**Node.js**

Node.js is a very powerful JavaScript-based framework/platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.

* Node.js is an open source server framework
* Node.js is free
* Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
* Node.js uses JavaScript on the server
* Node.js can generate dynamic page content
* Node.js can create, open, read, write, delete, and close files on the server
* Node.js can collect form data
* Node.js can add, delete, modify data in your database.

**Node.js file**

* Node.js files contain tasks that will be executed on certain events
* A typical event is someone trying to access a port on the server
* Node.js files must be initiated on the server before having any effect
* Node.js files have extension ".js"

**Angular.js 1.x**

AngularJS is what HTML would have been, had it been designed for building web-apps. Declarative templates with data-binding, MVW, MVVM, MVC, dependency injection and great testability story all implemented with pure client-side JavaScript.

* AngularJS extends HTML with new attributes.
* AngularJS is perfect for Single Page Applications (SPAs).
* AngularJS is easy to learn

Ex.

<!DOCTYPE html>  
<html lang="en-US">  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>Name : <input type="text" ng-model="name"></p>  
  <h1>Hello {{name}}</h1>  
</div>  
  
</body>  
</html>

* AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag.
* AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

**MongoDB**

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

**Amazon S3**

Amazon Simple Storage Service (S3) can solve all these problems for us. S3 *buckets* are units of storage for storing files. In this guide, I’m going to show you how to use a Node.js backend to issue *signed URL’s*to authenticated users, each of which will allow our Swift application to upload a single file directly to the S3 bucket. The uploaded image will be available at its own URL.

**Scrapping the web with node.js**

Before web based API's became the prominent way of sharing data between services we had web scraping. Web scraping is a technique in data extraction where you pull information from websites.

There are many ways this can be accomplished. It can be done manually by copy and pasting data from a website, using specialized software, or building your own scripts to scrape data. In this tutorial, we will be showing you how to build a **simple web scraper** that gets some general movie information from IMDB.

**Application**

Our web scraper is going to be very minimalistic. The basic flow will be as follows:

1. Launch web server
2. Visit a URL on our server that activates the web scraper
3. The scraper will make a request to the website we want to scrape
4. The request will capture the HTML of the website and pass it along to our server
5. We will traverse the DOM and extract the information we want
6. Next, we will format the extracted data into a format we need
7. Finally, we will save this formatted data into a JSON file on our machine

Server.js file

var express = require('express');

var fs = require('fs');

var request = require('request');

var cheerio = require('cheerio');

var app = express();

app.get('/scrape', function(req, res){

//All the web scraping magic will happen here

})

app.listen('8081')

console.log('Magic happens on port 8081');

exports = module.exports = app;

Making the request

Now that we have the boilerplate of the application done, let's get into the fun stuff. We are now on Step 3, and that is **making the request to the external website** we would like to scrape.

var express = require('express');

var fs = require('fs');

var request = require('request');

var cheerio = require('cheerio');

var app = express();

app.get('/scrape', function(req, res){

// The URL we will scrape from - in our example Anchorman 2.

url = 'http://www.imdb.com/title/tt1229340/';

// The structure of our request call

// The first parameter is our URL

// The callback function takes 3 parameters, an error, response status code and the html

request(url, function(error, response, html){

// First we'll check to make sure no errors occurred when making the request

if(!error){

// Next, we'll utilize the cheerio library on the returned html which will essentially give us jQuery functionality

var $ = cheerio.load(html);

// Finally, we'll define the variables we're going to capture

var title, release, rating;

var json = { title : "", release : "", rating : ""};

}

})

})

app.listen('8081')

console.log('Magic happens on port 8081');

exports = module.exports = app;

Formatting and Extracting

Now that we have the data extracted, let's format it and save it to our project folder. We have been storing our extracted data to a variable called json. Let's save the data in this variable to our project folder. If we didn't know what this was for, this library gives us access to our computer's file system.

var express = require('express');

var fs = require('fs');

var request = require('request');

var cheerio = require('cheerio');

var app = express();

app.get('/scrape', function(req, res){

url = 'http://www.imdb.com/title/tt1229340/';

request(url, function(error, response, html){

if(!error){

var $ = cheerio.load(html);

var title, release, rating;

var json = { title : "", release : "", rating : ""};

$('.header').filter(function(){

var data = $(this);

title = data.children().first().text();

release = data.children().last().children().text();

json.title = title;

json.release = release;

})

$('.star-box-giga-star').filter(function(){

var data = $(this);

rating = data.text();

json.rating = rating;

})

}

// To write to the system we will use the built in 'fs' library.

// In this example we will pass 3 parameters to the writeFile function

// Parameter 1 : output.json - this is what the created filename will be called

// Parameter 2 : JSON.stringify(json, null, 4) - the data to write, here we do an extra step by calling JSON.stringify to make our JSON easier to read

// Parameter 3 : callback function - a callback function to let us know the status of our function

fs.writeFile('output.json', JSON.stringify(json, null, 4), function(err){

console.log('File successfully written! - Check your project directory for the output.json file');

})

// Finally, we'll just send out a message to the browser reminding you that this app does not have a UI.

res.send('Check your console!')

}) ;

})

app.listen('8081') console.log('Magic happens on port 8081'); exports = module.exports = app;

With this code in place you are set to scrape and save the scraped data. Let's start up our node server, navigate to http://localhost:8081/scrape and see what happens.

* If everything went smoothly our browser should display a message telling we to check our command prompt.
* When we check your command prompt you should see a message saying that your file was successfully written and that you should check your project folder.
* Once we get to our project folder we should see a new file created called output.json.
* Opening this file, will give we a nicely formatted JSON document that will have the extracted data.

**Store/Display an image in mongodb using mongoose/express**

|  |
| --- |
| /\*\* |
|  | \* Module dependencies |
|  | \*/ |
|  |  |
|  | var express = require('express'); |
|  | var fs = require('fs'); |
|  | var mongoose = require('mongoose'); |
|  | var Schema = mongoose.Schema; |
|  |  |
|  | // img path |
|  | var imgPath = '/path/to/some/img.png'; |
|  |  |
|  | // connect to mongo |
|  | mongoose.connect('localhost', 'testing\_storeImg'); |
|  |  |
|  | // example schema |
|  | var schema = new Schema({ |
|  | img: { data: Buffer, contentType: String } |
|  | }); |
|  |  |
|  | // our model |
|  | var A = mongoose.model('A', schema); |
|  |  |
|  | mongoose.connection.on('open', function () { |
|  | console.error('mongo is open'); |
|  |  |
|  | // empty the collection |
|  | A.remove(function (err) { |
|  | if (err) throw err; |
|  |  |
|  | console.error('removed old docs'); |
|  |  |
|  | // store an img in binary in mongo |
|  | var a = new A; |
|  | a.img.data = fs.readFileSync(imgPath); |
|  | a.img.contentType = 'image/png'; |
|  | a.save(function (err, a) { |
|  | if (err) throw err; |
|  |  |
|  | console.error('saved img to mongo'); |
|  |  |
|  | // start a demo server |
|  | var server = express.createServer(); |
|  | server.get('/', function (req, res, next) { |
|  | A.findById(a, function (err, doc) { |
|  | if (err) return next(err); |
|  | res.contentType(doc.img.contentType); |
|  | res.send(doc.img.data); |
|  | }); |
|  | }); |
|  |  |
|  | server.on('close', function () { |
|  | console.error('dropping db'); |
|  | mongoose.connection.db.dropDatabase(function () { |
|  | console.error('closing db connection'); |
|  | mongoose.connection.close(); |
|  | }); |
|  | }); |
|  |  |
|  | server.listen(3333, function (err) { |
|  | var address = server.address(); |
|  | console.error('server listening on http://%s:%d', address.address, address.port); |
|  | console.error('press CTRL+C to exit'); |
|  | }); |
|  |  |
|  | process.on('SIGINT', function () { |
|  | server.close(); |
|  | }); |
|  | }); |
|  | }); |
|  |  |
|  | }); |