

ПМЗ Разработка модулей ПО.

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

Тема 1. JS: DOM API.

Лекция 1. JavaScript в браузере.

Цели занятия:

- Познакомиться с ролью JavaScript в браузере.
- Рассмотреть глобальный объект `window` и основные ВОМ-объекты (`navigator`, `location`, `history`).
- Изучить базовые методы взаимодействия с пользователем (`alert`, `confirm`, `setTimeout`).
- Понять разницу между DOM-элементами и текстовыми узлами (`children` и `childNodes`).

Учебные вопросы:

- 1. Роль JavaScript в браузере.**
- 2. Первый скрипт на странице.**
- 3. Глобальный объект window.**

1. Роль JavaScript в браузере.

Что такое JavaScript в браузере?

JavaScript — это язык программирования, встроенный в браузеры, который делает веб-страницы динамичными и интерактивными.

Если HTML отвечает за структуру, а CSS — за оформление, то JS добавляет логику и поведение.

Что умеет JS в браузере?

Работа с содержимым страницы (DOM):

- изменять текст, HTML-элементы, атрибуты;
- добавлять и удалять элементы;
- управлять стилями.

Реакция на действия пользователя:

- обработка кликов, движения мыши, нажатий клавиш;
- работа с формами (валидация, отправка данных);
- перехват событий (например, прокрутка страницы).



Взаимодействие с сервером:

- отправка и получение данных без перезагрузки страницы (fetch, AJAX);
- обновление содержимого «на лету».



Работа с браузером и окружением:

- хранение данных (localStorage, cookies);
- управление историей переходов и адресной строкой (History API, Location API);
- работа с таймерами (setTimeout, setInterval).

Чего JS в браузере не умеет:

- ✗ Не имеет прямого доступа к файловой системе (кроме ограниченного API, например FileReader).
- ✗ Не может напрямую управлять железом компьютера (камера/микрофон — только через специальные API с разрешения пользователя).
- ✗ Не выполняет низкоуровневые операции (работа с памятью, сетью вне браузера).
- ✗ Работает в «песочнице» (ограничённой среде браузера) — это защита безопасности.

Взаимодействие JS, HTML и CSS.

- HTML создаёт «каркас» страницы.
- CSS описывает, как страница выглядит.
- JS управляет ими: может изменить HTML (DOM-дерево) и применить другие CSS-стили.

 Принцип: HTML + CSS = статическая страница, HTML + CSS + JS = динамическое приложение.

Пример «живой» страницы с JS.

2. Первый скрипт на странице.

Что такое JavaScript в браузере?

JavaScript — это язык программирования, встроенный в браузеры, который делает веб-страницы динамичными и интерактивными.

Если HTML отвечает за структуру, а CSS — за оформление, то JS добавляет логику и поведение.

Способы подключения.

1. Встроенный скрипт (inline).

Код пишется прямо внутри HTML, в теге `<script>`:

```
<head>
    <title>Пример</title>
    <script>
        alert("Привет, мир!");
    </script>
</head>
```

👉 Такой код выполнится сразу в месте, где браузер дошёл до тега `<script>`.

Если скрипт стоит в `<head>`, он может выполниться до того, как элементы `<body>` созданы.

2. Внешний файл (стандартный способ)

Код хранится отдельно:

```
<script src="script.js"></script>
```

- ✓ Удобно поддерживать и масштабировать.
- ✓ Один файл можно использовать на разных страницах.
- ✓ Браузер кэширует файл → быстрее работает сайт.
- 👉 Это основной способ в реальных проектах.

Где подключать скрипт?

- ◆ В `<head>` без атрибутов

```
<head>
|   <script src="script.js"></script>
</head>
```

- Скрипт загружается и выполняется до рендеринга страницы.
- Может заблокировать загрузку HTML → задержка отображения.
- Практически не используется.

◆ В <body> перед </body>

```
<body>
  <h1>Контент</h1>
  <script src="script.js"></script>
</body>
```

- Скрипт выполняется после того, как весь HTML загружен.
- Не блокирует отрисовку страницы.
- ✓ Один из самых надёжных способов.

◆ В <head> с атрибутами defer и async

Defer

```
<head>
|  <script src="script.js" defer></script>
</head>
```

- Скрипт загружается параллельно с HTML.
- Выполняется после полной загрузки HTML, в порядке подключения.
- ✓ Стандарт для основного кода сайта.

async

```
<head>
|  <script src="script.js" async></script>
</head>
```

- Скрипт загружается параллельно.
- Выполняется сразу после загрузки, порядок не гарантирован.
- ✓ Используется для независимых скриптов (аналитика, реклама).



Главное, что нужно понимать:

- JS не работает сам по себе — его нужно подключить.
- Есть встроенный и внешний способы, но внешний файл + defer — стандарт.
- async и defer помогают оптимизировать загрузку страницы.

3. Глобальный объект `window`.

Среда выполнения JavaScript.

JavaScript сам по себе — это язык, в нём нет встроенного доступа к файлам, сети или экрану.

Все возможности зависят от того, где выполняется код:

- В браузере — есть доступ к окну вкладки, документу, истории переходов, сетевым запросам.
- В Node.js — есть доступ к файловой системе, процессам, модулям, но нет DOM.

 То есть JavaScript — «движок» (например, V8), а функции для работы с браузером предоставляет среда выполнения.

Глобальный объект window.

В браузере таким объектом является window.

Это глобальный объект вкладки.

Он хранит всё окружение:

- DOM (страница, элементы, события).
- BOM (объекты браузера).
- Глобальные переменные и функции. *

Пример:

```
alert("Привет"); // на самом деле это window.alert("Привет")
console.log(window.document.title); // доступ к DOM
```

```
function sum(x,y){
    return x + y;
}

var y = 20;
console.log(window.y); // 20
console.log(window.sum(2, 2)); // 4
```

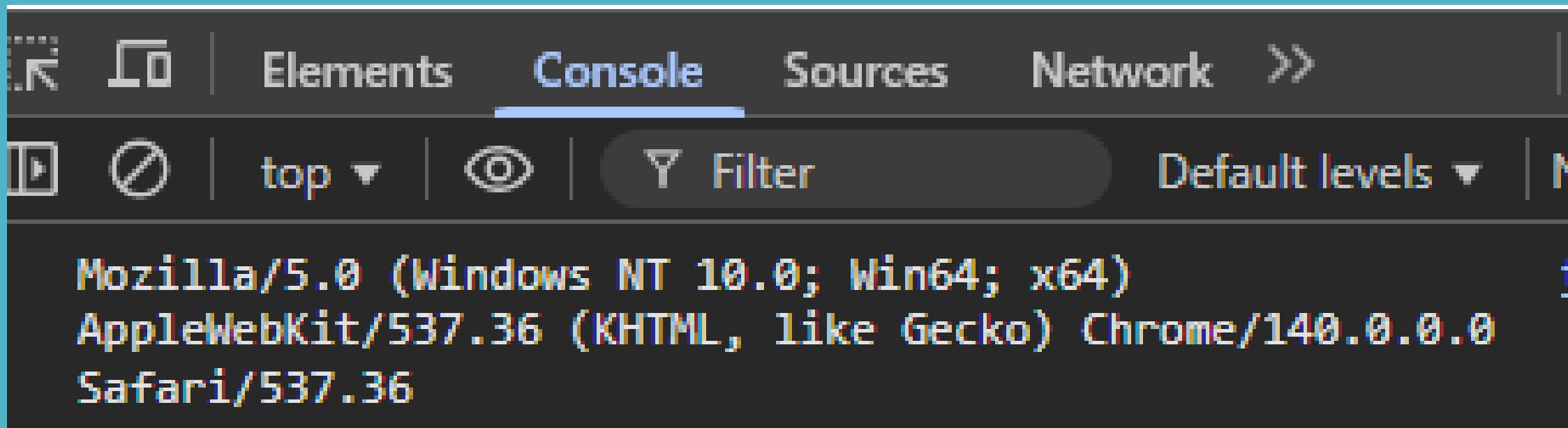
ВОМ (Browser Object Model).

ВОМ — это «надстройка» над JS, которая описывает объекты браузера.

- ◆ navigator — информация о браузере и системе
 - navigator.userAgent — строка с данными о браузере.
 - navigator.language — язык интерфейса.
 - navigator.onLine — онлайн/оффлайн статус.

Пример:

```
console.log(navigator.userAgent);
```



◆ **location** — текущий URL и переходы

- `location.href` — полный адрес страницы.
- `location.reload()` — перезагрузка.
- `location.assign("https://example.com")` — переход на сайт.

Пример:

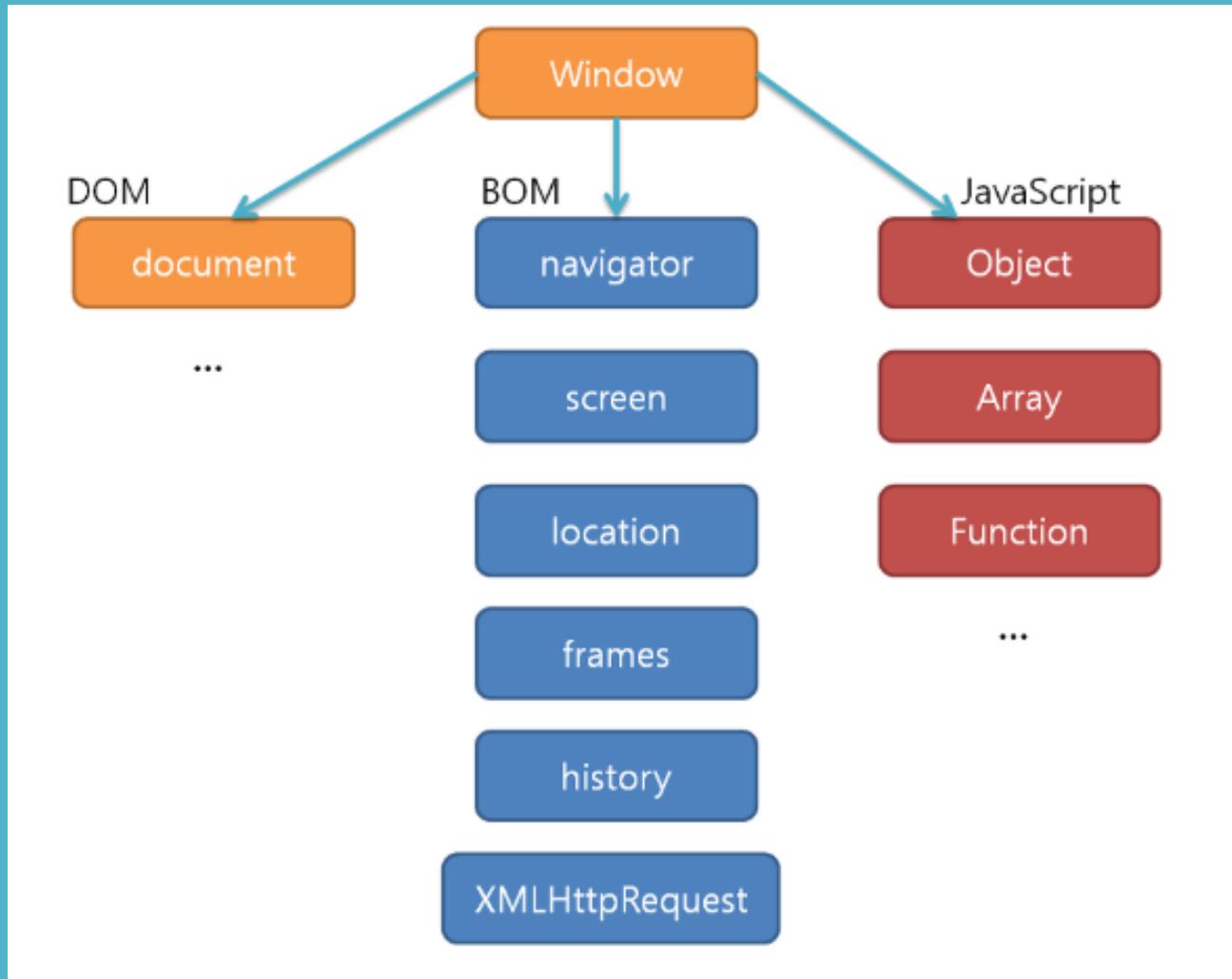
```
console.log(location.href);
```



◆ **history — управление историей браузера.**

- `history.back()` — шаг назад.
- `history.forward()` — шаг вперёд.
- `history.go(-2)` — два шага назад.
- `history.length` — количество записей в истории.

Связь window, BOM и DOM



Главные выводы:

- JavaScript работает в разных средах выполнения (браузер, Node.js).
- В браузере глобальный объект — window, он объединяет всё окружение.
- ВОМ даёт доступ к функциям браузера:
 - navigator — информация о браузере.
 - location — текущий адрес, переходы.
 - history — работа с историей.
- DOM (document) тоже часть window, но отвечает за содержимое страницы.

Контрольные вопросы:

- Для чего используется JavaScript в браузере?
- Как подключить JS к HTML-странице? В чём разница inline- и внешнего скрипта?
- Что такое глобальный объект window и зачем он нужен?

Домашнее задание:

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