







Frontender[1.0] JavaScript - DOM - создание, получение, добавление, удаление элементов, querySelector

 YouTube	https://youtu.be/DftDGDZBjZs
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	29



Задачи к этому уроку тут https://boosty.to/dmitry_ko



Полезные ссылки:

<https://learn.javascript.ru/searching-elements-dom>

<https://learn.javascript.ru/basic-dom-node-properties>

<https://learn.javascript.ru/modifying-document>

Основные и самые популярные методы для поиска элементов DOM дерева.

Есть 6 основных методов поиска элементов в DOM:

самые популярные

Метод	Ищет по...	Ищет внутри элемента?	Возвращает живую коллекцию?
<code>querySelector</code>	CSS-selector	✓	-
<code>querySelectorAll</code>	CSS-selector	✓	-
<code>getElementById</code>	id	-	-
<code>getElementsByName</code>	name	-	✓
<code>getElementsByTagName</code>	tag or '*'	✓	✓
<code>getElementsByClassName</code>	class	✓	✓

Все методы

getElementsBy* возвращают **живую** коллекцию. Такие коллекции всегда отражают текущее состояние документа и автоматически обновляются при его изменении.

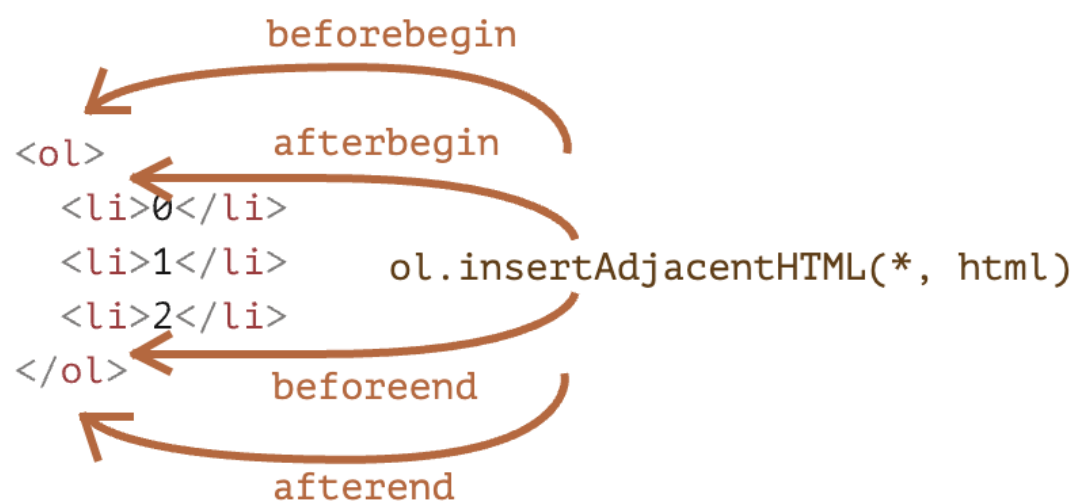
Напротив,

querySelectorAll возвращает **статическую** коллекцию. Это похоже на фиксированный массив элементов.



insertAdjacentHTML/Text/Element

insertAdjacentHTML - самый удобный и универсальный



```
<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="script.js" defer></script>
  <title>DOM - get elements</title>
</head>

<body>
  <div class="wrapper">
    <p id="first-text">text 1</p>
    <p>text 2</p>
    <p>text 3</p>
    <p class="last-text">text 4</p>
  </div>

  <div class="container"></div>
</body>

</html>

```

```

// DOM - создание, получение, добавление, удаление элементов

```

```

const log = console.log;
const dir = console.dir;

```

```

// Методы получения элементов

```

```

// querySelector - самый универсальный, удобный и популярный

```

```

const p = document.querySelector("p"); // самый универсальный
log(p);
dir(p);

```

```

const allP = document.querySelectorAll("p");
log(allP);
log(Array.from(allP));

```

```

const lastText = document.querySelector(".last-text");
// const lastText = document.querySelector("p[class='last-text']");
// const lastText = document.querySelector("[class='last-text']");
dir(lastText);

```

```

// const firstText = document.querySelector("#first-text");
// const firstText = document.getElementById("first-text");
// dir(firstText);

```

```

const wrapper = document.querySelector(".wrapper");
log(wrapper);

```

```

// const firstText = wrapper.getElementById("first-text"); // ошибка! getElementById используется только через document

```

```

// querySelector можно использовать на элементах не только через document

```

```

const firstText = wrapper.querySelector("#first-text");
dir(firstText);

```

```

// 💎-----💎-----💎

```

```

// getElementsByTagName* - возвращают живые коллекции, но лучше использовать querySelector

```

```

const allTexts = document.getElementsByTagName("p");

```

```

dir(allTexts);

// allTexts.forEach(element => { // ошибка
//     log(element)
// });

allP.forEach(element => { // allP - NodeList (массив на минималках)
    log(element);
});
Array.from(allP).forEach(element => { // Array.from(allP) - полноценный массив
    log(element);
});

// 💎-----💎-----💎
log(wrapper.textContent); // строка только с текстом внутри
log(wrapper.innerHTML); // строка html элементов с контентом внутри
log(wrapper.outerHTML); // строка html элементов с контентом + сам элемент wrapper

// wrapper.textContent = "hello";
// log(wrapper);

// wrapper.innerHTML = "<h2>text from JavaScript</h2>";
// wrapper.innerHTML += "<h2>text from JavaScript</h2>";
// log(wrapper);

wrapper.prepend("text with prepend"); // добавляет текст внутри перед контентом
wrapper.append("text with append"); // добавляет текст внутри после контента

wrapper.before("text with before"); // добавляет текст снаружи перед контентом
wrapper.after("text with after"); // добавляет текст снаружи после контента

// wrapper.append("<h2>text with append</h2>"); // добавится как строка

const h2 = document.createElement("h2"); // создали элемент h2
h2.textContent = "наш созданный h2"; // добавили элементу контент

wrapper.append(h2);
log(wrapper);

// lastText.innerHTML = ""; // удаляет все внутри элемента
lastText.remove(); // удаляет элемент

// 💎-----💎-----💎
// insertAdjacentHTML/Text/Element

const container = document.querySelector(".container");
const h3 = document.createElement("h3");
h3.textContent = "наш созданный h3";

// -- beforebegin, afterbegin, beforeend, afterend --
// container.insertAdjacentElement("afterbegin", h3);

const copiedContainer = container;

container.insertAdjacentText("beforebegin", "1) текст над контейнером");
container.insertAdjacentText("afterbegin", "2) текст вначале в контейнере");
container.insertAdjacentText("beforeend", "3) текст вконец в контейнере");
container.insertAdjacentText("afterend", "4) текст под контейнером");

```

```
// insertAdjacentHTML - самый удобный и универсальный
container.insertAdjacentHTML("afterbegin", `
  <hr>
  <button>click me</button>
  <hr>
`);

// 💎-----💎-----💎

log(copiedContainer === container);
log(copiedContainer.textContent, container.textContent);

// cloneNode - копирует элемент
// если true создаётся «глубокий» клон элемента со всеми атрибутами и дочерними элементами
// если false, то клон будет без дочерних элементов
const deepCopiedContainer = container.cloneNode(true);

log(deepCopiedContainer === container);
```