NodeJS Core Modules

https://nodejs.org/api/

More on Globals

https://nodejs.org/api/globals.html
We have already covered global variables in previous session.

filename, dirname, setInterval, setTimeout, clearInterval, clearTimeout

```
// https://nodejs.org/api/globals.html
// console.log(exports);
// console.log(global);

console.log(__dirname);
console.log(__filename);

// Runs every second
let seconds = 0;
let interval = setInterval(() => {
  console.log(`Running ${++seconds} ...`);
}, 1000, "abc");

// Stops interval after 5 seconds
let timeout = setTimeout(() => {
  console.log(`Clearing interval.`);
  clearInterval(interval);
}, 5000);

// Stops timeout
// clearTimeout(timeout);
```

application arguments, environment variables, process

https://nodejs.org/api/process.html#process_process

```
console.log("process.env = ", process.env);
console.log("process.argv = ", process.argv);

process.on("exit", function (status) {
  console.log(`Application is about to end: ${status}`);
});

// process.exit("sheraz");
process.exit(0);
```

```
===========
/Users/sheraz/dev/workspace/lunch_and_learn/lal_105_nodejs_core_modules_mysql
/Users/sheraz/dev/workspace/lunch_and_learn/lal_105_nodejs_core_modules_mysql/app01_gl
obal.js
Running 1 ...
Running 2 ...
Running 3 ...
Running 4 ...
Clearing interval.
===========
```

Path

https://nodejs.org/api/path.html

```
import path from "path";

let path01String = "//home/./mypath//../mypath/myfile.log";

console.log("path.normalize() = " + path.normalize(path01String));
console.log("path.dirname() = " + path.dirname(path01String));
console.log("path.basename() = " + path.basename(path01String));
console.log("path.extname() = " + path.extname(path01String));
```

```
=========
$ babel-node app03_path.js --presets es2015
path.normalize() = /home/mypath/myfile.log
path.dirname() = //home/./mypath//../mypath
path.basename() = myfile.log
path.extname() = .log
=========
```

File system

https://nodejs.org/api/fs.html

```
import fs from "fs";

let fileName = "file01.log";
// Write
fs.writeFileSync(fileName, "Content in the file.");
console.log("File Ceated: " + fileName);

// Read
let fileContent = fs.readFileSync(fileName).toString();
console.log("File Reading: " + fileName);
console.log("File Content: " + fileContent);

// Delete - unlink is alias of rm
fs.unlinkSync(fileName);
console.log("File Deleted: " + fileName);
==========
$ babel-node app04_file_system.js --presets es2015
File Ceated: file01.log
File Reading: file01.log
File Content: Content in the file.
File Deleted: file01.log
==========
```

Event Emitter

https://nodejs.org/api/events.html

```
import EventEmitter from "events";

// Event name constants
const myEvents = {
  SALARY: "SALARY",
  NAME: "NAME"
};

// Defining Observable / Observable data / EventEmitter
class MyObservable {
  constructor(name, salary) {
     this.observableData = {
        name: name,
        salary: salary
     };
     this.eventEmitter = new EventEmitter();
  }
  setSalary(salary) {
     this.observableData.salary = salary;
     this.eventEmitter.emit(myEvents.SALARY, this.observableData);
  }

  setName(name) {
     this.observableData.name = name;
     this.eventEmitter.emit(myEvents.NAME, this.observableData);
  }

  addObserver(eventName, observer) {
     this.eventEmitter.on(eventName, observer);
  }

  removeObserver(eventName, observer) {
     this.eventEmitter.removeListener(eventName, observer)
  }
}

// Defining Observer Functions
let nameObserver = (observableData) => {
  console.log("Name Changed", observableData);
};

let salaryObserver = (observableData) => {
  console.log("Salary Changed", observableData);
};

let profileObserver = (observableData) => {
  console.log("Profile Changed", observableData);
};


// Initializing Observable/EventEmitter
let myObservable = new MyObservable("Sheraz", 100);

// Adding Observers/Listener to Events
myObservable.addObserver(myEvents.NAME, nameObserver);
myObservable.addObserver(myEvents.SALARY, salaryObserver);

// NOTE: profileObserver is added on both myEvents.NAME, and myEvents.SALARY events
myObservable.addObserver(myEvents.NAME, profileObserver);
myObservable.addObserver(myEvents.SALARY, profileObserver);
```

```
// Changing value will emit events.
// It will invoke Observer/Listener
myObservable.setName("Chaudhry");
console.log("#########");
myObservable.setSalary(1000);

// Removing Observers/Listener from Events
console.log("#########");
myObservable.removeObserver(myEvents.SALARY, salaryObserver);

// changing value after Observers/Listener is removed from Events
myObservable.setSalary(2000);
```

```
==========
$ babel-node app05_events_eventemitter.js --presets es2015
Name Changed { name: 'Chaudhry', salary: 100 }
Profile Changed { name: 'Chaudhry', salary: 100 }
#########
Salary Changed { name: 'Chaudhry', salary: 1000 }
Profile Changed { name: 'Chaudhry', salary: 1000 }
#########
Profile Changed { name: 'Chaudhry', salary: 2000 }
==========
```

Http

https://nodejs.org/api/http.html

Simple Example

```
// import http from "http";
let http = require("http");

// Engine
const engine = (request, response) => {
  response.writeHead(200, {"Content-Type": "text/html"});
  response.end("<h1>Node.js http</h1>");
};

// Create Server
let server = http.createServer(engine);
```

```
// Start Server
server.listen(8080, () => {
  console.log("Server started. Listening on port 8080.");
});
```

Sending back static file and JSON

```
// import http from "http";
let http = require("http");
let fs = require("fs");

const engine = (request, response) => {
  console.log(request.url);
  if (request.url.indexOf("/rest") > -1) {
    let myObject = {
      name: "My Name",
      age: 30
    };
    response.writeHead(200, {"Content-Type": "application/json"});
    // Can not send back object. It has to be String
    response.end(JSON.stringify(myObject));
  } else if (request.url.indexOf("/profile") > -1) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.end(fs.readFileSync("./views/static_profile.html").toString());
  } else {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.end(fs.readFileSync("./views/static_home.html").toString());
  }
};

let server = http.createServer(engine);
server.listen(8080, () => {
  console.log("Server started. Listening on port 8080.");
});
```

MySQL

https://www.npmjs.com/package/mysql
https://github.com/mysqljs/mysql

Connection and error handling

```
let mysql = require("mysql");

// Connection Config
let connectionConfig = {
  host: "localhost",
  user: "root",
  password: "root",
  port: 8889,
  database: "LAL"
};

// Create Connection Factory
let connection = mysql.createConnection(connectionConfig);

// Connect
connection.connect((error) => {
  if (error) {
      console.log("Error connecting.", error);
  } else {
      console.log("Connected to DB");
  }
});

// Query
connection.query('SELECT 1', function (error, results, fields) {
  if (error) {
      console.log(error.code); // 'ECONNREFUSED'
      console.log(error.fatal); // true
  } else {
      console.log("Database hit successful");
  }
});

// End Connection
connection.end((error) => {
  if (error) {
      console.log("Error disconnecting.", error);
  } else {
      console.log("Disconnected DB");
  }
});
```

Transaction

https://github.com/mysqljs/mysql/blob/master/Readme.md#transactions

DML

NOTE: INSERT statement's "VALUES" clause do not work. Not sure why. I can't find an example even in the node mysql documentation.

```
let mysql = require("mysql");

let connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "root",
  port: 8889,
  database: "LAL"
});

connection.connect();

let newProfileObject = {
  NAME: "Profile Object",
  SALARY: 400
};

// Set Object
let query1 = connection.query('INSERT INTO USER_PROFILE SET ?', newProfileObject, function
(error, result) {
  console.log("Insert query: ", query1.sql);
  console.log("Inserted ID: ", result.insertId);
  console.log("Ending");
});

// Set individual items
let query2 = connection.query('INSERT INTO USER_PROFILE SET NAME=?, SALARY=?',
["Profile Individual",200], function (error, result) {
  console.log("Insert query: ", query2.sql);
  console.log("Inserted ID: ", result.insertId);
  console.log("Ending");
});
```

```
connection.end();
```

SQL

```
let mysql = require("mysql");

let connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "root",
  port: 8889,
  database: "LAL"
});

connection.connect();

let query = connection.query('SELECT * FROM USER_PROFILE WHERE SALARY > ?', 10,
function (error, result) {
  result.forEach(record => {
     console.log(`${record.ID} | ${record.NAME} | ${record.SALARY}`);
  });
});

connection.end();
```