1. **Приведите примеры GET-вызовов с параметрами (2-3 возможных адреса URL, можно реальных, можно из головы)?** - <https://www.lamoda.ru/c/6659/home_accs-tovarydlyaspalnikomplektyevro/?price=990%2C3000> – выбирает постельное белье, комплектации евро, с фильтром по цене от 900 до 3000 руб.

<https://www.ozon.ru/highlight/programmirovanie-264362/?category=16500&isdiscount=t> – выбирает книги для программистов со скидкой

1. **Расскажите своими словами, чем отличаются GET и POST-запросы?** – Get передает данные в URL (пара-значение), POST передает данные в теле запроса. Все что содержит важную и конфиденциальную (логин, пароль,форма отправки заказа и тд.) инфо отправляем через POST.
2. **Какой вид запроса лучше использовать для получения отфильтрованного списка товаров?** – GET.
3. **А какой - для формы регистрации?** – POST.
4. **Посмотрите на запрос и разберитесь, что он делает и что означают и на что влияют параметры api\_key, q, limit, lang? Как сделать так, чтобы запрос возвращал не 25, а 5 картинок?**

Документация к API находится здесь: <https://developers.giphy.com/explorer>

<https://api.giphy.com/v1/gifs/search?api_key=SET_KEY_HERE&q=cats&limit=25&offset=0&rating=g&lang=en>

Это GET – запрос в заголовке, в котором указан ключ и доп атрибуты в виде фильтра по «кошкам» и выводит первые 25 картинок. api\_key – ключ, который дает доступ к api данного сайта, q – фильтр, limit- кол-во выводимой инфо, lang – язык интерфейса. В limit изменить 25 на 5 - <https://api.giphy.com/v1/gifs/search?api_key=SET_KEY_HERE&q=cats&limit=5&offset=0&rating=g&lang=en>

1. **Как посмотреть заголовки запроса к странице или API?** - F12 Dev Tools – вкладка Network – выбираем запрос – вкладка Headers – блок Response headers.
2. **Самостоятельно разберитесь, что такое CORS. Приведите пример их включения в запросе fetch.** - Cross-Origin Resource Sharing ([CORS](https://developer.mozilla.org/ru/docs/Glossary/CORS)) — механизм, использующий дополнительные [HTTP](https://developer.mozilla.org/ru/docs/Glossary/HTTP)-заголовки, чтобы дать возможность [агенту пользователя](https://developer.mozilla.org/ru/docs/Glossary/User_agent) получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент.

Если мы сделаем запрос fetch на другой веб-сайт, он, вероятно, завершится неудачей.

Например, давайте попробуем запросить http://example.com:

try {

await fetch('http://example.com');

} catch(err) {

alert(err); // Failed to fetch

}

Вызов fetch не удался, как и ожидалось.

Ключевым понятием здесь является источник (origin) – комбинация домен/порт/протокол.

Запросы на другой источник – отправленные на другой домен (или даже поддомен), или протокол, или порт – требуют специальных заголовков от удалённой стороны.

Эта политика называется «CORS»: Cross-Origin Resource Sharing («совместное использование ресурсов между разными источниками»).

Допустим нам нужно разрешить работу JavaScript на сторонних сайтах (например, 127.0.0.1:8000) что бы получать доступ к нашим ответам API. Для этого нам нужно включить CORS в заголовок ответа от сервера. Это делается на стороне сервера:

app.get('/public', function(req, res) {

res.set('Access-Control-Allow-Origin', '\*')

res.send(...)

})

Здесь мы устанавливаем заголовку Access-Control-Allow-Origin значение \*, что означает: что любому хосту разрешен доступ к этому URL и ответу в браузере.

**8.** **Что можно писать в параметре заголовков Content-Type ? –**

application/x-www-form-urlencoded,  multipart/form-data или text/plain

**9. Давайте отправим информацию о собачке в API по адресу localhost/pets/add**

JSON с информацией о собаке:

{

"breed": "Beagle",

"size": "large",

"color": "orange",

"age": 6

}

Как будет выглядеть fetch в таком случае? –

<script>

let dog = {

"breed":"Beagle",

"size":"large",

"color":"orange",

"age": 6

}

fetch("https://localhost/pets/add",

{

method:'POST',

body: JSON.stringify(dog),

headers:{

'Content-Type':'application/json; charset=utf-8'

},

})

.then(response => response.json())

.then(dog => {

console.log(dog);})

.catch(error => console.log(error));

</script>

**10. Изучите статью** [**https://learn.javascript.ru/promise-basics**](https://learn.javascript.ru/promise-basics) **и самостоятельно разберитесь, что такое промисы. Напишите в 1-2 предложения, что это такое и приведите пример, где вы уже сталкивались с промисами? -**

Промис(Обещание) — это объект, который содержит будущее значение асинхронной операции. Например, если вы запрашиваете некоторые данные с сервера, промис обещает нам получить эти данные, которые мы сможем использовать в будущем.

new Promise(/\* Выполняемая функция \*/ function (resolve, reject) { … } );