The World of Node.js

on IBM i

Presented by

Scott Klement

http://www.profoundlogic.com

To understand recursion
one must first understand recursion

The Agenda

Agenda for this session:

1. Node.js introduction
   - what it is
   - how popular it is
   - philosophy
2. Node.js basics
   - How it works
   - Hello World
3. Node.js Ecosystem
   - Node Package Manager (npm)
   - Demonstrations
     - Express
     - Debugging
     - E-mail
     - Spreadsheet

What is Node.js

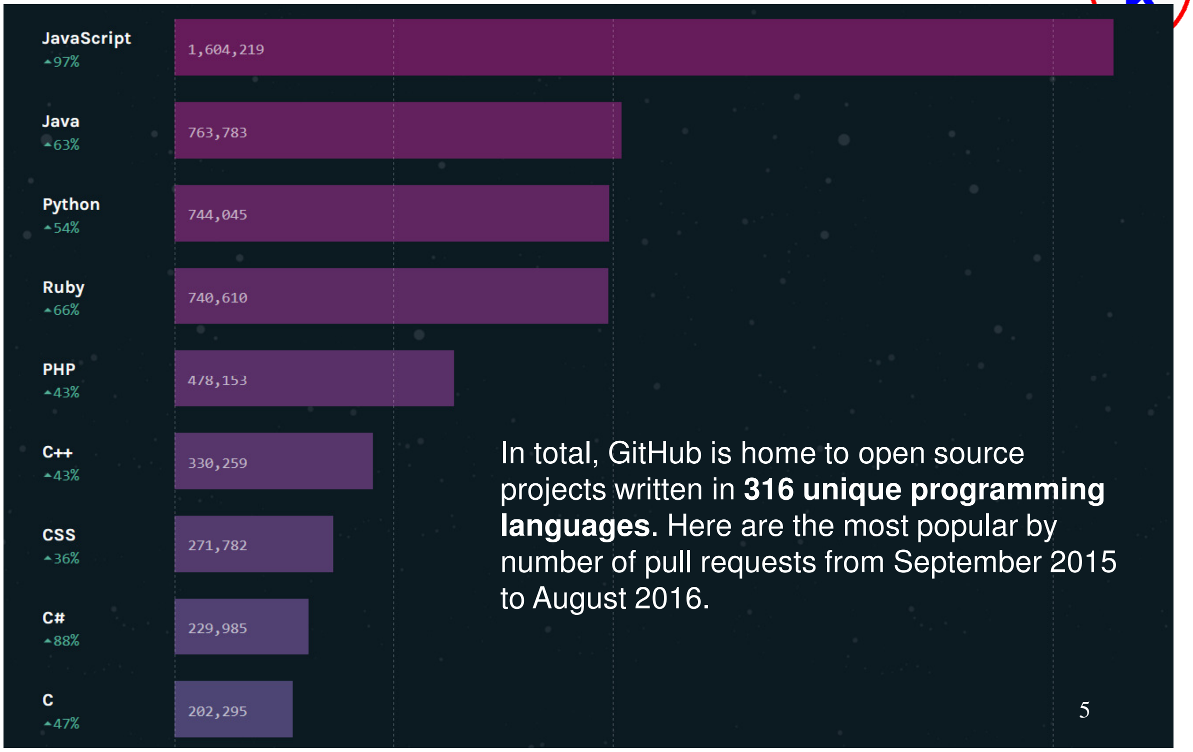
- JavaScript runtime outside of the browser
- Usually used for server-side applications
- Originally only for Linux in 2009
- Now available everywhere
  - (Linux, Windows, Unix, Mac OS X, IBM i, HP NonStop, etc)
- Event loop lets you create efficient code without need to code threads
- Provides access to file systems (i.e. IFS on IBM i)
- Provides basic network capabilities

Want the Latest and Greatest?

Cobol	1959
RPG II	1968
C	1972
RPG III	1978
C++	1983
Python	1991
RPG IV	1994
Java	1995
PHP	1995
Ruby	1995
Node.js	2009

New enough to have learned from previous languages, old enough to have grown very robust

Popularity of JavaScript



In total, GitHub is home to open source projects written in **316 unique programming languages.** Here are the most popular by number of pull requests from September 2015 to August 2016.

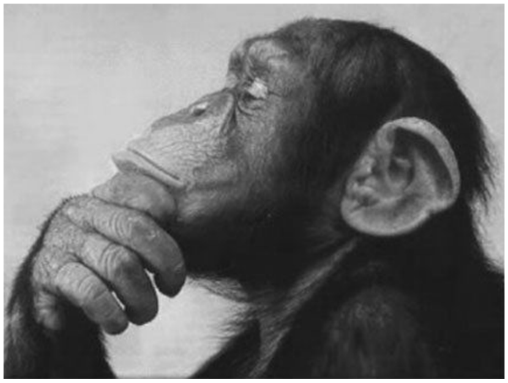
Think About It

In most other languages, web is the main way of making displays…

…modern web displays require JavaScript…

… coding in Java, PHP, Ruby, Python, and even modern RPG will require also coding in JavaScript.

So JavaScript will be the most popular. Why not use it on the back-end, too?



Companies Using Node.js

Walmart
ebay
PayPal

DOW JONES
intuit
NETFLIX

LinkedIn
The New York Times
Microsoft

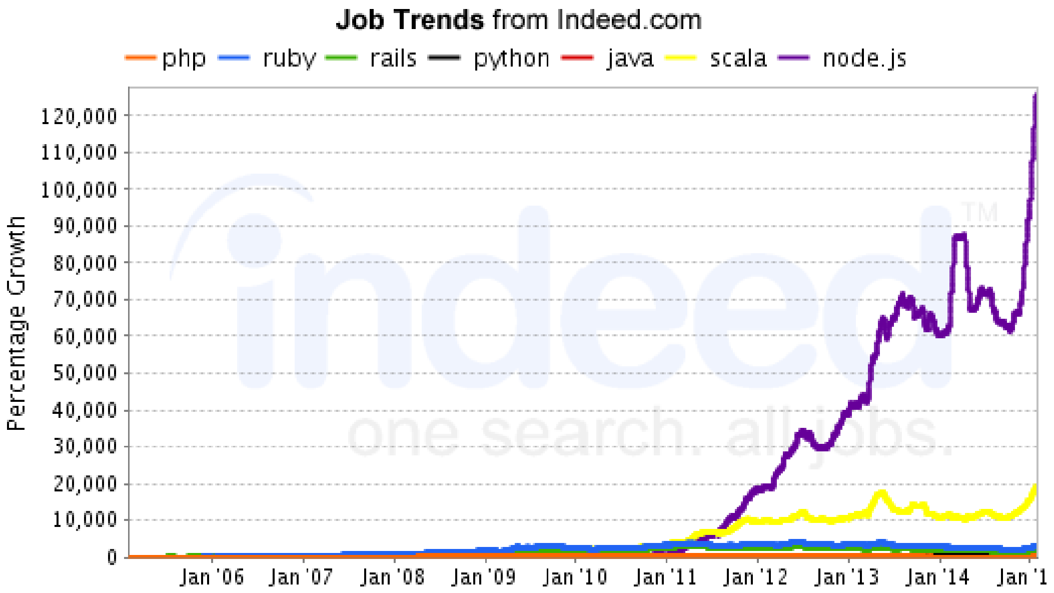
UBER
YAHOO!
Kingfisher

More Companies…

Big companies have moved from experimenting with Node to deploying it into production. Companies like...

NBC
Bank of America Merrill Lynch
STAPLES
TARGET
BEST BUY
Cigna.
Bloomberg
Alibaba.com
intuit
FANDANGO
PG&E
WELLS FARGO
Symantec.
Go Daddy
macy's
DIRECTV

Node Job Explosion



Node.js Philosophy

The philosophy of Node.js is different than other languages.

- Languages like RPG or Java try to include everything you'll need
- Node includes very little
- Like Unix philosophy:
  - Write modules that do one thing and do it well
  - Write modules that work together with other modules
  - Write modules that handle streams and events
- What makes Node unique and powerful is the ecosystem!

Support on IBM i

IBM provides support

- via the new yum package system
- (older support was available via LICPGM 5733OPS, but this is now passé)
- Runs in PASE
- DB2 module in provided via open source idb-connector module
- Source kept in IFS
- Edit with any editor….
  - RDi works fine
  - Notepad++ works
  - VS Code is free, and is our favorite (so far). Also has integrated debugging!
- Integration with IBM I via XMLSERVICE or iToolKit
  - Call native programs, access objects, etc

Demand comes from <u>development community</u>

- No company backing like Zend for PHP or PowerRuby for Ruby
- Lots of community support in forums, web sites, etc.

Learning Node / JavaScript

I cannot teach the whole language in this session!

- But, maybe you already know JavaScript?
- Or, once you know one language, learning another isn't hard.

Some resources we've found:

- *Sams Teach Yourself Node.js in 24 Hours, by George Ornbo*
  http://a.co/7t52WX3

- *Pro Node.js for Developers by Colin J. Ihrig*
  http://a.co/48uAr1W

- Main website for Node.js, including API docs:
  http://nodejs.org

Also, just search Google for what you're looking for. This can be really helpful!

JavaScript Syntax Basics (1 of 3)

Declaring and Comparing

- declare a variable with var
- type of variable is determine by what is assigned to it – and can change.
- A single = means "assign value"
- Double == means "compare" data types will be converted if necessary
- Triple === means "compare" but result only matches if it's the same data type

```
var myString = "Hello World";
var myNumber = 1;
var myArray = [ "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" ];

if ( myVariable == "something") {
  doSomething();
}
else {
  doSomethingElse();
}

if ( myNumber == "1" ) // works!

if ( myNumber === "1" ) // doesn't work, different types
```

JavaScript Syntax Basics (2 of 3)

```
while ( myNumber > 1) {
  // do something while myNumber > 1
}

do {
  // do while myNumber > 1, test condition at end of loop.
}
while ( myNumber > 1 );

for (var x=1; x<=10; x++) {
  // do something 10 times.
}

var employee_rec = {
    first: "Scott",
    last: "Klement",
    num: 1000
}

for (field in employee_rec) {
  // do something for each field in an "object" (data structure)
  console.log(field + " = " + employee_rec[field]);
}
```

JavaScript Syntax Basics (3 of 3)

```
function printSomething( something ) {
  console.log(something);
}

// prints "Hello World"
printSomething("Hello World");

// functions can be variables - allow for "callbacks"
//   this function automatically calls a routine "num" times.
function repeat( num, callback ) {
  for (var x=1; x<=num; x++) {
    callback(x);
  }
}

// this will print 1-5 on the console display
repeat( 5, printSomething);

// anonymous functions are functions defined on-the-fly just to assign
//  to variables or parameters.  This prints 1-5 with "call number" before it.
repeat( 5, function(n) { printSomething("call number " + n)});
```

Node Modules

Node provides the concept of "modules"

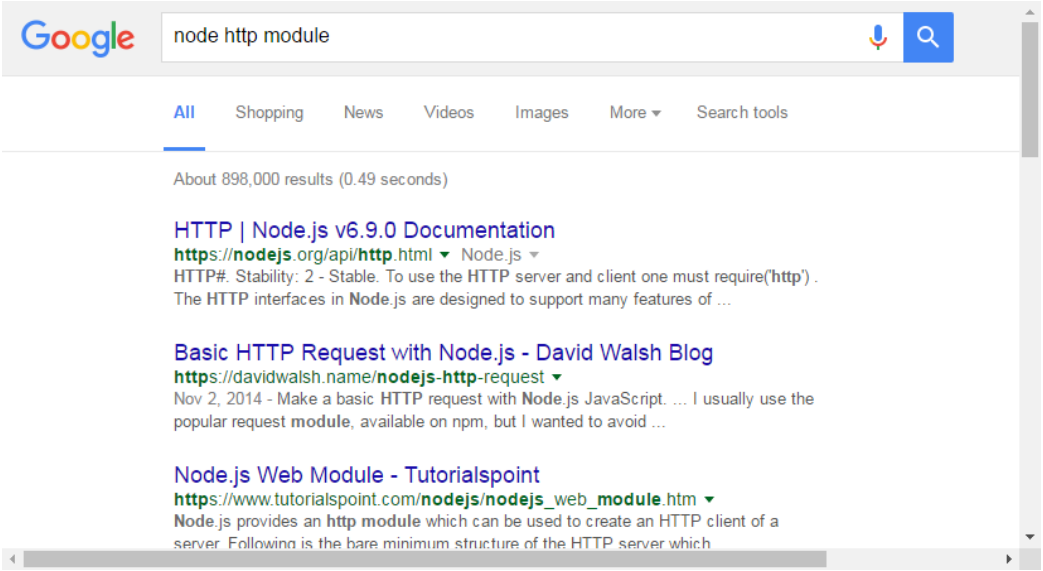
- Libraries (like service programs?) that you can use in your program
- Provided as downloadable packages (lots and lots of them available)
- Or, you can write your own.
- Bring them into your program with the require statement

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

var module = require("module name here");

module.doSomething(); // runs something in the module
```

Easy To Find Docs With Google

Basically just google "node <thing to find>"

- In this case, the built-in http module documentation…



About 898,000 results (0.49 seconds)

HTTP | Node.js v6.9.0 Documentation
https://nodejs.org/api/http.html ▾ Node.js ▾
HTTP#. Stability: 2 - Stable. To use the HTTP server and client one must require('http') . The HTTP interfaces in Node.js are designed to support many features of ...

Basic HTTP Request with Node.js - David Walsh Blog
https://davidwalsh.name/nodejs-http-request ▾
Nov 2, 2014 - Make a basic HTTP request with Node.js JavaScript. ... I usually use the popular request module, available on npm, but I wanted to avoid ...

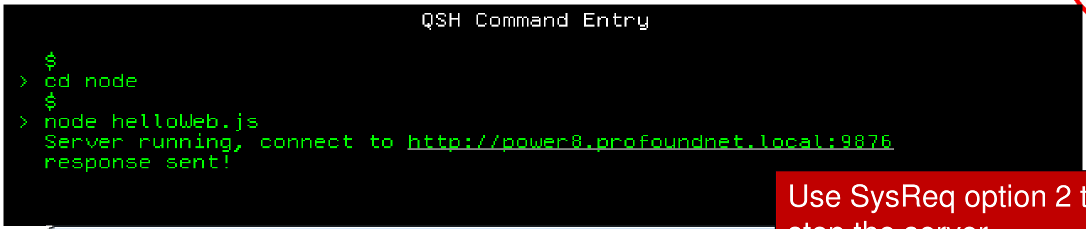
Node.js Web Module - Tutorialspoint
https://www.tutorialspoint.com/nodejs/nodejs_web_module.htm ▾
Node.js provides an http module which can be used to create an HTTP client of a server. Following is the bare minimum structure of the HTTP server which

Hello World (1 of 2)

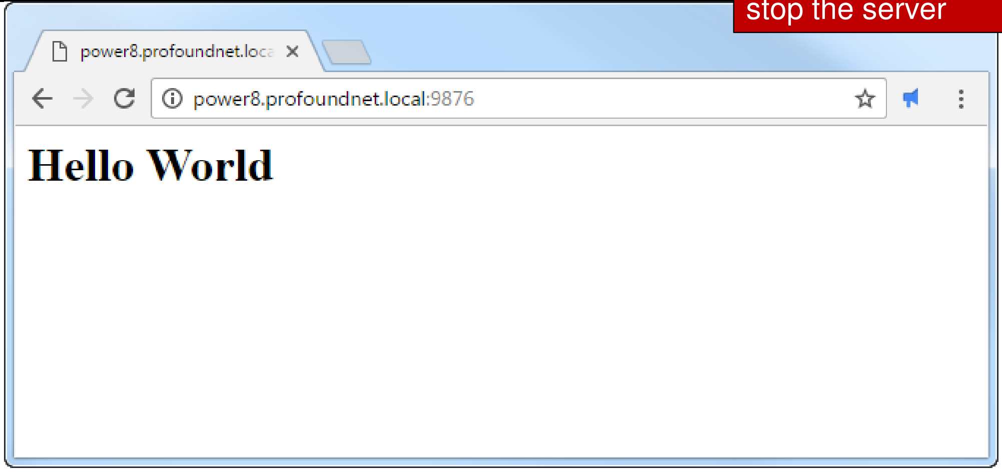
Since the "http" module contains all the code needed to communicate with HTTP, it's very easy to write a simple web server in Node.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

// Load the http module to create an http server.
var httpServer = require('http');

// Configure our HTTP server to respond with Hello World to all requests.
var server = httpServer.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/html"});
  response.end("<h1>Hello World</h1>");
  console.log("response sent!");
});

// Listen on port 9876
server.listen(9876);

// Even though this ends the program, it will remain active since
// the node event loop has more to do (due to http server above)
console.log("Server running, connect to http://your-server:9876");
```

Hello World (2 of 2)



```
                          QSH Command Entry

  $
> cd node
  $
> node helloWeb.js
  Server running, connect to http://power8.profoundnet.local:9876
  response sent!
```



Hello World

Use SysReq option 2 to stop the server

The Real Power of Node.js

The real power of Node.js comes from the "ecosystem" of modules that are available to use!

- Most are available with the Node Package Manager (npm)
  - Finds the module and installs it
  - Allows upgrades when needed
  - Installs any dependencies
  - Allows uninstalling when needed
- Sometimes modules are provided other ways
  - commercial software
  - sites like github, etc.
  - examples posted on internet

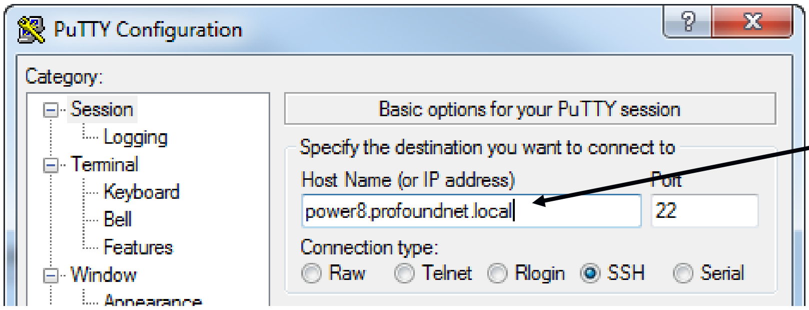
Using a Real Unix Terminal

Node.js on IBM i was written for AIX and runs under PASE. Some tools (npm is one, there are others) try to use Unix terminal features, such a colors, and so assume you are using a Unix terminal emulator.

This is not required for using Node.js, but it does work very nicely, and we prefer it!

A good way to do that on IBM i is to start the IBM-supplied SSHD and connect with a Unix terminal program such as Putty. http://www.chiark.greenend.org.uk/~sgtatham/putty/

```
STRTCPSVR SERVER(*SSHD)
```



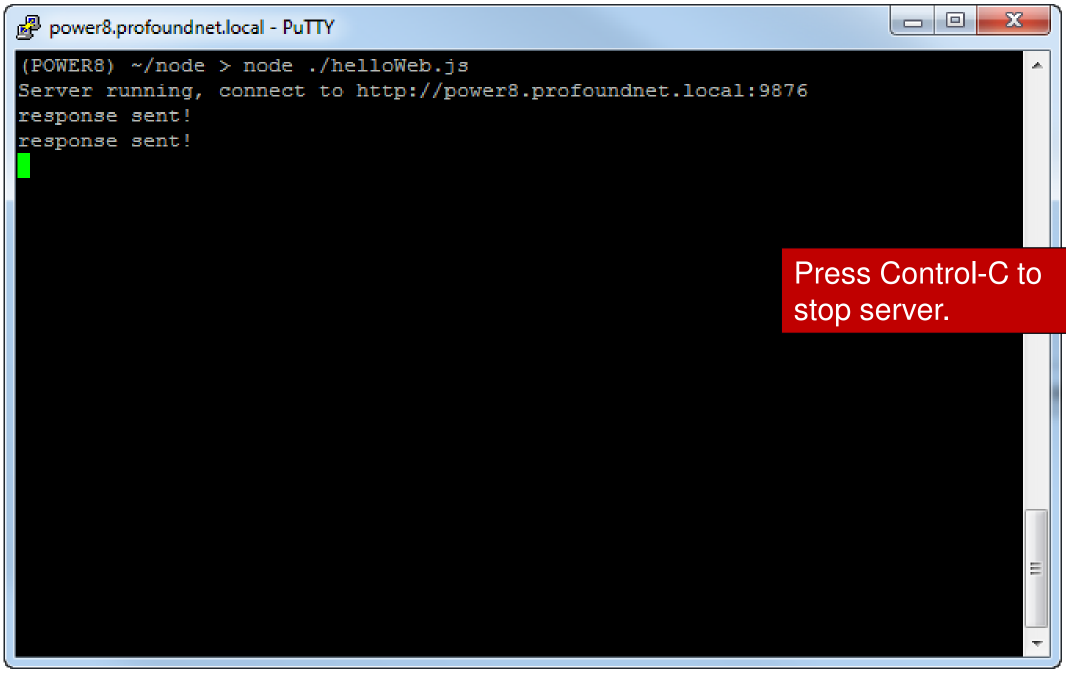
Host Name (or IP address): power8.profoundnet.local

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Use the IP address or hostname of your IBM i here

Run Hello World from Putty



```
(POWER8) ~/node > node ./helloWeb.js
Server running, connect to http://power8.profoundnet.local:9876
response sent!
response sent!
```

Press Control-C to stop server.

Node Package Manager (npm)

NPM is used from the PASE command like (via Putty or 5250)

- `npm list` will list the packages currently installed
- `npm install <package>` will install a package
- `npm update <package>` will update an existing package
- `npm uninstall <package>` will remove an existing package
- `npm search <keyword>` will search for a package

By default packages are listed/installed in the local project. Add `--global` to make it install system-wide. (or `-g` for short!)

By default, it tries to use the most current version of a package available. You can add @VERSION to the package name to install a particular version, such as

```
npm install my-package@1.0.0 --global
```

For example, for a better HTTP server try
google: `npm express`

Express-Generator

The basic http module in Node makes it easy to write your own HTTP server, but maybe you want something that does a little more work for you?

Express.js is a very popular web application framework for Node.js.

- makes it easy to build a powerful web site.
- adds routing support
- template engines
- file uploading
- single-page applications
- but still provides support for coding whatever you want in your logic.

Express Generator

- add-on for Express.js that generates applications for you
- installs a command line tool that you run to make things
- build a shell application in seconds!

```
npm install express-generator --global
```

(this will install Express.js as a dependency)

Express-Generator

Once installed, you can generate an application skeleton like this:

```
express yourApp
```

This creates a yourApp directory and generates all of the files and subdirectories to provide a good skeleton of an Express.js application.

```
cd /home/sklement/node
$
express -e demoApp
[messages show generated files]
$
cd demoApp
$
npm install
[messages show building files]
$
npm start
```

Accessing DB2 for i in Node.js

IBM created a Node.js module (open source, available via npm) for IBM i database access. The name of the module is idb-connector

https://www.npmjs.com/package/idb-connector

```
cd /path/to/myproject
npm install idb-connector
```

Like all modules, you require it – then can use it.

```
var db2i = require("idb-connector");

var dbconn = new db2i.dbconn();

dbconn.conn("*LOCAL"); // connect to local database

// Same as naming = *SYS in RPG
dbconn.setConnAttr(db2i.SQL_ATTR_DBC_SYS_NAMING, db2i.SQL_TRUE);
```

Running an SQL Statement

The SQL results are returned asynchrously in a JavaScript array of objects

```
var stm = new db2i.dbstmt(dbconn);

var sql = "select ORDERREF, ORDERLINE, PRODREF, ORDERQTY," +
              "SALETOTAL from SALESDTL";

stm.exec(sql, function(result) {

  // statement is run in the background
  // this function is called when it is ready

  result.forEach(function(row) {

    // this function is repeated for each row available
    // row.ORDERREF, row.ORDERLINE, etc will contain the fields
    // in string format

  }
}
```

Example, ExcelJs

Distributing reports as spreadsheets has been very successful for me, and so I wanted to try it in Node.js.

Google: `nodejs excel spreadsheet`

There are a bunch of them, there are also tools for other spreadsheets (such as Google docs spreadsheets) ExcelJs looked decent, so we installed it.

For the sake of example, this will be installed locally into the project (not system-wide)

```
$
  mkdir spreadsheet
$
  cd spreadsheet
$
  npm install exceljs
```

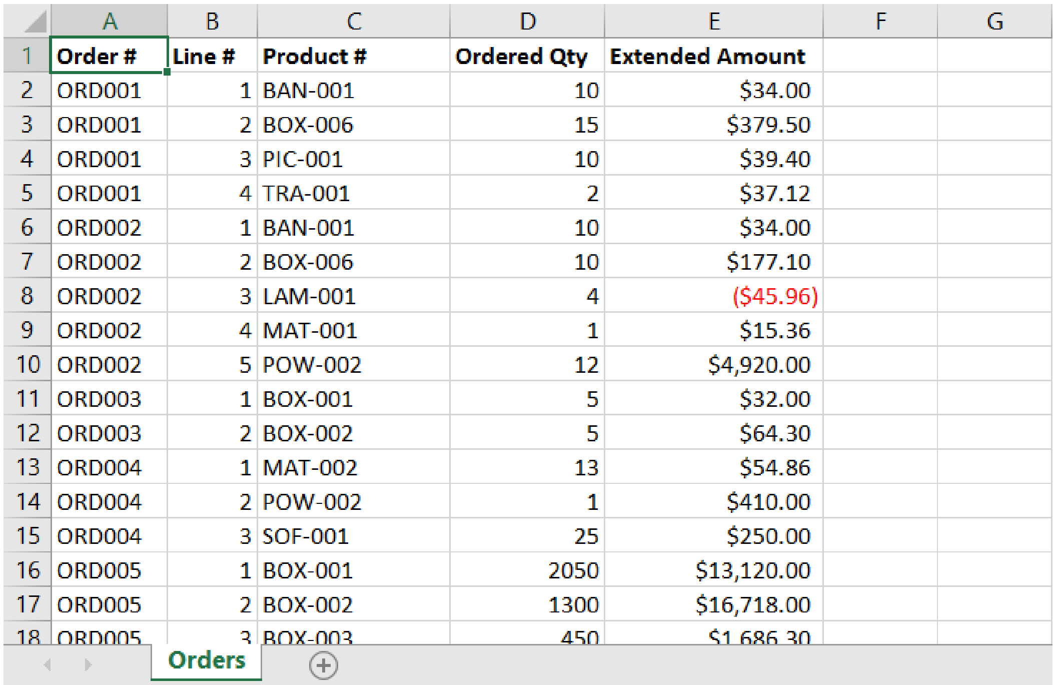
ExcelJs, Configuring Columns

```
// Create an Excel workbook with one worksheet named "Orders".
var Excel = require("exceljs");
var workbook = new Excel.Workbook();
var worksheet = workbook.addWorksheet("Orders");

// Add column definitions to the worksheet.
// The column keys are named after the DB2 column names.
worksheet.columns = [
    { header: "Order #", key: "ORDERREF", width: 9 },
    { header: "Line #", key: "ORDERLINE", width: 7,
               style:{ numFmt: "0" } },
    { header: "Product #", key: "PRODREF", width: 15 },
    { header: "Ordered Qty", key: "ORDERQTY", width: 12,
               style:{ numFmt: "0" }  },
    { header: "Extended Amount", key: "SALETOTAL", width: 17,
               style:{ numFmt: "$#,##0.00_);[Red]($#,##0.00)" }}
];

worksheet.getRow(1).font = { name: "Calibri", size: 11,
                              bold: true };
```

ExcelJs, Populating Rows

```
var stm = new db2i.dbstmt(dbconn);
var sql = "select orderref, orderline, prodref, orderqty,
saletotal from salesdtl";

stm.exec(sql, function(rs) {
  rs.forEach(function(row) {

   // Cast numeric fields to numbers (so Excel sees them as
   // numbers rather than strings)
    row.ORDERLINE = Number(row.ORDERLINE);
    row.ORDERQTY = Number(row.ORDERQTY);
    row.SALETOTAL = Number(row.SALETOTAL);

    worksheet.addRow(row);

  });

  // Write spreadsheet to IFS
  workbook.xlsx.writeFile("orders.xlsx");
});
```

Example, Output



	A	B	C	D	E	F	G
1	**Order #**	**Line #**	**Product #**	**Ordered Qty**	**Extended Amount**		
2	ORD001	1	BAN-001	10	$34.00		
3	ORD001	2	BOX-006	15	$379.50		
4	ORD001	3	PIC-001	10	$39.40		
5	ORD001	4	TRA-001	2	$37.12		
6	ORD002	1	BAN-001	10	$34.00		
7	ORD002	2	BOX-006	10	$177.10		
8	ORD002	3	LAM-001	4	($45.96)		
9	ORD002	4	MAT-001	1	$15.36		
10	ORD002	5	POW-002	12	$4,920.00		
11	ORD003	1	BOX-001	5	$32.00		
12	ORD003	2	BOX-002	5	$64.30		
13	ORD004	1	MAT-002	13	$54.86		
14	ORD004	2	POW-002	1	$410.00		
15	ORD004	3	SOF-001	25	$250.00		
16	ORD005	1	BOX-001	2050	$13,120.00		
17	ORD005	2	BOX-002	1300	$16,718.00		
18	ORD005	3	BOX-003	450	$1,686.30		

Debugging With VS Code

Visual Studio Code (VS Code) is an open source tool from Microsoft (originally based on their commercial Visual Studio tool.)

(Full info on how to use VS Code is beyond the scope of this presentation.)

- Debugging on your own PC is easiest. I recommend you start with this.
- Sometimes, it's nice to be able to debug code that's running on the IBM i

VS Code has built-in support for debugging Node.js both on your PC and remotely. For details,. See here:
https://code.visualstudio.com/docs/nodejs/nodejs-debugging

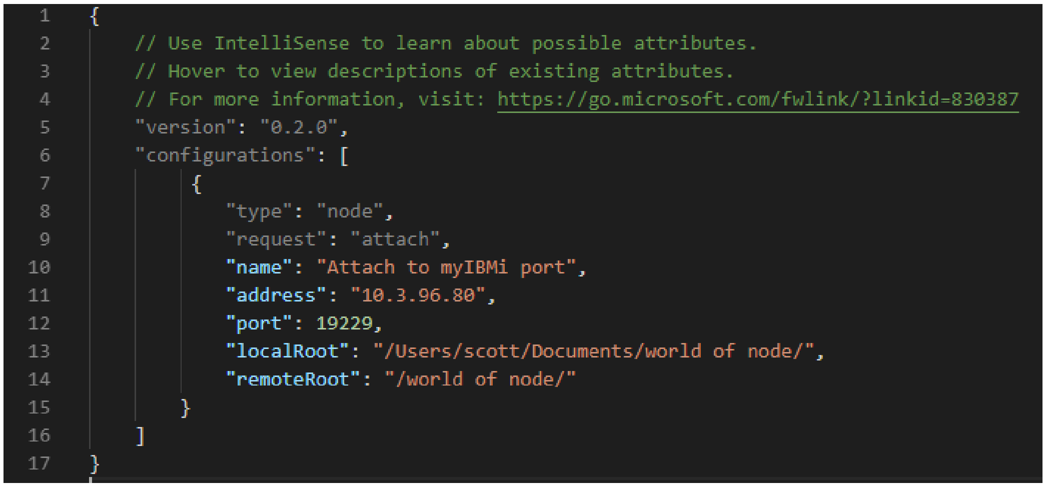
To do remote debugging:

- Copy the source code to your PC
- Create a debugging profile (VS Code refers to this as 'launch.json')
  - Use the "attach" request type
  - Use "address" and "port" to configure the address/port on your IBM I
  - Use "localRoot" and "remoteRoot" to point to the locations of the source on your PC and IBM i, respectively
- Run on the IBM i with your IBM i's IP address and port:
  - `node --inspect=address:port example.js` To start running immediately
  - `node --inspect-brk=address.port example.js` To wait for the debugger before starting
- Start the debug profile in VS Code.

Example Debugging Profile

In VS Code "Debug" tab:

- Click "Add Config (your workspace)"



```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "attach",
            "name": "Attach to myIBMi port",
            "address": "10.3.96.80",
            "port": 19229,
            "localRoot": "/Users/scott/Documents/world of node/",
            "remoteRoot": "/world of node/"
        }
    ]
}
```

- Save changes
- I had to restart VS Code for it to show the new profile
- Now you can start the "Attach to myIBMi" profile to connect to the port.

Example, Node Mailer

It would be convenient to be able to e-mail the spreadsheets we created. For example, a nightly batch job might create reports, and need to send them to the appropriate people.

Google: `nodejs send email`

We picked nodemailer, looked like an easy to use tool for e-mailing.

```
$
  mkdir email
$
  cd email
$
  npm install nodemailer
```

Nodemailer, configuring SMTP

To send e-mail using the IBM i SMTP server, just create a transport using SMTP to the local system.

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

var transporter = nodemailer.createTransport(
      "smtp://localhost");
```

Or, perhaps you'd rather use a separate e-mail server? For example, an Exchange server?

```
var transporter = nodemailer.createTransport(
      "smtp://smtp.example.com");
```

Perhaps the server requires a userid/password?

```
var transporter = nodemailer.createTransport(
      "smtp://myUserId:myPassword@smtp.office365.com");`
```

Nodemailer, Building a Message

A JavaScript object (like an RPG data structure) contains fields for the various things needed in an e-mail message

- from = sender's e-mail address
- to = recipient's e-mail address
- subject = message subject
- html = body of e-mail in HTML format (can be as long as needed)
- attachments = JavaScript array of IFS files to include as attachments

```
var message = {
    from: "Node.js Email Example <example@company.com>",
    to: "Scott Klement <sklement@example.com>",
    subject: "Node.js Email Example",
    html: "<b>Sent from Node.js on IBM i</b>",
    attachments: [{
      path: "orders.xlsx"
    }]
};
```

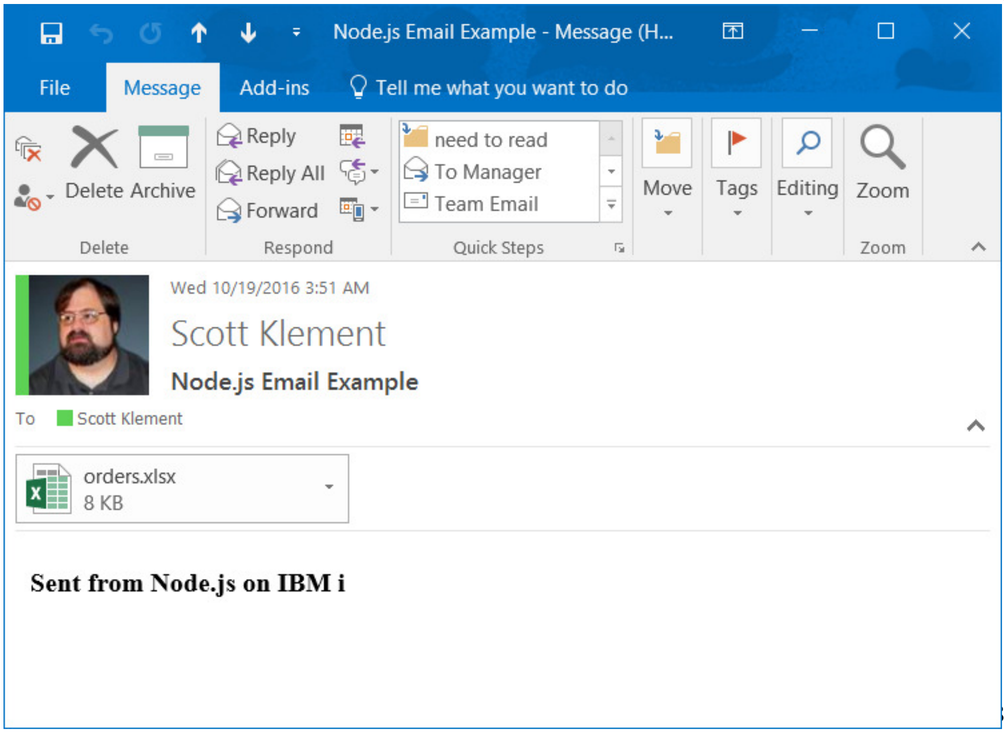
Nodemailer, Sending a Message

The transporter we created earlier can send the message.

- message = pass the message object (see last slide) as first parameter
- callback function = second parameter is a function that is called when the message has been sent. (message is sent asynchronously)

```
transporter.sendMail(message, function(error, info) {
  if (error) {
    console.log(error);
  }
});
```

Nodemailer, Output (in Outlook)



Wed 10/19/2016 3:51 AM

Scott Klement

Node.js Email Example

To Scott Klement

orders.xlsx
8 KB

Sent from Node.js on IBM i

Final Thoughts

- JavaScript is a powerful, robust, popular language
- Now you can run server-side JavaScript on IBM i with Node.js
- The real power of Node.js comes from the ecosystem
- Small tools that are well-written and powerful
- Designed to fit together with other tools so you can build whatever you need

- We have demonstrated only a small number of tools
- But, enough that you can see how powerful this is?

This Presentation

You can download a PDF copy of this presentation and the sample code that we used from

http://www.scottklement.com/presentations/

Thank you!