Node.js Twitter Lab

Introduction

In this lab we'll make a simple node app that retrieves tweets containing a keyword. It runs in a terminal, but you can easily turn it into a web app.

You'll need a twitter account for this lab and a api key.

Step 1: setup and dependencies

```
touch package.json
touch twitter.js
```

We'll use the `ntwitter` module to connect with twitter, therefore we add a dependency in the `package.json` file:

```
{
  "name": "twitter",
  "dependencies": {
    "ntwitter": "0.5.x"
  }
}
```

After that we can install the dependencies by running npm:

```
npm install
```

Step 2: Client API keys

We need some client keys from twitter for this. You can get them at `https://dev.twitter.com`. We set them as environment variables to avoid having o hardcode them.

```
export TWITTER_CONSUMER_KEY=xxxxxxxxxxxxx
export TWITTER_CONSUMER_SECRET=xxxxxxxxxxxxx
export TWITTER_ACCESS_TOKEN_KEY=xxxxxxxxxxxxx
export TWITTER_TOKEN_SECRET=xxxxxxxxxxxxx
```

Step 3: using the API

Now we can use the API, let's try a simple search in the terminal:

```
var twitter = require('ntwitter');

var twit = new twitter({
  consumer_key: process.env.TWITTER_CONSUMER_KEY,
  consumer_secret: process.env.TWITTER_CONSUMER_SECRET,
  access_token_key: process.env.TWITTER_ACCESS_TOKEN_KEY,
  access_token_secret: process.env.TWITTER_TOKEN_SECRET
});

function queryTwitter(q) {
  twit.stream('statuses/filter', {'track':q}, function(stream) {
    stream.on('data', function (data) {
      if (data != undefined && data.user != undefined) {
        console.log('tweet', data.user.screen_name+' tweets: '+data.
            text);
      }
    });

    stream.on('error', function(error) {
      console.log('error', error);
    });
  });
};

queryTwitter("playing");
```

Now we can run it ans see what happens:

```
node twitter.js
```

Step 4: create a webserver

Ok, thats all very nice, but it would be much more fun if we could see all those incredible tweets on a website. So let's create a webserver that updates the page every time a new tweet comes in.

If we want to create a nice website for this, we'll probably end up serving some static content like css files, images and javascript files for the client. We are going to use `node-static` for that. To install that module we'll update the `package.json` file to include this new dependency.

```
{
  "name": "twitter",
  "dependencies": {
    "ntwitter": "0.5.x",
    "node-static":"x"
  }
}
```

After that we can run npm again to install it:

```
npm install
```

Now we create the webserver in our twitter app by adding these lines:

```
var http = require('http');
var static = require('node-static');

// http server
var app = http.createServer(handler);
app.listen(1337);
var file = new static.Server('./public');
function handler(req, res) {
  file.serve(req, res);
}
```

We've configured the webserver to serve static content from the public folder, so we'll have to create that folder:

```
mkdir public
touch pubic/index.html
touch public/client.js
```

in that folder we can put all our static content like a HTML file:

```html
<!DOCTYPE html>
<html>
  <head>
    <script src='/socket.io/socket.io.js'></script>
    <script src='/client.js'></script>
    <title>Twitter Lab</title>
  </head>
  <body>
  </body>
</html>
```

Step 5: Tweet events

and client side javascript: