1)<https://www.douglas.de/de/c/haare/14?q=%3Arelevance%3AHaartyp%3ATrockenes%2BHaar> – выбирает средства для сухих волос

<https://www.ebay.com/b/Canon-Digital-Cameras/31388/bn_740> - выбирает камеры бренда canon

<https://www.rituals.com/de-de/koerper/koerperpflege/koerpercremes-oele> - выбирает масла и крема для тела из раздела уход за телом

2) GET — метод для чтения данных с сайта. Например, для доступа к указанной странице. Он говорит серверу, что клиент хочет прочитать указанный документ. На практике этот метод используется чаще всего, например, в интернет-магазинах на странице каталога. Фильтры, которые выбирает пользователь, передаются через метод GET. POST — метод для отправки данных на сайт. Чаще всего с помощью метода POST передаются формы.

3)GET

4)POST

5) 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> .

Api\_key – ключ доступа к api сайта. q -поиск картинок по заголовку или названию. Limit - количество картинок, которые нам необходимо вывести. Lang – язык. Для того, чтобы вывести не 25, а 5 картинок, нужно установить лимит =5.

6)Правой кнопкой мыши на странице в браузере заходим в Исследовать или при помощи F12. Затем переходим во вкладку сеть или Networks. Здесь мы можем увидеть Заголовки, они же Headers.

7) [CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS) ) — это механизм, основанный на [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP) -заголовке, который позволяет серверу указывать любые [источники](https://developer.mozilla.org/en-US/docs/Glossary/Origin) (домен, схему или порт), отличные от его собственных, из которых браузер должен разрешать загрузку ресурсов.

Скрипт на https: //foo.invalid/ хочет получить некоторые данные с https: //bar.invalid/. (Ни учетные данные, ни доступ к заголовку ответа не важны.)

var url = "https://bar.invalid/api?key=730d67a37d7f3d802e96396d00280768773813fbe726d116944d814422fc1a45&data=about:unicorn";

fetch(url).then(success, failure)

При этом будет использоваться протокол CORS, хотя он полностью прозрачен для разработчика из foo.invalid. Как часть протокола CORS, пользовательский агент будет включать заголовок **Origin** в запрос:

Origin: https://foo.invalid

Получив ответ от bar.invalid, пользовательский агент проверит заголовок ответа «***Access-Control-Allow-Origin***». Если его значение равно *https: // foo.invalid* или *\**, пользовательский агент вызовет успешный обратный (**success**) вызов. Если оно имеет какое-либо другое значение или отсутствует, пользовательский агент вызовет **failure**.

8)application/x-www-form-urlencoded, multipart/form-data,  text/plain, application/json

9)  fetch("localhost/pets/add",

    {

        method:'POST',

        body: new FormData(form),

        headers:{

            "Content-Type":"application/x-www-form-urlencoded;charset=UTF-8"

        },

    })

    .then(response=>response.json())

    .then(user=>{

        console.log(user);

    })

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

}

10) Promise – это специальный объект в JavaScript, который связывает «создающий» и «потребляющий» коды вместе. Например, функция с выводом времени