**Метод document.write**

Метод document.write – один из наиболее древних методов добавления текста к документу.

У него есть существенные ограничения, поэтому он используется редко, но по своей сути он совершенно уникален и иногда, хоть и редко, может быть полезен.

**[Как работает document.write](http://learn.javascript.ru/document-write" \l "kak-rabotaet-document-write)**

Метод document.write(str) работает только пока HTML-страница находится в процессе загрузки. Он дописывает текст в текущее место HTML ещё до того, как браузер построит из него DOM.

HTML-документ ниже будет содержать 1 2 3.

<body>

1

<script>

document.write(2);

</script>

3

</body>

**Нет никаких ограничений на содержимое document.write**.

Строка просто пишется в HTML-документ без проверки структуры тегов, как будто она всегда там была.

Например:

<script>

document.write('<style> td { color: #F40 } </style>');

</script>

<table>

<tr>

<script>

document.write('<td>')

</script>

Текст внутри TD.

<script>

document.write('</td>')

</script>

</tr>

</table>

Также существует метод document.writeln(str) – не менее древний, который добавляет после str символ перевода строки "\n".

**[Только до конца загрузки](http://learn.javascript.ru/document-write" \l "tolko-do-kontsa-zagruzki)**

Во время загрузки браузер читает документ и тут же строит из него DOM, по мере получения информации достраивая новые и новые узлы, и тут же отображая их. Этот процесс идет непрерывным потоком. Вы наверняка видели это, когда заходили на сайты в качестве посетителя – браузер зачастую отображает неполный документ, добавляя его новыми узлами по мере их получения.

**Методы document.write и document.writeln пишут напрямую в текст документа, до того как браузер построит из него DOM, поэтому они могут записать в документ все, что угодно, любые стили и незакрытые теги.**

Браузер учтёт их при построении DOM, точно так же, как учитывает очередную порцию HTML-текста.

Технически, вызвать document.write можно в любое время, однако, когда HTML загрузился, и браузер полностью построил DOM, документ становится *«закрытым»*. Попытка дописать что-то в закрытый документ открывает его заново. При этом все текущее содержимое удаляется.

Текущая страница, скорее всего, уже загрузилась, поэтому если вы нажмёте на эту кнопку – её содержимое удалится:

Из-за этой особенности document.write для загруженных документов не используют.

**XHTML и document.write**

В некоторых современных браузерах при получении страницы с заголовком Content-Type: text/xml или Content-Type: text/xhtml+xml включается «XML-режим» чтения документа. Метод document.write при этом не работает.

Это лишь одна из причин, по которой XML-режим обычно не используют.

**[Преимущества перед innerHTML](http://learn.javascript.ru/document-write" \l "preimuschestva-pered-innerhtml)**

Метод document.write – динозавр, он существовал десятки ~~миллионов~~ лет назад. С тех пор, как появился и стал стандартным метод innerHTML, нужда в нём возникает редко, но некоторые преимущества всё же есть.

* Метод document.write работает быстрее, фактически это самый быстрый способ добавить на страницу текст, сгенерированный скриптом.

Это естественно, ведь он не модифицирует существующий DOM, а пишет в текст страницы до его генерации.

* Метод document.write вставляет любой текст на страницу «как есть», в то время как innerHTML может вписать лишь валидный HTML (при попытке подсунуть невалидный – браузер скорректирует его).

Эти преимущества являются скорее средством оптимизации, которое нужно использовать именно там, где подобная оптимизация нужна или уместна.

Однако, document.write по своей природе уникален: он добавляет текст «в текущее место документа», без всяких хитроумных DOM. Поэтому он бывает просто-напросто удобен, из-за чего его нередко используют не по назначению.

**[Антипример: реклама](http://learn.javascript.ru/document-write" \l "antiprimer-reklama)**

Например, document.write используют для вставки рекламных скриптов и различных счетчиков, когда URL скрипта необходимо генерировать динамически, добавляя в него параметры из JavaScript, например:

<script>

// в url указано текущее разрешение экрана посетителя

var url = 'http://ads.com/buyme?screen=' + screen.width + "x" + screen.height;

// загрузить такой скрипт прямо сейчас

document.write('<script src="' + url + '"></scr' + 'ipt>');

</script>

**На заметку:**

Закрывающий тег </script> в строке разделён, чтобы браузер не увидел </script> и не посчитал его концом скрипта.

Также используют запись:

document.write('<script src="' + url + '"><\/script>');

Здесь <\/script> вместо </script>: обратный слеш \ обычно используется для вставки спецсимволов типа \n, а если такого спецсимвола нет, в данном случае \/ не является спецсимволом, то будет проигнорирован. Так что получается такой альтернативный способ безопасно вставить строку </script>.

Сервер, получив запрос с такими параметрами, обрабатывает его и, учитывая переданную информацию, генерирует текст скрипта, в котором обычно есть какой-то другой document.write, рисующий на этом месте баннер.

**Проблема здесь в том, что загрузка такого скрипта блокирует отрисовку всей страницы.**

То есть, дело даже не в самом document.write, а в том, что в страницу вставляется сторонний скрипт, а браузер устроен так, что пока он его не загрузит и не выполнит – он не будет дальше строить DOM и показывать документ.

Представим на минуту, что сервер ads.com, с которого грузится скрипт, работает медленно или вообще завис – зависнет и наша страница.

Что делать?

В современных браузерах у скриптов есть атрибуты async и defer, которые разрешают браузеру продолжать обработку страницы, но применить их здесь нельзя, так как рекламный скрипт захочет вызвать document.writeименно на этом месте, и браузер не должен двигаться вперёд по документу.

Альтернатива – использовать другие техники вставки рекламы и счётчиков. Примеры вы можете увидеть в коде Google Analytics, Яндекс.Метрики и других.

Если это невозможно – применяют всякие хитрые оптимизации, например заменяют метод document.write своей функцией, и она уже разбирается со скриптами и баннерами.

**[Итого](http://learn.javascript.ru/document-write" \l "itogo)**

Метод document.write (или writeln) пишет текст прямо в HTML, как будто он там всегда был.

* Этот метод редко используется, так как работает только из скриптов, выполняемых в процессе загрузки страницы.

Запуск после загрузки приведёт к очистке документа.

* Метод document.write очень быстр.

В отличие от установки innerHTML и DOM-методов, он не изменяет существующий документ, а работает на стадии текста, до того как DOM-структура сформирована.

* Иногда document.write используют для добавления скриптов с динамическим URL.

Рекомендуется избегать этого, так как браузер остановится на месте добавления скрипта и будет ждать его загрузки. Если скрипт будет тормозить, то и страница – тоже.

Поэтому желательно подключать внешние скрипты, используя вставку скрипта через DOM или async/defer. Современные системы рекламы и статистики так и делают.