ПМ3 Разработка модулей ПО.

РО 3.2 Разрабатывать модули с применением DOM API, Regexp, HTTP

Тема 1. JS: DOM API.

Лекция 3. DOM: структура и навигация.

Цели занятия:

Научиться работать с деревом DOM: понимать его структуру, различать типы узлов и использовать свойства для перемещения между ними.

Учебные вопросы:

1. DOM-дерево: структура и узлы.
2. Навигация по DOM.

1. DOM-дерево: структура и узлы.

- Как HTML превращается в дерево объектов?

Когда браузер получает HTML-файл, он парсит его построчно.

На основе тегов, текста и комментариев создаётся DOM-дерево (Document Object Model).

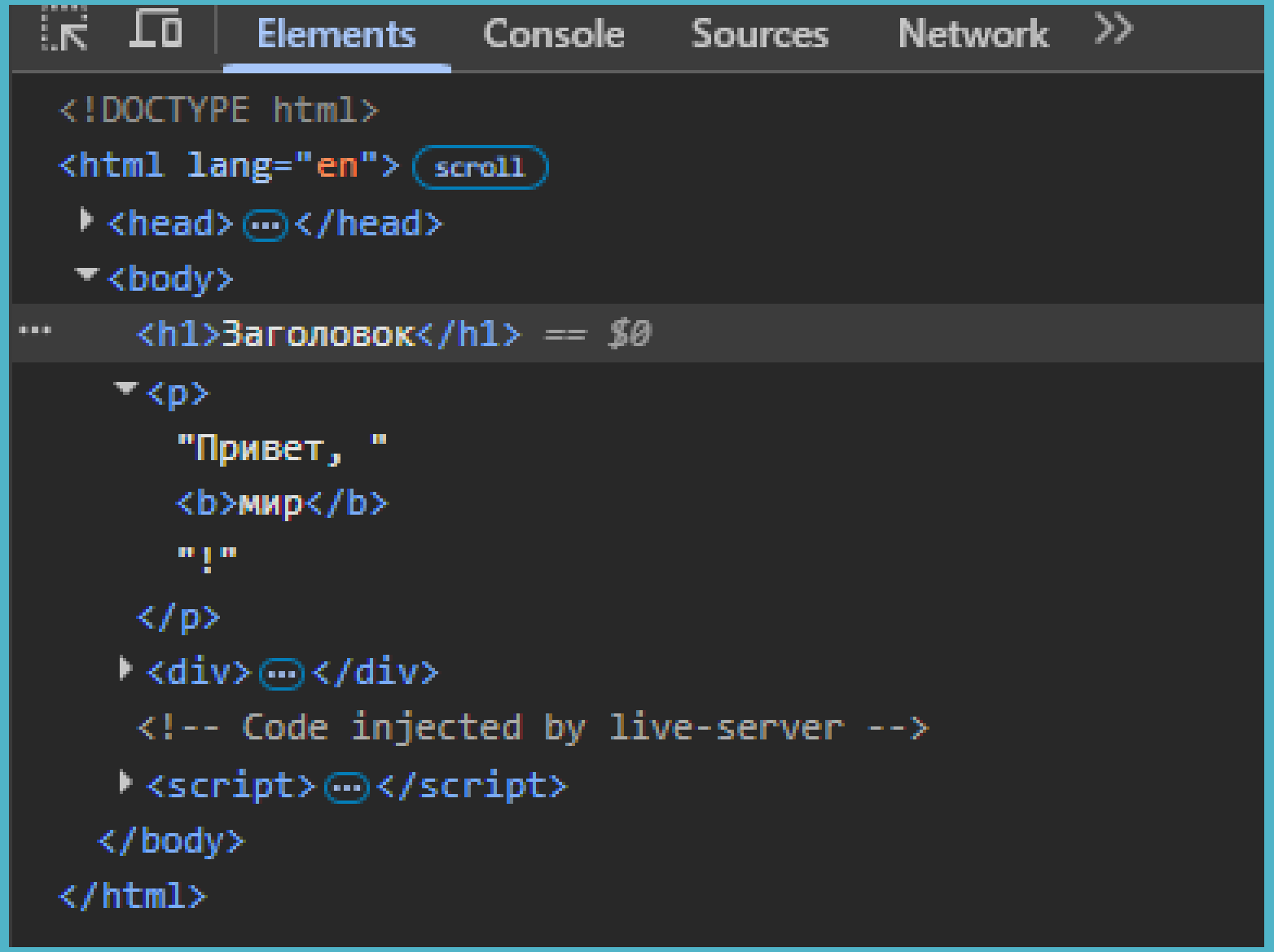
DOM — это не просто текст, а структура объектов, к которым можно обращаться из JavaScript.

Каждый элемент HTML превращается в узел (node) со своими свойствами и связями с другими узлами.

Пример: HTML

```
<body>
    <h1>Заголовок</h1>
    <p>Привет, <b>мир</b>!</p>
    <div>
        <p>text1</p>
        <p>text2</p>
        <p>text3</p>
    </div>
</body>
```

DOM-дерево


Elements
Console
Sources
Network
<!DOCTYPE html>
<html lang="en">
scroll
<head>…</head>
<body>
<h1>Заголовок</h1> == $0

"Привет, "
мир
"!"

…
<!-- Code injected by live-server -->
<script>…</script>
</body>
</html>

◇ Типы узлов.

В DOM всё — это **узлы** (nodes), но они бывают разных типов:

- **Document** — корень документа (доступен как document).
- **Element** — теги (<body>, , <p>).
- **Text** — текст внутри элементов ("Привет").
- **Comment** — комментарии <!-- ... -->.
- (есть и **другие**, но они встречаются реже).

🔷 Свойство nodeType.

Каждый узел имеет числовое свойство nodeType:

1 → Element

3 → Text

8 → Comment

9 → Document

Пример в консоли:

```
<body>
    <h1>Заголовок</h1>
</body>
```

```
3
#text
```

первый ребёнок — перенос строки (текстовый узел).

```
<body><h1>Заголовок</h1></body>
```

```
1
H1
```

первый ребёнок <h1>.

document.body → это <body> ... </body>.

firstChild возвращает самый первый дочерний узел внутри <body>.

Но этот первый узел может быть не элементом, а, например, текстовым узлом (перенос строки или пробел).

◆ Как исследовать DOM в DevTools?

- Открыть панель Elements → увидеть дерево HTML (это и есть DOM).
- Нажать на элемент → он доступен в консоли как $0.
- Ввести в консоли:

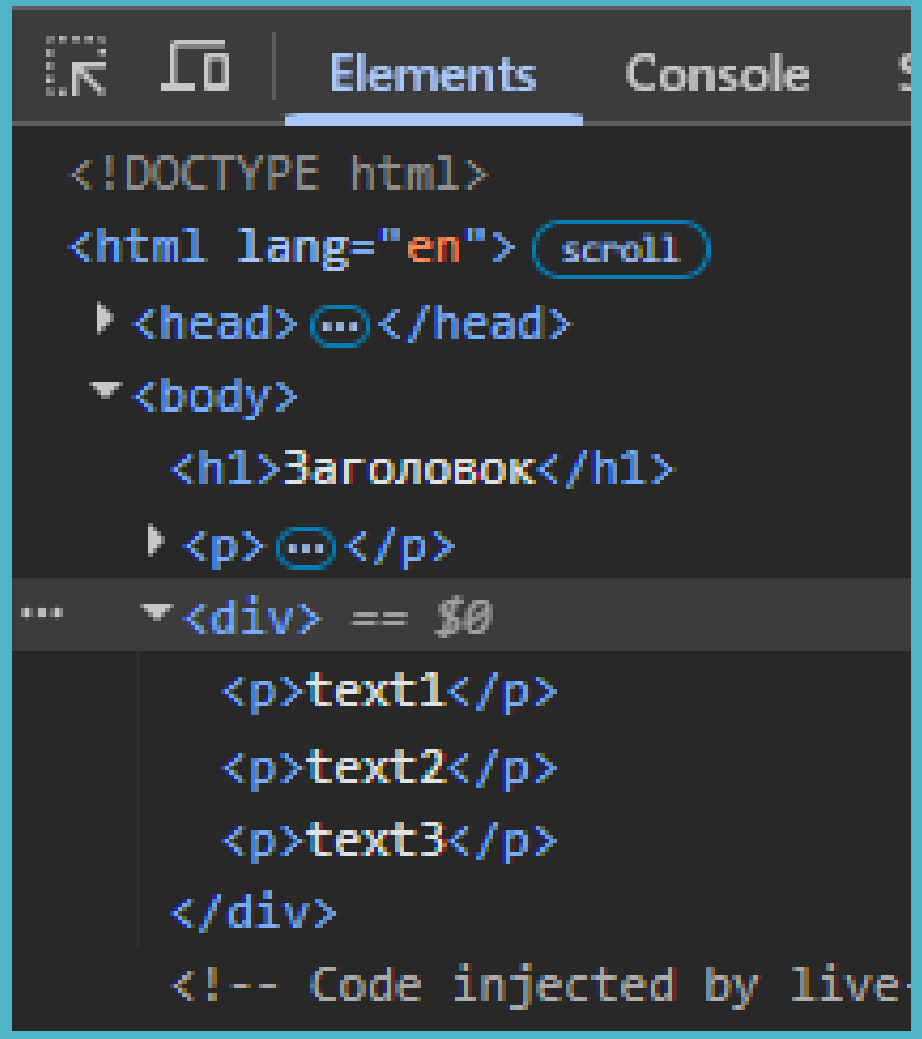
```
console.dir($0);
```

- увидеть объект со всеми свойствами.

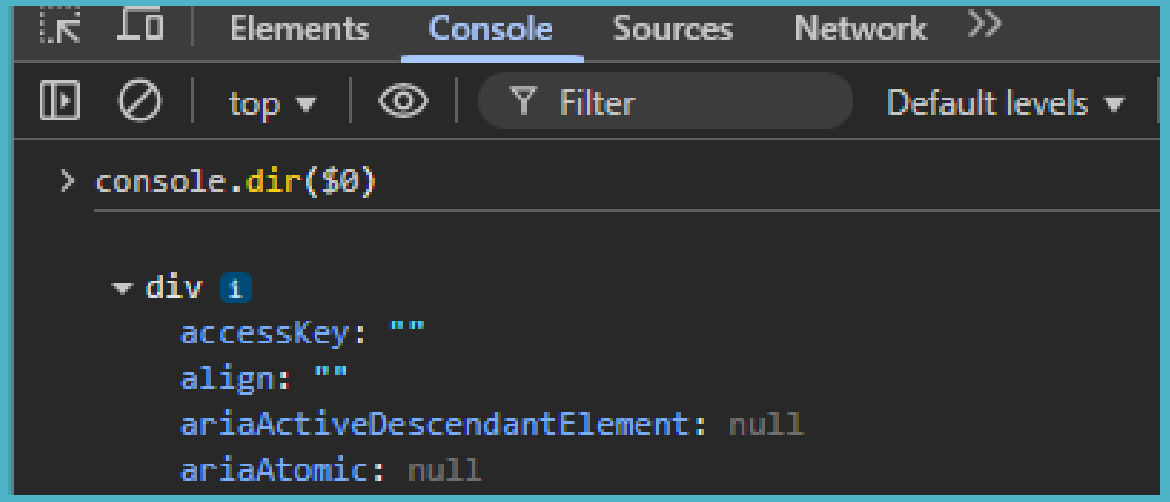
◆ **console.dir()** - Используется для объектного представления. Показывает элемент как JavaScript-объект со всеми свойствами и методами.

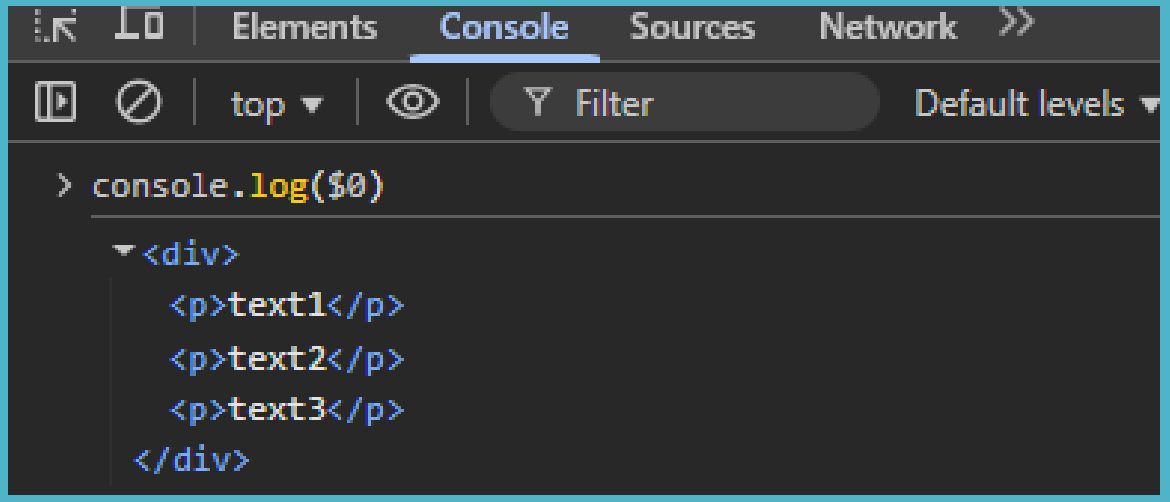
◆ **console.log()** - Используется для «человеческого» вывода. Если передать DOM-элемент, браузер покажет его как HTML-дерево (как в инспекторе).

Пример:


Elements
Console
<!DOCTYPE html>
<html lang="en"> scroll
<head> … </head>
<body>
<h1>Заголовок</h1>
 …
 == $0
text1
text2
text3

<!-- Code injected by live


Elements
Console
Sources
Network
top
Filter
Default levels
> console.dir($0)
div i
accessKey: ""
align: ""
ariaActiveDescendantElement: null
ariaAtomic: null


Elements
Console
Sources
Network
top
Filter
Default levels
> console.log($0)

text1
text2
text3

✅ Вывод:

- DOM — это дерево объектов, созданное из HTML.
- Всё в DOM называется узлами, но узлы бывают разных типов.
- nodeType помогает определить, какой именно это узел: элемент, текст или комментарий.

2. Навигация по DOM.

DOM-дерево устроено как «семейное дерево»: у каждого узла есть родители, дети и соседи.

JS даёт свойства для перемещения по этой структуре.

◇ Родители.

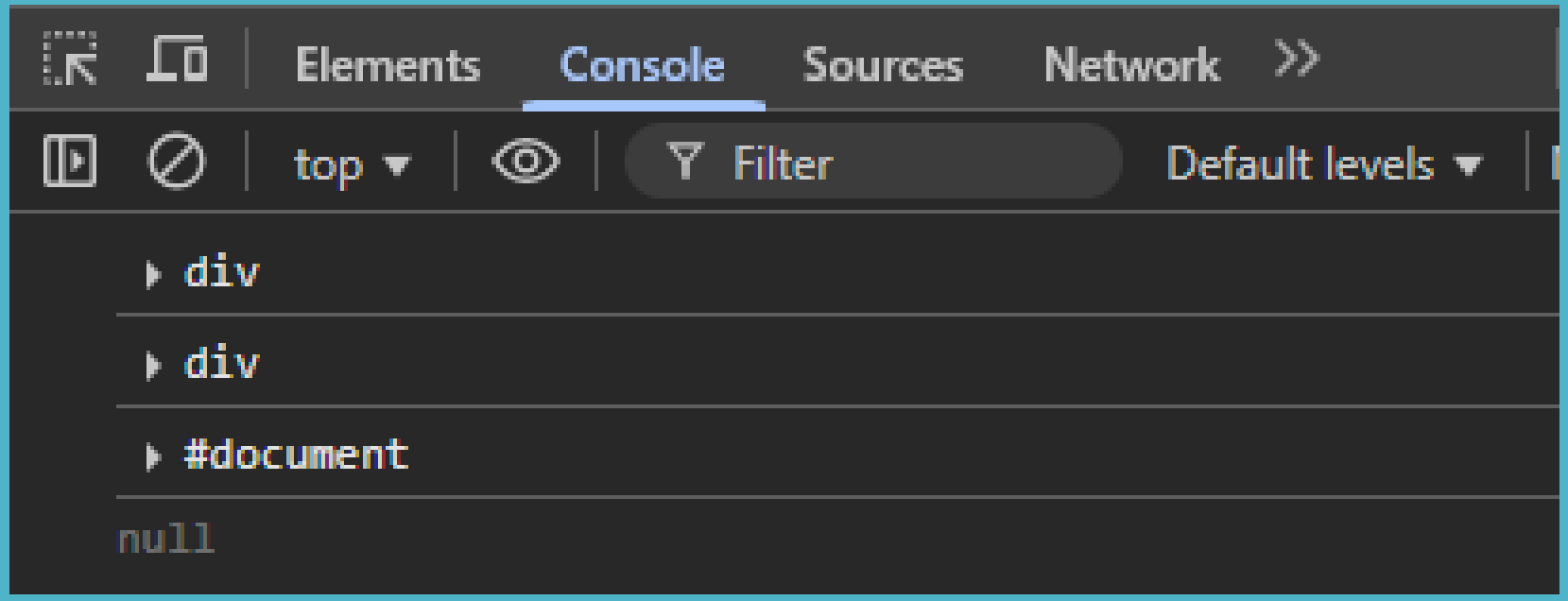
- **parentNode** → возвращает родителя любого узла (элемент, документ).
- **parentElement** → возвращает только родителя-элемент (если родитель документ (например у <html>) — будет null).

Пример:

```
<body>
    <div>
        <p id="p1">text1</p>
        <p id="p2">text2</p>
        <p id="p3">text3</p>
    </div>
</body>
```

```
let p = document.querySelector("#p1");
console.log(p.parentNode);
console.log(p.parentElement);

let html = document.querySelector("html");
console.log(html.parentNode);
console.log(html.parentElement);
```



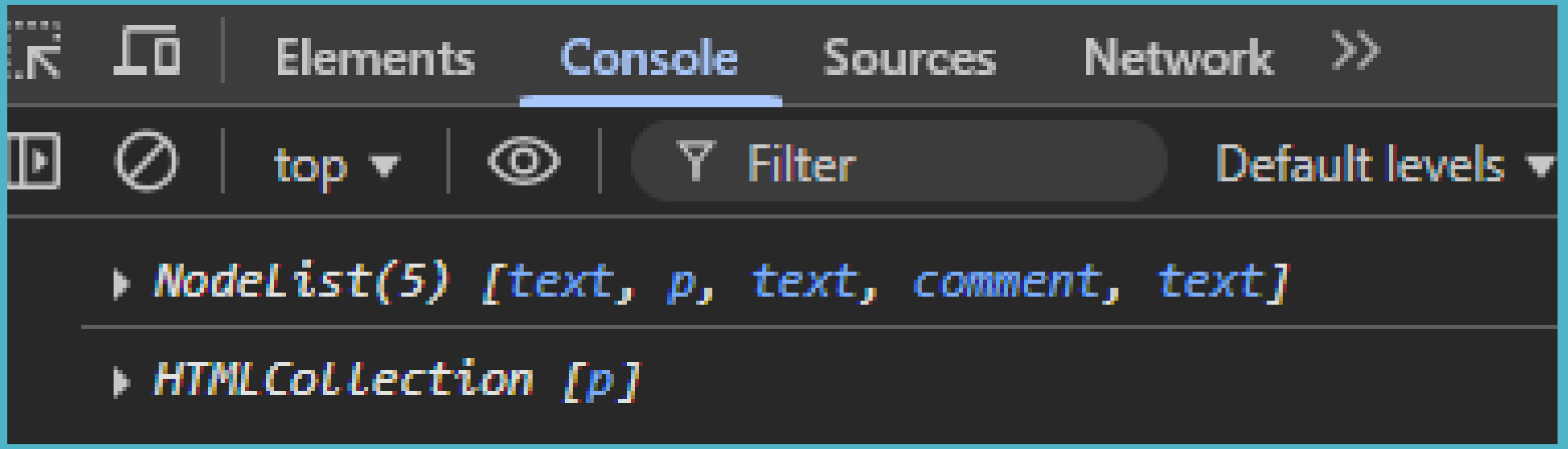
◇ Дети.

- childNodes → все дочерние узлы (включая текстовые и комментарии).
- children → только элементы (HTML-теги).

Пример:

```
<body>
    <div id="box">
        Привет!
        <p>text1</p>
        <!--  комментарий! -->
    </div>
</body>
```

```
let el = document.getElementById("box");
console.log(el.childNodes);
console.log(el.children);
```

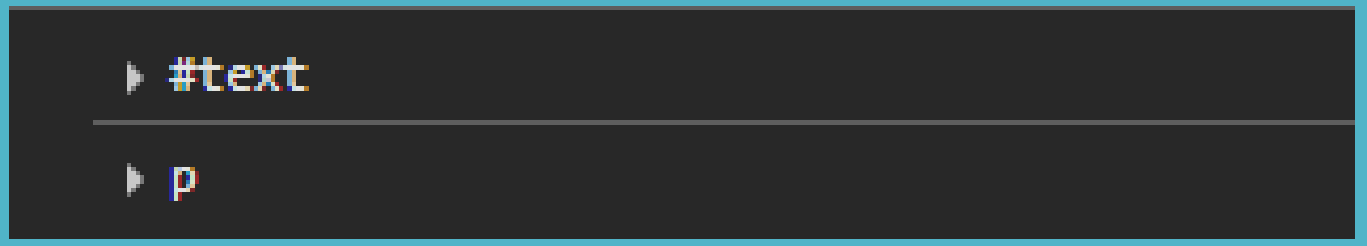


◆ Первый и последний ребёнок

- firstChild / lastChild → могут быть любым узлом (текст, комментарий, элемент).
- firstElementChild / lastElementChild → гарантированно возвращают элементы.

Пример:

```
let el = document.getElementById("box");
console.log(el.firstChild);
console.log(el.firstElementChild);
```



◇ Соседи

- nextSibling / previousSibling → любой сосед (включая текст и комментарии).
- nextElementSibling / previousElementSibling → только сосед-элемент.

Пример:

```
<body>
    <h1> Title </h1>
    <div id="box">
        Привет!
        <p>text1</p>
        <!--  комментарий! -->
    </div>
</body>
```

```
let h1 = document.querySelector("h1");
console.log(h1.nextSibling);
console.log(h1.nextElementSibling);
```

```
▸ #text
▸ div#box
```

✅ Вывод:

- Для «сырых» узлов используем childNodes, firstChild, nextSibling.
- Для только элементов — children, firstElementChild, nextElementSibling.
- Часто на практике используют именно «Element»-варианты, чтобы игнорировать текстовые узлы и переносы строк.

Контрольные вопросы:

- Какие бывают типы узлов в DOM?
- Чем отличаются childNodes и children?
- Как найти родителя или соседа у элемента?

Домашнее задание:

1. https://ru.hexlet.io/courses/js-dom