voluptatum quaerat odio repellendus, nesciunt optio,
suscipit assumenda hic accusamus beatae, illum recusandae amet ipsum. Quos,
eaque.</p>
    </div>
    <div class="block2">
        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti
fugiat voluptatem sint unde voluptatum quaerat odio repellendus, nesciunt optio,
suscipit assumenda hic accusamus beatae, illum recusandae amet ipsum. Quos,
eaque.</p>
    </div>
    <script>
        let div = document.querySelector('.block1').remove();
    </script>
```

Задане: Создайте функцию clear(elem), которая удаляет все элементы списка.

```
    <ol>
        <li>Привет</li>
        <li>Мир</li>
    </ol>

    <script>
        function clear(elem) { /* ваш код */ }

        clear('li');
    </script>
```

Решение:

```
function clear(elem) {
        elements = document.querySelectorAll(elem);
        console.log(elements);
        for (let list of elements) {
             list.remove();
}
```

Задание: Напишите интерфейс для создания списка.

Для каждого пункта:

1. Запрашивайте содержимое пункта у пользователя с помощью prompt.
2. Создавайте элемент <li> и добавляйте его к <ul>.
3. Продолжайте до тех пор, пока пользователь не отменит ввод (введя пустую строку).

Решение:

```
let ul = document.createElement('ul');
document.body.append(ul);

while (true) {
    let data = prompt("Введите текст для элемента списка", "");

    if (data=="") {
        break;
    }

    let li = document.createElement('li');
    li.innerHTML = data;
    ul.append(li);
}
```

CallBack функция

Простыми словами: коллбэк — это функция, которая должна быть выполнена после того, как другая функция завершила выполнение (отсюда и название: callback — функция обратного вызова).

Чуть сложнее: В JavaScript функции — это объекты. Поэтому функции могут принимать другие функции в качестве аргументов, а также возвращать функции в качестве результата. Функции, которые это умеют, называются функциями высшего порядка. А любая функция, которая передается как аргумент, называется callback-функцией.

```
function myFunc(callback){
    let a = [4, 1, 8];
    let element = document.querySelector('.paragraph');
    callback(element, a);
}

function out(elem, arr){
    elem.innerHTML = arr.join(',');
}

myFunc(out);
```

Метод **join([separator])** объединяет все элементы массива (или массивоподобного объекта) в строку.

Separator - Определяет строку, разделяющую элементы массива. В случае необходимости тип разделителя приводится к типу Строка. Если он не задан, элементы массива разделяются запятой ','. Если разделитель - пустая строка, элементы массива ничем не разделяются в возвращаемой строке.