ПМ3 Разработка модулей ПО.

РО 3.2 Разрабатывать модули с применением DOM API, Regexp, HTTP

Тема 1. JS: DOM API.

Лекция 5. Поиск элементов и работа с коллекциями.

Цель занятия:

Научиться находить элементы в DOM разными методами, разбираться в типах коллекций (HTMLCollection, NodeList) и уметь использовать CSS-селекторы и атрибуты для выборки элементов.

Учебные вопросы:

1. Методы поиска элементов в DOM.
2. HTMLCollection и NodeList.

1. Методы поиска элементов в DOM.

1. getElementById(id)

- Находит один элемент по его уникальному **id**.
- Если не найден → вернёт **null**.
- Работает очень быстро (браузер оптимизирован под id).

```
<p id="info"></p>
```

```
let el = document.getElementById("info");
el.textContent = "Hello!";
console.log(el.textContent); // "Hello!"
```

◇ 2. getElementsByClassName(className)

- Возвращает живую коллекцию (HTMLCollection) всех элементов с данным классом.
- Если класс у элемента изменится → коллекция обновится.

```
<p class="note">Первый</p>
<p class="note">Второй</p>
```

```
let notes = document.getElementsByClassName("note");
console.log(notes.length); // 2
console.log(notes[0]); // <p class="note">Первый</p>
```

🔷 3. getElementsByTagName(tagName)

Возвращает живую коллекцию элементов по тегу (<li>, <p>, и т.д.).

Можно вызывать у любого элемента, чтобы искать внутри него.

```
<ul>
  <li>Элемент 1</li>
  <li>Элемент 2</li>
</ul>
```

```
let items = document.getElementsByTagName("li");
console.log(items[0].textContent); // "Элемент 1"
```

◇ 4. querySelector(cssSelector)

- Принимает любой CSS-селектор.
- Возвращает первый подходящий элемент или null.
- Удобен, если нужен один конкретный элемент.

```
<p class="note">Пример</p>
```

```
let el = document.querySelector(".note");
console.log(el.textContent); // "Пример"
```

◇ 5. querySelectorAll(cssSelector)

Возвращает статическую коллекцию (NodeList) всех элементов по селектору.

Коллекция не обновляется при изменении DOM.

У NodeList есть метод forEach() → удобно обходить.

```
<div class="box"></div>
<div class="box"></div>
```

```
let boxes = document.querySelectorAll(".box");
boxes.forEach(box => box.style.border = "1px solid red");
```

Сравнительная таблица

Метод	Что возвращает	Живой/Статический	Когда использовать
getElementById	Один элемент	–	Уникальный id
getElementsByClassName	HTMLCollection (по классу)	Живой	Массовый выбор по классу
getElementsByTagName	HTMLCollection (по тегу)	Живой	Все элементы определённого тега
querySelector	Первый элемент	–	Любой CSS-селектор (один элемент)
querySelectorAll	NodeList (по селектору)	Статический	Массовый выбор через CSS-селекторы

getElement* → для простых случаев (id, класс, тег).

querySelector* → для сложных селекторов и комбинаций.

2. HTMLCollection и NodeList.

- **Зачем нужны коллекции?**

Когда мы ищем элементы в DOM (getElementsByClassName, querySelectorAll и др.), результатом часто является список элементов.

Это не массив, а специальные коллекции.

◇ HTMLCollection

Возвращается методами:

- document.getElementsByTagName("div")
- document.getElementsByClassName("item")
- element.children

Особенности:

- Это живая коллекция — автоматически обновляется при изменении DOM.
- Содержит только элементы (узлы типа element).

👉 Пример:

```
<body>
  <ul>
    <li>Первый</li>
    <li>Второй</li>
  </ul>

<script>
  let items = document.getElementsByTagName("li");
  console.log(items.length); // 2
  document.querySelector("ul").append(document.createElement("li"));
  console.log(items.length); // уже 3 (коллекция обновилась!)
</script>
```

◆ NodeList.

Возвращается методами:

- document.querySelectorAll("div")
- element.childNodes

Особенности:

- Может содержать не только элементы, но и текстовые/комментарии.
- Чаще всего это статическая коллекция (например, querySelectorAll).
- Есть исключения: Node.childNodes — живая NodeList.

👉 Пример:

```
  <ul>
    <li>Первый</li>
    <li>Второй</li>
  </ul>

<script>
  let items = document.querySelectorAll("li");
  console.log(items.length); // 2
  document.querySelector("ul").append(document.createElement("li"));
  console.log(items.length); // всё ещё 2 (список статический)
</script>
```

◇ Обход коллекций.

Коллекции похожи на массивы, но это не массивы:

У них нет методов map, filter, reduce.

Можно обходить:

```
for (let el of items) {  } // работает
items.forEach(el => {  });     // работает у NodeList
```

Преобразование в массив:

```
let arr1 = Array.from(items);
// или
let arr2 = [...items];
```

👉 Выводы:

- HTMLCollection → живая, только элементы.
- NodeList → обычно статическая, может содержать любые узлы.
- Для удобной работы коллекции можно превращать в массивы (Array.from, spread-оператор [...]).

Контрольные вопросы:

- Чем отличается getElementById от querySelector?
- В чём разница между HTMLCollection и NodeList?
- Что значит, что коллекция «живая»?

Домашнее задание:

1. https://ru.hexlet.io/courses/js-dom