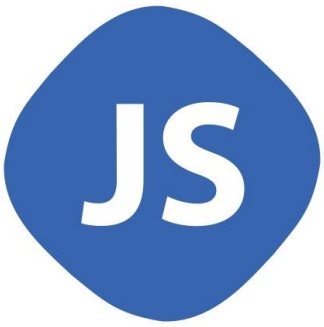


# Доступ к DOM

## Методы выбора DOM элементов

**Объектная модель документа** (**DOM**)— это независимый от платформы и языка интерфейс, который позволяет программам и скриптам динамически обращаться и изменять **содержимое**, **структуру** и **стили** документа. DOM представляет документ в виде иерархичного дерева узлов (элементов).

Если коротко, то DOM-дерево, состоит из узлов, которые вместе образуют иерархию, аналогичную HTML. В терминах DOM каждый тег — это узел (с типом: узел элемента). Просто текст тоже представлен узлом (с типом: текстовый узел).

Вообще существует 12 типов узлов, но на практике чаще всего приходится работать с узлами следующих типов:

- **document** – "входная точка" в DOM;
- **узлы-элементы** – HTML-теги (основные строительные блоки);
- **текстовые узлы** – содержат текст (в том числе пустой текст);
- **комментарии** – иногда в них можно включить информацию, которая не будет показана, но доступна в DOM для чтения с помощью JS.

Дерево состоит из узлов, но только **некоторые из узлов являются HTML-элементами**.

Часть узлов, при этом, являются **листовыми**, то есть не содержат внутри себя других узлов (детей), а часть **внутренними** – у них есть дети.

| Дети | |
|---|---|
| **childNodes** | содержит все дочерние узлы, в т. ч. текстовые |
| **children** | содержит все дочерние узлы-элементы |
| **firstChild** | первый дочерний узел |

| | |
|---|---|
| **firstElementChild** | первый дочерний узел-элемент |
| **lastChild** | последний дочерний узел |
| **lastElementChild** | последний дочерний узел-элемент |
| **Родитель** | |
| **parentNode** | родительский узел |
| **parentElement** | родительский узел-элемент |
| **Братья** (сестринские) | |
| **previousSibling** | предыдущий сестринский узел |
| **previousElementSibling** | предыдущий сестринский узел-элемент |
| **nextSibling** | следующий сестринский узел |
| **nextElementSibling** | следующий сестринский узел-элемент |

Конкретные узлы, чаще всего, описывают собой конкретные теги из HTML и содержат их атрибуты внутри себя. У узлов есть тип, который определяет набор свойств и методов узла.

> ***Важно***. Свойства и методы узла определяются типом, к которому он принадлежит. **Узлы-элементы** (теги) имеют один набор, **комментарии** другой и т.д.

Методы для работы с DOM содержатся в глобальном объекте **document**. Document содержит все узлы DOM и функции для работы с ним:

- **document.documentElement** - ссылка на узел **html**
- **document.body** - ссылка на узел **body**
- **document.head** - ссылка на узел **head**

**example_1**. Глобальный объект document

```
<script>

    // ссылка на узел html

    let html = document.documentElement;
```

```
    console.log(html);
    //ссылка на узел head
    let head = document.head;
    console.log(head);
    // ссылка на узел body
    let body = document.body;
    console.log(body);
</script>
```

> ***Важно***. 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**.

Свойства, позволяющие получить **первый** и **последний дочерние узлы** (это может быть текст, комментарий и т.д.):

- **firstChild**
- **lastChild**

### Свойства firstChild, lastchild

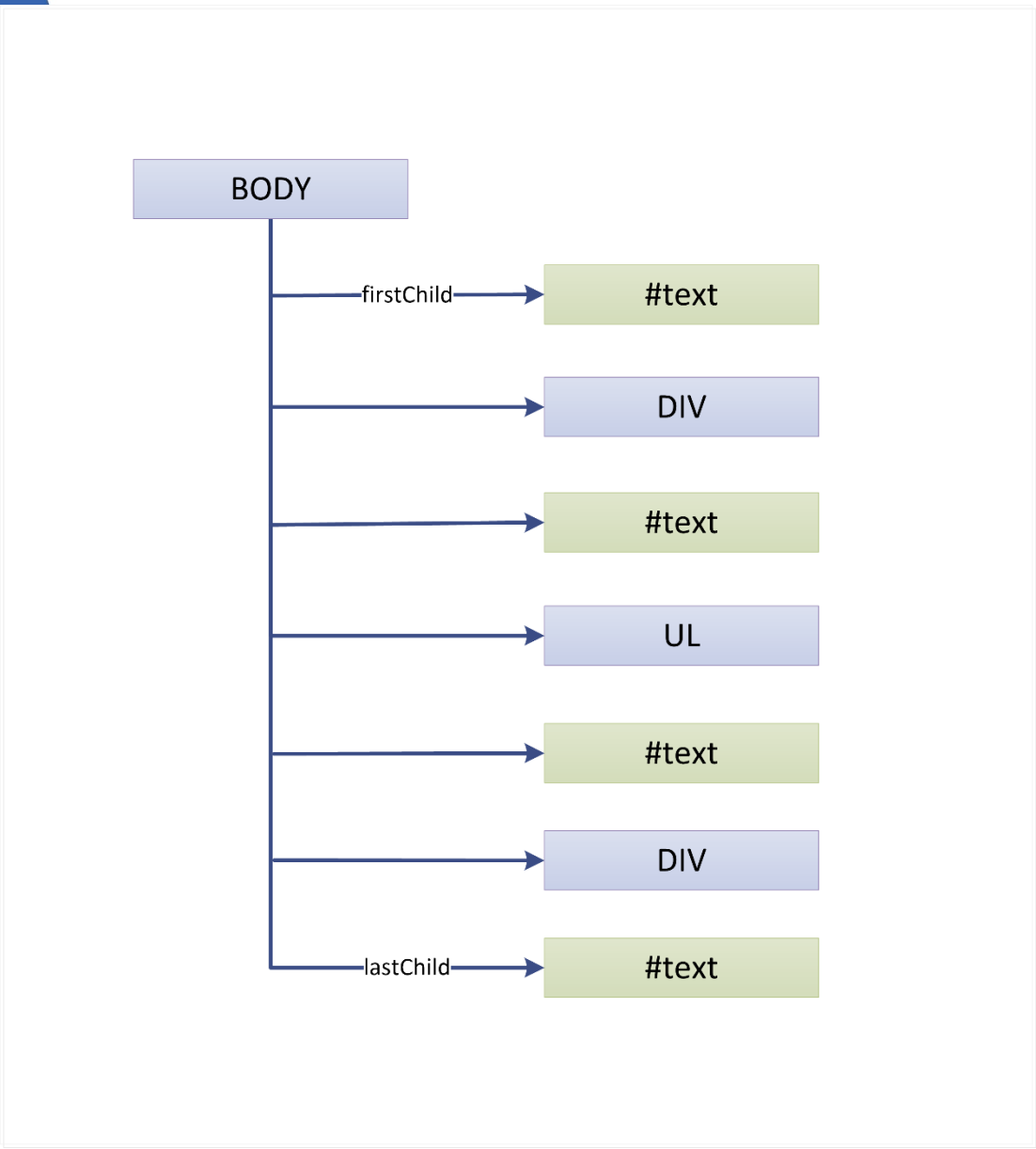



**Рис.1**. Свойства firstChild, lastChild

**example_2**. Свойства firstChild, lastChild

```
<script>
    // firstChild, lastChild – первый и последний дочерние узлы
    (это может быть текст, комментарий и т.д.)
    // ссылка на узел body
    let body = document.body;
    console.log(body);
```

```
    // первый дочерний узел для body это #text
    let text = body.firstChild;
    console.log(text);
    // последний дочерний узел для body это script
    let script = body.lastChild;
    console.log(script);
</script>
```

Дополним код примера example_2 созданием объекта. Объект в качестве значения свойства может хранить любой тип. Запишем полученные узлы дерева DOM в соответствующие свойства объекта.

Обратите внимание на инструкцию вывода в браузер:

```
document.write(myObject.body);
```

Узел выводим как **объект дерева узлов** DOM. С содержимым узлов будем работать в следующем уроке.

**example_3**. Записываем узлы в объект

```
<script>
    // ссылка на узел body
    let body = document.body;
    // первый дочерний узел для body это #text
    let text = body.firstChild;
    // последний дочерний узел для body это script
    let script = body.lastChild;
    // создаем пустой объект
    let myObject = {};
    // создаем свойства объекта
```

```
    // в свойства помещаем узлы
    myObject.body = body;
    myObject.text = text;
    myObject.script = script;
    // вывод объекта в консоль
    console.log(myObject);
    // полезно посмотреть вывод - мы выводим узел (объект)
    // доступ к содержимому узла - следующий урок
    document.write(myObject.body);
</script>
```

Свойства, позволяющие получить **первый** и **последний дочерние html узлы**:

- **firstElementChild**
- **lastElementChild**

**Свойства firstElementChild, lastElementChild**

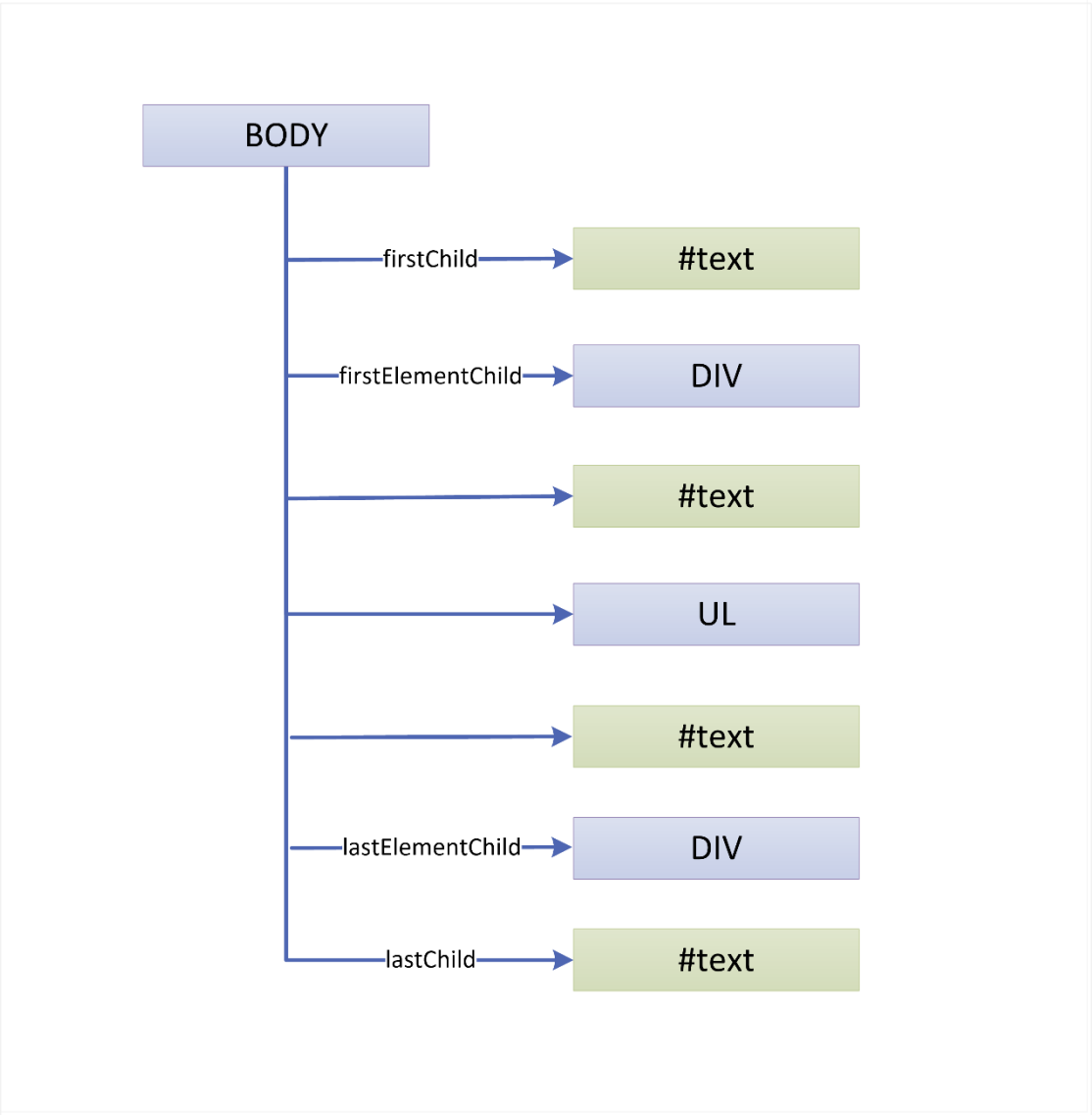



**Рис.2**. firstElementChild, lastElementChild

**example_4**. Свойства firstElementChild, lastElementChild

```
<script>
    // firstElementChild, lastElementChild – первый и последний
    html узлы.
    // ссылка на узел body
    let body = document.body;
    console.log(body);
    // первый дочерний html-узел для body это header
```

```
    let header = body.firstElementChild;
    console.log(header);
    // последний дочерний html-узел для body это script
    let script = body.lastElementChild;
    console.log(script);
</script>
```

Свойства, позволяющие получить **следующий** и **предыдущий узел** (в том числе, текст и комментарии):

- **nextSibling**
- **previousSibling**

### Свойства nextSibling, previousSibling

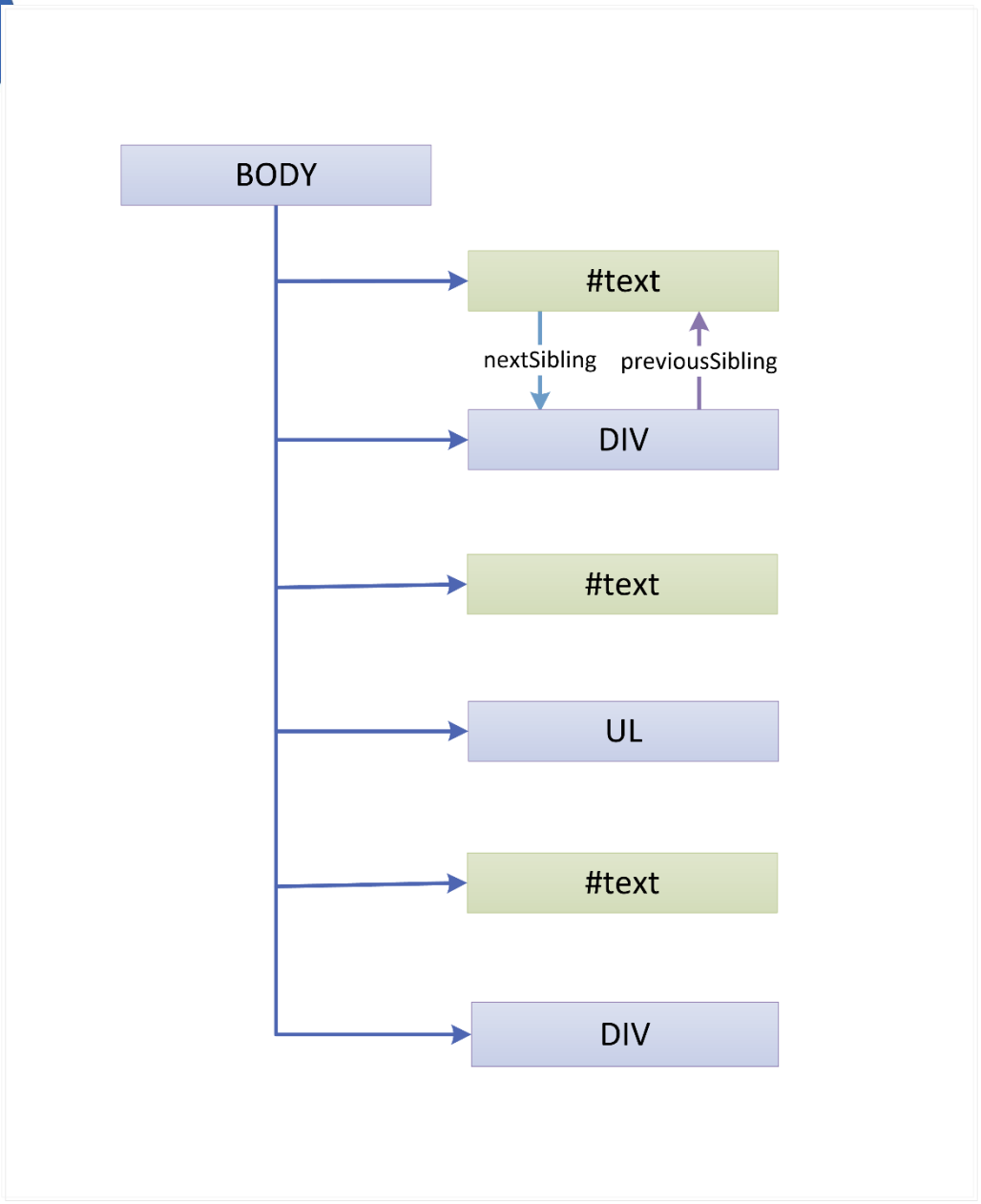



**Рис.3**. Свойства nextSibling, previousSibling

**example_5**. Свойства nextSibling, previousSibling

```
<script>

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

    // ссылка на узел body

    let body = document.body;
```

```
    console.log(body);
    // первый дочерний html-узел для body это header
    let header = body.firstElementChild;
    console.log("--> ", header);
    // следующий узел за header это пустой #text
    let txt1 = header.nextSibling;
    console.log("--> ", txt1);
    // следующий узел за txt1 это div#citate
    let div1 = txt1.nextSibling;
    console.log("--> ", div1);
    // следующий узел за div#citate это пустой #text
    let txt2 = div1.nextSibling;
    console.log("--> ", txt2);
    // вернемся обратно к div#citate
    let div2 = txt2.previousSibling;
    console.log("--> ", div2);
</script>
```

В следующем примере запишем полученные узлы дерева DOM в элементы массива.

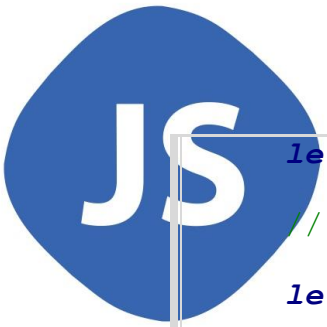
Обратите внимание на инструкцию вывода в браузер:

```
document.write(myArray[0]);
```

В данной инструкции выводим узел как **объект дерева узлов** DOM.

**example_6**. Записываем узлы в массив

```
<script>
    // ссылка на узел body
```



```
    let body = document.body;
    // первый дочерний html-узел для body это header
    let header = body.firstElementChild;
    // следующий узел за header это пустой #text
    let txt1 = header.nextSibling;
    // следующий узел за txt1 это div#citate
    let div = txt1.nextSibling;
    // внутри тега div#citate находятся узлы:
    // p -> #text -> p -> #text
    let p1 = div.firstElementChild;
    txt2 = p1.nextSibling;
    p2 = txt2.nextSibling;
    txt3 = p2.nextSibling;
    p3 = txt3.nextSibling;
    // создаем пустой массив
    myArray = Array();
    // в массив помещаем узлы абзацев
    myArray[0] = p1;
    myArray[1] = p2;
    myArray[2] = p3;
    console.log(myArray);
    // полезно посмотреть вывод - мы выводим узел (объект)
    // доступ к содержимому узла - следующий урок
    document.write(myArray[0]);
</script>
```

***Важно***. Будьте крайне внимательны при "вышагивании" по дереву DOM. Узел-элемент, пустой текстовый узел, узел-комментарий, все это необходимо учитывать. Именно по этой причине я не могу вставить поясняющие комментарии в HTML-код, сильно изменится **дерево узлов**.

Свойства, позволяющие получить следующий и предыдущий html-узел (элемент):

- **nextElementSibling**
- **previousElementSibling**

### Свойства nextElementSibling, previousElementSibling

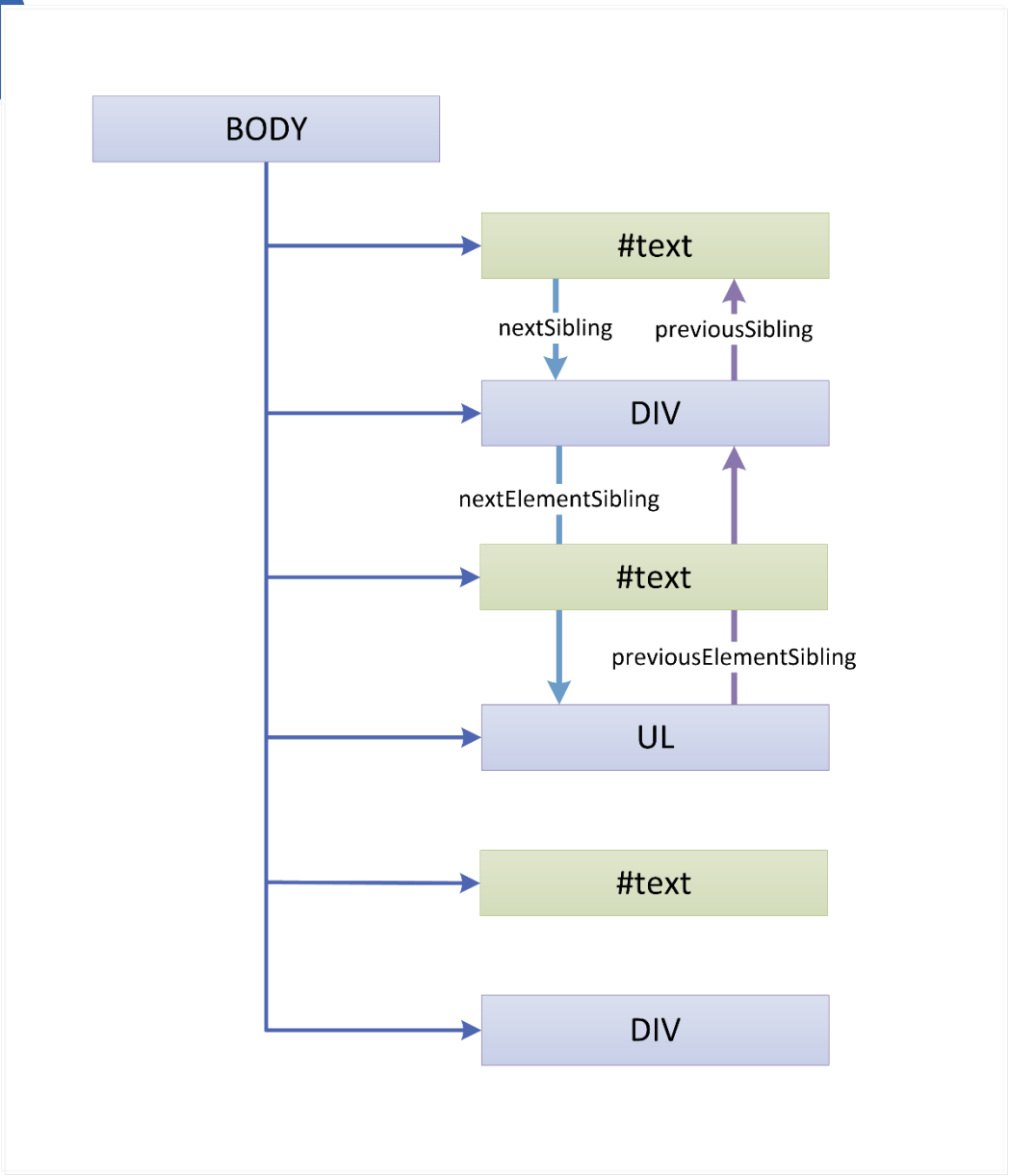



**Рис.4**. Свойства nextElementSibling, previousElementSibling

**example_7**. Свойства nextElementSibling, previousElementSibling

```
<script>
    // nextElementSibling, previousElementSibling – следующий и
    предыдущий html узел (элемент)
    // ссылка на узел body
    let body = document.body;
    console.log(body);
```

```
    // первый дочерний html-узел для body это header
    let header = body.firstElementChild;
    console.log("--> ", header);
    // следующий html-узел за header это div#citate
    let div = header.nextElementSibling;
    console.log("--> ", div);
    // следующий html-узел за div это h3
    let h3 = div.nextElementSibling;
    console.log("--> ", h3);
    // следующий html-узел за h3 это form
    let form = h3.nextElementSibling;
    console.log("--> ", form);
</script>
```

Свойства, позволяющие получить **дочерние узлы** (непосредственно лежат в данном узле):

- **childNodes** — дочерние узлы. К дочерним узлам также относятся текст, комментарии, переносы строки и другие символы
- **children** — дочерние html-узлы (элементы)

### Свойства childNodes, children

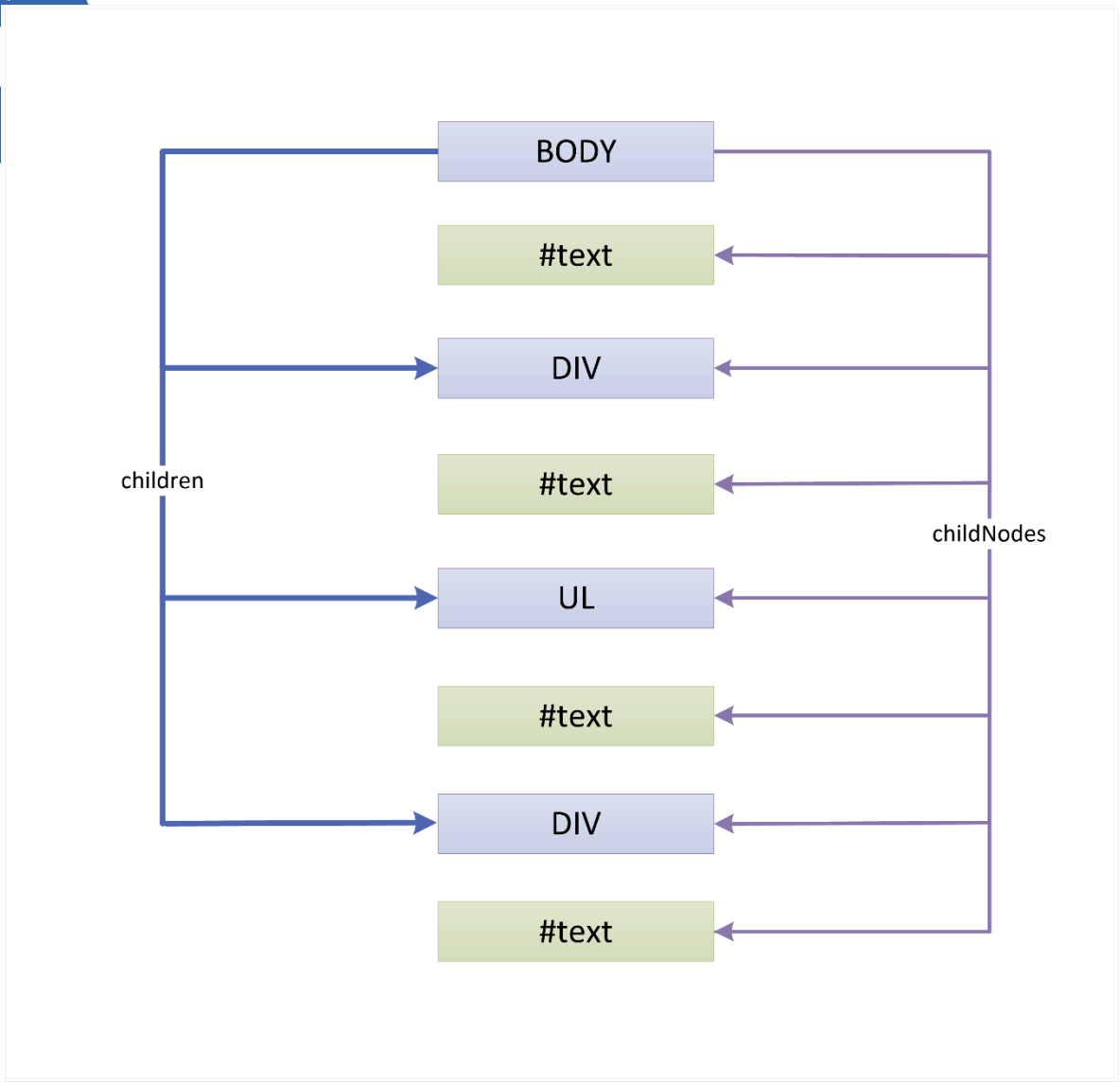



**Рис.5**. Свойства childNodes, children

**example_8**. Свойства childNodes, children

```
<script>
    // 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)
    // ссылка на узел body
    let body = document.body;
```

```
    console.log(body);
    // первый дочерний html-узел для body это header
    let header = body.firstElementChild;
    console.log("--> ", header);
    // следующий html-узел за header это div
    let div = header.nextElementSibling;
    console.log("--> ", div);
    // дочерние узлы для узла div
    let nodes = div.childNodes;
    console.log("--> ", nodes);
    // дочерние html-узлы для узла div
    let html_nodes = div.children;
    console.log("--> ", html_nodes);
</script>
```

До сих пор мы создавали переменные и присваивали каждой переменной (будь это простая переменная, свойство объекта или массив) свой узел. Количество переменных можно сократить, объединив ссылки на дерево узлов в **цепочки вызовов**.

Например строки:

```
let body = document.body;
let header = body.firstElementChild;
let div = header.nextElementSibling;
```

Можно заменить строкой:

```
let div = document.body.firstElementChild.nextElementSibling;
```

**example_9**. Цепочка вызовов

```
<script>
    // последовательное получение доступа к нужному узлу
    let body = document.body;
    let header = body.firstElementChild;
    let div = header.nextElementSibling;
    let h3 = div.nextElementSibling;
    let form = h3.nextElementSibling;
    console.log(form);
    // последовательный доступ можно объединить в цепочку
    вызовов
    // вряд ли это сделает код более читабельным
    form =
    document.body.firstElementChild.nextElementSibling.nextEleme
    ntSibling.nextElementSibling;
    console.log(form);
</script>
```

Свойства, позволяющие получить непосредственного родителя:

- **parentNode** — непосредственный родитель
- **parentElement** — непосредственный родитель (html-элемент)

### Свойства parentNode, parentElement

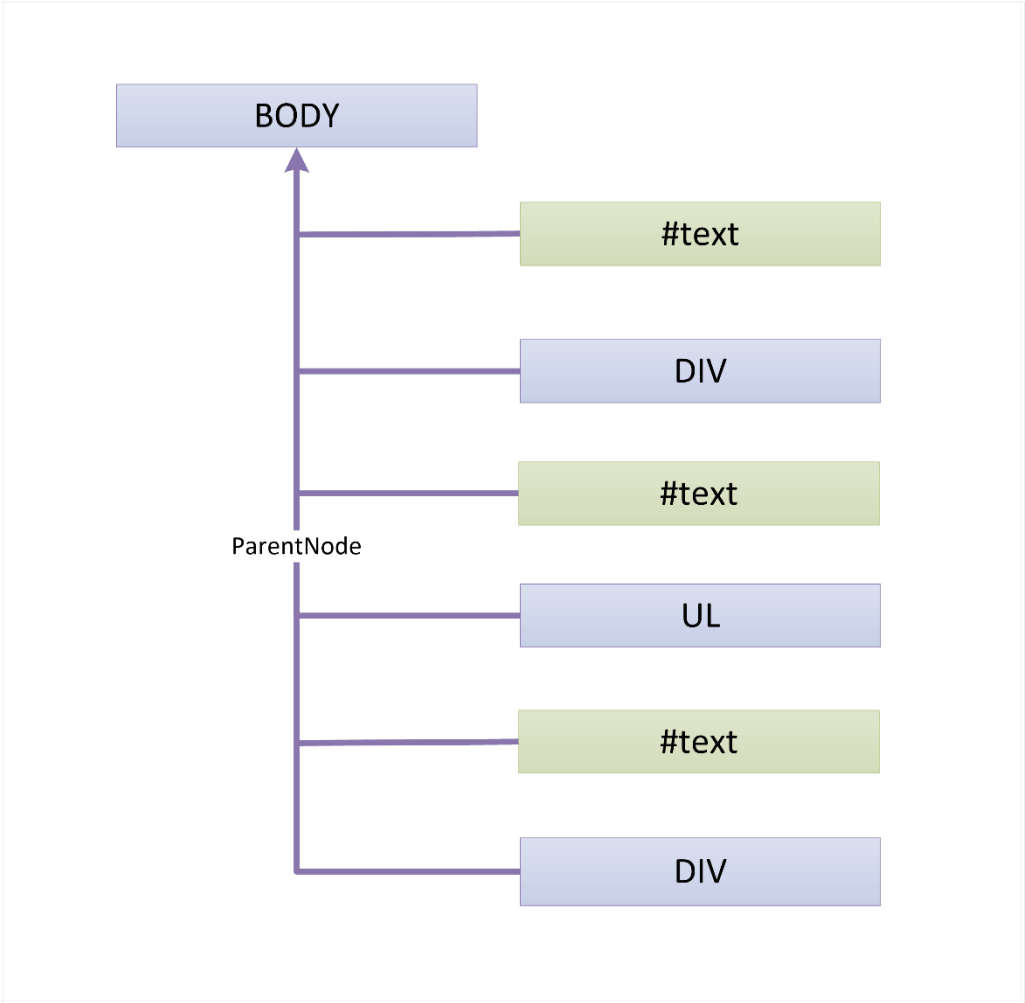



**Рис. 6**. Свойства parentNode, parentElement

**example_10**. Свойства parentNode, parentElement

```
<script>
    // parentNode — свойство, содержащее ссылку на
    непосредственного родителя элемента
    // parentElement  — содержит ссылку на непосредственного
    родителя (html элемент)
    // ссылка на узел body
    let body = document.body;
    console.log(body);
    // первый дочерний html-узел для body это header
```

```
    let header = body.firstElementChild;
    console.log("--> ", header);
    // следующий html-узел за header это div
    let div = header.nextElementSibling;
    console.log("--> ", div);
    // первый дочерний узел для узла div
    let p1 = div.firstElementChild;
    console.log("--> ", p1);
    // следующий за p1 html-узел
    let p2 = p1.nextElementSibling;
    console.log("--> ", p2);
    // родительский элемент для второго абзаца p
    let node = p2.parentNode;
    console.log("--> ", node);
</script>
```

### Метод item

item() - возвращает узел с указанным индексом в DOM.

Существует два способа получить доступ к узлу по указанному индексу:

**Синтаксис**

```
nodes.item(index);
```

или

```
nodes[index];
```

**Параметр**

- **index** - индекс узла в списке, узлы нумеруются по мере их появления в документе. Индекс начинается с 0.

**Возвращаемое значение**

Узел с указанным индексом. **Null** если индекс находится вне диапазона.

Самый простой и распространенный способ – указание индекса в квадратных скобках [index].

**example_11**. Метод item

```
<script>
    // item() - возвращает узел с указанным индексом в DOM
    // ссылка на узел body
    let body = document.body;
    // получаем доступ к узлу div#citate
    let div = body.firstElementChild.nextElementSibling;
    // получаем набор дочерних узлов-элементов (теги <p>)
    let nodes = div.children;
    // вывод в консоль дочерних узлов
    console.log("0 --> ", nodes.item(0));
    console.log("1 --> ", nodes.item(1));
    console.log("2 --> ", nodes.item(2));
</script>
```

Подведем небольшой итог уроку. Создадим скрипт, в котором выберем узлы дерева DOM и запишем в свойства созданного объекта. В свойство объекта массив запишем набор узлов, обозначающих абзацы HTML-документа.

**example_12**. Работа с узлами, объектом, массивом

```
<script>
```

```
    // получаем доступ к корневому элементу
    let body = document.body;
    // создаем пустой объект
    let myObject = {};
    // инициализируем свойства объекта
    myObject.header = body.firstElementChild;
    myObject.div =  body.firstElementChild.nextElementSibling;
    myObject.h3 =
    body.firstElementChild.nextElementSibling.nextElementSibling
    ;
    myObject.form =
    body.firstElementChild.nextElementSibling.nextElementSibling
    .nextElementSibling;
    // получим список дочерних узлов блока div#citate
    elements = myObject.div.children;
    // объявляем свойство-массив объекта
    myObject.array = [];
    // записываем узлы в массив
    myObject.array[0] = elements.item(0);
    myObject.array[1] = elements[1];
    myObject.array[2] = elements[2];
    // вывод свойства объекта в консоль
    console.log(myObject.array);
</script>
```

В **самостоятельных работах** к уроку:

- Происхождение узлов в дереве DOM HTML-документа.

- Навигация по узлам DOM HTML-документа.
- Манипулирование узлами
- Динамический доступ к узлам документа.

---

**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.