GoSquared

Node.JS Help Sheet

Getting Started

Node.JS is evented I/O for V8 JavaScript. It is asynchronous in nature, with handlers to I/O and other events being function callbacks. It is particularly suited to distributed computing environments with high concurrency.

node script.js	Run script
npm install <package>	Install package with npm

Globals

var variable	Initialize variable local to module
process	Properties & methods for current process
console	Used to print to stdout & stderr
require()	To require modules
require.resolve	Lookup location of module
require.paths	Paths to search when requiring modules
_filename	File name of script being executed
_dirname	Directory name of script being executed
module	Reference to current module
setTimeout(), clearTimeout()	
setInterval(), clearInterval()	

Modules

stdio

Object for printing to stdout and stderr, like in a browser.

console.log(string)	Print to stdout with newline
console.error(string)	Same as console.log() but to stderr
console.time(label)	Set time marker
console.timeEnd(label)	Finish timer, record output
console.trace()	Print stack trace to stderr of current position

Process

Global object. Instance of *EventEmitter*

Events:

process.on(SIGNAL, callback)	Signal events emitted when process receives a signal
exit	Process is about to exit
uncaughtException	Exception bubbled back to event loop

Properties: **process.stdout**, **processs.stderr**, **process.stdin**, **process.argv**, **process.env**, **process.pid**

Modules (continued)

util

Useful methods:

util.debug(message)	Synchronous console.error(message)
util.log(message)	Print timestamped message to stdout

events

Callback functions executed when events occur are *listeners*. *emitter* is an instance of *EventEmitter*.

emitter.on(event, listener)	Add a listener for *event*
emitter.once(event, listener)	Fire listener once
emitter.removeListener(event,listener)	Remove a listener
emitter.removeAllListeners(event)	Remove all listeners
emitter.emit(event, [[arg1], [arg2], [...]])	Execute listeners for this event with supplied args

net

Asynchronous network wrapper for creating streams.

net.Server:

net.createServer([options], [connectionListener])	Create TCP server. Returns *net.Server*
server.listen(port, [host], [callback])	Bind on host:port. *listener* is executed when bound
server.listenFD(fd)	Listen on file descriptor *fd*
server.close()	Stop accepting new connections

net.Socket:

new net.Socket([{fd: file descriptor, type: socket type, allowHalfOpen: bool}])	Construct new socket object
socket.connect(port, [host], [callback])	Open connection to socket
socket.bufferSize	Number of characters in internal write buffer
socket.write(data, [encoding], [callback])	Send data on socket
socket.end()	Send FIN packet
socket.pause()	Pause reading of data

event

Emitted when:

connect	Socket connection established
data	Data is received
end	Other end of socket sent FIN packet
timeout	Timed out from inactivity
drain	Write buffer has become empty
error	Error has occurred. close event emitted after
close	Socket fully closed