

The World of Node.js



on IBM i

Presented by

Scott Klement and David Russo

<http://www.profoundlogic.com>

© 2016, Scott Klement, David Russo

*To understand recursion
one must first understand recursion*

The Agenda

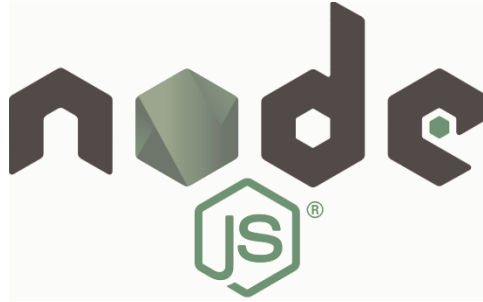


Agenda for this session:



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
 - Node Inspector
 - 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

3

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

4

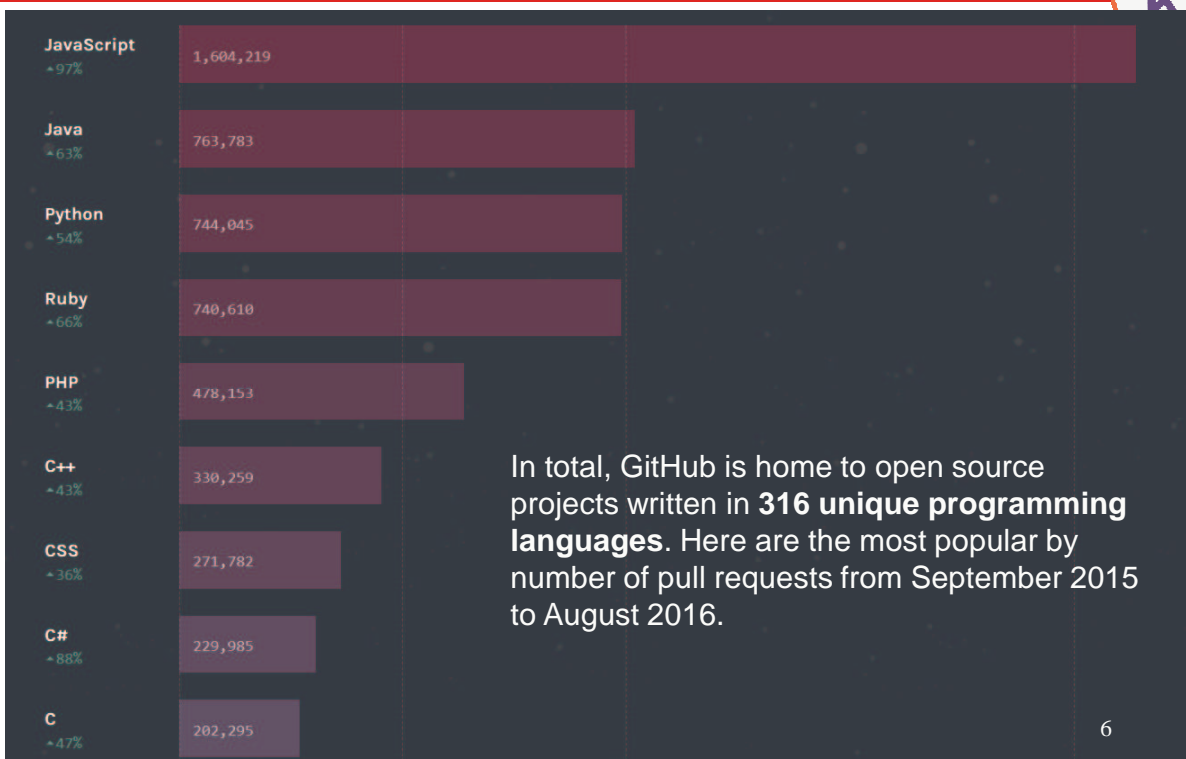
Browser vs. Server



- Assignments, comparisons, etc
- if, while, do, for, foreach, etc.
- curly braces for grouping
- function definitions, including anonymous functions
- closures and variable scoping
- Console routines (mainly for debugging or diagnostics)
- Routines for manipulating the page.
- DOM objects provided by browser
- Use require to bring in provided routines
- Network, file system, database, etc routines

5

Popularity of JavaScript



6

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?



7

Companies Using Node.js



Walmart

ebay

PayPal

DOW JONES

intuit

NETFLIX

LinkedIn

The New York Times

Microsoft

UBER

YAHOO!

Kingfisher

8

More Companies...



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



Bank of America
Merrill Lynch



Cigna.

Bloomberg

Alibaba.com

intuit

FANDANGO



WELLS
FARGO

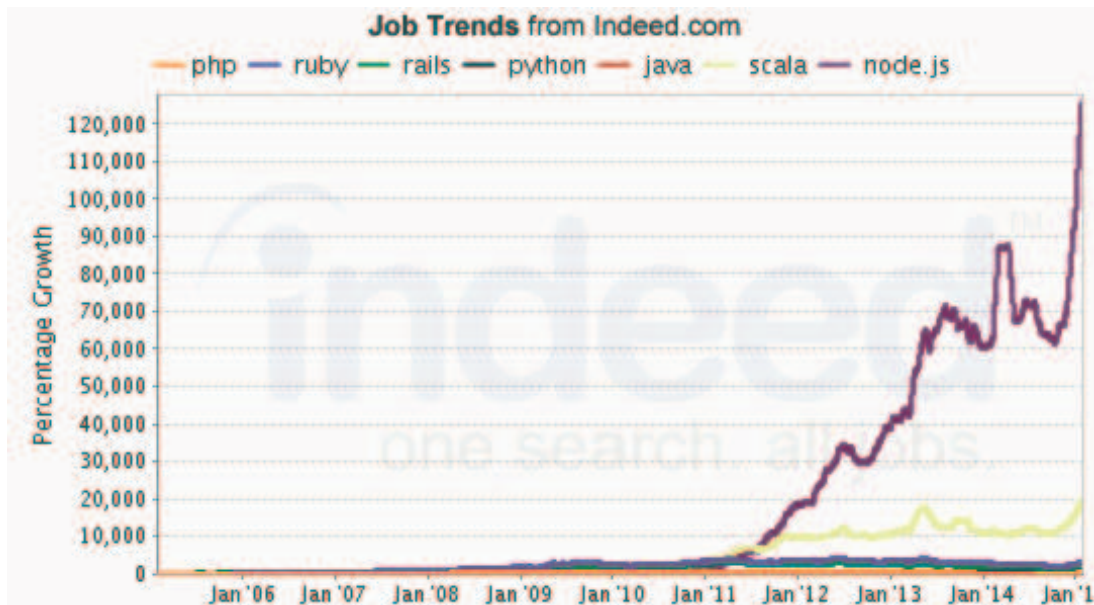


Go Daddy

★ macy's



Node Job Explosion



Node.js Philosophy



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

David's take:

- 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!

11

Support on IBM i



IBM provides support

- **5733OPS** no charge option
- Runs in **PASE**
- **DB2** module is provided
- Source kept in IFS
- Edit with any editor.... RDI and Notepad++ (free) work great.
- **Integration** with IBM I via XMLSERVICE
 - Call native programs, access objects, etc

Demand comes from development community

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

12

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!

13

JavaScript Syntax Basics (1 of 3)



Declaring and Comparing

- declare a variable with **var**
- type of variable is determined 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
```

14

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]);  
}
```

15

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)});
```

16

Node Modules



Node provides the concept of "modules"

- Libraries (like service programs?) that you can use in your program
- Sometimes included with the Node.js install
- Or, 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
```

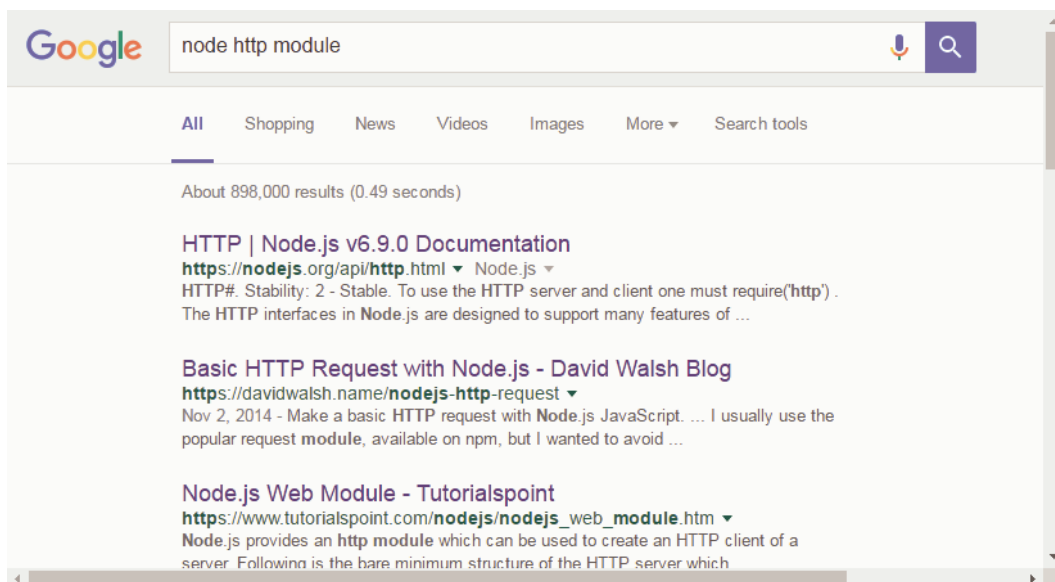
17

Easy To Find Docs With Google



Basically just google "node <thing to find>"

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



18

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");
```

19

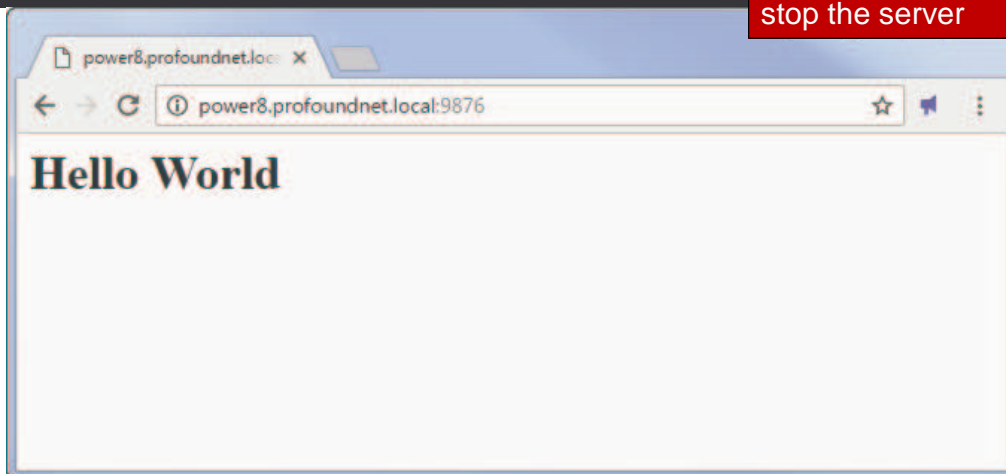
Hello World (2 of 2)



```
QSH Command Entry

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

Use SysReq option 2 to stop the server



20

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

21

Using a Real Unix Terminal

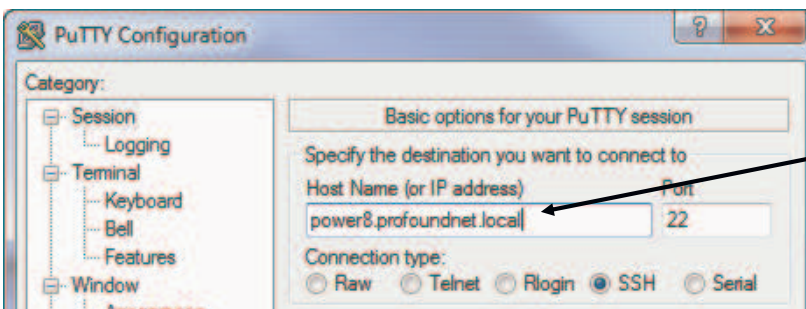


Node.js on IBM i was written for AIX and runs under PASE. Some tools (npm is one, there are others) try to use Unix terminal features, such as colors, and so assume you are using a Unix terminal emulator.

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)
```



Use the IP address or hostname of your IBM i here

22

Run Hello World from Putty



23

Node Package Manager (npm)



NPM is used from the PASE command line (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 install express`

24

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)

25

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
```

26

Accessing DB2 for i in Node.js



IBM includes support for DB2 for i (the database built in to IBM i) in the form of another Node.js package. It is installed with 5733OPS, PTF SI61394

```
// Connect to *LOCAL database and set system naming convention.
// The job's library list will be used to resolve objects in SQL
statements.

var db2i = require("/QOpenSys/QIBM/ProdData/OPS/Node4/os400" +
                  "/db2i/lib/db2a.js");

var dbconn = new db2i.dbconn();

dbconn.conn( "*LOCAL" );

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

27

Running an SQL Statement



The DB2 for i support can run statements in "blocking" (synchronous) or "non-blocking" (asynchronous) mode. IBM recommends the latter, as they perform better.

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

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

  })
})
```

28

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
```

29

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 };
```

30

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");
});
```

31

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		

32

Debugging With Node-Inspector



If you've ever debugged JavaScript in the browser, you'll find it just as easy and powerful to debug Node.js with Node Inspector. It uses Chrome's built-in debugger remotely.

```
npm install node-inspector --global
```

```
node-debug --web-host your-ibmi.com --web-port=13000 yourNodeApp.js
```

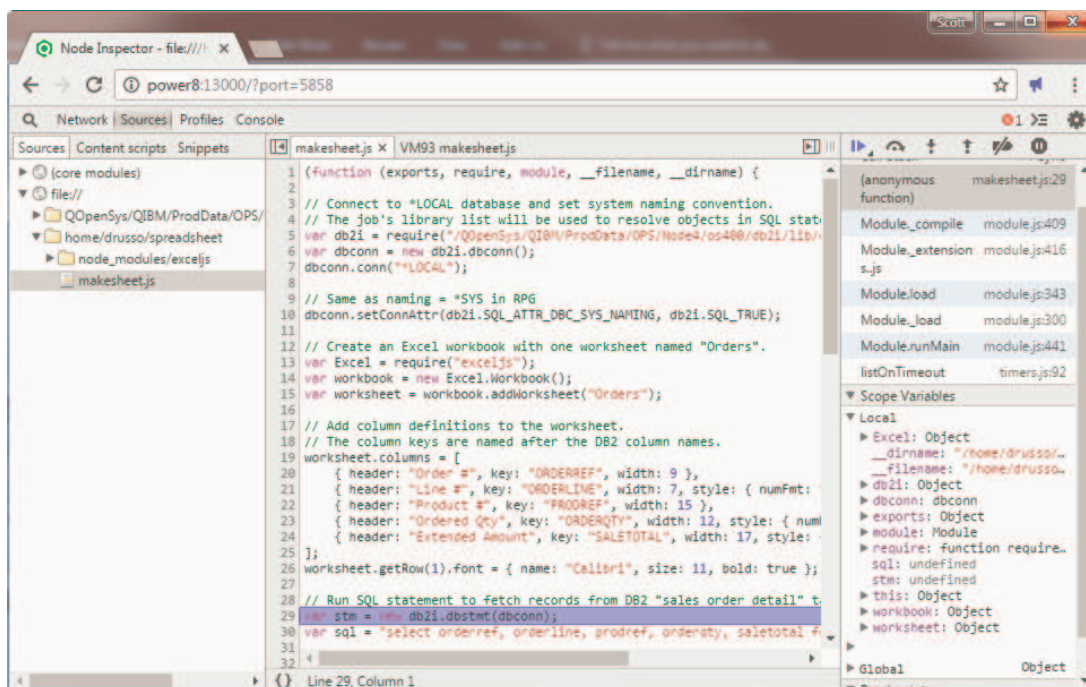
Then point chrome to <http://mysystem:13000/?port=5858>

If you run this on your PC, it'll automatically open Chrome for you. If you debug it from IBM i, you'll have to open Chrome manually.

This will open the Chrome debugger, and you can debug it, step through it, display variables, etc. Very powerful!

33

Node-Inspector, Screenshot



34

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
```

35

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");`
```

36

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"  
  }]  
};
```

37

Nodemailer, Sending a Message



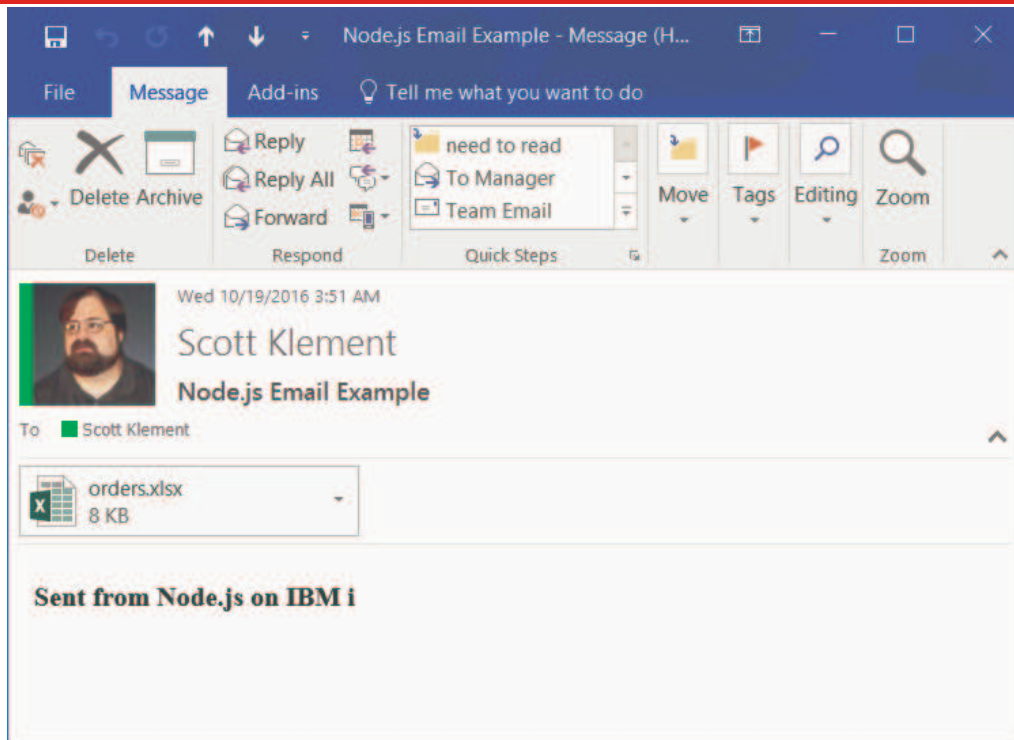
The transporter we created earlier can send the message.

- **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);  
  
  }  
  
});
```

38

Nodemailer, Output (in Outlook)



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!