Node.js

Ryan Farnell
@criscokid
ryan.farnell@me.com

About Me

- Web developer at Bizzuka Inc.
- Owner of Laughing Lark LLC

What is Node JS

- Tool designed to make building scalable network programs easily
- Built on top of V8
- Uses Javascript as it's primary language
- HTTP as a first class citizen

Javascript

- Node uses Javascript as it’s language
- For web developers this can mean only needing to know one language for all your work
- Not a browser so certain globals are missing (document, alert(), etc.)

Evented I/O

- Rather than using threads to scale, Node prefers preventing the main thread from being blocked in the first place
- Similar to Event Machine for Ruby or Twisted for Python
- Javascript in the browser is already based on events, Node moves them to a system level

Asynchronous

- Node performs all operations that take time in asynchronous manner
- There are no synchronous APIs built into Node.

Demo

Asynchronous

- Involves requesting something time consuming to occur.
    - Open/Reading a file
    - Connecting to a network socket and reading data
    - Querying an API
- Once the request is made we continue on to the next line of code before waiting for the time consuming request to finish.

Lambda Expression

- Chunk of code that can be used later.
- Can be passed around as data.

Lambda Expression

```
function(){
  //some code in
  here
}

function(a, b, c){
  return a+b*c;
}
```

```
function(a, b, c){
  return a+b*c;
}
```

Parameters

```
function(a, b, c){
  return a+b*c;
}
```

Body

```
function(a, b, c){
  return a+b*c;
}
```

Callbacks

- Piece of code that should be called after an event occurs.
- Normally receives information about the event.

Callback

```
$('.button').click(function(){
 $(this).css({ 'color' : 'blue'});
});
```

Callbacks

```
fs.readFile('/etc/passwd',
function (err, data) {
  if (err) throw err;
  console.log(data);
});
```

Evented I/O

```
var read_stream = fs.createReadStream('README.md',
{encoding: 'ascii'});

read_stream.on("data", function(data){
  process.stdout.write(data);
});
read_stream.on("error", function(err){
  console.error("An error occurred: %s", err)
});
read_stream.on("close", function(){
  console.log("File closed.")
});
```

HTTP

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");

console.log('Server running at http://127.0.0.1:1337/');
```

TCP Server

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

var server = net.createServer(function (socket) {
  socket.write("Echo server\r\n");
  socket.pipe(socket);
});

server.listen(1337, "127.0.0.1");
```

Event Emitters

- Let you emit and listen for your own custom events.
- Emit method takes an event name and a list for parameters.
- To listen for an event specify a callback to event emitter with a function that takes the passed parameters.

Event Emitters

```
var events = require('events');
var tweetEmitter = new events.EventEmitter();

tweetEmitter.on(‘newTweets’, function(tweets){
  doSomething(tweets);
});

tweetEmitter.emit(‘newTweets’, someTweets);
```

NPM (Node Packet Manager)

- Packet manager designed to fetch and install node libraries.
- Installs all libraries in the current working directory unless specific otherwise.

Modules System

- Loading system/NPM installed modules is easy.
  - require('moduleName');
- When loading modules created on your own, specify a path.
  - require('./myModule');
  - require('/home/ryan/myModule.js');

Module Systems

- Modules export the functionality that should be made public.
  - Assigning properties to “exports” object.
  - Using export.modules to export a specific object.

Module Systems

```
var PI = Math.PI;

exports.area = function (r) {
  return PI * r * r;
};

exports.circumference = function (r) {
  return 2 * PI * r;
};
```

Module Systems

```
var circle = require('./circle.js');
console.log( 'The area of a circle of radius 4 is '
           + circle.area(4));
```

3rd Party Libraries

- Lots of code already written to do common things you would do in a web app
- Check NPM or github for modules

Express

- Light weight REST framework (similar to Sinatra in Ruby)
- Flexible enough to host the web parts of most applications you write in Node

Socket.io

- Great library for realtime communication between browser and server
- Handles the dirty work of Websockets for you (falls back to long polling, flash sockets, etc. automatically)
- Can tie in with Express to automatically serve the client side script required.

Few More

- nodeunit - unit testing
- ldap.js - create a LDAP interface over anything you want
- connect - middleware for web frameworks (used by Express)
- node.io - web page scraping framework

Demo

More Info

- nodejs.org
- IRC channel #nodejs on Freenode
- https://github.com/joyent/node/wiki/Community

Node.js

Ryan Farnell
@criscokid
ryan.farnell@me.com