Node.js and Express

Paul Fodor

CSE316: Fundamentals of Software Development

Stony Brook University

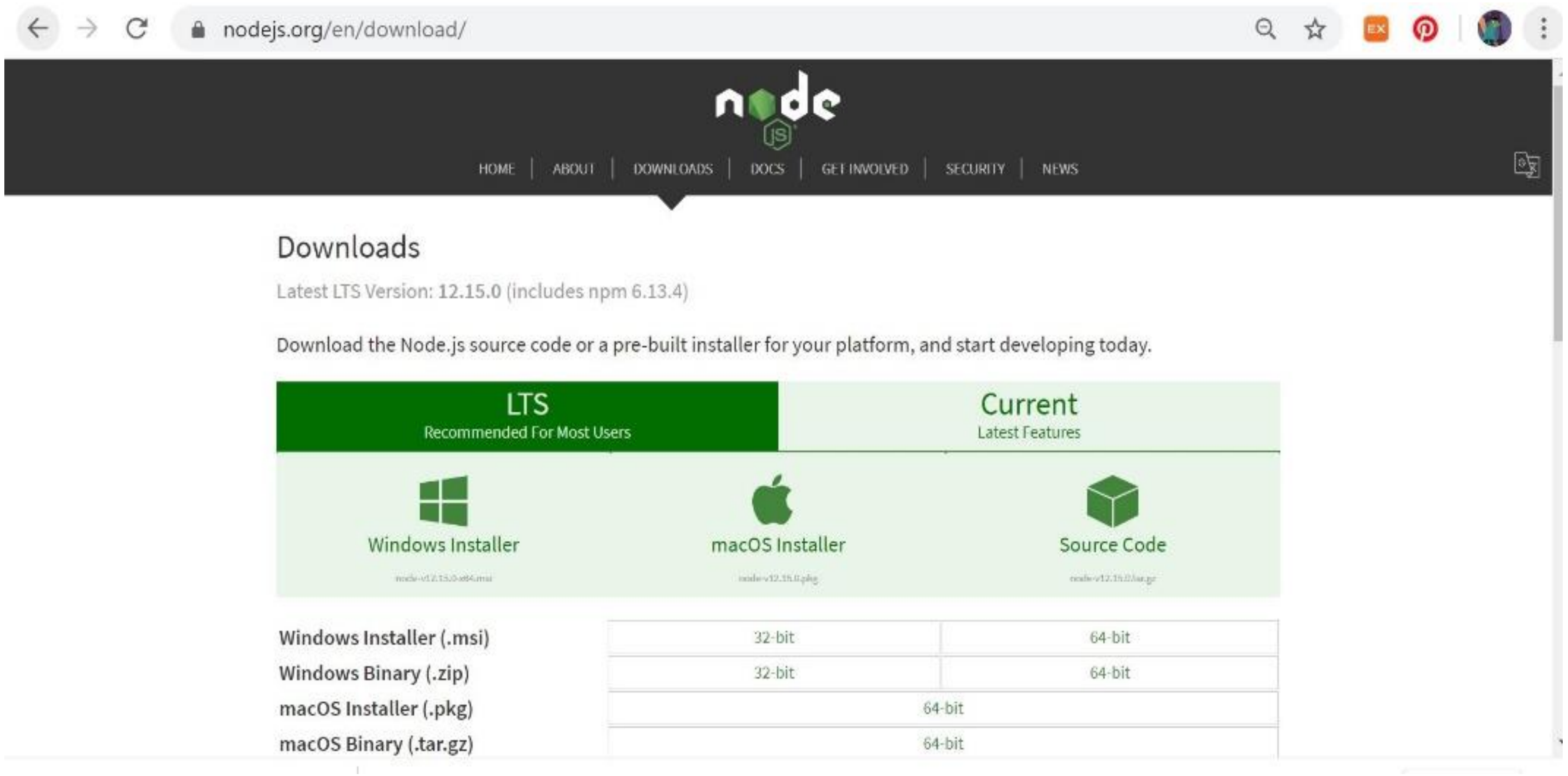
http://www.cs.stonybrook.edu/~cse316

Node.js

- Node.js (Original author: Ryan Dahl, 2009-2012)
  - Open Source Server Environment
    - Lets developers use JavaScript to write command line tools and server-side scripting
  - Runs on various platforms (MacOS, Windows, Linux/Unix, etc)
  - Based on Chome v8 engine
    - Written in C++, V8 compiles JavaScript source code to native machine code at runtime
    - As of 2016, it also includes Ignition, a bytecode interpreter.
  - Stable release: 14.11.0 / September 15, 2020; 4 days ago
- Capabilities
  - Generates dynamic page content [like PHP]
  - Create, open, read, write, delete files on a server [like PHP]
  - Collect form data
  - Add, modify, and delete data to/from a database

Installing Node JS

- Navigate to: https://nodejs.org/en/download/
  - Select either the 64-bit or 32-bit installer (depending on your machine's architecture)


nodejs.org/en/download/
node JS
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS

Downloads

Latest LTS Version: 12.15.0 (includes npm 6.13.4)

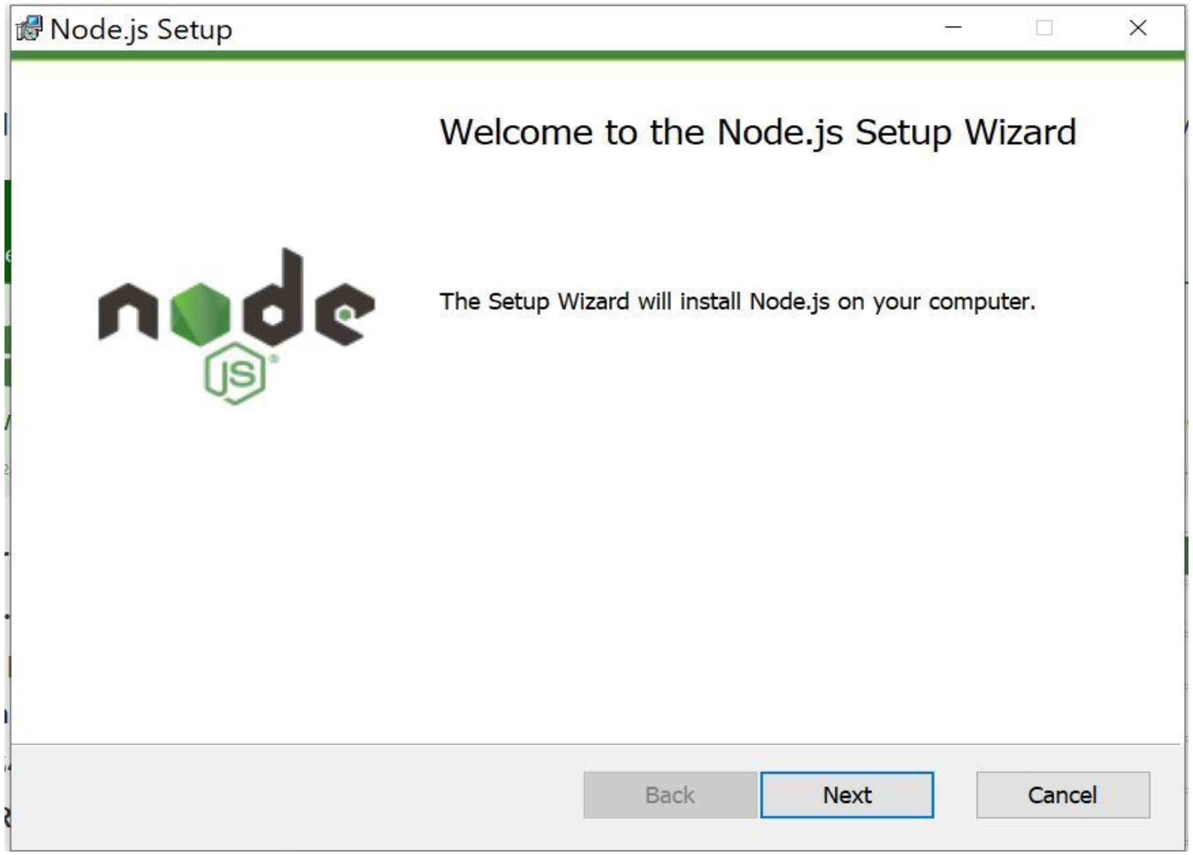
Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users		Current Latest Features
Windows Installer node-v12.15.0-x64.msi	macOS Installer node-v12.15.0.pkg	Source Code node-v12.15.0.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	

Installing Node JS

- Install



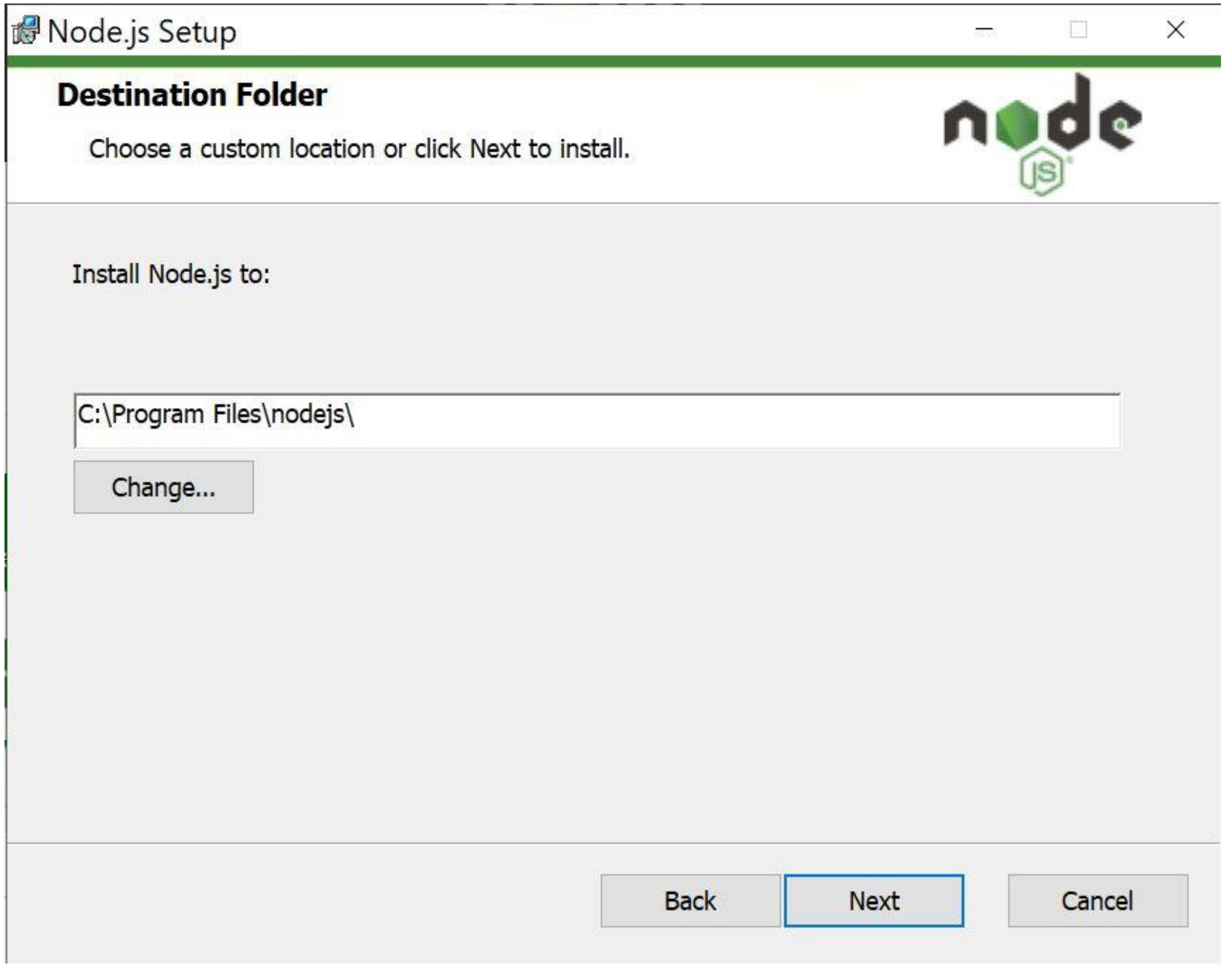
Node.js Setup

Welcome to the Node.js Setup Wizard

The Setup Wizard will install Node.js on your computer.

Installing Node JS

- Install



Node.js Setup

Destination Folder

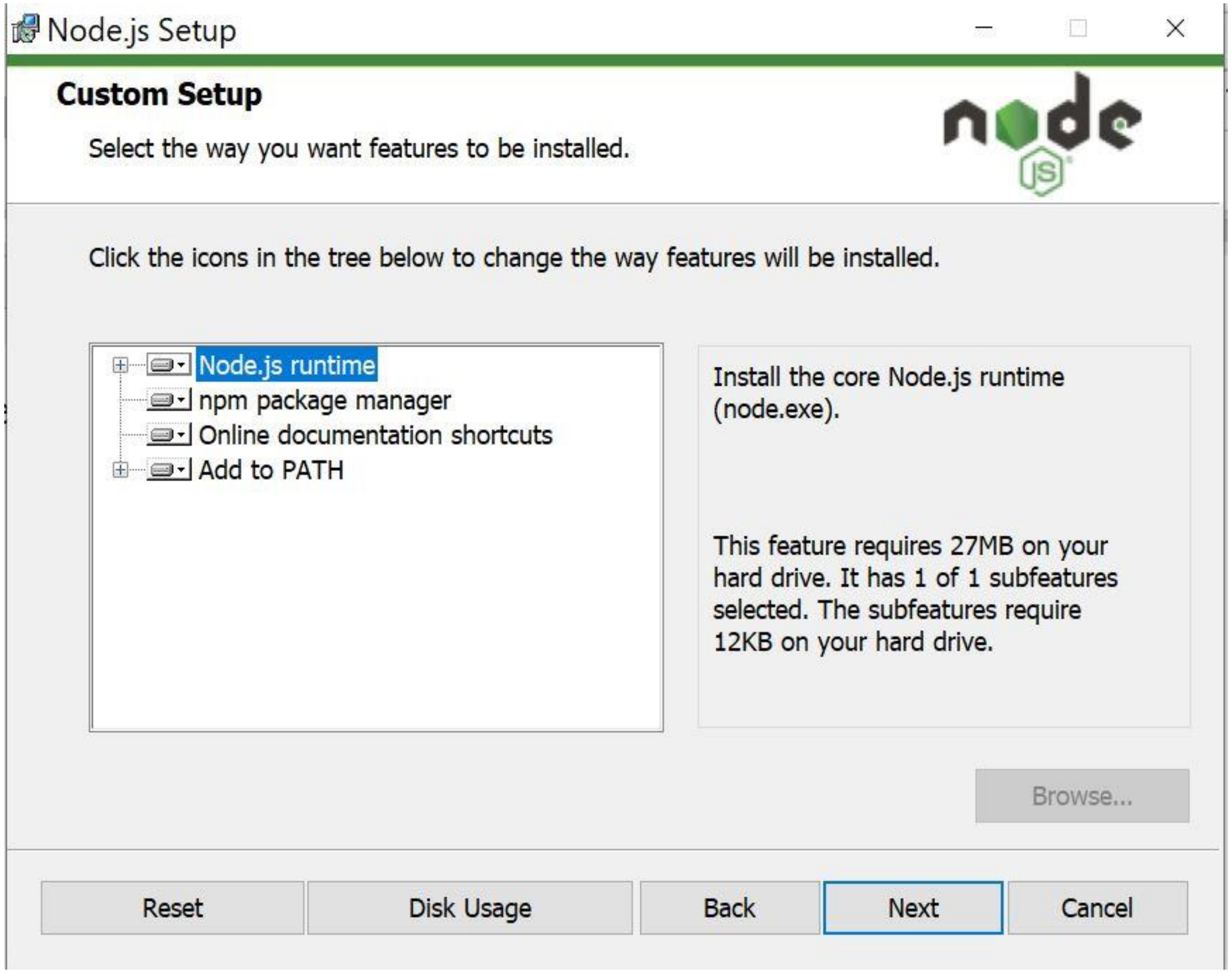
Choose a custom location or click Next to install.

Install Node.js to:

C:\Program Files\nodejs\

Installing Node JS

- Install



Node.js Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

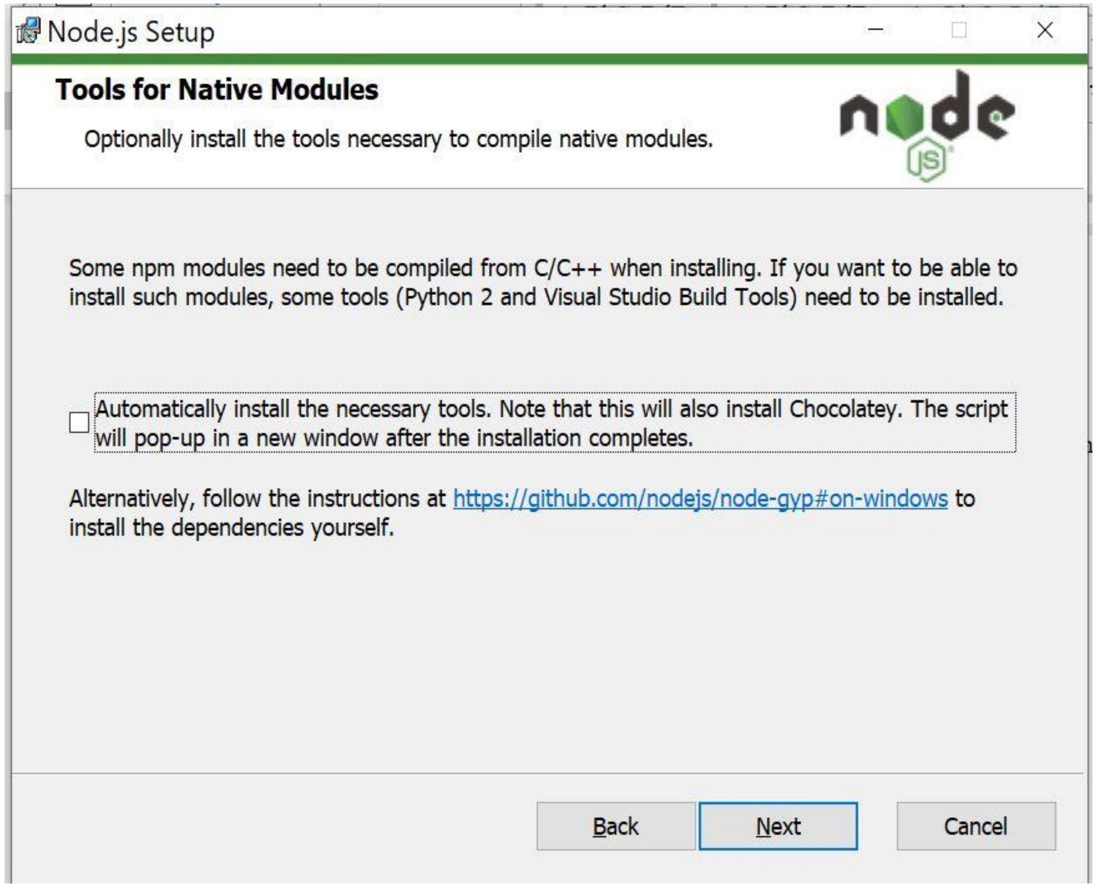
- Node.js runtime
- npm package manager
- Online documentation shortcuts
- Add to PATH

Install the core Node.js runtime (node.exe).

This feature requires 27MB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 12KB on your hard drive.

Installing Node JS

- You shouldn’t need the Native Modules so don’t bother clicking the checkbox. Click ‘Next’.



Tools for Native Modules

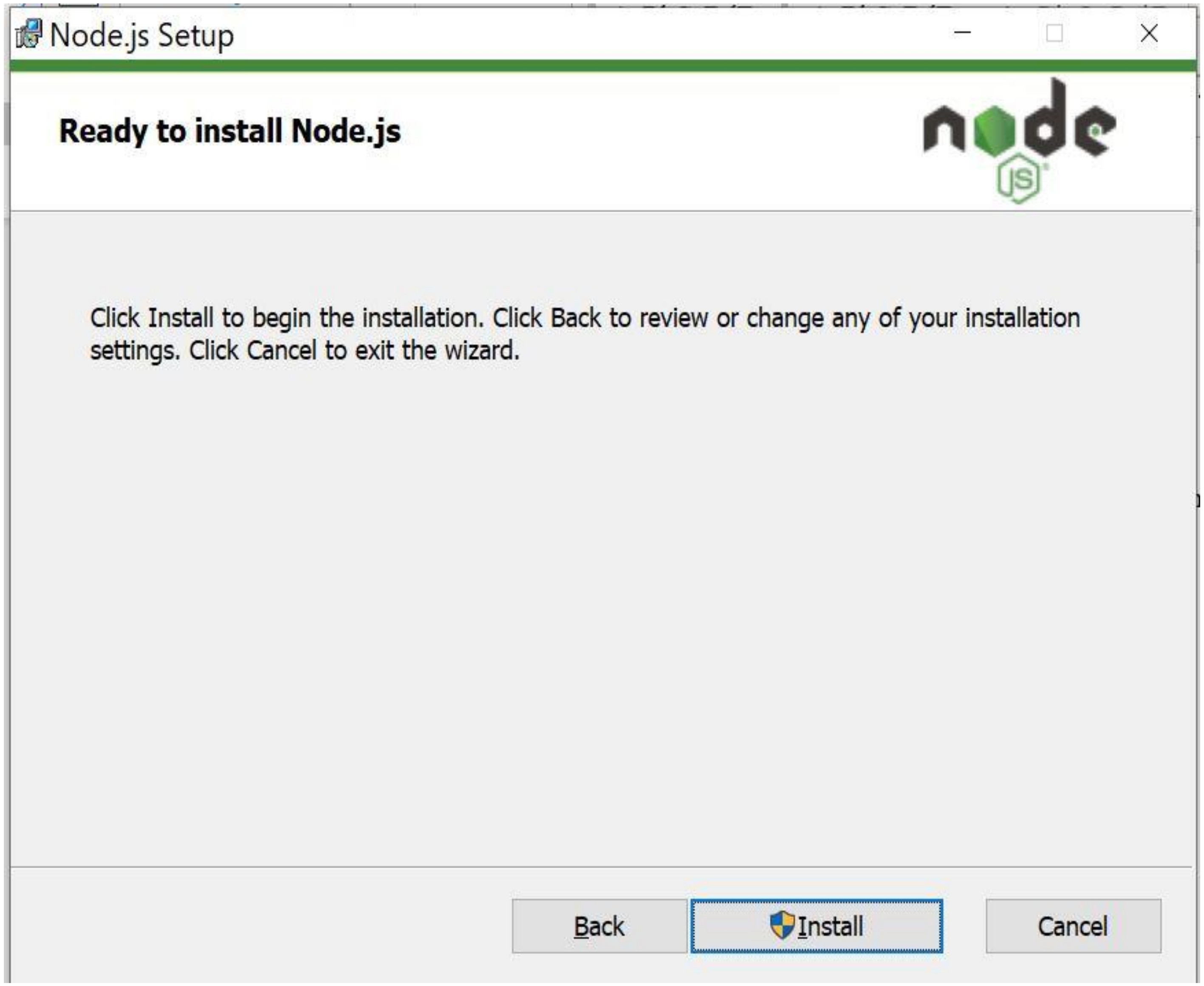
Optionally install the tools necessary to compile native modules.

Some npm modules need to be compiled from C/C++ when installing. If you want to be able to install such modules, some tools (Python 2 and Visual Studio Build Tools) need to be installed.

☐ Automatically install the necessary tools. Note that this will also install Chocolatey. The script will pop-up in a new window after the installation completes.

Alternatively, follow the instructions at https://github.com/nodejs/node-gyp#on-windows to install the dependencies yourself.

Installing Node JS

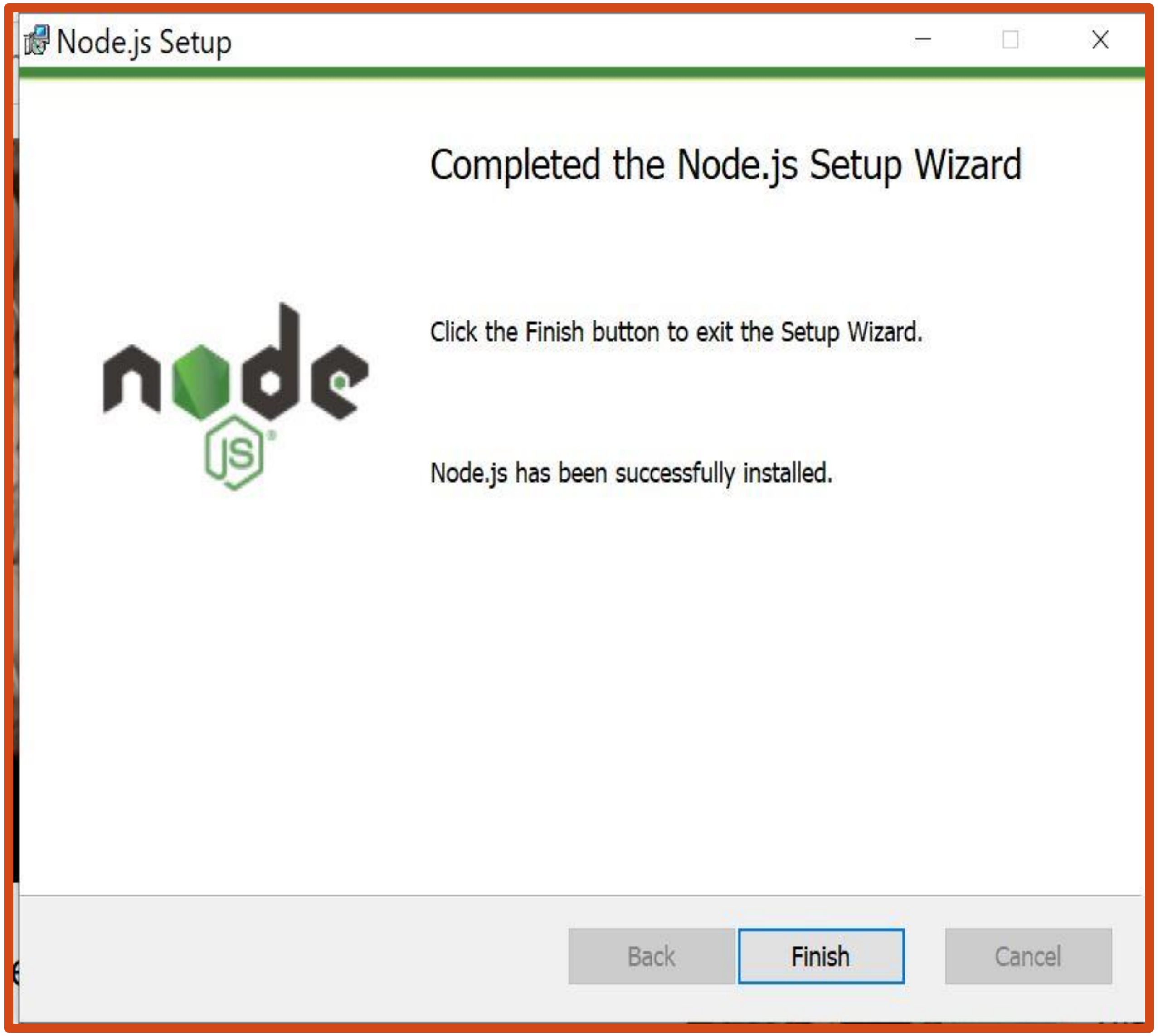


Node.js Setup

Ready to install Node.js

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Installing Node JS



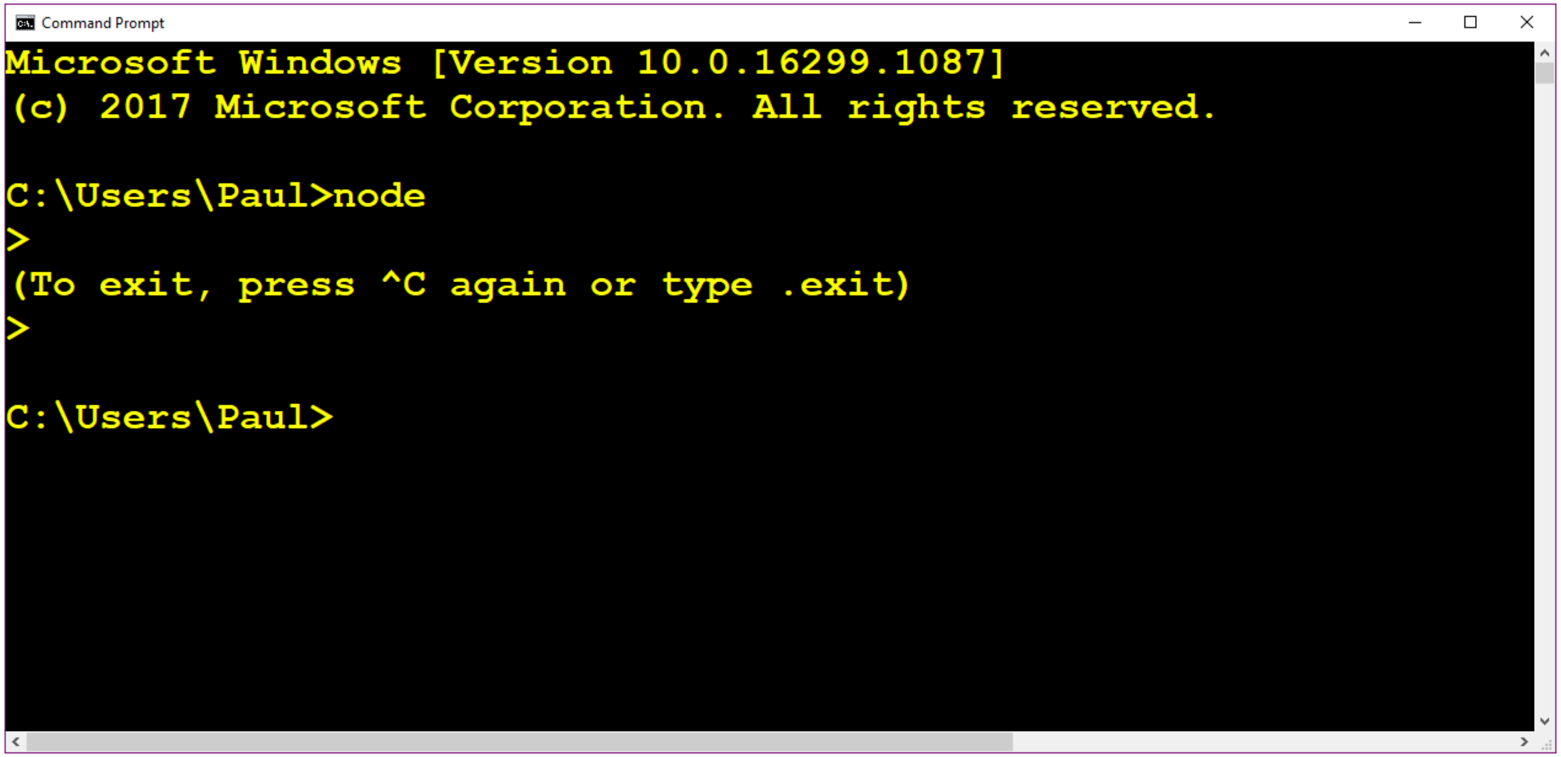
Node.js Setup

Completed the Node.js Setup Wizard

Click the Finish button to exit the Setup Wizard.

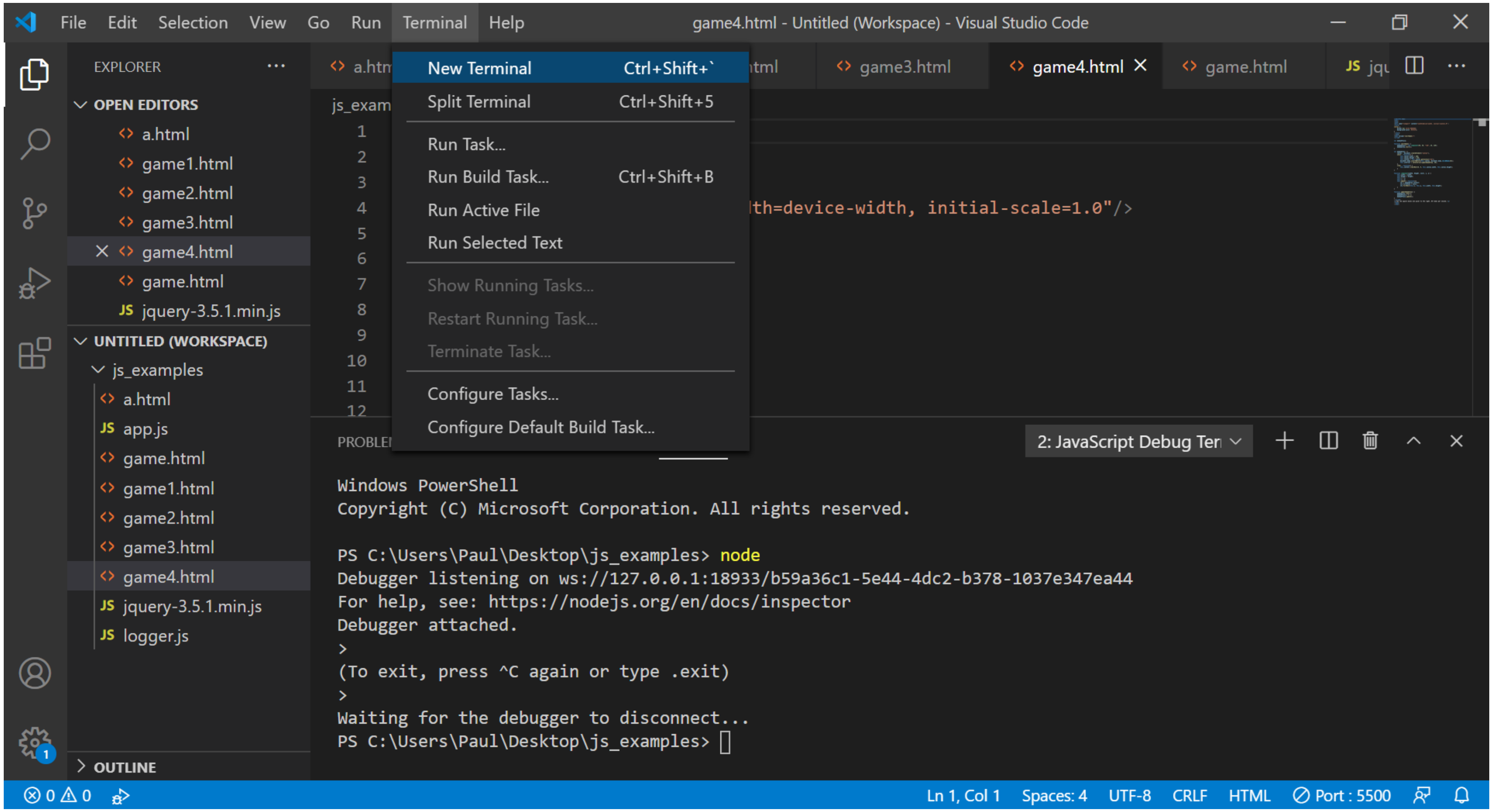
Node.js has been successfully installed.

Installing Node JS



```
Microsoft Windows [Version 10.0.16299.1087]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Paul>node
>
(To exit, press ^C again or type .exit)
>

C:\Users\Paul>
```

Terminal in VSCode



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Paul\Desktop\js_examples> node
Debugger listening on ws://127.0.0.1:18933/b59a36c1-5e44-4dc2-b378-1037e347ea44
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
>
(To exit, press ^C again or type .exit)
>
Waiting for the debugger to disconnect...
PS C:\Users\Paul\Desktop\js_examples>
```

Node.js vs Javascript

- There is no 'window' or 'document' object
- Node.js has a 'global' scope
  - Global objects [i.e. setTimeout(), clearTimeout, setInterval(), clearInterval()]
  - Inside the 'window' object in Javascript
    - setInterval() is equivalent to window.setInterval()
  - Inside the global scope in Node.js
    - setInterval() is equivalent to global.setInterval()

Concept of Modules

- Each file is a module
- Functions and variables inside of files need to be 'exported' to be available in other files.

logger.js

```
var url = 'http://www.cs.stonybrook.edu';

function log(message) {
  console.log(message)
}

module.exports.log = log;

module.exports.endpoint = url;
```

app.js

```
var logger = require('./logger')

logger.log('Hi, Paul')

logger.log(logger.endpoint)
```

```
> node app.js
Hi, Paul
http://www.cs.stonybrook.edu
```

Node.js - Modules

- Node.js has a large number (thousands) of built-in modules that perform common tasks. A few are:
  - dns – Handle dns queries
  - crypto – Perform cryptographic operations with OpenSSL
  - dgram – Implements UDP datagram sockets
  - event – Implements events for server side Javascript
  - fs – Supports filesystem operations
  - http – Allows Node.js to act as an http server
  - https – Allows Node.js to act as a secure http server
  - os – Provides information about the operating system
  - path – Handles file paths
  - querystring – Handles URL query strings
  - url – Parses URL strings

Node.js – Simple Example

Outputs a single page when a connection is made.

demo_server.js:

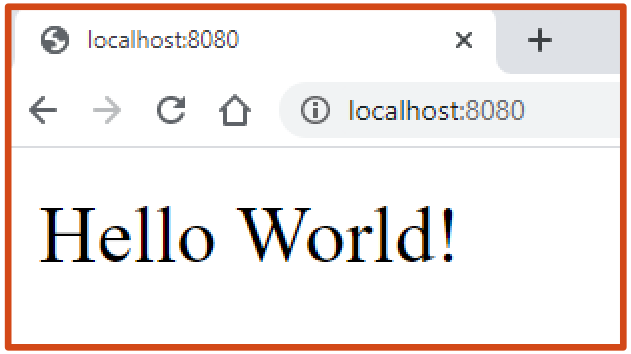
```
const http = require('http');

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

Creates a server listening on port 8080

Start Node.js by typing on a console:

```
node demo_server.js
```

Open localhost:8080 in the browser



Hello World!

Node.js - Modules

- Access functionality of a module using require

  var http = require('http');

  Loads the http built in module

Node.js – HTTP Module

```
const http = require('http');
http.createServer(function (req, res) {
    //code for server
})
```

createServer creates a Web server

arguments:

- req – An object holding the incoming request (full URL and query string that can be parsed)
- res – An object to collect data for the response.

Node.js – HTTP Module

res.write()	Writes in the response.
res.end()	End the response process.
res.json()	Send a JSON response.

res.download() sends a file back.

Node.js – URL Module

- url supports parsing of URL strings
  - req object in the createServer() function contains the URL in the member .url
  - req.headers.host is the hostname and port from the URL string

var q = url.parse(req.url, true)

- q.pathname – This is the path from the 1st slash (/) through the end of the file path
- q.search – This holdes the search parameters from the url querystring
- q.query – returns a structure with fields for each query parameters

Node.js – URL Module

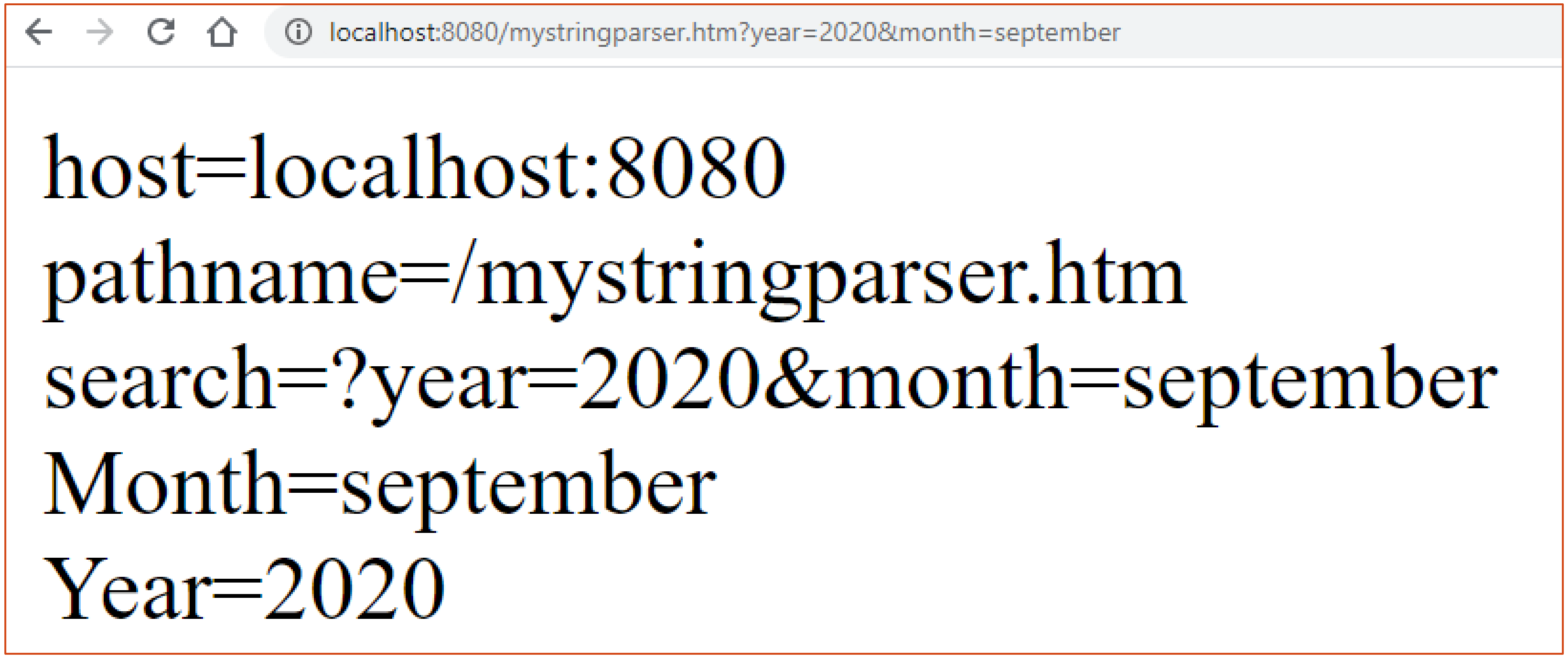
- demo_server2.js:

```
const http = require('http');
const url = require('url');
http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    var q = url.parse(req.url, true)
    res.write("host=" + req.headers.host
      + "<br>pathname=" + q.pathname
      + "<br>search=" + q.search);
    var qdata = q.query;
    res.write('<br>Month=' + qdata.month);
    res.write('<br>Year=' + qdata.year);
    res.end();
}).listen(8080);
```

Node.js – URL Module

- Given the URL :

  http://localhost:8080/mystringparser.htm?year=2020&month=september



```
host=localhost:8080
pathname=/mystringparser.htm
search=?year=2020&month=september
Month=september
Year=2020
```

Node.js - Filesystem

- Node.js provides support for reading, writing, creating, and deleting files
- Need to require module 'fs'

Node.js – Filesystem

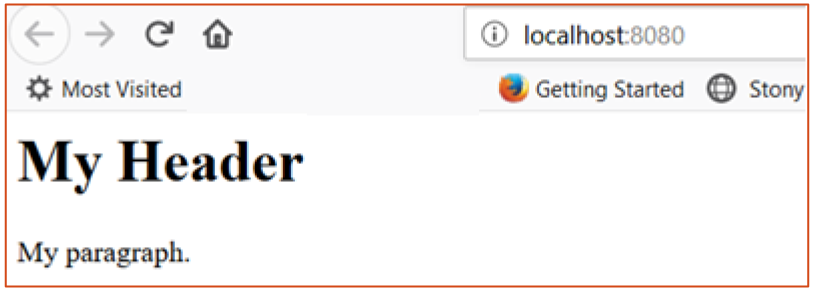
demo_server3.js:

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('demofile1.html', function(err, data) {
      res.writeHead(200, {'Content-Type': 'text/html'});
      res.write(data);
      res.end();
        });
}).listen(8080);
```

Read demofile1.html and send it as the response of type text/html

demofile1.html:

```
<html>
  <body>
    <h1>My Header</h1>
    <p>My paragraph.</p>
  </body>
</html>
```



My Header

My paragraph.

Node.js - npm

- npm is the Node Package Manager
- It is part of the Node.js installation
  - Used to install supplemental software packages

```
npm install jquery
```

  - Supplemental packages are placed in subfolder **<u>node-packages</u>**
  - Installed packages can be used in applications by using **require()**

Node.js Events

- The event module supports emitting and catching events: emit() and on()
- Often, the Event class is subclassed by an app to add functionality to the emit() and on() mechanisms.

Node.js - Events

- Handling of events:

```
var fs = require('fs');
var rs = fs.createReadStream('./demofile1.html');
rs.on('open', function () {
  console.log('The file is open');
});
```

Displays console message when the 'open' event fires for the file.

Node.js - Events

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

var rs = fs.createReadStream('./demofile1.html');
rs.on('open', function () {
  console.log('The file is open');
});

http.createServer(function (req, res) {
  fs.readFile('demofile1.html', function(err, data) {
      res.writeHead(200, {'Content-Type': 'text/html'});
      res.write(data);
      res.end();
        });
}).listen(8080);
```

```
The file is open
```

Express

- Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License.
  - Initial release: November 16, 2010

Express - setup

- Once node is installed, from a terminal window, type:

```
npm install express
```

Express – Simple example

Create a file like index.js or app.js in the directory you created

Add the following:

```
const express = require('express')
const app = express()
```

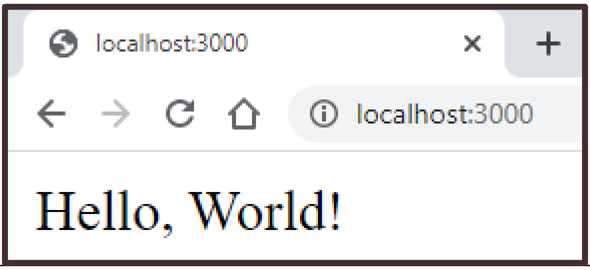
This creates the express application

```
app.get('/', (req,res) => {
  res.send('Hello, World!');
});
// … Other 'routes' go here'
```

This is a route to handle a 'get' request.

```
port = process.env.PORT || 3000
app.listen(port, () => { console.log('server started!')});
```

This starts the server on port 3000



Hello, World!

Setting up and using Routes

- Routes indicate the path on the website and the code associated with that 'page'.
- Express supports all the HTTP 'methods' for a web interface -> In REST you have:
  - GET - retrieve resource
  - POST - create new resource
  - PUT - update existing resource
  - DELETE - delete resource

General Format

- app.<method>(<path>, <handlerFunction>)

```
app.get('/', (req, res) => {
        // Code to handle and respond to the 'get' request.
        // req is the request object and has info about the url, body ,etc
        // res is the respons object where you build the information to
        //   return to the requester
    }
```

- The '<path>' can include parameters for the request
  - Precede field with ':'

Handling Get requests

- Get a request to fetch an array of values
  - An array holds 'member' information with id, name, email, and status
    - the array is hard coded (a real app puts this in a database)

```
const members = [
 {
   id: 1,
   name: "Paul Fodor",
   email: "pfodor@cs.stonybrook.edu",
   status: "active"
 },
...
]
```

Handling Get requests

This is the 'route' or path to the operation

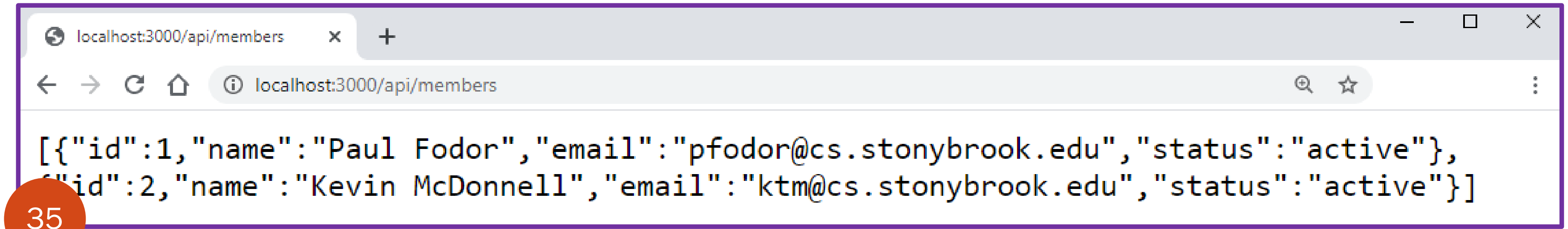
```
app.get("/api/members", (req, res) => {
 res.json(members);
});
```

This formats into json the array and returns it as part of the response.

app2.js:

```
const express = require('express')
const app = express()
const members = [
   {
      id: 1,
      name: "Paul Fodor",
      email: "pfodor@cs.stonybrook.edu",
      status: "active"
   },
   {
      id: 2,
      name: "Kevin McDonnell",
      email: "ktm@cs.stonybrook.edu",
      status: "active"
   }
]
```

```
app.get("/api/members", (req, res) => {
   res.json(members);
 });
port = process.env.PORT || 3000
app.listen(port, () => {
 console.log('server started!')
});
```



```
[{"id":1,"name":"Paul Fodor","email":"pfodor@cs.stonybrook.edu","status":"active"},
{"id":2,"name":"Kevin McDonnell","email":"ktm@cs.stonybrook.edu","status":"active"}]
```

Handling Get requests

- Return 1 value from the array based on the 'id'
- This needs a parameter which is specified with ':id'

The path or route now includes the parameter ***id***

```
app.get("/api/members/:id", (req, res) => {
  const found = members.some(member => member.id === parseInt(req.params.id));

  if (found) {
   res.json(members.filter(member => member.id === parseInt(req.params.id)));
  } else {
   res
    .status(400)
    .json({ msg: `No member with the id ${req.params.id} was found!` });
  }
});
```

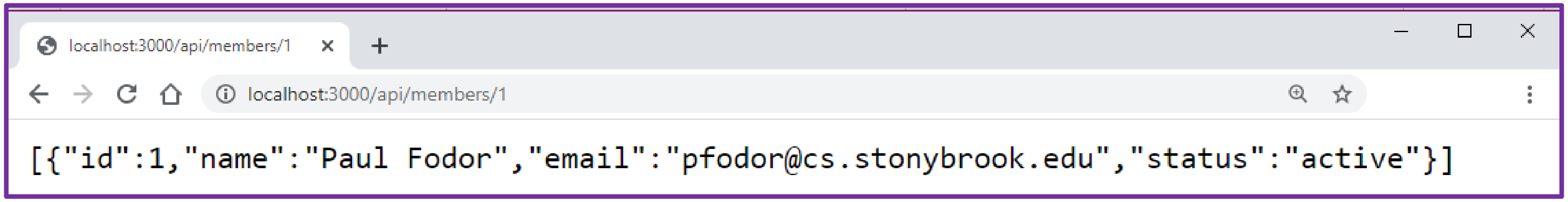
If the id matches at least one, we use 'filter' to extract that record. We return it in json format.

If no id matches, we return an error message with a status of 400 (bad request)

app3.js:

```
const express = require('express')
const app = express()
const members = [
   {
      id: 1,
      name: "Paul Fodor",
      email: "pfodor@cs.stonybrook.edu",
      status: "active"
   },
   {
      id: 2,
      name: "Kevin McDonnell",
      email: "ktm@cs.stonybrook.edu",
      status: "active"
   }
]

console.log(members.some(member => member.id === 1))
```

```
app.get("/api/members/:id", (req, res) => {
   const found = members.some(member => member.id === parseInt(req.params.id));
   if (found) {
     res.json(members.filter(member => member.id === parseInt(req.params.id)));
   } else {
     res
       .status(400)
       .json({ msg: `No member with the id ${req.params.id} was found!` });
   }
});
port = process.env.PORT || 3000
app.listen(port, () => { console.log('server started!') });
```



```
[{"id":1,"name":"Paul Fodor","email":"pfodor@cs.stonybrook.edu","status":"active"}]
```

More than get requests

```
app.route('/api/members/')
  .get(function (req, res) {
   res.send('Get a random member')
  })
  .post(function (req, res) {
   res.send('Add a member')
  })
  .put(function (req, res) {
   res.send('Update a member')
  })
```

express.Router

- Use the express.Router class to create modular, mountable route handlers.
  - Create a router file named members.js in the app directory:

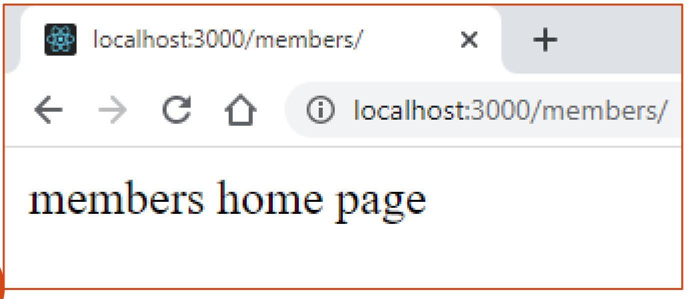
```
var express = require('express')
var router = express.Router()
router.get('/', function (req, res) {
  res.send('members home page')
})
router.get('/about', function (req, res) {
  res.send('About members')
})
module.exports = router
```

express.Router

- Then, load the router module in the app.js:

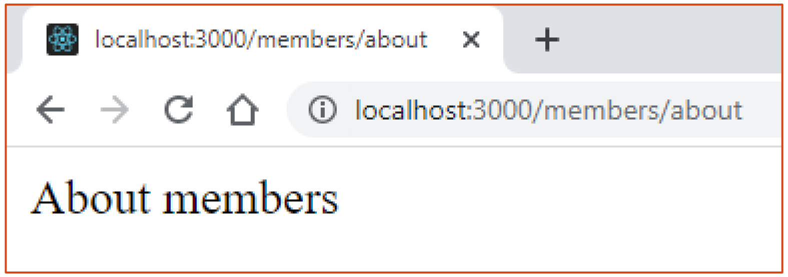
```
var express = require('express')
var app = express()
var members = require('./members.js')
app.use('/members', members)
port = process.env.PORT || 3000
app.listen(port, () => { console.log('server started!') });
```

- The app will now be able to handle requests to /members and /members/about.


localhost:3000/members/

localhost:3000/members/

members home page


localhost:3000/members/about

localhost:3000/members/about

About members