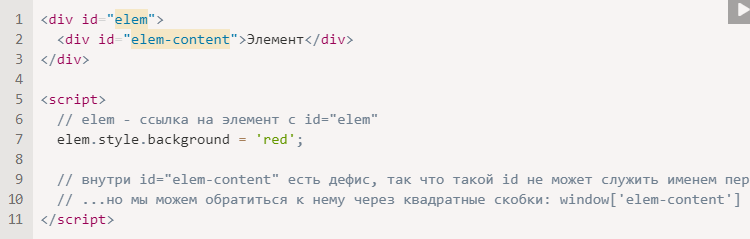
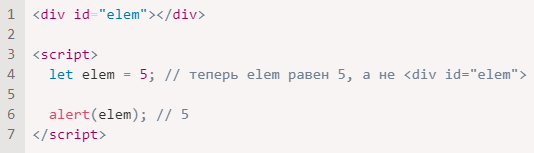
Помимо методов getElement\*, querySelector\* есть глобальная переменная с именем, указанным в id:



…Но это только если мы не объявили в JavaScript переменную с таким же именем, иначе она будет иметь приоритет:



**Пожалуйста, не используйте такие глобальные переменные для доступа к элементам**

Это поведение соответствует [стандарту](https://html.spec.whatwg.org/#dom-window-nameditem), но поддерживается в основном для совместимости, как осколок далёкого прошлого.

Браузер пытается помочь нам, смешивая пространства имён JS и DOM. Это удобно для простых скриптов, которые находятся прямо в HTML, но, вообще говоря, не очень хорошо. Возможны конфликты имён. Кроме того, при чтении JS-кода, не видя HTML, непонятно, откуда берётся переменная.

**Только document.getElementById, а не anyElem.getElementById**

Метод getElementById можно вызвать только для объекта document. Он осуществляет поиск по id по всему документу.

**Псевдоклассы тоже работают**

Псевдоклассы в CSS-селекторе, в частности :hover и :active, также поддерживаются. Например, document.querySelectorAll(':hover') вернёт коллекцию (в порядке вложенности: от внешнего к внутреннему) из текущих элементов под курсором мыши.

**[querySelector](https://learn.javascript.ru/searching-elements-dom" \l "querySelector)**

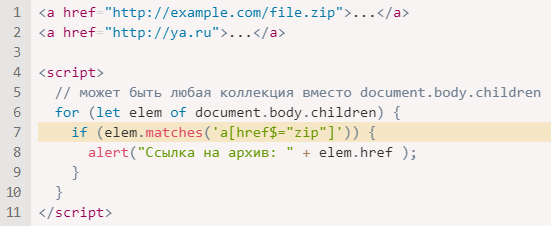
Метод elem.querySelector(css) возвращает первый элемент, соответствующий данному CSS-селектору.

Иначе говоря, результат такой же, как при вызове elem.querySelectorAll(css)[0], но он сначала найдёт *все* элементы, а потом возьмёт первый, в то время как elem.querySelector найдёт только первый и остановится. Это быстрее, кроме того, его короче писать.

## [matches](https://learn.javascript.ru/searching-elements-dom" \l "matches)

Метод проверяет, удовлетворяет ли elem CSS-селектору, и возвращает true или false.

Этот метод удобен, когда мы перебираем элементы (например, в массиве или в чём-то подобном) и пытаемся выбрать те из них, которые нас интересуют.



## [closest](https://learn.javascript.ru/searching-elements-dom" \l "closest)

Метод elem.closest(css) ищет ближайшего предка, который соответствует CSS-селектору. Сам элемент также включается в поиск.

Другими словами, метод closest поднимается вверх от элемента и проверяет каждого из родителей. Если он соответствует селектору, поиск прекращается. Метод возвращает либо предка, либо null, если такой элемент не найден.



## [getElementsBy\*](https://learn.javascript.ru/searching-elements-dom" \l "getelementsby)

Существуют также другие методы поиска элементов по тегу, классу и так далее.

На данный момент, они скорее исторические, так как querySelector более чем эффективен.

Hассмотрим их для полноты картины, также вы можете встретить их в старом коде.

* elem.getElementsByTagName(tag) ищет элементы с данным тегом и возвращает их коллекцию. Передав "\*" вместо тега, можно получить всех потомков.
* elem.getElementsByClassName(className) возвращает элементы, которые имеют данный CSS-класс.
* document.getElementsByName(name) возвращает элементы с заданным атрибутом name. Очень редко используется.

## [Живые коллекции](https://learn.javascript.ru/searching-elements-dom" \l "zhivye-kollektsii)

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

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

* elemA.contains(elemB) вернёт true, если elemB находится внутри elemA (elemB потомок elemA) или когда elemA==elemB.