apigee

V111414

Developer Bootcamp

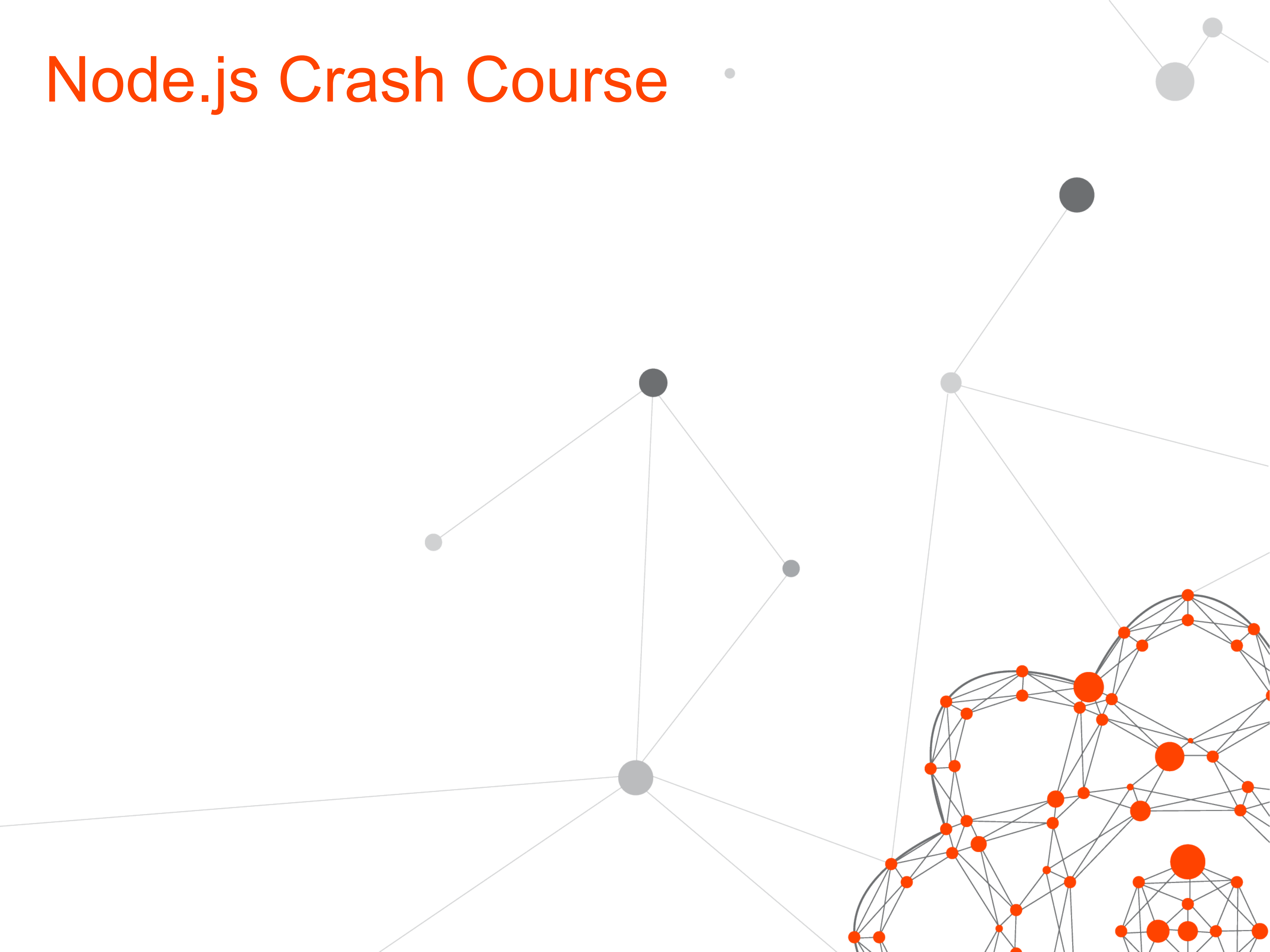
Node.js and Apigee-127

Course Topics

- **Node.js:**
  - Node.js Crash Course
  - Deploying and maintain a Node.js application in Apigee Edge
  - Volos: Caching, quota, OAuth and DB connectors
  - Troubleshooting, testing, debugging, and help

- **Apigee-127**
  - Building API Specs / Docs
  - Creating Controllers
  - API Management
  - Deploying the APIs

Node.js Crash Course



Introduction to server-side JavaScript, Node.js, and NPM

- Node.js – software platform for scalable server-side and networking applications.

- Why use Node.js?
  - Asynchronous, event-driven
  - HTTP is a first-class citizen of the Node world
  - Node has a strong user community with a wide array of open-source modules

Node.js and Apigee Edge

- Node allows you to fully develop offline and deploy to the Apigee cloud, taking full advantage of Apigee Edge

- Code can be easier than configuration
  - Asynchronous HTTP requests can be orchestrated and optimized much simpler using Node.js
  - If you are heavily manipulating and mashing up payloads, Node keeps this process clean and human-readable
  - Node can connect to many backend databases easily using Node modules

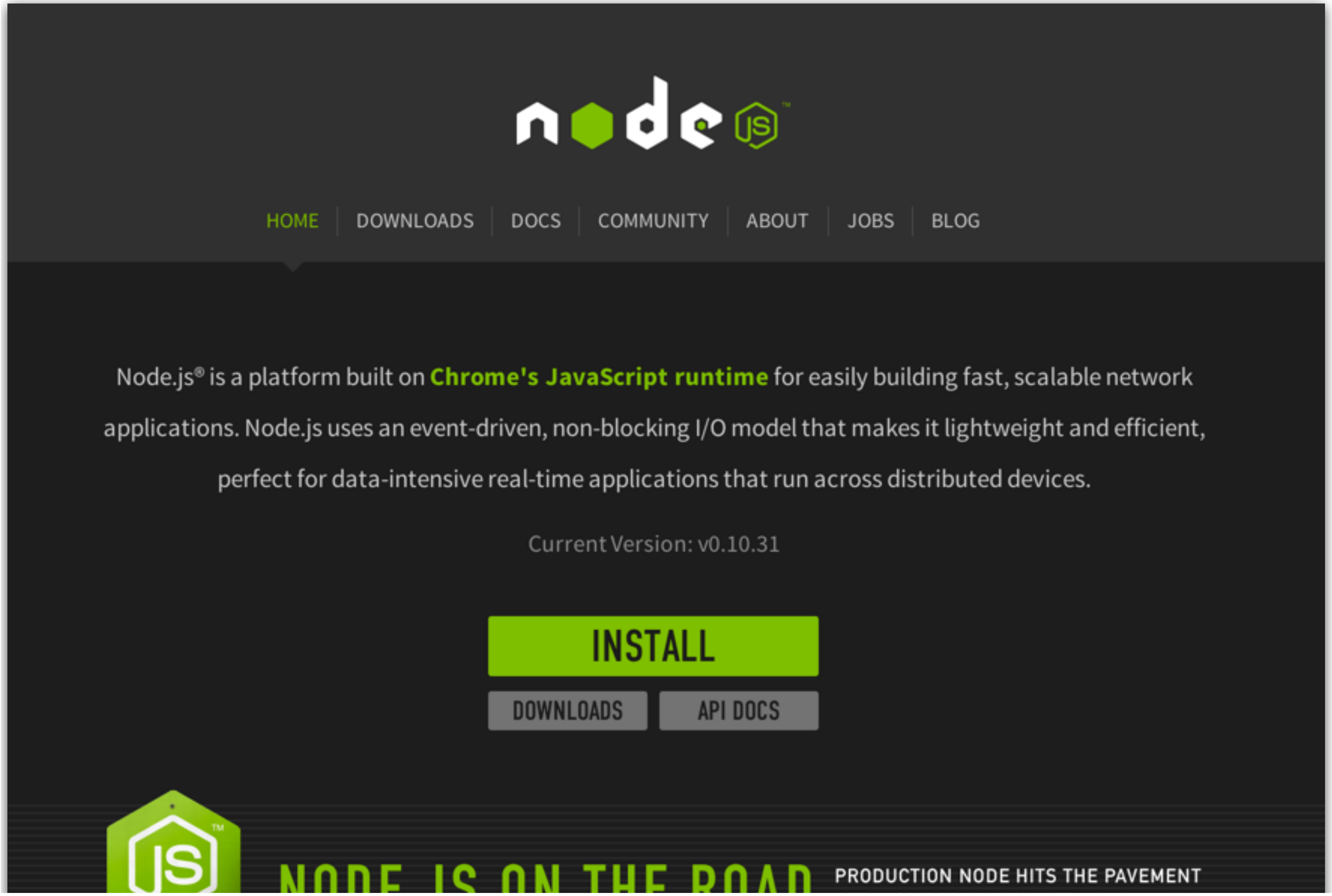
When to use Node.js in Apigee Edge

- When should I use Node.js in Apigee Edge?
  - When you need intelligent, asynchronous processing logic
  - When you've already built your API in Node and don't want to rebuild it from scratch

- What does Node.js look like in Apigee Edge?
- Handling and manipulating JSON data with JavaScript

http://jsfiddle.net/remus/z4Y68/

Node Installation



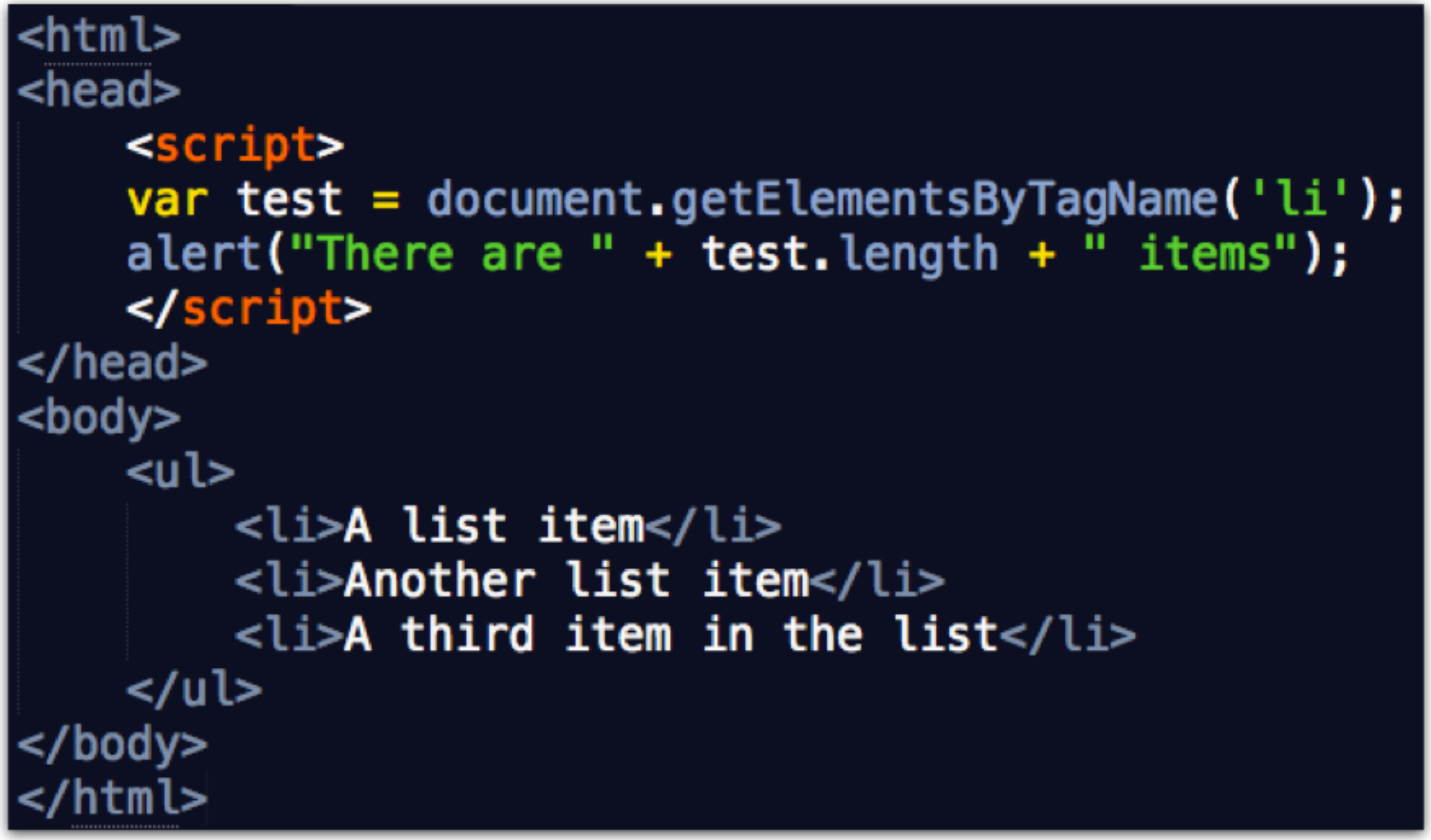
Node.js® is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Current Version: v0.10.31

Asynchronous Programming

- JavaScript objects and parsing response JSON
- “What” and “why” of asynchronous programming with Node.js
    - Understanding blocking code
    - Callbacks, promises and basis of asynchronous design

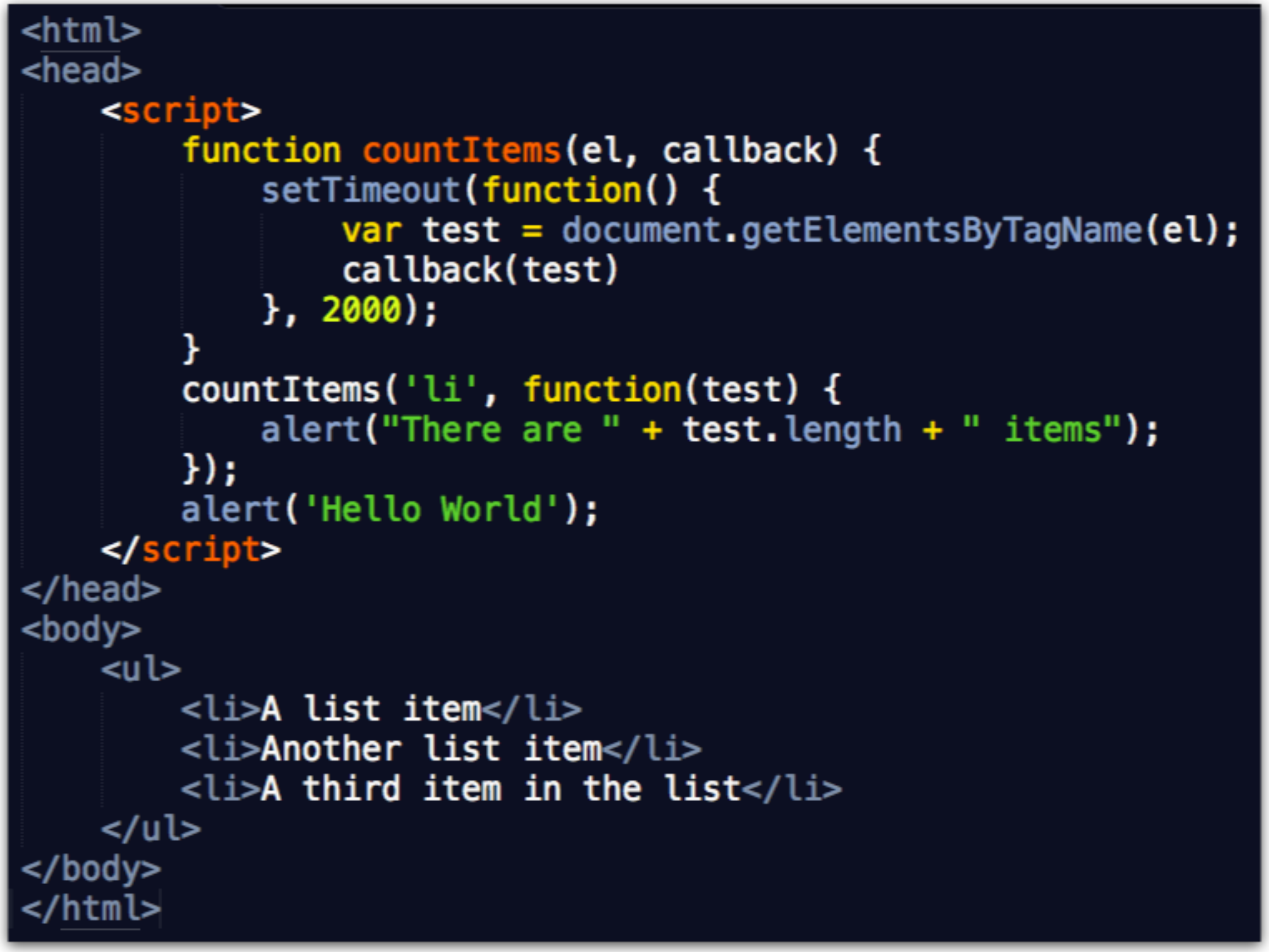
Blocking



```
<html>
<head>
    <script>
    var test = document.getElementsByTagName('li');
    alert("There are " + test.length + " items");
    </script>
</head>
<body>
    <ul>
        <li>A list item</li>
        <li>Another list item</li>
        <li>A third item in the list</li>
    </ul>
</body>
</html>
```

http://jsfiddle.net/grewis/p5nf8hdf/

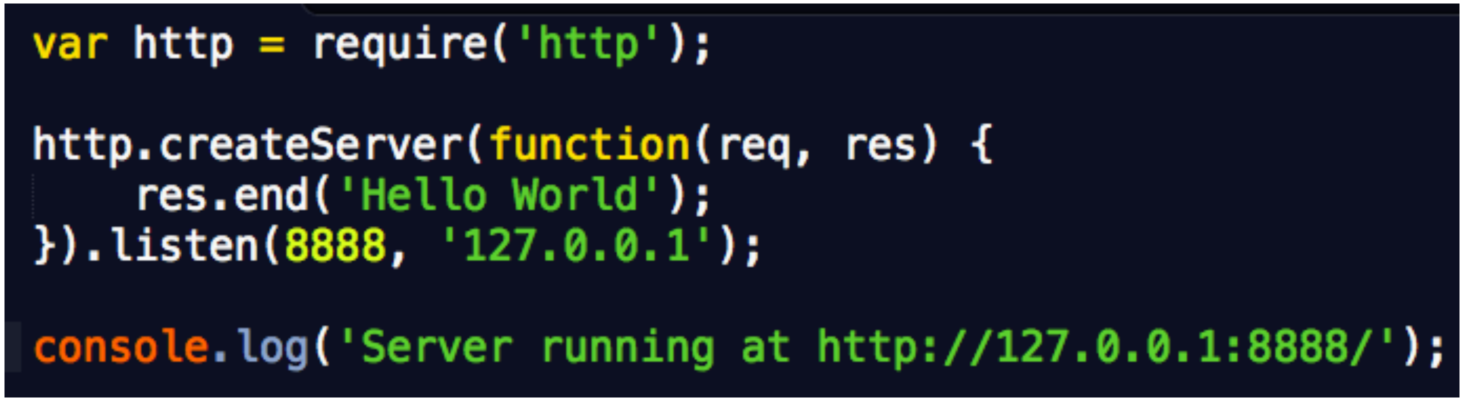
Non-blocking



```
<html>
<head>
    <script>
        function countItems(el, callback) {
            setTimeout(function() {
                var test = document.getElementsByTagName(el);
                callback(test)
            }, 2000);
        }
        countItems('li', function(test) {
            alert("There are " + test.length + " items");
        });
        alert('Hello World');
    </script>
</head>
<body>
    <ul>
        <li>A list item</li>
        <li>Another list item</li>
        <li>A third item in the list</li>
    </ul>
</body>
</html>
```

http://jsfiddle.net/grewis/3mm2qtac/

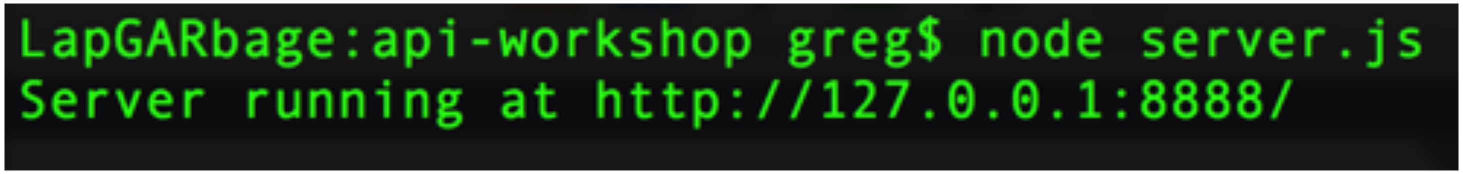
Simple Web Server



```
var http = require('http');

http.createServer(function(req, res) {
    res.end('Hello World');
}).listen(8888, '127.0.0.1');

console.log('Server running at http://127.0.0.1:8888/');
```



```
LapGARbage:api-workshop greg$ node server.js
Server running at http://127.0.0.1:8888/
```

j.mp/api-v1-node-simple

Day 4 – Lab 14 Build HTTP Server with NodeJS

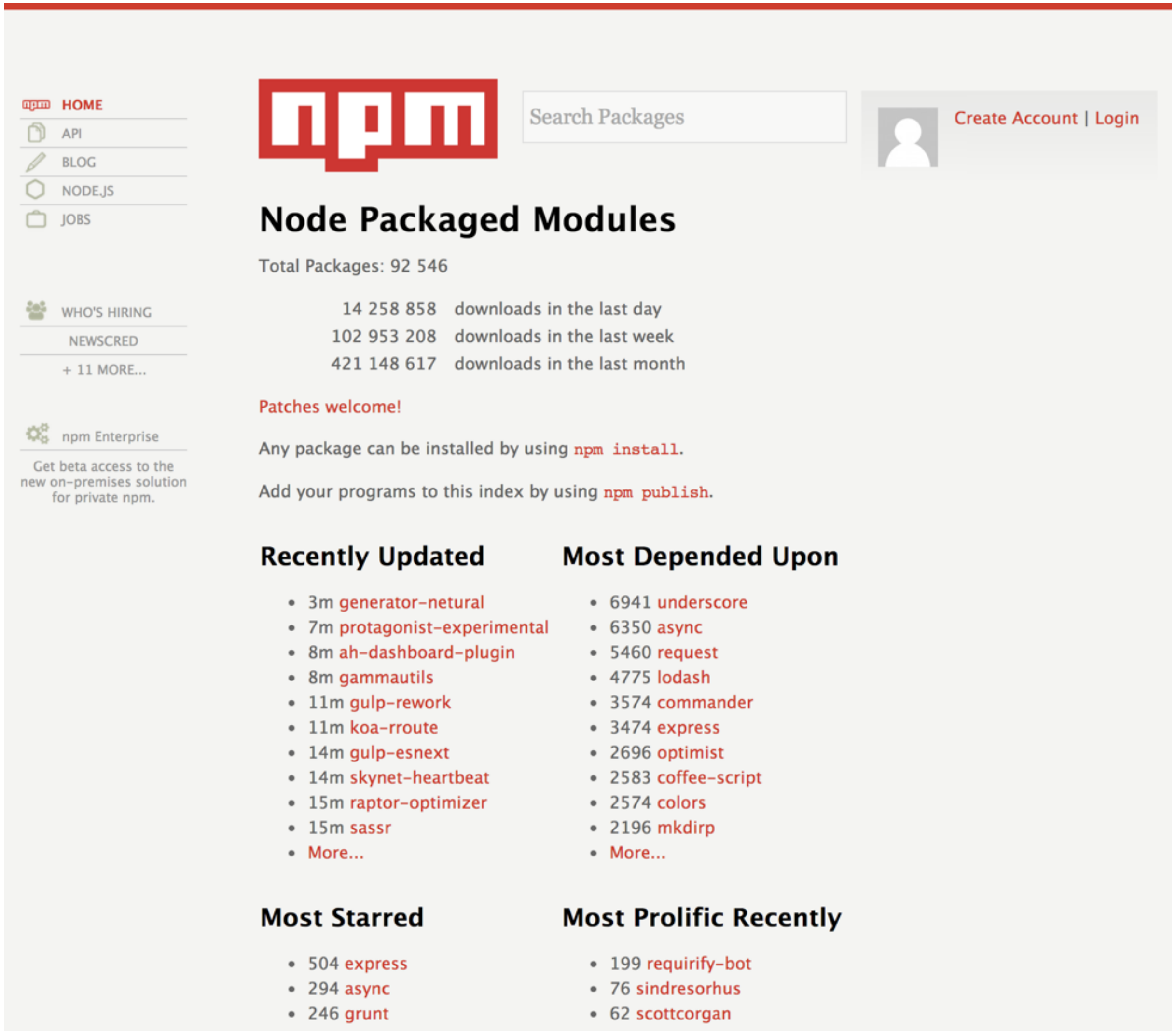
- Try building a nodejs HTTP Server
  - Create a text file http-server.js where nodejs is installed

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

- Test using postman to see if ‘Hello World’ is returned on browser

The World of NPM



HOME
API
BLOG
NODE.JS
JOBS

WHO'S HIRING
NEWSCRED
+ 11 MORE...

npm Enterprise
Get beta access to the new on-premises solution for private npm.

Search Packages

Create Account | Login

Node Packaged Modules

Total Packages: 92 546

```
 14 258 858   downloads in the last day
102 953 208   downloads in the last week
421 148 617   downloads in the last month
```

Patches welcome!

Any package can be installed by using `npm install`.

Add your programs to this index by using `npm publish`.

Recently Updated

- 3m generator-netural
- 7m protagonist-experimental
- 8m ah-dashboard-plugin
- 8m gammautils
- 11m gulp-rework
- 11m koa-rroute
- 14m gulp-esnext
- 14m skynet-heartbeat
- 15m raptor-optimizer
- 15m sassr
- More...

Most Depended Upon

- 6941 underscore
- 6350 async
- 5460 request
- 4775 lodash
- 3574 commander
- 3474 express
- 2696 optimist
- 2583 coffee-script
- 2574 colors
- 2196 mkdirp
- More...

Most Starred

- 504 express
- 294 async
- 246 grunt

Most Prolific Recently

- 199 requirify-bot
- 76 sindresorhus
- 62 scottcorgan

NPM Packages and other resources

Apigee developed NPM packages:

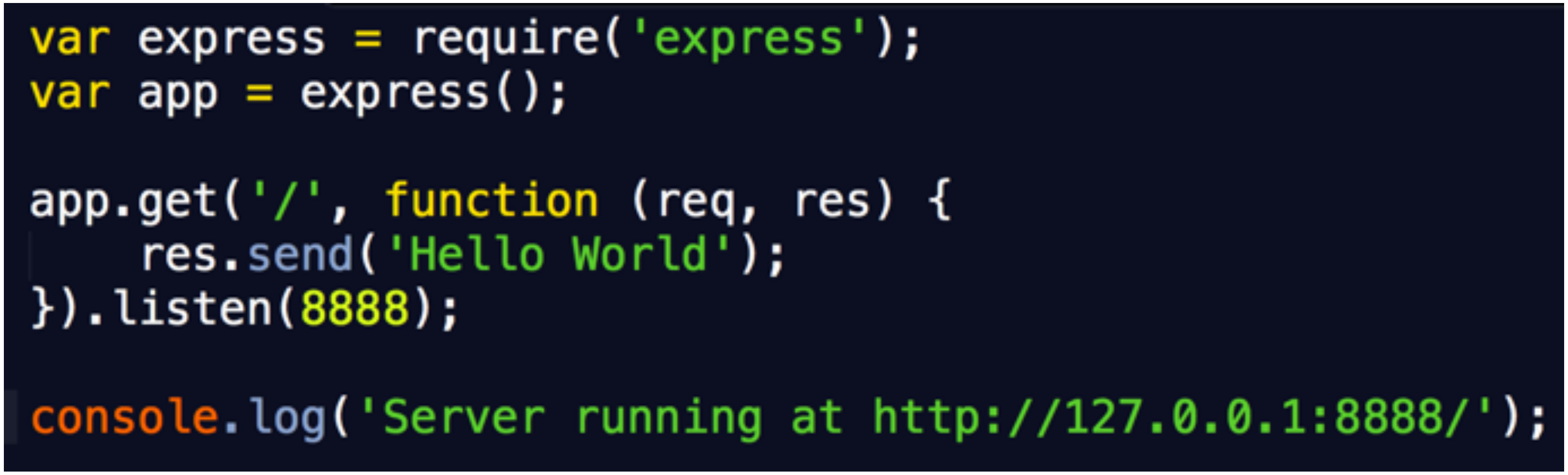
- `apigee` / `apigee-access`
- `volos` (cache, quota, oauth)
- `Usergrid`

Popular Editors/IDE used by Apigee developers:

- http://www.sublimetext.com/3
- https://www.jetbrains.com/webstorm/

If using Sublime, you can configure a build system with either regular Node.js or nodemon. Nodemon will automatically restart your node app upon code saves. http://nodemon.io/

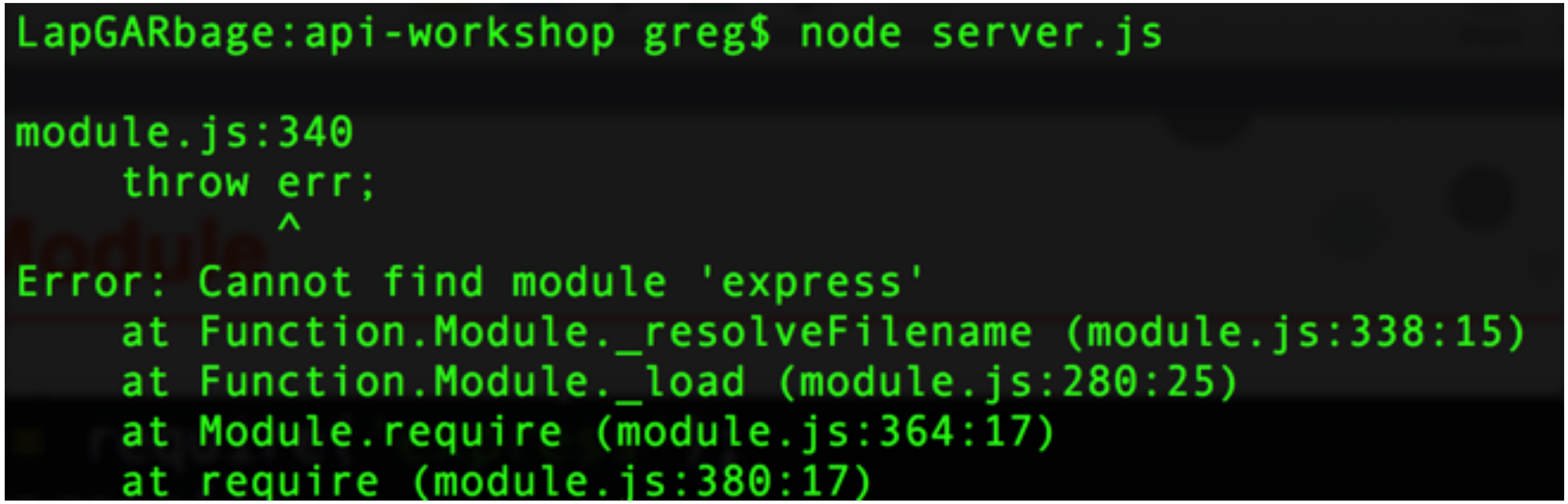
Modules Must be Installed



```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
    res.send('Hello World');
}).listen(8888);

console.log('Server running at http://127.0.0.1:8888/');
```

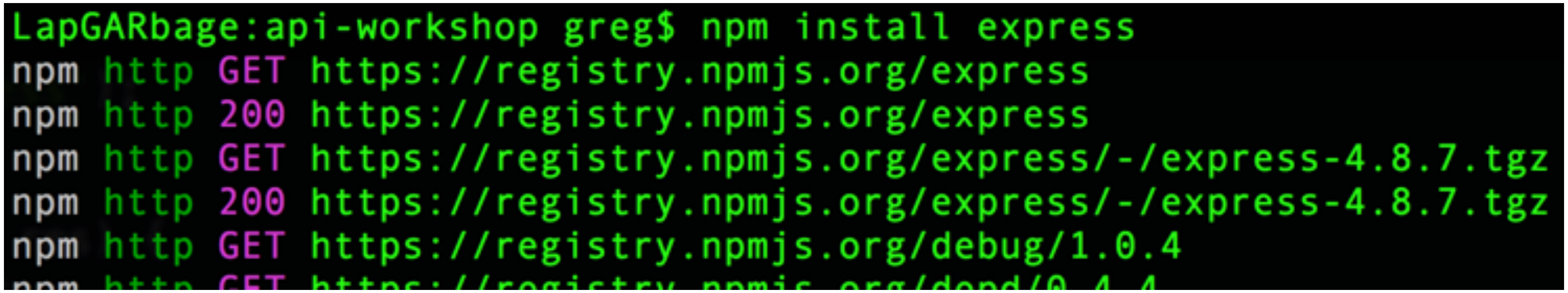


```
LapGARbage:api-workshop greg$ node server.js

module.js:340
    throw err;
          ^
Error: Cannot find module 'express'
    at Function.Module._resolveFilename (module.js:338:15)
    at Function.Module._load (module.js:280:25)
    at Module.require (module.js:364:17)
    at require (module.js:380:17)
```

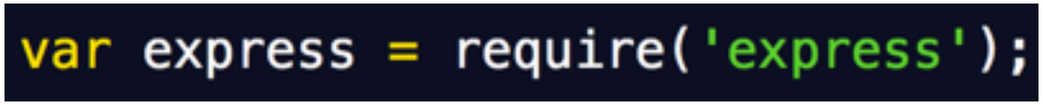
Installing Modules with NPM

- Individual modules must be installed using the NPM command line



```
LapGARbage:api-workshop greg$ npm install express
npm http GET https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/express/-/express-4.8.7.tgz
npm http 200 https://registry.npmjs.org/express/-/express-4.8.7.tgz
npm http GET https://registry.npmjs.org/debug/1.0.4
```

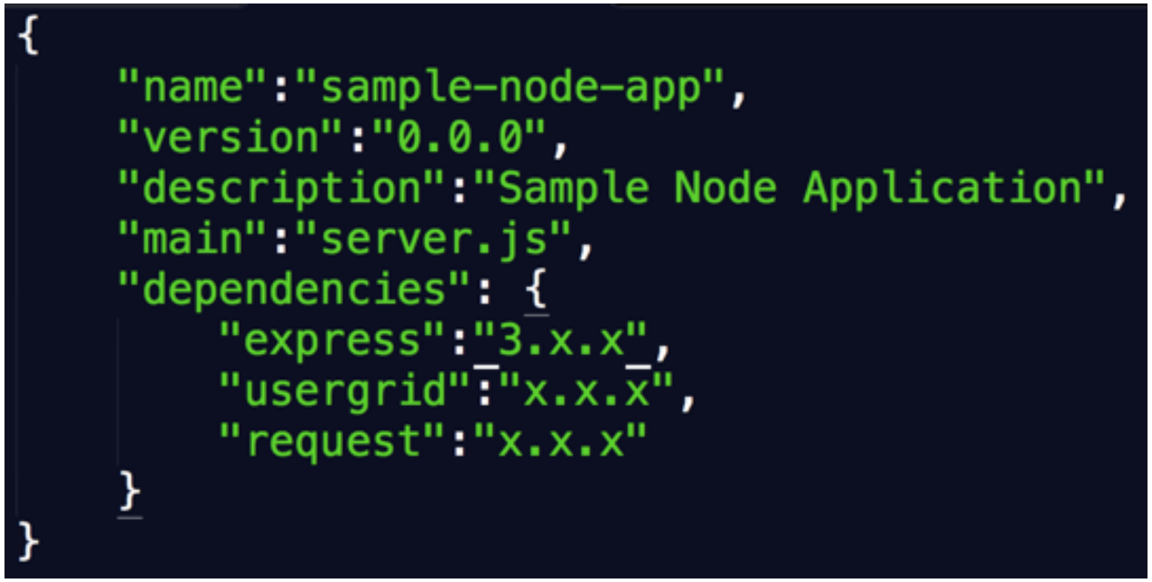
- Once installed (node_modules folder), they are added to the app using *require('module name')*



```
var express = require('express');
```

Installing Multiple Modules

- When a Node.js app launches, it looks for a file, *package.json*, in the same folder and the main JS file to determine if all of its ‘pieces’ are present



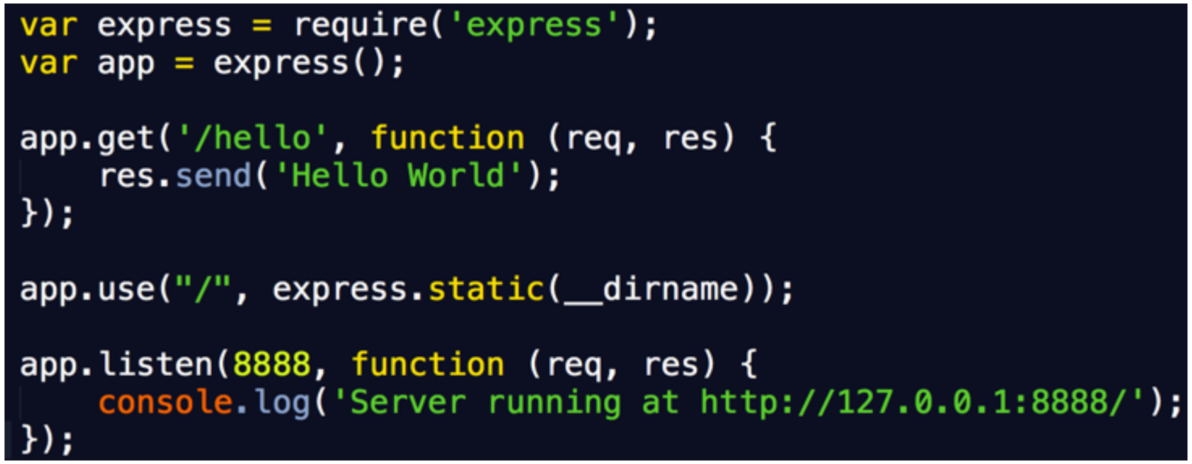
```
{
    "name":"sample-node-app",
    "version":"0.0.0",
    "description":"Sample Node Application",
    "main":"server.js",
    "dependencies": {
        "express":"3.x.x",
        "usergrid":"x.x.x",
        "request":"x.x.x"
    }
}
```

- Using **npm install** will install all of the listed dependencies at one time.

Express Routes

Express uses routes and provides simple routing out of the box: app.METHOD(path, [callback...], callback).

- **app.get('/hello', callback)** … will catch a GET request to path /hello
- The callback **function (req,res){}** allows you to work with the request (req) and response (res) objects.
- **app.use("/", [function...], function)** mounts middleware functions to be executed at the path. This case will have the express.static() function execute for every request in the app.



```
var express = require('express');
var app = express();

app.get('/hello', function (req, res) {
    res.send('Hello World');
});

app.use("/", express.static(__dirname));

app.listen(8888, function (req, res) {
    console.log('Server running at http://127.0.0.1:8888/');
});
```

j.mp/api-v1-node-simple-02

Use Multiple Routes to Expose API Resources

- Handling multiple (conditional) routes and multiple HTTP request methods
  - Route syntax `/` , `/ path`, `/ path/:parameter`
  - Use multiple route listeners for specific HTTP requests for resources
  - '**express-params**' module allows you to use app.param() to define a request parameter and perform validation
- Accessing and sending HTTP headers and request parameters
  - Routes and headers
  - `req.params.{routeParameter}` and `req.headers.{header}`

```
var express = require('express');
var params = require('express-params');
var app = express;
params.extend(app);

app.get('/', function(req, res){
    res.send('Hello, world. We meet again.');
});

app.param('id', /^\d+$/);
app.get('/user/:id', function(req, res){
    res.send('user ' + req.params.id);
});

app.param('range', /^(\w+)\.\.(\w+)?$/);
app.get('/range/:range', function(req, res){
    var range = req.params.range;
    res.send('from ' + range[1] + ' to ' + range[2]);
});
```

Executing 3rd Party HTTP/S Requests from Node.js

- **request** module allows you to execute HTTP requests from your node.js app
  - common module used for 3rd party HTTP requests
  - Can perform GET, POST, PUT and DELETE requests from your Node.js server
  - `pipe()` function allows you to pipe the response of the the 3rd party request to the response being sent back to the client from your node.js app

```
app.all('/:uuid', function(req, res){
    if (req.method === 'PUT') {
        req.pipe(request.put(basepath + req.params.uuid));
    } else if (req.method === 'GET' || req.method === 'HEAD') {
        request.get(basepath + req.params.uuid).pipe(res)
    }
});
```

Day 4 – Lab 15 Build HTTP Server with NodeJS

- Try building a nodejs Express Server using express package
  - Create a text file express-server.js where nodejs is installed

```
var express = require('express');
var app = express();
app.get('/hello', function(req, res) {
    res.send('Hello World!');
}).listen(1337);

app.get('/today', function(req, res) {
    res.send({today: new Date()})
});
app.use(express.static(__dirname+'\www'));
console.log("Server has started.");
```

- Test using postman both /hello and /today targets

Data Manipulation and Mashups with Node.js

- **Improve the API response by:**
  - Parsing JSON into a JavaScript object
  - Manipulating (mashing-up) arrays and dictionaries/key:value maps
  - Stringifying and writing the output (`res.json` or `JSON.stringify()`)
  - Setting the proper HTTP status codes
    - 200 OK, 401 Unauthorized, 404 Not Found, etc.

- **Mashing up multiple API calls with *async* module**
  - Require `async` in your Express app
  - Create a parallel request with named callbacks
  - Mash up products, ratings, and reviews

Node.js Application Architecture – export modules

- Models, (views?), and controllers
- Route Performance impact to module separation (only at run-time!)
- Node's `require` module for separation of reusable functions
- Creating and including modules with `module.exports`

Create a new file, misc.js and add the following:

```
var x = 5;
var addX = function(value) {
    return value + x;
};
module.exports.x = x;
module.exports.addX = addX;
```

Include misc.js in the main app.js to leverage the exported variables/functions:

```
var misc = require('./misc.js');
console.log("Adding %d to 10 gives us %d", misc.x, misc.addX(10));
```

Day 4 – Lab 16 NodeJS Integration with BaaS

Demo NodeJS direct integration with BaaS

Using product-app

Day 4 – Lab 16 NodeJS Integration with BaaS

- Create a new target /users to connect to BaaS
  - Create a file users-app.js

```
var express = require('express');
var request = require('request');
var async = require('async');

var basepath = "http://api.usergrid.com/apigee.certification/sandbox/";
var app = express();
app.get('/users', function(req, res) {
        request(basepath + "users", function(error, response, body) {
                var bodyParsed = JSON.parse(body);
                delete bodyParsed.uri;
                res.json(bodyParsed);
        });
});
app.listen(8000);
//http://api.usergrid.com/apigee.certification/sandbox/products
//http://api.usergrid.com/apigee.certification/sandbox/users
```

Deploying and maintaining a Node.js application in Apigee Edge

Supported Modules and Pre-installed Packages in Edge

Apigee Edge does not support all modules that can be used in node.js apps. A detailed listing can be found on this documentation page:

http://apigee.com/docs/api-services/content/understanding-edge-support-nodejs-modules

Configuring an API Proxy to Run a Node Application

When deploying a node application to Apigee edge, the node app is configured inside of the existing API proxy structure. There are slight modifications to the

```
-node-demo-proxy/
    -apiproxy/
      -policies/
      -proxies/
      -targets/
      -resources/
        -node/
          -app.js
          -node_modules.zip
```

- The node_modules directory must be zipped.

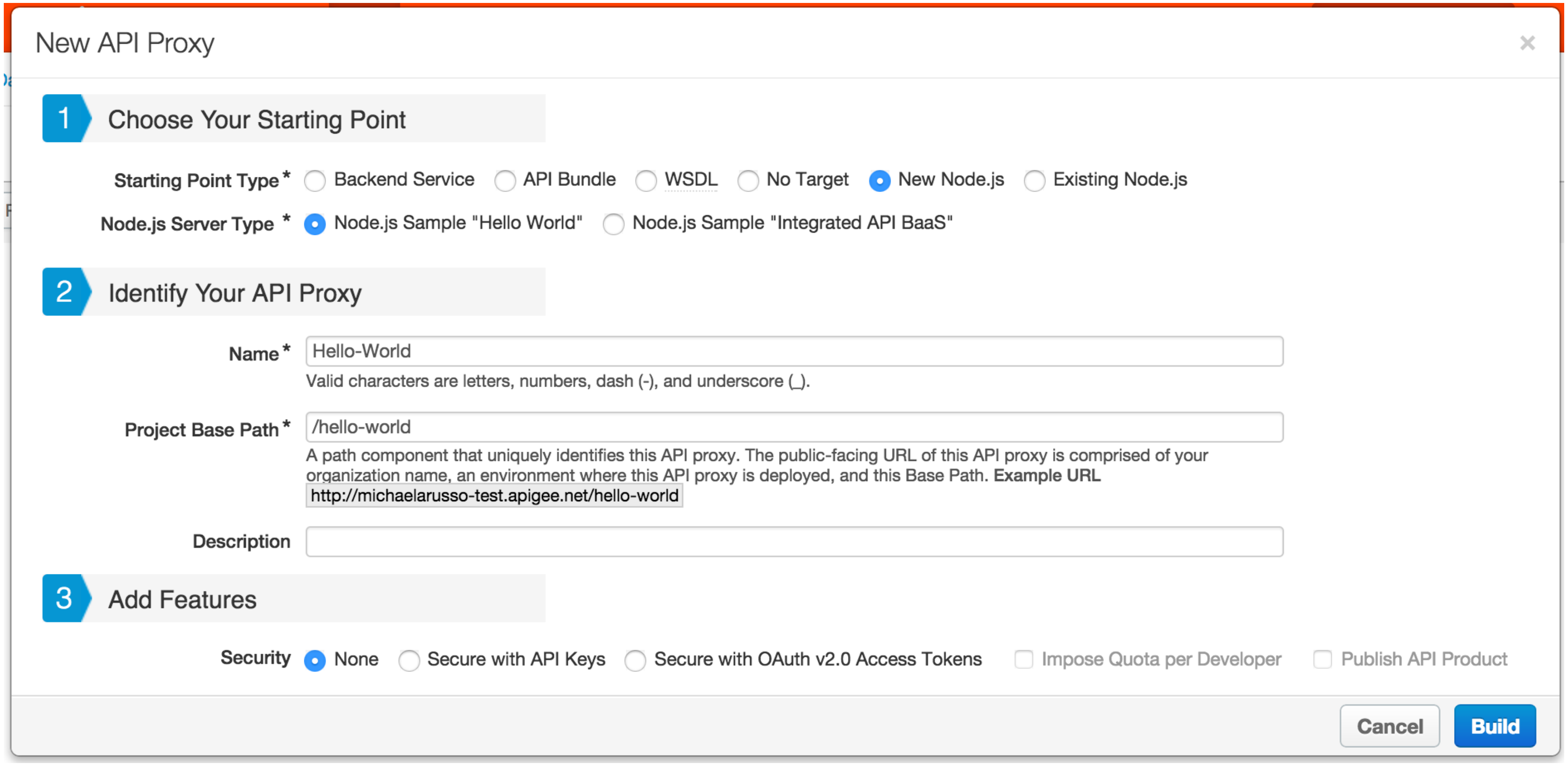
```
<ProxyEndpoint name="default">
   <HTTPProxyConnection>
        <BasePath>/employees</BasePath>
        <VirtualHost>default</VirtualHost>
    </HTTPProxyConnection>
    <RouteRule name="default">
        <TargetEndpoint>default</TargetEndpoint>
    </RouteRule>
    <PreFlow name="PreFlow"
        <Request/>
        <Response/>
   </PreFlow>
</ProxyEndpoint>
```

```
<TargetEndpoint name="default">
    <ScriptTarget>
        <ResourceURL>node://server.js
        </ResourceURL>
    </ScriptTarget>
    <EnvironmentVariables>
         <EnvironmentVariable name=”myVariable”>VALUE
         </EnvironmentVariable>
     </EnvironmentVariables>
     <Arguments>
         <Argument>ARG</Argument>
     </Arguments>
</TargetEndpoint>
```

- Specify main node app as a <ScriptTarget><ResourceURL>
- <EnvironmentVariables> can be specified in TargetEndpoint and then referenced in the node.js code as ***`var myVariable = process.env.myVariable;`***

UI Wizard for creating Node.js API Proxy Skeletons

Create a new Node.js proxy in Apigee Edge UI



New API Proxy

1 Choose Your Starting Point

Starting Point Type * ( ) Backend Service ( ) API Bundle ( ) WSDL ( ) No Target [✓ New Node.js] ( ) Existing Node.js

Node.js Server Type * [✓ Node.js Sample "Hello World"] ( ) Node.js Sample "Integrated API BaaS"

2 Identify Your API Proxy

Name * Hello-World

Valid characters are letters, numbers, dash (-), and underscore (_).

Project Base Path * /hello-world

A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Base Path. **Example URL** http://michaelarusso-test.apigee.net/hello-world

Description

3 Add Features

Security [✓ None] ( ) Secure with API Keys ( ) Secure with OAuth v2.0 Access Tokens ☐ Impose Quota per Developer ☐ Publish API Product

Deploying and Maintaining a Node app in Apigee Edge

- Limitations of creating a Node.js proxy through the Edge UI
  - If a bundle is uploaded, *node_modules* directory is not auto compressed
  - Handy for creating simple or on-the-fly Node scripts

- Apigee `api-platform-tools`
  - You should have it installed already: https://github.com/apigee/api-platform-tools
  - For Windows, install python for Windows (2.7): https://www.python.org/downloads/windows/

Use apigeetool to deploy node apps from command line

Deploying a Node.js proxy from the command-line tool

```
apigeetool deploynodeapp -o michaelarusso -e test -n node-demo -u mrusso@apigee.com
-d . -m app.js -b /node-demo-app

-o {apigee organization}
-e {apigee environment}
-n {apigee API proxy name}
-u {apigee edge username}
-d {directory where the node app.js file exists}
-m {node app file name used for TargetEndoint ResourceURL (app.js in this example)}
-b {basepath to use for the imported proxy}
```

– Example directory structure for above command:

```
workspace/
    -node-demo-app/
        -app.js
        -node_modules/
            -express-params/
```

– Testing our success
  - You should be able to access your Node.js proxy from here: http(s)://{org}-{env}.apigee.net/{basepath}/

Day 4 – Lab 17 NodeJS app in Apigee

DEMO Deploying NodeJS app in Apigee

using

Apigee-tool

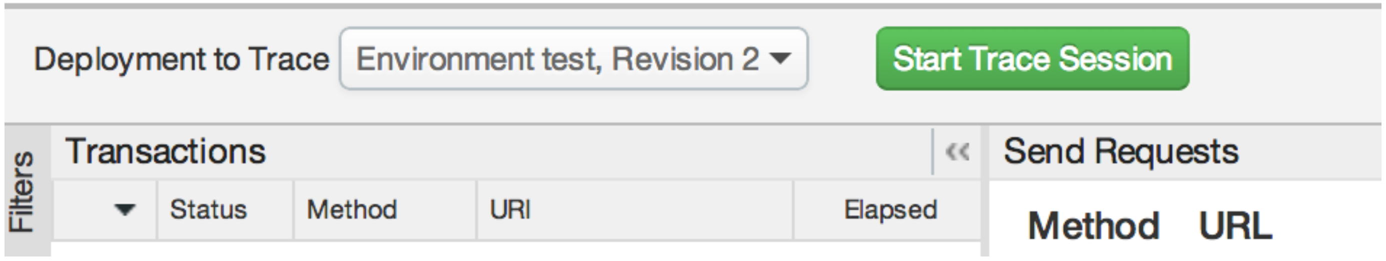
Day 4 – NodeJS APIProxy and BaaS

DEMO Nodejs in Proxy with BaaS Integration

Troubleshooting, testing, debugging, and help

Troubleshooting

- Troubleshooting Node.js using the Apigee Trace tool



- Inspecting requests, responses, and HTTP status codes
- `console.log()`

Troubleshooting, debugging, and where to get help

- The `node-inspector` tool
  - Browser-based Node.js debugger
  - Navigate in your source files
  - Set breakpoints (and specify trigger conditions)
  - Step over, step in, step out, resume (continue)
  - Inspect scopes, variables, object properties

- Webstorm IDE
  - A feature-rich IDE for JavaScript/Node.js

Getting help

- Getting help
- StackOverflow has an incredible wealth of Node.js knowledge
- How to ask: http://stackoverflow.com/help/how-to-ask

- Apigee Community
  - http://community.apigee.com/

- Apigee doc
  - http://apigee.com/docs/api-services/content/getting-started-nodejs-apigee-edge

Thank you

apigee