# 4. Навигация по DOM-элементам

DOM позволяет делать что угодно с HTML-элементом и его содержимым, но для этого нужно сначала нужный элемент получить.
Доступ к DOM начинается с объекта `document`. Из него можно добраться до любых узлов.

## Сверху documentElement и body

Самые верхние элементы дерева доступны напрямую из `document`.
**`<HTML>` = `document.documentElement`**
Первая точка входа – `document.documentElement`. Это свойство ссылается на DOM-объект для тега `<html>`.
**`<BODY>` = `document.body`**
Вторая точка входа – `document.body`, который соответствует тегу `<body>`.

## Дети: childNodes, firstChild, lastChild

Здесь и далее мы будем использовать два принципиально разных термина.

- **Дочерние элементы (или дети)** – элементы, которые лежат *непосредственно* внутри данного. Например, внутри `<HTML>` обычно лежат `<HEAD>` и `<BODY>`.
- **Потомки** – все элементы, которые лежат внутри данного, вместе с их детьми, детьми их детей и так далее. То есть, всё поддерево DOM.

Псевдо-массив `childNodes` хранит все дочерние элементы, включая текстовые.

## Коллекции – не массивы

DOM-коллекции, такие как `childNodes` и другие, которые мы увидим далее, не являются JavaScript-массивами.
В них нет методов массивов, таких как `forEach`, `map`, `push`, `pop` и других.

## Навигация только по элементам

Навигационные ссылки, описанные выше, равно касаются всех узлов в документе. В частности, в `childNodes` сосуществуют и текстовые узлы и узлы-элементы и узлы-комментарии, если есть.

Эти ссылки похожи на те, что раньше, только в ряде мест стоит слово `Element`:

- `children` – только дочерние узлы-элементы, то есть соответствующие тегам.
- `firstElementChild`, `lastElementChild` – соответственно, первый и последний дети-элементы.
- `previousElementSibling`, `nextElementSibling` – соседи-элементы.
- `parentElement` – родитель-элемент.

В DOM доступна навигация по соседним узлам через ссылки:

- По любым узлам.
- Только по элементам.

# 5. Поиск: getElement*, querySelector*

**document.getElementById или просто id**
Если у элемента есть атрибут `id`, то мы можем получить его вызовом `document.getElementbyId(id)`, где бы он ни находился.

Значение `id` должно быть уникальным. В документе может быть только один элемент с данным `id`.
Если в документе есть несколько элементов с одинаковым значением `id`, то поведение методов поиска непредсказуемо. Браузер может вернуть любой из них случайным образом. Поэтому, пожалуйста, придерживайтесь правила сохранения уникальности `id`.

Метод `getElementById` можно вызвать только для объекта `document`. Он осуществляет поиск по `id` по всему документу.

**querySelectorAll**
Самый универсальный метод поиска – это `elem.querySelectorAll(css)`, он возвращает все элементы внутри `elem`, удовлетворяющие данному CSS-селектору.
Следующий запрос получает все элементы `<li>`, которые являются последними потомками в `<ul>`

Псевдоклассы в CSS-селекторе, в частности `:hover` и `:active`, также поддерживаются. Например, `document.querySelectorAll(':hover')` вернёт коллекцию (в порядке вложенности: от внешнего к внутреннему) из текущих элементов под курсором мыши.

**querySelector**
Метод `elem.querySelector(css)` возвращает первый элемент, соответствующий данному CSS-селектору.
Иначе говоря, результат такой же, как при вызове `elem.querySelectorAll(css)[0]`, но он сначала найдёт *все* элементы, а потом возьмёт первый, в то время как `elem.querySelector` найдёт только первый и остановится. Это быстрее, кроме того, его короче писать.

**matches**
Предыдущие методы искали по DOM.
Метод elem.matches(css) ничего не ищет, а проверяет, удовлетворяет ли `elem` CSS-селектору, и возвращает `true` или `false`.
Этот метод удобен, когда мы перебираем элементы (например, в массиве или в чём-то подобном) и пытаемся выбрать те из них, которые нас интересуют.

**closest**
*Предки* элемента – родитель, родитель родителя, его родитель и так далее. Вместе они образуют цепочку иерархии от элемента до вершины.
Метод `elem.closest(css)` ищет ближайшего предка, который соответствует CSS-селектору. Сам элемент также включается в поиск.
Другими словами, метод `closest` поднимается вверх от элемента и проверяет каждого из родителей. Если он соответствует селектору, поиск прекращается. Метод возвращает либо предка, либо `null`, если такой элемент не найден.