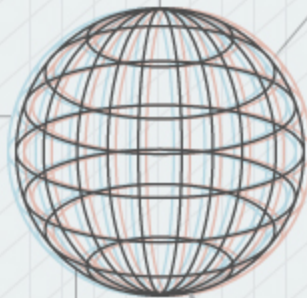




```
var readline = require('readline'),
    rl = readline.createInterface(process.stdin,
    process.stdout),
    prefix = 'OHAI> ';

rl.on('line', function(line) {
  switch(line.trim()) {
    case 'hello':
      console.log('world!');
      break;
    default:
      console.log('Say what? I might have heard \'' +
      line.trim() + '\'');
      break;
  }
  rl.setPrompt(prefix, prefix.length);
  rl.prompt();
}).on('close', function() {
  console.log('Have a great day!');
  process.exit(0);
});
```



Real Time Web

WITH NODE.JS

```
console.log('Starting directory: ' +
process.cwd());
try {
  process.chdir('/tmp');
  console.log('New directory: ' +
process.cwd());
}
catch (err) {
  console.log('chdir: ' + err);
}
```





INTRO TO NODE.JS

- LEVEL ONE -





WHAT IS NODE.JS?



Allows you to build scalable network applications using JavaScript on the server-side.

Node.js

V8 JavaScript Runtime

It's fast because it's mostly C++ code



INTRO TO NODE.JS





WHAT COULD YOU BUILD?



- **Websocket Server** *Like a chat server*
- **Fast File Upload Client**
- **Ad Server**
- **Any Real-Time Data Apps**



INTRO TO NODE.JS





WHAT IS NODE.JS NOT?



- A Web Framework
- For Beginners *It's very low level*
- Multi-threaded
You can think of it as a single threaded server



INTRO TO NODE.JS





OBJECTIVE: PRINT FILE CONTENTS



- Blocking Code

```
Read file from Filesystem, set equal to "contents"  
Print contents  
Do something else
```

- Non-Blocking Code

```
Read file from Filesystem  
  whenever you're complete, print the contents  
Do Something else
```

This is a "Callback"





BLOCKING VS NON-BLOCKING



- Blocking Code

```
var contents = fs.readFileSync('/etc/hosts');  
console.log(contents);  
console.log('Doing something else');
```

Stop process until complete



- Non-Blocking Code

```
fs.readFile('/etc/hosts', function(err, contents) {  
  console.log(contents);  
});  
console.log('Doing something else');
```





CALLBACK ALTERNATE SYNTAX



```
fs.readFile('/etc/hosts', function(err, contents) {  
  console.log(contents);  
});
```



Same as

```
var callback = function(err, contents) {  
  console.log(contents);  
}  
fs.readFile('/etc/hosts', callback);
```





BLOCKING VS NON-BLOCKING



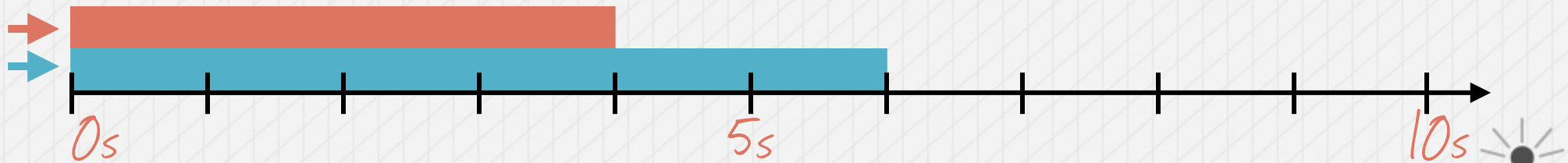
```
var callback = function(err, contents) {  
  console.log(contents);  
}  
  
fs.readFile('/etc/hosts', callback);  
fs.readFile('/etc/inetcfg', callback);
```



blocking



non-blocking





NODE.JS HELLO DOG

hello.js



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

How we require modules

```
http.createServer(function(request, response) {
```

```
  response.writeHead(200);
```

Status code in header

```
  response.write("Hello, this is dog.");
```

Response body

```
  response.end();
```

Close the connection

```
}).listen(8080);
```

Listen for connections on this port

```
console.log('Listening on port 8080...');
```

```
$ node hello.js
```

Run the server

```
$ curl http://localhost:8080
```

-----> Listening on port 8080...

-----> Hello, this is dog.





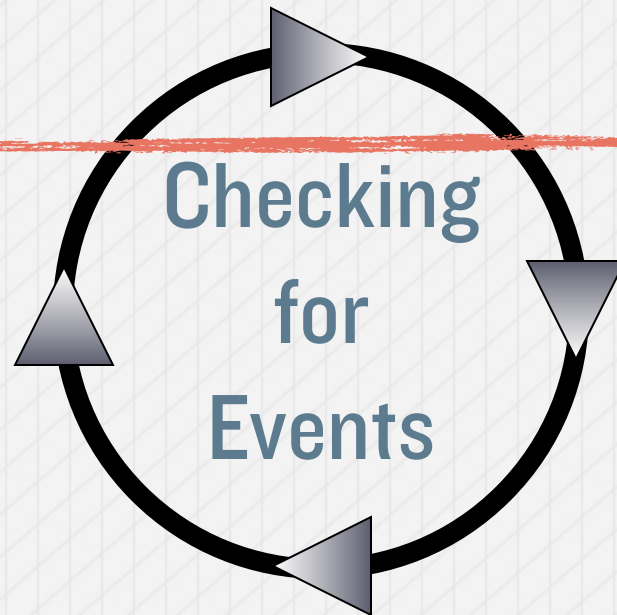
THE EVENT LOOP



```
var http = require('http');  
http.createServer(function(request, response) {  
  ...  
}).listen(8080);  
console.log('Listening on port 8080...');
```

Starts the Event Loop when finished

Run the Callback



Known Events

request





WHY JAVASCRIPT?



“JavaScript has certain characteristics that make it very different than other dynamic languages, namely that it has no concept of threads. Its model of concurrency is completely based around events.”

- Ryan Dahl

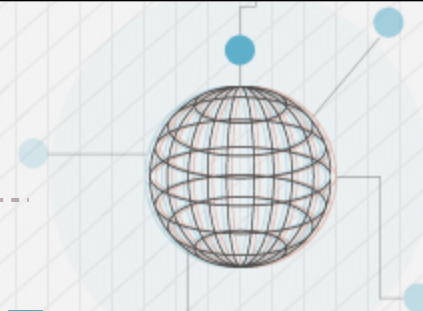


INTRO TO NODE.JS





THE EVENT LOOP



Event Queue

close

request

Checking
for
Events

Known Events

request

connection

close

Events processed one at a time





WITH LONG RUNNING PROCESS



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

```
http.createServer(function(request, response) {
```

```
  response.writeHead(200);
```

```
  response.write("Dog is running.");
```

```
  setTimeout(function() { Represent long running process
```

```
    response.write("Dog is done.");
```

```
    response.end();
```

```
  }, 5000); 5000ms = 5 seconds
```

```
}).listen(8080);
```





TWO CALLBACKS HERE



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

```
http.createServer(function(request, response) {
```

request

```
  response.writeHead(200);
```

```
  response.write("Dog is running.");
```

```
  setTimeout(function()
```

timeout

```
    response.write("Dog is done.");
```

```
    response.end();
```

```
  }, 5000);
```

```
}).listen(8080);
```





TWO CALLBACKS TIMELINE



- Request comes in, triggers request event
- Request Callback executes
- setTimeout registered
- ■ Request comes in, triggers request event
- Request Callback executes
- setTimeout registered

request

timeout

- triggers setTimeout event
- setTimeout Callback executes
- triggers setTimeout event
- setTimeout Callback



0s

5s

10s





WITH BLOCKING TIMELINE



Request comes in, triggers request event

Request Callback executes



setTimeout executed



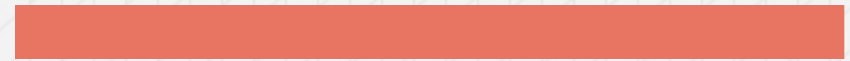
Request comes in, waits for server

triggers setTimeout event

setTimeout Callback executed

Request comes in

Request Callback executes



0s

5s

10s





TYPICAL BLOCKING THINGS



- Calls out to web services
- Reads/Writes on the Database
- Calls to extensions



INTRO TO NODE.JS





EVENTS

- LEVEL TWO -



.....

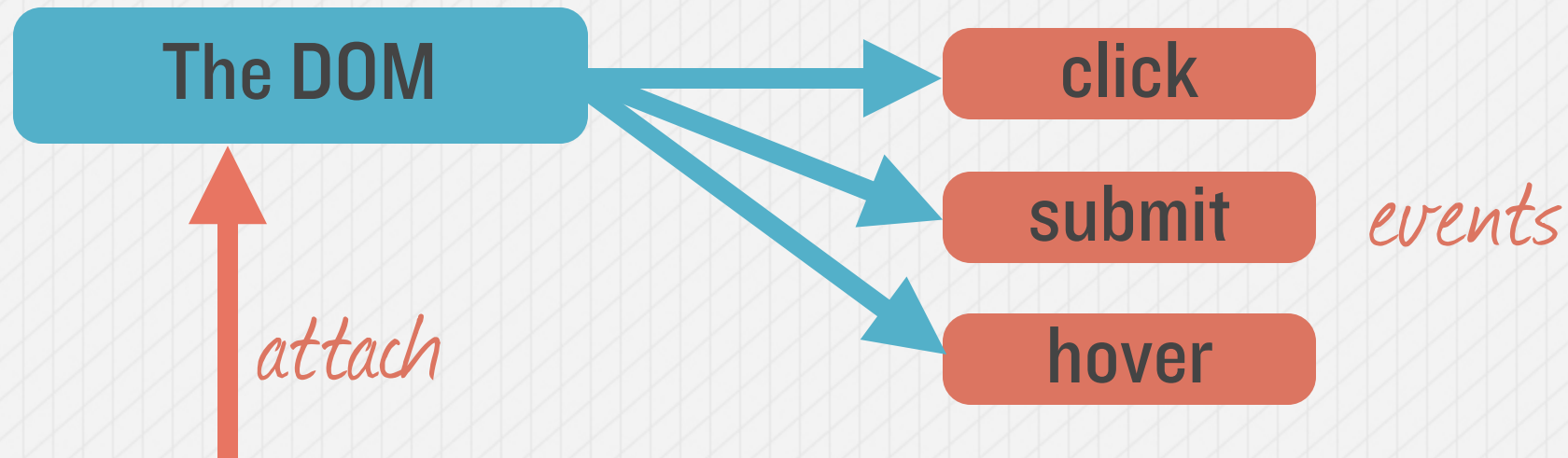




EVENTS IN THE DOM



*The DOM triggers Events
you can listen for those events*



```
$("p").on("click", function(){ ... });
```

When 'click' event is triggered



EVENTS





EVENTS IN NODE



Many objects in Node emit events

net.Server
EventEmitter

request
event

fs.readStream
EventEmitter

data
event



EVENTS





CUSTOM EVENT EMITTERS



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

```
var logger = new EventEmitter();
```

error

warn

info

```
logger.on('error', function(message){  
  console.log('ERR: ' + message);  
});
```

listen for error event

```
logger.emit('error', 'Spilled Milk');
```

-> ERR: Spilled Milk

```
logger.emit('error', 'Eggs Cracked');
```

-> ERR: Eggs Cracked



EVENTS

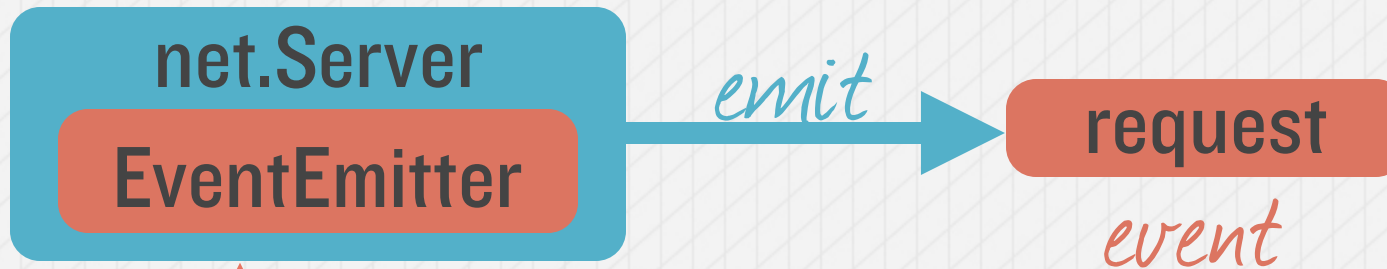




EVENTS IN NODE



Many objects in Node emit events



```
function(request, response){ .. }
```

When 'request' event is emitted



EVENTS





HTTP ECHO SERVER



```
http.createServer(function(request, response){ ... });
```

But what is really going on here?

<http://nodejs.org/api/>



EVENTS





BREAKING IT DOWN



```
http.createServer(function(request, response){ ... });
```

http.createServer([requestListener])

Returns a new web server object.

The `requestListener` is a function which is automatically added to the `'request'` event.

Class: http.Server

This is an `EventEmitter` with the following events:

Event: 'request'

```
function (request, response) { }
```

Emitted each time there is a request.



EVENTS





ALTERNATE SYNTAX



```
http.createServer(function(request, response){ ... });
```

Same as



```
var server = http.createServer();  
server.on('request', function(request, response){ ... });
```

*This is how we add
add event listeners*

Event: 'close'

```
function () { }
```

Emitted when the server closes.

```
server.on('close', function(){ ... });
```



EVENTS





MODULES

- LEVEL THREE -





REQUIRING MODULES



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

-----➔ http.js

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

-----➔ fs.js

How does 'require' return the libraries?

How does it find these files?

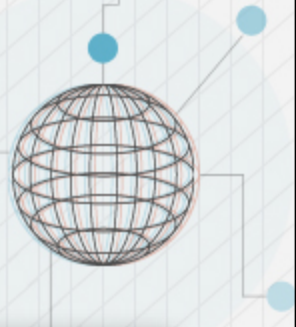


MODULES





LETS CREATE OUR OWN MODULE



custom_hello.js

```
var hello = function() {  
  console.log("hello!");  
}  
exports = hello;
```

custom_goodbye.js

```
exports.goodbye = function() {  
  console.log("bye!");  
}
```

app.js

exports defines what require returns

```
var hello = require('./custom_hello');  
var gb = require('./custom_goodbye');  
hello();  
gb.goodbye();
```

```
require('./custom_goodbye').goodbye();
```

If we only need to call once



MODULES





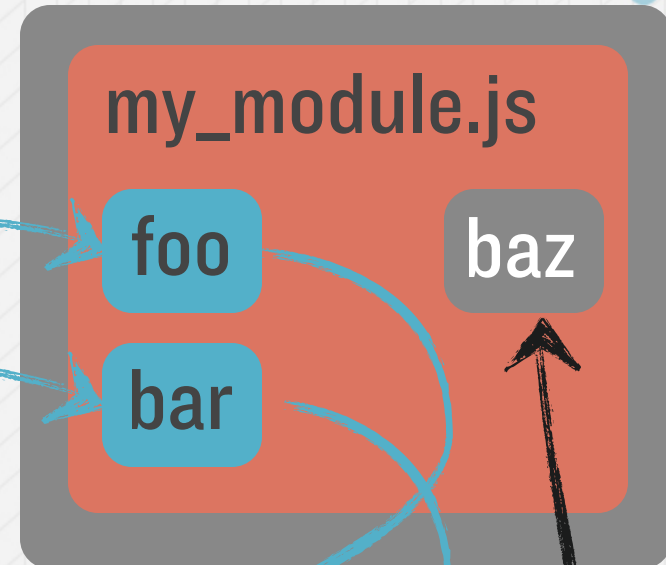
EXPORT MULTIPLE FUNCTIONS



my_module.js

```
var foo = function() { ... }  
var bar = function() { ... }  
var baz = function() { ... }
```

```
exports.foo = foo  
exports.bar = bar
```



app.js

```
var myMod = require('./my_module');  
myMod.foo();  
myMod.bar();
```

"private"



MODULES





MAKING HTTP REQUESTS



app.js

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

var message = "Here's looking at you, kid.";
var options = {
  host: 'localhost', port: 8080, path: '/', method: 'POST'
}

var request = http.request(options, function(response){
  response.on('data', function(data){
    console.log(data); logs response body
  });
});
request.write(message); begins request
request.end(); finishes request
```



MODULES





ENCAPSULATING THE FUNCTION



app.js

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

var makeRequest = function(message) {
  var options = {
    host: 'localhost', port: 8080, path: '/', method: 'POST'
  }

  var request = http.request(options, function(response){
    response.on('data', function(data){
      console.log(data);
    });
  });
  request.write(message);
  request.end();
}

makeRequest("Here's looking at you, kid.");
```



MODULES





CREATING & USING A MODULE



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

make_request.js

```
var makeRequest = function(message) {  
  ...  
}
```

```
exports = makeRequest;
```

```
var makeRequest = require('./make_request');
```

app.js

```
makeRequest("Here's looking at you, kid");  
makeRequest("Hello, this is dog");
```

Where does require look for modules?

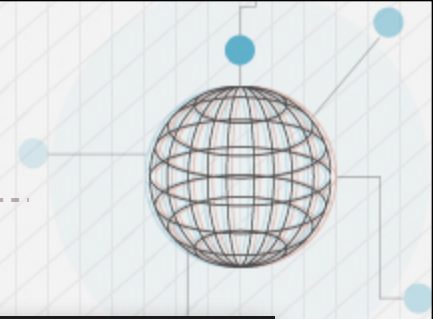


MODULES





REQUIRE SEARCH



```
var make_request = require('./make_request') look in same directory  
var make_request = require('../make_request') look in parent directory  
var make_request = require('/Users/user/nodes/make_request')
```

`/Home/user/my_app/app.js` *Search in node_modules directories*

```
var make_request = require('make_request')
```

- `/Home/user/my_app/node_modules/`
- `/Home/user/node_modules/make_request.js`
- `/Home/node_modules/make_request.js`
- `/node_modules/make_request.js`

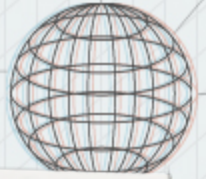


MODULES





NPM: THE USERLAND SEA



Package manager for node

- Comes with node
- Module Repository
- Dependency Management
- Easily publish modules
- “Local Only”



“Core” is small. “Userland” is large.



MODULES





INSTALLING A NPM MODULE



In /Home/my_app

```
$ npm install request
```

 <https://github.com/mikeal/request>

Installs into local node_modules directory

Home



my_app



node_modules



request

In /Home/my_app/app.js

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

Loads from local node_modules directory



MODULES





LOCAL VS GLOBAL



Install modules with executables globally

```
$ npm install coffee-script -g
```

global

```
$ coffee app.coffee
```

Global npm modules can't be required

```
var coffee = require('coffee-script');
```



```
$ npm install coffee-script
```

Install them locally

```
var coffee = require('coffee-script');
```

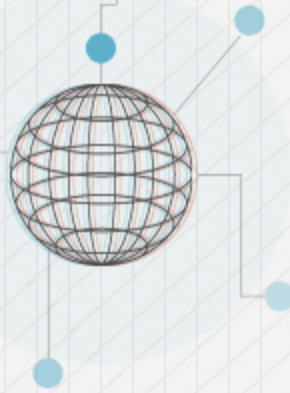


MODULES





FINDING MODULES



npm registry

Find packages...

or [browse](#)

Latest Updates		Most Depended
adstream-data	7 minutes ago	underscore
grover	7 minutes ago	coffee-script
tint	28 minutes ago	request
genetics	44 minutes ago	express
oauth2-node	49 minutes ago	async
kdtree	54 minutes ago	optimist
stateful	1 hour ago	colors
siq-mesh	1 hour ago	connect

npm command line

```
$ npm search request
```

eirikb.github.io/nipster

github search

github

Search...

Explore Repositories

Advanced Search

npm + github = Perfect rating!

Last update: 2014-08-21 11:11:22Z

Search: request

Package	Repo	Description	Author	Modified	Forks	Stargazers	Watchers
angular-animate	angular.js	This repo is for distribution on "bower". The source for this module is in the [main AngularJS repo]		2014-08-01 10:01:4	27508	27508	
request	request	Simplified HTTP request client.	Mikael Rogers	2014-06-01 07:07	5398	5398	
request	request	Simplified HTTP request client.	Mikael Rogers	2014-06-01 07:07	5398	5398	
hapijs-request	request	Simplified HTTP request client.	Mikael Rogers	2013-02-11 07:07	5398	5398	
deathbycaptcha2	request	deathbycaptcha API wrapper for Node.js	Eric Abovoff	2013-01-01 07:07	5398	5398	
kineticjs	KineticJS	Hi all! I will be taking a 3 month break from KineticJS (until July 2014) to focus on other endeavors.	Eric Bowell	2014-05-01 558	2736	2736	
requester	node-requester	swiss army knife for requests	Chad Scia	2013-10-01 10	2591	2591	
pg-arrays	node-postgres	PostgreSQL client - pure javascript & libpq with the same API // This is the fork of Brian's library for	Brian Carlson	2013-10-11 262	1762	1762	
recordrtc	WebRTC-Recorder	Records audio/video separately as wav/webm. POST both files in single HttpPost-Request to Node.js	Muaz Khan	2014-07-01 872	1762	1762	
jquery-mockjax	jquery-mockjax	Mockjax. The jQuery Mockjax Plugin provides a simple and extremely flexible interface for mocking	Jonathan Sharp	2014-04-21 233	1197	1197	
request	request	route incoming http requests to http servers	James Halliday	2014-03-01 62	746	746	
request	request	A wrapper for asynchronous http requests	Dustin Diaz	2014-07-31 117	614	614	
multi-node	multi-node	Multi-node provides launching of multiple NodeJS processes for TCP/HTTP serving. With multi-node	Kris Zyp	2012-07-01 37	437	437	
hellojs	hello.js	A clientside Javascript library for standardizing requests to OAuth2 web services (and OAuth1 - with	Andrew Dodson	2014-05-11 75	371	371	
hyperquest	hyperquest	make streaming http requests	James Halliday	2014-03-21 30	306	306	
httpinvoke	httpinvoke	A no-dependencies HTTP client library for browsers and Node.js with a promise-based or Node.js-s	Vytautas Jakutis	2014-08-01 12	297	297	



MODULES





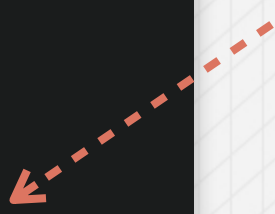
DEFINING YOUR DEPENDENCIES



my_app/package.json

```
{  
  "name": "My App",  
  "version": "1",  
  "dependencies": {  
    "connect": "1.8.7"  
  }  
}
```

version number



```
$ npm install
```

Installs into the node_modules directory

my_app



node_modules



connect



MODULES





DEPENDENCIES



my_app/package.json

```
"dependencies": {  
  "connect": "1.8.7"  
}
```

No conflicting modules!

Installs sub-dependencies

my_app



node_modules



connect

connect



node_modules



qs

connect



node_modules



mime

connect



node_modules



formidable



MODULES



SEMANTIC VERSIONING

"connect": "1.8.7"

Major *Minor* *Patch*
1 . 8 . 7

Ranges

"connect": "~1"



$\geq 1.0.0$ $< 2.0.0$

"connect": "~1.8"



≥ 1.8 $< 2.0.0$

"connect": "~1.8.7"



$\geq 1.8.7$ $< 1.9.0$

Safest

"connect": "^1.8.7"



$\geq 1.8.7$ $< 2.0.0$

Default

<http://semver.org/>



MODULES

