**Работа с элементами страницы в JavaScript**

*В данном уроке мы разберем работу с элементами страницы на языке JavaScript.*

**Работа с innerHTML, outerHTML**

В прошлом уроке мы с вами учились считывать и записывать значения атрибутов, а теперь будем делать тоже самое, но уже со **внутренним содержимым** тегов. Это делается с помощью свойства **innerHTML**, которое можно считывать и перезаписывать, меняя тем самым содержимое тегов.

В примере ниже у нас есть абзац. Давайте сменим ему текст с помощью свойства **innerHTML**. Для этого получим ссылку на этот абзац с помощью **getElementById** в переменную **elem**. Затем таким образом - *elem.innerHTML = 'Новый текст'* - запишем в него новый текст:

<p id="test">Текст абзаца.</p>

<input type="submit" onclick="func()">

//Функция func срабатывает по клику на кнопку input:

function func() {

var elem **=** document.getElementById('test'); //получаем наш абзац

elem.innerHTML **=** '!'; //записываем в него новый текст

}

HTML код станет выглядеть так:

<p id="test">!</p>

<input type="submit" onclick="func()">

Можно записывать не только текст, но и **теги** и они будут работать (в нашем случае текст станет жирным):

<p id="test">Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

elem.innerHTML **=** '<b>!</b>';

}

HTML код станет выглядеть так:

<p id="test"><b>!</b></p>

<input type="submit" onclick="func()">

Кроме свойства **innerHTML** существует также свойство **outerHTML**, которое перезаписывает не только внутренний текст тега, но и **сам тег**. Смотрите пример:

<p id="test">Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

elem.outerHTML **=** '<b>!</b>';

}

HTML код станет выглядеть так:

<b>!</b>

<input type="submit" onclick="func()">

Как вы видите, наш **абзац исчез** и заменился на тег b.

**Работа с getElementsByTagName**

Во всех примерах выше мы с вами работали с методом **getElementById**, который получал только одно свойство по его атрибуту **id**. Однако существуют и другие способы получения свойств, к примеру, с помощью метода **getElementsByTagName**, который получает группу тегов по их имени. Например, можно получить все абзацы **p** или все заголовки **h2**.

Учтите только один нюанс - наш метод возвращает не одно свойство, а **массив** свойств. Получим, к примеру, все абзацы в переменную **elems**:

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elems **=** document.getElementsByTagName('p');

}

В переменной **elems** теперь лежит массив свойств, это значит, что **elems[0]** - это первый абзац, **elems[1]** - второй и так далее (нумерация с нуля).

Мы можем убедится в этом, если сделаем так: *elems[0].innerHTML = '!'* - в этом случае **первый** абзац сменит свой текст.

Давайте поменяем текст всех трех абзацев:

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

//Получаем массив абзацев в переменную elems:

var elems **=** document.getElementsByTagName('p');

elems[0].innerHTML **=** '1'; //первому абзацу ставим текст '1'

elems[1].innerHTML **=** '2'; //второму абзацу ставим текст '2'

elems[2].innerHTML **=** '3'; //третьему абзацу ставим текст '3'

}

HTML код станет выглядеть так:

<p>1</p>

<p>2</p>

<p>3</p>

<input type="submit" onclick="func()">

Учтите, что менять одновременно все свойства как-нибудь так *elems.innerHTML = '!'* не получится (потому что **elems** - массив). Необходимо или вручную перебрать все свойства, как это сделано в примере выше, или воспользоваться **циклом**.

В следующем примере мы сменим содержимое всех абзацев на страницы на '!', воспользовавшись для этого циклом **for**:

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elems **=** document.getElementsByTagName('p');

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

elems[i].innerHTML **=** '!';

}

}

HTML код станет выглядеть так:

<p>!</p>

<p>!</p>

<p>!</p>

<input type="submit" onclick="func()">

**Обращение к свойствам через свойства document**

Некоторые теги страницы можно не получать методами типа **getElementById**, а обратиться к ним как свойствам объекта **document**. К примеру, к тегу **<body>** можно обратиться так: **document.body**, а к тегу **<head>** - вот так: **document.head**. Такое работает не для всех тегов, а только для избранных.

Учтите, что нельзя получить доступ к тому свойству, которого еще не существует в момент выполнения скрипта. Поэтому, если скрипт находится в **<head>**, то в нем недоступен **document.body**.

В следующем примере первый alert выведет **null**:

<!DOCTYPE HTML>

<html>

<head>

<script>

alert(document.body); //null, так как body еще нет

</script>

</head>

<body>

<script>

alert(document.body); //body уже есть

</script>

</body>

</html>

**Работа с формами**

Можно получить все [HTML формы](http://code.mu/html/form.html) на странице через **document.forms**. В результате мы получим массив форм, будто он был получен с помощью [getElementsByTagName](http://code.mu/javascript/dom/getElementsByTagName.html) (точнее это будет не массив, а [псевдомассив (или коллекция)](http://code.mu/javascript/pages/pseudo-array.html) свойств).

Мы можем, к примеру, обратиться к любой форме, как к энному свойству массива, например, к форме с номером 0:

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

</body>

document.forms[0].innerHTML **=** '!';

HTML код станет выглядеть так:

<body>

<form>

!

</form>

<form>

<input type="text" value="1">

<input type="text" value="2">

<input type="text" value="2">

</form>

</body>

А можем перебрать все формы с помощью цикла:

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

</body>

var forms **=** document.forms;

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

forms[i].innerHTML **=** '!';

}

HTML код станет выглядеть так:

<body>

<form>

!

</form>

<form>

!

</form>

</body>

Следующем образом - **document.forms[i].elements** - можно найти общее количество форм на странице.

Кроме того, к формам можно обращаться по имени вместо номера. Это имя задается в атрибуте **name** тега **<form>**. Например, если форме дать имя test, то к ней можно обратиться так - **document.forms['test']** или так - **document.forms.test**.

В следующем примере форме с именем **form1** поставим текст '!', а форме с именем **form2** - текст '?' (двумя разными способами):

<body>

<form name="form1">

<input value="1">

<input value="2">

<input value="3">

</form>

<form name="form2">

<input value="1">

<input value="2">

<input value="3">

</form>

</body>

document.forms.form1.innerHTML **=** '!';

document.forms['form2'].innerHTML **=** '?';

HTML код станет выглядеть так:

<body>

<form name="form1">

!

</form>

<form name="form2">

?

</form>

</body>

**Обращение к свойствам форм**

У каждой формы есть свойство **elements**, которое содержит массив свойств формы. С его помощью можно обратиться к определенному свойству формы по его номеру.

Для примера давайте обратимся к форме номер 1, а затем к ее свойству номер 0:

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

</body>

document.forms[1].elements[0].value **=** '!';

HTML код станет выглядеть так:

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="!">

<input value="2">

<input value="3">

</form>

</body>

Можно также перебрать все свойства формы с помощью цикла. Давайте получим форму с номером 1 и всем свойствам этой формы поставим **value** '!':

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

</body>

var elements **=** document.forms[1].elements;

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

elements[i] **=** '!';

}

HTML код станет выглядеть так:

<body>

<form>

<input value="1">

<input value="2">

<input value="3">

</form>

<form>

<input value="!">

<input value="!">

<input value="!">

</form>

</body>

При необходимости с помощью двух вложенных циклов можно перебрать все формы и все свойства внутри них.

К свойствам формы, так же, как и к самим формам, можно обращаться по имени из атрибута **name**:

<body>

<form name="form1">

<input value="1" name="input1">

<input value="2" name="input2">

<input value="3" name="input3">

</form>

</body>

document.forms.form1.elements.input1.value **=** '!';

HTML код станет выглядеть так:

<body>

<form name="form1">

<input value="!" name="input1">

<input value="2" name="input2">

<input value="3" name="input3">

</form>

</body>

Это не все приемы работы с формами, при необходимости смотрите подробности в справочнике JavaScript тут: [работа с формами](http://code.mu/books/javascript/dom/raznye-poleznye-temy-javascript.html).

**Продвинутая работа с атрибутами**

Вы уже знаете один из способов работы с атрибутами тегов, однако есть **альтернативный** вариант - с помощью методов **getAttribute**, **setAttribute**, **removeAttribute** и **hasAttribute**.

Метод **getAttribute** считывает значение указанного атрибута (и атрибут **class** тут уже не будет исключением), **setAttribute** - записывает новое значение в атрибут,**removeAttribute** удаляет атрибут совсем, а **hasAttribute** проверяет наличие или отсутствие атрибута у свойства.

Давайте получим значение атрибута **value** для инпута с помощью **getAttribute**:

<input type="text" value="!" id="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

alert(elem.getAttribute('value'));

}

А теперь запишем новое значение 'www' в атрибут **value**:

<input type="text" value="!" id="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

elem.setAttribute('value', 'www');

}

Удалим атрибут **value** совсем:

<input type="text" value="!" id="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

elem.removeAttribute('value');

}

HTML код станет выглядеть так (исчезнет атрибут value):

<input type="text" id="test">

<input type="submit" onclick="func()">

Ну, а теперь проверим наличие атрибута **value** у нашего свойства:

<input type="text" value="!" id="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('test');

alert(elem.hasAttribute('value')); //выведет true

}

**Получение свойств по их классу**

Вы уже знаете метод **getElementsByTagName**, который получает **группу** свойств по названию тега. Существует **аналогичный** метод **getElementsByClassName**, который получает свойства по их **классу** (то есть по содержимому [атрибута class](http://code.mu/attr/class.html)).

Так же, как и для других подобных методов, **нельзя** обратиться ко всем свойствам **одновременно**, нужно использовать **цикл**. В примере ниже мы получаем все свойства с классом **www** и ставим им текст '!':

<p class="www">Текст абзаца.</p>

<p class="www">Текст абзаца.</p>

<p class="www">Текст абзаца.</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elems **=** document.getElementsByClassName('www');

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

elems[i].innerHTML **=** '!';

}

}

HTML код станет выглядеть так:

<p class="www">!</p>

<p class="www">!</p>

<p class="www">!</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

**Получение свойств селектором CSS**

В настоящее время в JavaScript появились более **продвинутые** методы получения свойств, в которых можно использовать любые [селекторы CSS](http://code.mu/css/pages/selectors.html).

Для этого можно использовать два метода: **querySelector** и **querySelectorAll**. Метод **querySelector** является аналогом **getElementById**, только принимает не **id** свойства, а любой **селектор**, а возвращает ссылку на **один** найденный свойство (если под селектор попадает много свойств, то она вернет ссылку на первый свойство).

В примере ниже с помощью **querySelector** мы получаем инпут с классом **test** и выводим на экран содержимое его атрибута **value**:

<input type="text" value="!" class="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.querySelector('input.test');

alert(elem.getAttribute('value'));

}

А сейчас мы получим свойство с **id** равным **test**:

<input type="text" value="!" id="test">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.querySelector('#test');

alert(elem.getAttribute('value'));

}

Давайте теперь разберем **querySelectorAll**. Этот метод, в отличии от **querySelector**, получает **группу** свойств и работать с ними нужно как с группой (перебирать **циклом** и т.п.). В примере ниже мы получаем все свойства с классом **www** и ставим им текст '!':

<p class="www">Текст абзаца.</p>

<p class="www">Текст абзаца.</p>

<p class="www">Текст абзаца.</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

function func() {

var elems **=** document.querySelectorAll('.www');

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

elems[i].innerHTML **=** '!';

}

}

HTML код станет выглядеть так:

<p class="www">!</p>

<p class="www">!</p>

<p class="www">!</p>

<p>Текст абзаца.</p>

<input type="submit" onclick="func()">

**Атрибуты data-**

В языке HTML5 разрешено добавлять свои атрибуты тегам, при этом они должны начинаться с **data-**, а затем должно идти любое название атрибута, которое вам удобно (это будет работать корректно во всех браузерах, кроме IE10 и ниже).

Если вы хотите обратиться к таким атрибутам как с свойствам объекта, то это делается не на прямую, а при помощи специального свойства **dataset**:

<div id="elem" data-price="1000" data-product-number="5">

Товар Джинсы

</div>

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('elem');

alert(elem.dataset.price); //выведет 1000

alert(elem.dataset.productNumber); //выведет 5

}

Обратите внимание – атрибут **data-price** превратился в **dataset.price**, а **data-product-number** превратился в **productNumber**.

К таким атрибутам можно также обращаться с помощью методов типа **getAttribute**, в этом случае следует писать полное название атрибута:

<div id="elem" data-price="1000" data-product-number="5">

Товар Джинсы

</div>

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('elem');

alert(elem.getAttribute('data-price'); //выведет 1000

alert(elem.getAttribute('data-product-number'); //выведет 5

}

**Разница между способами получения атрибутов**

На самом деле работа с атрибутами через свойства (так: **elem.value**) и через getAttribute (так: **elem.getAttribute('value')**) - это не одно и то же.

При изменении свойства **elem.value** атрибут **elem.getAttribute('value')** не меняется:

<input id="test" type="text" value="old">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('elem');

elem.value **=** 'new'; //поменяли свойство

alert(elem.getAttribute('value')); //выведет 'old' - не изменилось!

}

То есть, изменение свойства на атрибут не влияет, он остается таким же.

А вот изменение атрибута обновляет свойство:

<input id="test" type="text" value="old">

<input type="submit" onclick="func()">

function func() {

var elem **=** document.getElementById('elem');

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

alert(elem.value); //выведет 'new' - elem.value изменилось!

}

Эту особенность можно использовать.

Получается, что атрибут **elem.getAttribute('value')** хранит 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.

<http://code.mu/books/javascript/dom/rabota-s-elementami-stranicy-v-javascript.html>