Create a new playground script: 3-arrow-challenge.js
Run nodemon 3-arrow-challenge.js

Copy the following:

```
const tasks = {
    tasks: [{
        text: 'Grocery shopping',
        completed: true
    }, {
        text: 'Clean yard',
        completed: false
    }, {
        text: 'Film course',
        completed: false
    }],
}
console.log(tasks.getTasksToDo())
```

Currently, it doesn't work. The function is not defined.
Define getTasksToDo method, use filter to return incomplete tasks using arrow syntax.

```
const tasks = {
    tasks: [{
        text: 'Grocery shopping',
        completed: true
    }, {
        text: 'Clean yard',
        completed: false
    }, {
        text: 'Film course',
        completed: false
    }],
            //getTasksToDo: function () {
            //} then remove function and semicolon!
    getTasksToDo() {
        Const tasksToDo = this.tasks.filter((task) => {
                        return task.completed === false
                        })
                        return tasksToDo
    }
}
console.log(tasks.getTasksToDo())
```

Save and see the results as expected.
It is also possible to return the result directly.

```
    getTasksToDo() {
        return this.tasks.filter((task) => {
                        return task.completed === false
```

```
            })
    }
```

One more step to make it more concise. Remove everything in the curly brackets.

```
    getTasksToDo() {
        return this.tasks.filter((task) => task.completed === false
    }
```

Now close two playground files.

In app.js .. If a function is a method, use ES6 method definition syntax otherwise, use most concise arrow function possible.

Go to app.js
In yargs.command handler remove function keyword.

handler: function (argv) becomes
handler(argv)

Do the same thing for add, remove, list and read.
For list and remove use handler()

Now go to notes.js
getNotes function can be an arrow function. It can be even more concise but we will add more content to it.

```
const getNotes = () =>  {
        return 'your notes..'
}
```

addNote function..

```
const addNote = (title, body) => {
```

Filter function in addNote..

```
const duplicateNotes = notes.filter((note) => note.title === title)
```

removeNote function..

```
const removeNote = (title) => {
```

Filter function in removeNote..

```
const notesToKeep = notes.filter((note) => note.title !== title)
```

saveNotes..

```
const saveNotes = (notes) => {
```

loadNotes..

```
const loadNotes = () => {
```

Now save and test.

Remove
node app.js remove —title=“List”
Run again. Must be error.
node app.js add —title=“t” —body=“b”
Run again.

List notes.
Create and export listNotes from notes.js
Your notes using chalk. Print title for each note.
Call listNotes from command handler

Add some notes first.
Go to notes.js and define a new function in arrow syntax.

```
const listNotes = () => {
}
```

Export it.

```
module.exports = {
..
        listNotes: listNotes
}
```

Go to app.js and customize list yargs command.

```
yargs.command({
        command: 'list',
        describe: 'List your notes',
        handler: function () {
                notes.listNotes()
        }
} )
```

Go back to notes.js and customize the function.

```
const listNotes = () => {
        const notes = loadNotes()

        console.log(chalk.inverse('Your notes'))

        notes.forEach((note) => {
                console.log(note.title)
        })
}
```

Test now.
node app.js list

Reading a note

First change this..
Note.js, addNote function, filter method. It checks every item even though it finds a duplicate earlier in the list. Let's stop the process when we find a duplicate. Let's use find method instead.

```
const duplicateNote = notes.find((note) => note.title === title)
```

Change the if condition

```
if (!duplicateNote) {
```

Or

```
if (duplicateNote === undefined)
```

Go to app.js
In the read yargs command, add a builder..

```
builder: {
        title: {
                describe: 'Note title',
                demandOption: true,
                type: 'string'
        }
}
```

Go to notes.js
Add a new method

```
Const readNote = (title) => {

}
```

Add to exports

```
Module.exports = {
…
        readNote: readNote
}
```

Implement the new method

```
Const readNote = (title) => {
        const notes = loadNotes()
        const note = notes.find((note) => note.title === title)

        if (note) {
                console.log(chalk.inverse(note.title))
                console.log(note.body)
        } else {
                console.log(chalk.red.inverse('Note not found!'))
        }
}
```

Go to app.js and add the handler to call the new function.

```
handler(argv) {
        notes.readNote(argv.title)
}
```

Test the command.
node app.js read —title="dd"
Rerun for a missing title
node app.js read —title="aaa"

Remove getNotes function at the top of the notes.js, also remove from the exports.

Debugging node.js
Console.log can be used for debugging

Go to notes.js, addNote function.
Lets assume that else never works. To understand why, we can print duplicateNote variable.
Before the if condition, add
Console.log(duplicateNote)

Run
node app.js add —title="Course" —body="nodejs"
See undefined. Then run again and see taken title.

Add to see the title also.
Console.log(title)

We can also use debugger
Before the if condition, add

debugger

Rerun the command. Nothing happens, the program doesn't stop.
Now run
node inspect app.js add —title="Course" —body="nodejs"
For windows, there is an error, use node —inspect-brk instead!

Open chrome and go to chrome://inspect/
Under remote target, select the process.

Open sources tab. There is app.js file.
The first line is for the main function. It is closed at the end.
On the left, click add folder to workspace and select the folder.
Now the file has changed and become similar to our code.

Green icon is the active file.
Nothing is executed yet.
Debugger is paused now. If click the resume button, it will resume and stop at the debugger line.
We can check the scope to see the values of variables.

If click the run again it will run to the end of the file.

We can type restart in the command line and target will be visible again.

Now remove debug from the code.

Error messages
Go to notes.js saveNotes function.
Change the variable dataJSON to dataJsON in the writeFileSync method.
See the error message.
Not defined error. In the trace, it shows the error line.