Crawling with NodeJS

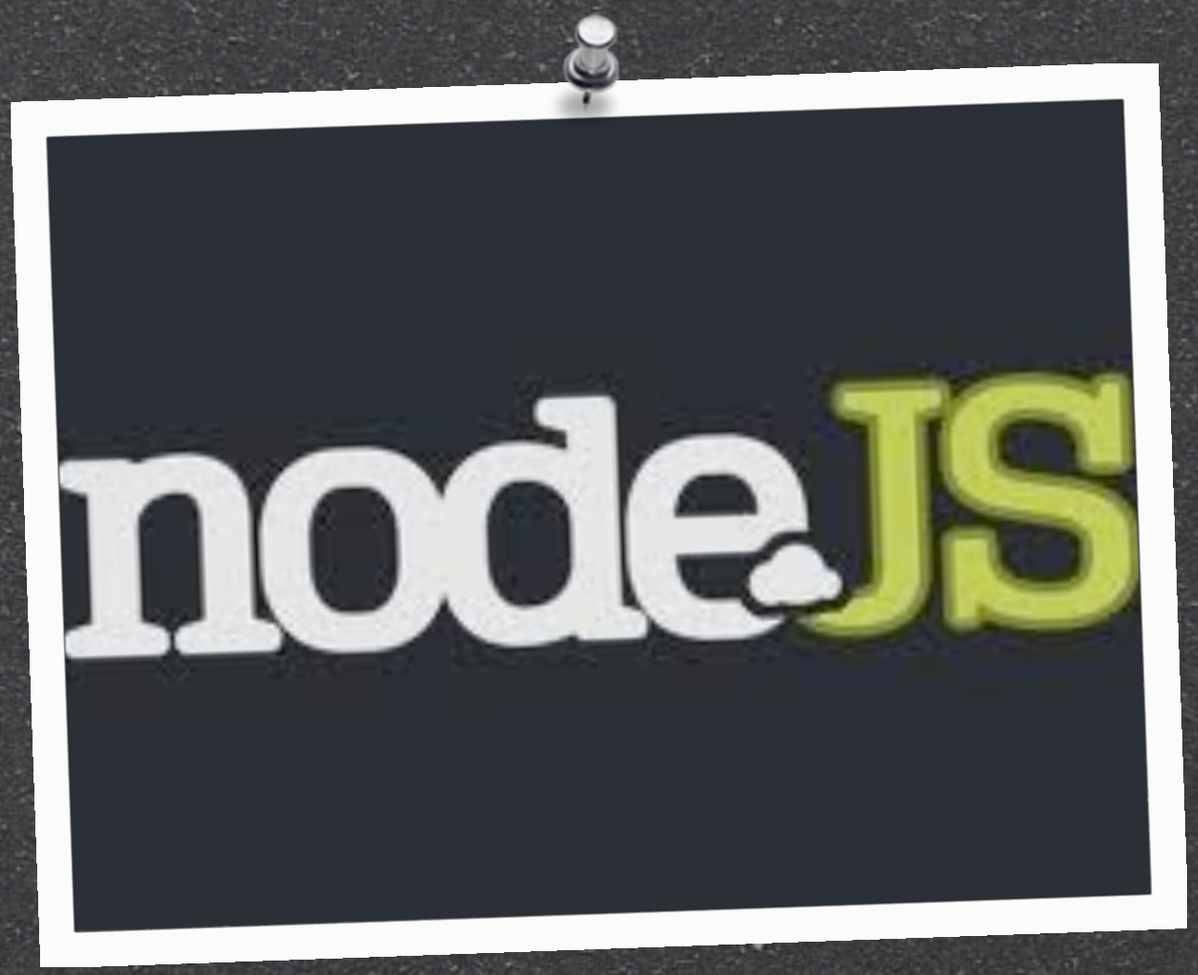
JSMeetup2@Paris 24.11.2010

@sylvinus

Crawling?

Web crawling

- Grab
- Process
- Store


node.JS

?

NodeJS

- Server-side Javascript
- Async / Event-driven / Reactor pattern
- Small stdlib, Exploding module ecosystem

Why?

- Boldly going where no one has g...
- Threads vs. Async
- ZOMG Server-side CSS3 selectors!

Apricot

- https://github.com/silentrob/Apricot
- HTML/DOM Parser, inspired by Hpricot
- Sizzle + JSDOM + XUI

Problems w/ Apricot

```
if (file.match(/^https?:\/\//)) {
    var urlInfo = url.parse(file, parseQueryString=false),
    host = http.createClient(((urlInfo.protocol === 'http:') ? 80 : 443),
urlInfo.hostname),
    req_url = urlInfo.pathname;

    if (urlInfo.search) {
      req_url += urlInfo.search;
    }
    var request = host.request('GET', req_url, { host: urlInfo.hostname });

    request.addListener('response', function (response) {
      var data = '';
      response.addListener('data', function (chunk) {
        data += chunk;
      });
      response.addListener("end", function() {
        fnLoaderHandle(null, data);
      });
    });
    if (request.end) {
      request.end();
    } else {
      request.close();
    }

} else {
    fs.readFile(file, encoding='utf8', fnLoaderHandle);
}
```

Problems w/ Apricot

- No advanced HTTP client in Node's lib
- npm install request
- https + redirects + buffering

Problems w/ Apricot

```
Apricot.parse("<p id='test'>An HTML Fragment</p>", function(doc) {
  doc.find("selector");     // Populates internal collection, See Sizzle selector syntax (rules)
  doc.each(callback);       // Itterates over the collection, applying a callback to each match
(element)
  doc.remove();             // Removes all elements in the internal collection (See XUI Syntax)

  doc.inner("fragment");    // See XUI Syntax
  doc.outer("fragment");    // See XUI Syntax
  doc.top("fragment");      // See XUI Syntax
  doc.bottom("fragment");   // See XUI Syntax
  doc.before("fragment");   // See XUI Syntax
  doc.after("fragment");    // See XUI Syntax
  doc.hasClass("class");    // See XUI Syntax
  doc.addClass("class");    // See XUI Syntax
  doc.removeClass("class"); // See XUI Syntax

  doc.toHTML;               // Returns the HTML
  doc.innerHTML;            // Returns the innerHTML of the body.
  doc.toDOM;                // Returns the DOM representation

  // Most methods are chainable, so this works
  doc.find("selector").addClass('foo').after(", just because");

});
```

Problems w/ Apricot

- XUI api?!
- jQuery please :)

- require("jsdom").jQueryify !!

```
var jsdom  = require("jsdom"),
    window = jsdom.jsdom().createWindow();

jsdom.jQueryify(window, 'http://code.jquery.com/jquery-1.4.2.min.js' , function() {
  window.$('body').append('<div class="testing">Hello World, It works</div>');
  console.log(window.$('.testing').text());
});
```

Concurrency?

- https://github.com/coopernurse/node-pool
- npm install generic-pool

generic-pool

```
// Create a MySQL connection pool with
// a max of 10 connections and a 30 second max idle time
var poolModule = require('generic-pool');
var pool = poolModule.Pool({
    name     : 'mysql',
    create   : function(callback) {
        var Client = require('mysql').Client;
        var c = new Client();
        c.user     = 'scott';
        c.password = 'tiger';
        c.database = 'mydb';
        c.connect();
        callback(c);
    },
    destroy  : function(client) { client.end(); },
    max      : 10,
    idleTimeoutMillis : 30000,
    log : false
});

// borrow connection - callback function is called
// once a resource becomes available
pool.borrow(function(client) {
    client.query("select * from foo", [], function() {
        // return object back to pool
        pool.returnToPool(client);
    });
});
```

So what?

Apricot - XUI + jQuery
+ request + generic-pool
+ qunit + ?

=

??

Simple API?

```
var Crawler = require("node-crawler").Crawler;

var c = new Crawler({
    "maxConnections":10,
    "timeout":60,
    "defaultHandler":function(error,result,$) {
        $("#content a:link").each(function(a) {
            c.queue(a.href);
        })
    }
});

c.queue(["http://jamendo.com/","http://tedxparis.com", ...]);
```

```
c.queue([{
    "uri":"http://parisjs.org/register",
    "method":"POST"
    "handler":function(error,result,$) {
        $("div:contains(Thank you)").after(" very much");
    }
}]);
```

Name contest! :)

- node-crawler ?
- Crawly ?
- ?????

Thanks!

- First code on github tonight
- Help & Forks welcomed
- (We're hiring HTML5/JS hackers ;-)
- Also, http://html5weekend.org/