Вопросы 💎

1. Напишите JSON к вашим ответам на вопрос 2 из прошлой недели (минимум 3 примера):

У меня были судьи Арбитражного суда Москвы, пример:

```
[{
    "fio": "Абрамова Екатерина Анатольевна",
    "unit": 4,
    "collegium": "Судебная коллегия по рассмотрению споров, возникающих из гражданских и иных правоотношений",
    "phone": 495-600-97-11
},{
    "fio": "Ваганова Евгения Александровна",
    "unit": 9,
    "collegium": "Судебная коллегия по рассмотрению споров, возникающих из административных правоотношений",
    "phone": 495-600-98-53
},{
    "fio": "Истомин Сергей Сергеевич",
    "unit": 1,
    "collegium": "Судебная коллегия по рассмотрению споров, возникающих из гражданских и иных правоотношений",
    "phone": 495-600-99-29
}]
```

2. Самостоятельно разберитесь, что за формат данных XML и чем он отличается от JSON? Приведите пример, как один и тот же объект собачки с картинки ниже будет выглядеть в JSON и в XML?

Как обсуждалось выше, JSON — это нотация объектов JavaScript для форматирования данных, тогда как XML — это язык разметки. Ключевое различие между JSON и XML заключается в том, что JSON имеет меньший размер и эффективно передает данные по сравнению с XML. JSON обращается к данным через объекты JSON, тогда как XML требует анализа данных. JSON легко читается, поскольку имеет более организованную структуру кода, а XML трудно интерпретировать из-за его сложной структуры.

JSON хорош для передачи данных, поскольку он не требует обработки, но при этом XML позволяет не только передавать данные, но также обрабатывать и форматировать файлы.

Также XML хранит данные иначе, чем JSON. JSON хранит данные как карту, XML хранит данные как древовидную структуру.

```
// JSON формат
{
    "Breed": "Beagle",
```

```
    "Size": "large",
    "Colour": "orange",
    "Age": 6 + " " + "years"
}
```

```
// XML формат

    <?xml version=«1.0» encoding=«UTF-8»?>
    <Breed>Beagle</Breed>
    <Size>large</Size>
    <Colour>orange</Colour>
    <Age>6 years</Age>
```

3. Что такое сериализация и десериализация (парсинг)? В каких ситуациях они нужны?

Сериализация - процесс преобразования объекта/массива/примитива в json-строку с помощью метода JSON.stringify.

Десериализация (парсинг) преобразование JSON обратно в объект с помощью метода JSON.parse.

Они нужны, например, для того, чтобы отправить данные на сервер, предварительно упаковав (сериализовав) их в JSON строку, или для того, чтобы поместить их в облачное хранилище или в локальное. И наоборот, мы можем распарсить данные, полученные от клиента с сервера в формате JSON обратно в JavaScript объект, чтобы продолжить с ними работу, отправить по сети, или вывести на страницу.

4. Можно ли обработать ответ от сервера одновременно и как текст, и как JSON?

Нет, одновременно нельзя. Можно выбрать только один метод чтения ответа. Если уже получен ответ с response.text(), тогда response.json() не сработает, так как данные уже были обработаны.

```
let text = await response.text(); // тело ответа обработано
let parsed = await response.json(); // ошибка (данные уже были обработаны)
```

5. В чем особенность асинхронных запросов?

«Асинхронный запрос» не блокирует работу страницы, браузер отправит запрос, а далее результат нужно будет получить через обработчики событий. При этом все происходит без определенного порядка. То есть действие (запрос) выполняется в фоне (не в основном потоке), другими словами, таким образом, что оно не мешает пользователю взаимодействовать со страницей.

6. В чем преимущество AJAX-запросов перед старым способом работы с сервером через <form action="имя скрипта на сервере">?

AJAX базируется на технологии обращения к серверу без перезагрузки страницы (XTMLHttpRequest, создание дочерних фреймов или тега script) или использовании DHTML, позволяющего динамически изменять содержимое. Формат передачи данных – XML или JSON. Самое привлекательное в Ajax — это его асинхронный принцип работы. С помощью этой технологии можно осуществлять взаимодействие с сервером

без необходимости перезагрузки страницы. Это позволяет обновлять содержимое страницы частично, в зависимости от действий пользователя.

Основными преимуществами использования AJAX называют:

- снижение трафика (из-за уменьшения объёма передаваемых данных между клиентом и сервером);
- уменьшение нагрузки на сервер (не нужно генерировать всю страницу, а только ту часть, которую нужно обновить);
- увеличение быстродействия и отзывчивости (нет необходимости в полной перезагрузки страницы, достаточно обновить содержимое только отдельных блоков);
- повышение интерактивности (с помощью AJAX можно сразу отображать результаты и сделать ресурс более удобным для пользования).

7. Напишите, как будет выглядеть `fetch` для получения данных вашего пользователя на github? Адрес URL для запроса должен выглядеть так: `'https://api.github.com/users/сюда подставьте свой логин с github'`

```
fetch("https://api.github.com/users/StoneTanya").then(response =>
response.json()).then(user => console.log(user))
```

8. Самостоятельно разберитесь, что такое SPA?

SPA (Single Page Application) – такой вариант архитектуры, появившейся в начале 2010-х, как нечто среднее между сайтом и приложением. Особенность SPA заключается в том, что при открытии страницы браузер загружает сразу весь код приложения. Но показывает только конкретный модуль — часть сайта, которая нужна пользователю. Когда пользователь переходит в другую часть приложения, браузер берёт уже загруженные данные и показывает ему. И, если нужно, динамически подгружает с сервера нужный контент без обновления страницы.

При этом SPA обменивается данными с сервером без перезагрузки страницы, с помощью ajax-запросов. Благодаря этому наполнение страницы может меняться динамически.

Например, есть интернет-магазин, я листаю «товары», но страница при этом не перезагружается, а динамически подтягивает фото, название, описание и цену. Реальные примеры: Веб-версии Gmail, Facebook, Netflix, AirBnB и Pinterest — одностраничные приложения.