Node.js

Node.js is an open source server environment. ***Node.js allows you to run JavaScript on the server.***

Displaying the result in the command line interface. It will show the result in a black screen on the right:

```
console.log('This example is different!');

console.log('The result is displayed in the Command Line Interface');
```

Download Node.js

The official Node.js website has installation instructions for Node.js: https://nodejs.org

Getting Started

Once you have downloaded and installed Node.js on your computer, let's try to display "Hello World" in a web browser.

Create a Node.js file named "myfirst.js", and add the following code:

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

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

Save the file on your computer: C:\Users\Your Name\myfirst.js

Command Line Interface

Node.js files must be ***initiated*** in the "Command Line Interface" program of your computer.

Navigate to the folder that contains the file "myfirst.js"

C:\Users\Your Name**>node myfirst.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!

Start your internet browser, and type in the address: http://localhost:8080

What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries. A set of functions you want to include in your application.

Include Modules

To include a module, use the **require()** function with the name of the module:

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

Now your application has access to the HTTP module, and is able to create a server:

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

Create Your Own Modules

You can create your own modules, and easily include them in your applications.

The following example creates a module that returns a date and time object:

Example

Create a module that returns the current date and time:

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

Use the **exports** keyword to make properties and methods available outside the module file.

Save the code above in a file called "myfirstmodule.js"

Include Your Own Module

Now you can include and use the module in any of your Node.js files.

Example

Use the module "myfirstmodule" in a Node.js file:

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

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);
```

Notice that we use ./ to locate the module, that means that the module is located in the same folder as the Node.js file.

Save the code above in a file called "demo_module.js", and initiate the file:

Initiate demo_module.js:

C:\Users\Your Name>**node demo_module.js**

If you have followed the same steps on your computer, you will see the same result as the example: http://localhost:8080

The Built-in URL Module

The URL module splits up a web address into readable parts. To include the URL module, use the require() method:

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

Parse an address with the url.parse() method, and it will return a URL object with each part of the address as properties:

Split a web address into readable parts:

```
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'
```

What is NPM?

NPM is a package manager for Node.js packages, or modules if you like. 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

Download a Package

Open the command line interface and tell NPM to download the package you want.

I want to download a package called "upper-case":

Download "upper-case":

C:\Users\Your Name>**npm install upper-case**

NPM creates a **folder named "node_modules**", where the package will be placed. All packages you install in the future will be placed in this folder.

Using a Package

Include the "upper-case" package the same way you include any other module:

```
var uc = require('upper-case');
```

Create a Node.js file that will convert the output "Hello World!" into upper-case letters:

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

var uc = require('upper-case');

http.createServer(function (req, res) {

 res.writeHead(200, {'Content-Type': 'text/html'});

 res.write(uc.upperCase("Hello World!"));

 res.end();

}).listen(8080);
```

Send an Email

The Nodemailer module makes it easy to send emails from your computer. The Nodemailer module can be downloaded and installed using npm:

C:\Users\Your Name>**npm install nodemailer**

Now you can include the module in any application:

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

Use the username and password from your selected email provider to send an email. This tutorial will show you how to use your Gmail account to send an email:

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

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'youremail@gmail.com',
    pass: 'yourpassword'
  }
});

var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
};

transporter.sendMail(mailOptions, function(error, info){
  if (error) {
    console.log(error);
  } else {
    console.log('Email sent: ' + info.response);
```

```
  }
});
```

Multiple Receivers

To send an email to more than one receiver, add them to the "to" property of the mailOptions object, separated by commas:

```
var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com, myotherfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
}
```

Send HTML

To send HTML formatted text in your email, use the "**html**" property instead of the "**text**" property:

```
var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  html: '<h1>Welcome</h1><p>That was easy!</p>'
}
```

MySQL

Node.js can be used in database applications. One of the most popular databases is MySQL.

Install MySQL Driver

Once you have MySQL up and running on your computer, you can access it by using Node.js.

To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.

To download and install the "mysql" module, open the Command Terminal and execute the following:

C:\Users\Your Name>**npm install mysql**

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

Create Connection

Start by creating a connection to the database. Use the username and password from your MySQL database.

demo_db_connection.js

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
```

```
  console.log("Connected!");
});
```

Save the code above in a file called "*demo_db_connection.js*" and run the file:

Run "demo_db_connection.js"

C:\Users\Your Name>**node demo_db_connection.js**

Which will give you this result:

Connected!

The connection object created in the example above, has a method for querying the database:

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```