

# СПОСОБЫ ПОИСКА НУЖНОГО HTML-ЭЛЕМЕНТА



**ВЛАДИМИР ЧЕБУКИН**



# ВЛАДИМИР ЧЕБУКИН

Frontend-разработчик



[vovachebr@mail.ru](mailto:vovachebr@mail.ru)



[fb.me/vovachebr](https://fb.me/vovachebr)



[@User123423](https://t.me/@User123423)

# ПЛАН ЗАНЯТИЯ

1. Получение DOM-элементов по их особенностям
2. querySelector и querySelectorAll
3. Коллекции. Объект HTMLCollection
4. HTMLCollection и NodeList
5. \$0 в консоли



# **ВСПОМНИМ ПРОШЛЫЕ ЗАНИЯТИЯ**



# ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

**Вопрос:** что может содержаться в любом HTML элементе?

# ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

**Ответ:** название тега, его атрибуты и содержимое.

```
...<html lang="en"> == $0
  ▶ <head>...</head>
  ▼ <body>
    <a href="ya.ru">Ссылка на яндекс</a>
    <script src="main.js"></script>
  </body>
</html>
```

**тег** (points to `<body>`)

**аттрибут** (points to `src="main.js"`)

**содержимое** (points to `<script>`)



# ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

**Вопрос:** А для чего используется JavaScript?



# ВСПОМНИМ ПРОШЛЫЕ ЗАНЯТИЯ

**Ответ:** Для динамического изменения элементов страницы.

Возникает проблема: если нужно взаимодействовать с HTML-элементами, то нужно их как-то получить и использовать.



---

# **ПОЛУЧЕНИЕ ДОМ-ЭЛЕМЕНТОВ ПО ИХ ОСОБЕННОСТЯМ**

# МЕТОДЫ

Для получения HTML-элементов со страницы существуют методы:

- `getElementsByTagName` ;
- `getElementsByName` ;
- `getElementsByClassName` ;
- `getElementById` .

**ВАЖНО!!!** Все эти функции есть у объекта `document` (а также у каждого HTML-элемента), который находится в глобальной области видимости.

Исключением является только функция `getElementById` , которая находится только у объекта `document`.

**Вопрос:** как думаете, почему функцию `getElementById` нельзя применить к отдельному элементу HTML-страницы?

# ПОЛУЧЕНИЕ DOM-ЭЛЕМЕНТОВ ПО ИХ ОСОБЕННОСТЯМ

**Ответ:** потому что элемент с уникальным id может быть только один на всей странице.

Методы получения HTML-элементов очень важны, так как именно они делают возможным взаимодействие с HTML-документом и тем самым обеспечивают интерактивность веб-страницы. А это есть основное назначение языка Javascript.

Давайте более подробно рассмотрим функции получения HTML-элементов.

# getElementsByTagName

Допустим, необходимо получить все изображения на странице.

Для этого можно воспользоваться функцией

`getElementsByTagName()`.

Функция `getElementsByTagName()` аргументом принимает название тега.

```
// Получение всех изображений со страницы  
let images = document.getElementsByTagName("img");
```

# getElementsByClassName

Для получения всех элементов, содержащих определенный класс, стоит воспользоваться функцией `getElementsByClassName()`.

Функция `getElementsByClassName()` аргументом принимает название класса.

**Это пригодится при выполнении домашнего задания**

```
1  const elementsRed = document.getElementsByClassName('red');
2  const elementsSelected = document.getElementsByClassName('selected');
3  const elementsBlue = document.getElementsByClassName('blue');
```

# getElementsByName

Некоторые элементы могут иметь атрибут `name`. Функция `getElementsByName()` позволяет получить все элементы с данным атрибутом `name`.

```
let elements = document.getElementsByName('age');
```

## getElementById

Чтобы получить определенный элемент по его ID, следует воспользоваться функцией `getElementById()`.

```
let loginButton = document.getElementById('loginBtn');
```

## ПОЛУЧЕНИЕ ЭЛЕМЕНТА ПО `id`

Помимо функции `getElementById()`, уникальный элемент находится в глобальной области видимости (`window`). Следующие вызовы будут идентичными.

```
let loginButton = document.getElementById('loginBtn');  
let loginButtonSame = window.loginBtn;
```



## `getElements*` И `getElement*`

Вопрос: функции для получения элементов начинаются с `getElements`, и только `getElementById` не содержит «s». Как вы думаете, почему?

## `getElements*` И `getElement*`

Ответ: все методы получения элементов (кроме `getElementById()`) возвращают коллекцию элементов. Метод `getElementById()` возвращает только один элемент, т.к. на странице может быть только один элемент с уникальным ID.



querySelector И  
querySelectorAll

## querySelector И querySelectorAll

**Это пригодится при выполнении домашнего задания**

Иногда нужно получить определенные элементы, для которых нельзя воспользоваться предыдущими функциями.

Например: нужно получить все элементы списка с классом `exclusive`.

Функция `querySelectorAll()` схожа с предыдущими функциями.

Она также возвращает коллекцию элементов по входному CSS-селектору.

## ПРИМЕР

```
1 // Получаем все элементы списка с классом "exclusive"
2 let exclusiveElements = document.querySelectorAll("ul.exclusive li");
3 // Получаем первый элемент списка, у которого есть класс "exclusive"
4 let exclusiveElements = document.querySelector("ul li.exclusive");
```

# ПОЛУЧЕНИЕ РОДИТЕЛЬСКОГО ЭЛЕМЕНТА

Если мы получили какой-нибудь HTML-элемент, то как мы можем получить его родителя?

**Это пригодится при выполнении домашнего задания**

1. Свойство `parentElement` указывает на ближайшего родителя;
2. Метод `closest`, который по CSS-селектору возвращает ближайшего родителя.

# ПОЛУЧЕНИЕ РОДИТЕЛЬСКОГО ЭЛЕМЕНТА

```
1 let listElements = document.getElementsByTagName("li");
2 let firstElement = listElements.item(0);
3 console.log(firstElement.parentElement); // <ul class="exclusive">...</ul>
4 console.log(firstElement.closest("ul")); // <ul class="exclusive">...</ul>
5 console.log(firstElement.closest("*")); // <li>Element 1</li>
6 console.log(document.parentElement); // null, т.к. document является корневым элементом
```

## РЕЗЮМИРУЕМ

- `getElementById` – получает один элемент по `id`;
- `getElementsByTagName` – получает коллекцию элементов по тегу;
- `getElementsByClassName` – получает коллекцию элементов по классу;
- `getElementsByName` – получает коллекцию элементов по имени;
- `querySelector` – получает первый элемент по CSS-селектору;
- `querySelectorAll` – получает все элементы по CSS-селектору;
- `closest` – по CSS-селектору возвращает ближайшего родителя.





# КОЛЛЕКЦИИ. ОБЪЕКТ HTMLCollection



**Вопрос:** является ли результат получения элементов массивом?

## ОБЪЕКТ `HTMLCollection`

**Ответ:** как и `arguments`, получение коллекции элементов является псевдомассивом, а значит к `HTMLCollection` нельзя применить методы для обработки массивов.

`HTMLCollection` — это список узлов. Отдельный узел может быть доступен по порядковому номеру или имени узла и атрибута.

# ПОЛУЧЕНИЕ МАССИВА HTML-ЭЛЕМЕНТОВ

Для получения массива HTML-элементов следует воспользоваться функцией `Array.from()`:

```
1 let links = document.getElementsByTagName("a");
2 console.log(Array.isArray(links)); // false
3 let arr = Array.from(links);
4 console.log(Array.isArray(arr)); // true
```

# ПОЛУЧЕНИЕ ОПРЕДЕЛЕННОГО ЭЛЕМЕНТА

Часто приходится взаимодействовать не с целой коллекцией, а с определенным элементом. Для этого у элемента `HTMLCollection` есть метод `item()`, позволяющий получить элемент по его позиции:

```
1 let listElements = document.getElementsByTagName("li");
2 let firstElement = listElements.item(0);
3 // Получение элемента как из массива тоже возможно
4 let secondElement = listElements[1];
```

# ПОЛУЧЕНИЕ ОПРЕДЕЛЕННОГО ЭЛЕМЕНТА ПО ЕГО ИМЕНИ

Иногда нужно получить элемент по его имени. Для этого можно воспользоваться методом `namedItem()`:

```
let ageInput = document.getElementsByTagName("input").namedItem("age");
```

# КОЛИЧЕСТВО НАЙДЕННЫХ ЭЛЕМЕНТОВ

Иногда необходимо узнать количество найденных элементов, для этого у коллекции `HTMLCollection` есть свойство `length` (доступно только для чтения):

```
let links = document.getElementsByTagName("a");  
console.log(links.length);
```

## ПРОБЛЕМЫ С `length`

Свойство `length` показывает количество элементов коллекции. Однако, если структура DOM изменяется, эти изменения могут затронуть и объект `HTMLCollection`, а значит и свойство `length`.

Рассмотрим пример:

1. Получим HTML-элементы;
2. Сделаем цикл по всем элементам коллекции (т.е. пока итератор не достигнет `length`);
3. Будем изменять HTML-элементы. (Например, добавлять новые.)

**Вопрос:** всё ли хорошо в таком алгоритме?



## ПРОБЛЕМЫ С `length`

**Ответ:** нет. Т.к. `length` обновляется динамически, то при добавлении элемента `length` будет изменяться (увеличиваться), а значит это грозит проблемой бесконечного цикла.

*Как исправить проблему?*

**Перед циклом определить начальное количество элементов и цикл сделать до этой переменной.**

Что получится в итоге?

Свойство `length` будет изменяться, но это никак не повлияет на значение, которое мы получили заранее, а значит итератор будет изменяться до начального значения `length`, которое мы получили до всех изменений.



# HTMLCollection и NodeList

## ОБЪЕКТ `NodeList`

Объект `NodeList` является коллекцией узлов. `NodeList` обеспечивает уровень абстракции над коллекцией узлов, не определяя и не ограничивая как эта коллекция используется.

Объект `NodeList` можно получить такими методами, как `Node.childNodes` и `document.querySelectorAll`.

# РАЗЛИЧИЯ МЕЖДУ `HTMLCollection` И `NodeList`

Коллекция HTML всегда находится в DOM, в то время как `NodeList` является более универсальной конструкцией, которая может или не может быть в DOM.



**\$Ø В КОНСОЛИ**

## \$0 В КОНСОЛИ

Иногда нужно «поэкспериментировать», выполняя какие-либо действия с HTML элементом.

```
> $0
< ▶<div class="About-module-about-2tLLvF" id="about">...</div>
> $0.id
< "about"
> $0.nextSibling
< ▶<div class="PopularPrograms-module-programs-3PB6i0" id="programs">...</div>
> $0.nodeName
< "DIV"
> $0.hasChildNodes()
< true
>
```

## \$0 В КОНСОЛИ

Чтобы не писать специальные запросы для получения элемента, его можно просто выбрать, а в консоли обращаться к нему с помощью `$0`.

```

    <section>
      <div class="presentation_wrapper" id="presentation">
        <div class="Presentation-module-presentation-1ovvxW">
          <video autoplay loop class="Presentation-module-video-1uaI8o" style="display: block; width: 1570px; height: 882px;">_</video>
          <div class="Presentation-module-videoOverlay-3Ywu8T" style="opacity: 0.6;"></div>
          <div class="Presentation-module-content-nWYi5F" style="transform: translateY(0px);">_</div>
        </div>
      </div>
    </section>
  ...
  <div class="About-module-about-2tLLvF" id="about">_</div> == $0
  <div class="PopularPrograms-module-programs-3PB6i0" id="programs">_</div>
  <div class="Mentors-module-wrapper--kgknn" data-name="reviews" id="mentors">_</div>
  <div class="direction-module-directions-1kLsLQ" id="direction">_</div>
  <div class="subscriptionForm-module-root-10_I-k subscriptionForm-module-common-3kmnMJ">_</div>
</section>
</section>
<div class="footer-module-root-2P1nxC">_</div>
<div class="cookiesAgreement-module-root-2INjMM">_</div>
</div>
</div>

```

Над выбранным элементом можно производить такие же действия, как над любым элементом `HTMLElement`.

## НЕДОСТАТОК \$0

**Вопрос:** как вы думаете, что неудовлетворительного в использовании \$0 ?

**Ответ:** использовать \$0 элемент можно только выбрав его вручную, а значит, его использование удобно только в процессе разработки (в реальном коде использовать нельзя).



---

**\$1, \$2, \$3 И Т.Д.**

Иногда необходимо получить элементы, которые были выбраны ранее.

Для этого в консоли можно использовать **\$1** — элемент, который был выбран до текущего. **\$2** — элемент, который был выбран до элемента **\$1**. **\$3** — элемент, который был выбран до элемента **\$2** и т.д.

# РЕЗЮМИРУЕМ

- Объект `HTMLCollection` является псевдо-массивом;
- Длина `HTMLCollection` обновляется динамически и всегда показывает количество элементов;
- `$0` упрощает отладку кода и взаимодействие с DOM-деревом.



# ЛИСТИНГ КОДА

Весь код, используемый в лекции доступен по ссылке

<https://repl.it/@vovachebr/findingDOMElements>

## ЧЕМУ МЫ НАУЧИЛИСЬ?

1. Изучили основные функции получения HTML-элементов по тегу, имени, классу, `id`;
2. Изучили функции для получения элемента и элементов по CSS-селектору;
3. Узнали, как быстро получить HTML-элемент для отладки.



# ЧТО МЫ УЗНАЕМ НА СЛЕДУЮЩЕМ ЗАНЯТИИ

1. Что такое события;
2. Как подписываться на события и отписываться от них;
3. Какие бывают события;
4. Что такое объект события;
5. Контекст вызова.

# ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаем в Slack!
- Работы должны соответствовать принятому [стилю оформления кода](#).
- Зачет по домашней работе проставляется после того, как приняты все **3 задачи**.



# МАТЕРИАЛ ДЛЯ САМОСТОЯТЕЛЬНОГО ОЗНАКОМЛЕНИЯ

[Поиск: getElement\\* и querySelector\\* и не только](#)

[Работа с DOM из консоли](#)

[Навигация по DOM-элементам](#)

[Внутреннее устройство поисковых методов](#)



Спасибо за внимание!  
Время задавать вопросы 😊  
**ВЛАДИМИР ЧЕБУКИН**

 [vovachebr@mail.ru](mailto:vovachebr@mail.ru)

 [fb.me/vovachebr](https://fb.me/vovachebr)

 [@User123423](https://t.me/@User123423)