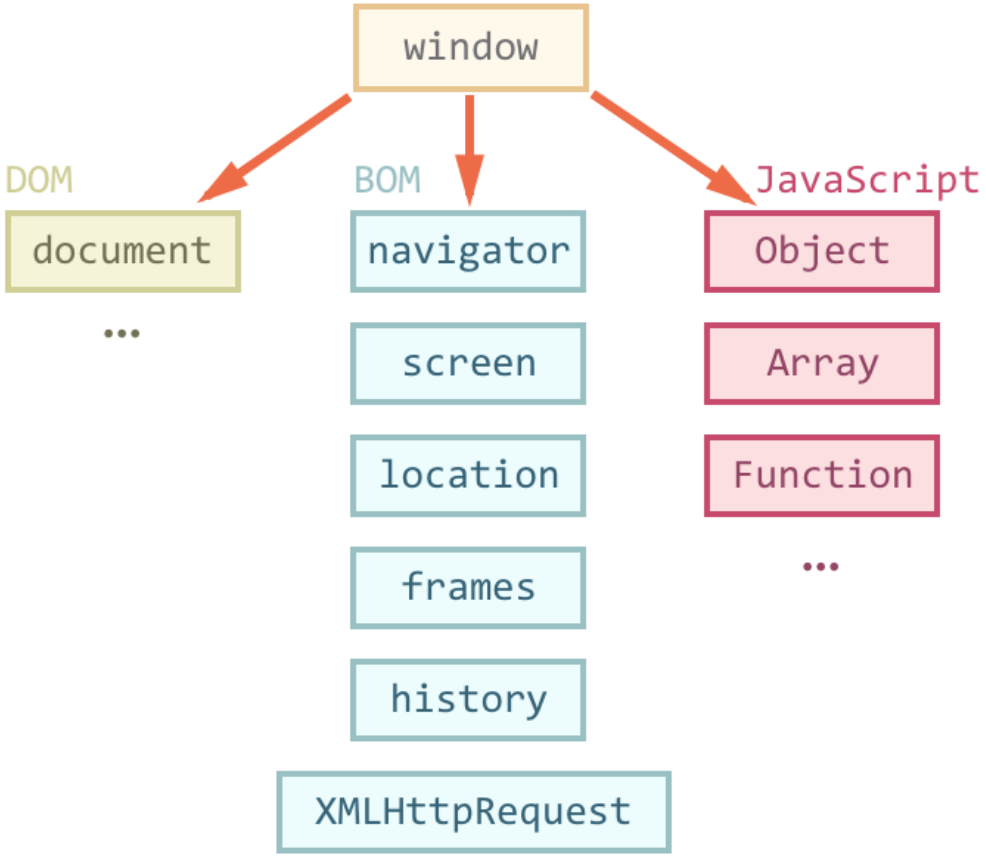
JavaScript

inside browser environment

Подключение JavaScript в HTML

```
<html>
    <head>
        <script>
        // embedded
        </script>
        <script src="/path/to/script.js"></script>
        <script async src="/path/to/script-1.js"></script>
        <script async src="/path/to/script-2.js"></script>
        <script defer src="/path/to/script-3.js"></script>
        <script defer src="/path/to/script-4.js"></script>
    </head>
    <body>
        ...
        <script>
        // embedded
        </script>
    </body>
</html>
```

Browser Environment

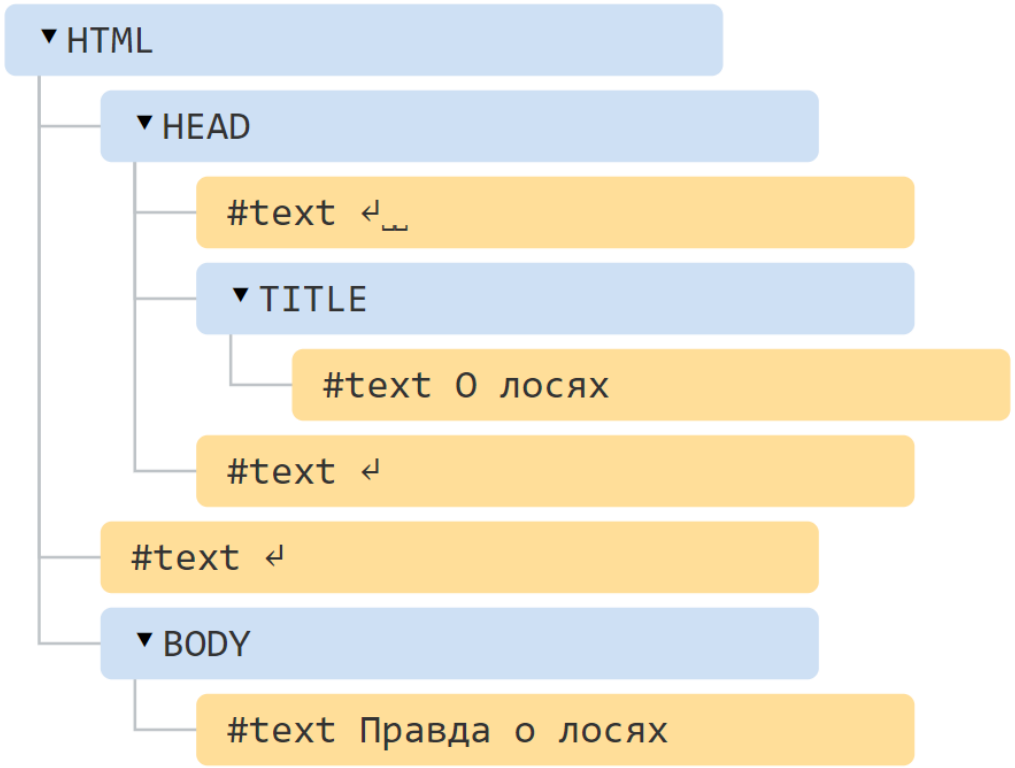


Browser Object Model

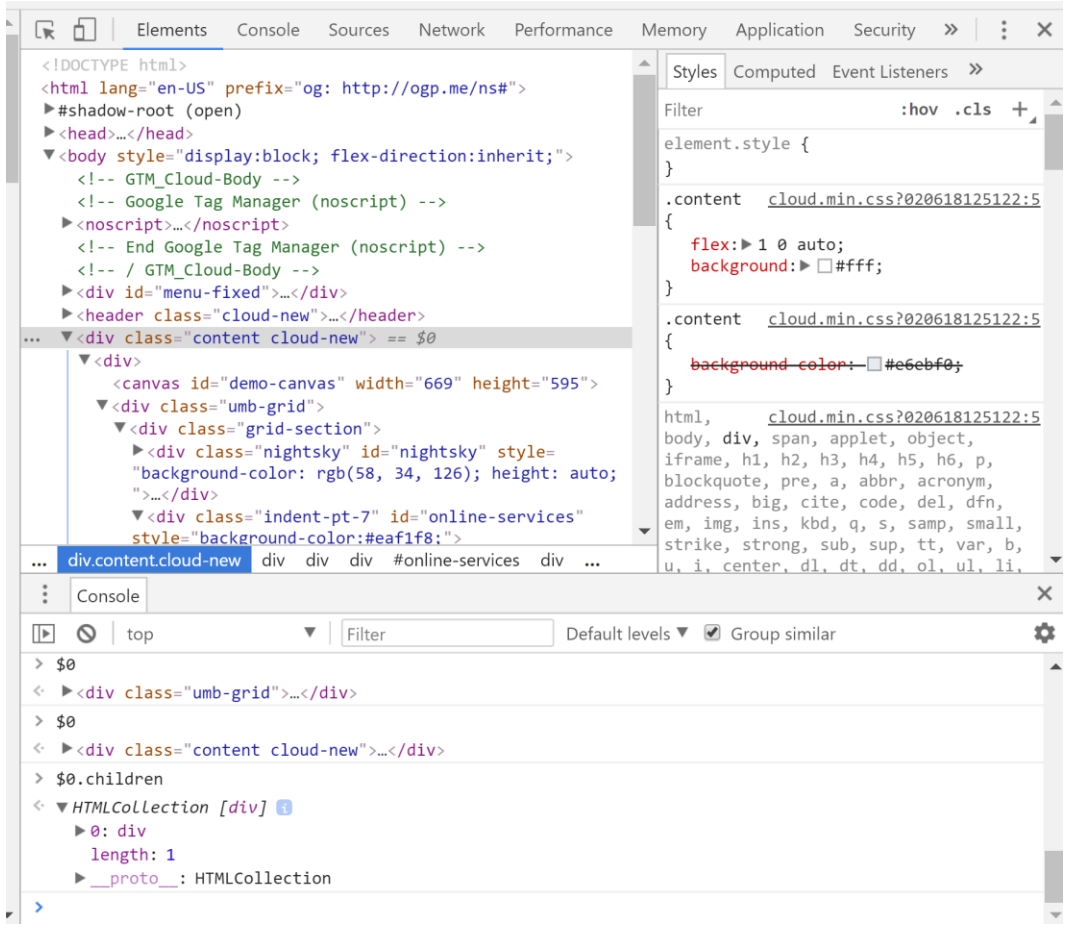
```
window.navigator; // Информация о самом браузере.

window.location; // Работа с адресной строкой.

window.history; // Навигация по истории.

window.screen; // Информация об экране.

// Всплывающие окна.
alert('тревога!');
prompt('вопрос?');
confirm("да или нет?");
```

Document Object Model

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4    <title>О лосях</title>
5  </head>
6  <body>
7    Правда о лосях
8  </body>
9  </html>
```



Работа с DOM из Chrome Developer Tools



Поиск элементов внутри DOM

```
// Возвращает элемент с id="foo".
document.getElementById('foo');

// Возвращает элемент с name="age".
document.getElementsByName('age');

// Возвращает все <div> элементы внутри element.
element.getElementsByTagName('div');

// Возвращает все элементы с классом "bar" внутри element.
element.getElementsByClassName('bar');
```

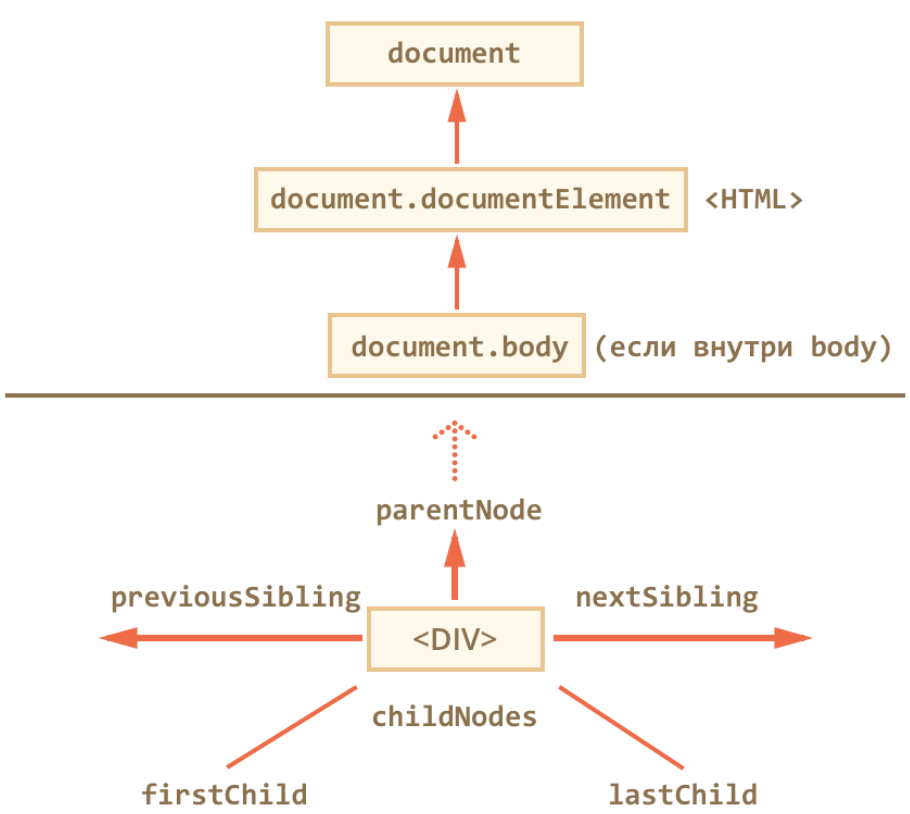
Результаты поиска getElementsBy* – живые.
При изменении документа – изменяется и результат запроса.

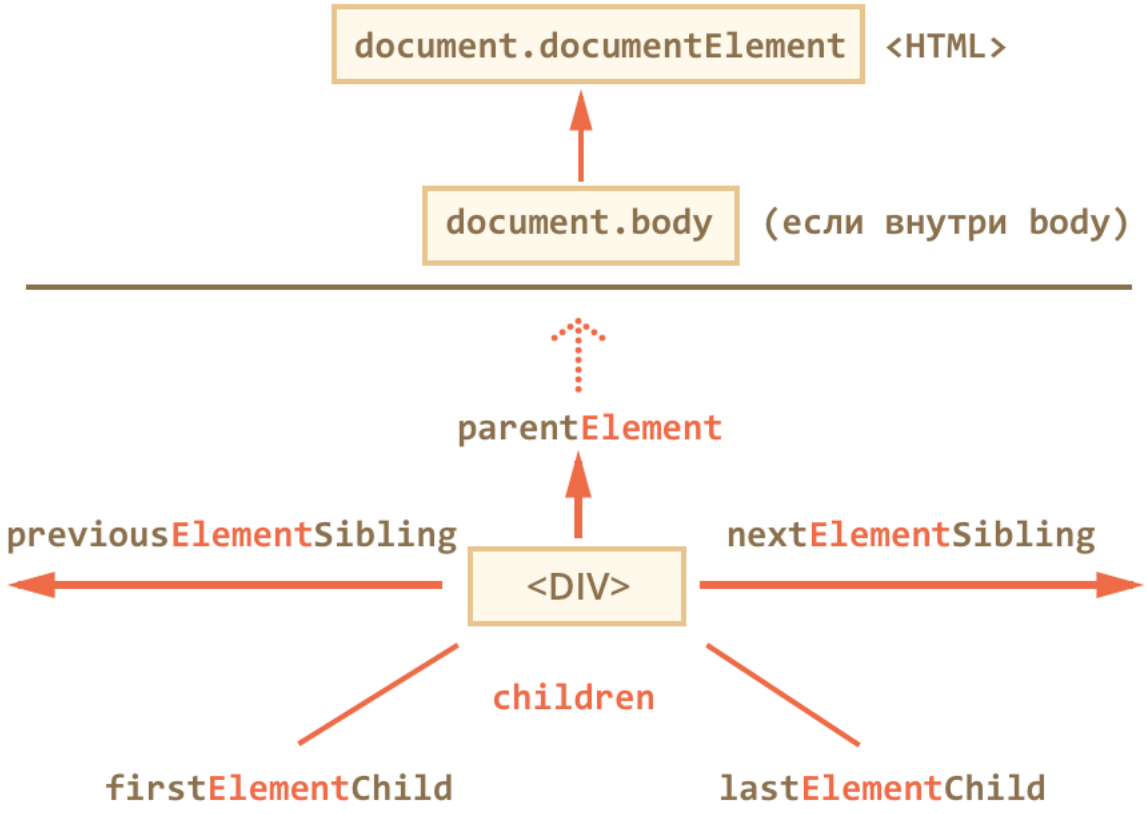
Поиск элементов внутри DOM

```
// Возвращает все элементы <li> внутри element,
// которые являются последними потомками в <ul>.
element.querySelectorAll('ul > li:last-child');

// Возвращает первый элемент внутри element,
// соответствующий css правилу.
element.querySelector('ul > li:last-child');
```

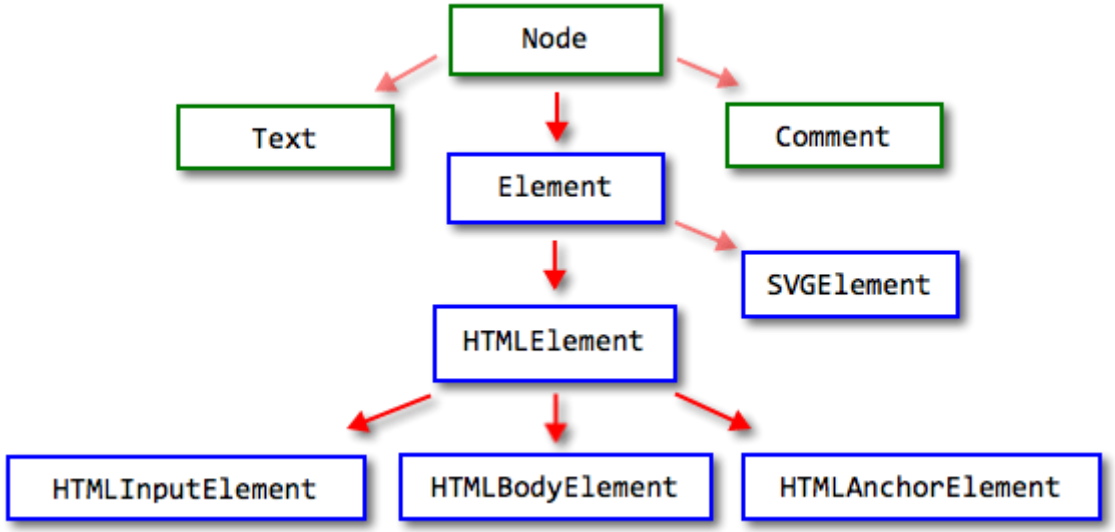
При изменении документа – результат запроса не изменяется.

Навигация по DOM-элементам





Иерархия классов в DOM



Добавление и удаление узлов

```
document.createElement(tag); // создает элемент

document.createTextNode(value); // создает текстовый узел

elem.cloneNode(deep); // клонирует элемент, если deep=true то со всеми потомками

parent.appendChild(elem); // добавляет elem в parent последним

parent.insertBefore(elem, nextSibling); // добавляет elem в parent перед nextSibling

parent.removeChild(elem); // удаляет elem из parent

parent.replaceChild(newElem, elem); // заменяет elem в parent
```

Свойства DOM узлов

```
// Тип узла. "1" – для элементов, "3" – для текстовых узлов.
node.nodeType;

// Содержит название тега. Есть только у элементов Element.
node.tagName;

// Содержит название тега (Element) или строку с типом узла (другие узлы).
node.nodeName;

// Внутреннее содержимое узла-элемента в виде HTML. Можно изменять.
node.innerHTML;

//Полный HTML узла-элемента. Можно изменять.
node.outerHTML;
```

HTML-атрибуты и DOM-свойства

```
elem.attributes; // возвращает все атрибуты

elem.hasAttribute(name); // проверяет наличие атрибута

elem.getAttribute(name); // получает значение атрибута

elem.setAttribute(name, value); // устанавливает атрибут

elem.removeAttribute(name); // удаляет атрибут
```

Браузер читает текстовые HTML-атрибуты, создаёт DOM-элементы со свойствами любого типа.

HTML-атрибуты и DOM-свойства

```
<body class="main page">
    <script>
        let classList = document.body.classList; // объект, а не строка
        for (let i = 0; i < classList.length; i++) {
            alert(classList[i]); // "main" "page"
        }
        alert(document.body.className); // строка "main page"
    </script>
</body>
```

```
<body>
    <input id="input" type="text" value="markup">
    <script>
        input.value = 'new'; // поменяли свойство
        alert(input.getAttribute('value')); // 'markup', не изменилось
    </script>
</body>
```

Браузерные события

Некоторые события мыши:

click – когда кликнули на элемент левой кнопкой
contextmenu – когда кликнули на элемент правой кнопкой
mouseover – когда на элемент наводится мышь
mousedown – когда кнопку мыши нажали или отжали
mouseup – когда кнопку мыши отжали
mousemove – при движении мыши

Некоторые события клавиатуры:

keydown – когда нажимается клавиша
keyup – когда отпускается клавиша

DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

Некоторые события на формах:

submit – посетитель отправил форму
focus – посетитель фокусируется на элементе

Обработчики из HTML-атрибутов

```
<div onclick="alert('clicked')">Click Me</div>
```

```
<script>
    function clickHandler() {
        alert('clicked');
    }
</script>
<div onclick="clickHandler()">Click Me</div>
```

Обработчики как DOM-свойства

```
<input type="button" id="button" value="Кнопка" />

<script>
    let buttonElement = document.getElementById('button');

    buttonElement.onclick = function () {
        alert('Клик');
    };
</script>
```

Как правильно назначать обработчики

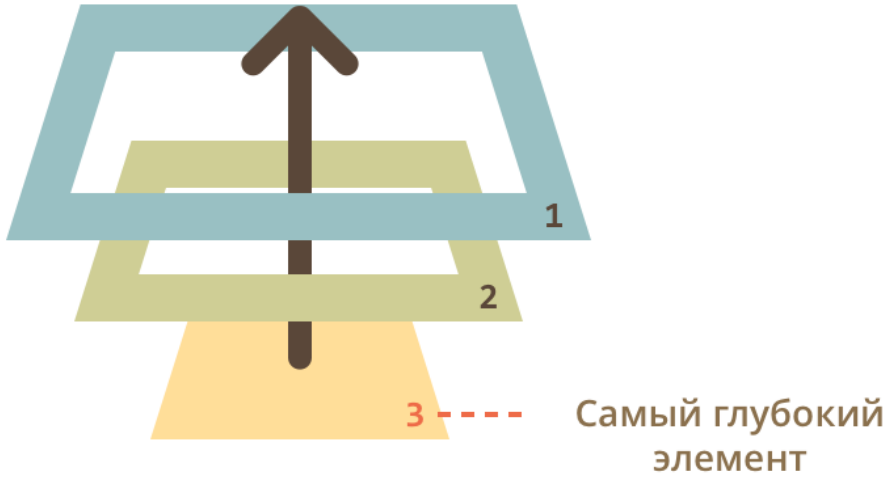
```
<input type="button" id="button" value="Кнопка" />

<script>
    let buttonElement = document.getElementById('button');

    function clickHandlerOne() {
        alert('Click one. Works always.');
    }

    function clickHandlerTwo() {
        alert('Click two. Works once.');
        buttonElement.removeEventListener('click', clickHandlerTwo);
    }

    buttonElement.addEventListener('click', clickHandlerOne);
    buttonElement.addEventListener('click', clickHandlerTwo);
</script>
```

Объект события

```
<input type="button" id="button" value="Кнопка" />

<script>
    let buttonElement = document.getElementById('button');

    buttonElement.addEventListener('click', function (event) {
        // тип события, "click"
        console.log(event.type);

        // элемент button
        console.log(event.currentTarget);

        // координаты клика относительно окна
        console.log(event.clientX, event.clientY);
    });
</script>
```

Event Bubbling (default)

```
<div id="outer">
    <div id="middle">
        <div id="inner">click me</div>
    </div>
</div>

<script>
    let divs = document.getElementsByTagName('div');

    function clickHandler(event) {
        // inner > middle > outer
        console.log(event.currentTarget.id);
    };

    for (let i = 0; i < divs.length; i++) {
        divs[i].addEventListener('click', clickHandler);
    }
</script>
```



Event capturing

```
<div id="outer">
    <div id="middle">
        <div id="inner">click me</div>
    </div>
</div>

<script>
    let divs = document.getElementsByTagName('div');

    function clickHandler(event) {
        // outer > middle > inner
        console.log(event.currentTarget.id);
    };

    for (let i = 0; i < divs.length; i++) {
        divs[i].addEventListener('click', clickHandler, true);
    }
</script>
```

Отмена дальнейшей обработки

```
<div id="outer">
    <div id="inner">click me</div>
</div>

<script>
    let outer = document.getElementById('outer');
    let inner = document.getElementById('inner');

    outer.addEventListener('click', function() {
        alert('never called');
    });

    inner.addEventListener('click', function(event) {
        alert('stop propagation');
        event.stopPropagation();
    });
</script>
```