Delete notes.txt, testJson.js, 1.json

In notes.js define a new function to create a new note. Get title and body and save it.

```
const addNote = function (title, body) {
}
```

How can we export two functions?

```
module.exports = getNotes
```

İnstead of this, change it to an object with multiple properties

```
module.exports = {
        getNotes: getNotes,
        addNote:addNote
}
```

Now go to app.js
In this file, we require notes.js but now we don't get a single function and the name is misleading. Change it to..

```
const notes = require('./notes.js')
```

Now call the addNote function using the property. Currently we print title and body to the console. Delete the log command and add

```
notes.addNote(argv.title, argv.body)
```

Go to notes.js
Change addNote function to save note to the file. First, we will load existing notes and parse them then add the new note to avoid overwriting existing notes.

Create a new function to load notes that can be used by other functions as well. It will return an array of notes.

```
Const loadNotes = function () {
}
```

Now change this function content

```
const addNote = function (title, body) {
}
```

And add loadNotes in the function..

```
const addNote = function (title, body) {
        const notes = loadNotes()
}
```

Add the fs requirement at the top.

```
const fs = require('fs')
```

Now read from the file in the load function.

```
const loadNotes = function () {
        const dataBuffer = fs.readFileSync('notes.json') //read the file
```

```
    const dataJSON = dataBuffer.toString() //convert to string
    return JSON.parse(dataJSON) //parse it and return
}
```

If there is no file, it will fail.
Try node app.js add —title="t" —body="b"

No such file error.
We can take a defensive action to address the problem.

```
const loadNotes = function () {
    try {
        const dataBuffer = fs.readFileSync('notes.json') //read the file
        const dataJSON = dataBuffer.toString() //convert to string
        return JSON.parse(dataJSON) //parse it and return
    } catch (e) {
        return [] //an empty array
    }
}
```

Add the following log to the end of the addNote function and try the same command.
Console.log(notes)

Now it prints and empty array.
Now add the push function on arrays.

```
const addNote = function (title, body) {
    const notes = loadNotes()

    notes.push({
        title: title,
        body: body
    })
    console.log(notes)
}
```

We will create a new reusable function to save our notes array to a json file.

```
const saveNote = function (notes) {
}
```

Add the following code to the addNote function.

```
const addNote = function (title, body) {
    const notes = loadNotes()

    notes.push({
        title: title,
        body: body
    })
    saveNotes(notes)
}
```

Add the content to the saveNote function. Stringify the data and then save it to the file.

```
const saveNote = function (notes) {
    const dataJson = JSON.stringify(notes)
```

```
        fs.writeFileSync('notes.json', dataJSON)
}
```

Use the same command again. Now the file should have been created. There is an array in the file.
Try adding one more note. Check if append or overwrite.

Add one more feature. Check if the given title already exists.
Create a function for the array filter.
The function will be called for each element in the array.

```
const addNote = function (title, body) {
        const notes = loadNotes()
        const duplicateNotes = notes.filter(function (note) {
                return note.title === title
        })

        if (duplicateNotes.lenth === 0) { //no duplicates

                notes.push({
                        title: title,
                        body: body
                })
                saveNotes(notes)
                console.log('New note added!')
        } else {
                console.log('Note title taken')
        }
}
```

Try duplicate note adding. Try new node addition.

Removing a node!

In app.js, customize the remove yargs command
Add a builder with an object of title and .

```
yargs.command({
    command: 'remove',
    describe: 'Remove a new note',
      builder: {
            title: {
                    describe: 'Note title',
                    demandOption: true,
                    type: 'string'
            }
      }
    handler: function () {
        console.log('Removing the note!')
    }
}
)
```

Go to note.js and add a removeNote function and then export it.

```
const removeNote = function (title) {
}

module.exports = {
        getNotes: getNotes,
        addNote: addNote,
        removeNote: removeNote
}
```

Go back to the app.js and call the defined&exported function in the yargs handler.

```
yargs.command({
    command: 'remove',
    describe: 'Remove a new note',
      builder: {
            title: {
                  describe: 'Note title',
                  demandOption: true,
                  type: 'string'
            }
      }
    handler: function (argv) {
        notes.removeNote(argv.title)
    }
}
)
```

Go to notes.js and add a placeholder in the removeNote function.

```
const removeNote = function (title) {
        console.log(title)
}
```

Now test with the following command
Node app.js remove —title="some title"

Next, wire up removeNote command. Load existing notes. Use array filter to remove matching node if any. Save the newly created array.

Go to notes.js

```
const removeNote = function (title) {
        const notes = loadNotes()
        const notesToKeep = notes.filter(function (note) {
              return note.title !== title
        })
        saveNotes(notesToKeep)
```

```
}
```

Test node app.js remove —title="t"
Try the same command.

If a note is remove, print "Note removed!" With a green background. Otherwise, print "No note found" with a red background.

Go to notes.js In the removeNote function add the following..

```
if (notes.length > notesToKeep.length) {
        console.log(chalk.green.inverse('Note removed!'))
        saveNotes(notesToKeep)
}  else {
        console.log(chalk.red.inverse('No note found!'))
}
```

Add the following to the top in notes.js

```
Const chalk = require('chalk')
```

Try success and error commands now.

Also update messages for the addNote function in notes.js

```
console.log(chalk.green.inverse('New note added!'))
console.log(chalk.red.inverse('Note title taken!'))
```

We will take a break and talk about a js feature, ES6 arrow functions.

Create a new file called 2-arrow-function.js

```
const square = function (x) {
        return x*x
}
console.log(square(3))
```

Run node 2-arrow-function.js
Now run nodemon 2-arrow-function.js

Comment our existing function. And create a new function using the arrow function syntax.

```
const square = (x) => { //First arguments! No need to write function.
        return x*x
}
```

We can create a more concise function.

```
const square = (x) => x*x //no need for return or curly brackets.
```

If you have if statements, etc. more complex, use the second version above.

Comment all above. Define an object.

```
const event = {
        name: 'Birthday Party',
        printGuestList: function () {
                console.log('Guest list for ' + this.name)
        }
}
```

event.printGuestList() and save to see the result.

Now change it to arrow syntax.

```
const event = {
	name: 'Birthday Party',
	printGuestList: () => {
		console.log('Guest list for ' + this.name)
	}
}
```

Save to see the result. It cannot find the event name! Because arrow functions cannot find its own value. Arrow functions don't have access to this. Change it back to earlier version.

There is one more alternative concise version. Remove function and semicolon.

```
const event = {
	name: 'Birthday Party',
	printGuestList() {
		console.log('Guest list for ' + this.name)
	}
}
```

Save and see the results.

Now add the guest list.

```
const event = {
	name: 'Birthday Party',
	guestList: ['A', 'B', 'C'],
	printGuestList() {
		console.log('Guest list for ' + this.name)

		this.guestList.forEach(function (guest) { //will be executed for each element of the array.
			console.log(guest + ' is attending ' + this.name)
		})
	}
}
```

Save and see the results. But this references to their own bindings. Event name is not accessible.
Access the parent function!
There is a trick to overcome the problem but is not ideal. Define a new variable
const that = this
And print that.name

It works but not ideal. Remove that again.. Remove function keyword.

```
const event = {
	name: 'Birthday Party',
	guestList: ['A', 'B', 'C'],
	printGuestList() {
		console.log('Guest list for ' + this.name)

		this.guestList.forEach((guest) => { //will be executed for each element of the array.
			console.log(guest + ' is attending ' + this.name)
		})
	}
}
```

When we save the file, it works as expected.