

Практика 2-JavaScript

Тлеубаева А.О.,

87071357797

- Сегодняшний мир веб-сайтов трудно представить без языка JavaScript. JavaScript - это то, что делает живыми веб-страницы, которые мы каждый день просматриваем в своем веб-браузере.
- JavaScript был создан в 1995 году в компании Netscape разработчиком Брендоном Айком (Brendon Eich) в качестве языка сценариев в браузере Netscape Navigator 2. Первоначально язык назывался LiveScript, но на волне популярности в тот момент другого языка Java LiveScript был переименован в JavaScript. Однако данный момент до сих пор иногда приводит к некоторой путанице: некоторые начинающие разработчики считают, что Java и JavaScript чуть ли не один и тот же язык. Нет, это абсолютно два разных языка, и они связаны только по названию.
- Первоначально JavaScript обладал довольно небольшими возможностями. Его цель состояла лишь в том, чтобы добавить немного поведения на веб-страницу. Например, обработать нажатие кнопок на веб-странице, произвести какие-нибудь другие действия, связанные прежде всего с элементами управления.

- Однако развитие веб-среды, появление HTML5 и технологии Node.js открыло перед JavaScript гораздо большие горизонты. Сейчас JavaScript продолжает использоваться для создания веб-сайтов, только теперь он предоставляет гораздо больше возможностей.
- Также он применяется как язык серверной стороны. То есть если раньше JavaScript применялся только на веб-странице, а на стороне сервера нам надо было использовать такие технологии, как PHP, ASP.NET, Ruby, Java, то сейчас благодаря Node.js мы можем обрабатывать все запросы к серверу также с помощью JavaScript.
- В последнее время переживает бум сфера мобильной разработки. И JavaScript опять же не остается в стороне: увеличение мощности устройств и повсеместное распространение стандарта HTML5 привело к тому, что для создания приложений для смартфонов, планшетов и настольных компьютеров мы также можем использовать JavaScript. То есть JavaScript уже перешагнул границы веб-браузера, которые ему были очерчены при его создании.

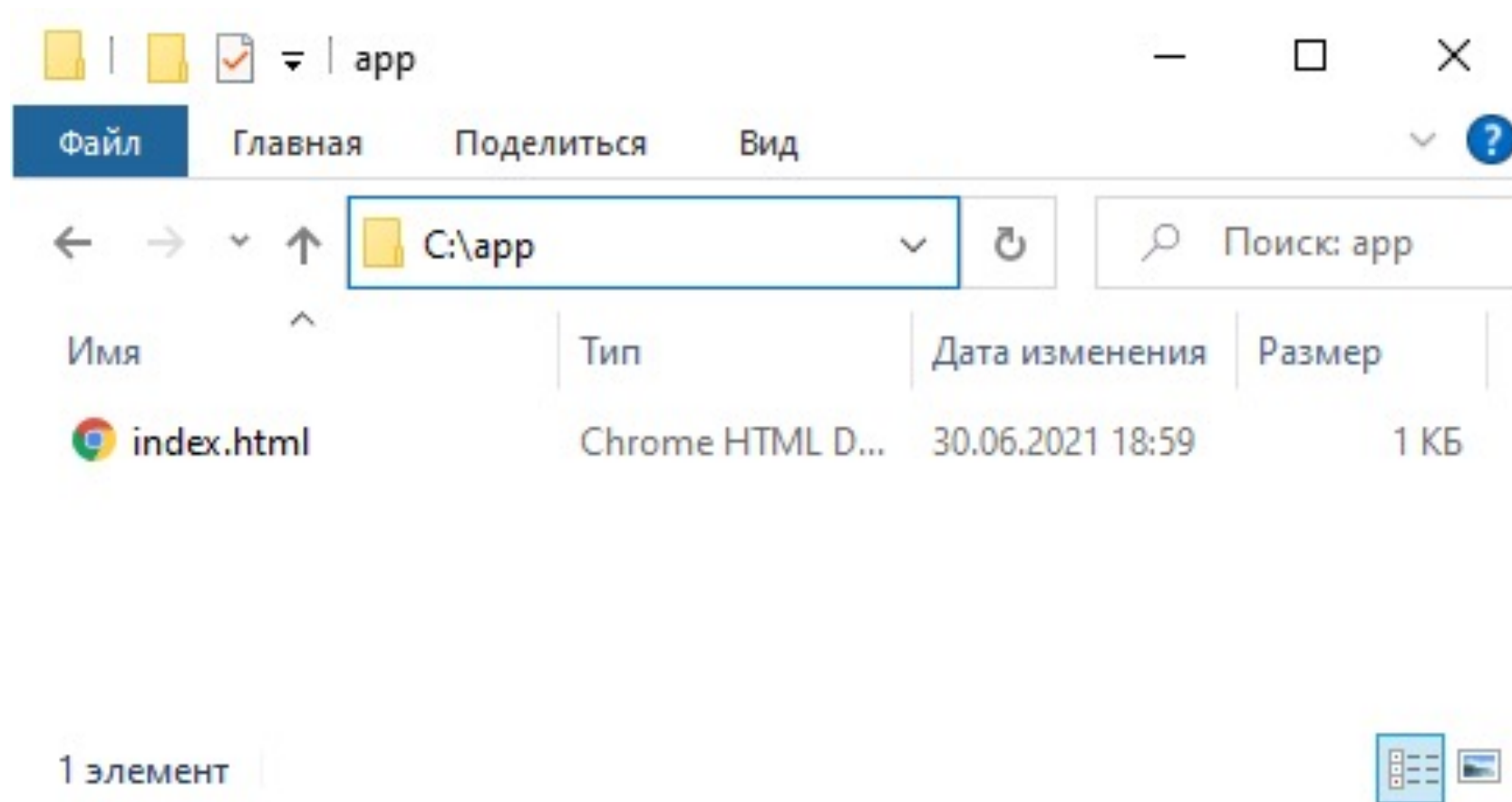
- И что вообще раньше казалось фантастикой, но сегодня стало реальностью - javascript может использоваться для набирающего популярность направления разработки для IoT(Internet of Things или Интернет вещей). То есть JavaScript можно использовать для программирования самых различных "умных" устройств, которые взаимодействуют с интернетом.
- Таким образом, вы можете встретить применение JavaScript практически повсюду. Сегодня это действительно один из самых популярных языков программирования, и его популярность еще будет расти.
- JavaScript является интерпретируемым языком. Это значит, что код на языке JavaScript выполняется с помощью интерпретатора. Интерпретатор получает инструкции языка JavaScript, которые определены на веб-странице, выполняет их (или интерпретирует).

- **Средства разработки**

- Для разработки на JavaScript нам потребуется текстовый редактор для написания кода и веб-браузер для его тестирования. В качестве текстового редактора я советую использовать такую программу как [Visual Studio Code](#). Он бесплатен, имеет много возможностей и может быть установлен как на Windows, так и на Linux и MacOS. Хотя этот может быть любой другой текстовый редактор.
- Также существуют различные среды разработки, которые поддерживают JavaScript и облегчают разработку на этом языке, например, Visual Studio, WebStorm, Netbeans и так далее. При желании можно использовать также эти среды разработки.
- Итак, приступим к созданию первой программы.

- Создадим первую программу на javascript. Для написания и тестирования программ на JavaScript нам потребуются две вещи: тестовый редактор и веб-браузер.
- В качестве текстового редактора можно взять любой, который нравится - Atom, Sublime Text, Visual Studio Code, Notepad++ и другие. В данном руководстве я буду ориентироваться на текстовый редактор [Visual Studio Code](#), поскольку он является наиболее популярным.
- В качестве браузера также можно взять последние версии любого предпочтительного веб-браузера. В настоящем руководстве я буду преимущественно ориентироваться на Google Chrome.

Для начала определим для нашего приложения какой-нибудь каталог. Например, создадим на диске C папку **app**. В этой папке создадим файл под названием **index.html**. То есть данный файл будет представлять веб-страницу с кодом HTML.

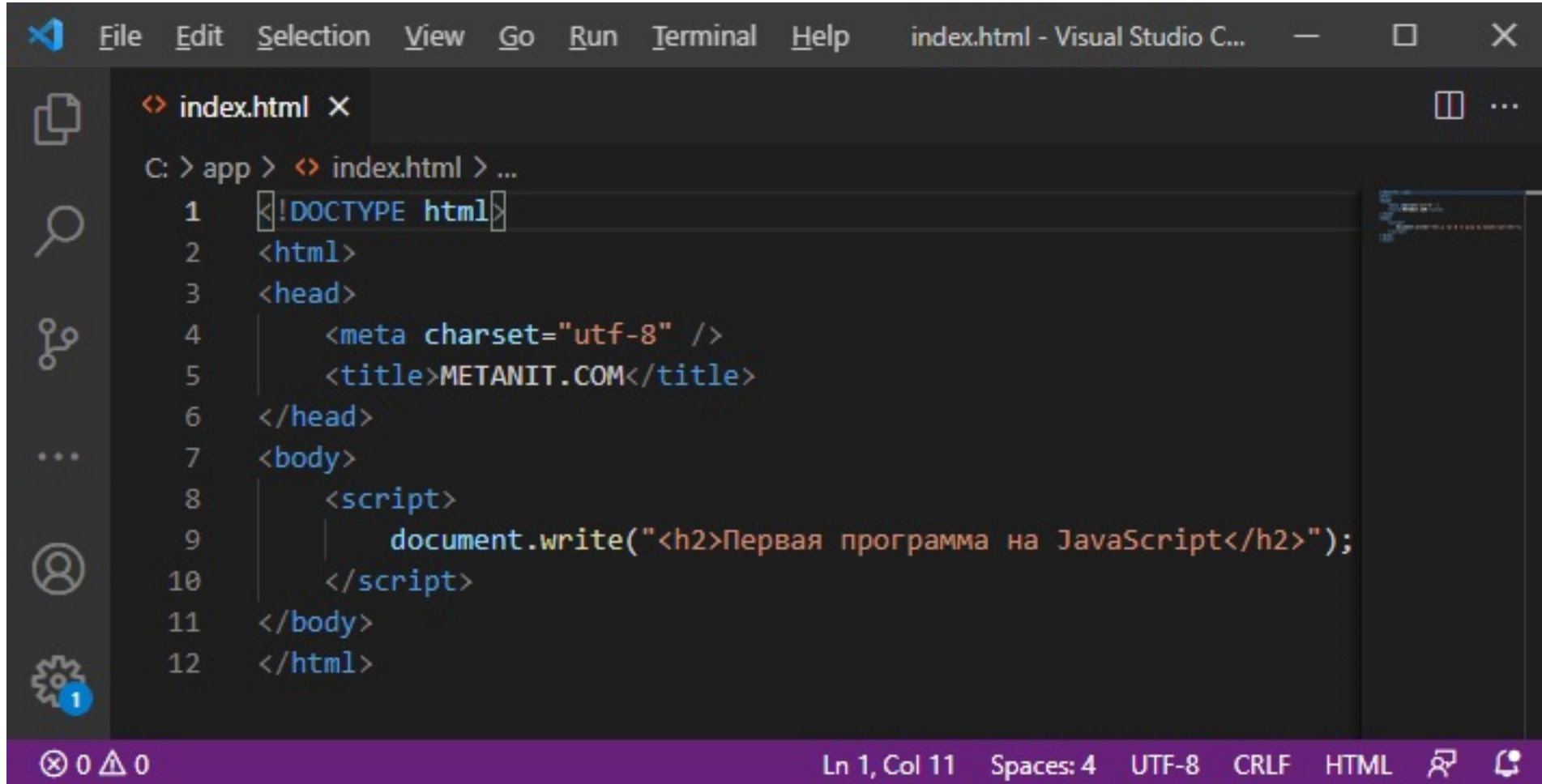


Откроем этот файл в текстовом редакторе и определим в файле следующий код:

- `<!DOCTYPE html>`
- `<html>`
- `<head>`
- `<meta charset="utf-8" />`
- `<title>METANIT.COM</title>`
- `</head>`
- `<body>`
- `<script>`
- `document.write("<h2>Первая программа на JavaScript</h2>");`
- `</script>`
- `</body>`
- `</html>`

- Здесь мы определяем стандартные элементы html. В элементе **head** определяется кодировка utf-8 и заголовок (элемент title). В элементе **body** определяется тело веб-страницы, которое в данном случае состоит только из одного элемента **<script>**
- Подключение кода javascript на html-страницу осуществляется с помощью тега **<script>**. Данный тег следует размещать либо в заголовке (между тегами <head> и </head>), либо в теле веб-странице (между тегами <body> и </body>). Нередко подключение скриптов происходит перед закрывающим тегом </body> для оптимизации загрузки веб-страницы.
- Раньше надо было в теге <script> указывать тип скрипта, так как данный тег может использоваться не только для подключения инструкций javascript, но и для других целей. Так, даже сейчас вы можете встретить на некоторых веб-страницах такое определение элемента script:
- <script type="text/javascript"> Но в настоящее время предпочтительнее опускать атрибут type, так как браузеры по умолчанию считают, что элемент script содержит инструкции javascript.
- Используемый нами код javascript содержит одно выражение:
- document.write("<h2>Первая программа на JavaScript</h2>"); Код javascript может содержать множество инструкций и каждая инструкция завершается точкой с запятой. Наша инструкция вызывает метод **document.write()**, который выводит на веб-страницу некоторое содержимое, в данном случае это заголовок <h2>Первая программа на JavaScript</h2>.

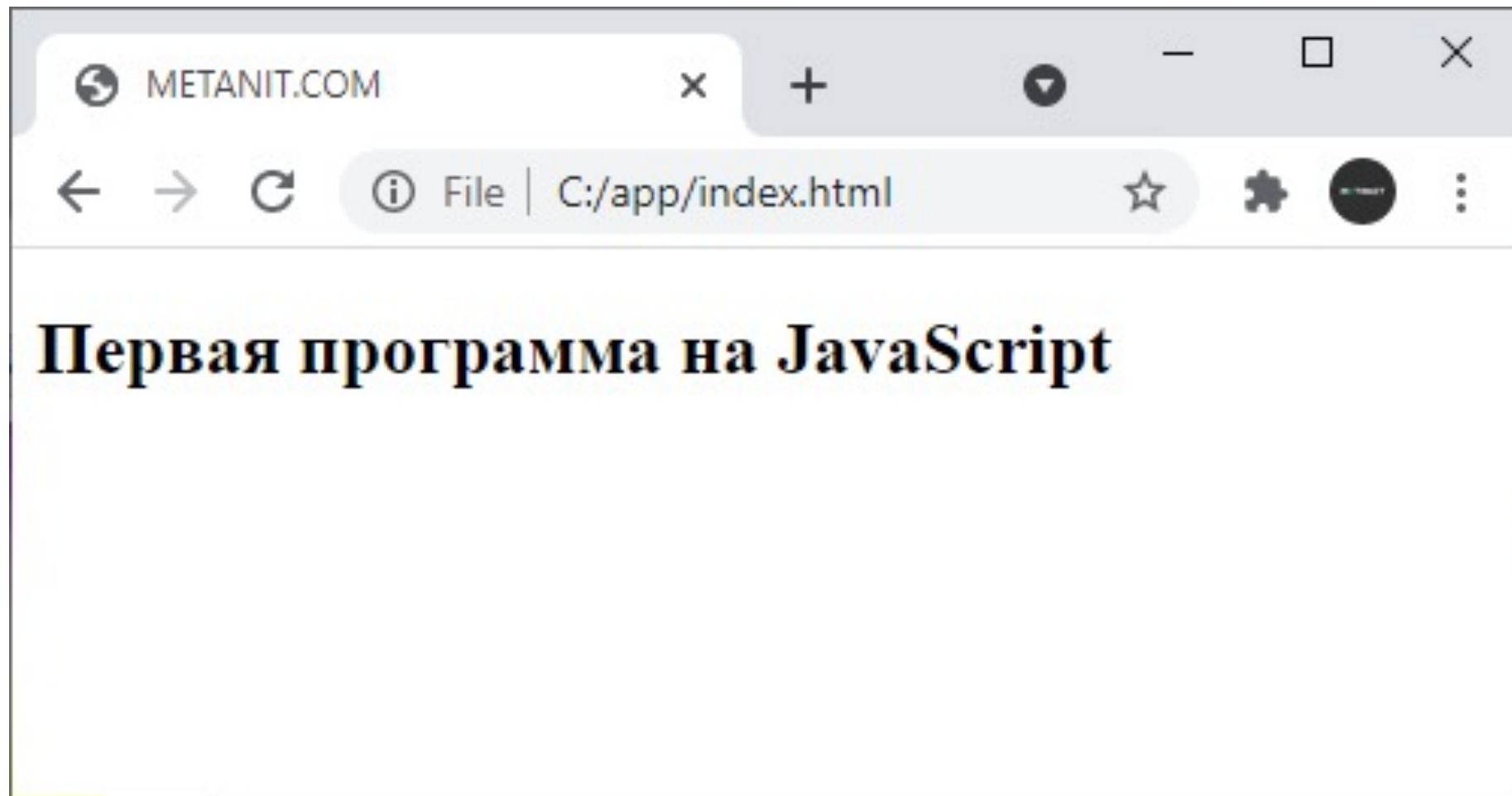
Вид файла в текстовом редакторе Visual Studio Code:



```
File Edit Selection View Go Run Terminal Help index.html - Visual Studio C...
index.html X
C: > app > index.html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8" />
5     <title>METANIT.COM</title>
6 </head>
7 <body>
8     <script>
9         document.write("<h2>Первая программа на JavaScript</h2>");
10    </script>
11 </body>
12 </html>
```

⊗ 0 △ 0 Ln 1, Col 11 Spaces: 4 UTF-8 CRLF HTML

Теперь, когда веб-страница готова, откроем ее в веб-браузере:



Когда браузер получает веб-страницу с кодом html и javascript, то он ее интерпретирует. Результат интерпретации в виде различных элементов - кнопок, полей ввода, текстовых блоков и т.д., мы видим перед собой в браузере. Интерпретация веб-страницы происходит последовательно сверху вниз.

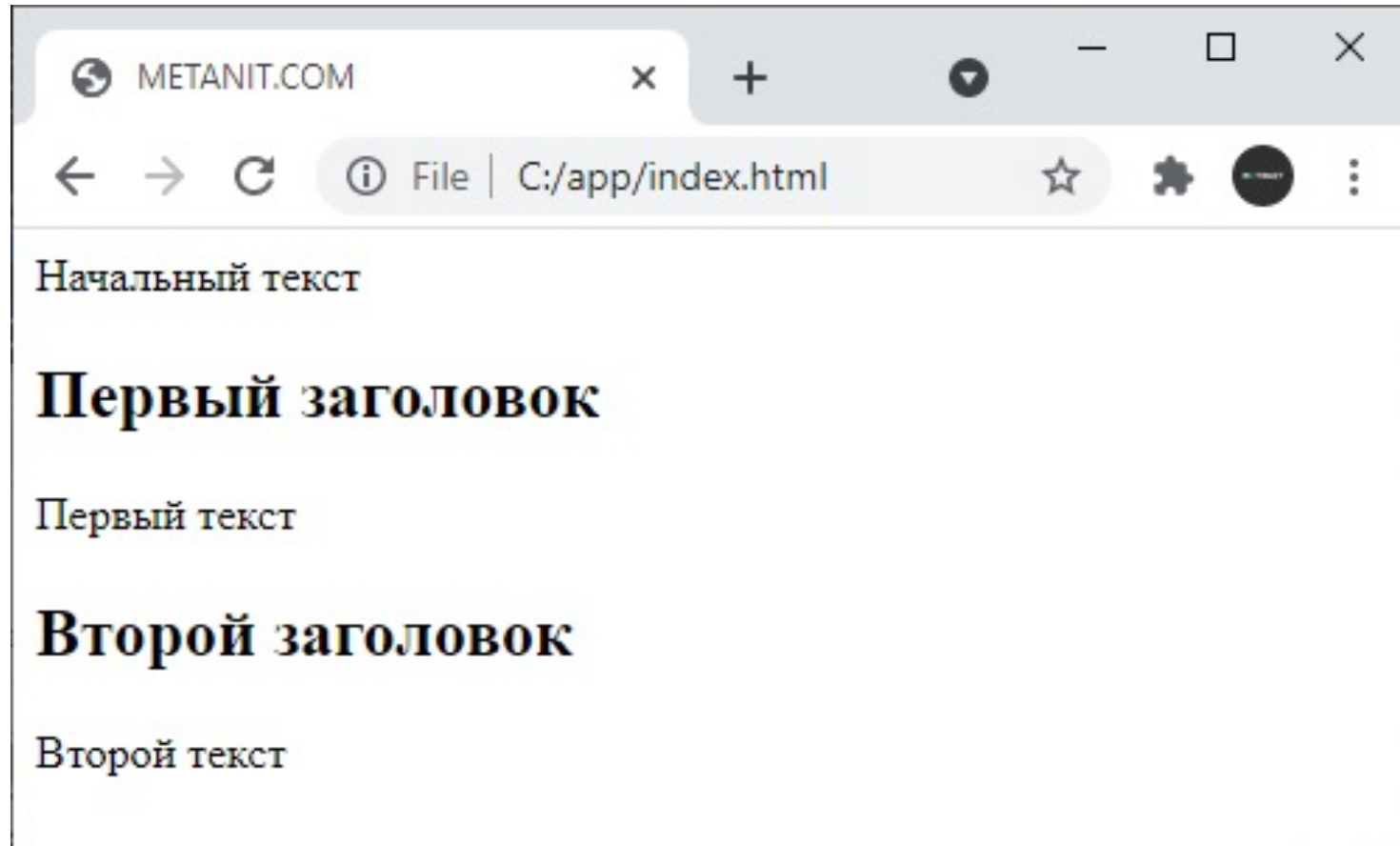
Когда браузер встречает на веб-странице элемент `<script>` с кодом javascript, то вступает в действие встроенный интерпретатор javascript. И пока он не закончит свою работу, дальше интерпретация веб-страницы не идет.

Рассмотрим небольшой пример и для этого изменим страницу index.html из прошлой темы следующим образом:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>METANIT.COM</title>
  <script>
    document.write("Начальный текст");
  </script>
</head>
<body>
  <h2>Первый заголовок</h2>
  <script>
    document.write("Первый текст");
  </script>
  <h2>Второй заголовок</h2>
  <script>
    document.write("Второй текст");
  </script>
</body>
</html>
```

Здесь три вставки кода javascript - один в секции <head> и по одному после каждого заголовка.

Откроем веб-страницу в браузере и мы увидим, что браузер последовательно выполняет код веб-страницы:



- Здесь мы видим, что вначале выполняется код javascript из секции head, который выводит на веб-страницу некоторый текст:
- `document.write("Начальный текст");` Далее выводится первый стандартный html-элемент `<h2>`:
- `<h2>Первый заголовок</h2>` После этого выполняется вторая вставка кода на javascript:
- `document.write("Первый текст");` Затем выводится второй html-элемент `<h2>`:
- `<h2>Второй заголовок</h2>` И в конце выполняется последняя вставка кода на javascript:
- `document.write("Второй текст");` После этого браузер закончит интерпретацию веб-страницы, и веб-страница окажется полностью загружена. Данный момент очень важен, поскольку может влиять на производительность. Поэтому нередко вставки кода javascript идут перед закрывающим тегом `</body>`, когда основная часть веб-страницы уже загружена в браузере.

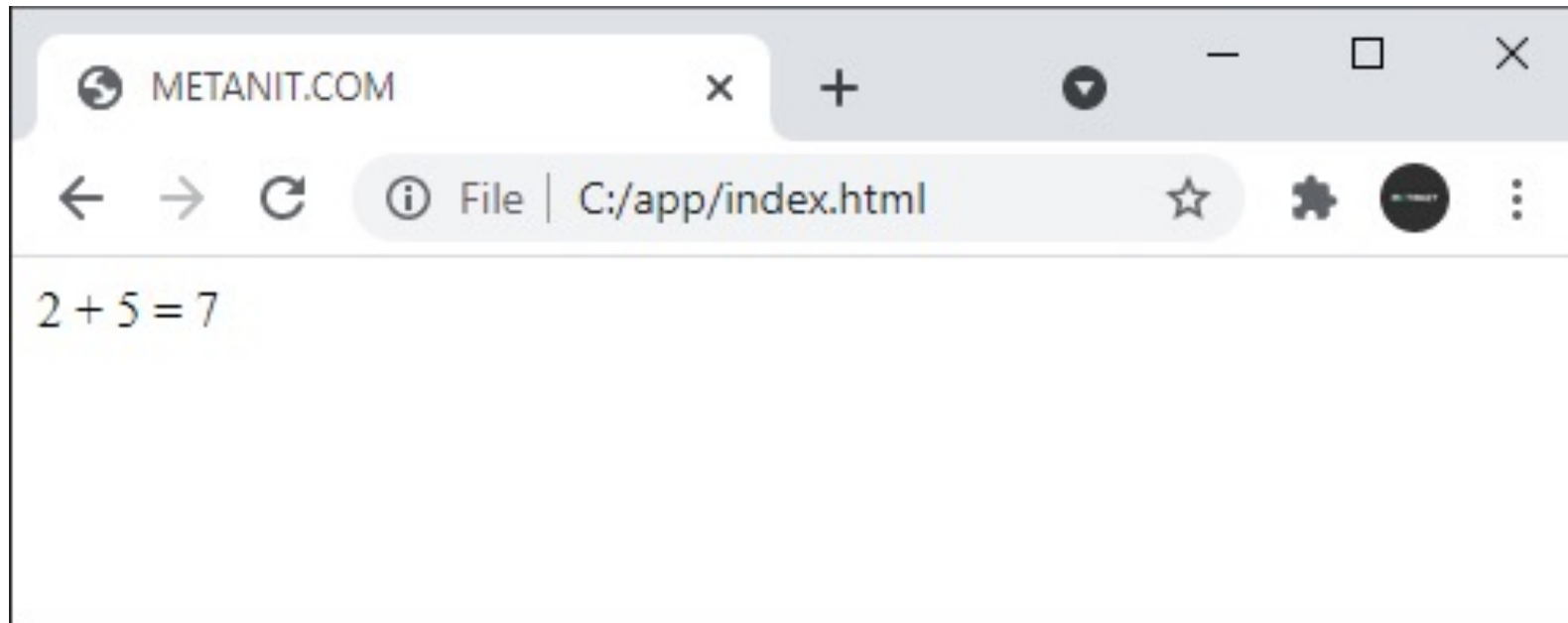
Основы синтаксиса javascript

Прежде чем переходить к детальному изучению основ языка программирования javascript, рассмотрим некоторые базовые моменты его синтаксиса.

Код javascript состоит из инструкций. Каждая инструкция представляет некоторое действие. И для отделения инструкций друг от друга в javascript после инструкции ставится точка с запятой:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>METANIT.COM</title>
</head>
<body>
  <script>
    document.write("2 + 5 = "); var sum = 2 + 5; document.write(sum);
  </script>
</body>
</html>
```

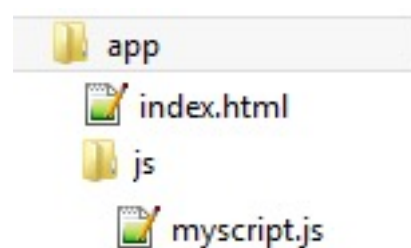

- Здесь в коде javascript определены три инструкции:
- `document.write("2 + 5 = ")` Выводит на страницу текст "2 + 5 = "
- `var sum = 2 + 5;` С помощью оператора **var** определяет переменную **sum**, которая равна сумме 2 + 5
- `document.write(sum);` Выводит на страницу значение переменной **sum** (то есть сумму 2 + 5)



Подключение внешнего файла JavaScript

Есть еще один способ подключения кода JavaScript, который представляет вынесение кода во внешние файлы и их подключение с помощью тега `<script>`

Итак, в прошлой теме мы создали html-страницу **index.html**, которая находится в каталоге **app**. Теперь создадим в этом каталоге новый подкаталог. Назовем его **js**. Он будет предназначен для хранения файлов с кодом javascript. В этом подкаталоге создадим новый текстовый файл, который назовем **myscript.js**. Файлы с кодом javascript имеют расширение **.js**. То есть у нас получится следующая структура проекта в папке **app**:



- Откроем файл **myscript.js** в текстовом редакторе и определим в нем следующий код:

```
1 document.write("<h2>Первая программа на JavaScript</h2>"); // выводим заголовок
2 document.write("Привет мир!"); // выводим обычный текст
```

Здесь для добавления на веб-страницу некоторого содержимого применяется метод `document.write`. Первый вызов метода `document.write` выводит заголовок `<h2>`, а второй вызов - обычный текст.

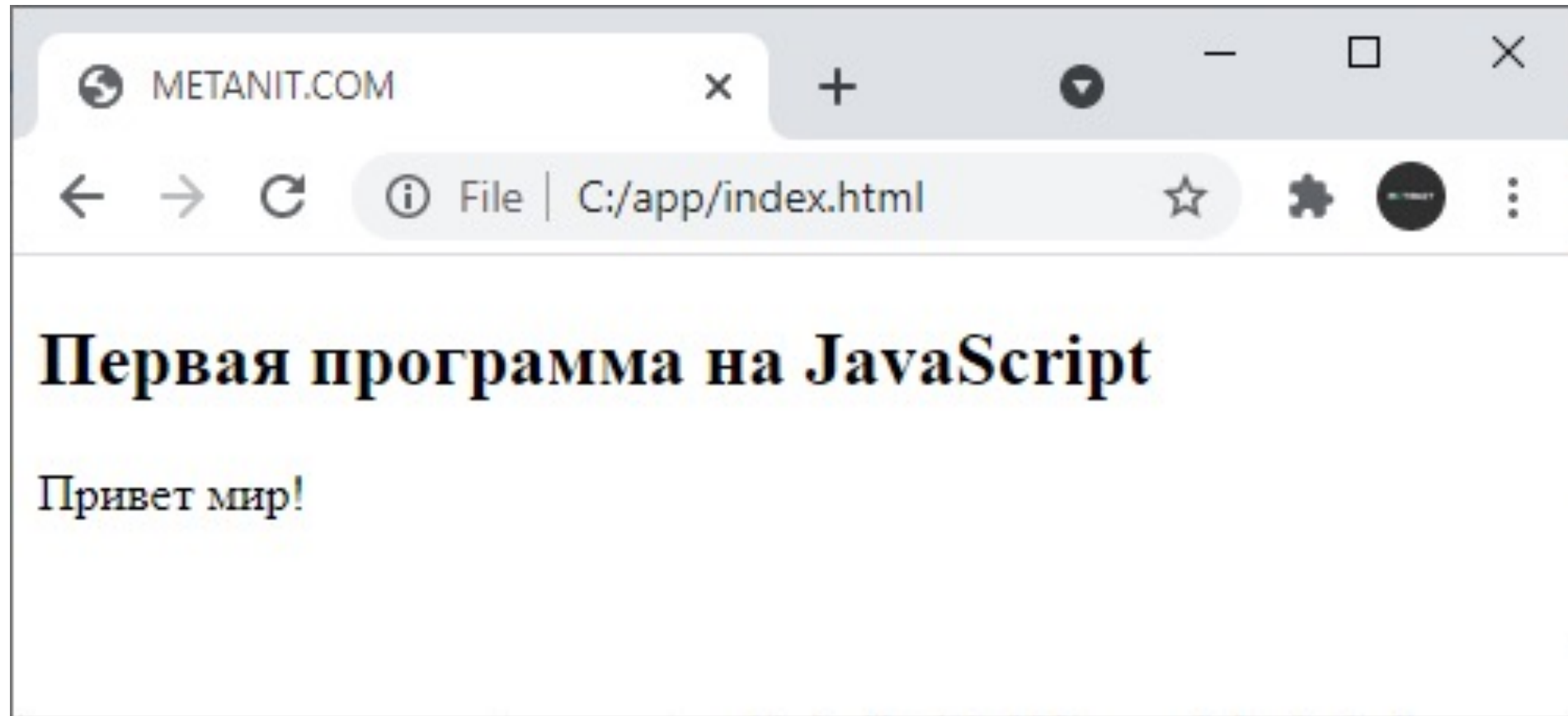
- Для совместимости с кодировкой страницы `index.html` для файла с кодом `javascript` также желательно устанавливать кодировку `utf-8`. При работе в Visual Studio Code этот редактор автоматически устанавливает кодировку `UTF-8`.

Теперь подключим этот файл на веб-страницу `index.html`:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title>METANIT.COM</title>
6  </head>
7  <body>
8      <script src="js/myscript.js"></script>
9  </body>
10 </html>
```

- Чтобы подключить файл с кодом javascript на веб-страницу, применяется также тег **<script>**, у которого устанавливается атрибут **src**. Этот атрибут указывает на путь к файлу скрипта. В нашем случае используется относительный путь. Так как веб-страница находится в одной папке с каталогом **js**, то в качестве пути мы можем написать **js/myscript.js**.
- Также надо учитывать, что за открывающим тегом **script** должен обязательно следовать закрывающий **</script>**

И после открытия файла index.html в браузере отобразится сообщение:



В отличие от определения кода javascript вынесение его во внешние файлы имеет ряд преимуществ:

- Мы можем повторно использовать один и тот же код на нескольких веб-страницах
- Внешние файлы javascript браузер может кэшировать, за счет этого при следующем обращении к странице браузер снижает нагрузку на сервер, а браузеру надо загрузить меньший объем информации
- Код веб-страницы становится "чище". Он содержит только html-разметку, а код поведения хранится во внешних файлах. В итоге можно отделить работу по созданию кода html-страницы от написания кода javascript

Поэтому, как правило, предпочтительнее использовать код javascript во внешних файлах, а не в прямых вставках на веб-страницу с помощью элемента script.

Консоль браузера и console.log

- <https://metanit.com/web/javascript/1.5.php>

Основы javascript

Переменные и константы

- <https://metanit.com/web/javascript/2.1.php>

Типы данных

- <https://metanit.com/web/javascript/2.2.php>

Операции

- [https://metanit.com/web/javascript/2.3.phphttps://metanit.com/web/javascript/2.3.php](https://metanit.com/web/javascript/2.3.php)

Условные операторы

- <https://metanit.com/web/javascript/2.8.php>

Преобразования данных

- <https://metanit.com/web/javascript/2.4.php>

Введение в массивы

- <https://metanit.com/web/javascript/2.5.php>

Условные конструкции

- <https://metanit.com/web/javascript/2.6.php>

Циклы

- <https://metanit.com/web/javascript/2.7.php>

Функциональное программирование

Функции

- <https://metanit.com/web/javascript/3.1.php>

Параметры функции

- <https://metanit.com/web/javascript/3.10.php>

Результат функции

- <https://metanit.com/web/javascript/3.11.php>

Область видимости переменных

- <https://metanit.com/web/javascript/3.2.php>

Замыкания и функции IIFE

- <https://metanit.com/web/javascript/3.3.php>

Паттерн Модуль

- <https://metanit.com/web/javascript/3.9.php>

Рекурсивные функции

- <https://metanit.com/web/javascript/3.4.php>

Переопределение функций

- <https://metanit.com/web/javascript/3.5.php>

Hoisting

- <https://metanit.com/web/javascript/3.6.php>

Передача параметров по значению и по ссылке

- <https://metanit.com/web/javascript/3.7.php>

Стрелочные функции

- <https://metanit.com/web/javascript/3.8.php>

Объектно-ориентированное программирование

- Объектно-ориентированное программирование на сегодняшний день является одной из господствующих парадигм в разработке приложений, и в JavaScript мы также можем использовать все преимущества ООП. В то же время применительно к JavaScript объектно-ориентированное программирование имеет некоторые особенности.
- <https://metanit.com/web/javascript/4.1.php>

- Выполнить все практические задания и загрузить в Гитхаб
- Отправить ссылку от репозиторий мне в плотонус либо в личку в телеграм