1 Session

Core

Node Training

nearForm

API Stability

- Overview Core modules
- Review of important points
  - Ultimately you’ll need to read the Core API documentation through end-to-end at least once

Core Modules

require	St.	Name	Description
assert	5	Assertions	Unit test assertions
buffer	3	Buffer	raw binary data
child_process	3	Child Process	Run child processes
cluster	1$^{0.10}$ 2$^{0.12}$	Cluster	Run multiple processes
console*	4	Console	Standard streams
crypto	2	Crypto	Crypto algorithms

Core Modules

require	St.	Name	Description
dns	3	DNS	Domain name lookup
domain	2	Domain	Collate IO errors
events	4	Events	Event handling
fs	3	File	Filesystem operations
http	3	HTTP	HTTP operations
https	3	HTTPS	Secure HTTP

nearForm

Core Modules

require	St.	Name	Description
module	_	Module	Module loading (internal)
net	3	Network	TCP Networking
os	4	Operating Sys.	Operating system info.
path	3	File Paths	File path operations
process*	3	Process	Process info
punycode	2	Punycode	Domain name I18N

nearForm

Core Modules

require	St.	Name	Description
querystring	3	Query String	Parse and build query str.
readline	2	Readline	Read streams line-byline
repl	3	REPL	REPL with context
stream	$2^{0.10}$ $3^{0.12}$	Stream	data processing
smalloc	$1^{0.12}$	Smalloc	direct mem access (v0.12)
string_decoder	3	Str. Decoder	Decode binary to string

nearForm

Core Modules

require	St.	Name	Description
timers	5	Timers	Encrypt streams
tls	3	TLS/SSL	Encrypt streams
tty	2	TTY	Terminal info
dgram	3	UDP	UDP networking
url	3	URL	URL parse and build
util	4	Utilities	General utilities

nearForm

Core Modules

require	St.	Name	Description
vm	$2^{0.10}$ $3^{0.12}$	VM	Virtual runtime contexts
zlib	3	ZLIB	Compression utilities

nearForm

API Stability

- Stability: 0 – Deprecated
- Stability: 1 – Experimental
- Stability: 2 – Unstable
- Stability: 3 – Stable
- Stability: 4 – API Frozen
- Stability: 5 – Locked

Core Module: assert

```
assert(false) //throw
assert.equal(1, 2) //throw
assert.strictEqual(1, '1') //throw
```

- Fundamental assertions, nothing fancy
- `assert` doesn't just have to be used for tests
  - It can be a nice way to throw on bad input
- The Chai library has extended assertion methods (http://chaijs.com)
  - It also has fluent API's that help to create more coherent tests

nearForm

Core Module: buffer

- Containers for binary data
  - because JavaScript does not have great support
  - Browser equivalent is ArrayBuffers, these have a significantly different interface
- The `buffer` module exports a `Buffer` constructor
- The `Buffer` constructor is a global, so just use it directly
  - It also does **not** require use of `new`.
- Streams deal in Buffers
  - data chunks must be explicitly converted to strings to string manipulation

Core Module: child_process

- `spawn`: launches a sub process, which you optionally detach
- `fork`: like spawn, but for Node processes
  - let's your communicate with your children over IPC
- `exec`: runs a shell command that you pass as a string
  - will buffer `stdin` and `stderr`, and give them to you via callback once complete
  - the object returned from exec also has stdin and stderr streams
    - however beware - exec stdio streams will stop emitting data after a point
    - if a process sends a lot of data to stdio, and access to this data is important: use `spawn`, even for short lived processes

nearForm

Core Module: cluster

- Easily scale process over CPU cores
- Essentially broken in Node <= 0.10 - the implementation relied too heavily on the OS, which turns out to be poor at clustering.
- Node 0.12 uses a round-robin approach, however this hasn't seen much action in the wild as yet
- This is an operations issue. Arguably, in many cases Node should be scaled (both horizontally and vertically) by mature configuration driven tools rather than programatically.

Core Module: console

- Most operations are synchronous
  - normally so fast it's not problem
  - but for high volume logging, take care
    - use bunyan
- `console.trace`
  - prints a stack trace of current position
- `console.time`
  - gives you a simple timer

nearForm

Core Module: crypto

- One of Node's earliest and most unfriendly APIs!
- Great for quickly prototyping ideas.
- WARNING: careless use will burn the CPU
  - consider moving into a micro-service
    - consider re-writing the micro-service in C
  - example: password hashing

Core Module: dns

- Uses C-Ares for resolution
  - cross-platform C library for DNS
- `dns.lookup` uses system calls to be consistent with what your machine would return
  - this is slower
- Use `dns.resolve` if you want speed

nearForm

Core Module: domain

- Not as useful as you think.
- The use-case is:
  - chain of I/O asynchronous calls
  - have one error handler
- Domains are not held in high regard

Core Module: domain

- The concept is under question, and the implementation is poor
  - A big overhead cost for using domains
  - Not all errors are correctly caught (bug)
- io.js (which in some form is extremely likely to feed into future Node version), has deprecated `domains`
- A better approach is to not worry about unexpected throws, allow the process to crash and restart (use this approach alongside a failover and/or microservices strategy).

nearForm

Core Module: events

- The `events` module exports an object with a single method: The `EventEmitter` constructor.
- The `EventEmitter` is an asynchronous logic decoupling abstraction, it's used extensively in core code and is the basis of streams.

Core Module: events

Inherit from EventEmitter with `util.inherits`

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

util.inherits(MyEmitter, EventEmitter)

function MyEmitter(){
  EventEmitter.call( this )
}

var foo = new MyEmitter()
foo.on('bar', function(arg1, arg2){
  console.log(arg1, arg2) // a b
})

foo.emit('bar','a','b')
```

nearForm

Core Module: fs

- Based closely on UNIX file system C API
- File system C docs can be helpful, particularly with error codes (or stackoverflow...)
- Only use `Sync` suffixed methods during initialization, use async methods otherwise

nearForm

Core Module: http

- Basic HTTP server and client
- All the fundamental primitives to build a web server
- `https` module works exactly the same except it takes an additional configuration object for keys and certificates

nearForm

Core Module: process

- For additional debug info, especially with many services and machines, use:
  - `process.versions` to get full details of Node build
  - `process.title` to set the name of your process
- `process.hrtime` high resolution timing at nanosecond level

Core Module: punycode

- Used to convert international domain names into ASCII.
  - e.g. 點看.com resolves to xn--c1yn36f.com
- Why is this in core?
  - It's related to networking

Core Module: tty

- Container for the standard input and standard output terminal streams
- `process.stdin` is a `tty.ReadStream`
- Are you running inside a terminal?

```
$ node -p -e "process.stdout.isTTY" #true
$ node -p -e "process.stdout.isTTY" | cat #false
```

Core Module: dgram

- User Datagram Protocol - thin layer over IP.
  - TCP is a reliable protocol, resending packets happens at the protocol layer
  - UDP is an unreliable protocol, application layer deals with lost packets
- No concept of server/client, just clients (so... peers), data is sent directly to a socket
- On Node <= 0.10, dgram doesn't work well with the clustering module (especially on Windows)

nearForm

Core Module: url

```
var o = url.parse('http://google.com');
var s = url.format(o);

url.resolve('http://example.com/one', '/two')
// 'http://example.com/two'
```

- Use the `url` module to parse URLs
  - don't use custom regular expressions!

nearForm

Core Module: util

- `util.inspect` is very useful
  - will give you a properties breakdown of most objects
  - if you add an inspect method to your own objects, you can define a custom string description - great for debugging

Core Module: vm

- Run sandboxed code from a string
  - `runInNewContext` takes an object for global context, code only has access to whatever is supplied in the object
- Unstable in Node <= 0.10
- Stable in Node 0.12+
- Running untrusted code in-process is difficult to do safely
  - A separate clean process plus `vm.runInNewContext`, using Node 0.12+ is the best approach but still... proceed with caution

nearForm

Core Module: vm

- Run sandboxed code from a string
  - `runInNewContext` takes an object for global context, code only has access to whatever is supplied in the object
- Unstable in Node <= 0.10
- Stable in Node 0.12+
- Running untrusted code in-process is difficult to do safely
  - A separate clean process plus `vm.runInNewContext`, using Node 0.12+ is the best approach but still... proceed with caution

nearForm

Core Module: zlib

- Compression module
- Streaming and Sync API's
  - Very few use cases for sync compression
- gzip or deflate
- deflate is faster (gzip actually contains raw deflate output), however browser consumption of deflate compressed data is inconsistent, some accept raw deflate data, some accept deflate data with checksum and headers, best to `Accept-Encoding: gzip`

nearForm