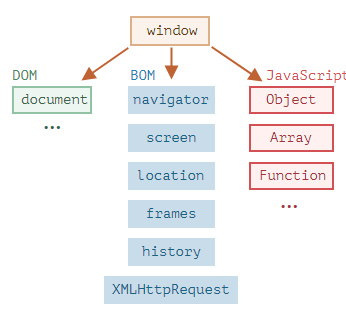
**JavaScript**

**Браузерное окружение, спецификации**

Язык JavaScript изначально был создан для веб-браузеров. Но с тех пор он значительно эволюционировал и превратился в кроссплатформенный язык программирования для решения широкого круга задач.

На картинке ниже в общих чертах показано, что доступно для JavaScript в браузерном окружении:



Как мы видим, имеется корневой объект window, который выступает в 2 ролях:

Во-первых, это глобальный объект для JavaScript-кода.

Во-вторых, он также представляет собой окно браузера и располагает методами для управления им.

Например, здесь мы используем window как глобальный объект:

function sayHi() {

alert("Hello");

}

// глобальные функции доступны как методы глобального объекта:

window.sayHi();

А здесь мы используем window как объект окна браузера, чтобы узнать его высоту:

alert(window.innerHeight); // внутренняя высота окна браузера

**DOM (Document Object Model)**

Document Object Model, сокращённо DOM – объектная модель документа, которая представляет все содержимое страницы в виде объектов, которые можно менять.

Объект document – основная «входная точка». С его помощью мы можем что-то создавать или менять на странице.

Например:

// заменим цвет фона на красный,

document.body.style.background = "red";

// а через секунду вернём как было

setTimeout(() => document.body.style.background = "", 1000);

**BOM (Browser Object Model)**

Объектная модель браузера (Browser Object Model, BOM) – это дополнительные объекты, предоставляемые браузером (окружением), чтобы работать со всем, кроме документа.

Например:

* Объект [navigator](https://developer.mozilla.org/ru/docs/Web/API/Window/navigator) даёт информацию о самом браузере и операционной системе. Среди множества его свойств самыми известными являются: navigator.userAgent – информация о текущем браузере, и navigator.platform – информация о платформе (может помочь в понимании того, в какой ОС открыт браузер – Windows/Linux/Mac и так далее).
* Объект [location](https://developer.mozilla.org/ru/docs/Web/API/Window/location) позволяет получить текущий URL и перенаправить браузер по новому адресу.

Вот как мы можем использовать объект location:

alert(location.href); // показывает текущий URL

if (confirm("Перейти на Wikipedia?")) {

location.href = "https://wikipedia.org"; // перенаправляет браузер на другой URL

}

**DOM-дерево**

Основой HTML-документа являются теги.

В соответствии с объектной моделью документа («Document Object Model», коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

Все эти объекты доступны при помощи JavaScript, мы можем использовать их для изменения страницы.

Например, document.body – объект для тега <body>.

## [Пример DOM](https://learn.javascript.ru/dom-nodes" \l "primer-dom)

Начнём с такого, простого, документа:

<!DOCTYPE HTML>

<html>

<head>

<title>О лосях</title>

</head>

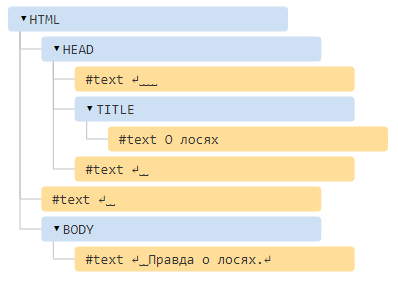
<body>

Правда о лосях.

</body>

</html>

DOM – это представление HTML-документа в виде дерева тегов. Вот как оно выглядит:



Каждый узел этого дерева – это объект.

Теги являются узлами-элементами (или просто элементами). Они образуют структуру дерева: <html> – это корневой узел, <head> и <body> его дочерние узлы и т.д.

Текст внутри элементов образует текстовые узлы, обозначенные как #text. Текстовый узел содержит в себе только строку текста. У него не может быть потомков, т.е. он находится всегда на самом нижнем уровне.

Например, в теге <title> есть текстовый узел "О лосях".

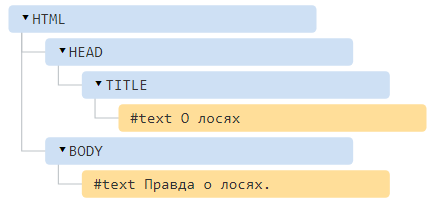
Обратите внимание на специальные символы в текстовых узлах:

* перевод строки: ↵ (в JavaScript он обозначается как \n)
* пробел: ␣

Здесь пробельных текстовых узлов нет:

<!DOCTYPE HTML>

<html><head><title>О лосях</title></head><body>Правда о лосях.</body></html>

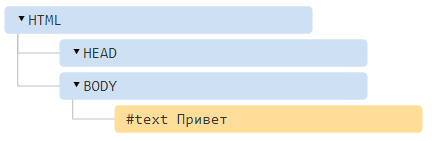


**Автоисправление**

Если браузер сталкивается с некорректно написанным HTML-кодом, он автоматически корректирует его при построении DOM.

Например, в начале документа всегда должен быть тег <html>. Даже если его нет в документе – он будет в дереве DOM, браузер его создаст. То же самое касается и тега <body>.

Например, если HTML-файл состоит из единственного слова "Привет", браузер обернёт его в теги <html> и <body>, добавит необходимый тег <head>, и DOM будет выглядеть так:



При генерации DOM браузер самостоятельно обрабатывает ошибки в документе, закрывает теги и так далее.

Есть такой документ с незакрытыми тегами:

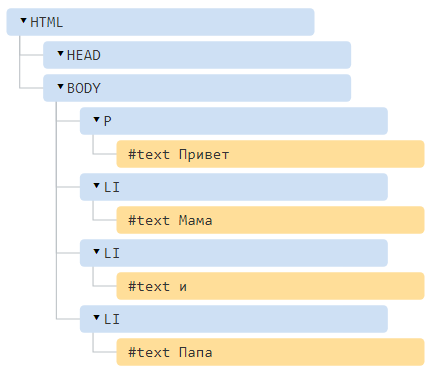
<p>Привет

<li>Мама

<li>и

<li>Папа

…Но DOM будет нормальным, потому что браузер сам закроет теги и восстановит отсутствующие детали:



**Все, что есть в HTML, даже комментарии, является частью DOM!**

Даже директива <!DOCTYPE...>, которую мы ставим в начале HTML, тоже является DOM-узлом. Она находится в дереве DOM прямо перед <html>

Существует [12 типов узлов](https://dom.spec.whatwg.org/#node). Но на практике мы в основном работаем с 4 из них:

1. document – «входная точка» в DOM.
2. узлы-элементы – HTML-теги, основные строительные блоки.
3. текстовые узлы – содержат текст.
4. комментарии – иногда в них можно включить информацию, которая не будет показана, но доступна в DOM для чтения JS.

**Поэкспериментируйте сами**

Чтобы посмотреть структуру DOM в реальном времени, попробуйте [Live DOM Viewer](http://software.hixie.ch/utilities/js/live-dom-viewer/). Просто введите что-нибудь в поле, и ниже вы увидите, как меняется DOM.

Другой способ исследовать DOM – это использовать инструменты разработчика браузера. Это то, что мы каждый день делаем при разработке.

Для этого откройте страницу [elks.html](https://learn.javascript.ru/article/dom-nodes/elks.html), включите инструменты разработчика и перейдите на вкладку Elements.

**Взаимодействие с консолью**

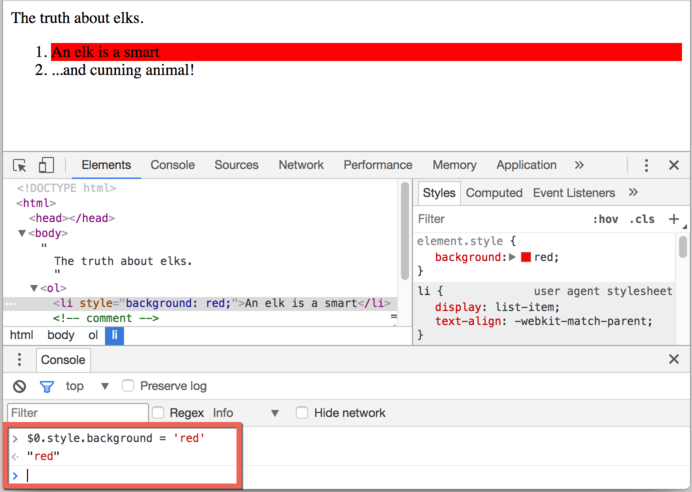
При работе с DOM нам часто требуется применить к нему JavaScript. Например: получить узел и запустить какой-нибудь код для его изменения, чтобы посмотреть результат. Вот несколько подсказок, как перемещаться между вкладками Elements и Console.

Для начала:

1. На вкладке Elements выберите первый элемент <li>.
2. Нажмите Esc – прямо под вкладкой Elements откроется Console.

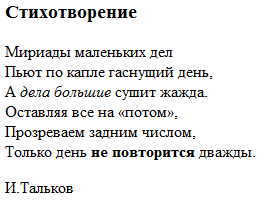
Последний элемент, выбранный во вкладке Elements, доступен в консоли как $0; предыдущий, выбранный до него, как $1 и т.д.

Теперь мы можем запускать на них команды. Например $0.style.background = 'red' сделает выбранный элемент красным, как здесь:



**Практическая работа 1.**

1. Используя[Live DOM Viewer](http://software.hixie.ch/utilities/js/live-dom-viewer/) наберите следующий стих с заданным форматированием.



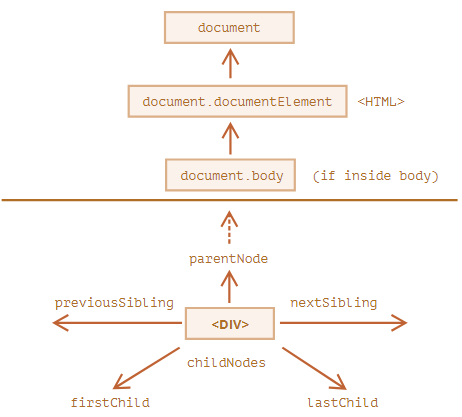
1. Используя консоль разработки закрасьте в разные цвета 2 и 5 строки стиха.

# Навигация по DOM-элементам

DOM позволяет нам делать что угодно с элементами и их содержимым, но для начала нужно получить соответствующий DOM-объект.

Все операции с DOM начинаются с объекта document. Это главная «точка входа» в DOM. Из него мы можем получить доступ к любому узлу.

Так выглядят основные ссылки, по которым можно переходить между узлами DOM:



## [Сверху: documentElement и body](https://learn.javascript.ru/dom-navigation" \l "sverhu-documentelement-i-body)

Самые верхние элементы дерева доступны как свойства объекта document:

**<html> = document.documentElement**

Самый верхний узел документа: document.documentElement. В DOM он соответствует тегу <html>.

**<body> = document.body**

Другой часто используемый DOM-узел – узел тега <body>: document.body.

**<head> = document.head**

Тег <head> доступен как document.head.

## [Дети: childNodes, firstChild, lastChild](https://learn.javascript.ru/dom-navigation" \l "deti-childnodes-firstchild-lastchild)

Здесь и далее мы будем использовать два принципиально разных термина:

* **Дочерние узлы (или дети)** – элементы, которые являются непосредственными детьми узла. Другими словами, элементы, которые лежат непосредственно внутри данного. Например, <head> и <body> являются детьми элемента <html>.
* **Потомки** – все элементы, которые лежат внутри данного, включая детей, их детей и т.д.

В примере ниже детьми тега <body> являются теги <div> и <ul> (и несколько пустых текстовых узлов):

<html>

<body>

<div>Начало</div>

<ul>

<li>

<b>Информация</b>

</li>

</ul>

</body>

</html>

…А потомки <body>– это и прямые дети <div>, <ul> и вложенные в них: <li> (ребёнок <ul>) и <b> (ребёнок <li>) – в общем, все элементы поддерева.

**Коллекция childNodes содержит список всех детей, включая текстовые узлы.**

Пример ниже последовательно выведет детей document.body:

<html>

<body>

<div>Начало</div>

<ul>

<li>Информация</li>

</ul>

<div>Конец</div>

<script>

for (let i = 0; i < document.body.childNodes.length; i++) {

alert( document.body.childNodes[i] ); // Text, DIV, Text, UL, ..., SCRIPT

}

</script>

...какой-то HTML-код...

</body>

</html>

**Свойства firstChild и lastChild обеспечивают быстрый доступ к первому и последнему дочернему элементу.**

Они, по сути, являются всего лишь сокращениями. Если у тега есть дочерние узлы, условие ниже всегда верно:

elem.childNodes[0] === elem.firstChild

elem.childNodes[elem.childNodes.length - 1] === elem.lastChild

Для проверки наличия дочерних узлов существует также специальная функция elem.hasChildNodes().

### [DOM-коллекции](https://learn.javascript.ru/dom-navigation" \l "dom-kollektsii)

Как мы уже видели, childNodes похож на массив. На самом деле это не массив, а коллекция – особый перебираемый объект-псевдомассив.

И есть два важных следствия из этого:

1. Для перебора *коллекции* мы можем использовать for..of:

for (let node of document.body.childNodes) {

alert(node); // покажет все узлы из коллекции

}

|  |
| --- |
| DOM-коллекции – только для чтения  DOM-коллекции, и даже более – *все* навигационные свойства, перечисленные в этой главе, доступны только для чтения.  Мы не можем заменить один дочерний узел на другой, просто написав childNodes[i] = ....  Для изменения DOM требуются другие методы. Мы увидим их в следующей главе. |

|  |
| --- |
| DOM-коллекции живые  Почти все DOM-коллекции, за небольшим исключением, *живые*. Другими словами, они отражают текущее состояние DOM.  Если мы сохраним ссылку на elem.childNodes и добавим/удалим узлы в DOM, то они появятся в сохранённой коллекции автоматически. |

## [Соседи и родитель](https://learn.javascript.ru/dom-navigation" \l "sosedi-i-roditel)

Соседи – это узлы, у которых один и тот же родитель.

Например, здесь <head> и <body> соседи:

<html>

<head>...</head><body>...</body>

</html>

* говорят, что <body> – «следующий» или «правый» сосед <head>
* также можно сказать, что <head> «предыдущий» или «левый» сосед <body>.

Следующий узел того же родителя (следующий сосед) – в свойстве nextSibling, а предыдущий – в previousSibling.

Родитель доступен через parentNode.

Например:

// родителем <body> является <html>

alert( document.body.parentNode === document.documentElement ); // выведет true

// после <head> идёт <body>

alert( document.head.nextSibling ); // HTMLBodyElement

// перед <body> находится <head>

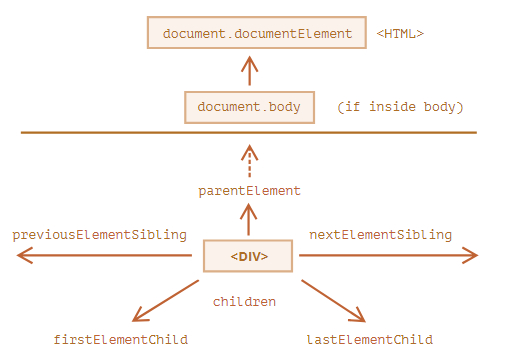
alert( document.body.previousSibling ); // HTMLHeadElement

## [Навигация только по элементам](https://learn.javascript.ru/dom-navigation" \l "navigatsiya-tolko-po-elementam)

Навигационные свойства, описанные выше, относятся ко всем узлам в документе. В частности, в childNodes находятся и текстовые узлы и узлы-элементы и узлы-комментарии, если они есть.

Но для большинства задач текстовые узлы и узлы-комментарии нам не нужны. Мы хотим манипулировать узлами-элементами, которые представляют собой теги и формируют структуру страницы.

Поэтому давайте рассмотрим дополнительный набор ссылок, которые учитывают только узлы-элементы:



Эти ссылки похожи на те, что раньше, только в ряде мест стоит слово Element:

* children – коллекция детей, которые являются элементами.
* firstElementChild, lastElementChild – первый и последний дочерний элемент.
* previousElementSibling, nextElementSibling – соседи-элементы.
* parentElement – родитель-элемент.

Изменим один из примеров выше: заменим childNodes на children. Теперь цикл выводит только элементы:

<html>

<body>

<div>Начало</div>

<ul>

<li>Информация</li>

</ul>

<div>Конец</div>

<script>

for (let elem of document.body.children) {

alert(elem); // DIV, UL, DIV, SCRIPT

}

</script>

...

</body>

</html>

## [Ещё немного ссылок: таблицы](https://learn.javascript.ru/dom-navigation" \l "dom-navigation-tables)

До сих пор мы описывали основные навигационные ссылки.

Некоторые типы DOM-элементов предоставляют для удобства дополнительные свойства, специфичные для их типа.

Таблицы – отличный пример таких элементов.

**Элемент <table>**, в дополнение к свойствам, о которых речь шла выше, поддерживает следующие:

* table.rows – коллекция строк <tr> таблицы.
* table.caption/tHead/tFoot – ссылки на элементы таблицы <caption>, <thead>, <tfoot>.
* table.tBodies – коллекция элементов таблицы <tbody> (по спецификации их может быть больше одного).

**<thead>, <tfoot>, <tbody>** предоставляют свойство rows:

* tbody.rows – коллекция строк <tr> секции.

**<tr>:**

* tr.cells – коллекция <td> и <th> ячеек, находящихся внутри строки <tr>.
* tr.sectionRowIndex – номер строки <tr> в текущей секции <thead>/<tbody>/<tfoot>.
* tr.rowIndex – номер строки <tr> в таблице (включая все строки таблицы).

**<td> and <th>:**

* td.cellIndex – номер ячейки в строке <tr>.

Пример использования:

<table id="table">

<tr>

<td>один</td><td>два</td>

</tr>

<tr>

<td>три</td><td>четыре</td>

</tr>

</table>

<script>

// выводит содержимое первой строки, второй ячейки

alert( table.rows[0].cells[1].innerHTML ) // "два"

</script>

## [Итого](https://learn.javascript.ru/dom-navigation" \l "itogo)

Получив DOM-узел, мы можем перейти к его ближайшим соседям используя навигационные ссылки.

Есть два основных набора ссылок:

* Для всех узлов: parentNode, childNodes, firstChild, lastChild, previousSibling, nextSibling.
* Только для узлов-элементов: parentElement, children, firstElementChild, lastElementChild, previousElementSibling, nextElementSibling.

Некоторые виды DOM-элементов, например таблицы, предоставляют дополнительные ссылки и коллекции для доступа к своему содержимому.

## **ЗАДАЧИ**

**Дочерние элементы в DOM**

Для страницы:

<html>

<body>

<div>Пользователи:</div>

<ul>

<li>Джон</li>

<li>Пит</li>

</ul>

</body>

</html>

Напишите код, как получить…

* элемент <div>?
* <ul>?
* второй <li> (с именем Пит)?

**Решение.**

Есть несколько способов для получения элементов, например:

DOM-узел элемента <div>:

document.body.firstElementChild

// или

document.body.children[0]

// или (первый узел пробел, поэтому выбираем второй)

document.body.childNodes[1]

DOM-узел элемента <ul>:

document.body.lastElementChild

// или

document.body.children[1]

Второй <li> (с именем Пит):

// получаем <ul>, и его последнего ребёнка

document.body.lastElementChild.lastElementChild

**Выделите ячейки по диагонали**

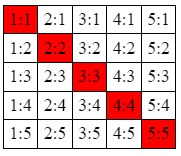
Напишите код, который выделит красным цветом все ячейки в таблице по диагонали.

Вам нужно получить из таблицы <table> все диагональные <td> и выделить их, используя код:

// в переменной td находится DOM-элемент для тега <td>

td.style.backgroundColor = 'red';

Должно получиться так:



**Решение.**

<!DOCTYPE HTML>

<html>

<head>

  <style>

    table {

      border-collapse: collapse;

    }

    td {

      border: 1px solid black;

      padding: 3px 5px;

    }

  </style>

</head>

<body>

  <table>

    <tr>

      <td>1:1</td>

      <td>2:1</td>

      <td>3:1</td>

      <td>4:1</td>

      <td>5:1</td>

    </tr>

    <tr>

      <td>1:2</td>

      <td>2:2</td>

      <td>3:2</td>

      <td>4:2</td>

      <td>5:2</td>

    </tr>

    <tr>

      <td>1:3</td>

      <td>2:3</td>

      <td>3:3</td>

      <td>4:3</td>

      <td>5:3</td>

    </tr>

    <tr>

      <td>1:4</td>

      <td>2:4</td>

      <td>3:4</td>

      <td>4:4</td>

      <td>5:4</td>

    </tr>

    <tr>

      <td>1:5</td>

      <td>2:5</td>

      <td>3:5</td>

      <td>4:5</td>

      <td>5:5</td>

    </tr>

  </table>

  <script>

    let table = document.body.firstElementChild;

    for (let i = 0; i < table.rows.length; i++) {

      let row = table.rows[i];

      row.cells[i].style.backgroundColor = 'red';

    }

  </script>

</body>

</html>

**Задание 3.**

На основе задания ПР № 2.3 из МП по HTML получите элементы:

* <li>Пункт 3.1</li>
* <li>Пункт 1</li>
* <ul> который вложен в <ol>
* <li>Пункт 4.2.3<li>
* <li>Пункт 5<li>

