Node.js

What is it?

- javascript became very popular on browsers.
- nodejs developers wanted to make javascript run on desktop.
- bundled javascript VM (google's V8) to allow one to create desktop programs in js.
- so now can run js on desktop!

- Also - huge number of libraries created.
- now can easily create a web server using some of these libraries.

Goals for this session

- show how to create and run simple js programs on a desktop.
- show how to create and use our own libraries (called modules on nodejs).
- js is very well suited for event based programming. we will show how to handle events.
- show how to create our own events.
- Next session – we will work on creating nodejs servers and web sockets.

Libraries we will use

standard lib

process.argv
console.log
setInterval(callback, time)
require(library)

path

extname

fs

readdir
readFile //read entire file
readFileSync
createReadStream //read in chunks

self check

1. What is nodejs?

2. what is the difference between readFile and createReadStream and readFileSync?

FIRST PROGRAM

Hello World

// CREATE JS FILE (show 1_basics.js)

1. plain old javascript!
2. no DOM related code << why??>>
3. node libraries accessible via require("<library");

// RUN CODE

prompt> node helloWorld.js

// DEBUGGING

6. node-inspector & // run this first
7. node –debug-brk a.js // run this next
8. now point to website

MODULES

Using Existing modules

```
var fs = require('fs');     // here's how
var path = require('path');
// typically an object or a function is returned.

var buf = fs.readdir(process.argv[2],
  function(err, data) {
      for (i = 0; i < data.length; i++) {
          var s = path.extname(data[i]);
          if (s === "." + process.argv[3]) {
              console.log(data[i]);
          }
      } // end of for
  } // end of callback function for readdir
```

Creating our own module

```
// FILE myModule.js
module.exports = function (dir, ext, callback) {
    var fs = require('fs');
    var path = require('path');
    var retValue =[];
   fs.readdir(dir, function(err, data) {
      if (err) return callback(err);
      retValue = data.filter(function(filename) {
         return path.extname(filename) === "." + ext;
      });
     callback(null, retValue);
   }); // end of callback to readdir

 }; // end of function
```

USERS of this module will need to provide dir, extension, and callback

Using the created module

```
var x = require('./mymodule');

// users need to provide dir, extension, callback
//
x(process.argv[2], process.argv[3], function(err, data)
  {
    if (err) return console.error ("error:", err);
      data.forEach(function(file) { // for each array elem
        console.log (file);
      });
  } // end of callback function
); // end of call to x
```

self check

1. How do we use a library on nodejs?
2. How do we create our own library?
3. what do we set module.exports to?
4. How do we use a library we created?

CALLBACK PATTERN

Synchronous i/o

Waits until i/o is done

EXAMPLE

```
var fs = require('fs');  // node's modular code

var buf = fs.readFileSync(process.argv[2]);
//WAIT!
var sArray = buf.toString().split("\n");
console.log(sArray.length-1); // print no of lines
```

Asynchronous i/o

NO WAIT until read is complete

EXAMPLE

```
var fs = require('fs');
var buf = fs.readFile(process.argv[2],
  function(err, data) { // CALLBACK
    var sArray = data.toString().split("\n");
    console.log(sArray.length-1);
  } );
// NO WAIT – DO THE NEXT INSTRN RIGHT AWAY
```

standard callback pattern

Callback function will look like

```
function (err, data) {
    if (err) { // handle error }
    else {
        // do something with data
     }
 });
```

This callback is called once when event happens (for example, i/o is complete)

self check

1. what is the purpose of a callback function?

2. what is the typical format of a callback function?

EVENT EMITTER PATTERN

Example

```
// Instead of only completed event, many events may be fired.
// Handlers can be registered for each event.
var fs = require('fs');
var file = fs.createReadStream('./' + process.argv[2]);

file.on('error', function(err) {
  console.log("Error:" + err);
  throw err;
});

file.on('data', function(data) {
  console.log("Data: " + data);
});
file.on ('end', function() {
  console.log("finished reading all of data");
});
```

1. createReadStream fires error, data, and end events

2. Using on function, we attach handlers.

EVENT EMITTER API

Event types (determined by emitter)

- error (special type)
- data
- end

API

- .on or .addListener
- .once (will be called at most once)
- .removeEventListener
- .removeAllEventListeners

Creating an Event Emitter

```
// file named myEmitter.js
var util = require('util'); // 1
var eventEmitter = require('events').EventEmitter; // 2
var Ticker = function() {
  var self = this;
  setInterval (function() {
      self.emit('tick'); // 3
    }, 1000) ;
};
util.inherits (Ticker, eventEmitter); //4
module.exports = Ticker;
```

Example (using Ticker)

```
// testingTicker
var Ticker = require("./myEmitter");

var ticker = new Ticker();

ticker.on ('tick', function() { // handler for 'tick' event
  console.log("Tick");
});
```

self check

1. can you register for the same event multiple times?
2. can you stop handling events?
3. how do you register for a different event?
4. To create your own events, what are four essential instructions that you need to use?