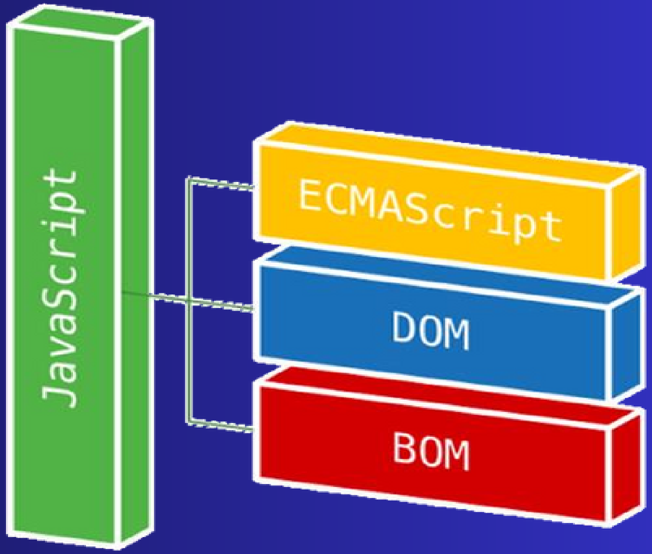
Java Script

Lecture 8

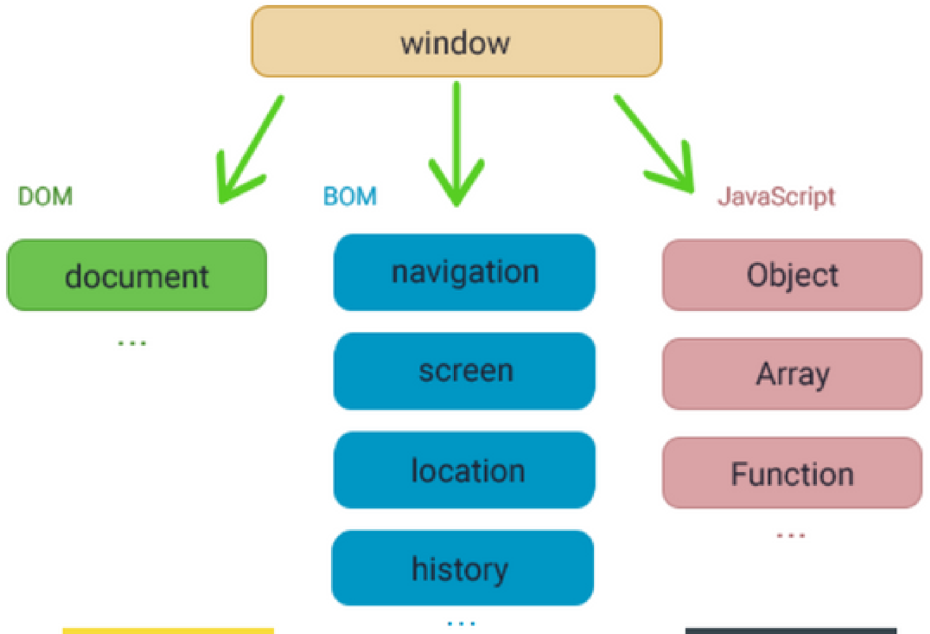
Table of Contents

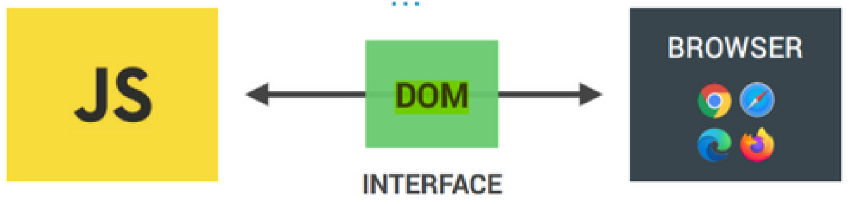


What is BOM in JavaScript ?

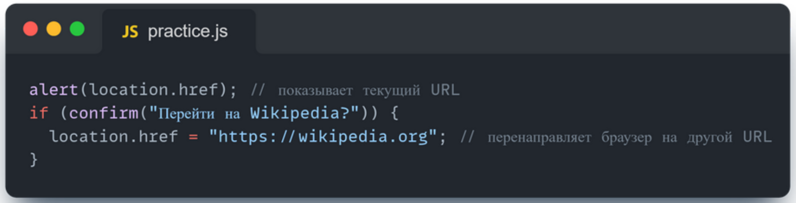
BOM - Browser Object Model

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.





The Browser Object Model (BOM) is the additional objects provided by the browser (environment) to work with everything other than the document.



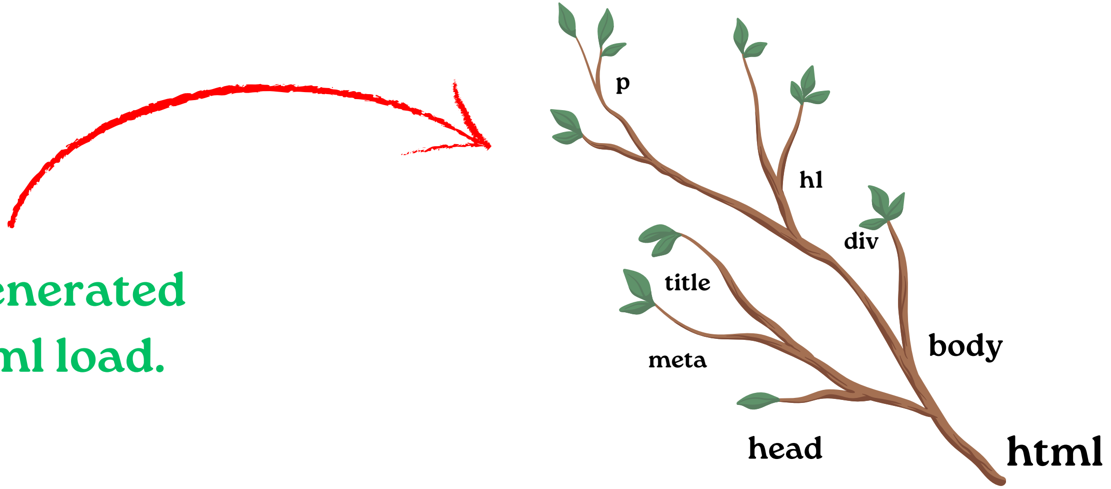
```
alert(location.href); // показывает текущий URL
if (confirm("Перейти на Wikipedia?")) {
  location.href = "https://wikipedia.org"; // перенаправляет браузер на другой URL
}
```

What is DOM in JavaScript ?

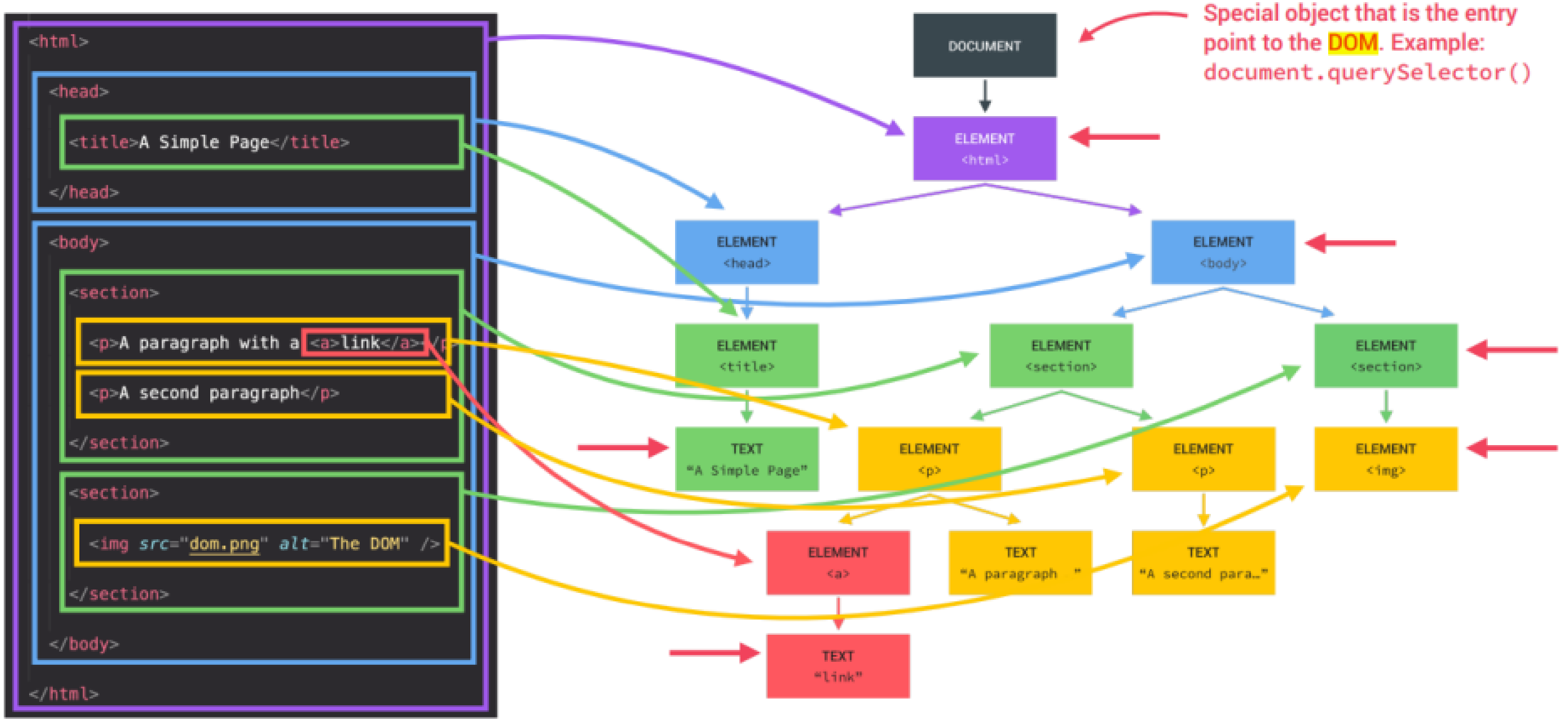
Document Object Model-(DOM)

DOM - "Document Object Model". It's a standardized way of representing the structure of a document (such as HTML) as a tree-like structure, where each node represents a part of the document, such as an element, attribute, text content, etc. DOM allows programs and scripts to access, manipulate, and update the content, structure, and styles of a web page. JavaScript is commonly used to interact with the DOM in web development, for tasks such as dynamically changing the content of a page or handling user events.

Tree structure, generated by browser on html load.



THE DOM TREE STRUCTURE

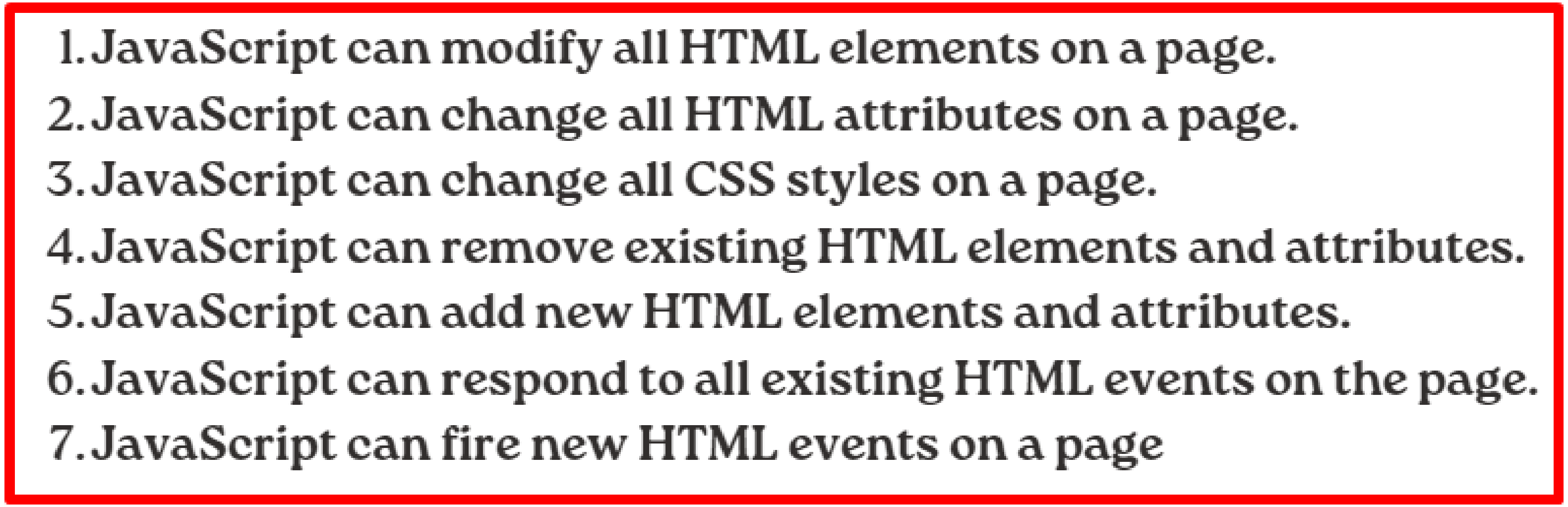


```
<html>
  <head>
    <title>A Simple Page</title>
  </head>
  <body>
    <section>
      <p>A paragraph with a <a>link</a></p>
      <p>A second paragraph</p>
    </section>
    <section>
      <img src="dom.png" alt="The DOM" />
    </section>
  </body>
</html>
```

Special object that is the entry point to the DOM. Example: `document.querySelector()`

Dom Features

According to the Document Object Model (DOM for short), every HTML tag is an object. Subtags are "children" of the parent element. The text that is inside the tag is also an object.All these objects are available with JavaScript, we can use them to modify the page.

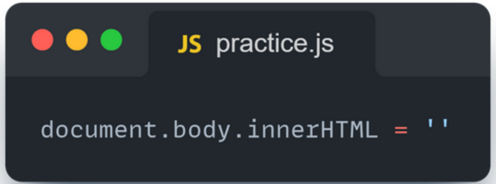


1. JavaScript can modify all HTML elements on a page.
2. JavaScript can change all HTML attributes on a page.
3. JavaScript can change all CSS styles on a page.
4. JavaScript can remove existing HTML elements and attributes.
5. JavaScript can add new HTML elements and attributes.
6. JavaScript can respond to all existing HTML events on the page.
7. JavaScript can fire new HTML events on a page

Definition and Usage. The querySelector() method returns the first child element that matches a specified CSS selector(s) of an element, querySelectorAll() method can be used to access all elements which match with a specified CSS selector.

HTML DOM Element innerHTML,style object

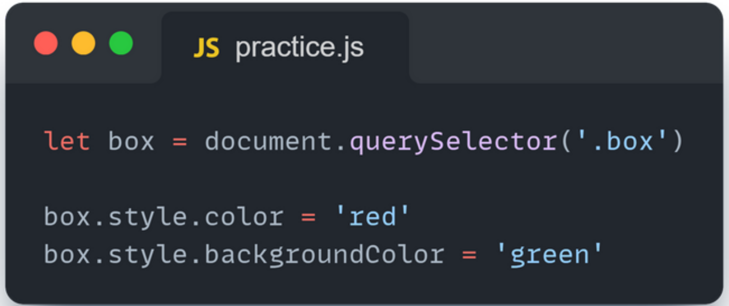
innerHTML - this property completely provides an easy way replace the elementary element. For example, all requirements the element's body can be removed:



practice.js

```
document.body.innerHTML = ''
```

The Style object represents an individual style statement.



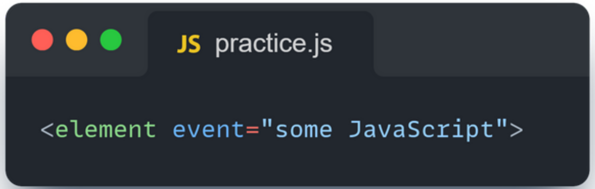
practice.js

```
let box = document.querySelector('.box')

box.style.color = 'red'
box.style.backgroundColor = 'green'
```

Dom Events

HTML events are "things" that happen to HTML elements.
When JavaScript is used in HTML pages, JavaScript can "react" on these events.

An HTML event can be something the browser does, or something a user does.



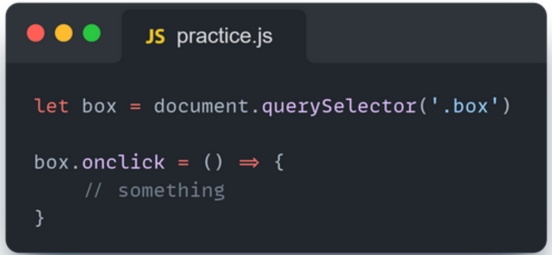
JS practice.js

```
<element event="some JavaScript">
```

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

JavaScript lets you execute code when events are detected.



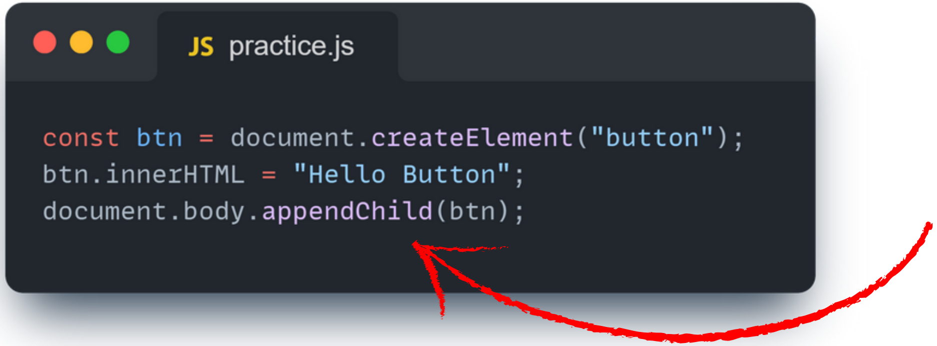
JS practice.js

```
let box = document.querySelector('.box')

box.onclick = () => {
    // something
}
```

onclick -The user clicks an HTML element

Dom Create element

The JavaScript document.createElement() method allows you to create and return a new element (an empty Element node) with the specified tag name.
1) createElement(elementName): Creates an html element whose tag is passed as a parameter. Returns the created elemen


practice.js

```
const btn = document.createElement("button");
btn.innerHTML = "Hello Button";
document.body.appendChild(btn);
```

The appendChild() method appends a node (element) as the last
child of an element.
appendChild() adds a node to the end of the list of children of the
specified parent node. If the given child element is a reference to
an existing node in the document, then the appendChild()
function moves it from its current position to the new position

Dom props: classList

Element.classList is a read-only property that contains the current DOMTokenList collection of all the element's class attributes.
Using classList provides a more convenient way than accessing an element's class list as a space-separated string via element.className.

ClassList is a getter. The object it returns has several methods:

- add(String [,String])

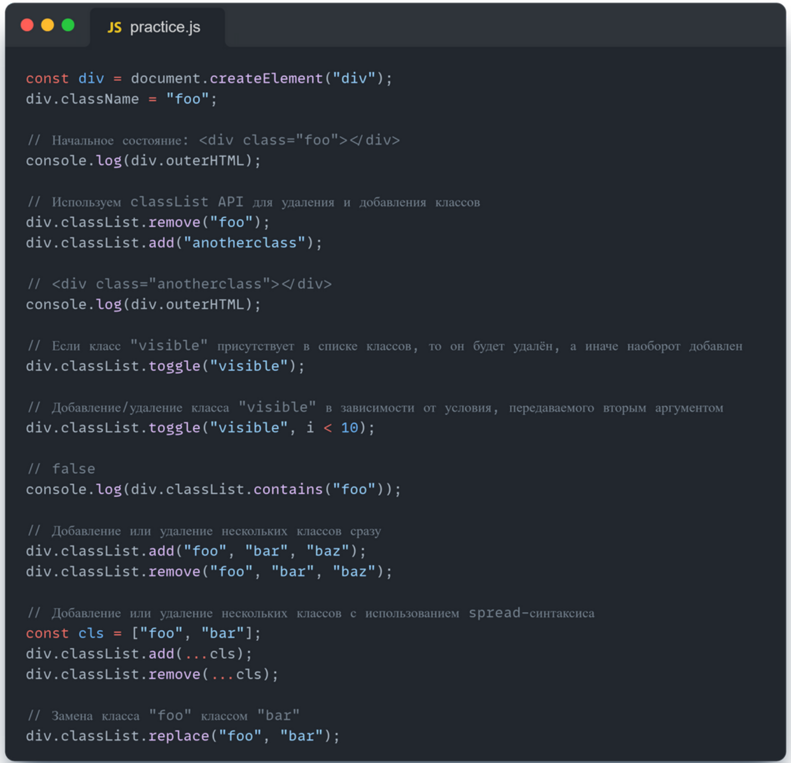
Adds the specified classes to the element

- remove(String [,String])

Removes the specified classes from the element

- toggle(String[, Boolean])

If the element has no class, it adds it, otherwise it removes it. When false is passed as the second parameter, it removes the specified class, and if true, it adds it.
If the second parameter is undefined or a variable with typeof == 'undefined', the behavior is the same as passing only the first parameter when calling

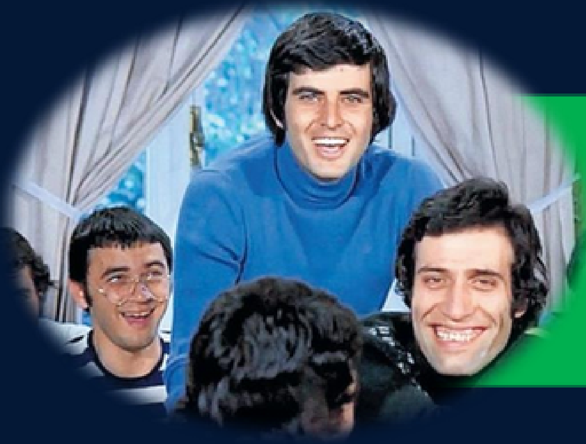


```js
const div = document.createElement("div");
div.className = "foo";

// Начальное состояние: <div class="foo"></div>
console.log(div.outerHTML);

// Используем classList API для удаления и добавления классов
div.classList.remove("foo");
div.classList.add("anotherclass");

// <div class="anotherclass"></div>
console.log(div.outerHTML);

// Если класс "visible" присутствует в списке классов, то он будет удалён, а иначе наоборот добавлен
div.classList.toggle("visible");

// Добавление/удаление класса "visible" в зависимости от условия, передаваемого вторым аргументом
div.classList.toggle("visible", i < 10);

// false
console.log(div.classList.contains("foo"));

// Добавление или удаление нескольких классов сразу
div.classList.add("foo", "bar", "baz");
div.classList.remove("foo", "bar", "baz");

// Добавление или удаление нескольких классов с использованием spread-синтаксиса
const cls = ["foo", "bar"];
div.classList.add(...cls);
div.classList.remove(...cls);

// Замена класса "foo" классом "bar"
div.classList.replace("foo", "bar");
```

SOFT CLUB

Thanks!

Be happy and Smile



The END

Lecture 8