1. **Поясните назначение заголовка Content-Type.**

Заголовок-сущность **Content-Type** используется для того, чтобы определить [MIME тип](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/MIME_type) ресурса.

В ответах сервера заголовок Content-Type сообщает клиенту, какой будет тип передаваемого контента. В некоторых случаях браузеры пытаются сами определить MIME тип передаваемого контента, но их реакция может быть неадекватной. Чтобы предотвратить такие ситуации, Вы можете установить в заголовке [X-Content-Type-Options](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/X-Content-Type-Options) значение nosniff.

1. **Поясните назначение заголовка Accept.**

HTTP заголовок запроса Accept указывает, какие типы контента, выраженные как [MIME типы](https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/MIME_types), клиент может понять. Используя [согласование контента](https://developer.mozilla.org/ru/docs/Web/HTTP/Content_negotiation), сервер затем выбирает одно из предложений, использует его и информирует клиента о своем выборе с помощью заголовка ответа [Content-Type](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Content-Type" \o "Документация об этом ещё не написана; пожалуйста, поспособствуйте её написанию!). Браузеры задают адекватные значения для этого заголовка в зависимости от контекста, в котором выполняется запрос: при получении таблицы стилей CSS для запроса задается другое значение, чем при получении изображения, видео или скрипта.

1. **Для чего используется значение Multipart/form-data заголовка Content-Type.**

Тип содержимого multipart/form-data — это составной тип содержимого, чаще всего использующийся для отправки [HTML-форм](https://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BEfjfjfh%D1%80%D0%BC%D0%B0_(HTML)&action=edit&redlink=1) с бинарными (не-[ASCII](https://ru.wikipedia.org/wiki/ASCII)) данными методом POST протокола [HTTP](https://ru.wikipedia.org/wiki/HTTP). Указывается в поле заголовка Content-Type (тип содержимого) и следует правилам для составных [MIME](https://ru.wikipedia.org/wiki/MIME)-данных в соответствии с [RFC 2045](https://tools.ietf.org/html/rfc2045). Для форм, не имеющих больших бинарных (не-ASCII) данных, может использоваться тип содержимого application/x-www-form-urlencoded.

1. **Как с помощью тега form, обеспечить значение Multipart/form-data заголовка Content-Type.**

В стандартных HTTP-формах для метода POST доступны [три кодировки](https://html.spec.whatwg.org/multipage/forms.html#submit-body), задаваемые через атрибут enctyp.

У форм есть две основные кодировки: application/x-www-form-urlencoded – по умолчанию и multipart/form-data – для POST запросов, если явно указана в enctype. Вторая кодировка обычно используется для больших данных и только для тела запроса.

1. **Какое значение заголовка Content-Type отправляется тегом form в запросе по умолчанию.**

application/x-www-form-urlencoded – по умолчанию

1. **Где и в каком формате передаются параметры в GET-запросе?**

В Query Params в виде ключ-значение.

Не существует конкретной максимальной величины GET-запроса. Один сервер может принимать максимум 8 Кб, а другой — 16 Кб. Средний размер запроса колеблется в пределах 512-1024 Кб.

Фактически в одном таком запросе должно быть не больше 5 параметров, иначе каждый из них будет сложно контролировать со стороны сервера и браузера. Если нужно передать большое количество информации, рекомендуется использовать метод POST.

* Могут кэшироваться
* Остаются в истории браузера
* Могут быть/стать "закладкой"
* Не должны использоваться для передачи паролей и всего такого
* Имеют ограничение по длине (URL и в некоторых браузерах свои заморочки)

1. **Где и в каком формате передаются параметры в POST-запросе?**

Значения отправляются в тело запроса в формате, указанном типом содержимого.

* Никогда не кэшируются
* Не остаются в истории браузера
* Не могут быть/стать "закладкой"
* Не имеют таких ограничений по длине (обычно в браузерах и на web серверах есть ограничение по умолчанию)

1. **Поясните понятие JSON?**

JSON (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве [языка программирования JavaScript](http://javascript.crockford.com/). JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными.

JSON основан на двух структурах данных:

* Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
* Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме. Логично предположить, что формат данных, независимый от языка программирования, должен быть основан на этих структурах.

1. **Поясните понятие XML?**

XML - это расширяемый язык разметки (Extensible Markup Language), разработанный специально для размещения информации в World Wide Web, наряду с HTML, который давно стал стандартным языком создания Web-страниц. В отличие от HTML, вместо использования ограниченного набора определённых элементов вы имеете возможность создавать ваши собственные элементы и присваивать им любые имена по вашему выбору.

XML решает ряд проблем, которые не решает HTML, например:

* Представление документов любого (не только текстового) типа, например, музыки, математических уравнений и т.д.
* Сортировка, фильтрация и поиск информации.
* Представление информации в структурированном (иерархическом) виде.

В зависимости от уровня соответствия стандартам документ может быть "верно сформированным" ("well-formed"), либо "валидным" ("valid"). Вот несколько основных правил создания верно сформированного документа:

* Каждый элемент XML должен содержать начальный и конечный тэг (либо пустой тэг типа <TAG />, который может нести информацию посредством своих атрибутов).
* Любой вложенный элемент должен быть полностью определён внутри элемента, в состав которого он входит.
* Документ должен иметь только один элемент верхнего уровня.
* Имена элементов чувствительны к регистру.

Есть три основных способа сообщить браузеру, как отображать каждый из созданных вами XML-элементов:

* Каскадная таблица стилей (Cascading Style Sheet - CSS) или расширяемая таблица в формате языка стилевых таблиц (Extensible Stylesheet Language - XSL).
* Связывание данных. Этот метод требует создания HTML-страницы, связывания с ней XML-документа и установления взаимодействий HTML-элементов с элементами XML. В дальнейшем HTML-элементы автоматически отображают информацию из связанных с ними XML-элементов.
* Написание сценария. Этот метод требует создания HTML-страницы, связывания с ней XML-документа и получение доступа к XML-элементам с помощью кода сценария JavaScript или VBScript.

Индивидуальный пользователь, компания или комитет по стандартам может определить необходимый набор элементов XML и структуру документа, которые будут применяться для особого класса документов. Подобный набор элементов и описание структуры документа называют XML-приложением или XML-словарём.

XML-приложение обычно определяется созданием описателя типа документа (DTD), который является допустимым компонентом XML-документа. DTD устанавливает и определяет имена элементов, которые могут быть использованы в документе, порядок, в котором элементы могут появляться, и доступные к применению атрибуты элементов. DTD обычно включается в XML-документ и ограничивает круг элементов и структур, которые будут использоваться. Примечание: приложение XML Schema позволяет разрабатывать подробные схемы для ваших XML-документов с использованием стандартного синтаксиса XML и является альтернативой DTD.