

Ajax: jQuery

- Что такое AJAX
- Сокращенные методы:
 - \$.get,
 - \$.post,
 - .load(),
 - \$.getScript,
 - \$.getJSON
- \$.ajax()

AJAX

Asynchronous Javascript and XML — «асинхронный JavaScript и XML»

- Действия с интерфейсом преобразуются в операции с элементами [DOM](#) (*Document Object Model*), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш.
- Каскадные таблицы стилей, или CSS (*Cascading Style Sheets*), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных.
- Три из этих четырёх технологий — [CSS](#), DOM и [JavaScript](#) — составляют [DHTML](#) (*Dynamic HTML*). По мнению некоторых специалистов средства DHTML, появившиеся в [1997 году](#), подавали большие надежды, но так и не оправдали их.
- В качестве формата передачи данных могут использоваться фрагменты простого текста, [HTML](#)-кода, [JSON](#) или [XML](#).

Преимущества технологии

Экономия трафика

- Использование AJAX позволяет значительно [сократить трафик](#) при работе с веб-приложением благодаря тому, что вместо загрузки всей страницы достаточно загрузить только изменившуюся часть, или вообще только получить/передать набор данных в формате [JSON](#) или [XML](#), а затем изменить содержимое страницы с помощью JavaScript.

Уменьшение нагрузки на сервер

- При правильной реализации, AJAX позволяет снизить нагрузку на сервер в несколько раз. В частности, все страницы сайта чаще всего генерируются по одному шаблону, включая неизменные элементы («шапка», «навигационная панель», «подвал» и т. д.) для генерации которых требуются обращения к разным файлам, время на обработку скриптов (а иногда и запросы к БД) — всё это можно опустить, если заменить полную загрузку страницы генерацией и передачей лишь содержательной части.

Ускорение реакции интерфейса

- Поскольку загрузка изменившейся части значительно быстрее, то пользователь видит результат своих действий быстрее и без мерцания страницы (возникающего при полной перезагрузке).

Почти безграничные возможности для интерактивной обработки

- Например, при вводе поискового запроса в [Google](#) выводится подсказка с возможными вариантами запроса. AJAX удобен для программирования [чатов](#), [административных панелей](#) и других инструментов, которые выводят меняющиеся со временем данные.

Недостатки технологии

Отсутствие интеграции со стандартными инструментами браузера

- Динамически создаваемые страницы не регистрируются браузером в истории посещения страниц, поэтому не работает кнопка «Назад», предоставляющая пользователям возможность вернуться к просмотренным ранее страницам, но существуют скрипты, которые могут решить эту проблему. Другой недостаток изменения содержимого страницы при постоянном [URL](#) заключается в невозможности сохранения закладки на желаемый материал. Проблему можно успешно решить с помощью `History.pushState`

Динамически загружаемое содержимое не доступно поисковикам

- Поисковые машины не могут выполнять [JavaScript](#), поэтому разработчики должны позаботиться об альтернативных способах доступа к содержимому сайта

Усложнение проекта

- Перераспределяется логика обработки данных — происходит выделение и частичный перенос на сторону клиента процессов первичного форматирования данных. Это усложняет контроль целостности форматов и типов. Конечный эффект технологии может быть нивелирован необоснованным ростом затрат на кодирование и управление проектом, а также риском снижения доступности сервиса для конечных пользователей.

Низкая скорость при грубом программировании

- Казалось бы, AJAX предназначен именно для повышения скорости. Но, когда AJAX-запросов на одной странице много и, например, по каждому щелчку подгружается список, AJAX-страница становится даже медленнее традиционной.

Риск [фабрикации запросов другими сайтами](#)

- Результат работы AJAX-запроса может являться [JavaScript](#)-кодом (в частности, [JSON](#)). [XMLHttpRequest](#) действует только [в пределах одного домена](#), а вот тег `<script>` — нет

Сокращенные методы

\$.get

```
jQuery.get(  
    url – Адрес запроса  
    [, data ] – Передаваемые данные  
    [, success(data, textStatus, jqXHR) ] – функция обратного вызова  
    [, dataType ] – тип данных, если не указан явно - определяется  
библиотекой. (xml, json, script, or html)  
)
```

\$.post

```
jQuery.post (
  url – Адрес запроса
  [, data ] – Передаваемые данные
  [, success(data, textStatus, jqXHR) ] – функция обратного вызова
  [, dataType ] – тип данных, если не указан явно - определяется
библиотекой. (xml, json, script, or html)
)
```

.load

```
.load(  
    url – Адрес запроса  
    [, data ] – Передаваемые данные  
    [, complete(responseText, textStatus, XMLHttpRequest) ] –  
    функция обратного вызова  
)
```


\$.getJSON

jQuery.getJSON(

url – Адрес запроса

[, data] – Передаваемые данные

[, success(data, textStatus, jqXHR)] – функция обратного

ВЫЗОВА

)

\$.getScript

```
jQuery.getScript(  
    url – Адрес запроса  
    [, success(script, textStatus, jqXHR) ] – функция обратного  
    ВЫЗОВА  
    )
```

\$.ajax()

\$.ajax настройки

`url` – адрес

`success` – функция при успешном запросе

`type` – http метод

`error` – функция при получении ошибки в запросе

`complete` – функция при завершении запроса

`async` – будет ли асинхронным запрос

`context` – указывает на элемент, на который будет ссылаться `this`

`data` – передаваемые данные

`dataType` – (xml, json, script, html, text)

`global` – сработают ли глобальные события

`timeout` – Указывает тайм-аут (в миллисекундах) запроса