State of Node

streams

Streams 3 `(0f8de5e)`, paraphrased from Isaac:

> “Streams 3 works the way people thought streams 2 should have worked.”

`cork()`, `uncork()` and `_writev()` `(60ed2c5)`

writev example

```
server.on('connection', function(c) {
  // Buffer all writes until uncore() or end()
  c.cork();
  c.write(chunk1);
  c.write(chunk2);
  e.end(chunk3);
});
```

net

Bind to “`::`” by default, if possible `(2272052)`

Emit dns `'lookup'` `event` `(b3d1e50)`

http

Chunked encoding defaults to writev `(2eddd74)`

Add `request.flush()` `(bd24ab2)`

`message.rawHeaders` delivers headers exactly as they were received `(e6c81bd6)`

crypto

RSA encryption/decryption `(9d3faf4)`

ECDH support `(6e453fa)`

Custom pbkdf2 digest methods `(74d9aa4)`

Signing private key accepts passphrase `(f755ecf)`

url

URL parsing now follows the whatwg spec `(6120472)`

```
> url.parse('https://good.com+.evil.org/');
// before
host: 'good.com'
path: '/+.evil.org/'
// now
host: 'good.com+.evil.org'
path: '/'
```

icu

Can side-load icu data via `--icu-data-dir=` and has build support for small (i.e. en) and full icu `(ac2857b1)`

All distributed binaries will come small icu, but source will build without, by default (pre v0.12.0)

buffer

fill() now returns buffer instance `(6af8788)` and accepts multibyte strings `(4b40358)`

`SlowBuffer` returns unsliced `Buffer` instance `(3a2f273)`

Generic `(read|write)(U)Int()` methods for 24, 40 and 48 bit reads/writes `(83d7d9e)`

`compare()` two buffers for equality `(226f98a3)`

smalloc

New API for allocating memory on any Object

Simple example:

```
var smalloc = require('smalloc');
// Allocate memory on new empty object
var obj = smalloc.alloc(16, smalloc.Types.Double);
```

```javascript
// Another small example
var smalloc = require('smalloc');

function ImageData(rows, cols) {
  // Bypass checks and all that
  this.rows = rows;
  this.cols = cols;
  smalloc.alloc(rows * cols, this, smalloc.Types.Double);
}

ImageData.prototype.doStuff = function doStuff() { /* ... */ };

var m = new ImageData(5, 5);
```

async hooks

Async Listener is dead for now `(b655955)`

Instead, introduced low level hooks for community experimentation while the final user facing API is hashed out `(709fc16)`

```javascript
var async_wrap = process.binding('async_wrap');
var flags = {}, uid = 0;

function init() {
  // Called when class is instantiated
  this._asyncQueue = { uid: ++uid };
}

// Called just before callback is called
function pre() { }

// Called just after callback is called
// (unless callback threw exception)
function post() { }

// Only run once
async_wrap.setupHooks(flags, init, pre, post);
```

misc

`nextTick()` `maxTickDepth` has been removed `(0761c90)`, no longer prints warning and allows infinite recursion `(5757642)`

process' `'beforeExit'` event `(a2eeb43)`

`{spawn,exec,execFile}Sync()` `(e8df267)`

cluster scheduling policy (e.g. round-robin) `(e72cd41)`

future of node

Platform agnostic performance probes

“Better” debug-ability

Time to visit the performance rabbit hole

(not official) future of node

New API interface options are here, and more are coming, that don’t belong in Node

Simplification of the API to allow user-land interfaces to be written that don’t incur a performance penalty because of forced abstractions