***Метод getElementById***

Метод getElementById позволяет получить элемент (тег) страницы по его атрибуту id.

С полученным элементом можно будет производить различные манипуляции: менять его текст, атрибуты, css стили и так далее.

См. также методы getElementsByTagName, getElementsByClassName, querySelector, querySelectorAll, которые тоже позволяют получить элементы страницы.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

document.getElementById(ай ди элемента);

В данном примере по нажатию на кнопку будет получен элемент с id, равным elem, и для него будет изменено содержимое атрибута value:

<input type="submit" onclick="func()">  
<input type="text" id="elem">  
function func() {  
 var elem = document.getElementById('elem');  
 elem.value = 'Привет!'; }

HTML код станет выглядеть так:

<input type="submit" onclick="func()">  
<input value="Привет!" type="text" id="elem">

***Метод getElementsByTagName***

Метод getElementsByTagName позволяет получить элементы страницы по имени тега.

С полученными элементами можно производить различные манипуляции: менять их текст, атрибуты, CSS стили и так далее.

Метод возвращает псевдомассив элементов, а не один элемент, поэтому по этому набору сначала нужно пройтись циклом и для каждого элемента сделать то, что вам нужно.

См. также методы getElementById, getElementsByClassName, querySelector, querySelectorAll, которые тоже позволяют получить элементы страницы.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

document.getElementsByTagName(имя тега);

В данном примере по нажатию на кнопку будут получены все абзацы и для них сменен текст с помощью innerHTML:

<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 = i+1;  
 } }

HTML код станет выглядеть так:

<p>1</p> <p>2</p> <p>3</p>

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

Учтите, что менять одновременно все элементы как-нибудь так elems.innerHTML = '!!!' не получится (потому что elems - массив).

Необходимо или или воспользоваться циклом или вручную перебрать элементы таким образом: elems[0].innerHTML = 'новый текст';

Давайте вручную 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()">

***Метод getElementsByClassName***

Метод getElementsByClassName позволяет получить элементы страницы по их классу в атрибуте class.

С полученными элементами можно будет производить различные манипуляции: менять их текст, атрибуты, css стили и так далее.

Метод возвращает массив элементов, а не один элемент, поэтому по этому набору сначала нужно пройтись циклом и для каждого элемента сделать то, что вам нужно.

См. методы getElementById, getElementsByTagName, querySelector, querySelectorAll, которые также позволяют получить элементы страницы.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

document.getElementsByClassName(имя класса);

В данном примере мы получаем все элементы с классом www и ставим им текст '!!!' с помощью innerHTML:

<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()">

***Метод querySelector***

Метод querySelector позволяет получить элемент (тег) страницы по произвольному селектору CSS.

Чтобы получить элемент с заданным id, нужно написать #его\_id, с заданным классом - .его\_класс и так далее - любой селектор.

С полученным элементом можно будет производить различные манипуляции: менять его текст, атрибуты, css стили и так далее.

См. также методы getElementById, getElementsByTagName, getElementsByClassName, querySelectorAll, которые тоже позволяют получить элементы страницы.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

document.querySelector(селектор);

Давайте получим абзац #elem и поменяем ему текст с помощью innerHTML:

<p id="elem">Абзац</p>

var elem = document.querySelector('#elem');  
elem.innerHTML = '!!!';

HTML код станет выглядеть так:

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

Давайте получим абзац с классом elem и поменяем ему текст с помощью innerHTML:

<p class="elem">Абзац</p>  
var elem = document.querySelector('.elem');  
elem.innerHTML = '!!!';

HTML код станет выглядеть так:

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

**Метод querySelectorAll**

Метод querySelectorAll позволяет получить элементы страницы по произвольному селектору CSS.

С полученными элементами можно будет производить различные манипуляции: менять их текст, атрибуты, css стили и так далее.

Метод возвращает псевдомассив элементов, а не один элемент, поэтому по этому набору сначала нужно пройтись циклом и для каждого элемента сделать то, что вам нужно.

См. методы getElementById, getElementsByTagName, getElementsByClassName, querySelector, которые также позволяют получить элементы страницы.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

document.querySelectorAll(селектор);

В данном примере мы получаем все элементы с классом www и ставим им текст '!!!' с помощью innerHTML (последний абзац останется нетронутым, так как у него нет класса www):

<p class="www">Текст абзаца.</p> <p class="www">Текст абзаца.</p>   
<p class="www">Текст абзаца.</p> <p>Текст абзаца.</p>

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>

**Метод matches**

Метод matches позволяет проверить удовлетворяет ли элемент указанному CSS селектору.

См. также методы closest, contains, которые осуществляют похожие операции.

элемент.matches('сss селектор');

Проверим, является ли наш элемент абзацем с классом www:

<p id="elem" class="www">Абзац</p>  
var elem = document.getElementById('elem');  
document.write(elem.matches('p.www'));

Результат выполнения кода: true

***Метод closest***

Метод closest ищет ближайший родительский элемент, подходящий под указанный CSS селектор (сам элемент тоже включается в поиск).

См. также методы matches, contains, которые осуществляют похожие операции.

элемент.closest('сss селектор');

Дан элемент #elem. Давайте найдем среди его родителей элемент с классом www и выведем на экран содержимое его атрибута id:

<div id="parent" class="www">  
 <div id="block">  
 <p id="elem">Абзац</p>  
 </div>  
</div>

var elem = document.getElementById('elem');  
var parent = elem.closest('.www')  
document.write(parent.id);

Результат выполнения кода: parent

***Метод contains***

Метод contains позволяет проверить, содержит ли один элемент внутри себя другой.

Параметром метода передается элемент, который будет проверяться на то, что он находится внутри элемента, к которому применился метод.

См. также методы matches, closest, которые осуществляют похожие операции.

родитель.contains(элемент);

Проверим, находится ли абзац #elem в блоке #parent:

<div id="parent"> <p id="elem"></p> </div>

var parent = document.getElementById('parent');  
var elem = document.getElementById('elem');  
document.write(parent.contains(child));

Результат выполнения кода: true

А теперь в элементе нет переданного элемента и поэтому метод возвращает false:

<div id="parent"></div>  
<p id="elem"></p>  
var parent = document.getElementById('parent');  
var elem = document.getElementById('elem');  
document.write(parent.contains(elem));

Результат выполнения кода: false

Можно передать тот же элемент, на котором метод был вызван, в этом случае метод тоже вернет true:

<div id="parent"></div>  
var parent = document.getElementById('parent');  
document.write(parent.contains(parent));

Результат выполнения кода: true

***Объект document***

Объект document содержит методы и свойства для взаимодействие со страницей. Через document можно получить любой элемент страницы, после чего с ним можно делать все что угодно: менять содержимое, менять css-свойства, удалять, клонировать и др.

Из document можно получить 3 основных элемента напрямую:

var html = document.documentElement //<html>  
var head = document.head //<head>  
var body = document.body //<body>

Для получения остальных элементов нужно использовать методы: getElementById,getElementsByClassName, getElementsByTagName, querySelector, querySelectorAll.

***Свойство innerHTML***

Свойство innerHTML позволяет получить и изменить текст элемента.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

элемент.innerHTML

В данном примере у нас есть абзац. Давайте выведем его текст с помощью свойства innerHTML.

Для этого получим ссылку на этот абзац с помощью getElementById в переменную elem. Затем таким образом - elem.innerHTML - получим доступ к его тексту и выведем его на экран:

<p id="elem">Текст абзаца.</p>  
var elem = document.getElementById('elem');  
document.write(elem.innerHTML);

Результат выполнения кода: Текст абзаца.

Можно не только считывать текст, но и записывать новый:

<p id="elem">Текст абзаца.</p>  
var elem = document.getElementById('elem');  
elem.innerHTML = '!!!';

HTML код станет выглядеть так:

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

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

<p id="elem">Текст абзаца.</p>  
var elem = document.getElementById('elem');  
elem.innerHTML = '<b>!!!</b>';

HTML код станет выглядеть так:

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

***Метод getAttribute***

Метод getAttribute считывает значение заданного атрибута.  
См. также методы setAttribute, removeAttribute, hasAttribute.  
См. также этот и этот уроки из учебника JavaScript для более полного понимания.

элемент.getAttribute(имя атрибута);

В данном примере по нажатию на кнопку будет получен элемент с id, равным elem, и для него будет получено содержимое атрибута value с помощью метода getAttribute:

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

<input value="Привет!" type="text" id="elem">  
function func() {  
 var elem = document.getElementById('elem');  
 document.write(elem.getAttribute('value')); }

Результат выполнения кода: Привет!

***Метод setAttribute***

Метод setAttribute изменяет значение заданного атрибута.

См. также методы getAttribute, removeAttribute, hasAttribute.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

элемент.setAttribute(имя атрибута, новое значение атрибута);

В данном примере у элемента #elem атрибут value будет установлен в значение '!!!' с помощью метода setAttribute:

<input value="Привет!" id="elem">  
var elem = document.getElementById('elem');  
elem.setAttribute('value', '!!!');

HTML код станет выглядеть так (исчезнет атрибут value):

<input value="!!!" id="elem">

***Метод removeAttribute***

Метод removeAttribute удаляет заданный атрибут у элемента.

См. также методы getAttribute, setAttribute, hasAttribute.

См. также этот и этот уроки из учебника JavaScript для более полного понимания.

элемент.removeAttribute(имя атрибута);

В данном примере у элемента #elem будет удален атрибут value с помощью метода removeAttribute:

<input value="Привет!" id="elem">  
var elem = document.getElementById('elem');  
elem.removeAttribute('value');

HTML код станет выглядеть так (исчезнет атрибут value):

<input id="elem">

***Метод hasAttribute***

Метод hasAttribute проверяет наличие заданного атрибута у элемента (тега).  
Если атрибут есть - выведет true, если нет - выведет false.  
См. также методы getAttribute, setAttribute, removeAttribute.  
См. также этот и этот уроки из учебника JavaScript для более полного понимания.

элемент.hasAttribute(имя атрибута);

В данном примере по нажатию проверяется наличие атрибута value (он есть) с помощью метода hasAttribute:

<input value="Привет!" id="elem">  
var elem = document.getElementById('elem');  
document.write(elem.hasAttribute('value'));

Результат выполнения кода: true

А теперь атрибута value нету:

<input id="elem">  
var elem = document.getElementById('elem');  
document.write(elem.hasAttribute('value'));

Результат выполнения кода: false

Свойство classList

Свойство classList содержит псевдомассив CSS классов элемента, а также позволяет добавлять и удалять классы элемента, проверять наличие определенного класса среди классов элемента.

Речь идет об атрибуте class, внутри которого можно писать несколько классов через пробел, например 'www ggg zzz'. С помощью classList можно удалить, к примеру, класс ggg, не затронув остальные классы.

У classList есть набор методов, которые можно использовать для работы с классами. Давайте их разберем.

Метод add добавляет указанный класс:

элемент.classList.add('класс');

Метод remove удаляет указанный класс:

элемент.classList.remove('класс');

Метод contains проверяет наличие указанного класса (возвращает true, если такой класс есть у элемента, false - если нет):

элемент.classList.contains('класс');

***Метод toggle***

Метод toggle добавляет класс, если его нет, удаляет если есть:

элемент.classList.toggle('класс');

Сам classList это псевдомассив классов элемента (их можно, к примеру, перебрать циклом):

элемент.classList

Так можно найти количество классов элемента:

элемент.classList.length

Пример . Добавление класса

Добавим элементу класс kkk:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
elem.classList.add('kkk');

HTML код станет выглядеть так:

<p id="elem" class="www ggg zzz kkk"></p>

Пример . Добавление существующего класса

Добавим элементу класс zzz, который уже есть в элементе - ничего не произойдет, так как дубли классов не добавляются:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
elem.classList.add('zzz');

HTML код станет выглядеть так (ничего не поменяется):

<p id="elem" class="www ggg zzz"></p>

Пример . Удаление класса

Удалим класс ggg:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
elem.classList.remove('ggg');

HTML код станет выглядеть так:

<p id="elem" class="www zzz"></p>

Пример . Проверка класса

Проверим, есть ли у элемента класс ggg:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
var checkClass = elem.classList.contains('ggg');  
document.write(checkClass);

Результат выполнения кода: true

Пример . Количество классов

Узнаем количество классов элемента:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
var length = elem.classList.length;  
document.write(length);

Результат выполнения кода: 3

Пример . Перебираем классы

Выведем столбец классов элемента:

<p id="elem" class="www ggg zzz"></p>  
var elem = document.getElementById('elem');  
var classNames = elem.classList;  
for (var i = 0; i < classNames.length; i++) {  
document.write(classNames[i] + '<br>'); }

Результат выполнения кода:  
www  
ggg  
zzz

Давайте сделаем кнопку, по нажатию на которую элемент будет то показываться, то скрываться. Пусть по умолчанию элемент скрыт (это реализуем с помощью display: none), а покажется он с помощью добавления класса active. Этот класс будем то добавлять, то убирать с помощью classList.toggle:

#elem { display: none;  
 width: 200px;  
 height: 200px;  
 border: 1px solid green; }  
#elem.active { display: block; }  
<button onclick="toggle()">Нажми на меня</button>  
<div id="elem"></div>  
function toggle() {  
 var elem = document.getElementById('elem');  
 classList.toggle('active'); }

***Метод dataset***

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

***Свойство cssText***

Свойство cssText позволяет задать CSS стили массово одной строкой. При этом все содержимое атрибута style перезаписывается.

Пожалуйста, не злоупотребляйте этим свойством. А то все, как узнают про него - начинают сувать везде. Но это ведет к ошибкам - ведь cssText затирает все из атрибута style, это значит, что все ранее установленные через JavaScript стили CSS просто затрутся.

элемент.style.cssText = 'свойство 1: значение; свойство 2: значение...'

Давайте зададим элементу несколько стилей:

<p id="elem"></p>  
var elem = document.getElementById('elem');  
elem.style.cssText = 'color: red; font-size: 20px;';

В этом примере у элемента изначально уже будут стили в атрибуте style, но свойство cssText перезапишет его:

<p id="elem" style="background: red;"></p>  
var elem = document.getElementById('elem');  
elem.style.cssText = 'color: red; font-size: 20px;';

HTML код станет выглядеть так:

<p id="elem" style="color: red; font-size: 20px;"></p>

Чтобы предыдущие стили не перезаписывались, можно сделать так:

<p id="elem" style="background: green;"></p>  
var elem = document.getElementById('elem');  
elem.style.cssText += 'color: red; font-size: 20px;';

HTML код станет выглядеть так:

<p id="elem" style="background: green; color: red; font-size: 20px;"></p>

***Функция getComputedStyle***

Функция getComputedStyle позволяет получить значение любого CSS свойства элемента, даже из CSS файла.

Для более полного понимания см. урок работа с метриками на JavaScript из учебника JavaScript.

var объект = getComputedStyle(элемент);

Подробности Как она работает (внимание: не так как мы ожидаем): параметром функция принимает элемент, а возвращает объект, который содержит в себе все CSS свойства переданного элемента.

Давайте положим этот объект в переменную style. Название произвольное, это просто переменная - как придумаем, так и будем обращаться:

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem); //в style лежат CSS свойства

Давайте выведем, к примеру, ширину. Это делается так - style.width:

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem);  
alert(style.width);

Чтобы вывести, к примеру, левый padding - делаем так - style.paddingLeft:

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem);  
alert(style.paddingLeft);

В следующем примере мы выведем все интересующие нас CSS свойства для нашего элемента:

<div id="elem">Текст</div>

#elem { width: 200px;  
 height: 200px;  
 padding: 30px;  
 border: 20px solid #BCBCBC;  
 overflow: auto; }

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem);  
alert('paddingLeft: ' + style.paddingLeft); //выведет '30px'  
alert('borderTopWidth: ' + style.borderTopWidth); //выведет '20px'  
alert('borderTopStyle: ' + style.borderTopStyle); //выведет 'solid'

Нажмите на кнопку - и вы увидите результат:

Lorem ipsum dolorsit amet consectetur adipiscing elit. Sed viverra mollis lorem, fringilla dapibus nisi commodo interdum. Vivamus in turpis quis purus dapibus aliquam id nec velit. In at aliquet sem, a rutrum neque. Vestibulum posuere lobortis accumsan. Etiam non tincidunt erat, vel condimentum turpis.

***Неточность getComputedStyle***

Иногда функция getComputedStyle работает не совсем корректно с шириной и высотой. Это связано с тем, что padding и граница расширяют блок. В следующем примере блоку задана ширина 200px, а также граница и padding. Реальная ширина блока 300px, но getComputedStyle все равно выведет 200px:

#elem { width: 200px;  
 height: 200px;  
 padding: 30px;  
 border: 20px solid #BCBCBC;  
 overflow: auto; }

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem);  
alert('width: ' + style.width); //выведет '200px'  
alert('height: ' + style.height); //выведет '200px'

Нажмите на кнопку - и вы увидите результат:

Lorem ipsum dolorsit amet consectetur adipiscing elit. Sed viverra mollis lorem, fringilla dapibus nisi commodo interdum. Vivamus in turpis quis purus dapibus aliquam id nec velit. In at aliquet sem, a rutrum neque. Vestibulum posuere lobortis accumsan. Etiam non tincidunt erat, vel condimentum turpis.

То есть: получается, что getComputedStyle игнорирует расширение блока показывает его размеры так, как будто этого расширения не было.

Но это еще не все: также имеет значение наличие или отсутствие полосы прокрутки - некоторые браузеры отнимают ширину полосы прокрутки от ширины, вычисленной через getComputedStyle, а некоторые не отнимают. В общем тут вообще все не кроссбраузерно и лучше getComputedStyle для определения ширины и высоты не использовать, а пользоваться метриками, которые мы изучим чуть ниже.

Вычисленные значения

Есть еще один нюанс: если ширина задана в % - то после работы getComputedStyle мы увидим ее в px. То есть по сути мы получаем не заданную ширину, а вычисленную. Смотрите на следующем примере:

#elem {  
 width: 30%; /\* ширина задана в % \*/  
 height: 200px;  
 padding: 30px;  
 border: 20px solid #BCBCBC;  
 overflow: auto; }

var elem = document.getElementById('elem');  
var style = getComputedStyle(elem);  
alert('width: ' + style.width);

Нажмите на кнопку - и вы увидите ширину в пикселях, а не в %: