

INLS 672

Web Development 2

Node.js

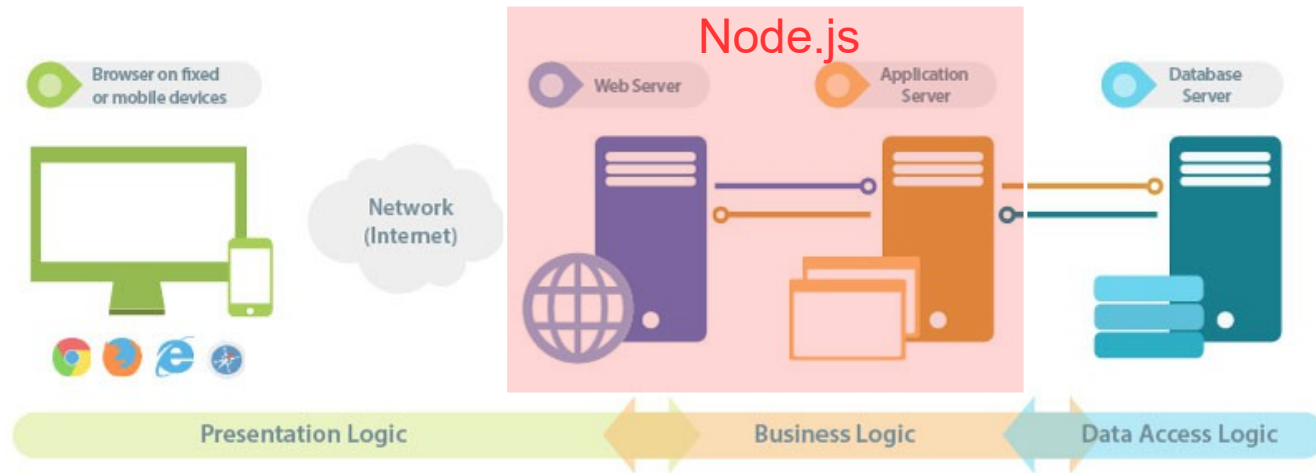
Introduction



UNC
SCHOOL OF INFORMATION
AND LIBRARY SCIENCE

Joan Boone
jpboone@email.unc.edu

Node.js



What is Node.js?

- Open source server environment for developing server-side applications in JavaScript
- Asynchronous, event-driven JavaScript runtime, designed to build scalable network apps
- Built on Chrome's V8 JavaScript engine

What can it do?

- Serve static and dynamic content
- Collect form data
- Create, open, read, write, delete, close files on the server
- Add, delete, modify data in your database

Why Node.js?

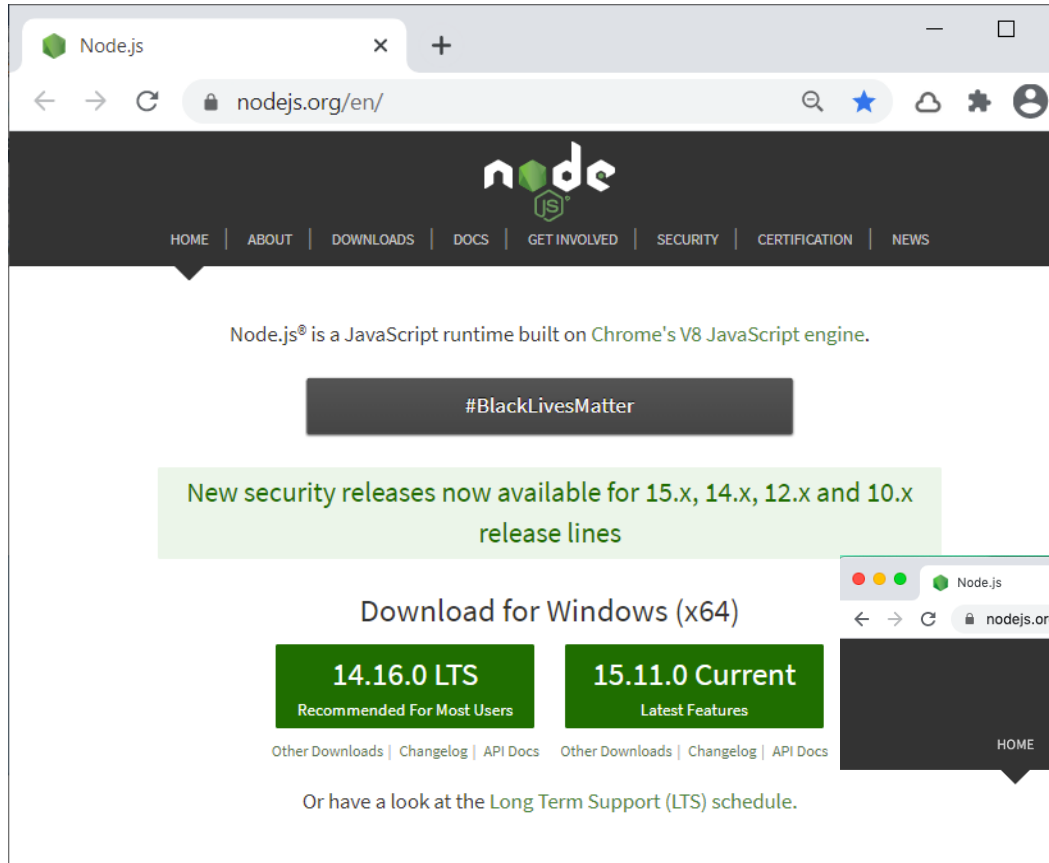
- Performance – designed to optimize throughput and scalability
- Code is written in “plain old JavaScript” which means less time is spent dealing with “context shift” between languages when writing both client-side and server-side code
- The node package manager (NPM) provides access to hundreds of thousands of reusable packages, and it supports dependency resolution.
- Portability – available on Windows, macOS, Linux, etc.
- Very active third party ecosystem and developer community.
[Stackoverflow Developer Survey 2020](#): Other Frameworks...

Who uses it? [9 Famous Apps Built with Node.js](#)

Comparison with PHP

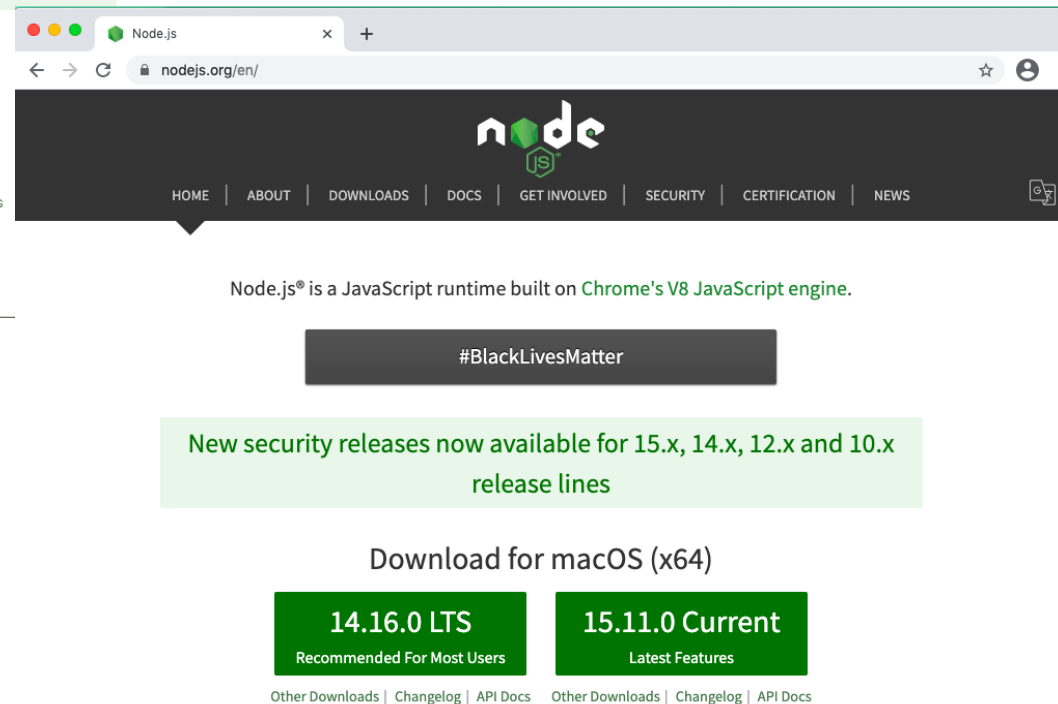
- [PHP vs. Node.js](#)
- [What is the difference between PHP and Node.js?](#)

Windows



But first...Install
Node.js
Download, and
double-click

Mac



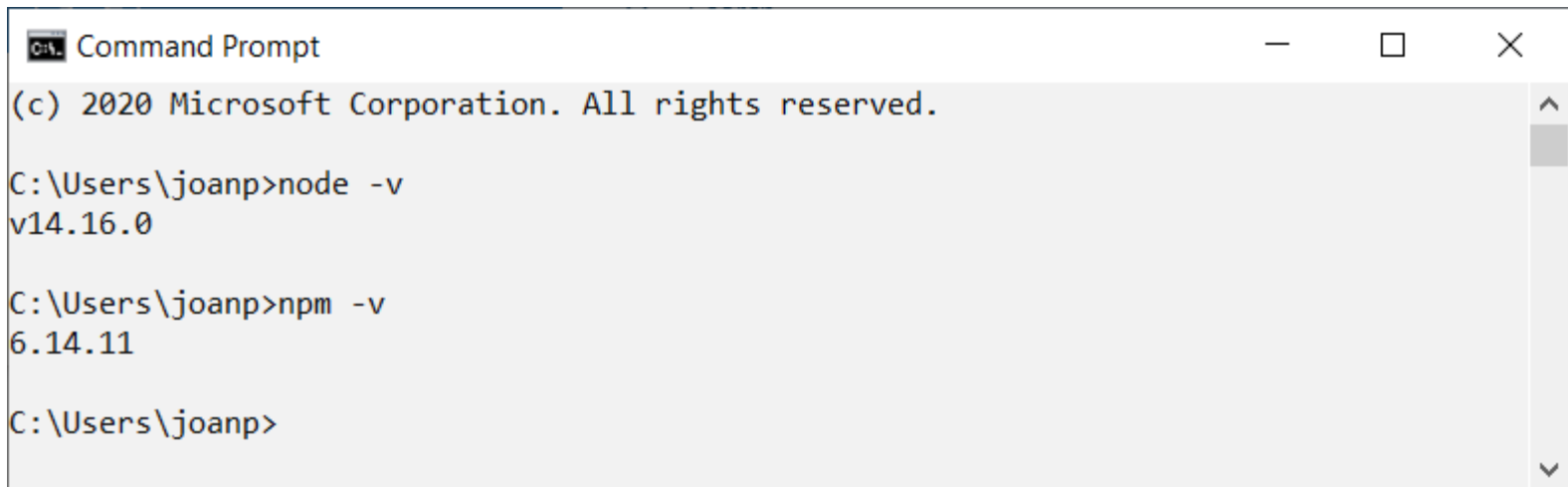
Test your installation

- The Node.js installation includes [NPM, a package manager](#) for Node.js. packages (or modules). Packages are JavaScript libraries that define functions that can be included in applications.
- Test your installation by opening a Terminal (Mac) or Command Prompt (Windows) window, and enter these commands that will return the installed version of Node.js and NPM.

node -v

npm -v

NOTE: if you do not see the version numbers displayed, then Node.js did not install properly.



```
Command Prompt
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\joanp>node -v
v14.16.0

C:\Users\joanp>npm -v
6.14.11

C:\Users\joanp>
```

Notes on using Mac Terminal or Windows Command Prompt window

- You need to be familiar with using the Mac Terminal or Windows Command Prompt window when starting a Node.js server.
- The most useful commands are
 - **cd** – this allows you to change directories so that you can navigate around your file folders (same on Mac and Win)
 - **dir** (Win) or **ls** (Mac) – will display the files in the current directory
 - **Ctrl-C** – to stop the Node.js server. This is important if you make changes to the startup script, e.g., **demo_fileserver.js**, because you will need to restart the server for the changes to take effect.

Some useful references

- Mac [Terminal for Beginners](#)
- [How to use the Windows command line](#)

Create a simple Node server

Follow the steps in [w3schools: Node.js Get Started](#) to create a simple Node server that displays “Hello World” in the browser.

- Create a folder (or directory) , e.g., [NodeProjects](#)
- In this folder, create the file, [myfirst.js](#) with these statements:

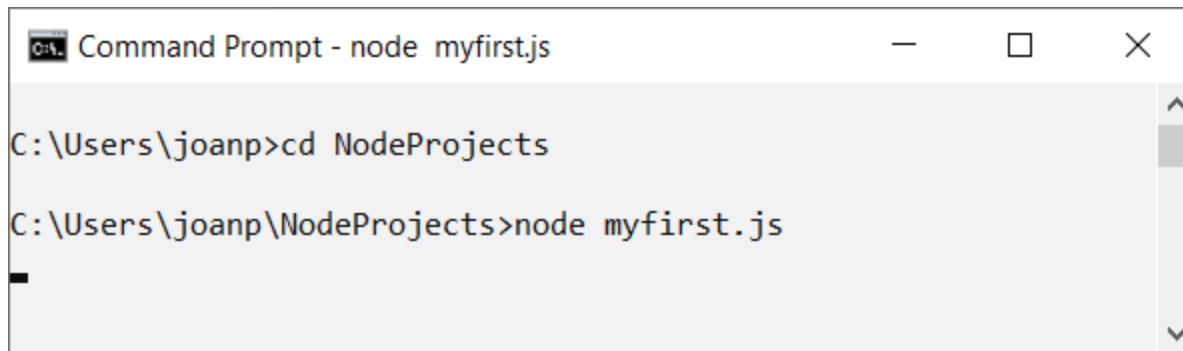
```
var http = require('http');  
  
http.createServer(function (req, res) {  
    res.writeHead(200, { 'Content-Type': 'text/html' });  
    res.write('Hello World!');  
    res.end();  
}).listen(8080);
```

Create a simple Node server

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

myfirst.js

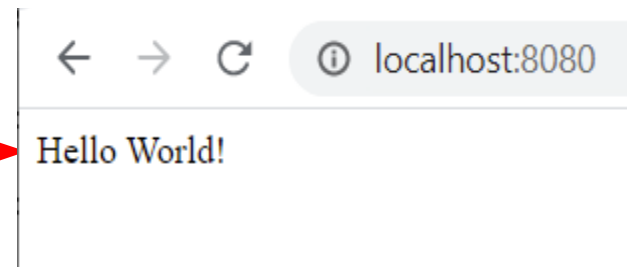
At the prompt, start the server by entering: **node myfirst.js**



```
Command Prompt - node myfirst.js  
C:\Users\joanp>cd NodeProjects  
C:\Users\joanp\NodeProjects>node myfirst.js  
-
```

In your browser, enter: **localhost:8080**

You should see this displayed in the browser window →



Source: [w3schools: Node.js Get Started](#)

Node.js HTTP Module

- In the `myfirst.js` program, the following statement includes the HTTP module so that it can be used to create an HTTP server:

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

- This module creates the HTTP server, and handles requests and responses between the server and client.
- Modules are similar to JavaScript libraries – they are collections of functions that are frequently used in applications. By packaging these functions as modules, developers can use them without having to write the code themselves.
- The HTTP module is a “built-in” module that is installed with Node.js.

Node.js HTTP Module

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

What this JavaScript program does:

- Creates a server that listens on port number 8080
- The server accepts a request (**req**), e.g., when you enter **localhost:8080** in the browser
- Sets the response (**res**) header with a 200 status code and the type of content for this response (html)
- Writes 'Hello World!' to the response, and ends the response

Node.js HTTP Module

If you open the Developer Tools (either Chrome or Firefox), you will see the request and response information

The screenshot shows a web browser at `localhost:8080` displaying "Hello World!". The Chrome DevTools Network tab is open, showing a request to `localhost` for `favicon.ico`. The request is successful with a status of 200 OK. The response is a 356 B HTML document.

Request info

- Request URL: `http://localhost:8080/`
- Request Method: GET
- Status Code: 200 OK
- Remote Address: `:::1]:8080`
- Referrer Policy: `strict-origin-when-cross-origin`

Response info

- Connection: keep-alive
- Content-Type: text/html
- Date: Fri, 05 Mar 2021 21:33:04 GMT
- Keep-Alive: timeout=5
- Transfer-Encoding: chunked

Node.js URL and File System Modules

The URL module provides functions that allow the parsing of a web address.

- For example, you can extract the domain address, pathname, or query parameters from the web address. This can be useful if an application needs to read a specific file, or to use the query parameters to search for specific data.

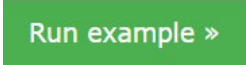
The File System module provides functions that allow you to work with files on your computer.

- For example, to read, modify, or delete files. When a Node server responds to a request for an HTML file, it is basically reading the file contents, and returning this data in the HTTP response.

Practice: URL and File System Modules

Work through the examples found in the w3schools Node.js Tutorial for [Node.js URL Module](#).

Example 1: [demo_url.js](#)

- This example illustrates how to parse a web address and display the component parts. You will need to copy and paste this example into a file, e.g., [demo_url.js](#), run it, and then ensure you see the same results as shown in  [Run example »](#)
- Try changing the contents of the web address (var adr) to see how the output changes.

Example 2: [demo_fileserver.js](#)

- This example reads an HTML file and renders it in the browser. If the file is not found, a 404 error message is displayed. In the tutorial, they suggest using 2 files, summer.html and winter.html, but you can also copy theme-change.html to your folder to test the code.