1. **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):**

|  |  |
| --- | --- |
| [{  "title": "Хуавей",  "color": "синий",  "memory": 256  }, {  "title": "Яблоко",  "color": "снийи океан",  "memory": 512  }, {  "title": "Хонор",  " color ": "черный",  "memory": 128  }] | [{  "name": "Иванов Иван",  "dateBirdth": 02.01.2014 ,  "grade": 5  }, {  "name": "Иванова Катя",  "dateBirdth": 15.06.2014 ,  "grade": 5  },{  "name": "Смирнов Олег",  "dateBirdth": 18.10.2013 ,  "grade": 5  }] |

**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?**

XML – это язык разметки, который используется для добавления дополнительной информации в обычный текст. JSON – это эффективный способ представления структурированных данных в удобочитаемом формате

**JSON**

{

"breed": "Beagle",

"size": "large",

"color": "orange",

"age": "6 years"

}

**XML**

**<breed>Beagle</bread>**

**<size>large</size>**

**<color>orange</color>**

**<age>6 years</age>**

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

В веб-разработке JSON очень часто применяется в качестве формата для передачи информации от веб-сервера клиенту (веб-браузеру).

*Сериализация -* сервер по запросу клиента сначала формирует некоторый набор данных в удобном формате, который затем можно было бы очень просто упаковать в строку JSON. Завершается работа на сервере отправкой JSON данных в качестве результата клиенту. Для сериализации объекта javascript в json применяется функция JSON.stringify().

*Десериализация (парсинг) -* клиент получает в качестве ответа от сервера строку JSON и распаковывает (десериализует, парсит) её, т.е. превращает в JavaScript объект. После этого на клиенте выполняются дальнейшие с ними действия, например, выводятся на страницу. Для обратной операции - десериализации или парсинга json-объекта в javascript применяется метод JSON.parse().

Также если нам нужно отправить какие-то данные на сервер, то мы должны сначала упаковать их в строку JSON-формата (сериализовать) и только потом отправлять.

Помимо работы с сервером сериализация и десериализация часто нужна для хранения и получения данных в веб-хранилищах (localStorage и sessionStorage).

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

Мы можем выбрать только один метод чтения ответа. Если мы уже получили ответ с response.text(), тогда response.json() не сработает, так как данные уже были обработаны.

**5. В чем особенность асинхронных запросов?**

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.

В синхронном коде выполнение функций происходит в том же месте, где они были вызваны, и в тот момент, когда происходит вызов. В асинхронном коде всё по-другому. Вызов функции не означает, что она отработает прямо здесь и сейчас. Более того, мы не знаем, когда она отработает.

Правильно написанные асинхронные программы (в тех ситуациях, где это нужно) значительно эффективнее синхронных. Иногда это настолько критично, что синхронная версия просто не справится с задачей.

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

AJAX означает асинхронный JavaScript и позволяет асинхронно обновлять веб-страницы, обмениваясь данными с веб-сервером во время работы приложения. Короче говоря, это означает, что вы можете обновлять части веб-страницы без перезагрузки всей страницы (их URL-адрес остается прежним). Это значит, что наша страничка будет работать гораздо быстрее - ведь перегружаем только ее кусочек, а не всю целиком

**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'**

document.addEventListener("DOMContentLoaded",

function(event) {

fetch("https://api.github.com/users/Elena-great")

.then(response => response.json())

.then(user => console.log(user))

.catch(error => console.log(error));

})

**8. Самостоятельно разберитесь, что такое SPA?**

“SPA – это web-приложение, размещенное на одной странице, которая для обеспечения работы загружает все javascript-файлы (модули, виджиты, контролы и т.д.) , а также файлы CSS вместе с загрузкой самой страницы.”

Основные преимущества Single Page Application:

1. Производительность. Так как SPA не обновляет всю страницу, а только нужную часть, это существенно повышает скорость работы.
2. Высокая скорость разработки. Готовые библиотеки и фреймворки дают мощные инструменты для разработки веб приложений. Над проектом могут параллельно работать back-end и front-end разработчики. Благодаря четкому разделение они не будут мешать друг другу.
3. Мобильные приложения. SPA позволяет легко разработать мобильное приложение на основе готового кода.

При всех своих достоинствах, Single Page Application имеет некоторые недостатки, которые сдерживают рост популярности:

1. Плохая SEO оптимизация. SPA работает на основе javascript и загружает информацию по запросу со стороны клиента. Поисковые системы с трудом могут имитировать данное поведение. Потому большинство страниц попросту недоступны для сканирования поисковыми ботами, поэтому [продвижение сайта в 2020](https://merehead.com/ru/blog/seo-trends-in-2020/) может быть усложнено.
2. Неактивный javascript. Некоторые пользователи отключают javascript в своих браузерах, а без него ваше приложение не будет работать.
3. Низкий уровень безопасности.