



location - объект для работы с адресной строкой (адрес страницы, дополнительные параметры в URL);

navigator - объект для работы с браузером (название, версия, операционная система, текущее местоположение);

history - объект для навигации и сохранения состояний (перемещение по разделам сайта, кнопки вперёд, назад);

screen - информация о дисплее (размеры экрана, разрешение);

frames - коллекция со всеми окнами в текущей странице (всё window).

Объектная модель документа - это независимый от платформы и языка интерфейс, который позволяет программам и скриптам динамически обращаться и изменять содержимое, структуру и стили документа.

Объектная модель документа - это также представление документа в виде иерархического дерева узлов (элементов).

В терминах DOM каждый тег - это узел (**node**). Просто **текст** тоже представлен **узлом** (node с типом “**текстовый**”)

Методы для работы с DOM содержатся в глобальном объекте **document**

Document содержит все узлы DOM и функции для работы с ними.

document.documentElement - ссылка на узел *html*

document.body - ссылка на узел *body*

document.head - ссылка на узел *head*

Если скрипт подключен до *body*, то *document.body* будет равен **null**.

Это справедливо для любого другого тега - если скрипт пытается получить доступ к элементу, который объявлен ниже по коду, результатом будет **null**.

previousSibling, nextSibling - свойства, позволяющие получить предыдущий и следующий элементы (в том числе, текст и комментарии).

previousElementSibling, nextElementSibling - предыдущий и следующий html узел (элемент)

parentNode - свойство, содержащее ссылку на непосредственного родителя элемента.

parentElement - содержит ссылку на непосредственного родителя (html элемент).

childNodes - дочерние узлы (дети, непосредственно лежат в данном узле). К дочерним узлам также относятся текст, комментарии, переносы строки и другие символы.

children - дочерние html узлы (элементы).

firstChild, lastChild - первый и последний дочерние узлы (это может быть текст, комментарий и т.д.)

Все эти свойства позволяют прочитать значение, но не поменять его.

firstElementChild, lastElementChild, - первый и последний html узлы.

previousSibling, nextSibling - свойства, позволяющие получить предыдущий и следующий элементы (в том числе, текст и комментарии).

previousElementSibling, nextElementSibling - предыдущий и следующий html узел (элемент)

parentNode - свойство, содержащее ссылку на непосредственного родителя элемента.

parentElement - содержит ссылку на непосредственного родителя (html элемент).

childNodes - дочерние узлы (дети, непосредственно лежат в данном узле). К дочерним узлам также относятся текст, комментарии, переносы строки и другие символы.

children - дочерние html узлы (элементы).

firstChild, lastChild - первый и последний дочерние узлы (это может быть текст, комментарий и т.д.)

Все эти свойства позволяют прочитать значение, но не поменять его.

firstElementChild, lastElementChild, - первый и последний html узлы.

Все эти свойства позволяют прочитать значение, но не поменять его.

DOM. Задачи.

```
<html>
<head></head>
<body>
  <div>
    <p>Text</p>
    <p>Other</p>
    <p>Next</p>
    <p>Last</p>
  </div>
  <div></div>
  <div></div>
</body>
</html>
```

Зная структуру html, с помощью изученных методов получить (в консоль):

1. head
 2. body
 3. все дочерние элементы body и вывести их в консоль.
 4. первый div и все его дочерние узлы
 - а) вывести все дочерние узлы в консоль
 - б) вывести в консоль все дочерние узлы, кроме первого и последнего
- Для навигации по DOM использовать методы, которые возвращают только элементы

getElementsByTagName возвращает коллекцию элементов, которые соответствуют имени тега. Коллекция живая, то есть все изменения в html будут отображены и в коллекции (в отличие, например, от `childNodes`).

```
document.getElementsByTagName('div');
```

// [...] вернет коллекцию со всеми дивами на странице

```
document.getElementsByTagName('*');
```

// звездочка означает любой тег - вернутся все элементы на странице

```
var firstElement = document.body.firstChild, // получили 1й элемент
```

```
    paragraphs = firstElement.getElementsByTagName('p');
```

// получили все параграфы p внутри первого элемента

getElementsByClassName возвращает коллекцию элементов, которые соответствуют имени класса (атрибут тега class). Коллекция живая.

```
<div class="test"></div>
<div></div>
<p class="test"></p>
```

```
document.getElementsByClassName('test');
// [div, p] вернет коллекцию со всеми элементами, у которых class="test"
```

```
document.body.firstChild.getElementsByClassName('test');
// вернёт элементы с классом "test" внутри первого дочернего элемента body
```

querySelectorAll('selector') - возвращает коллекцию узлов, соответствующих селектору, переданному в метод. Селектор имеет такое же правило написания, как и селекторы CSS. Возвращает снимок узлов (то есть коллекция не живая).

```
document.querySelector('div'); // вернет все дивы на странице  
document.querySelector('a.link'); // вернет все ссылки с классом link  
document.querySelector('p span'); // вернет все спаны внутри параграфов
```

```
document.body.lastElementChild.querySelector('p');  
// вернет все параграфы внутри последнего дочернего элемента body
```

document.links - вернет коллекцию всех ссылок или элементов area на странице.

document.forms - вернет коллекцию всех форм на странице

getElementById - возвращает один элемент с заданным id. Подразумевается, что элемент с определенным id присутствует в единственном экземпляре на странице.

Метод **getElementById** может быть вызван только на **document**.

```
<a href="#" id="unic-link">Click me</a>
```

```
document.getElementById('unic-link'); // вернет элемент с id=unic-link
```

querySelector - возвращает первый элемент, который соответствует переданному селектору (как **querySelectorAll**).

```
document.querySelector('p'); // вернет первый параграф
```

`element.matches('selector')` - проверяет, соответствует ли элемент данному селектору.

```
<div class="one">One</div>
```

```
<div class="box">Two</div>
```

```
<div>Three</div>
```

```
var divs = document.getElementsByTagName('div');
```

```
for (var index = 0, max = divs.length; index < max; index++) {
```

```
    if (divs[index].matches('box')) {
```

```
        console.log(divs[index]);
```

```
    }
```

```
} // выведет в консоль все дивы с классом box
```

`element.closest('selector')` - ищет предка (родителя) в соответствии с переданным селектором. Синтаксис селектора аналогичен CSS. В случае совпадения вернет элемент, если ничего не найдено - `null`.

```
<div>
  <p>
    Other Text
    <span class="mark">info</span>
  </p>
</div>
document.querySelector('.mark').closest('div');
// найдёт первый элемент с классом mark и получит ближайший обрамляющий тег div
```

DOM. Разметка для задач.

```
<div>
  <article>
    <p>Lorem ipsum dolor sit amet, odio omnesque ius cu, quo ex atqui antiopam. At detracto menandri eos.
    Duo in causae viderer, graeci <a href="#">reprehendunt</a> has in. Decore <mark>nemore</mark>
    philosophia te pro, nobis legere causae ex mei, odio putant mentitum ea ius. Vix nostro deserunt explicari
    eu.</p>
  </article>
</div>
<ul>
  <li><a href="#">Link1</a></li>
  <li><a href="#">Link2</a></li>
  <li><a href="#">Link3</a></li>
  <li><a href="#">Link4</a></li>
</ul><span></span>
<a href="#">Some link</a>
```

DOM. Задачи.

1. Создать функцию, которая принимает два элемента. Функция проверяет, является ли первый элемент родителем для второго:

```
isParent(parent, child);
```

```
isParent(document.body.children[0], document.querySelector('mark'));
```

// true так как первый div является родительским элементом для mark

```
isParent(document.querySelector('ul'), document.querySelector('mark'));
```

// false так ul НЕ является родительским элементом для mark

Функция принимает только DOM объекты.

2. Получить список всех ссылок, которые не находятся внутри списка ul

3. Найти элемент, который находится перед и после списка ul