Когда браузер загружает страницу, он «читает» (также говорят: «парсит») HTML и генерирует из него DOM-объекты. Для узлов-элементов большинство стандартных HTML-атрибутов автоматически становятся свойствами DOM-объектов.

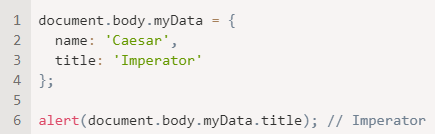
Например, для такого тега <body id="page"> у DOM-объекта будет такое свойство body.id="page".

Но преобразование атрибута в свойство происходит не один-в-один!

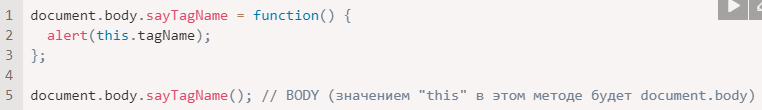
**[DOM-свойства](https://learn.javascript.ru/dom-attributes-and-properties" \l "dom-svoystva)**

Помимо встроенных DOM-свойств можемно добавить своё собственное свойство. DOM-узлы – это обычные объекты JavaScript. Мы можем их изменять.

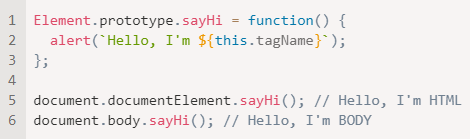
создадим новое свойство для document.body:



Мы можем добавить и метод:



Также можно изменять встроенные прототипы, такие как Element.prototype и добавлять новые методы ко всем элементам:



Итак, DOM-свойства и методы ведут себя так же, как и обычные объекты JavaScript:

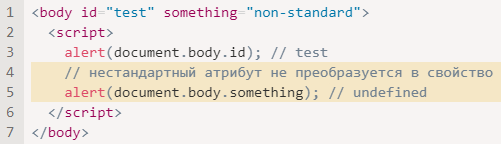
* Им можно присвоить любое значение.
* Они регистрозависимы (нужно писать elem.nodeType, не elem.NoDeTyPe).

## [HTML-атрибуты](https://learn.javascript.ru/dom-attributes-and-properties" \l "html-atributy)

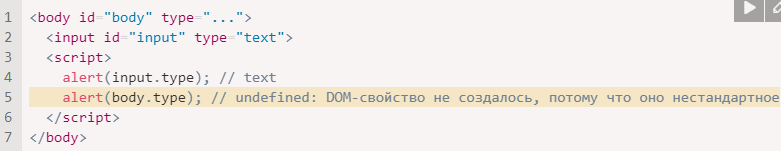
В HTML у тегов могут быть атрибуты. Когда браузер парсит HTML, чтобы создать DOM-объекты для тегов, он распознаёт *стандартные* атрибуты и создаёт DOM-свойства для них.

Таким образом, когда у элемента есть id или другой *стандартный* атрибут, создаётся соответствующее свойство. Но этого не происходит, если атрибут нестандартный.

Например:



Пожалуйста, учтите, что стандартный атрибут для одного тега может быть нестандартным для другого. Например, атрибут "type" является стандартным для элемента <input> ([HTMLInputElement](https://html.spec.whatwg.org/" \l "htmlinputelement)), но не является стандартным для <body> ([HTMLBodyElement](https://html.spec.whatwg.org/" \l "htmlbodyelement)).



Таким образом, для нестандартных атрибутов не будет соответствующих DOM-свойств. Есть ли способ получить такие атрибуты?

Конечно. Все атрибуты доступны с помощью следующих методов:

* elem.hasAttribute(name) – проверяет наличие атрибута.
* elem.getAttribute(name) – получает значение атрибута.
* elem.setAttribute(name, value) – устанавливает значение атрибута.
* elem.removeAttribute(name) – удаляет атрибут.

Эти методы работают именно с тем, что написано в HTML.

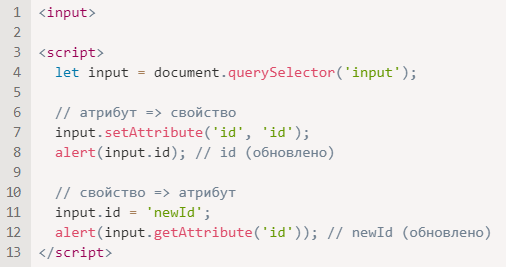
Кроме этого, получить все атрибуты элемента можно с помощью свойства elem.attributes: коллекция объектов, которая принадлежит ко встроенному классу [Attr](https://dom.spec.whatwg.org/" \l "attr) со свойствами name и value.

У HTML-атрибутов есть следующие особенности:

* Их имена регистронезависимы (id то же самое, что и ID).
* Их значения всегда являются строками.

## [Синхронизация между атрибутами и свойствами](https://learn.javascript.ru/dom-attributes-and-properties" \l "sinhronizatsiya-mezhdu-atributami-i-svoystvami)

Когда стандартный атрибут изменяется, соответствующее свойство автоматически обновляется. Это работает и в обратную сторону (за некоторыми исключениями). Например



Но есть и исключения, например, input.value синхронизируется только в одну сторону – атрибут → значение, но не в обратную:



В примере выше:

* Изменение атрибута value обновило свойство.
* Но изменение свойства не повлияло на атрибут.

Иногда эта «особенность» может пригодиться, потому что действия пользователя могут приводить к изменениям value, и если после этого мы захотим восстановить «оригинальное» значение из HTML, оно будет в атрибуте.

## [DOM-свойства типизированы](https://learn.javascript.ru/dom-attributes-and-properties" \l "dom-svoystva-tipizirovany)

DOM-свойства не всегда являются строками. Например, свойство input.checked (для чекбоксов) имеет логический тип.

Атрибут style – строка, но свойство style является объектом:



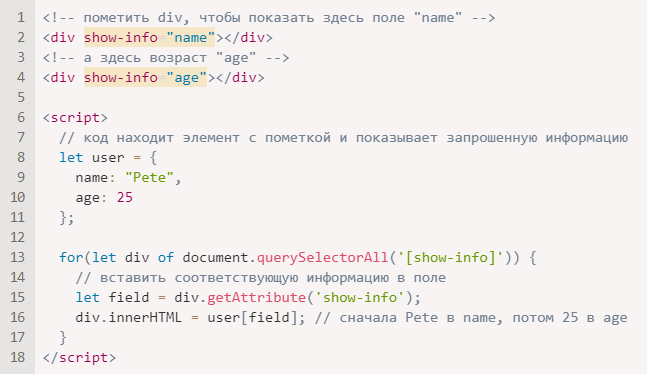
При этом некоторые из них, хоть и строки, могут отличаться от атрибутов. Например, DOM-свойство href всегда содержит полный URL, даже если атрибут содержит относительный URL или просто #hash.



Если же нужно значение href или любого другого атрибута в точности, как оно записано в HTML, можно воспользоваться getAttribute.

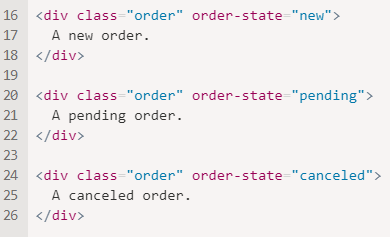
## [Нестандартные атрибуты, dataset](https://learn.javascript.ru/dom-attributes-and-properties" \l "nestandartnye-atributy-dataset)

Иногда нестандартные атрибуты используются для передачи пользовательских данных из HTML в JavaScript, или чтобы «помечать» HTML-элементы для JavaScript.



Почему атрибут может быть предпочтительнее классов

Это потому, что атрибутом удобнее управлять. Состояние может быть изменено достаточно просто:



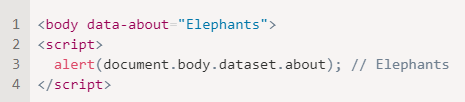


Но с пользовательскими атрибутами могут возникнуть проблемы. Что если мы используем нестандартный атрибут для наших целей, а позже он появится в стандарте и будет выполнять какую-то функцию? Язык HTML живой, он растёт, появляется больше атрибутов, чтобы удовлетворить потребности разработчиков. В этом случае могут возникнуть неожиданные эффекты.

Чтобы избежать конфликтов, существуют атрибуты вида [data-\*](https://html.spec.whatwg.org/#embedding-custom-non-visible-data-with-the-data-*-attributes).

**Все атрибуты, начинающиеся с префикса «data-», зарезервированы для использования программистами. Они доступны в свойстве dataset.**

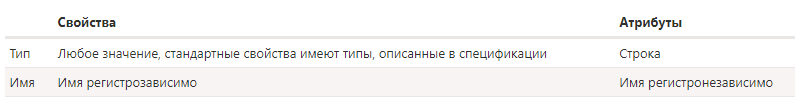
Например, если у elem есть атрибут "data-about", то обратиться к нему можно как elem.dataset.about.



Атрибуты, состоящие из нескольких слов, к примеру data-order-state, становятся свойствами, записанными с помощью верблюжьей нотации: dataset.orderState.

Использование data-\* атрибутов – валидный, безопасный способ передачи пользовательских данных.

Пожалуйста, примите во внимание, что мы можем не только читать, но и изменять data-атрибуты.



В большинстве ситуаций предпочтительнее использовать DOM-свойства. Нужно использовать атрибуты только тогда, когда DOM-свойства не подходят, когда нужны именно атрибуты, например:

* Нужен нестандартный атрибут. Но если он начинается с data-, тогда нужно использовать dataset.
* Мы хотим получить именно то значение, которое написано в HTML. Значение DOM-свойства может быть другим, например, свойство href – всегда полный URL, а нам может понадобиться получить «оригинальное» значение.