**Атрибуты и DOM-свойства**

При чтении HTML браузер генерирует DOM-модель. При этом большинство стандартных HTML-атрибутов становятся свойствами соответствующих объектов.

Например, если тег выглядит как <body id="page">, то у объекта будет свойство body.id = "page".

Но это преобразование – не один-в-один. Бывают ситуации, когда атрибут имеет одно значение, а свойство – другое. Бывает и так, что атрибут есть, а свойства с таким названием не создаётся.

Если коротко – HTML-атрибуты и DOM-свойства обычно, но не всегда соответствуют друг другу, нужно понимать, что такое свойство и что такое атрибут, чтобы работать с ними правильно.

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

Ранее мы видели некоторые встроенные свойства DOM-узлов. Но, технически, никто нас ими не ограничивает.

**Узел DOM – это объект, поэтому, как и любой объект в JavaScript, он может содержать пользовательские свойства и методы.**

Например, создадим в document.body новое свойство и запишем в него объект:

document.body.myData = {

name: 'Петр',

familyName: 'Петрович'

};

alert( document.body.myData.name ); // Петр

Можно добавить и новую функцию:

document.body.sayHi = function() {

alert( this.nodeName );

}

document.body.sayHi(); // BODY, выполнилась с правильным this

Нестандартные свойства и методы видны только в JavaScript и никак не влияют на отображение соответствующего тега.

Обратим внимание, пользовательские DOM-свойства:

* Могут иметь любое значение.
* Названия свойств *чувствительны* к регистру.
* Работают за счет того, что DOM-узлы являются объектами JavaScript.

**[Атрибуты](http://learn.javascript.ru/attributes-and-custom-properties" \l "atributy)**

Элементам DOM, с другой стороны, соответствуют HTML-теги, у которых есть текстовые атрибуты.

Конечно, здесь речь именно об узлах-элементах, не о текстовых узлах или комментариях.

Доступ к атрибутам осуществляется при помощи стандартных методов:

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

Эти методы работают со значением, которое находится в HTML.

Также все атрибуты элемента можно получить с помощью свойства elem.attributes, которое содержит псевдо-массив объектов типа [Attr](http://www.w3.org/TR/DOM-Level-3-Core/core.html" \l "ID-637646024).

В отличие от свойств, атрибуты:

* Всегда являются строками.
* Их имя *нечувствительно* к регистру (ведь это HTML)
* Видны в innerHTML (за исключением старых IE)

Рассмотрим отличия между DOM-свойствами и атрибутами на примере HTML-кода:

<body>

<div id="elem" about="Elephant" class="smiling"></div>

</body>

Пример ниже устанавливает атрибуты и демонстрирует их особенности.

<body>

<div id="elem" about="Elephant"></div>

<script>

alert( elem.getAttribute('About') ); // (1) 'Elephant', атрибут получен

elem.setAttribute('Test', 123); // (2) атрибут Test установлен

alert( document.body.innerHTML ); // (3) в HTML видны все атрибуты!

var attrs = elem.attributes; // (4) можно получить коллекцию атрибутов

for (var i = 0; i < attrs.length; i++) {

alert( attrs[i].name + " = " + attrs[i].value );

}

</script>

</body>

При запуске кода выше обратите внимание:

1. getAttribute('About') – первая буква имени атрибута About написана в верхнем регистре, а в HTML – в нижнем, но это не имеет значения, так как имена нечувствительны к регистру.
2. Мы можем записать в атрибут любое значение, но оно будет превращено в строку. Объекты также будут автоматически преобразованы.
3. После добавления атрибута его можно увидеть в innerHTML элемента.
4. Коллекция attributes содержит все атрибуты в виде объектов со свойствами name и value.

**[Когда полезен доступ к атрибутам?](http://learn.javascript.ru/attributes-and-custom-properties" \l "kogda-polezen-dostup-k-atributam)**

Когда браузер читает HTML и создаёт DOM-модель, то он создаёт свойства для всех *стандартных* атрибутов.

Например, свойства тега 'A' описаны в спецификации DOM: [HTMLAnchorElement](http://www.w3.org/TR/REC-DOM-Level-1/level-one-html.html" \l "ID-48250443).

Например, у него есть свойство "href". Кроме того, он имеет "id" и другие свойства, общие для всех элементов, которые описаны в спецификации в [HTMLElement](http://www.w3.org/TR/REC-DOM-Level-1/level-one-html.html" \l "ID-58190037).

Все стандартные свойства DOM синхронизируются с атрибутами, однако не всегда такая синхронизация происходит 1-в-1, поэтому иногда нам нужно значение именно из HTML, то есть атрибут.

Рассмотрим несколько примеров.

**[Ссылка «как есть» из атрибута href](http://learn.javascript.ru/attributes-and-custom-properties" \l "ssylka-kak-est-iz-atributa-href)**

Синхронизация не гарантирует одинакового значения в атрибуте и свойстве.

Для примера, посмотрим, что произойдет с атрибутом "href" при изменении свойства:

<a id="a" href="#"></a>

<script>

a.href = '/';

alert( 'атрибут:' + a.getAttribute('href') ); // '/'

alert( 'свойство:' + a.href ); // полный URL

</script>

Это происходит потому, что атрибут может быть любым, а свойство href, [в соответствии со спецификацией W3C](http://www.w3.org/TR/REC-html40/struct/links.html#adef-href), должно быть полной ссылкой.

Стало быть, если мы хотим именно то, что в HTML, то нужно обращаться через атрибут.

**Есть и другие подобные атрибуты**

Кстати, есть и другие атрибуты, которые не копируются в точности. Например, DOM-свойство input.checked имеет логическое значение true/false, а HTML-атрибут checked – любое строковое, важно лишь его наличие.

Работа с checked через атрибут и свойство:

<input id="input" type="checkbox" checked>

<script>

// работа с checked через атрибут

alert( input.getAttribute('checked') ); // пустая строка

input.removeAttribute('checked'); // снять галочку

// работа с checked через свойство

alert( input.checked ); // false <-- может быть только true/false

input.checked = true; // поставить галочку (при этом атрибут в элементе не появится)

</script>

**[Исходное значение value](http://learn.javascript.ru/attributes-and-custom-properties" \l "ishodnoe-znachenie-value)**

Изменение некоторых свойств обновляет атрибут. Но это скорее исключение, чем правило.

**Чаще синхронизация – односторонняя: свойство зависит от атрибута, но не наоборот.**

Например, при изменении свойства input.value атрибут input.getAttribute('value') не меняется:

<body>

<input id="input" type="text" value="markup">

<script>

input.value = 'new'; // поменяли свойство

alert( input.getAttribute('value') ); // 'markup', не изменилось!

</script>

</body>

То есть, изменение DOM-свойства value на атрибут не влияет, он остаётся таким же.

А вот изменение атрибута обновляет свойство:

<body>

<input id="input" type="text" value="markup">

<script>

input.setAttribute('value', 'new'); // поменяли атрибут

alert( input.value ); // 'new', input.value изменилось!

</script>

</body>

Эту особенность можно красиво использовать.

Получается, что атрибут input.getAttribute('value') хранит оригинальное (исходное) значение даже после того, как пользователь заполнил поле и свойство изменилось.

Например, можно взять изначальное значение из атрибута и сравнить со свойством, чтобы узнать, изменилось ли значение. А при необходимости и перезаписать свойство атрибутом, отменив изменения.

**[Классы в виде строки: className](http://learn.javascript.ru/attributes-and-custom-properties" \l "klassy-v-vide-stroki-classname)**

Атрибуту "class" соответствует свойство className.

Так как слово "class" является зарезервированным словом в Javascript, то при проектировании DOM решили, что соответствующее свойство будет называться className.

Например:

<body class="main page">

<script>

// прочитать класс элемента

alert( document.body.className ); // main page

// поменять класс элемента

document.body.className = "class1 class2";

</script>

</body>

Кстати, есть и другие атрибуты, которые называются иначе, чем свойство. Например, атрибуту for (<label for="...">) соответствует свойство с названием htmlFor.

**[Классы в виде объекта: classList](http://learn.javascript.ru/attributes-and-custom-properties" \l "klassy-v-vide-obekta-classlist)**

Атрибут class – уникален. Ему соответствует аж целых два свойства!

Работать с классами как со строкой неудобно. Поэтому, кроме className, в современных браузерах есть свойство classList.

**Свойство classList – это объект для работы с классами.**

Оно поддерживается в IE начиная с IE10, но его можно эмулировать в IE8+, подключив мини-библиотеку [classList.js](https://github.com/eligrey/classList.js).

Методы classList:

* elem.classList.contains("class") – возвращает true/false, в зависимости от того, есть ли у элемента класс class.
* elem.classList.add/remove("class") – добавляет/удаляет класс class
* elem.classList.toggle("class") – если класса class нет, добавляет его, если есть – удаляет.

Кроме того, можно перебрать классы через for, так как classList – это псевдо-массив.

Например:

<body class="main page">

<script>

var classList = document.body.classList;

classList.remove('page'); // удалить класс

classList.add('post'); // добавить класс

for (var i = 0; i < classList.length; i++) { // перечислить классы

alert( classList[i] ); // main, затем post

}

alert( classList.contains('post') ); // проверить наличие класса

alert( document.body.className ); // main post, тоже работает

</script>

</body>

**[Нестандартные атрибуты](http://learn.javascript.ru/attributes-and-custom-properties" \l "nestandartnye-atributy)**

У каждого элемента есть некоторый набор стандартных свойств, например для <a> это будут href, name, а для <img> это будут src, alt, и так далее.

Точный набор свойств описан в стандарте, обычно мы более-менее представляем, если пользуемся HTML, какие свойства могут быть, а какие – нет.

Для нестандартных атрибутов DOM-свойство не создаётся.

Например:

<div id="elem" href="http://ya.ru" about="Elephant"></div>

<script>

alert( elem.id ); // elem

alert( elem.about ); // undefined

</script>

Свойство является стандартным, только если оно описано в стандарте именно для этого элемента.

То есть, если назначить элементу <img> атрибут href, то свойство img.href от этого не появится. Как, впрочем, и если назначить ссылке <a> атрибут alt:

<img id="img" href="test">

<a id="link" alt="test"></a>

<script>

alert( img.href ); // undefined

alert( link.alt ); // undefined

</script>

Нестандартные атрибуты иногда используют для CSS.

В примере ниже для показа «состояния заказа» используется атрибут order-state:

<style>

.order[order-state="new"] {

color: green;

}

.order[order-state="pending"] {

color: blue;

}

.order[order-state="canceled"] {

color: red;

}

</style>

<div class="order" order-state="new">

Новый заказ.

</div>

<div class="order" order-state="pending">

Ожидающий заказ.

</div>

<div class="order" order-state="canceled">

Заказ отменён.

</div>

Почему именно атрибут? Разве нельзя было сделать классы .order-state-new, .order-state-pending, order-state-canceled?

Конечно можно, но манипулировать атрибутом из JavaScript гораздо проще.

Например, если нужно отменить заказ, неважно в каком он состоянии сейчас – это сделает код:

div.setAttribute('order-state', 'canceled');

Для классов – нужно знать, какой класс у заказа сейчас. И тогда мы можем снять старый класс, и поставить новый:

div.classList.remove('order-state-new');

div.classList.add('order-state-canceled');

…То есть, требуется больше исходной информации и надо написать больше букв. Это менее удобно.

Проще говоря, значение атрибута – произвольная строка, значение класса – это «есть» или «нет», поэтому естественно, что атрибуты «мощнее» и бывают удобнее классов как в JS так и в CSS.

**[Свойство dataset, data-атрибуты](http://learn.javascript.ru/attributes-and-custom-properties" \l "svoystvo-dataset-data-atributy)**

С помощью нестандартных атрибутов можно привязать к элементу данные, которые будут доступны в JavaScript.

Как правило, это делается при помощи атрибутов с названиями, начинающимися на data-, например:

<div id="elem" data-about="Elephant" data-user-location="street">

По улице прошёлся слон. Весьма красив и толст был он.

</div>

<script>

alert( elem.getAttribute('data-about') ); // Elephant

alert( elem.getAttribute('data-user-location') ); // street

</script>

[Стандарт HTML5](http://www.w3.org/TR/2010/WD-html5-20101019/elements.html#embedding-custom-non-visible-data-with-the-data-attributes) специально разрешает атрибуты data-\* и резервирует их для пользовательских данных.

При этом во всех браузерах, кроме IE10-, к таким атрибутам можно обратиться не только как к атрибутам, но и как к свойствам, при помощи специального свойства dataset:

<div id="elem" data-about="Elephant" data-user-location="street">

По улице прошёлся слон. Весьма красив и толст был он.

</div>

<script>

alert( elem.dataset.about ); // Elephant

alert( elem.dataset.userLocation ); // street

</script>

Обратим внимание – название data-user-location трансформировалось в dataset.userLocation. Дефис превращается в большую букву.

**[Полифилл для атрибута hidden](http://learn.javascript.ru/attributes-and-custom-properties" \l "polifill-dlya-atributa-hidden)**

Для старых браузеров современные атрибуты иногда нуждаются в полифилле. Как правило, такой полифилл включает в себя не только JavaScript, но и CSS.

Этот атрибут должен прятать элемент, действие весьма простое, для его поддержки в HTML достаточно такого CSS:

<style>

[hidden] { display: none }

</style>

<div>Текст</div>

<div hidden>С атрибутом hidden</div>

<div id="last">Со свойством hidden</div>

<script>

last.hidden = true;

</script>

Если запустить в IE11- пример выше, то <div hidden> будет скрыт, а вот последний div, которому поставили свойство hidden в JavaScript – по-прежнему виден.

Это потому что CSS «не видит» присвоенное свойство, нужно синхронизировать его в атрибут.

Вот так – уже работает:

<style>

[hidden] { display: none }

</style>

<script>

if (document.documentElement.hidden === undefined) {

Object.defineProperty(Element.prototype, "hidden", {

set: function(value) {

this.setAttribute('hidden', value);

},

get: function() {

return this.getAttribute('hidden');

}

});

}

</script>

<div>Текст</div>

<div hidden>С атрибутом hidden</div>

<div id="last">Со свойством hidden</div>

<script>

last.hidden = true;

</script>

**[«Особенности» IE8](http://learn.javascript.ru/attributes-and-custom-properties" \l "osobennosti-ie8)**

Если вам нужна поддержка этих версий IE – есть пара нюансов.

1. Во-первых, версии IE8- синхронизируют все свойства и атрибуты, а не только стандартные:

document.body.setAttribute('my', 123);

alert( document.body.my ); // 123 в IE8-

При этом даже тип данных не меняется. Атрибут не становится строкой, как ему положено.

1. Ещё одна некорректность IE8-: для изменения класса нужно использовать именно свойство className, вызов setAttribute('class', ...) не сработает.

Вывод из этого довольно прост – чтобы не иметь проблем в IE8, нужно использовать всегда только свойства, кроме тех ситуаций, когда нужны именно атрибуты. Впрочем, это в любом случае хорошая практика.

**[Итого](http://learn.javascript.ru/attributes-and-custom-properties" \l "itogo)**

* Атрибуты – это то, что написано в HTML.
* Свойство – это то, что находится внутри DOM-объекта.

Таблица сравнений для атрибутов и свойств:

| **Свойства** | **Атрибуты** |
| --- | --- |
| Любое значение | Строка |
| Названия регистрозависимы | Не чувствительны к регистру |
| Не видны в innerHTML | Видны в innerHTML |

Синхронизация между атрибутами и свойствами:

* Стандартные свойства и атрибуты синхронизируются: установка атрибута автоматически ставит свойство DOM. Некоторые свойства синхронизируются в обе стороны.
* Бывает так, что свойство не совсем соответствует атрибуту. Например, «логические» свойства вроде checked, selected всегда имеют значение true/false, а в атрибут можно записать произвольную строку.Выше мы видели другие примеры на эту тему, например href.

Нестандартные атрибуты:

* Нестандартный атрибут (если забыть глюки старых IE) никогда не попадёт в свойство, так что для кросс-браузерного доступа к нему нужно обязательно использовать getAttribute.
* Атрибуты, название которых начинается с data-, можно прочитать через dataset. Эта возможность не поддерживается IE10-.

Для того, чтобы избежать проблем со старыми IE, а также для более короткого и понятного кода старайтесь везде использовать свойства, а атрибуты – только там, где это *действительно* нужно.

А *действительно* нужны атрибуты очень редко – лишь в следующих трёх случаях:

1. Когда нужно кросс-браузерно получить нестандартный HTML-атрибут.
2. Когда нужно получить «оригинальное значение» стандартного HTML-атрибута, например, <input value="...">.
3. Когда нужно получить список всех атрибутов, включая пользовательские. Для этого используется коллекция attributes.

Если вы хотите использовать собственные атрибуты в HTML, то помните, что атрибуты с именем, начинающимся на data- валидны в HTML5 и современные браузеры поддерживают доступ к ним через свойство dataset.