Работа с DOM

Кирилл Талецкий

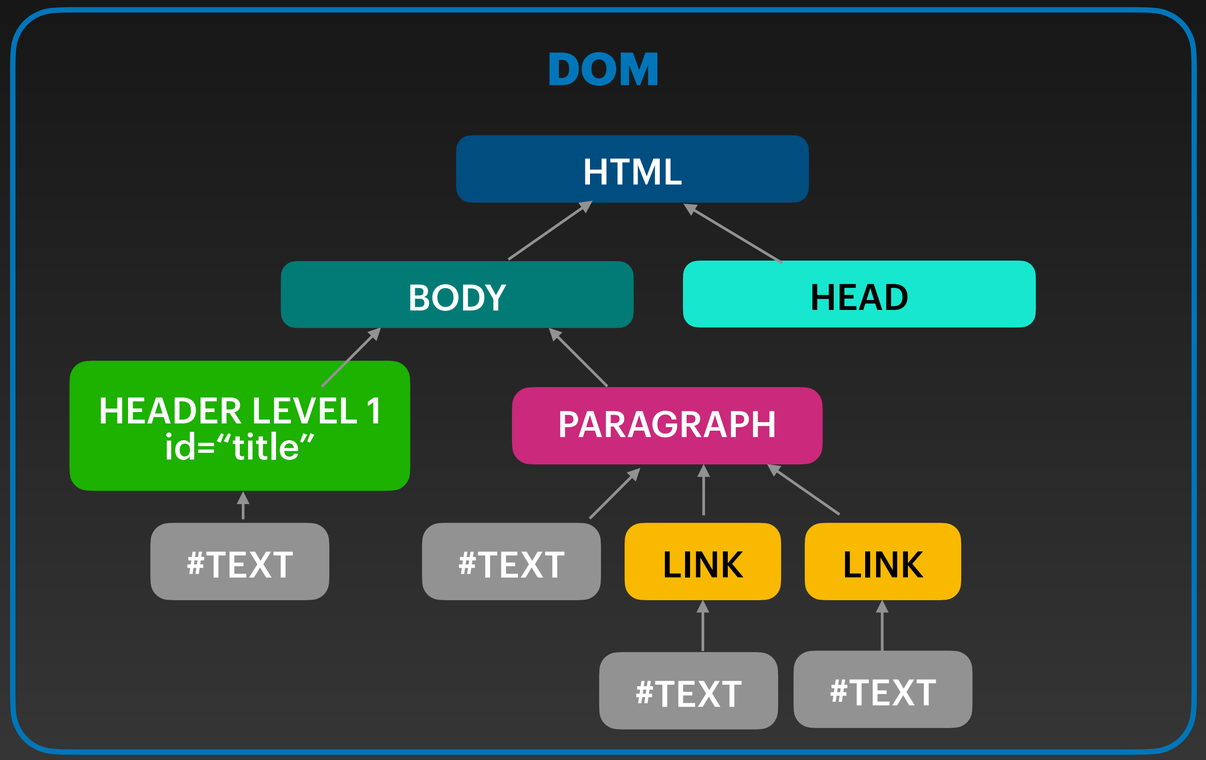
TeachMeSkills
28 августа 2023

Пример из домашки

Поиск по DOM дереву

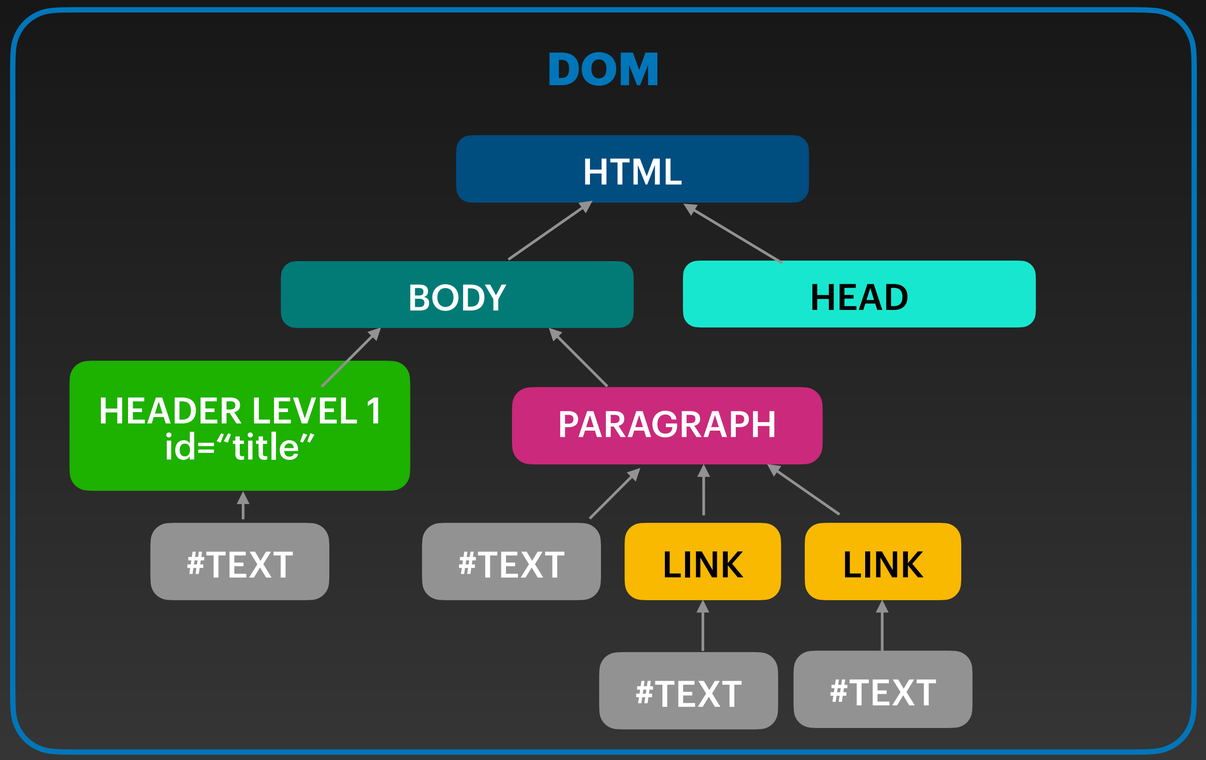
Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```



Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```



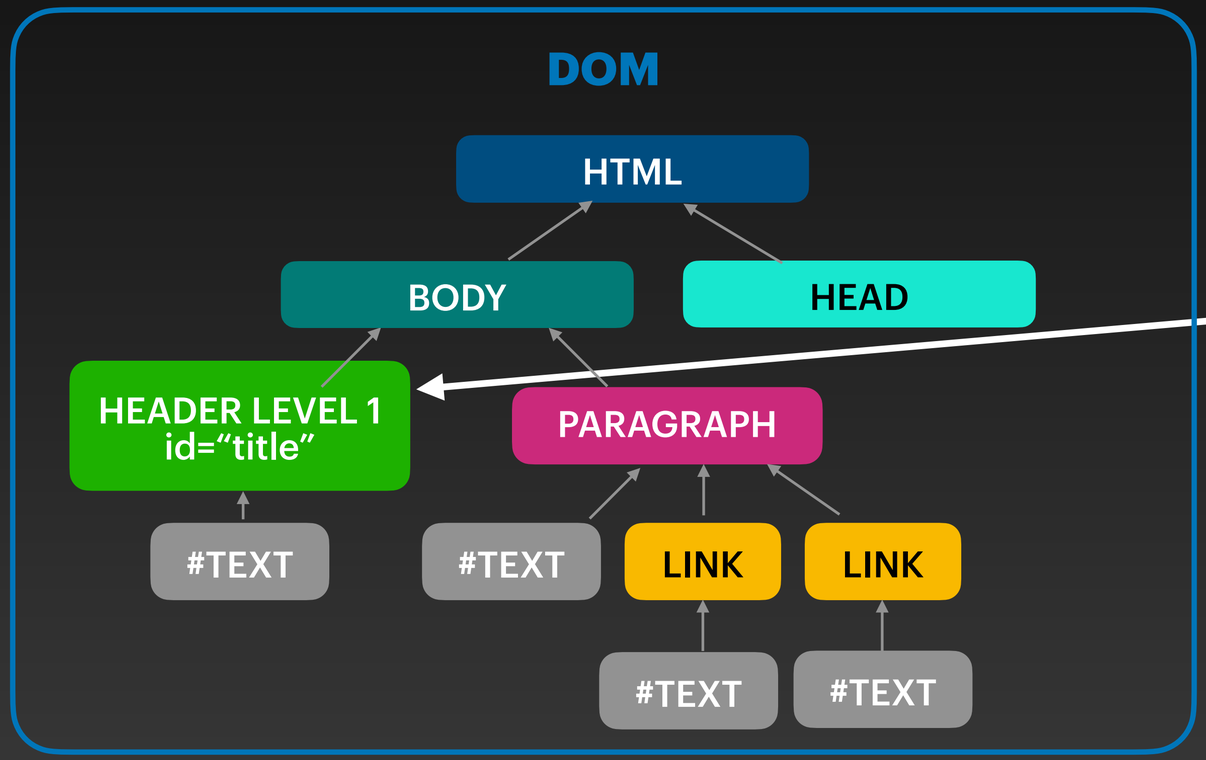

```
document.getElementById("title");

body.querySelectorAll(“p > a");

body.querySelector("p > a");
```

Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```



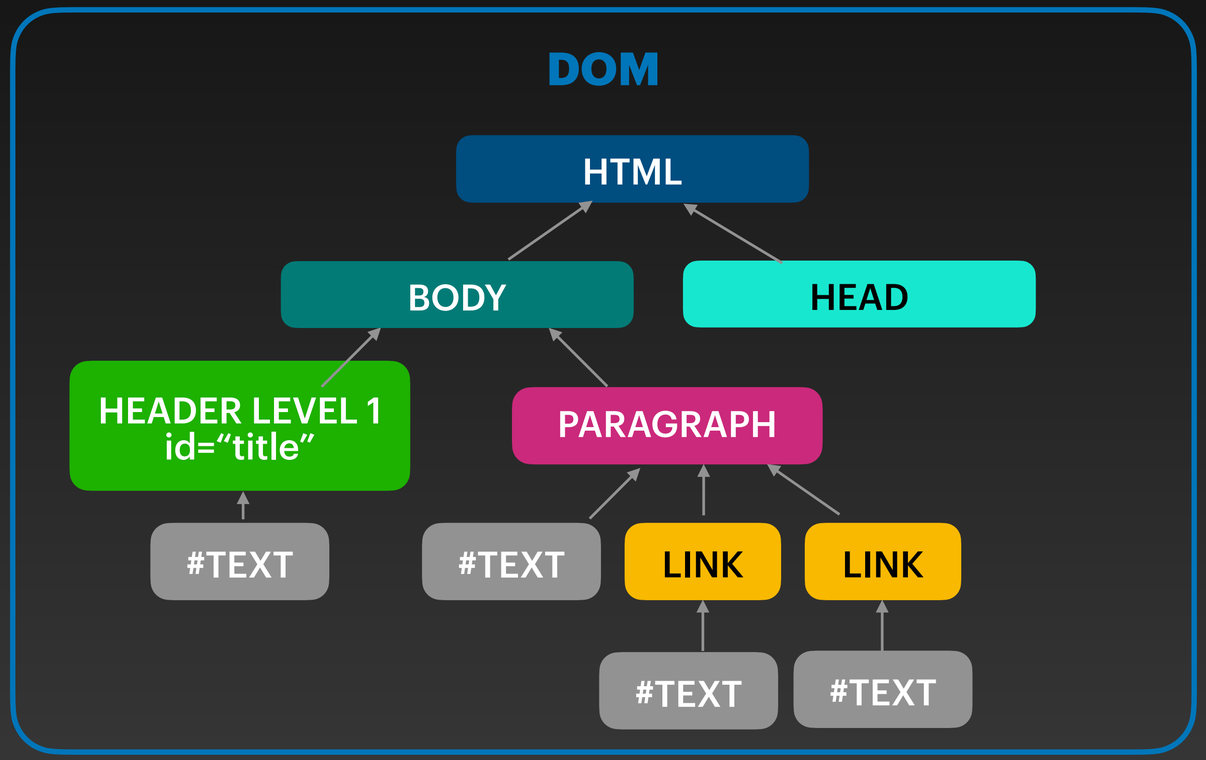

```
document.getElementById("title");

body.querySelectorAll("p > a");

body.querySelector("p > a");
```

Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```

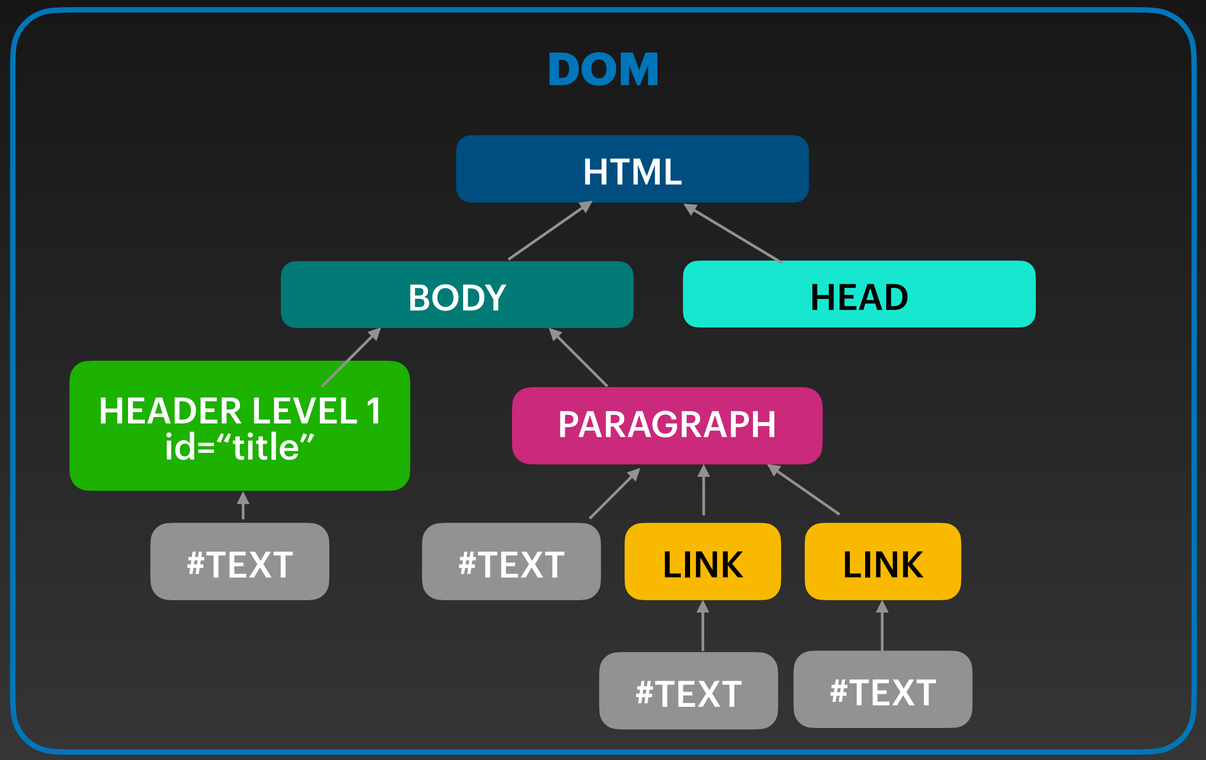



```
document.getElementById("title");

body.querySelectorAll("p > a");

body.querySelector("p > a");
```

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```



Поиск по DOM

Что выбирает селектор `p > a` в CSS?

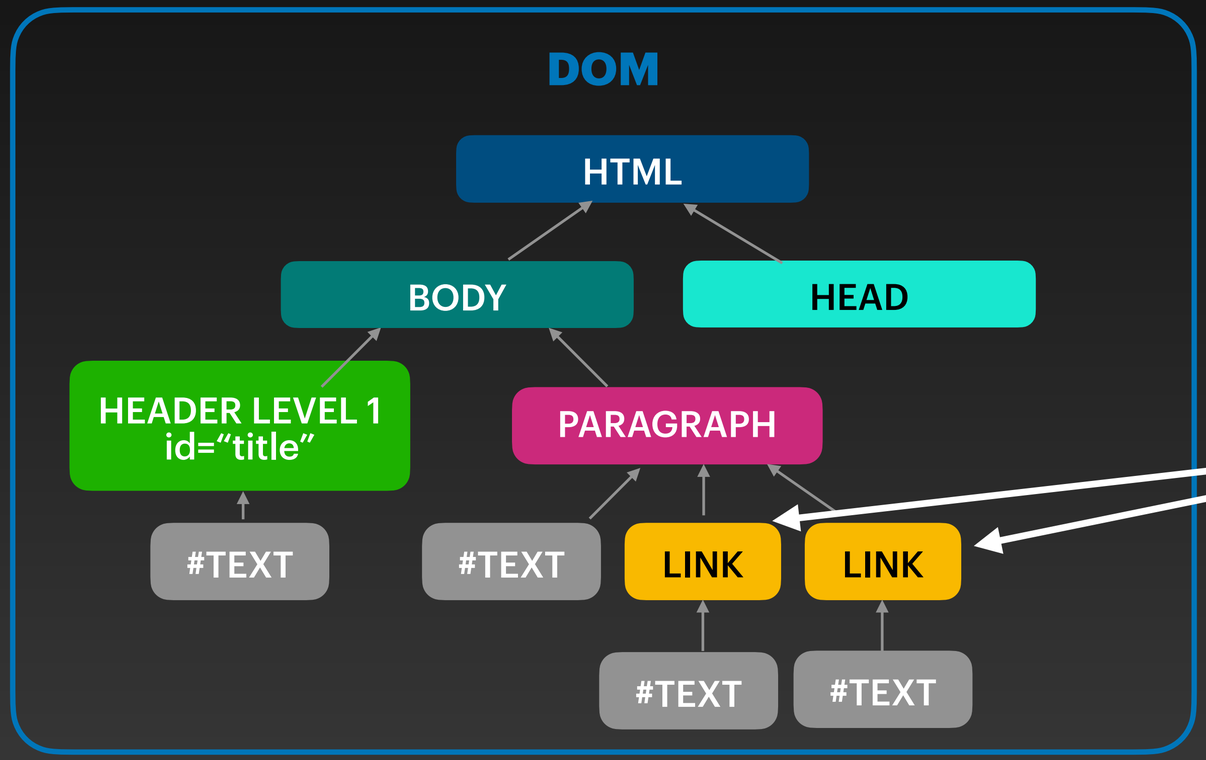
```
document.getElementById("title");

body.querySelectorAll("p > a");

body.querySelector("p > a");
```

Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```



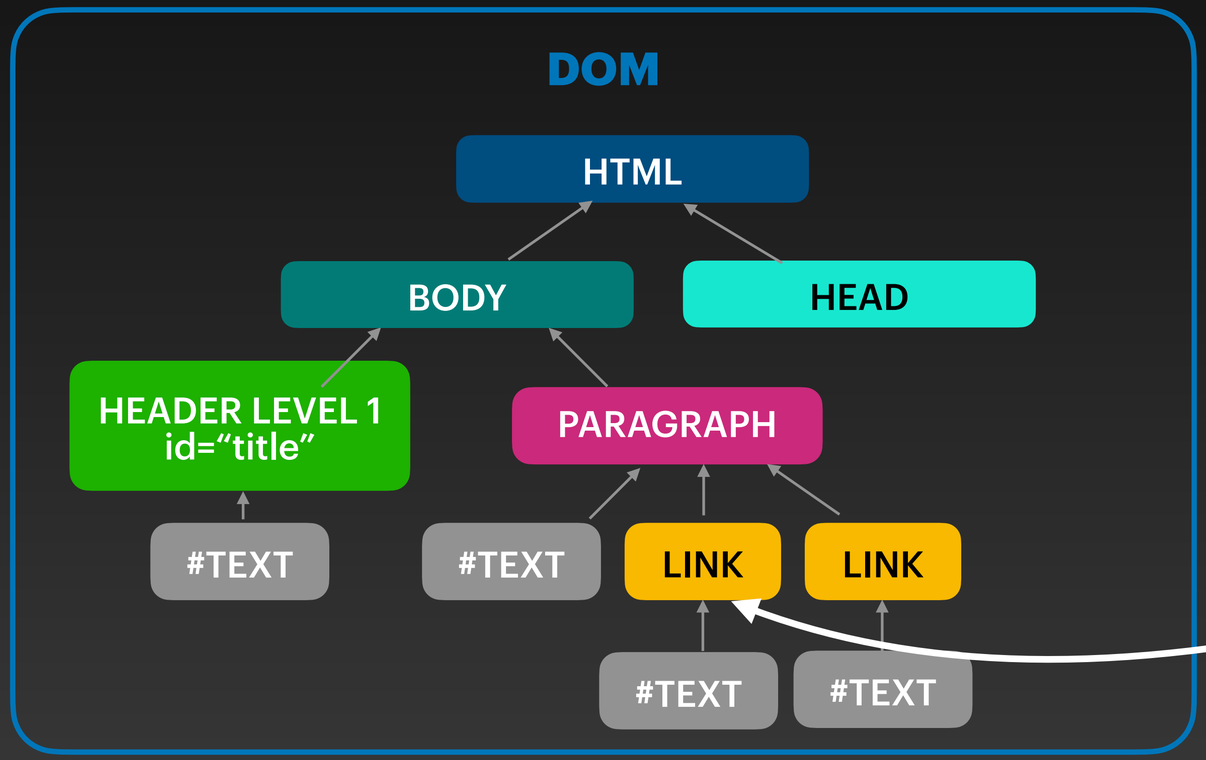

```
document.getElementById("title");

body.querySelectorAll("p > a");

body.querySelector("p > a");
```

Поиск по DOM

```
<body>
  <h1 id="title">Hello world!</h1>
  <p>
    This is some paragraph text. It is
    long and it has a <a>link</a> inside
    of <a>it</a>
  </p>
</body>
```




```
document.getElementById("title");

body.querySelectorAll("p > a");

body.querySelector("p > a");
```

Все методы поиска произвольных элементов

Метод	Ищет по...	Ищет внутри элемента	Возвращает живую коллекцию?
querySelector()	CSS селектор	✅	–
querySelectorAll()	CSS селектор	✅	–
getElementById	id	–	–
getElementByName	name	–	✅
getElementByTagName	Название тэга или * (любое название)	✅	✅
getElementByClassName	Имя класса	✅	✅

Поиск по DOM

- Поиск по цепочке предков

```
elem.closest('body > p');
```

- Проверка на вложенность

```
elem.contains(elem2);
```

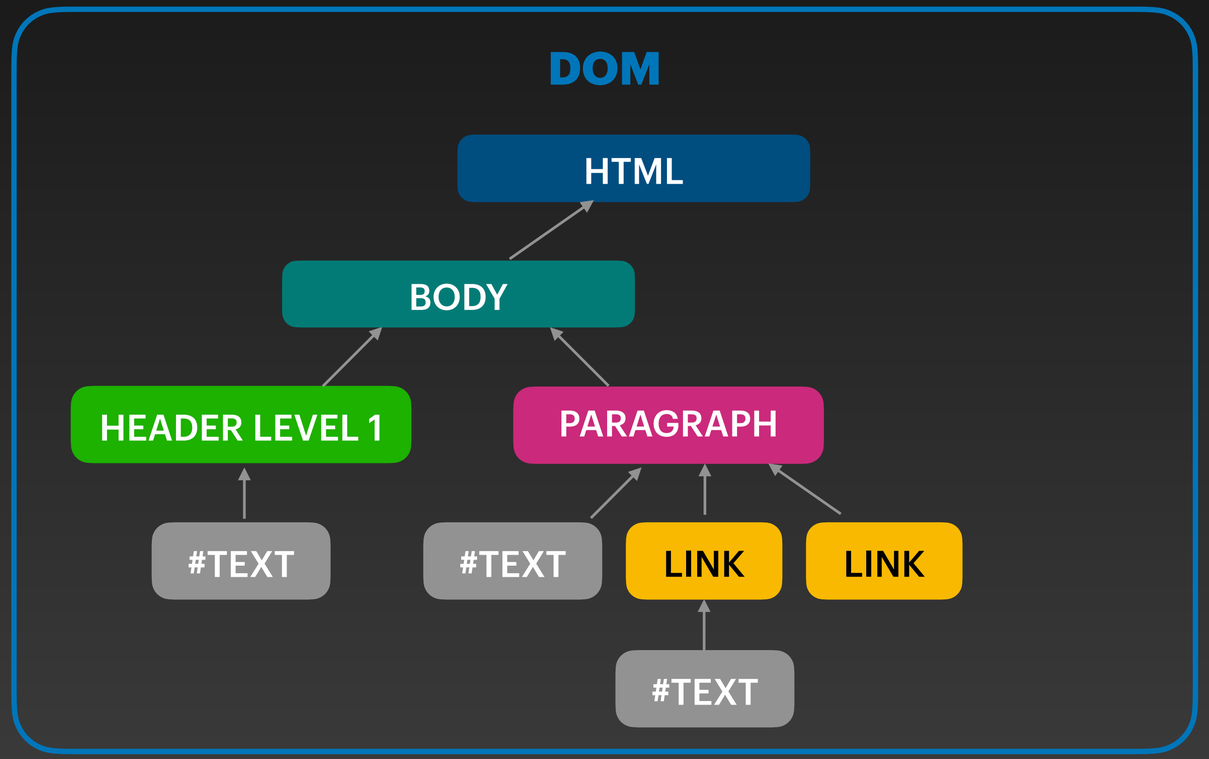
- Проверка на CSS селектор

```
elem.matches('p > a');
```

Поиск по DOM
closest

- Поиск по цепочке предков

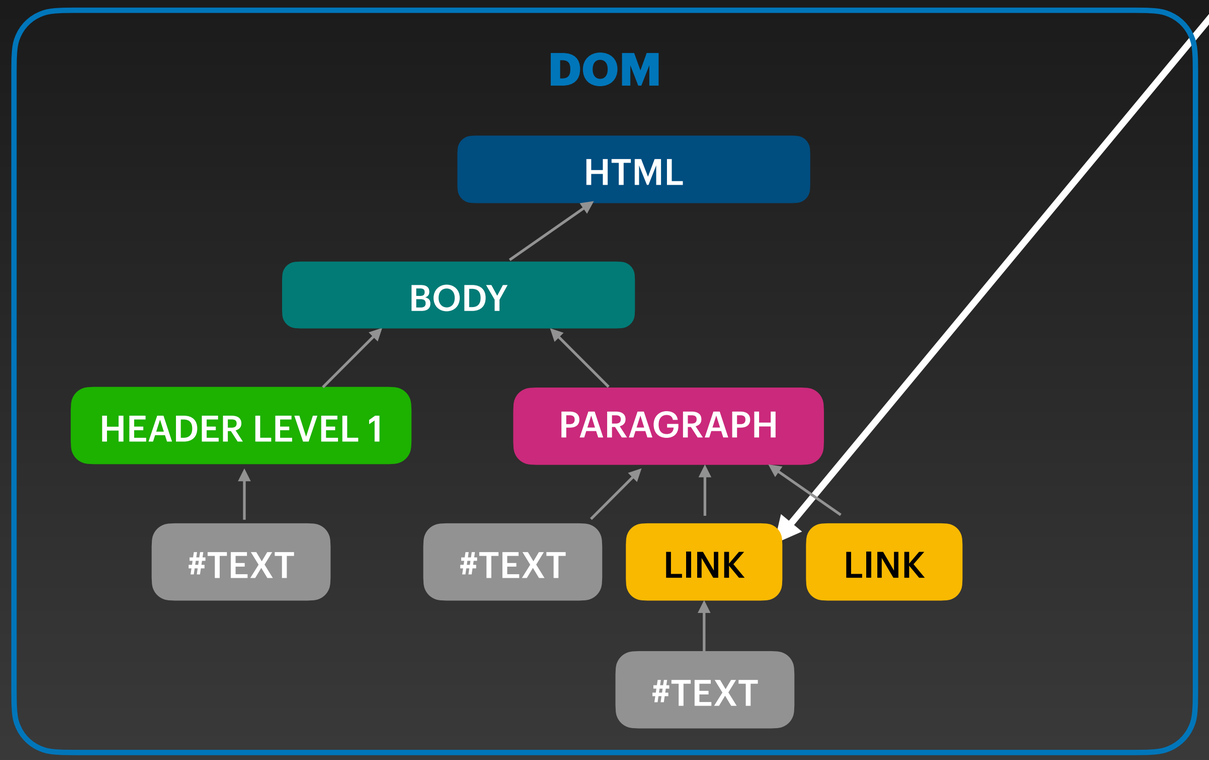
```
const p = document.querySelector('p > a');
```



Поиск по DOM
closest

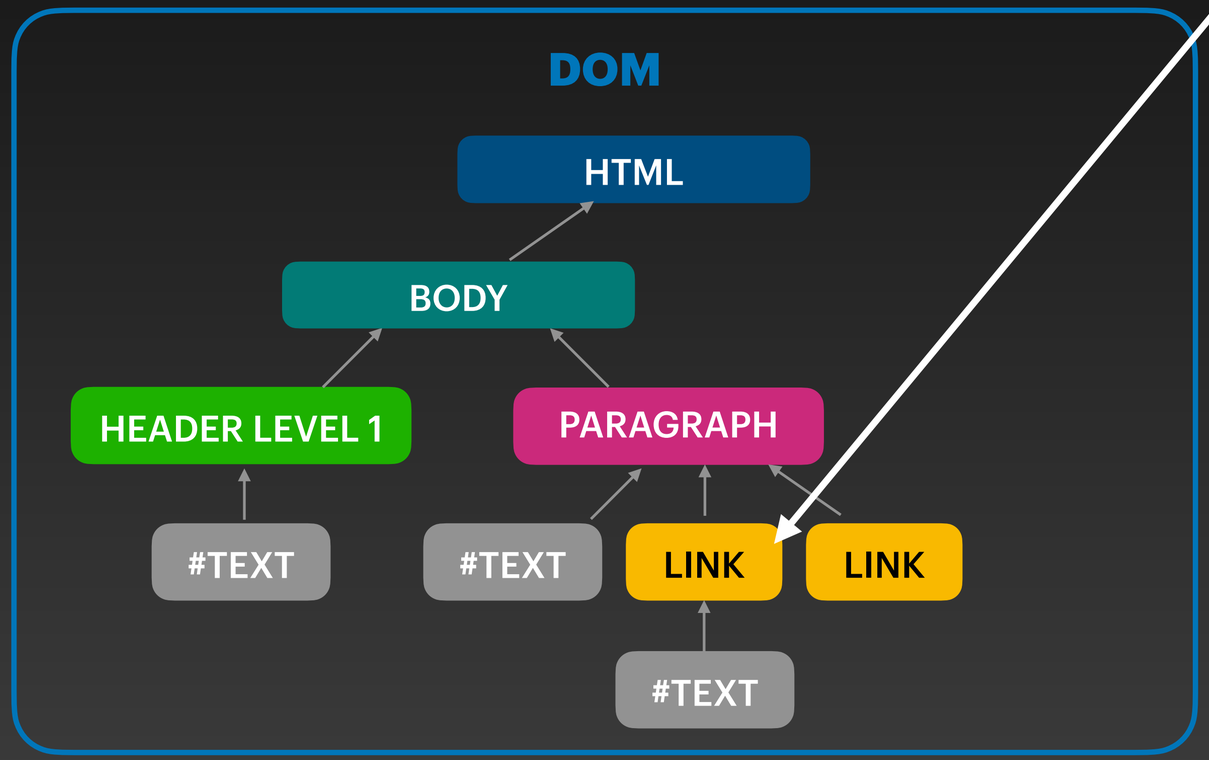
- Поиск по цепочке предков

```
const p = document.querySelector('p > a');
```



Поиск по DOM
closest

- Поиск по цепочке предков

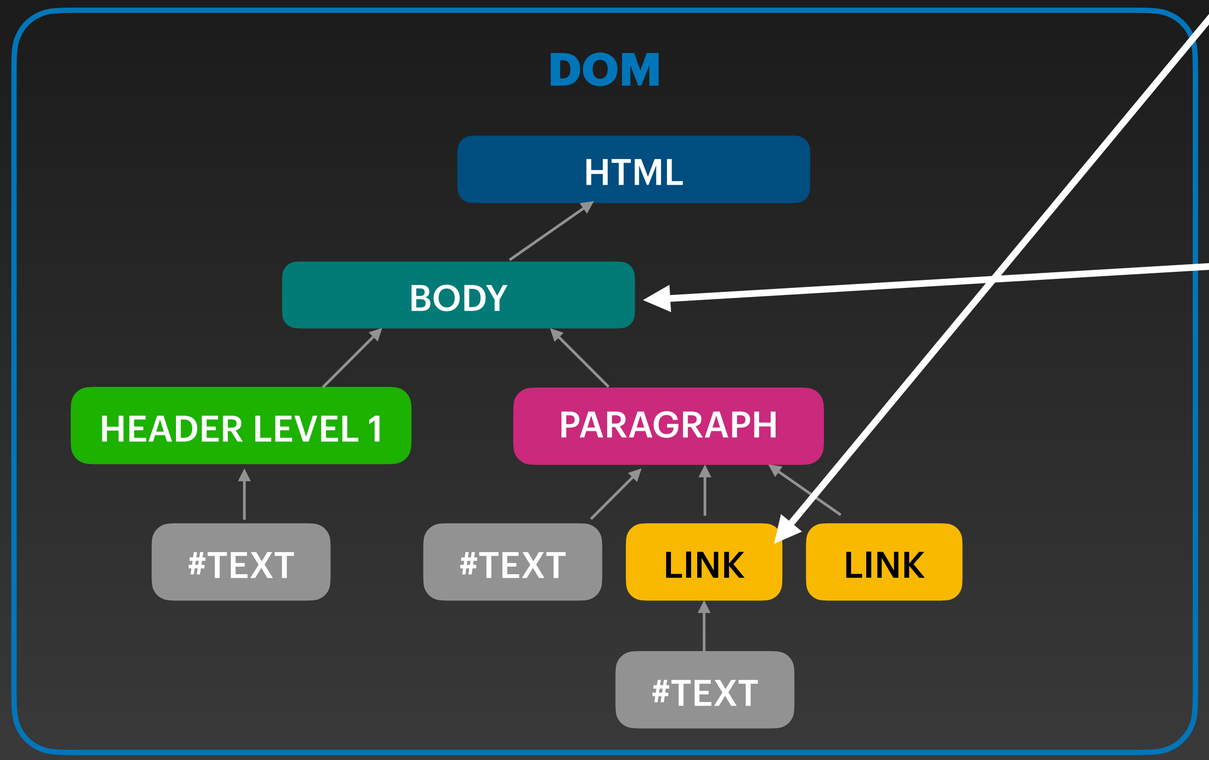



```
const p = document.querySelector('p > a');
```

```
p.closest('body');
```

Поиск по DOM
closest

- Поиск по цепочке предков

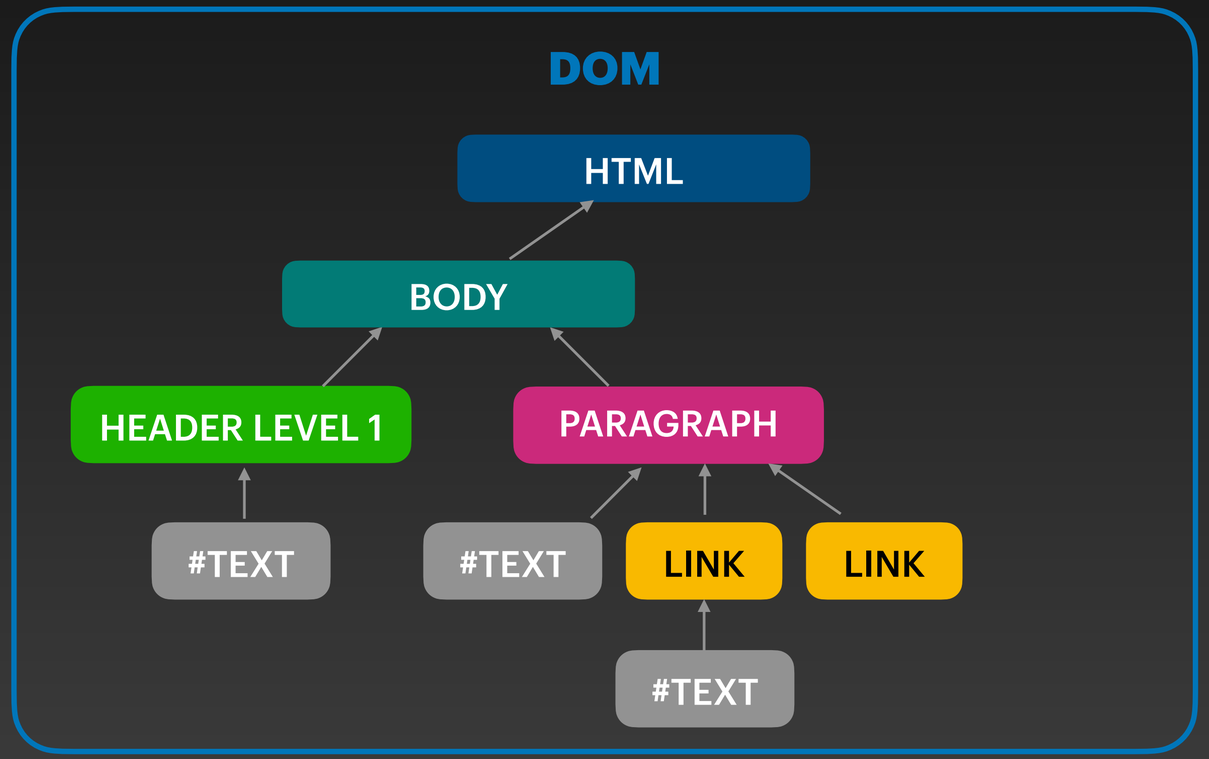



```
const p = document.querySelector('p > a');
```

```
p.closest('body');
```

Поиск по DOM
contains

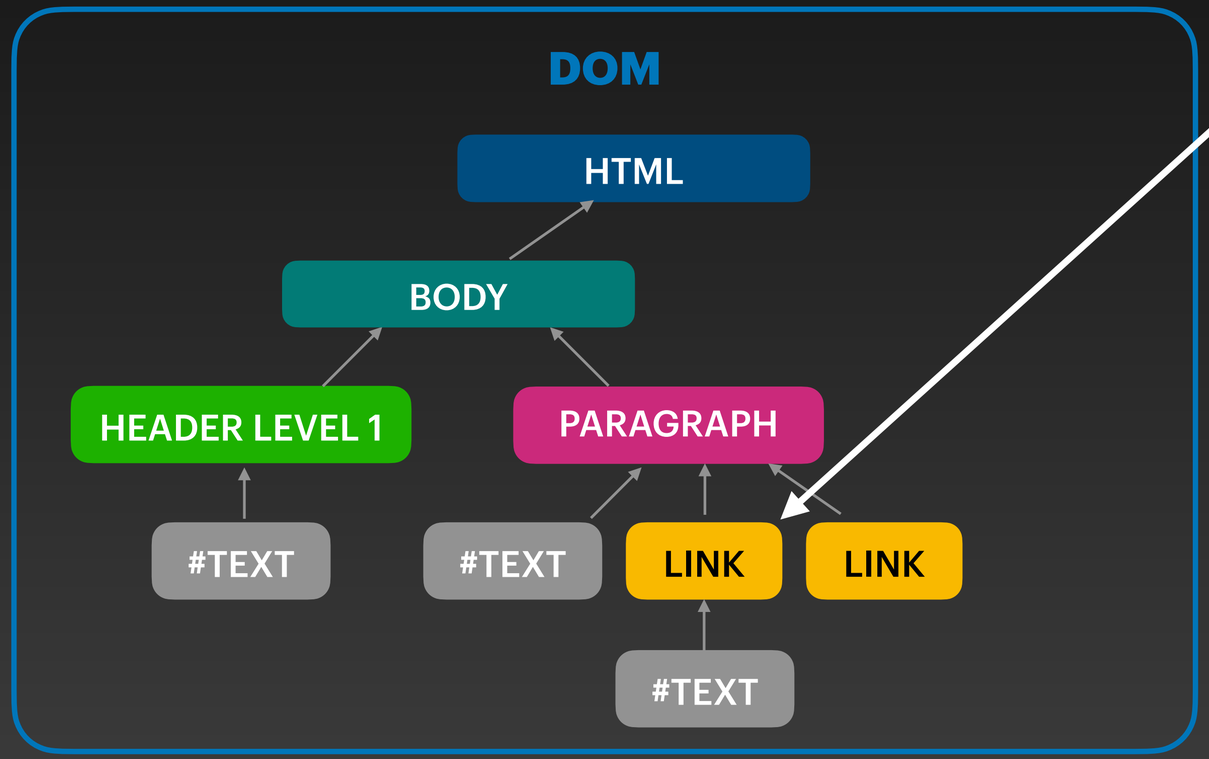
- Поиск по цепочке предков




```
const el1 = document.querySelector(‘body > a');
```

Поиск по DOM
contains

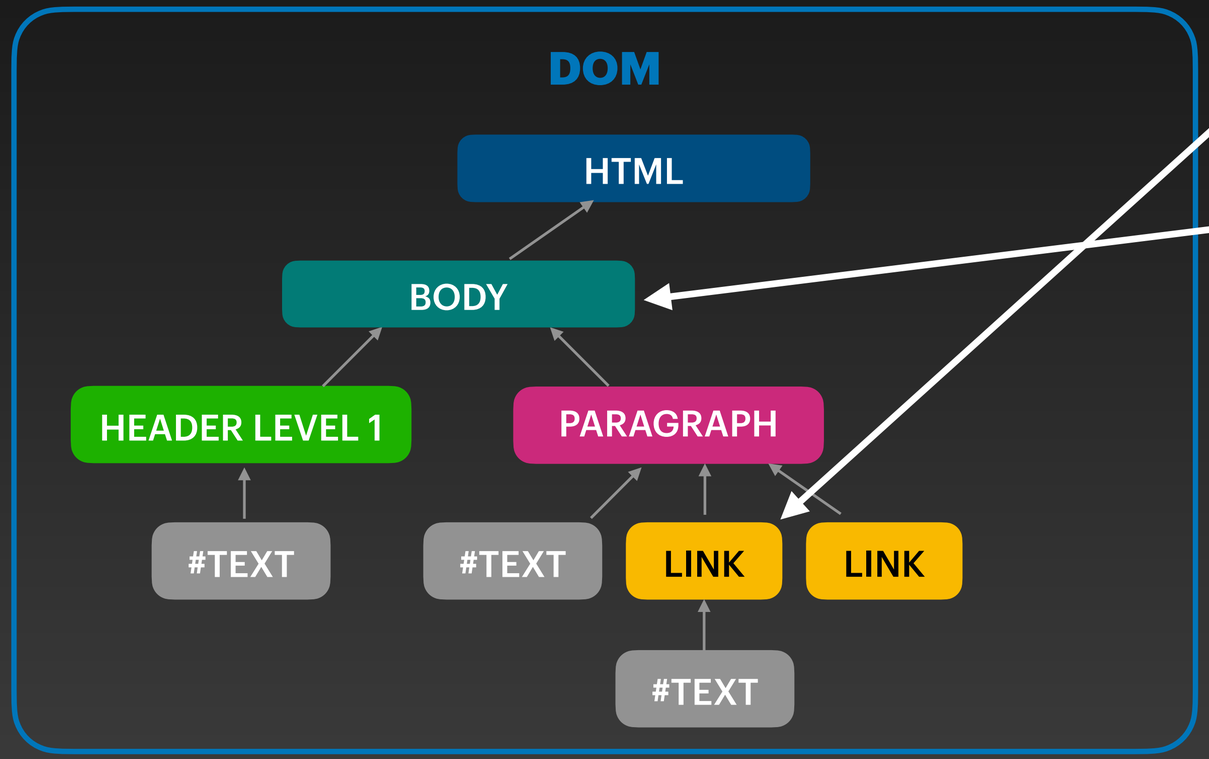
- Поиск по цепочке предков




```
const el1 = document.querySelector(‘body > a');
```

Поиск по DOM
contains

- Поиск по цепочке предков

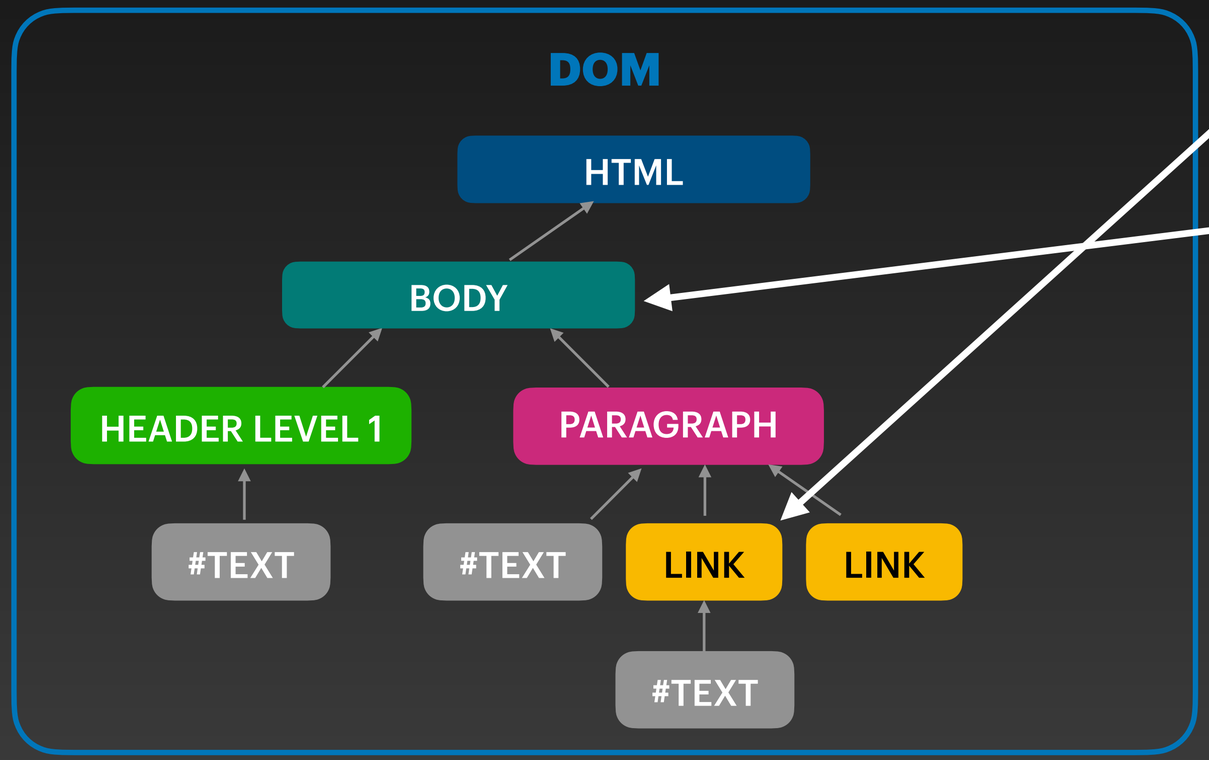



```
const el1 = document.querySelector(‘body > a');
```

```
const el2 = document.body;
```

Поиск по DOM
contains

- Поиск по цепочке предков



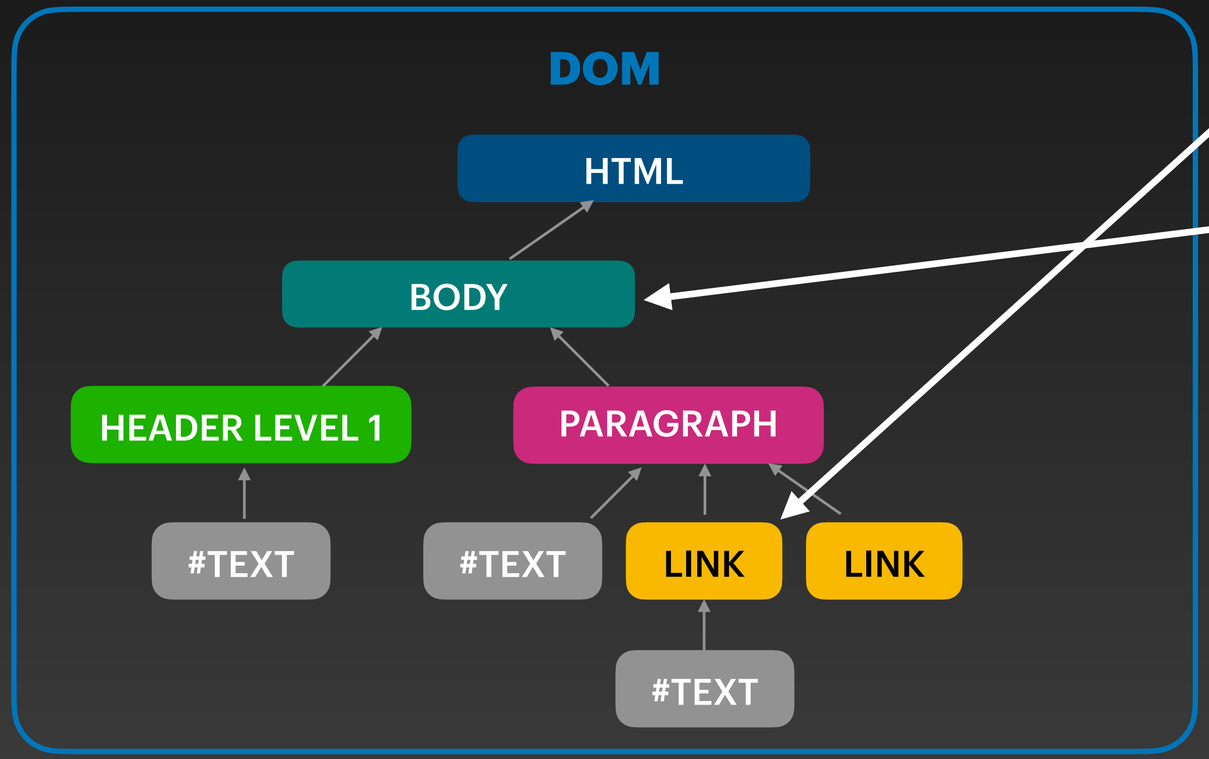

```
const el1 = document.querySelector(‘body > a');
```

```
const el2 = document.body;
```

```
el2.contains(el1)
```

Поиск по DOM
contains

- Поиск по цепочке предков




```
const el1 = document.querySelector(‘body > a');
```

```
const el2 = document.body;
```

```
el2.contains(el1)
```

true

Свойства узлов DOM дерева

Название тега

```
<body>
  <!-- комментарий -->

  <script>
    // для комментария
    console.log( document.body.firstChild.tagName ); // undefined (не элемент)
    console.log( document.body.firstChild.nodeName ); // #comment

    // для div'a
    console.log( document.body.lastChild.tagName ); // DIV
    console.log( document.body.lastChild.nodeName ); // DIV
  </script>

  <div>My Div</div>
</body>
```

Содержимое элемента
innerHTML

- Возвращает содержимое элемента в виде строки
- Позволяет менять содержимое элемента
  - Исправляет ошибки (например, незакрытые теги)

```
<body>
  <p>Параграф</p>
  <div>DIV</div>

  <script>
    console.log( document.body.innerHTML ); // читаем текущее содержимое
    document.body.innerHTML = 'Новый BODY!'; // заменяем содержимое
  </script>

</body>
```

Другие свойства

Свойство	Описание
outerHtml	Полный HTML узла-элемента. Позволяет менять содержимое элемента, но не сам элемент
nodeValue/data	Содержимое узла не являющегося элементом. Почти идентичны друг-другу, можно переписывать.
textContent	Текст внутри элемента, без учёта вложенных элементов (тегов). Запись в него помещает текст в элемент, при этом, все спец символы будут интерпретированы как текст. Можно использовать для безопасной записи в HTML.
hidden	Аналог `display: none` в CSS
Специальные свойства	Почти у каждого HTML атрибута есть соответствующее свойство в DOM. У <input> элементов будет value, у <a> — будет href, и так далее

Атрибуты

Атрибуты тегов

- Браузер парсит не только теги, но и атрибуты
- Например, если в документе есть тег с айди равным `my_element`, то у него будет доступно свойство с соответствующим значением

```
<body id="my_element">
  <script>
    console.log(document.body.id) // my_element
  </script>
</body>
```

Перезапись атрибутов

- Значение атрибутов можно менять

```
<body id="my_element">
  <script>
    document.body.id = ‘new-id'
  </script>
</body>
```

```
<body id=“new-id">
  <script>
    document.body.id = 'new-id'
  </script>
</body>
```

- Исключением является атрибут `value` у тега <input> — он доступен только для чтения

Работа с атрибутами

- elem.attributes – коллекция всех атрибутов
- elem.hasAttribute(name) – проверить на наличие
- elem.getAttribute(name) – получить значение
- elem.setAttribute(name, value) – установить значение
- elem.removeAttribute(name) – удалить атрибут

Data-атрибуты

- Все свойства, начинающиеся с префикса `data-` зарезервированы для использования программистами
- Они доступны в специальном свойстве element.dataset

- Преобразование имён
  - У атрибута — kebab-case
  - У свойства — camelCase

Изменение документа

Создание узлов

- document.createElement(tag) – создаёт элемент с заданным тегом,
- document.createTextNode(value) – создаёт текстовый узел
- elem.cloneNode(deep) – клонирует элемент, если deep==true, то со всеми дочерними элементами

Создание узлов

- node.append(...nodes or strings) – вставляет в node в конец
- node.prepend(...nodes or strings) – вставляет в node в начало
- node.before(...nodes or strings) – вставляет прямо перед node
- node.after(...nodes or strings) – вставляет сразу после node
- node.replaceWith(...nodes or strings) – заменяет node
- node.remove() – удаляет node

Изменение документа

- Эффективное изменение DOM — нетривиальная задача
- Обычно её решают библиотеки/фреймворки, такие как React, Angular, Vue, Svelte и т.д.
- Мы впоследствии будем использовать React, но знакомство с методами изменения документа важно для общего понимания работы браузера

Стили

Стили

- Стили можно задавать двумя способами
  - Изменяя классы в элементе
  - Напрямую меняя свойство style — используется только в крайних случаях, когда через классы сделать не получается
    - Пример: перемещение элемента по странице с помощью JS

Стили

- className – строковое значение, удобно для управления всем набором классов
- classList – объект с методами add/remove/toggle/contains, удобно для управления отдельными классами

Практика