

Яндекс

Яндекс

AJAX



Как браузер получает код web-страницы?

Что мы знаем про браузер на данный момент

При переходе по ссылке браузер умеет сделать http запрос, получить ответ и разобрать его - например, отобразить веб-страницу, или картинку

Вот было бы здорово, если бы можно было сделать запрос на какой-то еще url, только не перезагружая страницу, а оставаясь на текущей, получить новые данные и/или готовый html, для добавления в ту или иную часть страницы.

AJAX (Asynchronous Javascript and XML)
— *подход* к построению веб-интерфейсов, заключающийся в «*фоновом*» обмене данными браузера с веб-сервером

Пригодилось бы для

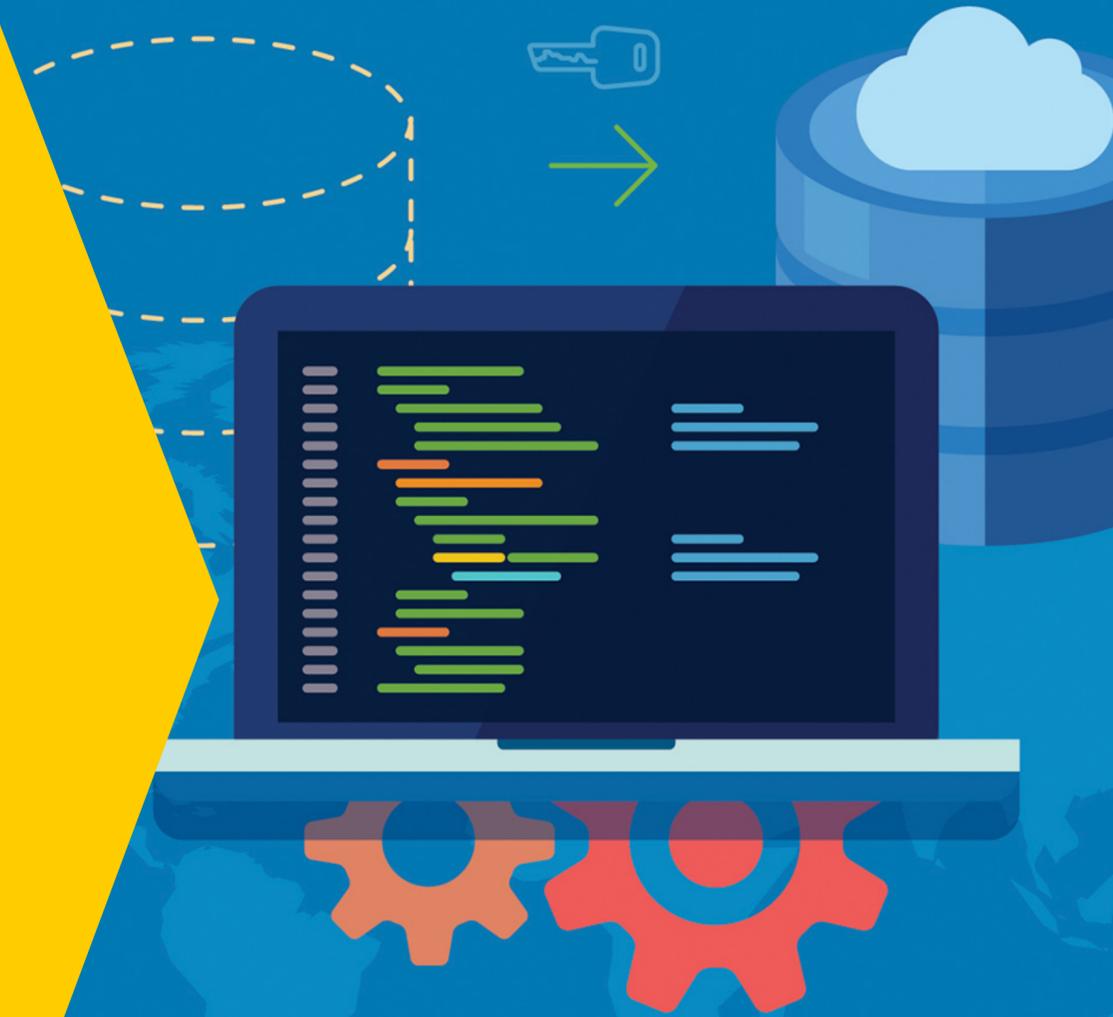
Бесконечного скроллинга соцсеточек/турбо

«Живого» обновления содержимого страницы по мере пользовательского ввода (сайджест, и т.п.)

Обновления результатов матча/переписки в чате по мере новых данных

Обработки форм без перезагрузки страницы

XMLHttpRequest



XMLHttpRequest

Самым «ванильным» и везде работающим способом сделать AJAX запрос является объект XMLHttpRequest

Это встроенный объект, который всегда будет в наличии в браузерном окружении, даже если у вас домофонный браузер



Синхронный запрос

Синхронный запрос

```
// 1. Создаём новый объект XMLHttpRequest
var xhr = new XMLHttpRequest();

// 2. Конфигурируем его: GET-запрос на URL 'data.json'
xhr.open('GET', 'data.json', false);

// 3. Отсылаем запрос
xhr.send();

// 4. Обрабатываем ответ
alert( xhr.responseText );
```

XMLHttpRequest

`open (method, URL, async)` – задаёт параметры запроса

`send ()` – отправляет запрос

`responseText` – текст ответа

Минусы синхронного запроса

«Вешает» страницу аналогично модалкам `alert` и компании, причем с учетом того, что интернет может быть нестабильным, способен наглухо повесить вкладку (а не до нажатия «ОК» как в `alert`)

В принципе помечен как `deprecated`



Асинхронный запрос

Асинхронный запрос

```
// 1. Создаём новый объект XMLHttpRequest
var xhr = new XMLHttpRequest();

// 2. Конфигурируем его: GET-запрос на URL 'data.json'
xhr.open('GET', 'data.json', true);

// 3. Отсылаем запрос
xhr.send();

// 4. Обрабатываем ответ
xhr.onload = function() {
    alert( xhr.responseText );
};
```

XMLHttpRequest

`open(method, URL, async)` – задаёт параметры запроса

`send()` – отправляет запрос

`responseText` – текст ответа

`status` – http статус ответа

`statusText` – текстовое описание статуса

`onload` – запрос выполнен успешно



Передаем «данные»

Передаем данные

Мы можем передавать данные на сервер разными способами:

- 1. GET-параметры в URL**
- 2. тело запроса в различных форматах (POST запросы)**
- 3. HTTP заголовки**



Форма обратной связи
(передаем данные с помощью
POST)

XMLHttpRequest

`open (method, URL, async)` – задаёт параметры запроса

`send ([body])` – отправляет запрос

`status` – http статус ответа

`statusText` – текстовое описание статуса

`responseText` – текст ответа

`onload` – запрос выполнен успешно

`onerror` – произошла ошибка выполнения запроса

Работа с удаленным бэкендом





АЈАХ запросы можно свободно отправлять только на тот сервер, с которого загружена страница.



CORS (Cross-origin resource sharing)

CORS

Позволяет получить доступ Аяксом к ресурсам другого домена.

Правила безопасности браузеров таковы, что требуется «согласие» сервера на то, чтобы можно было запросить его содержимое аяксом. Согласие выдается в виде HTTP заголовок.

Браузер при наличии этого заголовок позволяет обработать запрос.

С точки зрения JavaScript кода ничего не меняется.

Настройки доступа нужны на сервере.

Сторонние API





Итак, мы знаем, что существуют HTTP API и умеем шаблонизировать данные

API

У многих сервисов в интернете есть HTTP API — набор запросов и их параметров, позволяющих работать с сервисом в обход привычного человеческого интерфейса.

Vk, Facebook, YouTube, сервисы финансов, погоды...

Там уже настроены правила CORS



Сервис погоды у себя на
сайте

Что дальше?
fetch, WebSocket





Спасибо

Ткаченко Александр

Разработчик интерфейсов



asany4@yandex-team.ru



[@asanych](https://t.me/asanych)