### 10. Поясните аббревиатуру MIME, какой организаций поддерживается допустимые MIME, в каких компонентах запросов и ответов используется MIME.

MIME (Multipurpose Internet Mail Extensions) — это стандарт, который определяет, как передавать различные типы данных через интернет. Первоначально он был разработан для электронной почты, но в настоящее время используется и в других протоколах, включая HTTP.

MIME-тип — это стандартный способ обозначения формата содержимого, передаваемого по сети. Он указывает, как интерпретировать данные, которые передаются в HTTP-запросаMIME-типы состоят из двух частей, разделенных слэшем (/):

1. Тип: Основная категория данных (например, application, text, image, audio и т. д.).

2. Подтип: Более конкретный формат данных (например, json, html, png, plain и т. д.).

- Организация: Поддержка MIME-типов осуществляется IETF (Internet Engineering Task Force), которая разрабатывает и публикует различные интернет-стандарты.

- Компоненты запросов и ответов:

- В заголовках HTTP: MIME-типы указываются в заголовках Content-Type и Accept.

- Content-Type определяет тип содержимого, который отправляется клиенту (например, text/html, application/json).

- Accept указывает, какие типы данных клиент может обрабатывать (например, браузер может указать, что он может принимать text/html или application/json).

### 11. Перечислите теги HTML, интерпретация которых приводит к HTTP-запросам.

Некоторые теги HTML, которые вызывают HTTP-запросы, включают:

1. <img> — загружает изображение по указанному src.

2. <link> — загружает стили по указанному href.

3. <script> — загружает JavaScript-файлы по указанному src.

4. <iframe> — загружает содержимое другого документа по указанному src.

5. <audio> и <video> — загружают медиафайлы по указанному src.

6. <source> — используется внутри <audio> и <video> для указания медиафайлов.

### 12. Перечислите способы выполнения HTTP-запросов из JS-сценария.

Существует несколько способов выполнения HTTP-запросов из JavaScript:

1. XMLHttpRequest:

- Это старый способ выполнения запросов, который используется для асинхронного обмена данными с сервером.

const xhr = new XMLHttpRequest();

xhr.open('GET', 'https://example.com/data');

xhr.onload = () => { console.log(xhr.responseText); };

xhr.send();

2. Fetch API:

- Это современный способ выполнения запросов, который возвращает промис.

fetch('https://example.com/data')

.then(response => response.json())

.then(data => console.log(data));

3. Axios:

- Это сторонняя библиотека, которая упрощает выполнение HTTP-запросов и поддерживает промисы.

axios.get('https://example.com/data')

.then(response => console.log(response.data));

4. jQuery:

- Используется в проектах, где подключена библиотека jQuery.

$.get('https://example.com/data', function(data) {

console.log(data);

});

### 13. Поясните понятие «параметризованный модуль».

Параметризованный модуль — это модуль, который принимает параметры (аргументы) при своей инициализации или подключении. Это позволяет создавать более гибкие и переиспользуемые компоненты, которые могут изменять свое поведение или конфигурацию в зависимости от переданных параметров.

#### Примеры:

1. В JavaScript:

- Модуль может экспортировать функцию, принимающую параметры для настройки.

// myModule.js

module.exports = (config) => {

return {

start: () => { console.log("Started with config:", config); }

};

};

// Использование

const myModule = require('./myModule')({ option1: true });

myModule.start();

2. В других языках программирования:

- В языках, таких как Python или Java, параметризованные модули могут быть реализованы с помощью классов или функций, которые принимают аргументы.