Node.Js Practical

```
const readline = require("readline")
const fs = require("fs")   // file sysstem
const http = require("http")
const url = require('url')
const events = require('events')
```

Console Input and Output

```
const rl =readline.createInterface({
input: process.stdin,
output: process.stdout
}) ;

rl.question("Please enter your name : ", (name) => {
console.log("You entered : " + name) ;
rl.close()
}) ;

rl.on('close', () => {
console.log("Interface Closed!") ;
process.exit(0)
})
```

Read and Write File

```
const readline = require("readline")
const fs = require("fs") // file sysstem

let textIn = fs.readFileSync("./files/random.txt", 'utf-8')
console.log(textIn) ;
console.log(typeof textIn) ; // string

let content = `Data read from input.txt: \n${textIn}. \nDate created: ${new Date()}` ;
fs.writeFileSync("./files/output.txt", content)
```

Asynchronous nature of Node.JS

```
let textIn = fs.readFileSync("./files/random.txt", 'utf-8')
if file is too large and thread takes time to read file till then thread is BLOCKED means
it will not execute further just wait to finish reading.
--> synchronous codes are BLOCKING here.

[ `fs.readFile` ] is ASYNCHRONOUS (non BLOCKING) method
fs.readFile("./files/random.txt", 'utf-8', (err, data) => {
   console.log(data) ;
```

```
}) ;
console.log("Reading the File...") ;

fs.readFile("./files/random.txt", 'utf-8', (error1, data1) => {
   console.log(data1) ;
   fs.readFile("./files/output.txt", 'utf-8', (error2, data2) => {
      console.log(data2) ;
      console.log(error2) ;
   }) ;
}) ;
```

callback-hell --> go for SYNCHRONOUS (use promise, async await)

Creating a simple web server

called whenever new request hits the server

step 1. CREATE A SERVER

```
const server = http.createServer((req, res) => {
   res.end("<H2>Hello from the server</H2>")
   console.log("A new request is recieved!")
   console.log(res)
   console.log(req)
})
```

step 2. START THE SERVER

```
server.listen(8000, '127.0.0.1', () => {
   console.log("Server has started!")
})
```

An overview of how web works

4 main Request types --> GET(fetch data from server)
POST(create a data)
PUT(update data)
DELETE(delete)

How Request & Response works

```
const html_content = fs.readFileSync("./template/index.html", 'utf-8')
const server = http.createServer((req, res) => {
   res.end(html_content)
})

server.listen(8000, '127.0.0.1', () => {
   console.log("Server has started!")
}) ;
```

What is ROUTING

We can make our applications to respond to different urls with different responses using routing
ROUTING basically means implementing different actions for different URLS
These actions can be implemented in different ways, for example, by creating function
1. resource based URL --> send resource of home ['www.nodeapp.com/home', 'www.nodeapp.com/about']
2. route parameter --> ['www.nodeapp.com/home/101'] (home--> resource, 101-->parameter)
3. query string --> ['www.nodeapp.com/Books?author=john&id=101'] ('author=john&id=101' --> query string)

Code given below --

```
const html_content = fs.readFileSync("./template/index.html", 'utf-8')
const server = http.createServer((req, res) => {
   // res.end(html_content)
   let path = req.url ;
   // res.end(path)
   if(path === '/' || path.toLocaleLowerCase() === '/home') {
      res.end("You are routing to the home page!")
   } else if(path.toLocaleLowerCase() === '/about') {
      res.end("You are routing to the about page!")
   } else if(path.toLocaleLowerCase() === '/contact') {
      res.end("You are routing to the contact page!")
   } // handling url routes
   else {
      res.end("ERROR 404: Page not Found")
   }
})
server.listen(8000, '127.0.0.1', () => {
   console.log("Server has started!")
}) ;
```

Sending HTML Response

Setting headers for Response

```
const html_content = fs.readFileSync("./template/index.html", 'utf-8')
const server = http.createServer((req, res) => {
   let path = req.url ;
   if(path === '/' || path.toLocaleLowerCase() === '/home') {
      res.writeHead(200, { // property
         'Content-Type': 'text-html',
         'my-header' : 'Hello, World'
      })
```

```
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in HOME Page'))
    } else if(path.toLocaleLowerCase() === '/about') {
        res.writeHead(200, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
    } else if(path.toLocaleLowerCase() === '/contact') {
        res.writeHead(200, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
    } // handling url routes
    else if(path.toLocaleLowerCase() === '/products') {
        res.writeHead(200, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in PRODUCTS Page'))
    } else {
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
})
server.listen(8000, '127.0.0.1', () => {
    console.log("Server has started!")
}) ;
```

Working with JSON data

```
const html_content = fs.readFileSync("./template/index.html", 'utf-8')
const products = JSON.parse(fs.readFileSync("./data/products.json", 'utf-8')) // convert into js
const server = http.createServer((req, res) => {
    let path = req.url ;
    if(path === '/' || path.toLocaleLowerCase() === '/home') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in HOME Page'))
    } else if(path.toLocaleLowerCase() === '/about') {
```

```
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
    } else if(path.toLocaleLowerCase() === '/contact') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
    } // handling url routes
    else if(path.toLocaleLowerCase() === '/products') {
        res.writeHead(200, { 'Content-Type': 'application/json' }) // property
        // fs.readFile("./data/products.json", 'utf-8', (error, data) => {
        //     res.end(data) ;
        // }) // it will read json file for each request made. better read once and store in variable
        // res.end(html_content.replace('{{%CONTENT%}}', 'You are in PRODUCTS Page'))

        res.end("You are in PRODUCTS page!")
        console.log(products)
    } else {
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
})
server.listen(8000, '127.0.0.1', () => {
    console.log("Server has started!")
}) ;
```

Transforming JSON data into HTML

```
const html_content = fs.readFileSync("./template/index.html", 'utf-8')
const products = JSON.parse(fs.readFileSync("./data/products.json", 'utf-8')) // convert into js obj
const product_list_html = fs.readFileSync("./template/product-list.html", 'utf-8')
const productDetailsHtml = fs.readFileSync("./template/product-details.html", 'utf-8')

let productHtmlArray = products.map((prod) => {
    let output = product_list_html.replace('{{%IMAGE%}}', prod.productImage) ;
    output = output.replace('{{%NAME%}}', prod.name)
    output = output.replace('{{%MODELNAME%}}', prod.modeName)
```

```
    output = output.replace('{{%MODELNO%}}', prod.modelNumber)
    output = output.replace('{{%SIZE%}}', prod.size)
    output = output.replace('{{%CAMERA%}}', prod.camera)
    output = output.replace('{{%PRICE%}}', prod.price)
    output = output.replace('{{%COLOR%}}', prod.color)
    output = output.replace('{{%ID%}}', prod.id)

    return output ;
}) ;

const server = http.createServer((req, res) => {
    let path = req.url ;
    if(path === '/' || path.toLocaleLowerCase() === '/home') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}','You are in HOME page.'))
    } else if(path.toLocaleLowerCase() === '/about') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
    } else if(path.toLocaleLowerCase() === '/contact') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
    }   // handling url routes
    else if(path.toLocaleLowerCase() === '/products') {
        let productResponse = html_content.replace('{{%CONTENT%}}', productHtmlArray.join(',')) ;
        // res.writeHead(200, { 'Content-Type': 'application/json' }) // property
        res.writeHead(200, { 'Content-Type': 'text/html' })
        res.end(productResponse) ;
        // console.log(productHtmlArray.join(',')) ;
    } else {
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
```

```
})
server.listen(8000, '127.0.0.1', () => {
   console.log("Server has started!")
}) ;
```

Parsing Query String from URL

Creating a reusable function

```
function replaceHtml(template, product) {
   let output = template.replace('{{%IMAGE%}}', product.productImage) ;
   output = output.replace('{{%NAME%}}', product.name)
   output = output.replace('{{%MODELNAME%}}', product.modeName)
   output = output.replace('{{%MODELNO%}}', product.modelNumber)
   output = output.replace('{{%SIZE%}}', product.size)
   output = output.replace('{{%CAMERA%}}', product.camera)
   output = output.replace('{{%PRICE%}}', product.price)
   output = output.replace('{{%COLOR%}}', product.color)
   output = output.replace('{{%ID%}}', product.id)
   output = output.replace('{{%ROM%}}', product.ROM)
   output = output.replace('{{%DESC%}}', product.Description)

   return output ;
}

const server = http.createServer((req, res) => {
   // making alias of pathname to path
   let {query, pathname: path} = url.parse(req.url, true) ;
   // console.log(x)
   // let path = req.url ;
   if(path === '/' || path.toLocaleLowerCase() === '/home') {
      res.writeHead(200, { // property
         'Content-Type': 'text/html',
         'my-header' : 'Hello, World'
      })
      res.end(html_content.replace('{{%CONTENT%}}','You are in HOME page.'))
   } else if(path.toLocaleLowerCase() === '/about') {
      res.writeHead(200, { // property
         'Content-Type': 'text/html',
         'my-header' : 'Hello, World'
      })
      res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
```

```
    } else if(path.toLocaleLowerCase() === '/contact') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
    } // handling url routes
    else if(path.toLocaleLowerCase() === '/products') {
        if(!query.id) { // if no query remain at products page that contain all products
            let productHtmlArray = products.map((prod) => {
                return replaceHtml(product_list_html, prod) ;
            })
            let productResponse = html_content.replace('{{%CONTENT%}}', productHtmlArray.join(',')) ;
            res.writeHead(200, { 'Content-Type': 'text/html' })
            res.end(productResponse) ;
        } else { // if querry is there just display product no | product detail
            let prod = products[query.id]
            let productDetailsResponseHtml =  replaceHtml(productDetailsHtml, prod)
            res.end(html_content.replace('{{%CONTENT%}}', productDetailsResponseHtml))
        }
    } else {
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
})

from url.parse
Url {
          protocol: null,
          slashes: null,
          auth: null,
          host: null,
          port: null,
          hostname: null,
          hash: null,
          search: null,
          query: [Object: null prototype] {},   ****
          pathname: '/',      ***
          path: '/',
          href: '/'
}
```

Third Party modules

Creating a Custom Module (created by developer)
define a seprate module and define fun replaceHtml import and use it

```
const server = http.createServer((req, res) => {
   // making alias of pathname to path
   let {query, pathname: path} = url.parse(req.url, true) ;
   // console.log(x)
   // let path = req.url ;
   if(path === '/' || path.toLocaleLowerCase() === '/home') {
      res.writeHead(200, { // property
         'Content-Type': 'text/html',
         'my-header' : 'Hello, World'
      })
      res.end(html_content.replace('{{%CONTENT%}}','You are in HOME page.'))
   } else if(path.toLocaleLowerCase() === '/about') {
      res.writeHead(200, { // property
         'Content-Type': 'text/html',
         'my-header' : 'Hello, World'
      })
      res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
   } else if(path.toLocaleLowerCase() === '/contact') {
      res.writeHead(200, { // property
         'Content-Type': 'text/html',
         'my-header' : 'Hello, World'
      })
      res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
   } // handling url routes
   else if(path.toLocaleLowerCase() === '/products') {
      if(!query.id) { // if no query remain at products page that contain all products
         let productHtmlArray = products.map((prod) => {
            return replaceHtml(product_list_html, prod) ;
         })
         let productResponse = html_content.replace('{{%CONTENT%}}', productHtmlArray.join(',')) ;
         res.writeHead(200, { 'Content-Type': 'text/html' })
         res.end(productResponse) ;
      } else { // if querry is there just display product no | product detail
         let prod = products[query.id]
         let productDetailsResponseHtml =  replaceHtml(productDetailsHtml, prod)
         res.end(html_content.replace('{{%CONTENT%}}', productDetailsResponseHtml))
      }
   } else {
```

```
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
})
```

Understanding event driven architecture

Event Emitter ----[Emits event]---> Event Listener -----[calls Event Handler]----> Event Handler
[server] [on()] [f() {...}]

```
const server = http.createServer() ;
// server inherits from Event Emitter
server.on('request', (req, res) => {
    // making alias of pathname to path
    let {query, pathname: path} = url.parse(req.url, true) ;

    if(path === '/' || path.toLocaleLowerCase() === '/home') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}','You are in HOME page.'))
    } else if(path.toLocaleLowerCase() === '/about') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in ABOUT Page'))
    } else if(path.toLocaleLowerCase() === '/contact') {
        res.writeHead(200, { // property
            'Content-Type': 'text/html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'You are in CONTACT Page'))
    } // handling url routes
    else if(path.toLocaleLowerCase() === '/products') {
```

```
        if(!query.id) { // if no query remain at products page that contain all products
            let productHtmlArray = products.map((prod) => {
                return replaceHtml(product_list_html, prod) ;
            })
            let productResponse = html_content.replace('{{%CONTENT%}}', productHtmlArray.join(',')) ;
            res.writeHead(200, { 'Content-Type': 'text/html' })
            res.end(productResponse) ;
        } else { // if querry is there just display product no | product detail
            let prod = products[query.id]
            let productDetailsResponseHtml =  replaceHtml(productDetailsHtml, prod)
            res.end(html_content.replace('{{%CONTENT%}}', productDetailsResponseHtml))
        }
    } else {
        res.writeHead(404, { // property
            'Content-Type': 'text-html',
            'my-header' : 'Hello, World'
        })
        res.end(html_content.replace('{{%CONTENT%}}', 'ERROR 404: Page not Found'))
    }
})

server.listen(8000, '127.0.0.1', ()=> {
    console.log('listening to the request...')
})
```

Emitting & Handling Custom Events

how to create custom events in NODE JS and how to emit & handle these events.

```
let myEmitter = new events.EventEmitter() ;
myEmitter.on('userEvent', (id, name) => {
    console.log(`New user ${name} with ID ${id} is created! A`)
})
myEmitter.on('userEvent', () => {
    console.log("New user event is emited! B")
})
// emit an event
myEmitter.emit('userEvent', 101, 'John') ;
output (execution in order of derivation) ---->
New user event is emited! A
New user event is emited! B
Server has started!

let myEmitter = new user() ; // user class
myEmitter.on('userEvent', (id, name) => {
```

```
  console.log(`New user ${name} with ID ${id} is created! A`)
})
myEmitter.on('userEvent', () => {
  console.log("New user event is emited! B")
})
// emit an event
myEmitter.emit('userEvent', 101, 'John') ;
```

Understanding Streams in NODE JS

```
fs.readFile('source-file.txt', 'utf-8', (err, data) => {
  fs.write('dest-file.txt', data, () => {
    console.log('file written from source to destination!')
  })
})
```

with Streams, we can process data piece by piece instead of reading or writing whole data at once.
reading small chunk using it, freeing space eg(videos on youtube netflix)

Advantages --

1. streaming makes data processing more efficient in term of memory. Because there is no need to keep all data in the memory
2. in the terms of performance and time also, streaming has its advantage because we can start processing the data as the first chunk of data services

readable stream ---|
writable stream ---|
duplex stream
transform stream

solution 1. without stream

```
server.on('request', (req, res) => {
  fs.readFile("./files/large-file.txt", (err, data) => {
    if(err) {
      res.end("Something went wromg!") ;
      return ;
    }
    else {
      res.end(data) ;
    }
  })
})
```

solution 2. using readable and writable stream

```
server.on('request', (req, res) => {
  let rs = fs.createReadStream('./files/large-file.txt') ; // readable stream
```

```
    rs.on('data', (chunk) => {
        res.write(chunk) ; // always a writable stream
        // res.end() // signal that no more data need to be written..
        // dont use end here
    }) // all writing of data ends here goes to end event raised

    rs.on('end', () => {
        res.end() ;
    })

    rs.on('error', (error) => {
        res.end(error.message) ;
    })
})
```

recieving fast -- sending slow -- backpressure

solution 3. using pipe method

```
server.on('request', (req, res) => {
    let rs = fs.createReadStream('./files/large-file.txt') ; // readable stream
    rs.pipe(res) ;
    // fixes problem of backpressure
    // 2 line code

})

server.listen(8000, '127.0.0.1', () => {
    console.log("Server has started!")
}) ;
```

NPM (Node package manager)

for complete javascript (both frontend and backend)

1. regular dependency (express)
2. dev dependency (nodemon)

```
"dependencies": {
    "express": "^4.19.2",
    "nodemon": "^3.1.0"
}
```

Types of package install

1. Local
2. Global (eg. nodemon) [`npm i -g nodemon --save -dev `]

```
console.log("Nodemon is working...")
```

LIBUV (written in C++) --
The V8 engine converts JavaScript code into machine code, while libuv is responsible for handling asynchronous I/O and implementing features such as the event loop and thread pool.
1. event loop
2. thread pool

Process -
A process is what fascilitates the execution of the program .
Thread is responsible for executing a program code in the process. By default every process has one main thread.

Event Loop in NODE JS

```
console.log execute synchronously
console.log("Program has started!")

stored in 2nd phase
fs.readFile('./files/random.txt', () => {
   console.log("File read completed!")

   // stored in 1st phase
   setTimeout(() => {
      console.log("Timer callback executed!")
   }, 0)

   stored in 3rd phase
   setImmediate(() => {
      console.log("SetImmediate callback executed!")
   })

   immediately after the execution of current phase
   process.nextTick(() => {
      console.log("Process.nextTick callback executed")
   })
})

console.log("Program has completed!")
output ----
Program has started!
Program has completed!
File read completed!
Process.nextTick callback executed
SetImmediate callback executed!
```

Timer callback executed!

Introduction to Express JS | Working with Express JS

free and open source web application framework for node.js

1. completely build on node.js
2. one of most popular framework for node.js
3. express contains very robust and useful set of features
4. allows to write node js application faster and simpler (predefined methods)
5. with express we can organize node js code in mvc architecture