Node JS

What is Node.js?

- JavaScript runtime built on V8 javascript engine.
- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database
- Anytype of application can be built using Javascript on Node js.
- Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

What is Node.js?

When to use?

- I/O bound application, Data streaming applications eg. facebook, Real time chat application eg. Whatsapp.

When not to use?

- CPU intensive applications.

Task of WebServer

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing and delivering webpages to users. A common task for a web server can be to open a file on the server and return the content to the client.

File request handled by PHP / ASP:

- Sends the task to the computer's file system.
- Waits while the file system opens and reads the file.
- Returns the content to the client.
- Ready to handle the next request.

Task of Node.js

How Node.js handles a file request:

- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.
- Node.js eliminates the waiting, and simply continues with the next request.

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node js files have extension ".js“
- To download, visit

https://nodejs.org

"Hello World" in a web browser

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

- The code tells the computer to write "Hello World!" if anyone (e.g. a web browser) tries to access your computer on port 8080.
- Save the file hello.js and write "cmd" in the search field.
- Navigate to the folder that contains the file “hello.js", the command line interface window should look something like this:
- C:\Users\Your Name>node hello.js
- Now, your computer works as a server!
- If anyone tries to access your computer on port 8080, they will get a "Hello World!" message in return.
- To check on your browser, type in the address: http://localhost:8080

Node Package Manager (NPM)

- NPM is a package manager for Node.js packages.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js

To install third party modules

npm install <package unique name>

Example: npm install upper-case

Node.js “require”

- The modules or packages available can be imported in .js file using “require” function.
- Example: var http = require(“http”);
- Now your application has access to the HTTP module, and is able to create a server.

Text Courtesy: https://www.w3schools.com/html/html_attributes.asp, https://www.techrepublic.com/article/basic-html-rules/, https://www.w3schools.com/html/html5_syntax.asp

Built-in Modules

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols
tty	Provides classes used by a text terminal
url	To parse URL strings
util	To access utility functions
v8	To access information about V8 (the JavaScript engine)
vm	To compile JavaScript code in a virtual machine
zlib	To compress or decompress files

Create Your Own Modules

- Exports keyword to make properties and methods available outside the module file.
- The following example creates a module that returns a date and time object:

```
exports.myDateTime = function () {
  return Date();
};
```

- Save the code in a file called "mymodule.js“

```
var http = require('http');
var dt = require('./mymodule');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

- Save the code in a file called "demo_module.js", and initiate the file:
- C:\Users\Your Name>node demo_module.js

The Built-in HTTP Module

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
- Node.js as a Web Server
- The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.
- Use the createServer() method to create an HTTP server:

```
var http = require('http');

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080);  //the server object listens on port 8080
```

Node.js as a File System

- The Node.js file system module allows you to work with the file system on your computer.

```
var fs = require('fs'); // import File system module
```

Common use for the File System module:

- Read files
- Create files
- Update files
- Delete files
- Rename files

Read Files

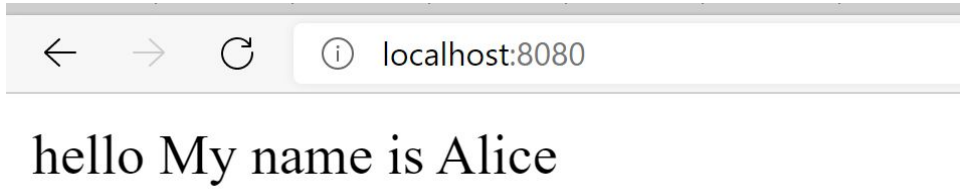
- The fs.readFile() method is used to read files on your computer.
- Assume we have the following txt file (located in the same folder as Node.js): demo.txt and content of the file us given below

Hello,

My name is Alice

Read Files

- Create a Node.js file that reads the HTML file, and return the content
-
```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('demo.txt', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```



- Save the code above in a file called "demo_readfile.js", and initiate the file

Create Files

- The File System module has methods for creating new files
- fs.appendFile(), fs.open(), fs.writeFile()
- The **fs.appendFile()** method appends specified content to a file. If the file does not exist, the file will be created:

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

fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {
  if (err) throw err;
  console.log('Saved!'); });
```

- The **fs.open()** method takes a "flag" as the second argument, if the flag is "w" for "writing", the specified file is opened for writing. If the file does not exist, an empty file is created:
- The **fs.writeFile()** method replaces the specified file and content if it exists. If the file does not exist, a new file, containing the specified content, will be created:

Delete Files

- To delete a file with the File System module, use the fs.unlink() method.
- The fs.unlink() method deletes the specified file:

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

fs.unlink('mynewfile2.txt', function (err) {
  if (err) throw err;
  console.log('File deleted!');
});
```

Rename Files

- To rename a file with the File System module, use the fs.rename() method.
- The fs.rename() method renames the specified file:
- Rename "mynewfile1.txt" to "myrenamedfile.txt":

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

fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed!');
});
```

URL Module

- The URL module splits up a web address into readable parts.
- Parse an address with the url.parse() method, and it will return a URL object with each part of the address as properties:
-
```
var url = require('url');
var adr = 'http://localhost:8080/default.htm?year=2017&month=february';
var q = url.parse(adr, true);
console.log(q.host); //returns 'localhost:8080'
console.log(q.pathname); //returns '/default.htm'
console.log(q.search); //returns '?year=2017&month=february'

var qdata = q.query; //returns an object: { year: 2017, month: 'february' }
console.log(qdata.month); //returns 'february'
```

URL Module

- Combine the URL module and File system module to serve the file requested by the client.

Create two html files and save them in the same folder as your node.js files.

Alice.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Alice</h1>

</body>
</html>
```

Bob.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Bob</h1>

</body>
</html>
```

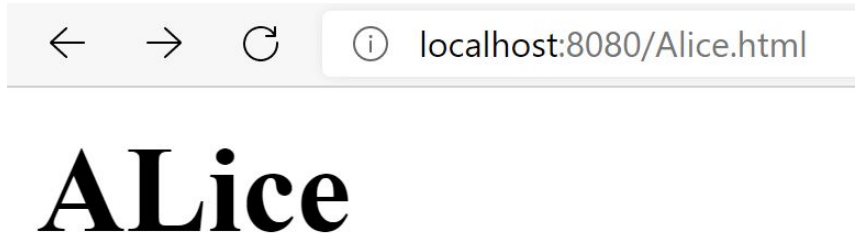
URL Module

Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error:

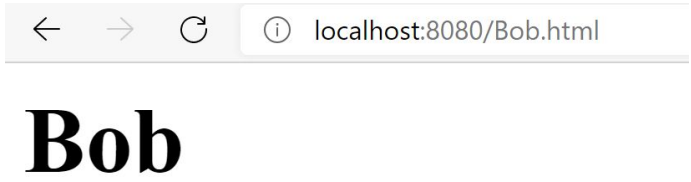
```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

URL Module

http://localhost:8080/Alice.html



http://localhost:8080/Bob.html



Useful Reference Links

- https://www.w3schools.com/nodejs/default.asp
- https://nodejs.dev/learn