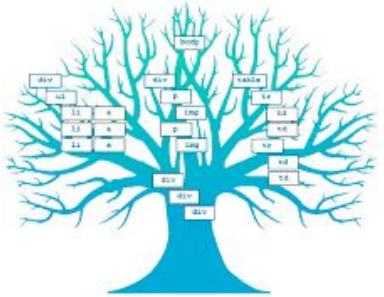
JS DOM, Events

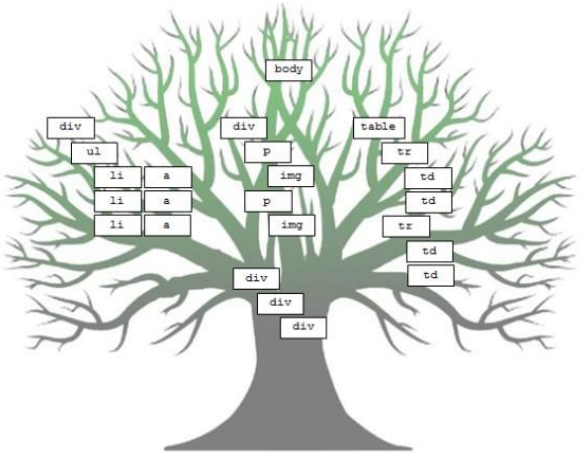
Подготовил:
Глеб Завертяев

DOM (Document Object Model)

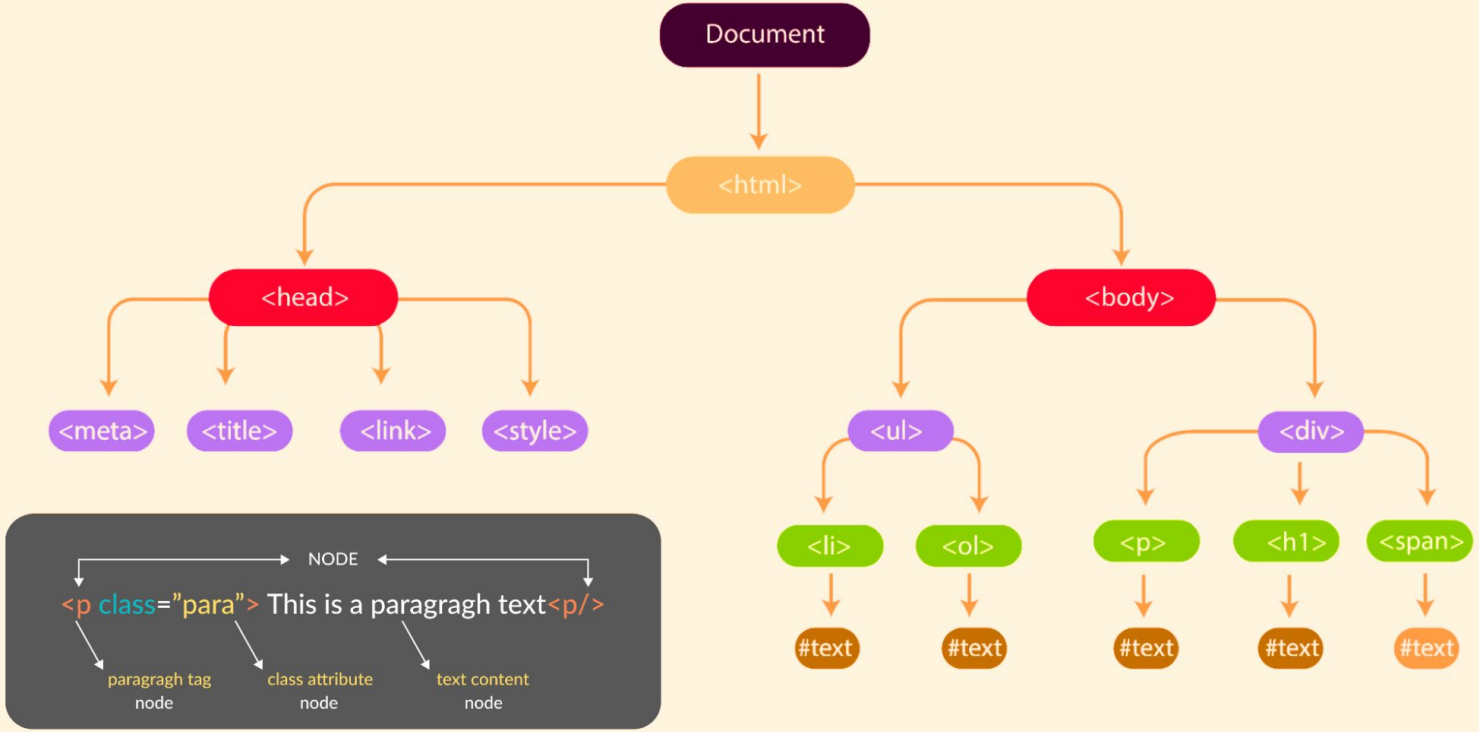


DOM (Document Object Model)– это объектная модель документа, которую браузер создает в памяти компьютера на основании HTML-кода.

Иными словами, это представление HTML-документа в виде дерева тегов. Такое дерево нужно для правильного отображения сайта и внесения изменений на страницах с помощью JavaScript



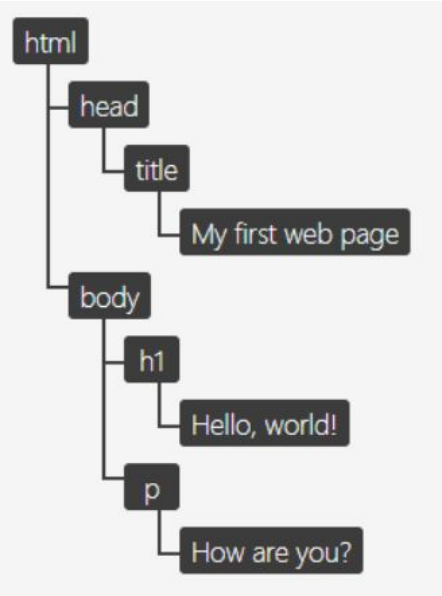
The DOM Structure/ DOM TREE



Возьмем простой документ:

```
<!doctype html>
<html lang="en">
  <head>
    <title>My first web page</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <p>How are you?</p>
  </body>
</html>
```

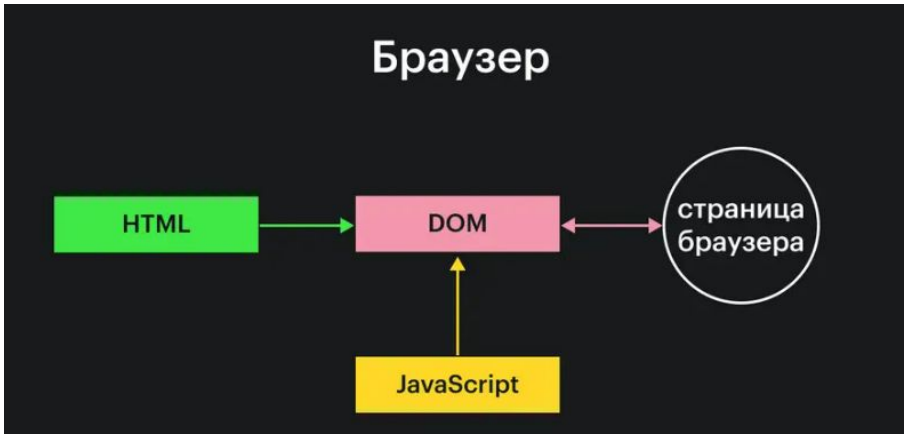
Он может быть представлен в виде дерева узлов:



DOM позволяет управлять HTML-разметкой из JavaScript-кода.

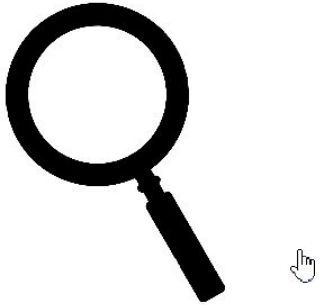
Управление обычно состоит из:

- добавления элементов
- удаления элементов
- изменения стилей и содержимого элементов



Прежде чем управлять элементом его нужно выбрать!

Методы поиска элементов



Метод 1

Поиск элемента по ID

```
document.getElementById(id)
```

Примечание: возвращает элемент с заданным id. элемент должен иметь атрибут **id**

Метод 1

Поиск элемента по ID

Задача: получить элемент с id=”elem”

```
let elem = document.getElementById('elem');
```

Метод 2

Поиск элемента по тегу

```
document.getElementsByTagName(tag)
```

Примечание: метод ищет элементы с данным тегом и возвращает их коллекцию. Передав "*" вместо тега, можно получить всех потомков

Метод 2

Поиск элемента по тегу

Задача: получить все элементы div в документе

```
let divs = document.getElementsByTagName('div');
```

Метод 3

Поиск элементов по названию класса

```
document.getElementsByClassName(className)
```

Примечание: метод возвращает элементы, которые имеют данный класс

Метод 3

Поиск элементов по названию класса

Задача: получить все элементы с классом article

```
let articles = document.getElementsByClassName('article');
```

Метод 4

Поиск элементов по значению атрибута name

```
document.getElementsByName(name)
```

Примечание: возвращает элементы с заданным атрибутом name

Метод 4

Поиск элементов по значению атрибута name

Задача: получить все элементы со значением атрибута name=”up”

```
let articles = document.getElementsByName(“up”);
```

Метод 5

Поиск по CSS-селектору

(самый универсальный метод поиска)

```
elem.querySelectorAll(css)
```

Примечание: возвращает все элементы внутри elem, удовлетворяющие данному CSS-селектору.

Метод 5

Поиск по CSS-селектору

Задача: получает все элементы <li>, которые являются последними потомками в

```
let elements = document.querySelectorAll('ul > li:last-child');
```

Есть 6 основных методов поиска элементов в DOM:

Метод	Ищет по...	Ищет внутри элемента?	Возвращает живую коллекцию?
`querySelector`	CSS-selector	✓	-
`querySelectorAll`	CSS-selector	✓	-
`getElementById`	`id`	-	-
`getElementsByName`	`name`	-	✓
`getElementsByTagName`	tag or `'*'`	✓	✓
`getElementsByClassName`	class	✓	✓

Добавление элементов

1 шаг - Создать элемент

Метод: `document.createElement("tag")`

Создание: `let el = document.createElement("div")`

2 шаг - Заполнить элемент

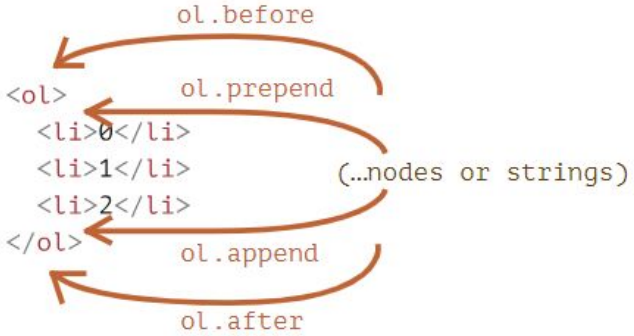
Метод: `node.textContent = “text”`

Заполнение: el.textContent = “Text”

3 шаг - Добавляем элемент в document

Методы:

- node.**append**(el) – добавляет узлы или строки в конец node,
- node.**prepend**(el) – вставляет узлы или строки в начало node,
- node.**before**(el) – вставляет узлы или строки до node,
- node.**after**(el) – вставляет узлы или строки после node,



Изменение элементов

- **textContent**

Позволяет задавать или получать текстовое содержимое элемента и его потомков.

```
var text = element.textContent;
element.textContent = "Это просто текст";
```

• innerHTML

Свойство innerHTML позволяет считать содержимое элемента в виде HTML-строки или установить новый HTML.

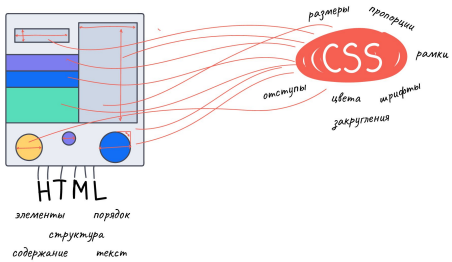
```
<form>
  <label>Логин</label>
  <input type="text" id="login" />
  <div class="error">Введите логин</div>
</form>
```

```
1 const form = document.querySelector('form')
2
3 console.log(form.innerHTML)
4 // '<label>Логин</label><input type="text" id="login" /><div class="error">Вве
5
6 // Меняем содержимое новым html
7 form.innerHTML = '<div class="success">Вход выполнен</div>'
```

• методы изменения атрибутов

В JavaScript есть четыре метода для изменения атрибутов элементов:

Метод	Описание	Пример
`hasAttribute()`	Возвращает `true` или `false`	`element.hasAttribute('href');`
`getAttribute()`	Возвращает значение определенного атрибута или `null`	`element.getAttribute('href');`
`setAttribute()`	Добавляет или обновляет заданный атрибут	`element.setAttribute('href', 'index.html');`
`removeAttribute()`	Удаляет атрибут элемента	`element.removeAttribute('href');`



изменение стилей

HTML DOM позволяет JavaScript изменять стиль HTML элементов.

Синтаксис:

```
document.getElementById(id).style.свойство = новый стиль
```

Применение:

```
document.getElementById("p2").style.color = "blue";
```

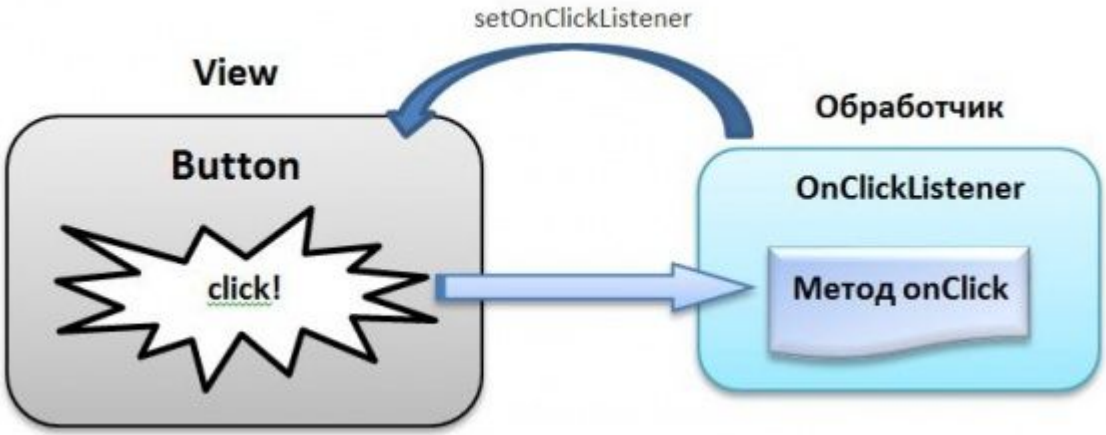
• удаление элементов

Для удаления узла есть метод **node.remove()**

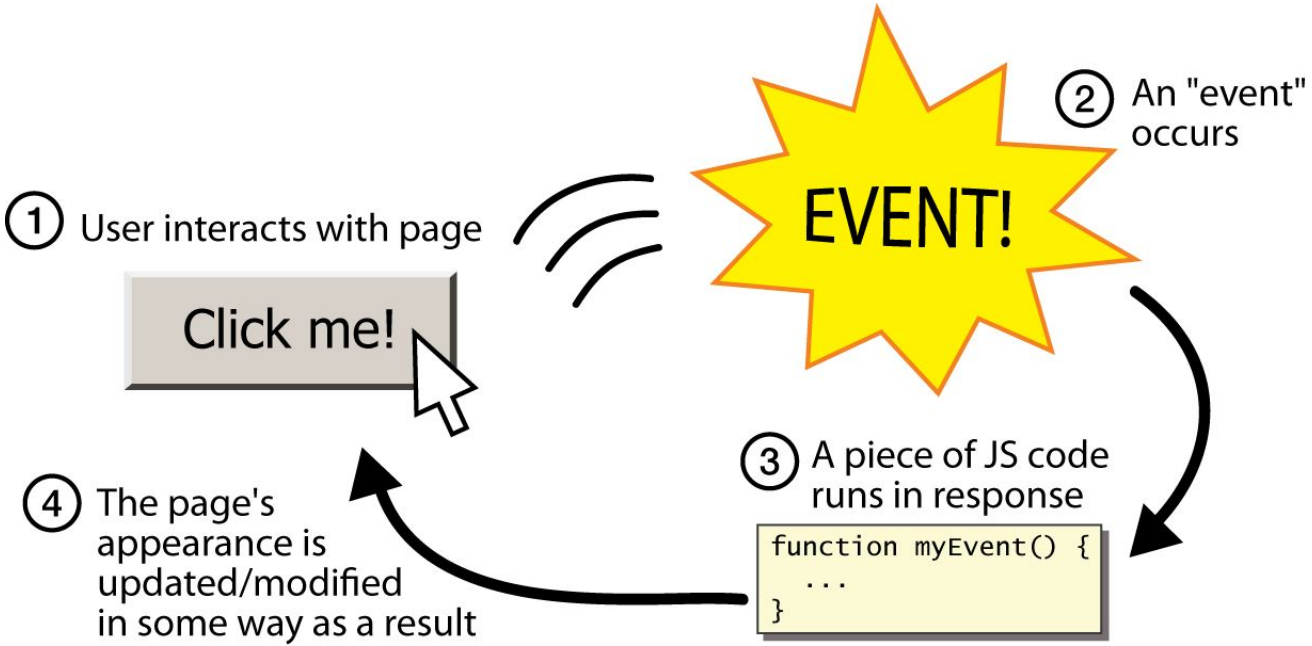
Пример использования: удаление элемента с id=”register”

```
document.getElementById("register").remove();
```

Events, event listeners

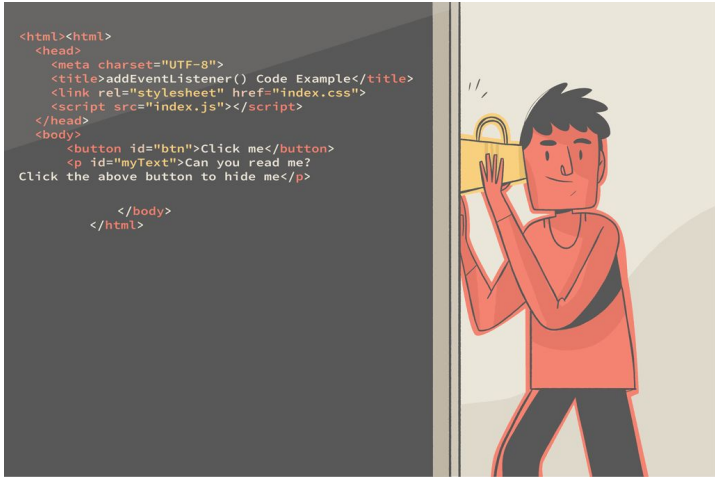


Любой DOM элемент запускает событие, когда мы с ним как-то взаимодействуем (кликаем, наводим мышь и др.). Обработчики событий в JS используются для того, чтобы реагировать на эти события.



Чтобы "повесить" обработчик событий на элемент, нужно использовать специальный метод - **addEventListener**. Этот метод принимает 2 аргумента:

1. **Тип события** (например "click").
2. Так называемую **колбэк** (callback) функцию, которая запускается после срабатывания нужного события.




```
element.addEventListener('click', handleClickFunction)
```

Пример

Найдём первую кнопку на странице и будем выводить сообщение в консоль, когда произошёл клик по этой кнопке.

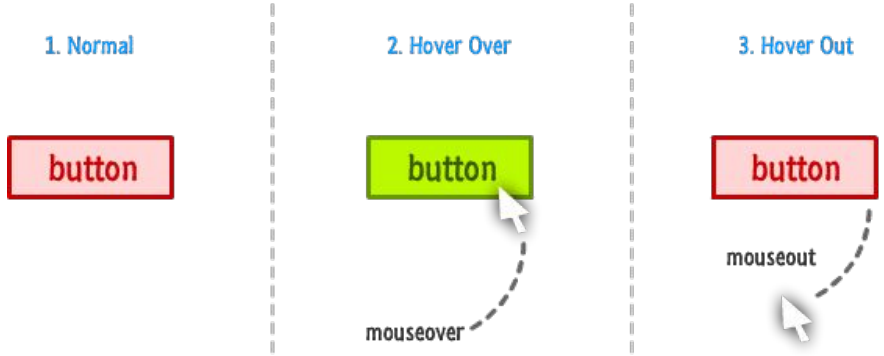
```
1 const element = document.querySelector('button')
2
3 element.addEventListener('click', function (event) {
4   console.log('Произошло событие', event.type)
5 })
```

Типы событий

eventType (первый аргумент addEventListener) - строка, содержащая название события.

Наиболее популярные события:

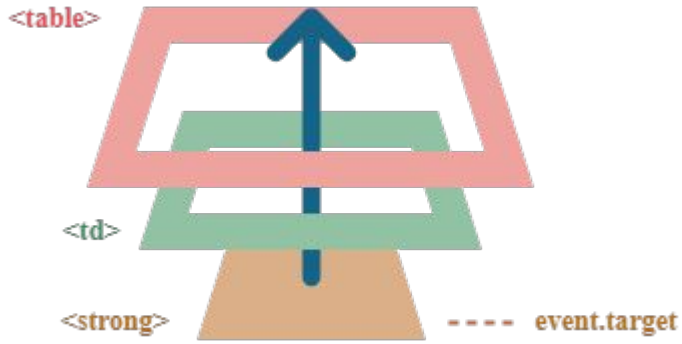
- 'click'
- 'change'
- 'submit'
- 'keydown'
- 'keyup'
- 'mousemove'
- 'mouseenter'
- 'mouseleave'



Объект Event

Когда происходит событие, браузер создаёт объект события - event, записывает в него детали и передаёт его в качестве аргумента функции-обработчику (второму аргументу addEventListener).

Объект Event - основные свойства



- **defaultPrevented** — отменено ли поведение события по умолчанию.
- **target** — ссылка на объект, которым было инициировано событие. Например, если событие произошло на поле ввода, мы получим ссылку на этот DOM элемент.
- **type** — тип события.

Объект Event - метод preventDefault

`event.preventDefault()` — предотвращает дефолтное поведение события.

Например, при нажатии на ссылку, отменить переход по адресу ссылки