Вопросы 💎

Напишите JSON к вашим ответам на вопрос 2 из прошлой недели. Например, если вы ответили *Книга (название, автор, год выпуска)*, то JSON может выглядеть так (минимум 3 примера): Ответ:

Ответ: Товары (категория , стоимость, вес),
Магазин :Сотрудники (ФИО, должность, зарплата) и Клиенты (ФИО, контакты),

```
[{

    «product": платье,
    "price":4000,
    "weight":400
    },{
    "product":джинсы,
    "price":4500,
    "weight":500
    },{
    "product":рубашка,
    "price":3500,
    "weight":300
    }]
[{

    «name": Иванов И.И.,
    «position": продавец,
    «salary": 15000
    },{
    «name": Кузнецова А.А.,
    "position":бухгалтер,
    «salary": 25000
    },{
    «name": Симонов В.В.,
    "position":менеджер,
    "salary":30000
    }]
```

1. 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?Ответ:
   JSON:
   ```
   [{

   «Breed»: Beagle,

   «Size»:large,

   «Color»: orange,

   «Age»: 6 years

   }]
   ```

```
XML:<Breed>Beagle</Breed>

<Size>large<Size>

<Color>orange</Color>

<age>6 years</age>
```

XML eXtensible Markup Language Расширяемый язык разметки	JSON Java Script Object Notation Обозначение объектов Java Script
Текстовые форматы, которые : • удобные для чтения, • имеют иерархическую структуру, • могут быть использованы многими языками программирования, • могут быть получены с помощью XMLHttpRequest.	
Язык	Текстовый формат
Представляет элементы данных	Используется для репрезентации объектов
Нет прямой поддержки массивов	Поддерживает текст и числовые типы данных, массивы и объекты
Использует открывающий и закрывающий теги	Не использует закрывающие теги, но использует для этого скобки (фигурные и квадратные)
Поддерживает пространство имен (namespace)	Не поддерживает пространство имен
Более защищен	Менее защищен
Поддерживает комментарии	Не поддерживает комментарии
Независимый формат данных, который поддерживает разные кодировки	Независимый от языка формат обмена данными, который поддерживает только UTF-8 кодирование

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

   Ответ:

Сериализация — это преобразование объекта или дерева объектов в какой-либо формат с тем, чтобы потом эти объекты можно было восстановить из этого формата. Используется,

например, для сохранения состояния программы (то есть, некоторых её объектов) между запусками. Или для передачи данных между различными экземплярами программы (или различными программами), например, по сети.

Главная идея состоит в том, что сериализованный формат — набор байт или строка, которую можно легко сохранить на диск или передать другому процессу или, например, по сети, в отличие от самого объекта. А значит, задача сохранения объекта/группы объектов при этом сводится к простой задаче сохранения набора байт или строки.

JSON — один из популярных форматов для сериализации, он текстовый, легковесный и легко читается человеком.Для обратной операции - **десериализации** или парсинг.

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

Ответ:нет

4. В чем особенность асинхронных запросов?ОТвет : большинстве языков программирования мы привыкли к тому, что операции выполняются по порядку (последовательно). Первая строка должна быть выполнена, прежде чем мы перейдем к следующей строке.

Но в JS у нас есть операции, которые выполняются в фоновом/активном режиме, и поэтому наше веб-приложение не зависает каждый раз, когда оно ожидает пользовательское событие.

Тем не менее, иногда все должно проходить по порядку, иначе это вызовет хаос и неожиданные результаты. По этой причине мы можем использовать асинхронные вызовы, чтобы все работало как нужно. Примером может быть проверка учетных данных пользователя перед переходом к следующей операции.

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

Основные преимущества использования AJAX:

- **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)

6. 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://github.com/AlsuAl/itgirls/tree/main')
```

```
.then(response => response.json())
        .then(data => console.log(data))
.catch(err => console.log(err));
```

7. Самостоятельно разберитесь, что такое SPA?

Главной идеи SPA — динамическая подгрузка контента на текущую страницу без загрузки всей страницы с сервера. Вы как бы получаете десктопное приложение
Все HTML, JS,CSS могут быть извлечены из сервера с первой же загрузки приложения, вместо того чтобы получать все файлы снова и снова при каждой загрузки, как это обычно без SPA.
Другой важный пункт, это то, что главная страница или контент никогда не перезагружается, но при этом, у вас разные URL адреса и история браузера при помощи пользовательского хешированного местоположения или API HTML5 history.