Общие сведения о JavaScript

В основу создания JavaScript была положена идея динамического управления объектами HTML-документов без перезагрузки текущей страницы (так называемые бессерверные сценарии). Со временем возможности языка расширились:

* С помощью JavaScript можно запускать одностраничные приложения на стороне клиента;
* JavaScript используется на стороне сервера с такими технологиями, как Node.js;
* JavaScript помогает создавать настольные приложения с Electron и может использоваться для работы с одноплатными компьютерами типа Raspberry Pi;
* Также, JavaScript используется для обучения моделей машинного обучения в браузере с помощью технологии TensorFlow.js.

Что такое JavaScript?

1) JavaScript — язык сценариев, или скриптов. Скрипт представляет собой программный код — набор инструкций, который не требует предварительной обработки (например, компиляции) перед запуском. Код JavaScript интерпретируется движком браузера во время загрузки веб-страницы. Интерпретатор браузера выполняет построчный анализ, обработку и выполнение исходной программы или запроса.

2) JavaScript — объектно-ориентированный язык с прототипным наследованием. Он поддерживает несколько встроенных объектов, а также позволяет создавать или удалять свои собственные (пользовательские) объекты. Объекты могут наследовать свойства непосредственно друг от друга, образуя цепочку объект-прототип.

* [HTML](https://developer.mozilla.org/ru/docs/Glossary/HTML) - это язык разметки, который мы используем для визуального и смыслового структурирования нашего web контента, например, определяем параграфы, заголовки, таблицы данных, или вставляем изображения и видео на страницу.
* [CSS](https://developer.mozilla.org/ru/docs/Glossary/CSS) - это язык стилей с помощью которого мы придаём стиль отображения нашего HTML контента, например придаём цвет фону (background) и шрифту, придаём контенту многоколоночный вид.
* [JavaScript](https://developer.mozilla.org/ru/docs/Glossary/JavaScript) язык программирования, который позволяет вам создать динамически обновляемый контент, управляет мультимедиа, анимирует изображения, впрочем, делает всё, что угодно. Окей, не все, что угодно, но всё равно, это удивительно, что можно достичь с помощью нескольких строк JavaScript кода.

Три слоя прекрасно выстраиваются друг над другом. Возьмём простой текст для примера. Для придания структуры и смыслового назначения тексту, разметим его с помощью HTML:

<p>Player 1: Chris</p>

Copy to Clipboard

https://mdn.mozillademos.org/files/13422/just-html.png

Затем мы добавим немного CSS, что бы это выглядело симпатичнее:

p {

font-family: 'helvetica neue', helvetica, sans-serif;

letter-spacing: 1px;

text-transform: uppercase;

text-align: center;

border: 2px solid rgba(0,0,200,0.6);

background: rgba(0,0,200,0.3);

color: rgba(0,0,200,0.6);

box-shadow: 1px 1px 2px rgba(0,0,200,0.4);

border-radius: 10px;

padding: 3px 10px;

display: inline-block;

cursor:pointer;

}

Copy to Clipboard

https://mdn.mozillademos.org/files/13424/html-and-css.png

И наконец, добавим немного JavaScript для придания динамического поведения:

const para = document.querySelector('p');

para.addEventListener('click', updateName);

function updateName() {

let name = prompt('Enter a new name');

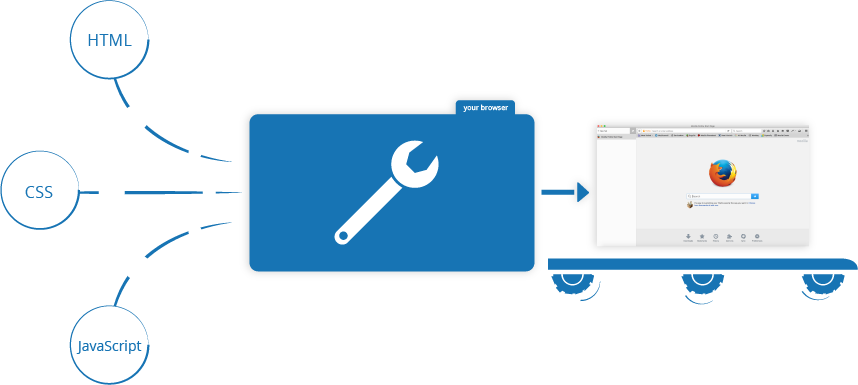
para.textContent = 'Player 1: ' + name;

}

Попробуйте кликнуть по тексту чтобы увидеть, что произойдёт

[**Что JavaScript делает на вашей странице?**](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/What_is_JavaScript#%D1%87%D1%82%D0%BE_javascript_%D0%B4%D0%B5%D0%BB%D0%B0%D0%B5%D1%82_%D0%BD%D0%B0_%D0%B2%D0%B0%D1%88%D0%B5%D0%B9_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B5)

Давайте составим краткий бриф, что же происходит когда мы загружаем страничку в браузере (первое упоминание в статье [Как работает CSS](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_works#how_does_css_actually_work)). Когда вы загружаете страничку в браузере, вы запускаете ваш код (HTML, CSS и JavaScript) внутри исполняемой среды (внутри вкладки браузера). Это как будто фабрика берёт сырьё (некий код) и выдаёт продукцию (веб-страничку).



Код JavaScript выполняется JavaScript-движком браузера, после того как код HTML и CSS был обработан и сформирован в веб-страницу. Это гарантирует, что структура и стиль страницы уже сформированы к моменту запуска JavaScript.

Это хорошо, так как часто использование JavaScript заключается в динамическом изменении HTML и CSS в целях обновления пользовательского интерфейса посредством Document Object Model API (как упоминалось выше). Если бы запуск JavaScript осуществлялся прежде загрузки HTML и CSS, то это привело бы к возникновению ошибок.

**1. Подключение сценариев к html-документу**

Сценарии JavaScript бывают **встроенные**, т.е. их содержимое является частью документа, и **внешние**, хранящиеся в отдельном файле с расширением .js. Сценарии можно внедрить в html-документ следующими способами:

**1.1. В виде гиперссылки**

Для этого нужно разместить код в отдельном файле и включить ссылку на файл в заголовок

<head>

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

</head>

HTML

или тело страницы.

<body>

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

</body>

HTML

Этот способ обычно применяется для сценариев большого размера или сценариев, многократно используемых на разных веб-страницах.

**1.2. В виде обработчика события**

Каждый html-элемент имеет JavaScript-события, которые срабатывают в определенный момент. Нужно добавить необходимое событие в html-элемент как атрибут, а в качестве значения этого атрибута указать требуемую функцию. Функция, вызываемая в ответ на срабатывание события, является **обработчиком события**. В результате срабатывания события исполнится связанный с ним код. Этот способ применяется в основном для коротких сценариев, например, можно установить смену цвета фона при нажатии на кнопку:

<script>

var colorArray = ["#5A9C6E", "#A8BF5A", "#FAC46E", "#FAD5BB", "#F2FEFF"]; // создаем массив с цветами фона

var i = 0;

function changeColor(){

document.body.style.background = colorArray[i];

i++;

if( i > colorArray.length - 1){

i = 0;

}

}

</script>

<button onclick="changeColor();">Change background</button>

HTML

See the Pen [qaOjYR](http://codepen.io/html5book/pen/qaOjYR/) by Elena ([@html5book](http://codepen.io/html5book)) on [CodePen](http://codepen.io/).

**1.3. Внутрь элемента <script>**

Элемент <script> может вставляться в любое место документа. Внутри тега располагается код, который выполняется сразу после прочтения браузером, или содержит описание функции, которая выполняется в момент ее вызова. Описание функции можно располагать в любом месте, главное, чтобы к моменту ее вызова код функции уже был загружен.

Обычно код JavaScript размещается в заголовке документа (элемент <head>) или после открывающего тега <body>. Если скрипт используется после загрузки страницы, например, код счетчика, то его лучше разместить в конце документа:

<footer>

<script>

document.write("Введите свое имя");

</script>

</footer>

</body>

HTML

[**Комментарии**](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/What_is_JavaScript#%D0%BA%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8)

Так же, как и в HTML и CSS, возможно писать комментарии в вашем JavaScript коде, что будет проигнорировано браузером, и существует только для того, чтобы давать подсказки вашим друзьям-разработчикам о том, как работает код (и лично вам, если вы вернётесь к коду спустя 6 месяцев и не сможете вспомнить, что вы делали). Комментарии очень полезны, и вы должны часто их использовать, особенно для больших приложений. Вот два типа комментариев:

* Однострочный комментарий пишется после двойного слеша (//), например:

// Я комментарий

Copy to Clipboard

* Комментарий на нескольких строках пишется между строками /\* и \*/, например:
* /\*
* Я тоже
* комментарий

\*/

Copy to Clipboard

Так, например, мы можем описать наш последний демо-пример JavaScript подобными комментариями:

// Функция: создаёт новый параграф и добавляет его вниз тела HTML.

function createParagraph() {

var para = document.createElement('p');

para.textContent = 'You clicked the button!';

document.body.appendChild(para);

}

/\*

1. Получаем ссылки на все кнопки на странице в виде массива.

2. Перебераем все кнопки и добавляем к ним отслеживатель события нажатия.

При нажатии любой кнопки, будет выполняться функция createParagraph().

\*/

var buttons = document.querySelectorAll('button');

for (var i = 0; i < buttons.length ; i++) {

buttons[i].addEventListener('click', createParagraph);

}

**Браузерный движок** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [*layout engine*](https://en.wikipedia.org/wiki/layout_engine)) — представляет собой [программу](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), преобразующую содержимое веб-страниц (файлы [HTML](https://ru.wikipedia.org/wiki/HTML), [XML](https://ru.wikipedia.org/wiki/Extensible_Markup_Language), цифровые изображения и т. д.) и информацию о форматировании (в форматах [CSS](https://ru.wikipedia.org/wiki/Cascading_Style_Sheets), [XSL](https://ru.wikipedia.org/wiki/Extensible_Stylesheet_Language) и т. д.) в интерактивное изображение форматированного содержимого на экране. Браузерный [движок](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) обычно используется в [веб-браузерах](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) (отсюда название), [почтовых клиентах](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) и других программах, нуждающихся в отображении и редактировании содержимого веб-страниц.

Термин «браузерный движок» получил распространение после того, как движки стали «отделимыми» от браузера. В число наиболее распространённых движков входят нижеследующие

* [Blink](https://ru.wikipedia.org/wiki/Blink_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)) — движок браузера [Chromium](https://ru.wikipedia.org/wiki/Chromium" \o "Chromium), браузера [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome" \o "Google Chrome) с 28 версии, [Microsoft Edge](https://ru.wikipedia.org/wiki/Microsoft_Edge" \o "Microsoft Edge) с 79 версии, [Opera](https://ru.wikipedia.org/wiki/Opera" \o "Opera) c 15 версии и [Vivaldi](https://ru.wikipedia.org/wiki/Vivaldi" \o "Vivaldi). Он является ответвлением [WebKit](https://ru.wikipedia.org/wiki/WebKit" \o ").
* [Gecko](https://ru.wikipedia.org/wiki/Gecko) — [открытый](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%81%D0%BE%D1%84%D1%82) движок проекта [Mozilla](https://ru.wikipedia.org/wiki/Mozilla" \o "EdgeHTML); используется в большом числе программ, основанных на коде Mozilla (браузере [Firefox](https://ru.wikipedia.org/wiki/Mozilla_Firefox" \o "Mozilla Firefox), почтовом клиенте [Thunderbird](https://ru.wikipedia.org/wiki/Mozilla_Thunderbird" \o "Mozilla Thunderbird), наборе программ [SeaMonkey](https://ru.wikipedia.org/wiki/SeaMonkey" \o "Движок)).
* [WebKit](https://ru.wikipedia.org/wiki/WebKit) — движок для браузера [Apple Safari](https://ru.wikipedia.org/wiki/Apple_Safari" \o "Apple Safari), включенного в операционную систему [Mac OS X](https://ru.wikipedia.org/wiki/Apple_Mac_OS_X" \o "Apple Mac OS X), и браузера [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome" \o "Google Chrome) (до 2013 года). Встроен в библиотеку [Qt](https://ru.wikipedia.org/wiki/Qt" \o "Qt) (Начиная с Qt 5.6 признан устаревшим).
* [KHTML](https://ru.wikipedia.org/wiki/KHTML) — разработан в рамках проекта [KDE](https://ru.wikipedia.org/wiki/KDE), используется в браузере [Konqueror](https://ru.wikipedia.org/wiki/Konqueror" \o "Konqueror) и послужил основой для [WebKit](https://ru.wikipedia.org/wiki/WebKit" \o "WebKit).