

Свойства узлов

- Классы DOM-узлов
- Главные свойства узлов
- Свойства, зависящие от класса
- Пользовательские свойства

Классы DOM-узлов

У разных DOM-узлов могут быть **разные свойства**. Например, у узла, соответствующего тегу **a**, есть свойства, связанные со ссылками, у узла, соответствующего тегу **input** – свойства, связанные с полем ввода и т.д.

Текстовые узлы отличаются от узлов-элементов. Но у них есть общие свойства и методы, потому что все классы DOM-узлов образуют единую иерархию.

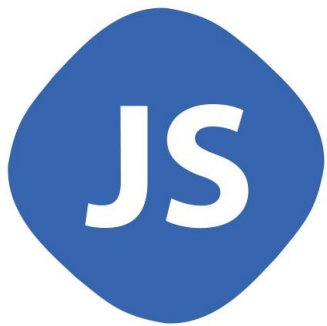
Важно. Все узлы объединяет набор **общих** свойств и методов.

Каждый DOM-узел принадлежит соответствующему встроенному классу.

Существуют следующие классы:

- **EventTarget** – это корневой абстрактный класс.
- **Node** – также является абстрактным классом, и **служит основой для DOM-узлов**. Он обеспечивает базовую функциональность: `parentNode`, `nextSibling`, `childNodes` и т.д.
- **Element** – это **базовый класс для DOM-элементов**. Он обеспечивает навигацию на уровне элементов: `nextElementSibling`, `children` и методы поиска: `getElementsByTagName`, `querySelector` (в следующих уроках).
- **HTMLElement** – является **базовым классом для всех остальных HTML-элементов**. От него наследуют конкретные элементы:
 - **HTMLInputElement** – класс для тега **input**,
 - **HTMLBodyElement** – класс для тега **body**,





- **HTMLAnchorElement** – класс для тега **a**,
- ...и т.д, **каждому тегу соответствует свой класс, который предоставляет определённые свойства и методы.**

Примечание. Если вы не знакомы с концепцией классов и объектов в программировании, то подробности иерархии классов пока можно просто прочитать.

Для того, чтобы узнать имя класса DOM-узла, необходимо воспользоваться свойством **constructor**. Оно ссылается на конструктор класса, и в свойстве **constructor.name** содержится его имя.

example_1. Демонстрация имен классов узлов

```
<script>

    // выбираем узлы документа

    let header = document.body.firstChild;

    let div = header.nextElementSibling;

    let p = div.nextElementSibling;

    let span = p.nextElementSibling;

    let txt = span.nextSibling;

    let comment = txt.nextSibling;

    let script = comment.nextElementSibling;

    // выводим имена классов узлов

    console.log(header.constructor.name);

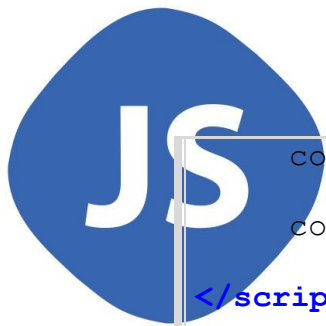
    console.log(div.constructor.name);

    console.log(p.constructor.name);

    console.log(span.constructor.name);

    console.log(txt.constructor.name);
```





```
console.log(comment.constructor.name) ;  
console.log(script.constructor.name) ;  
</script>
```

Встроенные свойства узлов

Самое главное различие между DOM-узлами – разные узлы являются **объектами различных классов**. Поэтому, к примеру, узел, соответствующий тегу **td** имеет один набор свойств, узел **form** – другой, узел тега **a** – третий.

Рассмотрим главные свойства узлов DOM, которые используются наиболее часто.

Свойство	Описание
data	Возвращает значение "текстового" узла и "комментария"
innerText	Задаёт или возвращает внутренний текст элемента
outerText	Задаёт или возвращает внешний текст элемента
textContent	Устанавливает или возвращает текстовое содержимое узла и его потомков
innerHTML	Задаёт или возвращает содержимое элемента
outerHTML	Возвращает HTML элемент целиком
nodeName	Возвращает имя указанного узла
nodeType	Возвращает тип узла узла
nodeValue	Задаёт или возвращает значение узла
tagName	Возвращает имя тега указанного элемента

Свойства, предназначенные для работы с текстовым содержимым элемента

innerText





Свойство **innerText** позволяет получить или установить внутренний **текст** узла-элемента.

Синтаксис

```
element.innerText; // получить внутренний текст узла
```

```
element.innerText = "new text"; // установить внутренний текст узла
```

example_2. Свойство innerText

```
<script>

    // ссылка на узел body

    let body = document.body;

    // получили доступ к элементу header

    let header = body.firstChild;

    // далее спускаемся вниз по дереву узлов

    let div = header.nextElementSibling.nextElementSibling;

    let p = div.firstChild;

    // выводим значение html-узла p

    console.log(p.innerText);

    // изменим значение узла

    p.innerText = "Очень опасно встретить женщину, которая полностью тебя понимает. Это обычно кончается женитьбой.";

</script>
```

outerText

Свойство **outerText** устанавливает или возвращает **текстовое** содержимое указанного узла. Это свойство похоже на внутреннее свойство **innerText**, фактически **получение** внешнего текста возвращает тот же результат, что и получение свойства **innerText**.





Есть важное отличие при **установке** внешнего текста элемента, потому что сам выбранный элемент удаляется.

```
element.outerText; // получить текстовое содержимое узла
```

```
element.outerText = "new text"; // установить текстовое  
содержимое (заменяв весь узел)
```

Наглядно отличие **innerText** от **outerText** продемонстрировано в примере example_3.

example_3. Свойство outerText

```
<script>  
  
  // ссылка на узел body  
  
  let body = document.body;  
  
  // получили доступ к элементу header  
  
  let header = body.firstChild;  
  
  // далее спускаемся вниз по дереву узлов  
  
  let div = header.nextElementSibling.nextElementSibling;  
  
  let p = div.firstChild;  
  
  // выводим значение html-узла p  
  
  console.log(p.outerText);  
  
  // изменим значение узла используя innerText  
  
  p.innerText = "<b>Очень опасно встретить женщину, которая  
  полностью тебя понимает. Это обычно кончается женитьбой.</b>";  
  
  // изменим значение узла используя outerText  
  
  // p.outerText = "<b>Очень опасно встретить женщину, которая  
  полностью тебя понимает. Это обычно кончается женитьбой.</b>";  
  
</script>
```

На рисунке видно, что при использовании свойства **outerText** сам тег **p** был заменен новым **текстовым узлом**. При использовании **innerText** такого не происходит.



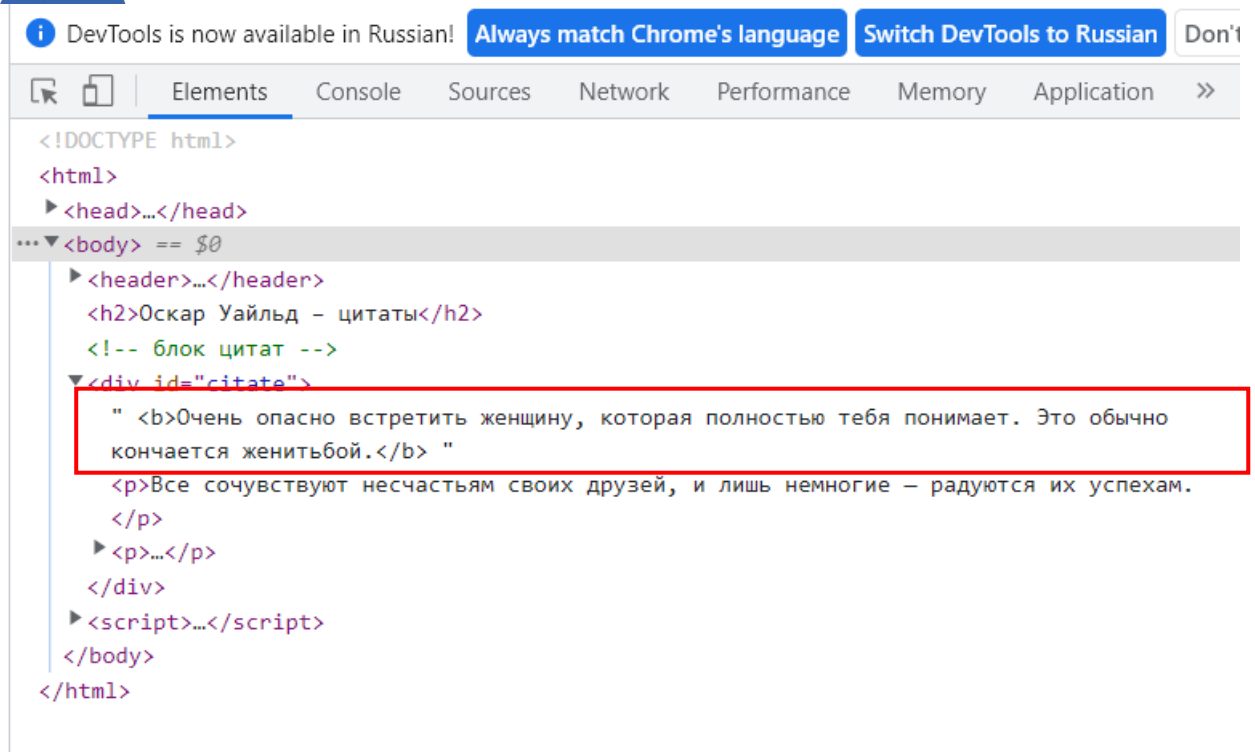


Рис.1. Результат вставки данных свойством outerText

textContent

Свойство **textContent** содержит только **текст** внутри этого текстового узла и его потомков, за вычетом всех тегов.

Синтаксис

```
element.textContent; // получить текстовое содержимое узла и
                      всех его потомков
```

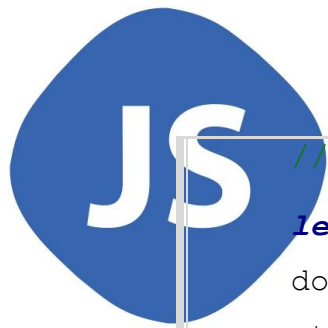
```
element.textContent = "text"; // установить текстовое
                               содержимое узла
```

При **установке** нового текстового содержимого, все дочерние узлы удаляются и заменяются одним текстовым узлом, содержащем указанную строку.

example_4. Свойство textContent

```
<script>
```





```
// ссылка на элемент div
let div =
document.body.firstChild.nextElementSibling.nextElementSibling;

// выведем textContent элемента
console.log(div.textContent);

// пока отличий от innerText не видно
console.log(div.innerText);

// определим новое текстовое содержимое узла div
// h1 - не работает ;)
div.textContent = "<h1>Упс..., содержимое блока div
удалено</h1>";

// на замене innerText работает аналогично
// div.innerText = "<h1>Упс..., содержимое блока div
удалено</h1>";

</script>
```

Отличие textContent от innerText

Для элемента, который содержит множество других узлов, textContent вернёт **конкатенацию (сложение) текстов всех его текстовых узлов**.

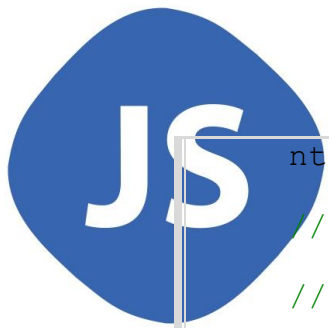
Внимательно проанализируйте **HTML-код** примера example_5, думаю вопросов не останется.

example_5. Особенности textContent

```
<script>

// ссылка на элемент div
let div =
document.body.firstChild.nextElementSibling.nextElementSibling;
```





```
ntSibling;

// вывод свойств в документ

// вот здесь все становится ясно

document.write("<h2>Свойство textContent</h2>");

document.write (div.textContent);

document.write("<h2>Свойство innerText</h2>");

document.write (div.innerText);

</script>
```

Свойства, предназначенные для работы с HTML содержимым элемента

innerHTML

Свойство **innerHTML** позволяет получить или установить **содержимое** узла-элемента.

Синтаксис

```
element.innerHTML; // получить содержимое узла
```

```
element.innerHTML = "element"; // установить содержимое узла
```

Значение, возвращаемое innerHTML – **всегда валидный HTML-код**. При записи можно попробовать записать что угодно, но браузер исправит ошибки.

example_6. Свойство innerHTML

```
<script>

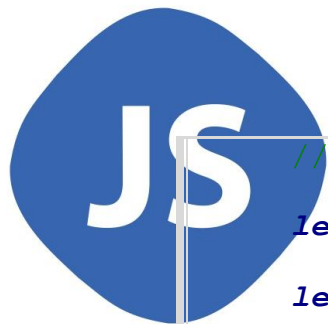
// ссылка на узел body

let body = document.body;

// получили доступ к элементу header

let header = body.firstChild;
```





```
// далее спускаемся вниз по дереву узлов
let h = header.nextElementSibling;

let div = h.nextElementSibling;

let p = div.firstElementChild.nextElementSibling;

// выводим значение html-узла p
console.log(p.innerHTML);

// изменим значение узла
p.innerHTML = "<b>Думай хорошо и мысли созреют в добрые
поступки.<b>";

// а вот здесь теги в строке уже имеют значение
</script>
```

outerHTML

Свойство **outerHTML** позволяет получить или установить элемент целиком.

Синтаксис

```
element.outerHTML; // получить элемент
```

```
element.outerHTML = "element"; // установить элемент
```

Комментарии

Не рекомендуется режим изменения.

example_7. Свойство outerHTML

```
<script>

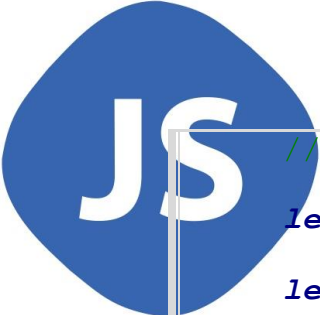
// ссылка на узел body

let body = document.body;

// получили доступ к элементу header

let header = body.firstElementChild;
```





```

// далее спускаемся вниз по дереву узлов
let h = header.nextElementSibling;

let div = h.nextElementSibling;

let p =
div.firstChild.nextElementSibling.nextElementSibling;

// выводим html-узел, соответствующий первому элементу p
console.log(p.outerHTML);

// заменим узел целиком

p.outerHTML = "<h2>Есть три вещи, которых боится большинство
людей: доверять, говорить правду и быть собой.</h2>";

</script>

```

Не забывайте проверять работу скрипта в режиме разработчика.

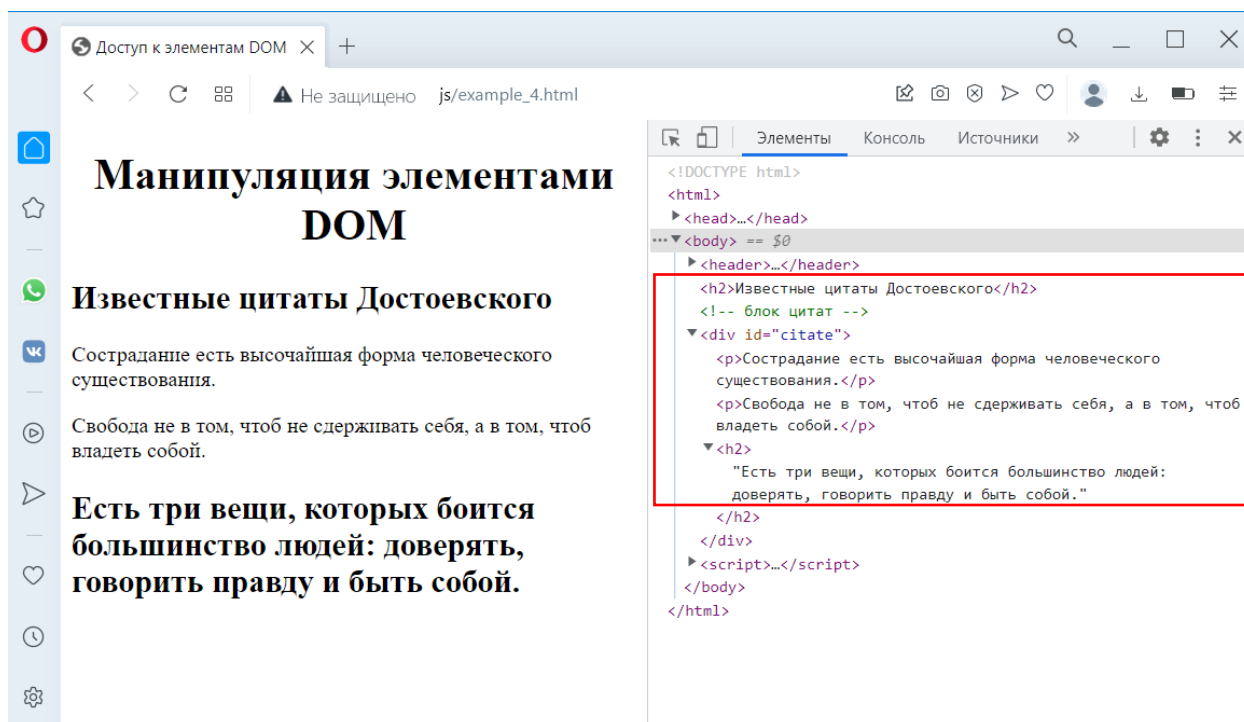


Рис.2. Результат вставки данных свойством outerHTML

Свойства, предназначенные для работы с узлами





Свойство **data** используется для получения или установки значений "текстовых" узлов и "комментариев". Для узлов "элементов" оно не доступно.

Синтаксис

```
node.data; // получить значение узла
```

```
node.data = "new node"; // установить значение узла
```

Важно. Следующий пример своего рода тест, насколько хорошо вы разобрались с понятием DOM дерева узлов. Внимательно прочитайте комментарий к примеру, это важно.

example_8. Свойство data

```
<script>

    // ссылка на узел body

    let body = document.body;

    // получили доступ к элементу header

    let header = body.firstChild;

    // далее спускаемся вниз по дереву узлов

    let h = header.nextElementSibling;

    let div = h.nextElementSibling;

    let p = div.firstChild;

    // получаем ссылку на текстовый узел первого элемента p

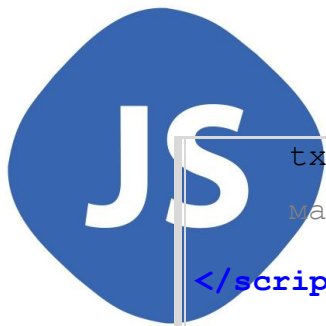
    let txt = p.firstChild;

    // выводим содержимое текстового узла

    console.log(txt.data);

    // изменим узел
```





```
txt.data = "Почему я такой большой дядя, а веду себя, как  
маленькая тетя?";  
</script>
```

nodeName

Свойство **nodeName** возвращает имя указанного узла.

Синтаксис

```
node.nodeName;
```

Возвращаемое значение

Строка, представляющая имя узла.

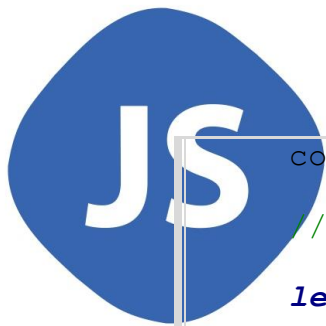
Возможные значения

- возвращает **тэг** для узлов элементов, в **верхнем регистре**;
- возвращает **имя атрибута** для узлов атрибутов;
- возвращает **"#text"** для текстовых узлов;
- возвращает **"#comment"** для узлов комментариев;
- возвращает **"#document"** для узлов документа.

example_9. Свойство nodeName

```
<script>  
  
  // ссылка на узел body  
  
  let body = document.body;  
  
  // элемент header  
  
  let header = body.firstChild;  
  
  console.log(header.nodeName); // HEADER  
  
  // элемент h2  
  
  let h = header.nextElementSibling;
```





```
console.log(h.nodeName); // H2
// пустой текстовый узел
let txt = h.nextSibling;
console.log(txt.nodeName); // #text
// комментарий
let comment = txt.nextSibling;
console.log(comment.nodeName); // #comment
</script>
```

nodeType

Свойство **nodeType** возвращает тип узла.

Синтаксис

```
node.nodeType;
```

Возвращаемое значение

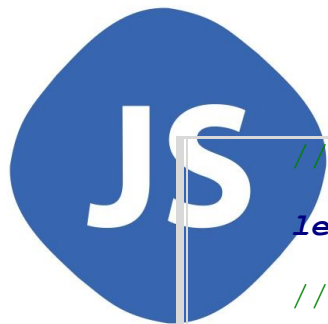
Число, представляющее тип узла. Всего существует 12 типов, в таблице представлены некоторые из узлов:

Значение	Описание
1	Узел элемента (возвращает корневой элемент документа, для HTML-документов это элемент HTML)
2	Узел атрибута (возвращает атрибут элемента XML- или HTML-документа)
3	Текстовый узел (#text)
8	Узел комментария

example_10. Свойство nodeType

```
<script>
```





```
// ссылка на узел body
let body = document.body;

// элемент header

let header = body.firstChild;

console.log(header.nodeType); // 1

// элемент h2

let h = header.nextElementSibling;

console.log(h.nodeType); // 1

// пустой текстовый узел

let txt = h.nextSibling;

console.log(txt.nodeType); // 3

// комментарий

let comment = txt.nextSibling;

console.log(comment.nodeType); // 8

</script>
```

nodeValue

Свойство **nodeValue** возвращает или устанавливает значение узла.

Эквивалентно свойству **data**. На практике чаще используют data, так как оно короче в записи.

Синтаксис

```
node.nodeValue; // получить значение узла
```

```
node.nodeValue = "value"; // установить значение узла
```

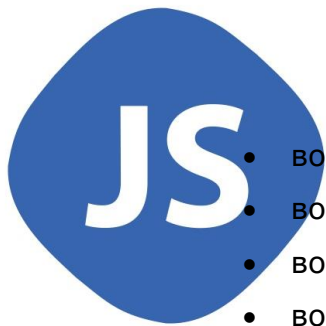
Возвращаемое значение

Строка, представляющая значение узла.

Возможные значения



Веб-разработка | Профессионалы | Образование
https://vk.com/pechora_pro



- возвращает пустое значение для узлов элементов и узлов документа;
- возвращает значение атрибута для узлов атрибутов;
- возвращает содержимое текстовых узлов;
- возвращает содержимое для комментариев узлов.

example_11. Свойство nodeValue

```
<script>

    // ссылка на узел body
    let body = document.body;

    // получили доступ к элементу h2
    let h = body.firstChild.nextElementSibling;

    // ссылка на узел комментарий
    let comment = h.nextSibling.nextSibling;

    // ссылка на текстовый узел
    let txt =
        comment.nextElementSibling.firstChild.firstChild;

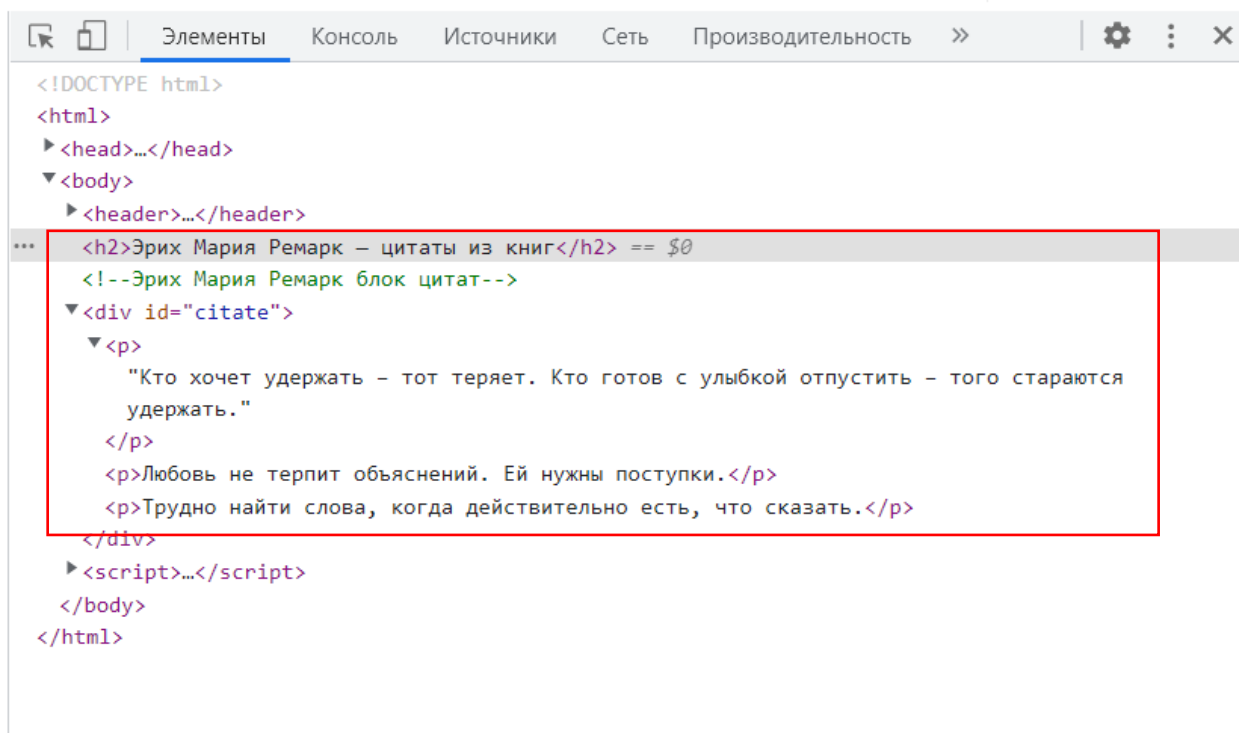
    // выводим в консоль значение узлов
    console.log(comment.nodeValue);
    console.log(txt.nodeValue);

    // устанавливаем новое значение узлов
    txt.nodeValue = "Кто хочет удержать - тот теряет. Кто готов
        с улыбкой отпустить - того стараются удержать.";
    comment.nodeValue = "Эрих Мария Ремарк блок цитат";

</script>
```

Режим разработчика покажет вам результат ваших преобразований.



**Рис.3.** Работа с узлами

tagName

Свойство **tagName** возвращает имя тега указанного элемента.

Синтаксис

```
element.tagName;
```

Возвращаемое значение

Строка, представляющая имя тега элемента в верхнем регистре.

example_12. Свойство tagName

```
<script>

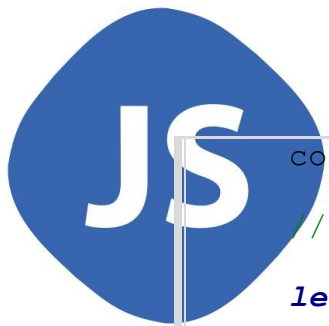
  // ссылка на узел body

  let body = document.body;

  // элемент header

  let header = body.firstChild;
```





```
console.log(header.tagName); // HEADER
// элемент h2
let h = header.nextElementSibling; // H2
console.log(h.tagName);
// текстовый узел
let txt = h.firstChild;
console.log(txt.tagName); // undefined
// комментарий
let comment = h.nextSibling.nextSibling;
console.log(comment.tagName); // undefined
// элемент div
let div = comment.nextElementSibling;
console.log(div.tagName); // DIV
</script>
```

hidden

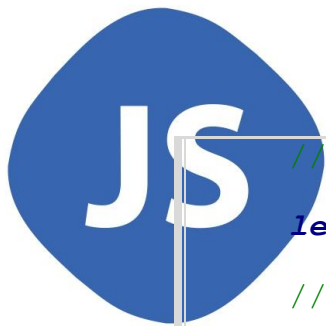
Атрибут и DOM-свойство **hidden** указывает на то, видим ли мы элемент или нет. Свойство может быть использовано в HTML или назначено при помощи JavaScript.

Примечание. Атрибуты и стили узлов разбираем в следующем уроке.

example_13. Свойство hidden

```
<script>
// ссылка на элемент div
let div =
document.body.lastElementChild.previousElementSibling;
```





```
// набор элементов p
let set = div.children;

// скрываем второй абзац (18+)
console.log(set[1].hidden = true);

// set[1] = set.item(1)

</script>
```

Закрепим пройденный материал. Соберем данные абзацев из блока цитат и выведем их в виде нумерованного списка.

example_14. Вывод узлов списком

```
<script>

// получаем ссылку на элемент div
let div =
document.body.firstChild.nextElementSibling.nextElementSibling;

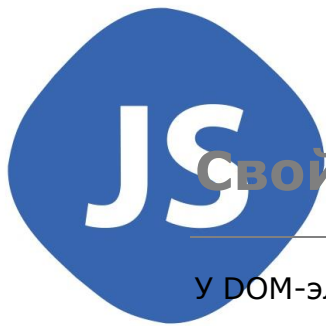
// получаем набор элементов p
let set = div.children;

// вывод в браузер содержимого элементов p
document.write("<ol>");

document.write("<li>", set.item(0).innerText, "</li>`");
document.write("<li>", set.item(1).innerText, "</li>");
document.write("<li>", set.item(2).innerText, "</li>");
document.write("</ol>");

</script>
```





Свойства, зависящие от класса

У DOM-элементов есть дополнительные свойства, в частности, зависящие от класса:

Вот только несколько из наиболее применяемых свойств:

- **value** – значение для **input**, **select** и **textarea** (HTMLInputElement, HTMLSelectElement...);
- **href** – адрес ссылки **href** для `` (HTMLAnchorElement);
- **id** – значение атрибута **id** для всех элементов (HTMLElement).

...и многие другие...

example_15. Дополнительные свойства

```
<script>

    let form =
    document.body.firstChild.nextElementSibling;

    // вывод свойств узла form

    console.log(form.tagName, form.id);

    let input = form.firstChild;

    // вывод свойств узла input

    console.log(input.nodeName, input.value);

    let a = form.nextElementSibling.lastElementChild;

    // вывод свойств узла a

    console.log(a.tagName, a.href);

</script>
```





Пользовательские свойства

Узлы DOM – это обычные **JavaScript объекты**. А у объектов есть свойства. Поэтому любому узлу можно назначить свойство, используя обычный синтаксис.

Значением свойства объекта может быть любой JS тип.

example_16. Добавление к узлу собственных свойств

```
<script>

    // ссылка на элемент div

    let div =
    document.body.firstChild.nextElementSibling;

    // добавим набор пользовательских свойств

    div._id = 1105;

    div._title = "Евгений Онегин";

    div._author = {name : "Александр", patronymic : "Сергеевич",
    surname : "Пушкин"};

    div._characters = ["Евгений Онегин", "Татьяна Ларина",
    "Ольга Ларина", "Владимир Ленский"];

    // выводим свойства элемента div

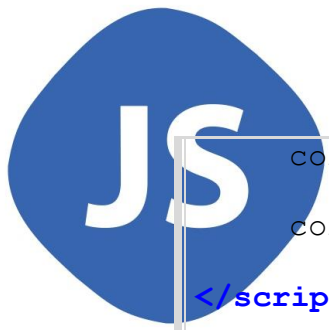
    // стандартные свойства узла-элемента

    console.log("Стандартные свойства:");
    console.log(div.id);
    console.log(div.nodeName);

    // пользовательские свойства узла-элемента

    console.log("Пользовательские свойства:");
    console.log(div._id);
    console.log(div._title);
```





```
console.log(div._author.surname);  
console.log(div._characters[1]);  
</script>
```

И еще раз подведем небольшой итог. В следующем примере создадим объект со следующими свойствами:

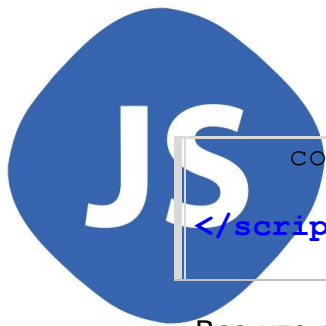
- автор цитат (**author**), тип данных – текст;
- цитаты (**citate**), тип данных – массив.

Значения свойств получим из соответствующих узлов дерева DOM.

example_17. Значения свойств объекта из узлов DOM

```
<script>  
  
    // ссылка на элемент h  
  
    let h = document.body.firstChild.nextElementSibling;  
  
    // ссылка на элемент div  
  
    let div = h.nextElementSibling;  
  
    // ссылка на абзацы p тега div  
  
    let set = div.children;  
  
    // создаем объект  
  
    let obj = new Object();  
  
    obj.author = h.innerText;  
  
    obj.citate = [  
  
        set.item(0).innerText,  
  
        set.item(1).innerText,  
  
        set.item(2).innerText,  
  
    ];  
  
    // вывод объекта в консоль
```





```
console.log(obj);  
</script>
```

Все что мы делаем – подготовка к манипулированию объектами при разработке клиентских приложений. Все знания, безусловно, будут востребованы в следующих уроках и в дальнейшем при веб-разработке.

В **самостоятельных работах** к уроку:

- Навигация по узлам DOM HTML-документа, цепочка вызовов.
- Работа со свойствами узлов.
- Получение доступа к узлам-комментариям.
- Создание JS-объекта со свойствами, содержащими значения из узлов документа.
- Вывод данных JS-объекта в HTML документ.

P.S.

Для отработки и закрепления учебного курса **донам** группы предоставляется следующий раздаточный материал.

К каждому уроку курса:

- Файлы **демонстрационного кода** (example);
- **Задачи** с решениями в контексте рассматриваемых вопросов урока (task).

К каждой теме курса:

- **Практические** работы.

