

ПМЗ Разработка модулей ПО.

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

Тема 1. JS: DOM API.

Лекция 27. 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.**
- 3. Исследование структуры в консоли.**
- 4. Практика обхода 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).

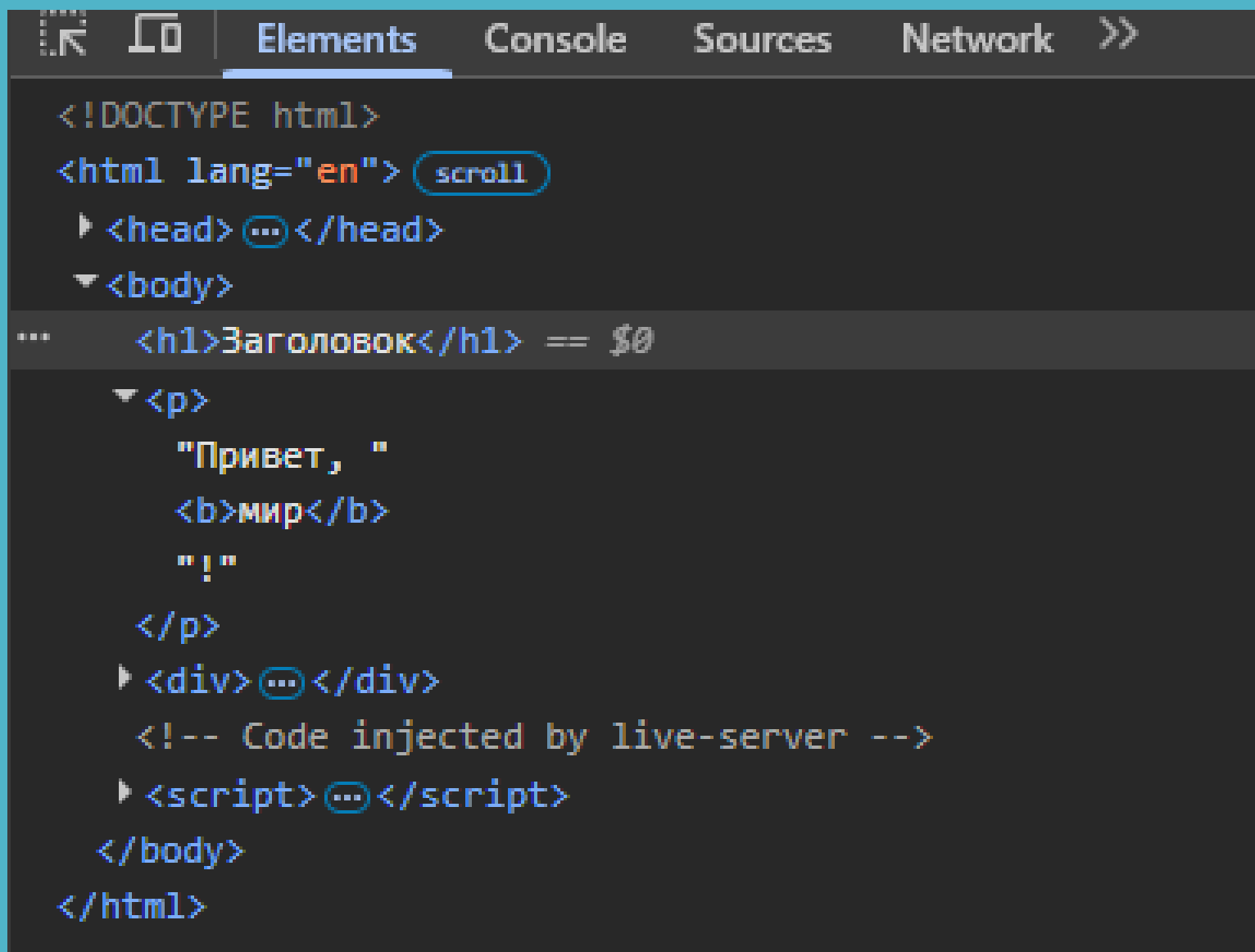
DOM — это не просто текст, а структура объектов, к которым можно обращаться из JavaScript.

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



```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>
      "Привет, "
      <b>мир</b>
      "!"
    </p>
    <div>
    <!-- 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>, <div>, <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">
  <head>
  <body>
    <h1>Заголовок</h1>
    <p>
    <div> == $0
      <p>text1</p>
      <p>text2</p>
      <p>text3</p>
    </div>
    <!-- Code injected by live-
```

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

```
Elements Console Sources Network
top | | Filter Default levels
> console.log($0)
<div>
  <p>text1</p>
  <p>text2</p>
  <p>text3</p>
</div>
```

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.

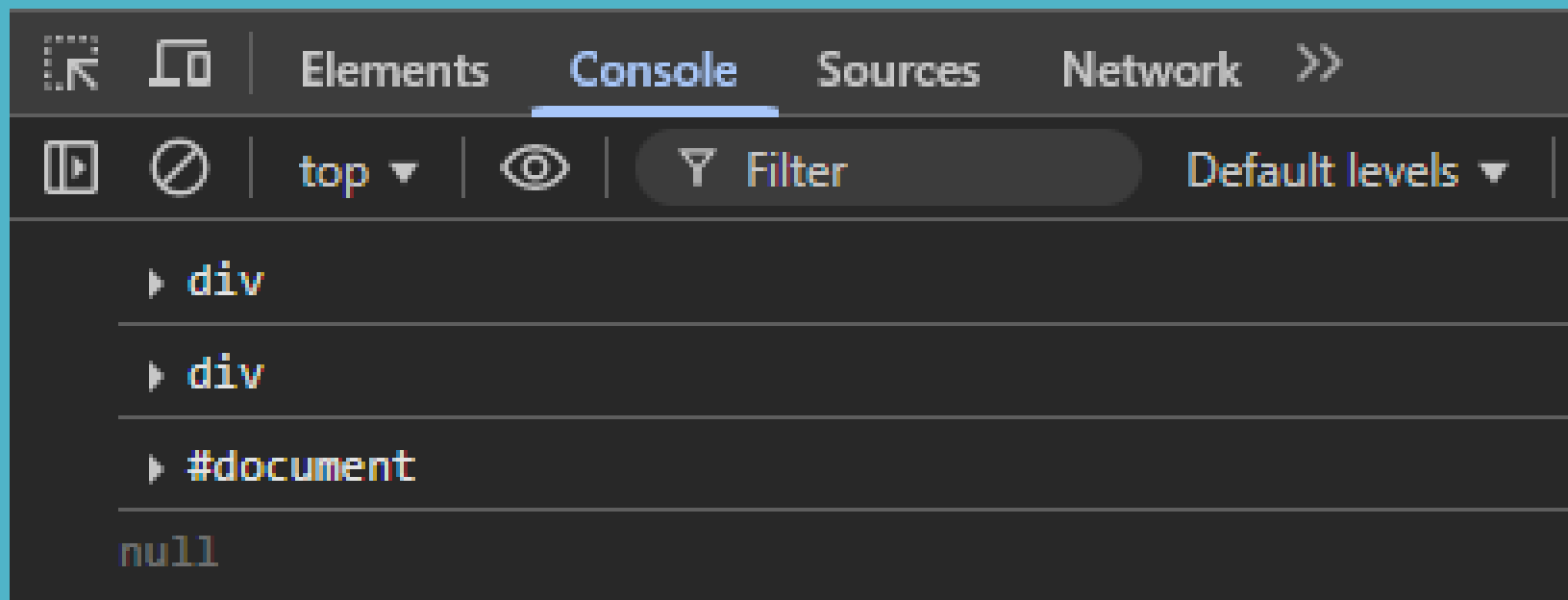
- **parentNode** → возвращает родителя любого узла (элемент, документ).
- **parentElement** → возвращает только родителя-элемент (если родитель документ (например у `<html>`) — будет `null`).

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);
```



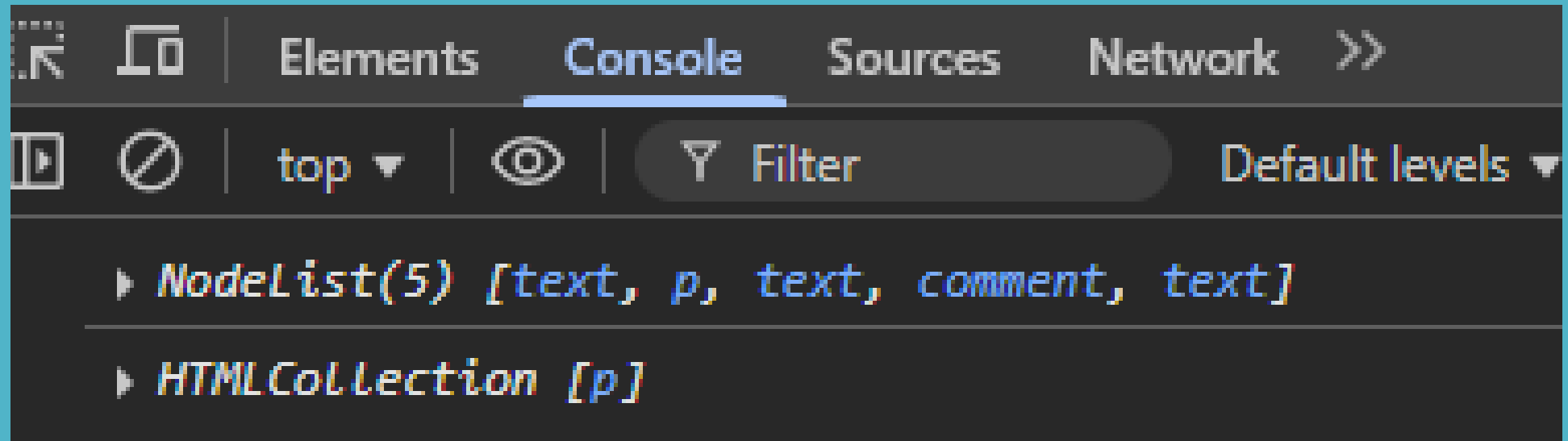
◆ Дети.

- `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);
```

```
▶ #text
```

```
▶ p
```

◆ 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.

3. Исследование структуры в консоли.

Современные браузеры предоставляют мощные инструменты разработчика (DevTools), которые позволяют изучать и отлаживать DOM прямо во время работы страницы.

◆ `console.log()` vs `console.dir()`

`console.log(node)`

- Показывает DOM-узел как HTML-структуру, похоже на инспектор элементов.
- 👉 Удобно, если нужно «увидеть» тег.

`console.dir(node)`

- Показывает объектную модель узла с его свойствами и методами.
- 👉 Удобно, если нужно изучить API: `children`, `classList`, `textContent` и др.

Пример:

```
let el = document.body;  
console.log(el); // <body>...</body>  
console.dir(el); // объект с кучей свойств и методов
```

◆ Специальные переменные DevTools

В консоли браузера есть «шорткаты» для работы с выбранными элементами:

- **\$0** → последний выделенный элемент в панели Elements.
- **\$1, \$2, ...** → более ранние выделенные элементы (история выбора).
- **\$_** → результат последнего выражения в консоли.
- **\$\$("selector")** → сокращение для `document.querySelectorAll()`.

Примеры:

✓ **\$0**. Последний выделенный элемент в панели Elements.

```
$0.style.border = "2px solid red"; // подсветить выделенный элемент  
'2px solid red'
```

✓ **\$1, \$2, ...** История выбора в Elements: **\$1** — предпоследний, **\$2** — ещё раньше.

```
> console.log($0); // текущий
```

```
  > <div id="box" style="border: 2px solid red;">...</div>
```

```
< undefined
```

```
> console.log($1); // предыдущий
```

```
  <p>text1</p>
```

✓ `$_` - Хранит результат последнего выражения в консоли.

```
> 2 + 3
```

```
<- 5
```

```
> $_ * 10
```

```
<- 50
```

✓ `$()` → сокращение для `document.querySelector()`.

```
> $("h1")  
← <h1> Title </h1>
```

✓ `$$("selector")` - Сокращение для `document.querySelectorAll()`. Возвращает массивоподобную коллекцию.

```
> $$("p");  
← ▶ (2) [p, p]
```

Вывод:

- **console.log()** показывает DOM как HTML.
- **console.dir()** раскрывает объектные свойства и методы.
- **\$0 ... \$4** — быстрый доступ к выбранным элементам.
- **_** — результат последнего выражения.
- **\$(), \$\$()** — сокращения для поиска по селекторам.

4. Практика обхода DOM.

Обход DOM-дерева нужен, чтобы анализировать структуру страницы, искать элементы и менять их свойства.

◆ Перебор `children` и `childNodes`

`children` → только элементы.

`childNodes` → все узлы (элементы, текст, комментарии).

Пример:

```
let box = document.getElementById("box");

console.log("children:");
for (let el of box.children) {
  console.log(el); // только <p>, <div> и т.п.
}

console.log("childNodes:");
for (let node of box.childNodes) {
  console.log(node); // текстовые узлы, комментарии и элементы
}
```

children:

▶ p

childNodes:

▶ #text

▶ p

▶ #text

▶ #comment

▶ #text

◆ Цикл for...of для детей

Так как children и childNodes — коллекции, их можно перебирать for...of:

```
for (let el of document.body.children) {  
  console.log("Элемент:", el.tagName);  
}
```

Элемент: H1

Элемент: DIV

Элемент: SCRIPT

Элемент: H1

Элемент: P

◆ Рекурсивный обход дерева

Рекурсия позволяет обойти всё дерево, а не только один уровень.

Пример, код выведет имена всех узлов в документе:

```
function walk(node) {  
  console.log(node.nodeName);  
  for (let child of node.childNodes) {  
    walk(child); // рекурсивный вызов  
  }  
}  
  
walk(document.body);
```

◆ Подсветка узлов через JS

Меняем стиль найденных элементов:

```
for (let el of document.body.children) {  
  | el.style.outline = "2px solid blue";  
}
```

Выводы:

- `children` → элементы, `childNodes` → все узлы.
- `for...of` удобен для перебора коллекций DOM.
- Рекурсия нужна для обхода всего дерева.
- Подсветка стилями помогает визуально изучить DOM-структуру.

Выводы по теме лекции:

- DOM = объектное представление HTML в виде дерева узлов.
- Узлы бывают разных типов: элементы, текст, комментарии и др.
- Для навигации используются свойства:
 - родители → parentNode, parentElement;
 - дети → childNodes, children;
 - соседи → nextSibling, previousSibling, nextElementSibling.
- console.log() и console.dir() помогают исследовать DOM, а в DevTools есть шорткаты \$0, \$_, \$\$().
- DOM можно обходить циклами и рекурсией, а подсветка стилями помогает визуально «увидеть» структуру дерева.

Контрольные вопросы:

- Какие бывают типы узлов в DOM?
- Чем отличаются `childNodes` и `children`?
- Как найти родителя или соседа у элемента?
- В чём разница между `nextSibling` и `nextElementSibling`?
- Как можно пройти циклом по всем дочерним элементам `body`?

Домашнее задание:

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

5 DOM

Выясняем, чем HTML отличается от DOM

6 Навигация по DOM-дереву

Знакомимся с структурой DOM-дерева и учимся перемещаться по ней

7 Декларативный поиск по DOM-дереву

Знакомимся с `getElementById`, `querySelector` и другими поисковыми методами

8 Консоль разработчика

Материалы лекций:

<https://github.com/ShViktor72/Education2025>

Обратная связь:

colledge20education23@gmail.com