Unit – 2 Node JS

Node JS File System

- The Node.js file system module allows you to work with the file system on your computer.
- We use this module for file operations like creating, reading, deleting, etc.,
- Node.js gives the functionality of file I/O by providing wrappers around the standard POSIX functions.
- All file system related function can be synchronous and asynchronous depending upon user requirements.
- There are multiple ways to work with file.

- **What is Synchronous approach?**
- In **synchronous** approach, suppose you call FunctionA() and then FunctionB(), in this case first FunctionA() will Complete it’e execution then only FunctionB() will execute.
- **What is asynchronous approach?**
- In **asynchronous** approach, suppose you call FunctionA() and then FunctionB(), in this case FunctionA() will start first but FunctionB() will also start.
- FunctionB() will not wait for FunctionA() to complete it’s execution.

Common File operations are

- Read Files
- Write Files
- Append Files
- Open Files
- Close Files
- Delete Files

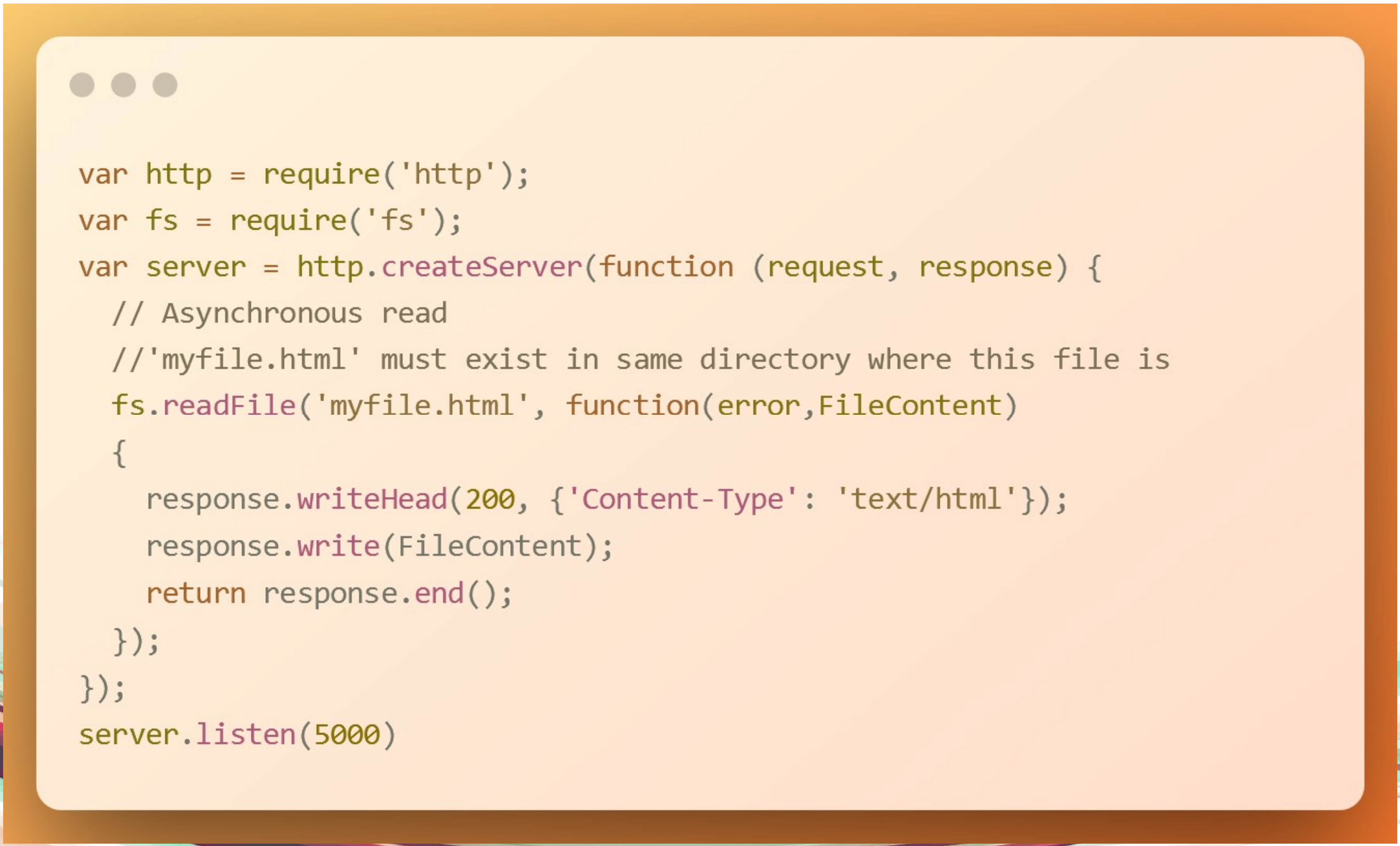
How to use file module

- To use file module we need to include the File System module useing the require() method:
- var fs = require('fs');

How to read data from file?

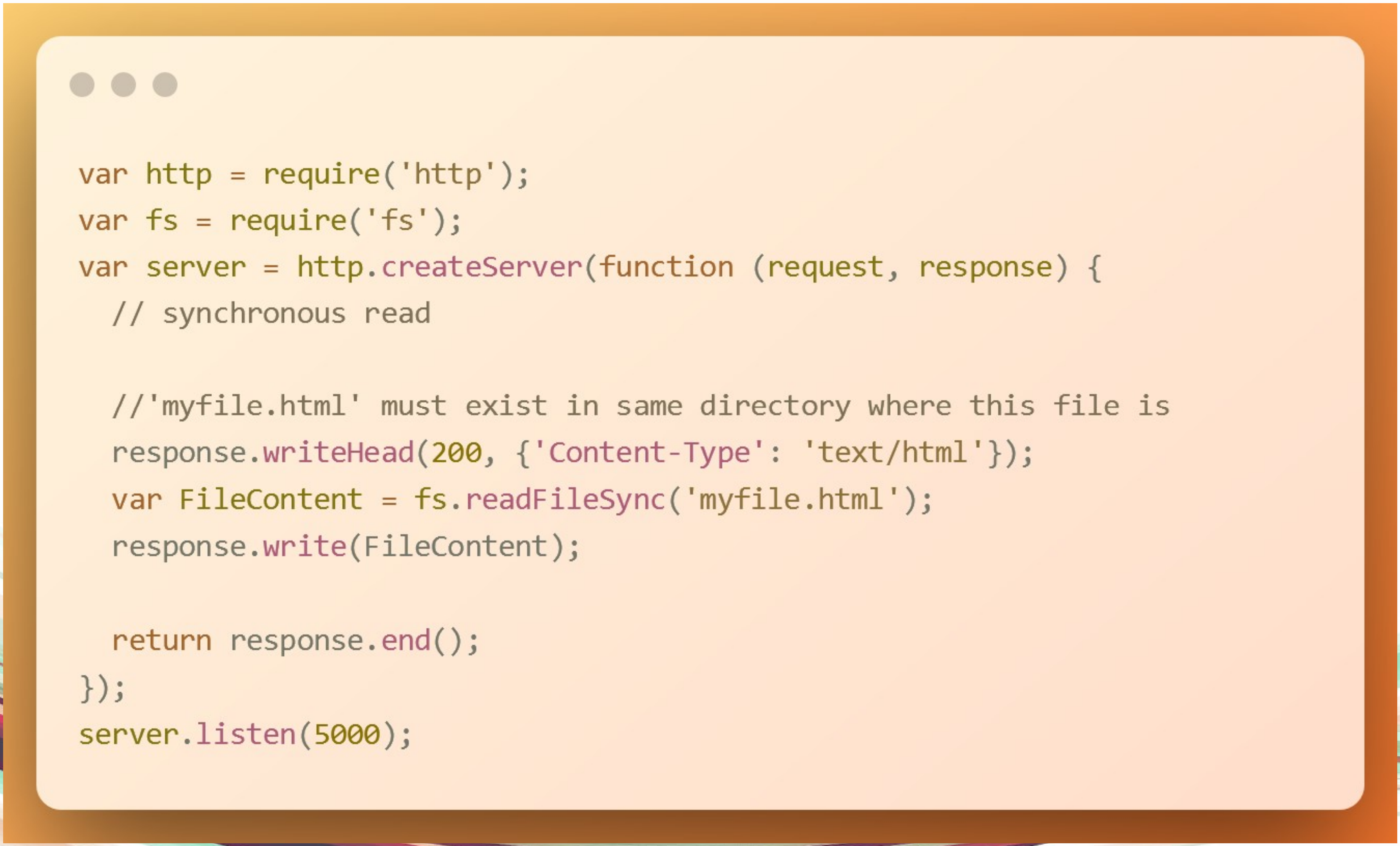
- First create file in project folder. For example myfile.html and write some content in it.
- The fs.readFile() method is used to read files on your computer in this case myfile.html. It works **asynchronously.**
- This method has 2 argument. 1st argument is filename to be read and 2nd argument is anonymous function that will run once when content is fetched from file.
- This function also has 2 argument, 1st argument is error and 2nd argument is data read from file.
- Let us see an example.

Example of reading data from file asynchronously?



```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (request, response) {
  // Asynchronous read
  //'myfile.html' must exist in same directory where this file is
  fs.readFile('myfile.html', function(error,FileContent)
  {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(FileContent);
    return response.end();
  });
});
server.listen(5000)
```

Example of reading data from file synchronously?

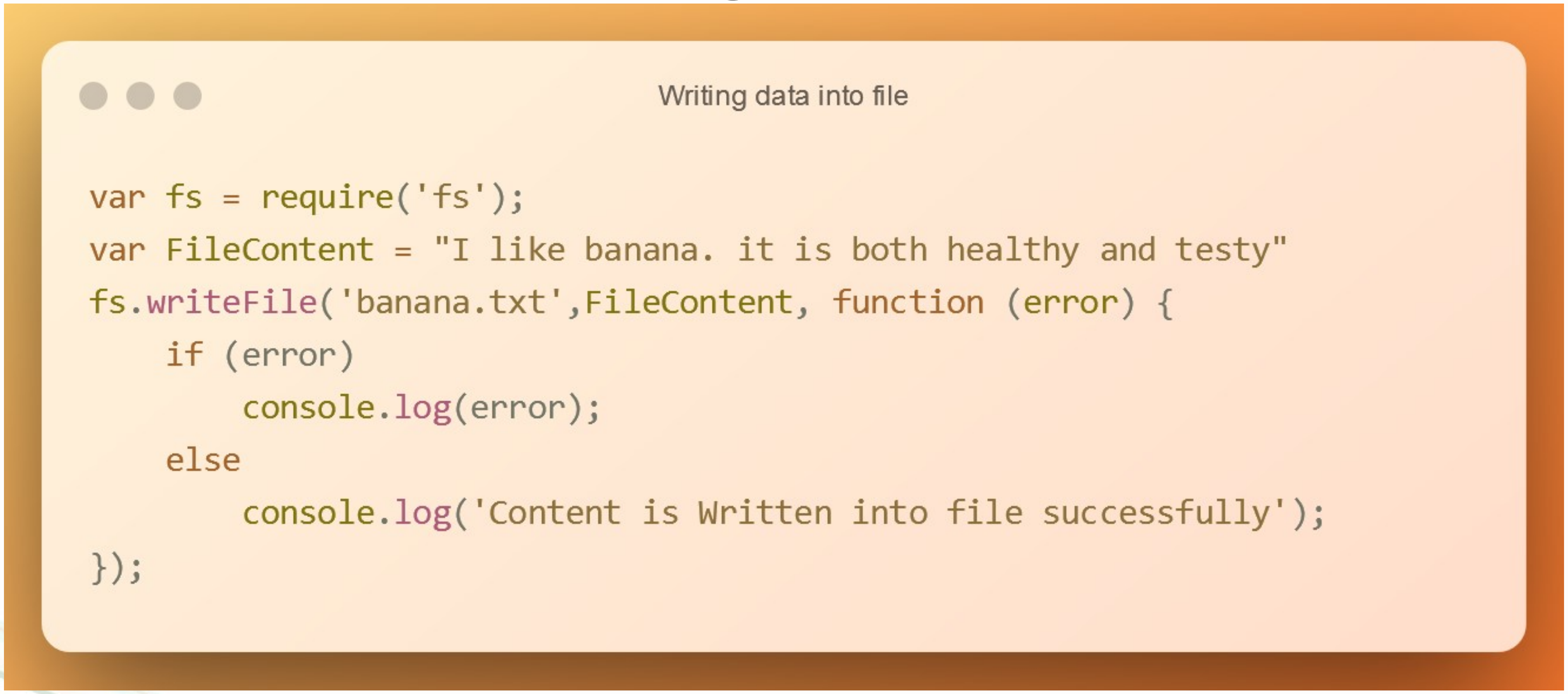


```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (request, response) {
  // synchronous read

  //'myfile.html' must exist in same directory where this file is
  response.writeHead(200, {'Content-Type': 'text/html'});
  var FileContent = fs.readFileSync('myfile.html');
  response.write(FileContent);

  return response.end();
});
server.listen(5000);
```

How to write data into a file?

- To write data into file **asynchronously** use fs.writeFile() method.
- If file already exists then it **overwrites** the existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.
  - 4th argument is callback function which will execute automatically after content is written into file.

Example of writing data into file

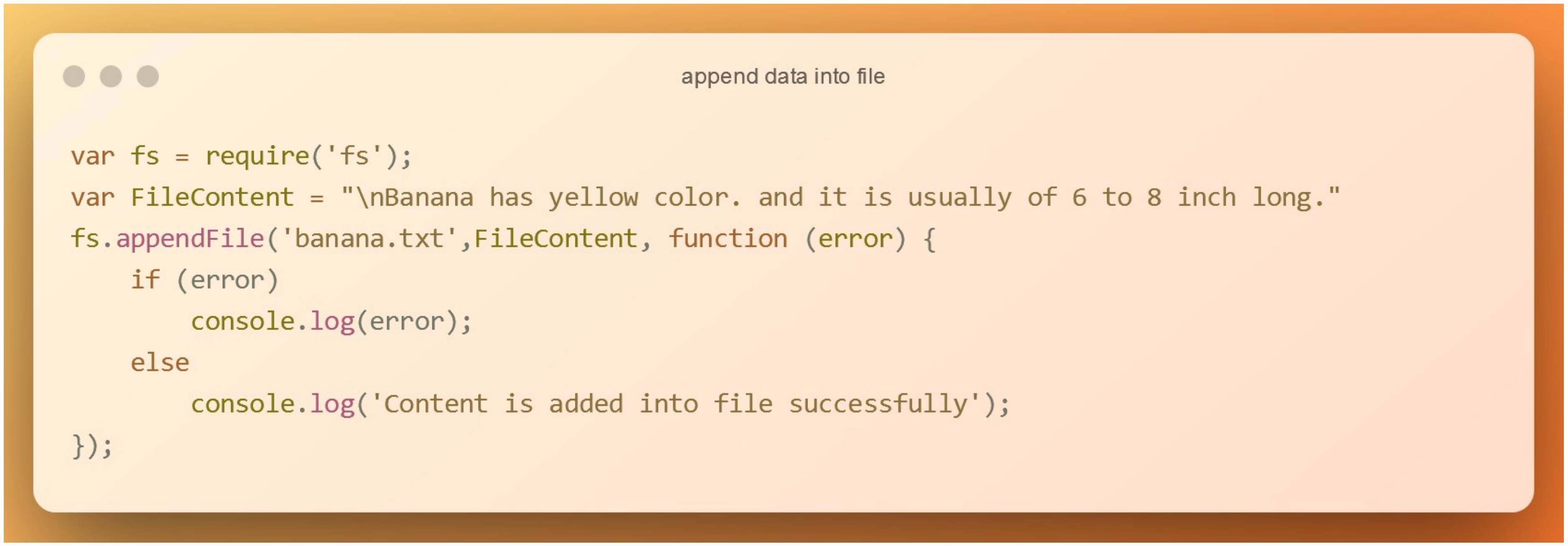


```
var fs = require('fs');
var FileContent = "I like banana. it is both healthy and testy"
fs.writeFile('banana.txt',FileContent, function (error) {
    if (error)
        console.log(error);
    else
        console.log('Content is Written into file successfully');
});
```

How to append (add new data) into existing file?

- To appned data into file asynchronously use fs.appendFile() method.
- If file already exists then it **add new content in** the existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.
  - 4th argument is callback function which will execute automatically after content is written into file.

Example of appending data into file



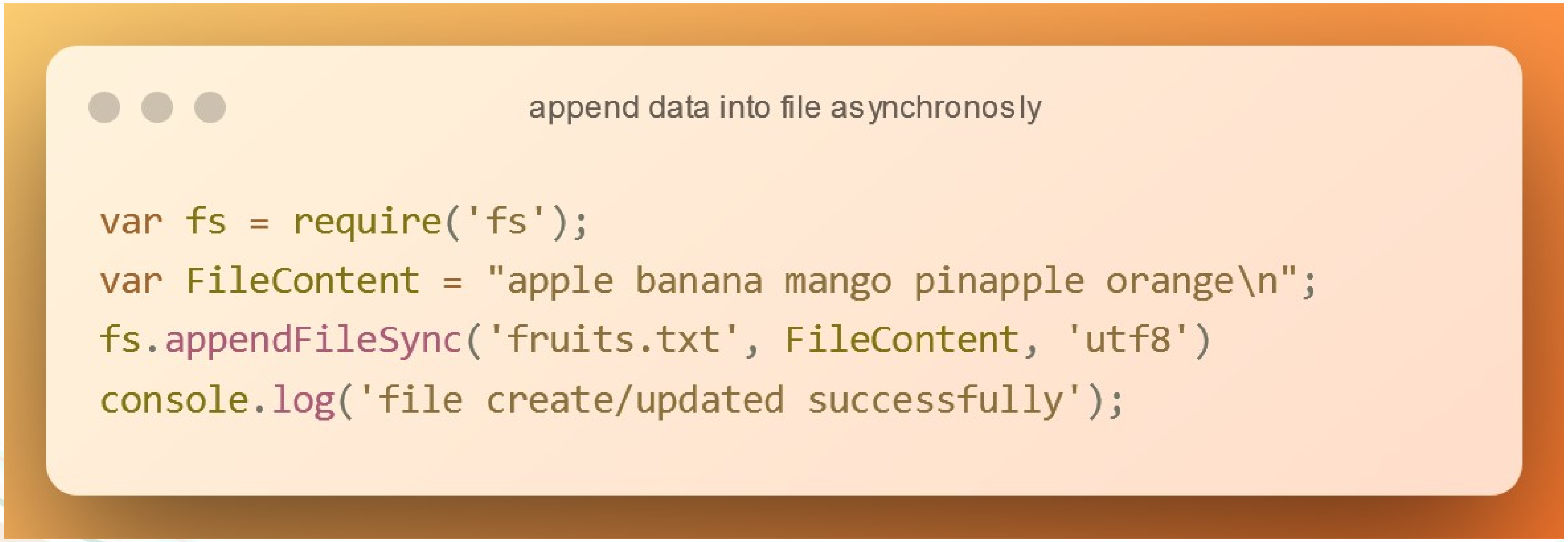
append data into file

```
var fs = require('fs');
var FileContent = "\nBanana has yellow color. and it is usually of 6 to 8 inch long."
fs.appendFile('banana.txt',FileContent, function (error) {
    if (error)
        console.log(error);
    else
        console.log('Content is added into file successfully');
});
```

How to write data into a file synchronously ?

- To write data into file synchronously use fs. appendFileSync() method.
- If file already exists then it **add new content in** the existing content otherwise it creates a **new** file and writes data into it.
- It has 4 arguments.
  - 1st argument is fillname.
  - 2nd argument is data to be written into file.
  - 3rd argument is optional, it include encoding, mode and flag.

Example of writing data into file asynchronously



```
var fs = require('fs');
var FileContent = "apple banana mango pinapple orange\n";
fs.appendFileSync('fruits.txt', FileContent, 'utf8')
console.log('file create/updated successfully');
```

How to Open and close file in specific mode?

- You can also open file in read or write or append mode using fs.open() function instead of using functions learned earlier.
- Syntax

  fs.open(path, flags, mode, callback)
- Parameters:
  - **path:** path and name of the file.
  - **flags:** Flags indicate the type of operation you want to make on file to be opened.
  - **mode:** Sets the mode of file i.e. r-read, w-write, r+ -readwrite. It sets to default as readwrite.
  - Callback : callback function that will execute after file opens. It has 2 argument. 1st argument is error object if any, 2nd argument is reference of the opened file known as fd..

Flags

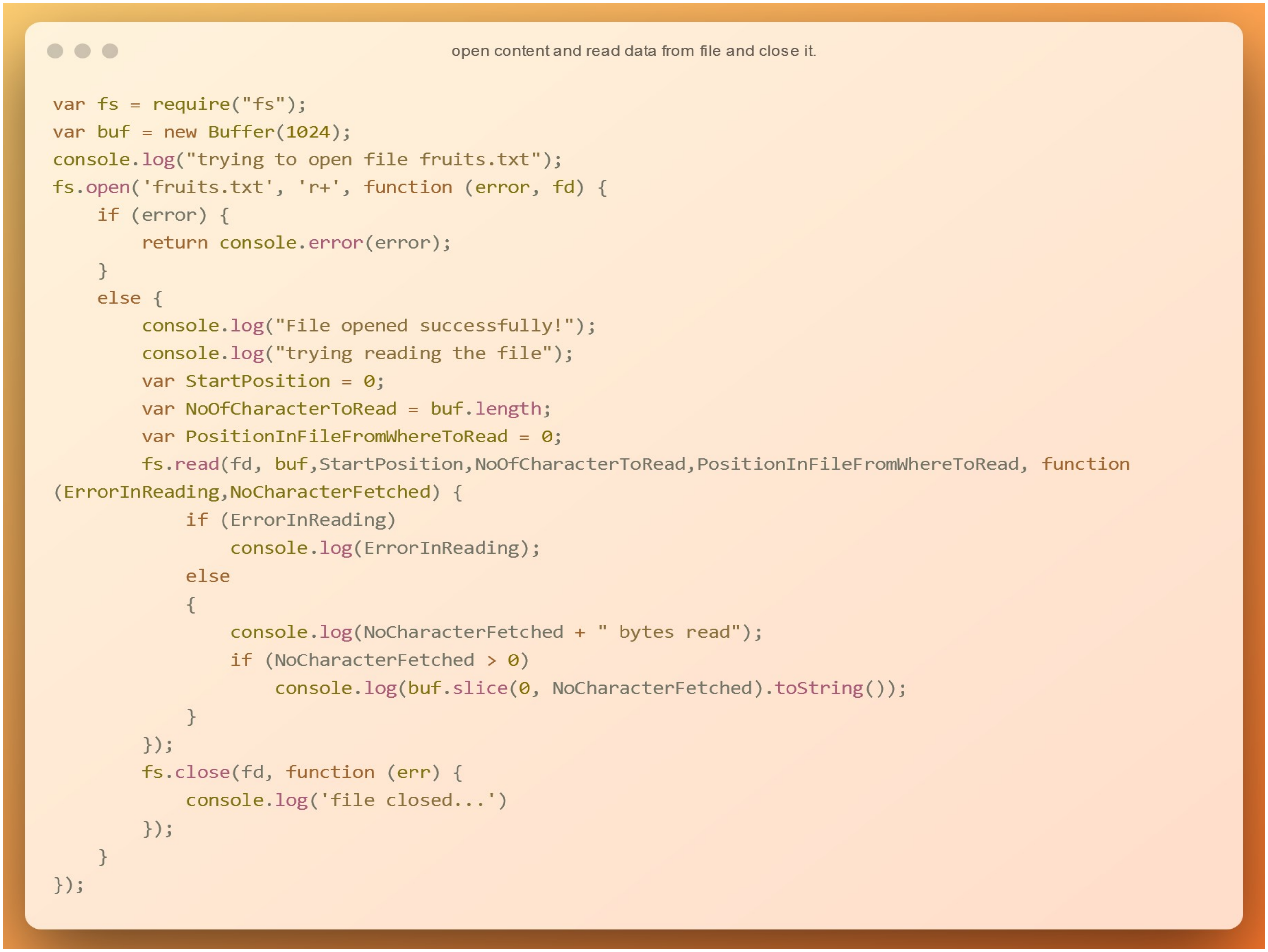
Flag	Description
r	Open file for reading. An exception occurs if the file does not exist.
r+	Open file for reading and writing. An exception occurs if the file does not exist.
rs	Open file for reading in synchronous mode.
rs+	Open file for reading and writing, telling the OS to open it synchronously. See notes for 'rs' about using this with caution.
w	Open file for writing. The file is created (if it does not exist) or truncated (if it exists).
wx	Like 'w' but fails if path exists.
w+	Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists).
wx+	Like 'w+' but fails if path exists.
a	Open file for appending. The file is created if it does not exist.
ax	Like 'a' but fails if path exists.
a+	Open file for reading and appending. The file is created if it does not exist.
ax+	Like 'a+' but fails if path exists.

Read data from file using read method

- The fs.read() method is used to read the file specified by fd. This method reads the entire file into the buffer.
- **Syntax:**
- fs.read(fd, buffer, offset, length, position, callback)
- **Parameters:**
  1. **fd:** This is the file descriptor returned by fs.open() method.
  2. **buffer:** This is the buffer that the data will be written to.
  3. **offset:** This is the offset in the buffer to start writing at.
  4. **length:** This is an integer specifying the number of bytes to read.
  5. **position:** This is an integer specifying where to begin reading from in the file. If the position is null, data will be read from the current file position.
  6. **callback:** It is a callback function that is called after reading of the file. It takes two parameters:
     1. **err:** If any error occurs.
     2. **count:** count of character read from file.

How to close file?

- To close open file, fs.close() method is used.
- It works asynchronously and close the given file descriptor.
- **Syntax:**
- fs.close(fd, callback)
- **Parameters:**
  1. **fd:** file descriptor of the file for which to be closed.
  2. **callback:** function that will execute after file is closed it has 1 argument which is error if any.

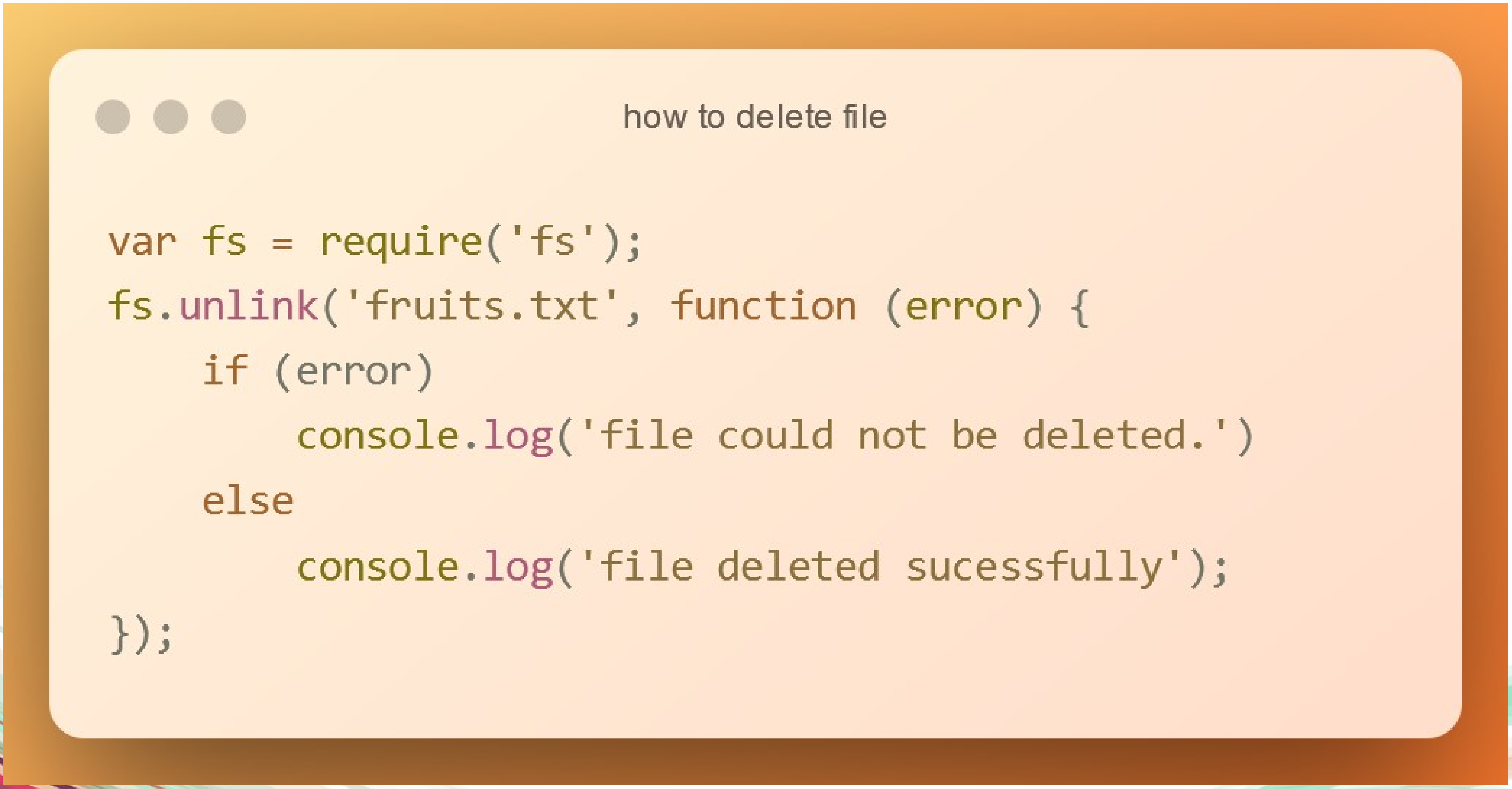


```javascript
var fs = require("fs");
var buf = new Buffer(1024);
console.log("trying to open file fruits.txt");
fs.open('fruits.txt', 'r+', function (error, fd) {
    if (error) {
        return console.error(error);
    }
    else {
        console.log("File opened successfully!");
        console.log("trying reading the file");
        var StartPosition = 0;
        var NoOfCharacterToRead = buf.length;
        var PositionInFileFromWhereToRead = 0;
        fs.read(fd, buf,StartPosition,NoOfCharacterToRead,PositionInFileFromWhereToRead, function
(ErrorInReading,NoCharacterFetched) {
            if (ErrorInReading)
                console.log(ErrorInReading);
            else
            {
                console.log(NoCharacterFetched + " bytes read");
                if (NoCharacterFetched > 0)
                    console.log(buf.slice(0, NoCharacterFetched).toString());
            }
        });
        fs.close(fd, function (err) {
            console.log('file closed...')
        });
    }
});
```

How to delete existing file?

- To delete file we use fs.unlink() method.
- Syntax
- fs.unlink(path, callback)
    1. 1st argument in this function is path and file name
    2. 2nd argument is callback function that will execute after file gets deleted.

Example of how to delete file?

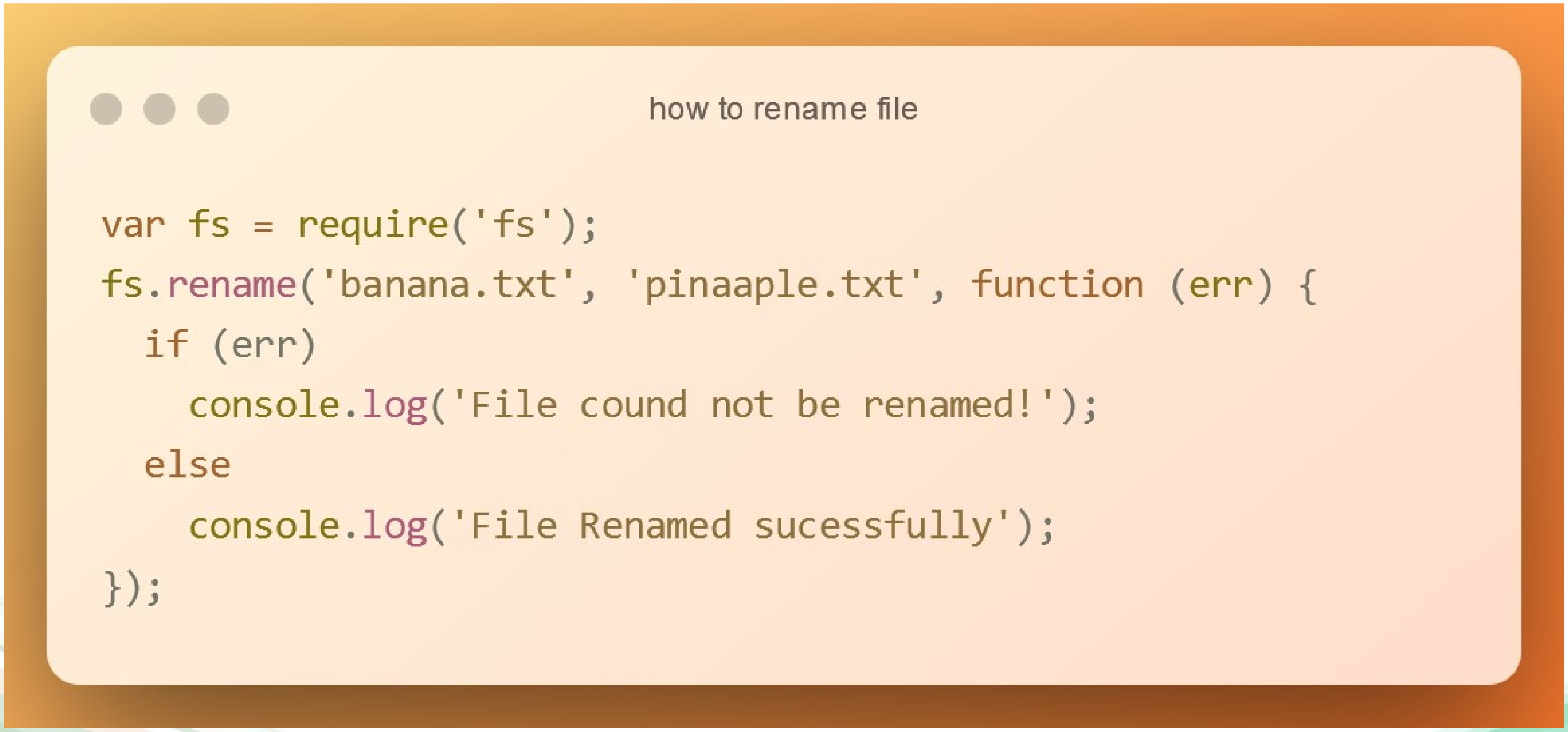


```
var fs = require('fs');
fs.unlink('fruits.txt', function (error) {
    if (error)
        console.log('file could not be deleted.')
    else
        console.log('file deleted sucessfully');
});
```

How to rename file?

- To rename a file with the File System module, use the fs.rename() method.
- The fs.rename() method has 3 arguments.
    1. 1st argument is current file name
    2. 2nd argument is new file name
    3. 3rd argument is callback function that will execute after file is renamed.

Example of how to rename file



```
var fs = require('fs');
fs.rename('banana.txt', 'pinaaple.txt', function (err) {
  if (err)
    console.log('File cound not be renamed!');
  else
    console.log('File Renamed sucessfully');
});
```