Тема 1. Введение в JavaScript.

javascript

хекслет колледж

Цель занятия:

Сформировать целостное представление о языке JavaScript, его назначении и роли в современной веб-разработке.

Учебные вопросы:

1. Понятие JavaScript.
2. Роль JavaScript в веб-разработке
3. Подключение JavaScript к HTML
4. Инструменты разработчика в браузере

1. Понятие JavaScript.

JavaScript — это высокоуровневый язык программирования, предназначенный в первую очередь для создания интерактивных элементов веб-страниц.

Он позволяет управлять поведением сайта: реагировать на действия пользователя, изменять содержимое страницы без перезагрузки и создавать динамические интерфейсы.

Основное назначение JavaScript

JavaScript используется для:

- обработки действий пользователя (нажатия кнопок, ввод данных, прокрутка страницы);
- изменения структуры и содержимого HTML-документа;
- управления стилями элементов страницы;
- реализации логики работы веб-сайта.

В связке с HTML и CSS JavaScript образует классическую тройку веб-разработки:

- HTML — структура страницы;
- CSS — внешний вид и адаптивность;
- JavaScript — поведение и интерактивность.

Особенности JavaScript

Ключевые характеристики языка:

- Интерпретируемый язык — код выполняется непосредственно в браузере без предварительной компиляции;
- Динамическая типизация — тип данных переменной определяется во время выполнения программы;
- Событийно-ориентированное программирование — выполнение кода часто зависит от событий (click, input и др.);
- Кроссбраузерность — поддерживается всеми современными браузерами.

Где выполняется JavaScript?

Изначально JavaScript выполнялся только в браузере, однако сегодня он используется и вне браузера:

- на стороне клиента (веб-страницы);
- на стороне сервера (например, с использованием Node.js);
- в мобильных и десктопных приложениях.

В рамках данного курса основной акцент делается на клиентский JavaScript, используемый для разработки веб-сайтов.

Вывод:

- JavaScript — это ключевой язык современной веб-разработки, который отвечает за **динамику и интерактивность** сайтов.
- Его изучение необходимо для создания полноценных пользовательских интерфейсов и последующей разработки сложных веб-приложений.

2. Роль JavaScript в веб-разработке

В веб-разработке используется разделение ответственности между технологиями:

- HTML отвечает за структуру и содержание страницы;
- CSS определяет внешний вид и адаптивность;
- JavaScript управляет логикой работы страницы и реакцией на действия пользователя.

JavaScript не заменяет HTML или CSS, а дополняет их, обеспечивая динамическое поведение элементов страницы.

Задачи, которые решает JavaScript на сайте

JavaScript используется для решения следующих задач:

- обработка пользовательских событий (click, submit, scroll);
- динамическое изменение содержимого страницы;
- показ и скрытие элементов интерфейса;
- управление стилями и состояниями элементов;
- валидация пользовательского ввода;
- организация навигации и интерактивных компонентов.

Динамика и интерактивность

JavaScript позволяет:

- изменять страницу без перезагрузки;
- обновлять данные и интерфейс в реальном времени;
- создавать отзывчивые пользовательские интерфейсы.

Именно благодаря JavaScript веб-страницы становятся интерактивными и удобными для пользователя.

JavaScript как основа современных веб-интерфейсов

Большинство современных веб-сайтов используют JavaScript для:

- реализации интерактивного меню;
- работы с формами;
- создания модальных окон и вкладок;
- адаптации интерфейса под действия пользователя.

Даже без серверной части JavaScript позволяет создавать функционально насыщенные веб-проекты.

Вывод:

- JavaScript является неотъемлемой частью веб-разработки, отвечающей за логику, интерактивность и динамическое поведение веб-страниц.
- Его знание необходимо для создания современных и удобных сайтов.

3. Подключение JavaScript к HTML

Для того чтобы JavaScript-код выполнялся на веб-странице, его необходимо подключить к HTML-документу.
Существует несколько способов подключения JavaScript, каждый из которых применяется в зависимости от задачи.

Подключение JavaScript прямо в HTML

JavaScript-код можно разместить непосредственно внутри HTML-документа с помощью тега **<script>**. Этот тег сообщает браузеру, что внутри находится JavaScript-код или ссылка на внешний файл со скриптом.
Тег <script> может:

- содержать JavaScript-код напрямую;
- подключать внешний JavaScript-файл.

Пример:

```
<body>
    <h1>Моя первая страница с JavaScript</h1>

    <script>
        console.log("JavaScript подключен и работает");
    </script>
</body>
```

Комментарии:

- код выполняется при загрузке страницы;
- подходит для учебных примеров;
- в реальных проектах используется редко.

2. Подключение внешнего JavaScript-файла

Наиболее правильный и распространённый способ — подключение внешнего JavaScript-файла.
Особенности:

- JavaScript-код хранится в отдельном файле с расширением **.js**;
- HTML отвечает только за структуру;
- JavaScript — за поведение страницы;
- улучшается читаемость и поддержка кода.

У этого способа есть два варианта:

1. Скрипт внизу страницы, перед закрывающим тегом </body>

2. Скрипт в <head> с **defer** (современный вариант)

Пример 1:

JS1

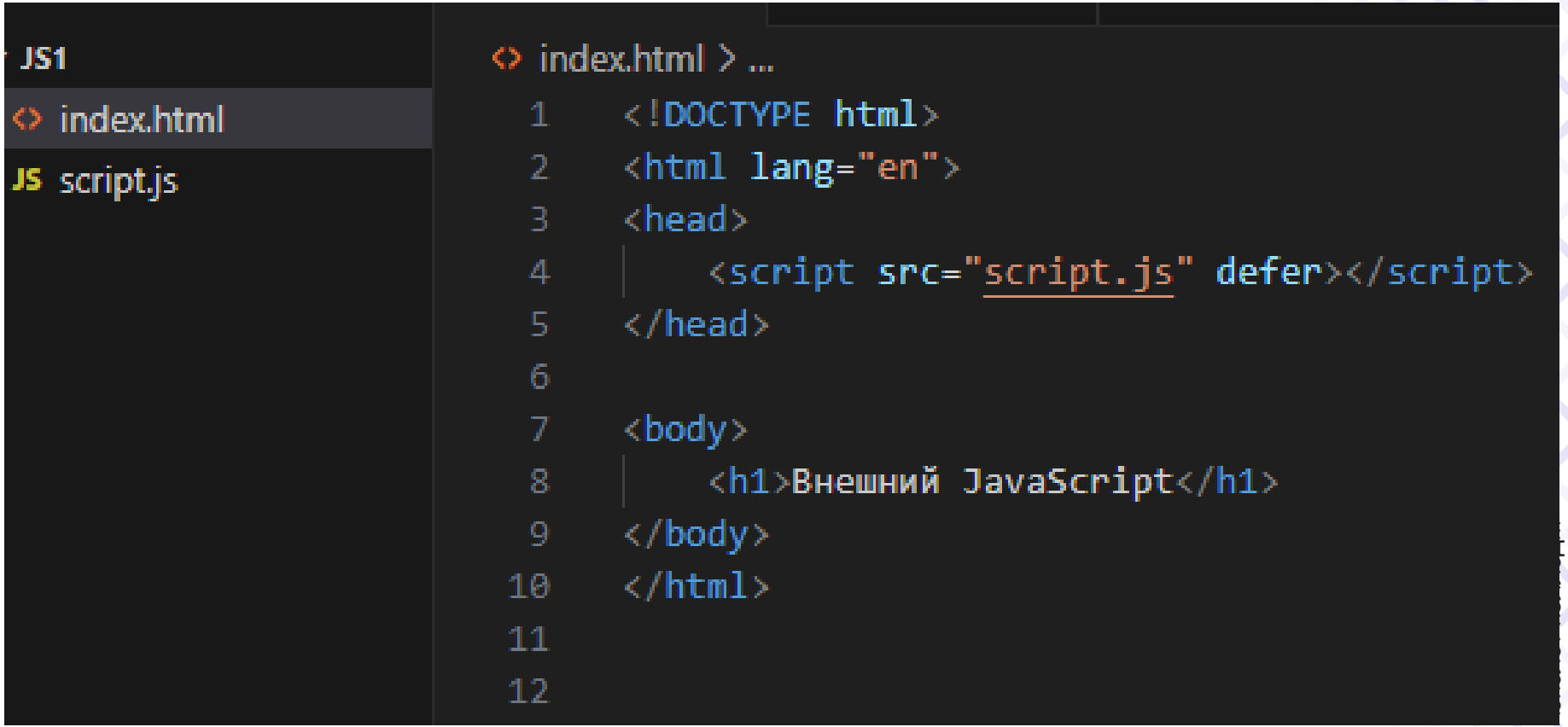
index.html

script.js

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <!-- <meta name="viewport" content=
6       <title>Document</title>
7   </head>
8   <body>
9
10      <h1>Внешний JavaScript</h1>
11
12      <script src="script.js"></script>
13  </body>
14  </html>
```

Пример 2:


JS1
index.html
script.js
index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <script src="script.js" defer></script>
5 </head>
6
7 <body>
8 <h1>Внешний JavaScript</h1>
9 </body>
10 </html>
11
12

Почему важно подключать скрипт в конце <body> и использовать defer

Браузер загружает страницу сверху вниз:

- читает HTML,
- создаёт элементы страницы,
- показывает страницу пользователю.

Если браузер встречает JavaScript слишком рано, он останавливается, выполняет скрипт и только потом продолжает загружать страницу.

Представим ситуацию:

- JavaScript пытается найти кнопку;
- кнопка ещё не загружена;
- браузер не понимает, с чем работать.

Почему скрипт подключают в конце <body>

- сначала браузер создаёт всю страницу;
- потом запускает JavaScript;
- JavaScript «видит» все элементы и может с ними работать.

Что делает defer:

- браузер загружает JavaScript параллельно с HTML;
- не останавливает загрузку страницы;
- запускает скрипт только после полной загрузки HTML.

Простыми словами:
«Сначала построй страницу, потом выполняй JavaScript».

Практические рекомендации:

- Использовать внешние JavaScript-файлы;
- Подключать скрипты:
- либо перед </body>,
- либо в <head> с атрибутом defer;
- Проверять подключение с помощью console.log;
- Следить за путями к файлам.

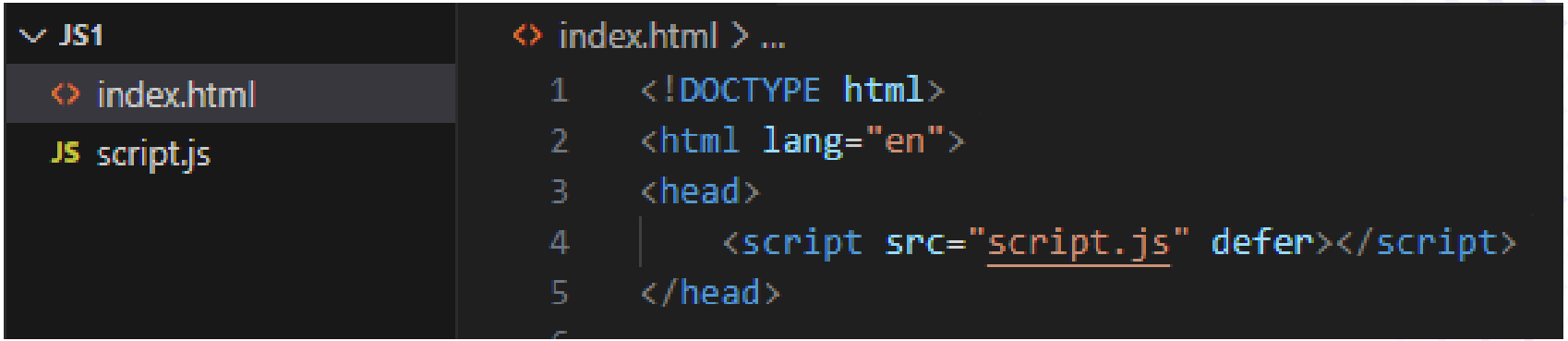
Пути к файлам при подключении JavaScript

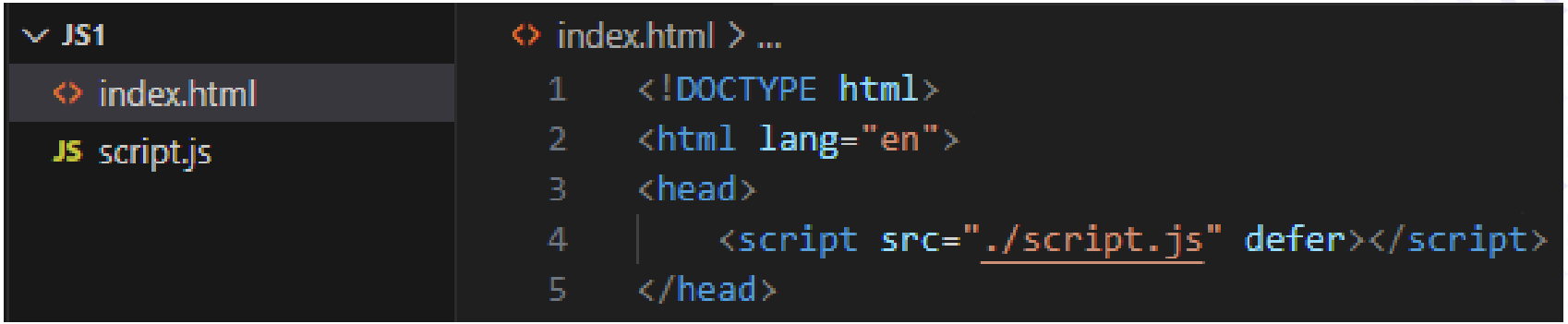
Чтобы браузер смог подключить внешний JavaScript-файл, необходимо правильно указать путь к нему.
Путь показывает, где именно находится файл относительно HTML-документа.

Относительные пути (основной вариант)

Относительный путь указывается относительно текущего HTML-файла.

Пример 1. Файл рядом с HTML





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

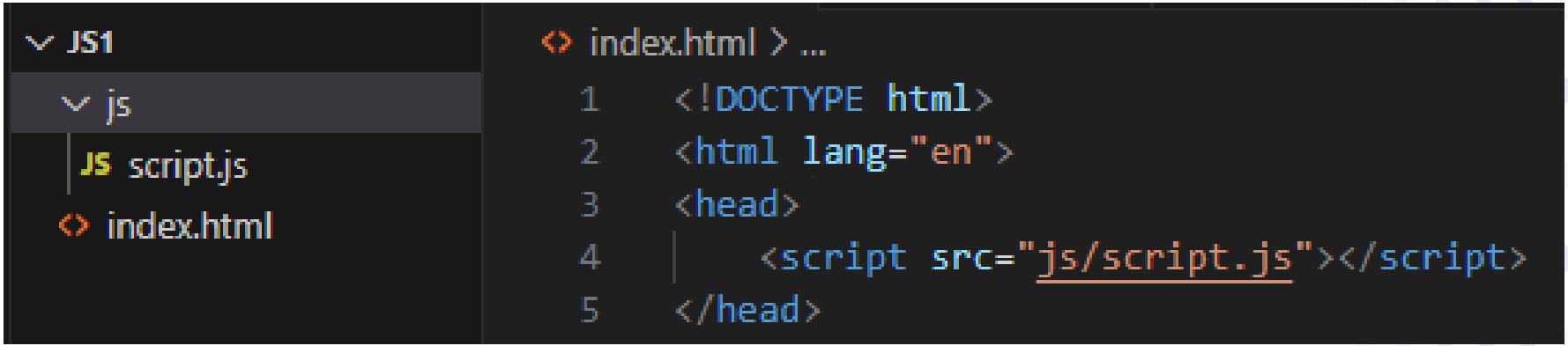
- Путь "script.js" — относительный путь.
- Браузер ищет файл в той же папке, где лежит HTML-документ.
- Если файл там есть — загрузка успешна.

или:

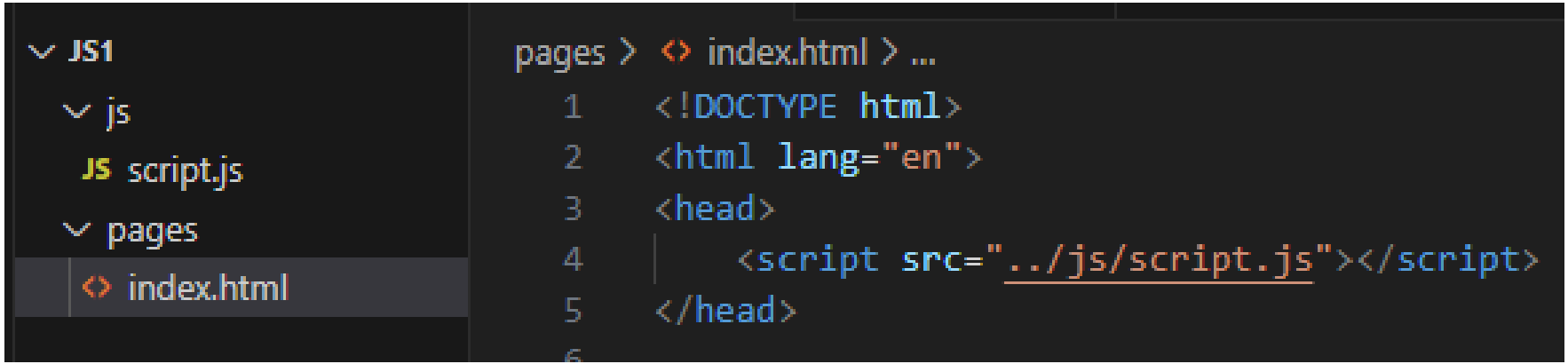
<script src="./script.js" defer></script>

- Путь "./script.js" тоже относительный.
- **./** буквально означает: «текущая папка».
- Результат точно такой же, как и просто "script.js".

Пример 2. JavaScript в папке js



Пример 3. Переход на уровень вверх (..)

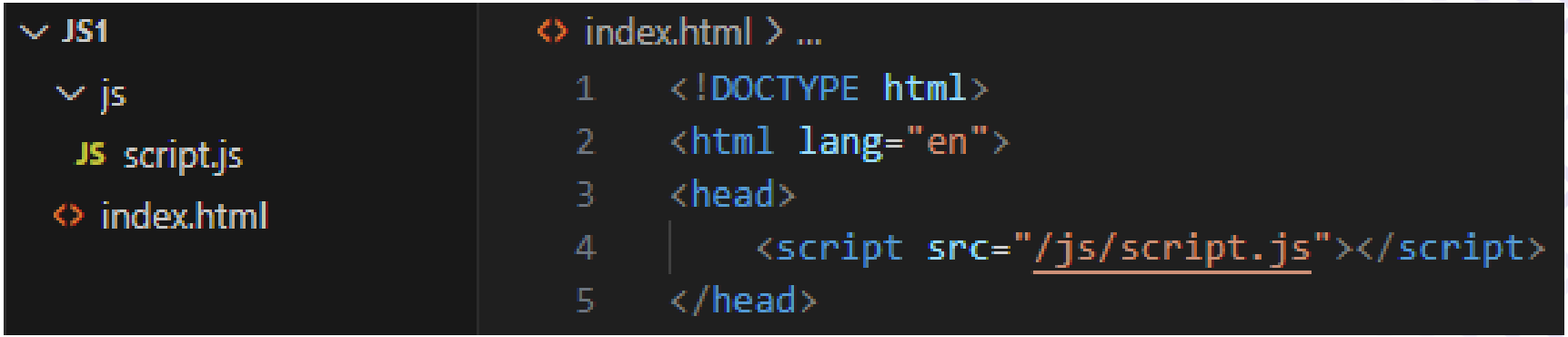


.. означает «подняться на уровень выше».

Абсолютный путь

- Абсолютный путь означает путь от корня веб-сайта.
- Он начинается с **/** и указывает браузеру: «найди файл начиная с корня сайта».

Пример:



Важно!

❌ Не использовать

Полные пути на диске типа

C:\Users\user\Documents\js1\script.js

— браузер не найдёт файл.

4. Инструменты разработчика в браузере

Инструменты разработчика (DevTools) — встроенный в браузер набор инструментов для отладки, тестирования и анализа веб-страниц. Открываются:

- F12 (Windows/Linux)
- Cmd + Option + I (Mac)
- Правая кнопка мыши → "Исследовать элемент"
- Меню браузера → Дополнительные инструменты

Обзор основных вкладок DevTools

Elements (Элементы):

- Просмотр и редактирование HTML/CSS в реальном времени
- Инспектор элементов (Ctrl+Shift+C)
- Изменение стилей "на лету"

Console (Консоль):

- Выполнение JS-кода
- Вывод сообщений и ошибок
- Ранее рассмотрена подробно

Sources (Источники):

- Просмотр файлов проекта
- Отладка кода: точки останова, пошаговое выполнение
- Редактирование и сохранение изменений

Network (Сеть):

- Мониторинг всех сетевых запросов
- Анализ времени загрузки
- Просмотр заголовков и ответов

Application (Приложение):

- Работа с LocalStorage, SessionStorage, Cookies
- Просмотр кэша
- Базы данных IndexedDB

Performance (Производительность):

- Запись и анализ производительности
- Выявление "узких мест"

Security (Безопасность):

- Информация о сертификатах
- Проверка HTTPS

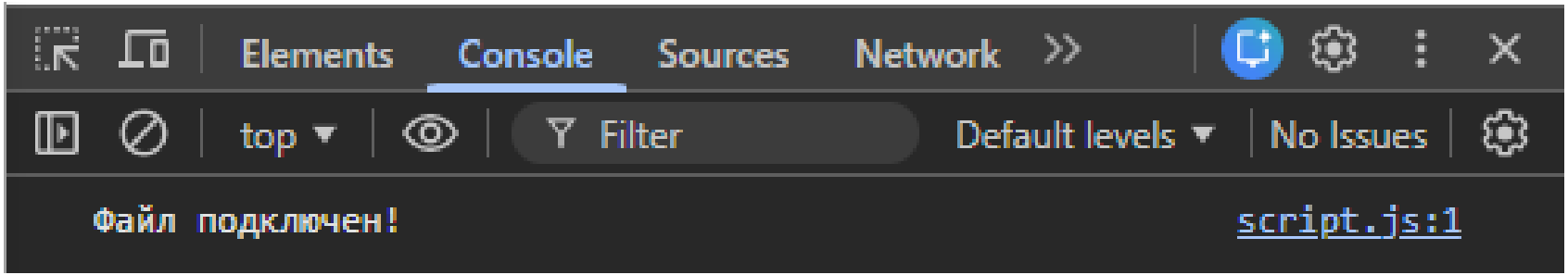
Консоль браузера как среда выполнения JS

- Консоль — интерактивная среда для выполнения JavaScript-кода в реальном времени:
- Вводите код → сразу видите результат
- Идеально для экспериментов и быстрой проверки
- Имеет доступ ко всем объектам текущей страницы

Основные методы console:

console.log() - основной вывод, пример:

console.log("Файл подключен!")



5. Инструкции в JavaScript.

JavaScript-программа состоит из инструкций (statements).
Инструкция — это законченное действие, выполняемое интерпретатором.
Код читается и выполняется сверху вниз, последовательно.

Определение инструкции.

Инструкция — строка кода, выполняющая одно действие.

```
1    let x = 5;          // объявление переменной
2    console.log(x);     // вызов функции
3    x = x + 2;          // присваивание
```

Разделение инструкций.

- Обычно пишутся с новой строки.
- В конце можно ставить точку с запятой ;.
- Автоматическая подстановка работает, но лучше ставить ; всегда (или нет 🙄).

Блок инструкций.

Несколько инструкций объединяются в блок { ... }:

```
{
  let x = 1;
  let y = 2;
  console.log(x + y);
}
```

- Переносы строк обычно также игнорируются и считаются разделителями инструкций:

```
let x = 5
let y = 10
console.log(x + y)
```

```
// работает так же, как с ;
let x = 5; let y = 10; console.log(x + y)
```

Комментарии

Однострочные:

```
// Это комментарий
```

Многострочные:

```
/*
  Это многострочный
  комментарий
*/
```

Инструкции для браузера (Web API)

- **alert(message)** — показывает сообщение.
- **prompt(message, default)** — спрашивает ввод у пользователя.
- **confirm(message)** — диалог «ОК / Отмена».

alert(message)

Синтаксис:

alert(message);

message — строка или значение, которое будет преобразовано в строку

Примеры использования:

Простое сообщение:

```
alert("Добро пожаловать на наш сайт!");
```

С переменными:

```
let userName = "Анна";
alert("Привет, " + userName);
```

prompt()

Синтаксис:

let result = prompt(message, default);

- message — текст сообщения для пользователя (обязательный)
- default — начальное значение в поле ввода (необязательный)

Возвращает строку или null

Пример использования:

```
let userName = prompt("Введите ваше имя:");
console.log("Привет, " + userName);
```

confirm()

это встроенная функция JavaScript, которая отображает модальное диалоговое окно с сообщением и двумя кнопками: "OK" (Подтвердить) и "Отмена".

```
let result = confirm("Вы уверены?");
console.log(result); // true или false
```

Контрольные вопросы:

- Что такое JavaScript и для чего он используется на веб-страницах?
- Чем JavaScript отличается от HTML и CSS?
- Назовите хотя бы два свойства JavaScript как языка программирования.
- Какие существуют два способа подключения JavaScript к HTML?
- Как правильно подключить внешний JavaScript-файл? Приведите пример.
- Почему важно подключать скрипт в конце <body> или использовать атрибут defer?
- В чем разница между относительным и абсолютным путём к файлу?
- Как проверить, что JavaScript-файл подключён и работает?
- Как с помощью DevTools можно быстро проверить, работает ли подключённый скрипт?

Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ