INFO3139 JAVASCRIPT

INFO 3139 Week 1

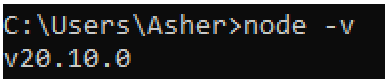
Software setup

1. Install node v. 20.10.0 LTS (Recommended for Most Users)

   https://nodejs.org/en

   Do Not install the optional necessary tools.

   Check node version in command prompt:

   ```
   C:\Users\Asher>node -v
   v20.10.0
   ```

2. Install Visual Studio Code

   https://code.visualstudio.com/

3. Add ***Prettier – Code formatter*** extension to Visual Studio

4. Create the following:

   Folder: Node Exercises -> week1 ->

   Files:

   ➔ arrow_example1.js

   ➔ arrow_example2.js

   ➔ blocking_example.js

   ➔ non_blocking_example.js

   ➔ got_example1.js

   ➔ got_example2.js

Programs (6)

1/6 - arrow_example1.js

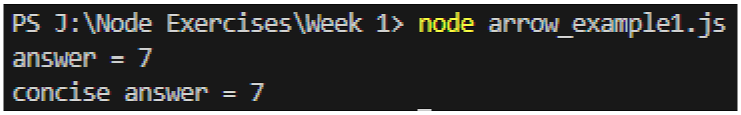
Arrow functions in Node.js

In Node.js, Arrow functions are a concise way to write functions in JavaScript. Arrow functions provide a more concise syntax compared to traditional function expressions.

Code:

```
//traditional way
let add = (x, y) => {
    return x + y;
};
console.log(`answer = ${add(4, 3)}`);
// a more concise way
let add2 = (x, y) => x + y;
console.log(`concise answer = ${add2(4, 3)}`);
```

Execute the code:



```
PS J:\Node Exercises\Week 1> node arrow_example1.js
answer = 7
concise answer = 7
```

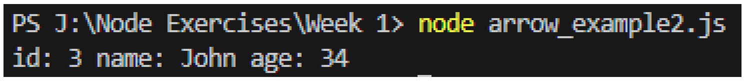
2/6 - arrow_example2.js

Creating objects and accessing object properties using arrow functions

Code:

```
// Arrow Functions Example #2 - returning object properties
let setValues = (id, name, age) => ({ id: id, name: name, age: age });
let student = setValues(3, "John", 34);
console.log(`id: ${student.id} name: ${student.name} age: ${student.age}`);
```

Execute the code:



```
PS J:\Node Exercises\Week 1> node arrow_example2.js
id: 3 name: John age: 34
```

3/6 - blocking_example.js

Create new folder within week1:

- sampledata

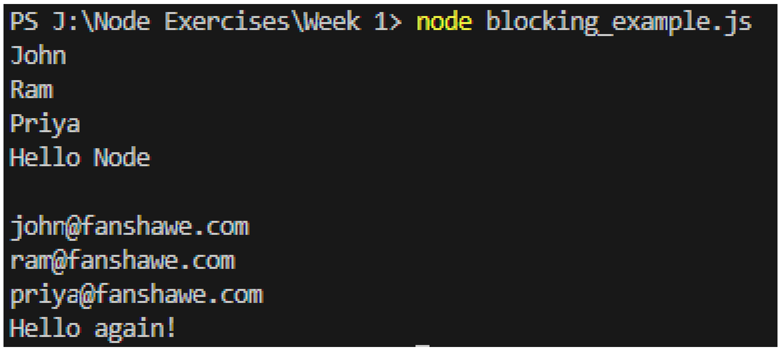
Create two files within sampledata:

- users
  - John
  - Ram
  - Priya
- emailids
  - john@fanshawe.com
  - ram@fanshawe.com
  - priya@fanshawe.com

Code in blocking_example.js

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

let users = fs.readFileSync('./sampledata/users', 'utf8');

console.log(users);

console.log('Hello Node\n');

let emails = fs.readFileSync('./sampledata/emailids', 'utf8');

console.log(emails);

console.log('Hello again!');
```

Execute the code:



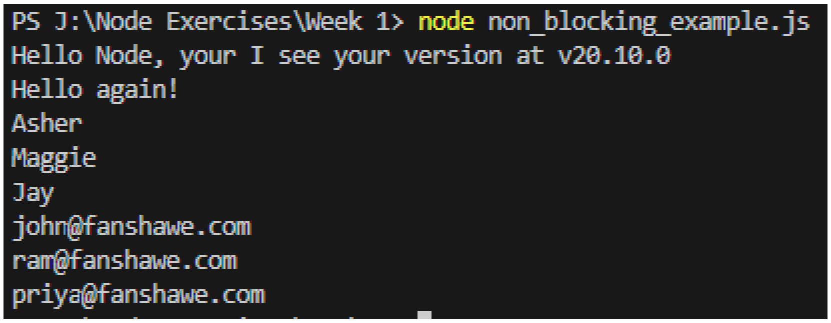
```
PS J:\Node Exercises\Week 1> node blocking_example.js
John
Ram
Priya
Hello Node

john@fanshawe.com
ram@fanshawe.com
priya@fanshawe.com
Hello again!
```

4/6 - non_blocking_example.js

Code:

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

fs.readFile('./sampledata/users', 'utf8', (err, contents) => {

    let users = contents;

    console.log(users);

});

console.log('Hello Node, your I see your version at ' + process.version);

fs.readFile('./sampledata/emailids', 'utf8', (err, contents) => {

    let emails = contents;

    console.log(emails);

});

console.log('Hello again!');
```

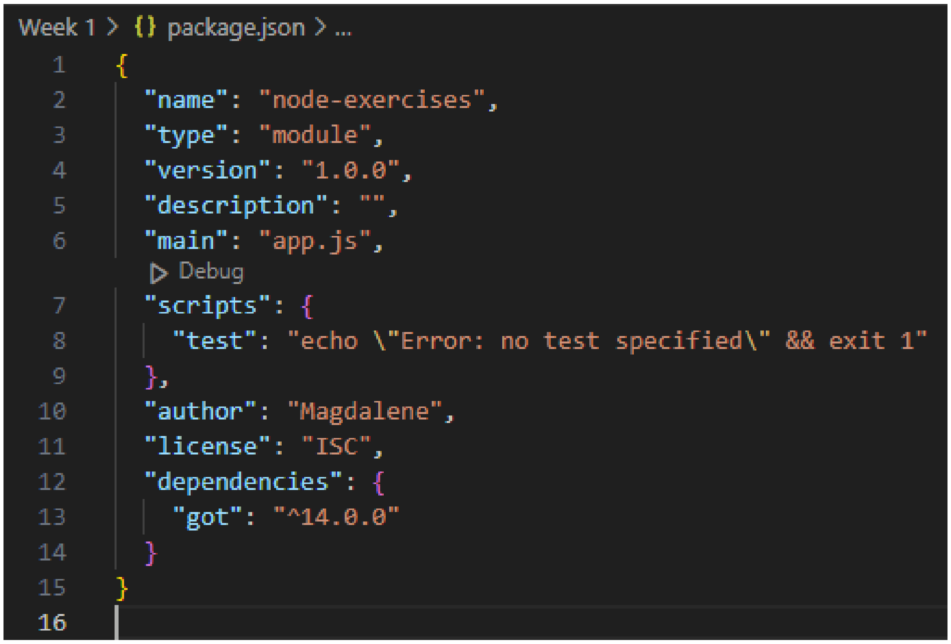
Execute the code:



```
PS J:\Node Exercises\Week 1> node non_blocking_example.js
Hello Node, your I see your version at v20.10.0
Hello again!
Asher
Maggie
Jay
john@fanshawe.com
ram@fanshawe.com
priya@fanshawe.com
```

Npm

- Return to the Terminal tab in and change directories to the nodeexercises root
- Issue the npm command npm init (npm init will create a file called package.json).
- Take most of the defaults but change the author to your name and the entry option to app.js:
- Look at the VSCode explorer now nodeeexercises directory and you should see the newly created package.json file.
- Now, install the got package with the package manager:
- Update package.json to include “type” : “module”



```
{
  "name": "node-exercises",
  "type": "module",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Magdalene",
  "license": "ISC",
  "dependencies": {
    "got": "^14.0.0"
  }
}
```

5/6 - got_example1.js

Code:

```
// Load the got module

import got from "got";

// Try to make an HTTP GET using got to Fanshawe's website.

const dumpPage = async () => {

try {

const response = await got("http://www.fanshaweonline.ca");

console.log(response.body);

//=> '<!doctype html> ...'

} catch (error) {

console.log(error.response.body);

//=> 'Internal server error ...'
}
};
dumpPage();
```

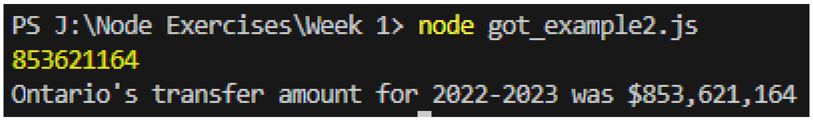
Execute Code:

Will throw the raw HTML data.

6/6 - got_example2.js

```
// Load the got module
import got from "got";
// Lets try to make a HTTP GET request to GOC's website and get some transfer info in JSON.
const dumpJson = async () => {
const srcAddr =
"http://www.infrastructure.gc.ca/alt-format/opendata/transfer-program-programmes-de-transfert-bil.json";
// Create a currency formatter.
const formatter = new Intl.NumberFormat("en-US", {
style: "currency",
currency: "USD",
minimumFractionDigits: 0,
});
try {
const response = await got(srcAddr, { responseType: "json" });
console.log(response.body.gtf.on["2021-2022"]);

//console.log(Object.keys(response.body.gtf.on.total));

// strip out the Ontario amount

let ont = response.body.gtf.on["2022-2023"];

// format to currency

ont = formatter.format(ont);

// dump to the console using template literal

console.log(`Ontario's transfer amount for 2022-2023 was ${ont}`);

} catch (error) {

console.log(error);

//=> 'Internal server error ...'

}

};

dumpJson();
```

Execute Code:



```
PS J:\Node Exercises\Week 1> node got_example2.js
853621164
Ontario's transfer amount for 2022-2023 was $853,621,164
```

Assigned Readings:

- https://javascript.info/intro
- https://www.digitalocean.com/community/tutorials/an-introduction-to-json

INFO3139 JAVASCRIPT

- https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm#:~:text=is%20as%20follows%20%E2%88%92-,Node.,that%20run%20across%20distributed%20devices
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals
- https://dmitripavlutin.com/differences-between-arrow-and-regular-functions/
- https://www.freecodecamp.org/news/what-is-npm-a-node-package-manager-tutorial-for-beginners/

Assignment:

One word document containing screenshots of executed code on VS for all six programs. Attempt to modify/edit/enhance the code to produce a different output.