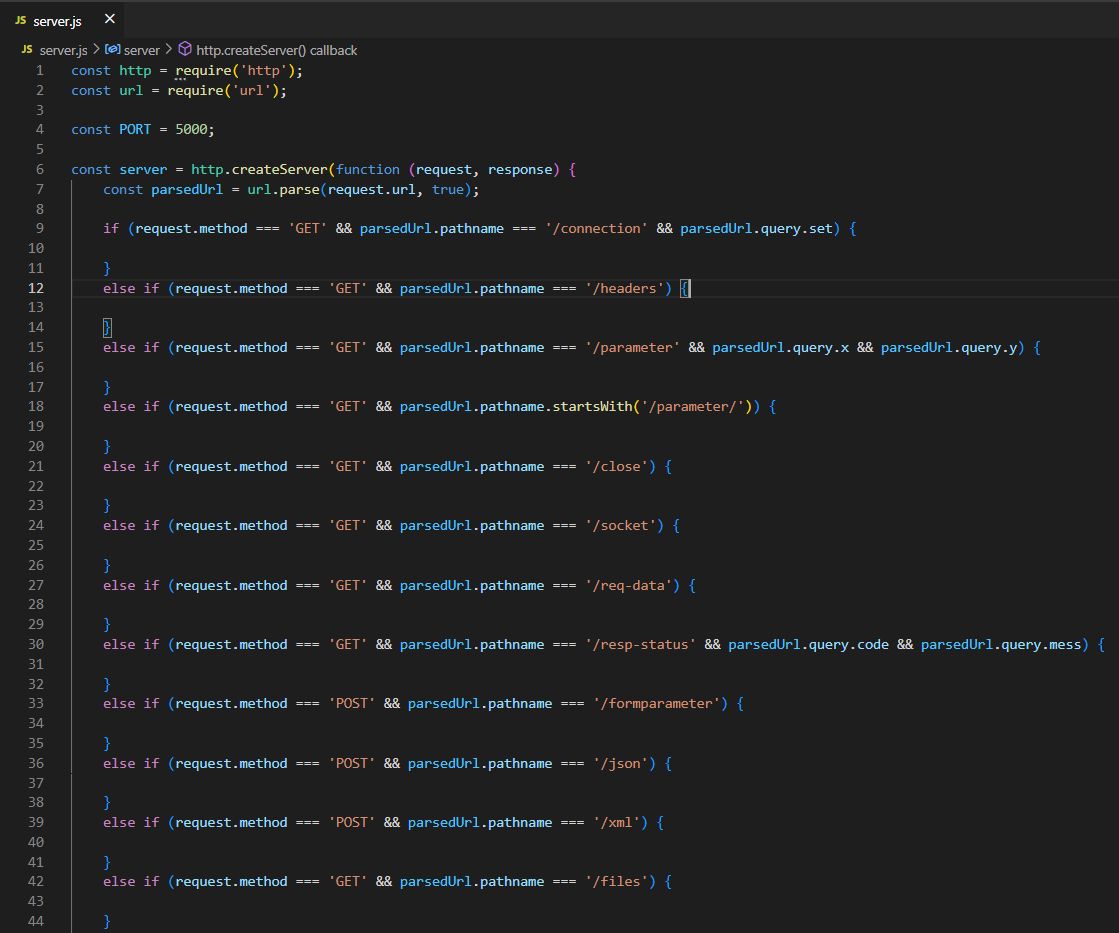
**Задание 00**

1. Разработайте приложение (сервер) **08-00** предназначенное для обработки следующих запросов.

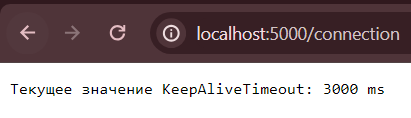
|  |  |  |
| --- | --- | --- |
| HTTP-метод | URI | Задание |
| GET | /connection?set=***set*** | 01 |
| GET | /headers | 02 |
| GET | /parameter?x=***x***&&y=***y*** | 03 |
| GET | /parameter/***x***/***y*** | 04 |
| GET | /close | 05 |
| GET | /socket | 06 |
| GET | /req-data | 07 |
| GET | /resp-status?code=***c***?mess=***m*** | 08 |
| POST | /formparameter | 09 |
| POST | /json | 10 |
| POST | /xml | 11 |
| GET | /files | 12 |
| GET | /files/***filename*** | 13 |
| GET/POST | /upload | 14 |



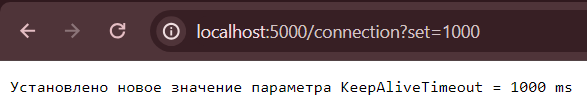


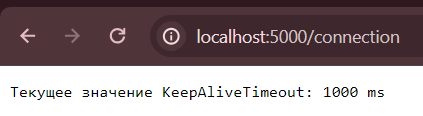
**Задание 01** /connection?set=***set***

1. При GET-запросе **/connection** в окно браузера вывести текущее значение системного параметра **KeepAliveTimeout.**



1. При GET-запросе **/connection/set=*set*** установить новое значение системного параметра **KeepAliveTimeout = *set*** и вывести в окно браузера сообщение установлено новое значение параметра **KepAliveTimeout=*set***.

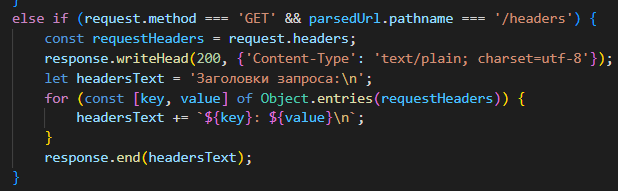


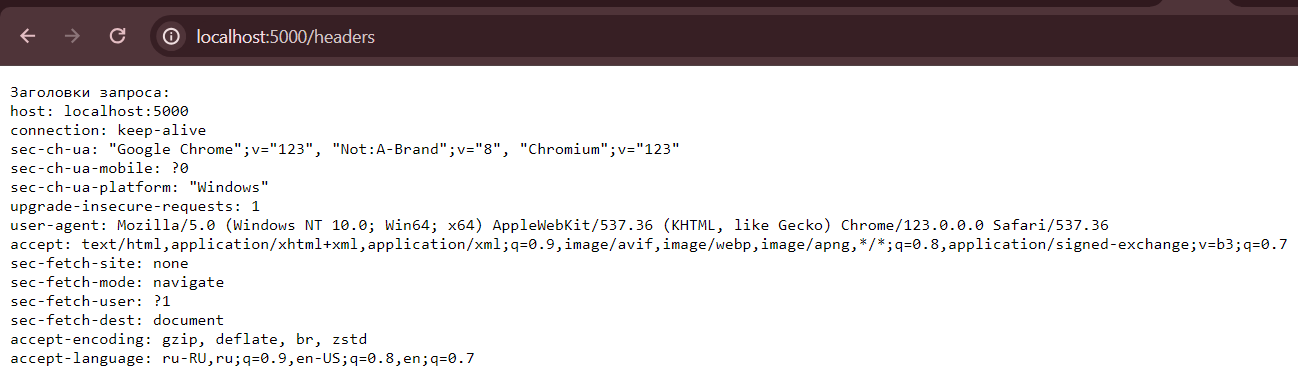


1. Продемонстрируйте влияние системного параметра **KeepAliveTimeout** на работу приложения**.**

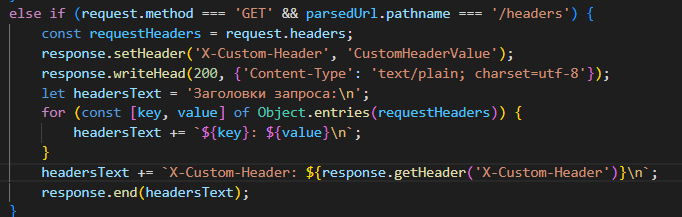
**Задание 02** /headers

1. Отобразите в окне браузера все заголовки запроса и ответа.





1. Сформируйте собственный пользовательский заголовок ответа.



1. Убедитесь, что созданный пользовательский заголовок ответа, доставлен клиенту (с помощью браузера или POSTMAN).



1. Объясните назначение каждого заголовка.

-- **host**: Указывает хост (домен) и порт, к которому был отправлен запрос. В данном случае это localhost:5000. Это обязательный заголовок для HTTP/1.1 и выше.

-- **connection**: Указывает, как будет управляться соединением после завершения запроса. Значение keep-alive означает, что соединение останется открытым для возможных последующих запросов, что позволяет избежать повторного открытия соединений для каждого запроса.

-- **cache-control**: Определяет правила кэширования для браузера или других кэш-систем. В данном случае, max-age=0 означает, что ответ не должен кэшироваться, и браузер всегда должен запрашивать свежую версию ресурса.

-- **sec-ch-ua**: Заголовок, использующий механизм **User-Agent Client Hints**. В нем указаны данные о браузере пользователя и его версии, чтобы сервер мог адаптировать ответ (например, разные версии сайта для разных браузеров). В данном случае это информация о браузере Google Chrome версии 123.

-- **sec-ch-ua-mobile**: Этот заголовок сообщает серверу, использует ли пользователь мобильное устройство. Значение ?0 означает, что устройство не является мобильным.

-- **sec-ch-ua-platform**: Этот заголовок сообщает серверу, на какой платформе работает браузер. В данном случае указано Windows, что означает, что запрос поступил с компьютера на платформе Windows.

-- **upgrade-insecure-requests**: Указывает, что браузер хочет, чтобы все соединения с сервером, которые не используют HTTPS, были автоматически заменены на защищенные (HTTPS).

-- **user-agent**: Содержит информацию о браузере и операционной системе пользователя. В вашем случае это Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36, что указывает на браузер Chrome на платформе Windows 10.

-- **accept**: Указывает, какие типы контента клиент готов принять. В данном случае это HTML, XHTML, XML, изображения (AVIF, WebP, Apng и другие), а также подписанные обмены.

-- **sec-fetch-site**: Указывает контекст, в котором был сделан запрос, например, с сайта или из локального хранилища. В данном случае значение none указывает, что запрос был сделан напрямую, а не с другого сайта (например, через <iframe>).

-- **sec-fetch-mode**: Определяет, какой тип запроса был выполнен. Значение navigate указывает, что запрос связан с переходом по ссылке, то есть пользователь запросил страницу.

-- **sec-fetch-user**: Указывает, был ли запрос выполнен в результате действий пользователя, например, нажатия на ссылку. Значение ?1 говорит, что запрос был вызван действиями пользователя.

-- **sec-fetch-dest**: Указывает, что за ресурс был запрашиваем. В вашем случае это document, что означает, что запрашиваемый ресурс — это документ (например, HTML-страница).

-- **accept-encoding**: Указывает, какие алгоритмы сжатия клиент может принять для ответа. В данном случае это gzip, deflate, br (Brotli), zstd. Это позволяет уменьшить объем передаваемых данных.

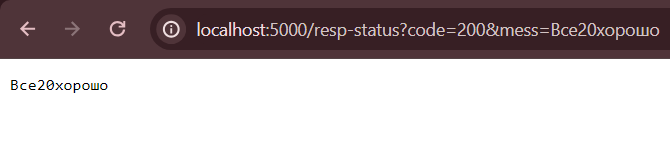
-- **accept-language**: Указывает, на каких языках клиент предпочитает получить ответ. В вашем случае указаны ru-RU (русский) и en-US (английский).

**Задание 03** /parameter?x=***x***&&y=***y***

1. Проанализируйте значения параметров **x** и **y**.
2. Если **x** и **y** имеют числовые значения, то выведите в окно браузера сумму, разность, произведение и частное этих чисел
3. Иначе выведите сообщение об ошибке.

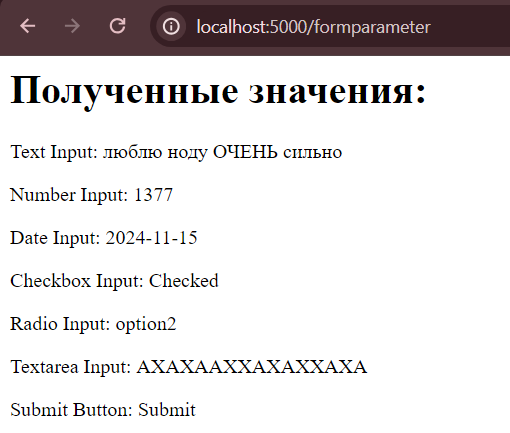
**Задание 08** resp-status?code=***c***?mess=***m***

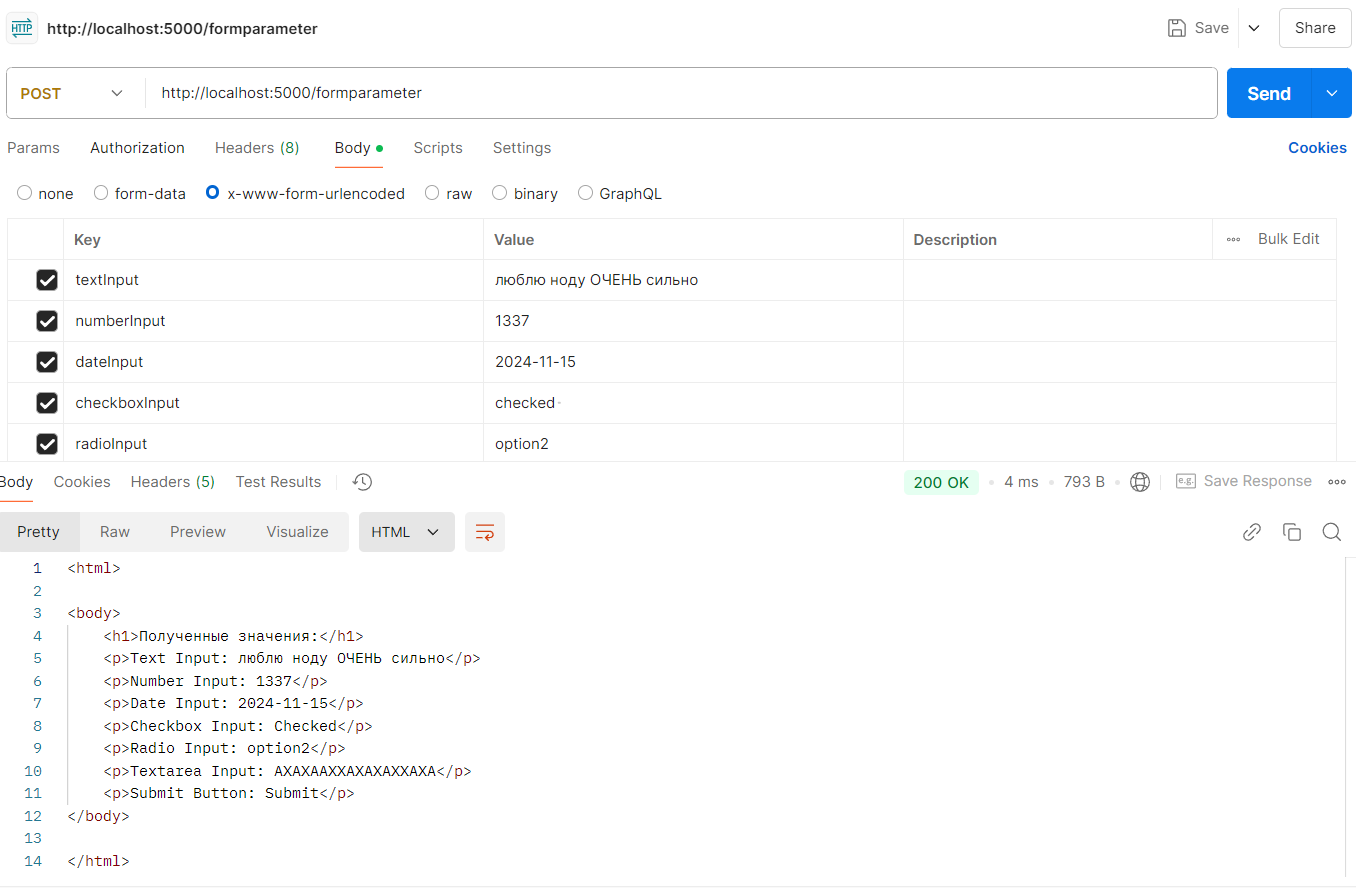
1. При получении этого запроса, сформируйте ответ, имеющий статус, заданный значением ***с*** и пояснение к статусу, заданное значением ***m***.



**Задание 09** /formparameter

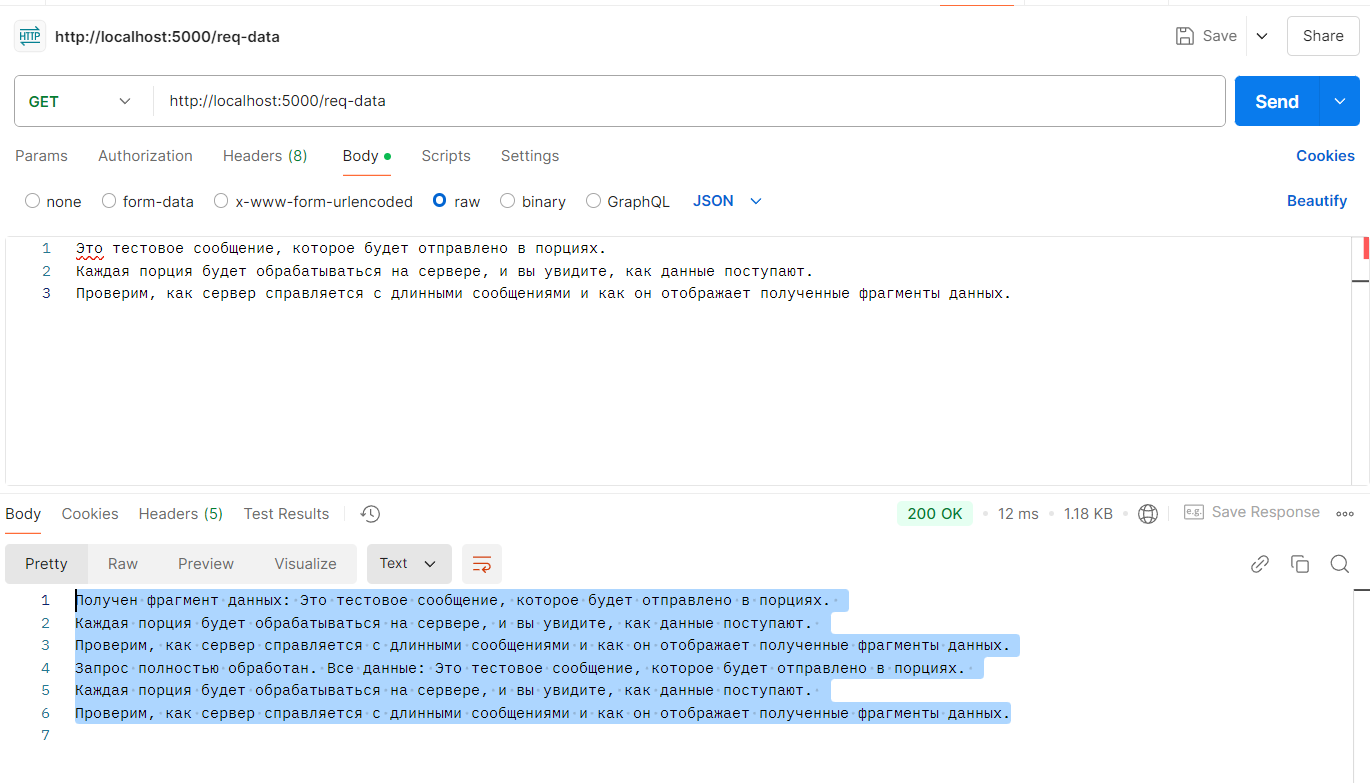
1. Использую HTML-форму включающую теги **input** с **type**: **text**, **number, date, checkbox, radiobutton,** тег **textarea**, а также два тега **input type=submit**, имеющих одно и тоже имя, но разные значения.
2. В окно браузера выведите значения параметров, полученных в запросе.





**Задание 07** /req-data

1. Продемонстрируйте в этом запросе, подобрав достаточно длинное сообщение в POSTMAN, порционную обработку запроса.



**Задание 10** /json

1. Принимаются POST-запросы, содержащие данные в json-формате и оправляет ответы в json-формате.
2. Сообщение в запросе имеет следующую структуру:

{

    "\_\_comment": " Запрос.Лабораторная работа 8/10",

    "x": 1,

    "y": 2,

    "s": "Сообщение",

    "m": ["a", "b", "c", "d"],

    "o":{"surname":"Иванов", "name":"Иван"}

}

**Задание 11. /**xml

1. Принимаются POST-запросы, содержащие данные в xml-формате и оправляет ответы в xml-формате.
2. Сообщение в запросе имеет следующую структуру.

<request id="28">

    <x value="1"/>

    <x value="2"/>

    <m value="a"/>

    <m value="b"/>

    <m value="c"/>

</request>