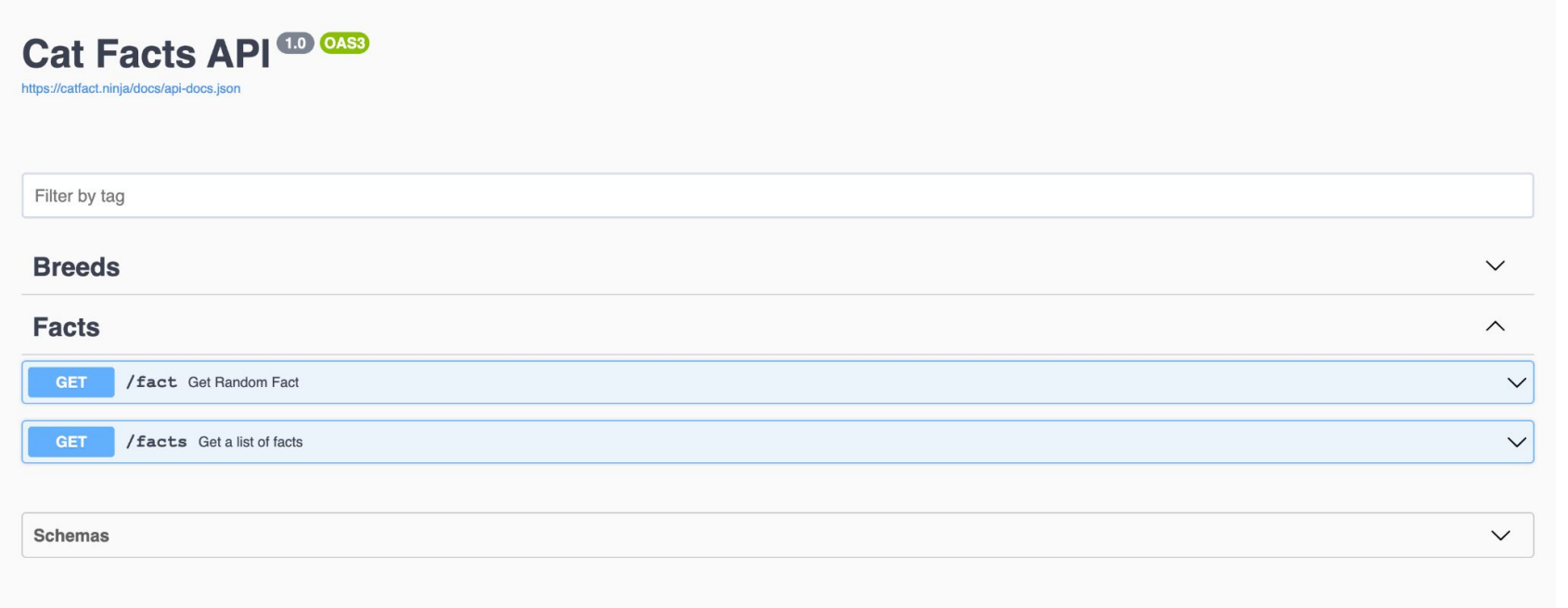
Работа с API

Frontend. Урок 24

Вопросы на повторение

1. Что такое асинхронность?
2. Что такое Promis?
3. Для чего существует async/await?
4. Для чего используется fetch?

Работа с API


Cat Facts API
1.0
OAS3
https://catfact.ninja/docs/api-docs.json
Filter by tag
Breeds
Facts
GET /fact Get Random Fact
GET /facts Get a list of facts
Schemas

Работа с API

Первое, что бросается в глаза это яркое слово GET - это один из методов http запросов
HTTP определяет множество методов запроса, которые указывают, какое желаемое действие выполнится для данного ресурса. Несмотря на то, что их названия могут быть существительными, эти методы запроса иногда называются HTTP глаголами.

- **GET**

Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

- **POST**

POST используется для отправки сущностей к определенному ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

- **PUT**

PUT заменяет все текущие представления ресурса данными запроса.

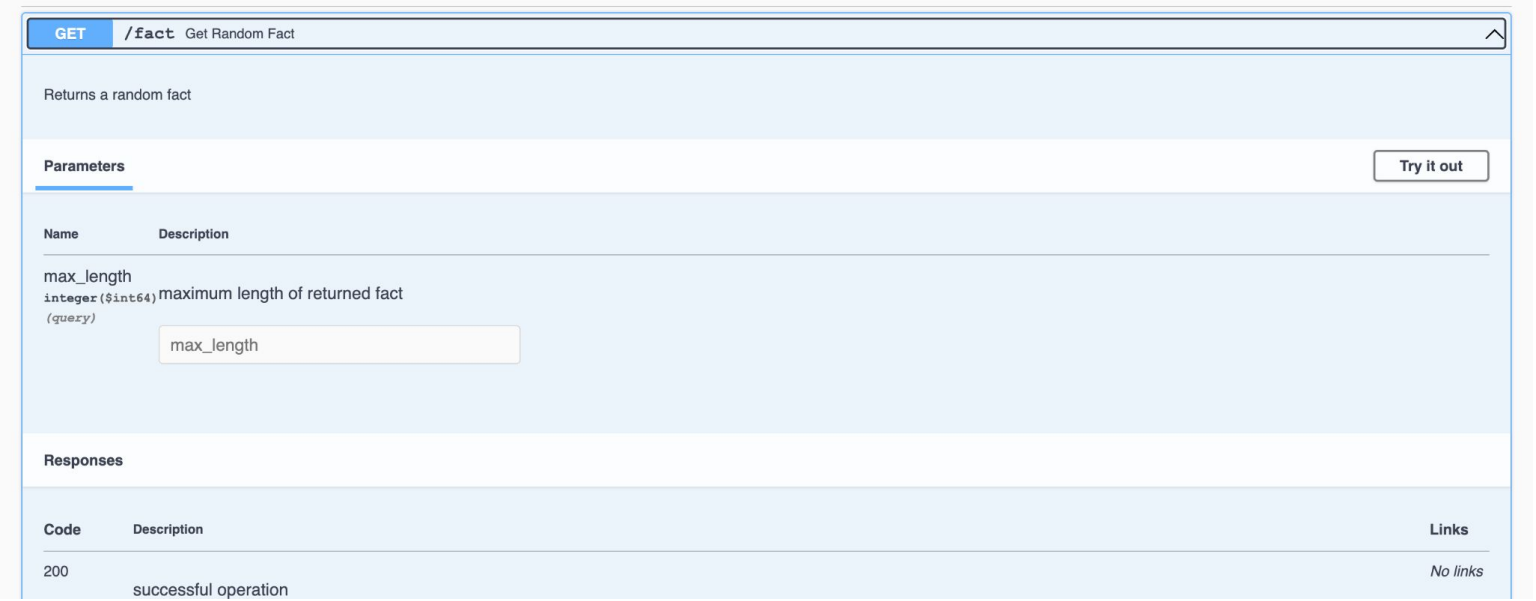
- **DELETE**

DELETE удаляет указанный ресурс.

Get API

В начале у нас описаны параметры, которые мы можем указать для правильного получения данных

Здесь мы можем указать длину факта, например до 200 символов, нужно нажать на кнопку try it out



Get API

Запрос можно увидеть в curl, там же и описан метод запроса.

```
curl -X 'GET' \
  'https://catfact.ninja/fact?max_length=200' \
```

Снизу тело ответа - объект - факт у которого длина до 200 символов
если спуститься ниже - там шаблон тела ответа

```
{
  "fact": "string",
  "length": 0
}
```

В породах у нас объект побольше, параметр также 1 - количество получаемых пород

Функция для получения данных

```
async function getCatBreeds() {
    // Отправляем запрос на сервер для получения данных о породах кошек
    const response = await fetch('https://catfact.ninja/breeds?limit=10');
    // Разбираем ответ сервера в формате JSON
    const data = await response.json();

    // Получаем массив объектов с данными о породах кошек
    const breedsData = data.data;
```

Создаем переменную response (ответ) и в нее помещаем ответ от сервера по структуре, которую мы изучили на прошлом уроке, а именно с помощью await (функция асинхронная async) и fetch. Внутри запроса мы можем сразу же увидеть передаваемый параметр.

В JavaScript, когда вы делаете запрос к серверу с помощью методов, таких как fetch(), сервер может отправить ответ в различных форматах, таких как HTML, JSON, XML и другие.

Формат данных

Сравним результат без json - первый, и второй с ним

В первом случае практически нереально найти запрашиваемые данные, во втором они всегда будут находится в data

Как раз таки в последней строчке на скрине с кодом мы и достаем эти данные

Практика А

Проанализировать API метод get/facts;
Создать в HTML структуру с span, input, куда пользователь будет вводить количество желаемых фактов, button для отправки данных на сервер
Стилизовать этот блок на свое усмотрение.

Практика В

Задача:

1) В зависимости от введенного числа пользователем, отправить запрос на сервер с этим числом. (Тут нужно установить связь в js с input/button)
2) Отразить полученные данные через js в html. (здесь мы на пустое поле не проверяем)
3) Добавить событие на клик по кнопке

Практика С

Задача:
Как можно заметить, если пользователь не ввел число, то мы все равно получаем факты о кошках, поэтому нужно сделать проверку на пустое поле, и в этом случае ничего не выводить.
Дополнительно: сделать простую валидацию - в случае, если пользователь ничего не ввел, мы подсвечиваем поле ввода красным border и сверху выводим ошибку “Enter the number”. Для вывода ошибки нам нужно добавить тег label.

Вопросы на закрепление

1. Как мы отправляем данные на сервер?
2. Какие методы запросов существуют?
3. В каком формате обрабатываем полученные данные?