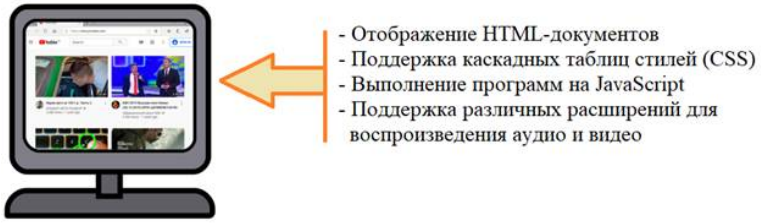
Введение в JavaScript

Смотреть материал на видео

Этим занятием мы открываем с вами курс обучения очень популярному языку программирования JavaScript. Сначала пару слов об этом языке. Жизнь современного человека неотделима от сети Интернет и в частности от просмотра различный сайтов с помощью вашего любимого браузера: Opera, FireFox, Chrome, Yandex и т. п. И именно здесь большинство из нас сталкиваются с JavaScript, то есть, со сценариями – небольшими программами, которые выполняются в браузере и заметно расширяют функциональность веб-страниц. А иногда даже более, чем заметно в случаях браузерных игр. Что же получается? Браузеры могут не только отображать HTML-страницу, но еще и выполнять программы, встроенные в HTML-документ? Да, современные браузеры – это полноценные платформы для отображения самой разнообразной информации сети Интернет.



Теперь вы знаете, что для запуска программ на JavaScript достаточно взять ваш любимый браузер. Что еще нам понадобится? Конечно же приложение в, котором удобно писать программы на JavaScript. Часто это текстовые редакторы, например:

- Atom (кроссплатформенный, бесплатный);
- Sublime Text (кроссплатформенный, условно-бесплатный);
- Notepad++ (Windows, бесплатный).

Я все свои примеры буду показывать в редакторе Sublime Text, который очень удобен как для создания HTML-документов, так и для написания скриптов на JavaScript. Этих двух программ: браузера и текстового редактора, будет вполне достаточно для изучения JavaScript в рамках наших занятий.

Язык JavaScript не следует путать с языком программирования Java. Это совершенно разные языки. Просто создатели JavaScript вначале хотели показать, что JavaScript является как бы «младшим братом» языка Java. Но, впоследствии оказалось, что у них, в общем-то, нет особого родства.

В целях безопасности на скрипты, запускаемые в браузере, наложен ряд ограничений. Основные такие:

- JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы.
- Существуют способы взаимодействия с камерой/микрофоном и другими устройствами, но они требуют явного разрешения пользователя.

- Различные окна/вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов.
- JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов/доменов ограничена.

А вот для чего обычно используется JavaScript:

- Изменять HTML-страницу и модифицировать стили.
- Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
- Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
- Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
- Рисовать в браузере, используя canvas (например, в браузерных играх).

Как и любой другой язык программирования JavaScript имеет свои стандарты, например, ES5 или ES6. Это связано с развитием этого языка: в каждом новом стандарте добавляются новые функции, правила, поддерживаемые им. А, какие-нибудь прежние приемы помечаются как устаревшие и их уже не рекомендуется использовать. Например, стандарт ES5 расшифровывается как ECMAScript 2009. А вот ES6 – это уже ECMAScript 2015 – более свежий стандарт. И здесь кроется некоторая проблема: создавая скрипт под последний свежий стандарт, нужно быть уверенным, что он будет корректно исполняться на браузере клиента. Не секрет, что некоторые пользователи не спешат обновлять свой браузер, что тормозит внедрение новых стандартов представления HTML-документов. И, значит, у части пользователей новые «фичи» на JavaScript могут не работать. Но, так как со времени ES6 прошло уже несколько лет, мы будем рассматривать его в наших занятиях. Кроме того, существует популярный JavaScript-компилятор Babel.JS, который позволяет скрипты стандарта ES6 корректно переводить в аналогичные скрипты стандарта ES5.

Следующий важный вопрос: как быстро исполняются скрипты в браузерах? Здесь можно сказать только, что браузеры делают все возможное для ускорения выполнения таких программ. Сначала они читают (парсят) текст скрипта. Затем, преобразуют (компилируют) в машинный код. И, наконец, запускают полученный код. Все эти этапы выполняет так называемая виртуальная машина JavaScript, встроенная в каждый современный браузер. Они еще называются движками. Например, браузеры Chrome и Opera используют движок V8. А браузер Firefox – движок SpiderMonkey. Другие браузеры применяют другие движки.

Давайте теперь посмотрим как внедряется JavaScript в HTML-документ. Предположим у нас есть вот такая простая страничка.

```
<!DOCTYPE html">
<html>
<head>
        <title>Уроки по JavaScrip
</head>
<body>
</body>
</html>
```

Чтобы браузеру указать где у нас здесь будет находиться программа на JavaScript, мы пишем тег

```
<script type="text/javascript"></
```

И видите? Sublime Text нам сразу добавляет вот этот необязательный атрибут type. Теперь в нем нет необходимости и по умолчанию, все что идет внутри тега <script> считается JavaScript,

поэтому все это можно записать короче, вот так:

```
<script></script>
```

Где можно располагать этот тег? Ограничений особых нет: его можно располагать и в head и в body, в начале страницы или в конце. Но принято код располагать или в разделе head или в конце перед закрывающимся тегом </body>. Почему так? Дело в том, что JavaScript часто работает с элементами (тегами) HTML-документа и если мы пропишем его в начале страницы, то можно столкнуться с проблемой, когда идет обращение к тегу, но он еще не был загружен. Например, вот такая простая программка:

```
        let obj = document.getEl
        console.log(obj);
```

будет работать, если тег

```
<div id="id_div"></div>
```

находится до этого скрипта. А вот если разместить этот тег после него, то в консоле мы увидим значение null, то есть, объект не был найден. Такого точно не произойдет, если скрипт подключается в конце страницы после загрузки всех тегов. Забегая вперед скажу, что в JavaScript есть возможность подождать загрузку документа целиком и только потом выполнять те или иные функции. Именно так реализовано большинство программ. Поэтому расположение самого скрипта в HTML-документе – это скорее дело традиции или привычки.

Часто скрипты хранятся во внешних файлах и лишь подключаются к HTML-документу. Для этого тег script следует записать с атрибутом src:

```
<script src="ex1.js"></script>
```

который указывает путь к файлу с программой на JavaScript. Расширение файла обязательно должно быть js. Этим мы обозначаем файл, содержащий JavaScript код. Теперь обновим HTML-страницу и убедимся, что все работает также.

Так как наш файл располагается в том же каталоге, что и HTML-документ, то в атрибуте src мы просто пишем его имя (без указания каталогов). Однако, в практике создания сайтов принято располагать такие скрипты в каталоге js. Мы так и сделаем. Создаем каталог js и перемещаем туда наш скрипт. Теперь относительный путь к файлу будет выглядеть так:

```
<script src="js/ex1.js"></script>
```

то есть, он прописывается по тем же правилам, что и пути, например, у тега img. Часто так подключается набор JavaScript программ в разделе head, необходимых для корректной работы HTML-документа. И, если нужно подключить несколько разных скриптов, то повторяют тег script:

```
<script src="js/ex1.js"></script>
<script src="js/ex2.js"></script>
```

В качестве примера посмотрим на сайт htmlbook.ru здесь в разделе head именно так подключается внешний скрипт.

В чем преимущество использования внешних скриптов? Во-первых, удобство исправления программ и внесения в них изменений. Если сайт состоит из большого числа HTML-страниц и каждая подключает определенный скрипт, то для изменения функциональности будет достаточно внести изменение в один файл – внешнюю программу на JavaScript и все страницы автоматически начнут работать по новому. Во-вторых, многие браузеры по умолчанию кэшируют загруженную информацию, то есть, скрипт, загруженный с сайта единожды не загружается повторно для других его страниц, а берется из кэша браузера. Это может заметно ускорять загрузку страниц.

И последнее, о чем следует сказать вначале и что сильно упрощает жизнь фронт-енд программиста – это консоль разработчика. В большинстве браузеров она открывается нажатием клавиши F12, в опере следует нажать комбинацию клавиш Ctrl+Shift+I. Здесь будут высвечиваться возможные ошибки в скриптах, а также выводимая ими информация. Вы только что видели как мы выводили информацию об объекте div. Давайте теперь, для примера, запишем программу с ошибкой:

```
let o = document.getElementById('
console.log(obj);
```

Здесь мы пытаемся на консоль вывести несуществующий объект obj, о чем и сообщается в консоле. Этим инструментом браузера мы будем активно пользоваться для отладки наших программ.

Видео по теме

JavaScipt #1: что это такое, с чего начать, как внедрять и запускать

JavaScipt #2: способы объявления переменных константа в стандарте ES

Следующая →