Лабораторная работа №4

Задача 1.

Установите пакет learnyounode глобально.
Для запуска приложения введите в консоль название пакета и выполните первые две задачи.

```
$ learnyounode
```

Выполните все тринадцать задач.

Задача 2.

Отдача видео контента через Node.js

Создадим следующую структуру проекта, включающий каталог сайта:

```
site
    video -
        - Introducing_Windows_95_Mobile.mp4
    index.html
server.js
```

Файл index.html содержит следующую разметку:

```
<html>
        <head>
                <title>Node-page</title>
                <meta charset="utf-8">
        </head>
        <body>
        <video id="mainPlayer" width="640" height="360" autoplay="autoplay" controls="controls">
                <source src="video/Introducing_Windows_95_Mobile.mp4">
        </video>
        </body>
</html>
```

Видео для тестирования Introducing_Windows_95_Mobile.mp4 скачайте по следующей ссылке:
https://yadi.sk/i/v6-eOCsU0BblDQ

Теперь займёмся наполнением файла server.js.
Сначала нам потребуется подключить модули:

```
let http = require('http'); // подключение модуля для работы с http
let fs = require('fs'); // подключение модуля для работы с файлом
let path = require('path');  //подключение модуля для работы с путями
```

Далее создадим объект с mime-типами:

```
let mimeTypes = {
        '.html': 'text/html',
        '.mp4': 'video/mp4' //mime тип для файлов формата .mp4
};
```

Затем создадим http сервер по аналогии как в лабораторной работе №2:

```
http.createServer((request, response) => {// вызов метода создания http сервера
```

```
let pathname, extname, mimeType;

console.log("Request: " + request.url);
console.log("Request: " + request.headers.range);

if (request.url === '/')
        pathname = 'site/index.html';
else
        pathname = 'site' + request.url;
extname = path.extname(pathname);
mimeType = mimeTypes[extname];

if (extname === '.mp4') {
        fs.readFile(pathname, (err, data) => {
                if (err) {
                        console.log('Could not find or open file for reading\n');
                        response.statusCode = 404;
                        response.end();
                } else {
                        console.log(`The file ${pathname} is read and sent to the client\n`);
                        response.writeHead(200, {
                                'Content-Type':mimeType
                        });
                        response.end(data);
                }
        });
} else {
        fs.readFile(pathname, 'utf8', (err, data) => {
                if (err) {
                        console.log('Could not find or open file for reading\n');
                        response.statusCode = 404;
                        response.end();
                } else {
                        console.log(`The file ${pathname} is read and sent to the client\n`);
                        response.writeHead(200, {
                                'Content-Type':mimeType
                        });
                        response.end(data);
                }
```

```
                });
        }
}).listen(8080, ()=>{
        console.log("HTTP server works in 8080 port!\n");
});
```

Запустите данный код и посмотрите весь ли видео контент получен браузером? Нормально ли управляется видео контент в браузере. Посмотрите в консоль (терминал) сервера, что он выводил нам, когда клиент запрашивал видео? Как вы думаете в чём недостаток такой отдачи видео контента?

Задача 3.

HTTP (код «206 Partial Content») предоставляет механизм частичной отдачи контента клиенту.
Схема взаимодействия:
1) Браузер запрашивает частичный контент у сервера используя http заголовок в запросе «Range».
2) Ответ сервера может быть следующим:

- Если сервер в состоянии отдавать частичный контент, то он формирует ответ с кодом «206 Partial Content». Отсылаемый диапазон байтов указывается в заголовке http ответа «Content-Range»;
- Если указанный в запросе диапазон не поддерживается (например, запрашиваемый диапазон больше реального размера файла), то сервер отдаёт ответ со статусом «416 Requested Range Not Satisfiable». Доступный диапазон указывается в заголовке «Content-Range».

Используемые http заголовки при частичной отдаче контента:
1) «Accept-Ranges»: bytes
Этот заголовок, отправляется сервером в ответ, и сигнализирует что сервер в состоянии вернуть контент частями. Значение заголовка указывает в каких единицах измеряется диапазон, обычно это байты в большинстве ситуаций.
2) «Range»: bytes=(start)-(end)
Данный заголовком браузер запрашивает конкретный диапазон у сервера. Если конечная позиция опущена, сервер возвращает содержимое из указанной начальной позиции в позицию последнего доступного байта. Если начальная позиция опущена, то конечная позиция будет описана как количество байтов, возвращаемых сервером, начиная с последнего доступного байта.
3) «Content-Range»: bytes (start)-(end)/(total)
Это заголовок устанавливает в ответе сервер если код ответа 206. Значения start и end представляют диапазон текущего содержимого. Значение total указывает общее количество доступных байтов.
4) «Content-Range»: */(total)
Это тот же заголовок, но в другом формате и устанавливает в ответе сервера если код ответа 416. Значение total также указывает общее количество доступных байтов содержимого.

Пример запросов и ответов:
1) Запрос первых 1024 байтов:
Браузер запрашивает:
GET /dota2/techies.mp4 HTTP/1.1
Host: localhost:8000
Range: bytes=0-1023

Сервер отвечает:

HTTP/1.1 206 Partial Content
Date: Mon, 15 Sep 2014 22:19:34 GMT
Content-Type: video/mp4
Content-Range: bytes 0-1023/2048
Content-Length: 1024

(Content...)
2) Запрос без указания конечной позиции:
Браузер запрашивает:
GET /dota2/techies.mp4 HTTP/1.1
Host: localhost:8000
Range: bytes=1024-

Сервер отвечает:
HTTP/1.1 206 Partial Content
Date: Mon, 15 Sep 2014 22:19:34 GMT
Content-Type: video/mp4
Content-Range: bytes 1024-2047/2048
Content-Length: 1024

(Content...)

Обратите внимание, что сервер не должен возвращать все оставшиеся байты в одном ответе, особенно если содержимое слишком большое или имеются другие соображения по производительности. Таким образом, следующие два примера также приемлемы в данном случае:
Content-Range: bytes 1024-1535/2048
Content-Length: 512
или, например, так:
Content-Range: bytes 1024-1279/2048
Content-Length: 256

3) Запрос последних 512 байтов:
Браузер запрашивает:
GET /dota2/techies.mp4 HTTP/1.1
Host: localhost:8000
Range: bytes=-512

Сервер отвечает:
HTTP/1.1 206 Partial Content
Date: Mon, 15 Sep 2014 22:19:34 GMT
Content-Type: video/mp4
Content-Range: bytes 1536-2047/2048
Content-Length: 512

(Content...)

4) Запрос с некорректным диапазоном:
Браузер запрашивает:
GET /dota2/techies.mp4 HTTP/1.1
Host: localhost:8000

Range: bytes=1024-4096

Сервер отвечает:
HTTP/1.1 416 Requested Range Not Satisfiable
Date: Mon, 15 Sep 2014 22:19:34 GMT
Content-Range: bytes */2048

Теперь займемся изменением кода http сервера из предыдущей задачи. Первоначально в конец файла server.js добавим функцию которая сможет обработать присланный браузером содержимое заголовка «Range»:

```
function readRangeHeader(range, totalLength) {
	//range – содержимое заголовка «Range»
	// totalLength – размер запрашиваемого файла

	//проверяем задано ли содержимое заголовка «Range»
	if (range == null || range.length == 0)
		return null;

	//с помощью регулярного выражения разбиваем строку из заголовка «Range»
	//на массив из четырех значений
	let array = range.split(/bytes=([0-9]*)-([0-9]*)/);
	//забираем начальную позицию
	let startRange = parseInt(array[1]);
	//забираем конечную позицию
	let endRange = parseInt(array[2]);
	//формируем результирующий объект с двумя свойствами начало и конец диапозона:
	let result = {
		//если не указан начало диапазона, то присваиваем в качестве его значения 0
		start: isNaN(startRange) ? 0 : startRange,
		//если не указан конец диапазона, то присваиваем в качестве его значение конца
		//запрашиваемого файла
		end: isNaN(endRange) ? (totalLength - 1) : endRange
	};
	//Проверяем случай: начало не указано, а конец указан – значит запрашиваются данные с
	//конца файла
	if (isNaN(startRange) && !isNaN(endRange)) {
		result.start = totalLength - endRange;
		result.end = totalLength - 1;
	}

	return result;
}
```

Теперь в файле serever.js полностью изменим код от строки «if (extname === '.mp4') {» и до конца блока «} else if (extname === ".jpg" || extname === ".gif") {» на:

```
if (!fs.existsSync(pathname)) { //Функция existsSync проверяет существование запрашиваемого файла
	console.log('Could not find or open file for reading\n');
	response.statusCode = 404;
	response.end();
	return null;
```

```
}
//создаем пустой объект для будущего хранения заголовков ответа сервера
let responseHeaders = {};
//Функция statSync возвращает описание файла.
let stat = fs.statSync(pathname);
//Разбираем содержимое заголовка Range в запросе от браузера
let rangeRequest = readRangeHeader(request.headers['range'], stat.size); //stat.size – размер файла
//Проверяем, а был ли заголовок Range в запросе от браузера
if (rangeRequest == null) {
        //Заголовка Range не было в запросе от браузера, а значит сразу полностью возвращаем
        //файл
        //Формируем заголовки ответа
        responseHeaders['Content-Type'] = mimeType;
        responseHeaders['Content-Length'] = stat.size;  //Говорим размер файла
        responseHeaders['Accept-Ranges'] = 'bytes'; //Говорим, что можем отдавать диапазонами

        //Читаем и возвращаем файл с кодом 200
        let video = fs.readFileSync(pathname);
        console.log(`The file ${pathname} is read and sent to the client\n`);
        response.writeHead(200, responseHeaders);
        response.end(video);
        return null;
}

let start = rangeRequest.start;
let end = rangeRequest.end;

//Проверяем соответствует ли запрашиваемый браузером диапазон размеру файла
if (start >= stat.size || end >= stat.size) {
        // Запрашиваемый диапазон больше файла возвращаем код 416
        responseHeaders['Content-Range'] = 'bytes */' + stat.size; //Размер файла
        console.log("Return the 416 'Requested Range Not Satisfiable'");
        response.writeHead(416, responseHeaders);
        response.end();
        return null;
}

let maxsize = 10240; /*Максимальный размер частичного контента 10KB (это значение выбрано для
примера, может быть задано другое из соображений производительности)*/
//Проверяем превышает ли запрашиваемый браузером диапазон максимальному размеру
if (end - start >= maxsize){
        //Превышает, поэтому изменяем конечную границу диапазона
        end = start + maxsize - 1;
}
//Формируем заголовки ответа с частичным контентом
responseHeaders['Content-Range'] = 'bytes ' + start + '-' + end + '/' + stat.size; /*Отправляемый диапазон
с байтами */
responseHeaders['Content-Length'] = start == end ? 0 : (end - start + 1);
responseHeaders['Content-Type'] = mimeType;
responseHeaders['Accept-Ranges'] = 'bytes'; //Говорим, что отдаем диапазон с байтами
```

```
responseHeaders['Cache-Control'] = 'no-cache'; //не кешировать результат
//Считываем дескриптор файла, необходим для чтения части файла
const fd = fs.openSync(pathname, 'r'); //'r' – режим работы с файлом только чтение
/*Создаем буфер (специализированный числовой массив) для хранения прочитанных из файлов байтов*/
let buf = Buffer.alloc(responseHeaders['Content-Length']);
/*Читаем часть файла. Вызываем функцию чтения со следующими аргументами: fd – дискриптор файла, buf – буфер куда сохранить почитанные байты, 0 – позиция с которой надо писать в буфер, buf.length – количество байтов для прочтения из файла, start – это позция в файле с которой читать байты */
fs.read(fd, buf, 0, buf.length, start, (err, bytes) => {
        if(err){
                console.log(err);
                response.statusCode = 500; //Отдаём ошибку в ответ клиенту
                response.end();
        } else {
                //Отдаем часть файла и соответствующий статус 206
                response.writeHead(206, responseHeaders);
                response.end(buf);
        }
});
```

Запустите данный код и посмотрите весь ли видео контент получен браузером? Нормально ли управляется видео контент в браузере. Посмотрите в консоль (терминал) сервера, что он выводил нам, когда клиент запрашивал видео?

Задача 4.

Доработайте программу в задаче 3 для отдачи аудио контента. Тестовый mp3 можно скачать по ссылке: https://yadi.sk/d/TEgPte7JPnnFvw

Так же для тестирования добавьте в index.html тег audio:

```
<audio controls>
        <source src="audio/kino-poslednijj-gerojj.mp3" type="audio/mpeg">
</audio>
```