ADALTAS

NODEJS

ASYNCHRONOUS SERVER TECHNOLOGIES

César Berezowski

Big Data Consultant @ Adaltas

cesar@adaltas.com

ABOUT THIS COURSE

- You will use / do :
  - JavaScript / CoffeeScript and Markdown
  - Node.JS / NPM
  - Git / Github
  - Unit tests & Travis CI
  - Read the doc ! Ask Google !
- You will apprehend tools of the Open Source

ABOUT THIS COURSE

- Evaluation :
  - Continuous using GitHub
  - Simple project at end of course
- Planning: https://github.com/adaltas/ece-nodejs/blob/master/CALENDAR.md

ADALTAS

FINAL PROJECT

- Based on code from class
- Using presented technologies
- Simple dashboard app :
  - User login
  - A user can insert simple metrics
  - A user can retrieve his metrics displayed nicely in a graph
  - A user can only access his own metrics

ADALTAS

QUESTIONS ?

ADALTAS

TOOLS & CONCEPTS

TERMINAL

- Most useful developper tool
- Any number of customizations
- On windows : Powershell, Cygwin, Git Bash, GOW

ADALTAS

VI(M)

- Bash text editor
- Use `:` to enter command mode
  - `w` to write file
  - `q` to quit
  - `q!` to quit without saving
  - 'x' to write & quit
- Use / to search for text
- Use `i` to enter edit mode and `esc` to exit it

ADALTAS

CLIENT VS SERVER

- Two parts of a distributed computing model :
  - Client requests the info and displays it
  - Server processes the request and services the result
- We will do server work + a bit of client side

ADALTAS

THE IP PROTOCOL

- Send data from one computer to another over a network (ex: client/server)
- Use of IPV4 addresses (ex: 172.16.254.1)
- Data packaged in IP packets with 2 sections
  - Header: IP version, addresses, TTL, ...
  - Data: the packet's content

THE HTTP PROTOCOL

- Application protocol for transmitting hypermedia documents (HTML)
- Two types of messages : *requests* & *responses*
- HTTP message split between *headers* & *body*
- HTTP response always contains
  - the *protocol* (HTTP/1.1)
  - a *status code* (200, 404, ...)
  - a *status text* (page not found)

SSL/TLS & HTTPS

- Establish an encrypted link over a network
- Exchange of public & private keys to secure the exchange
  - Server sends SSL certificate + public key
  - Client checks the certificate & answers with an encrypted session key
  - Client & server exchange messages encrypted with the keys to authenticate
- HTTPS: HTTP secured with SSL/TLS

SSH - SECURE SHELL

- Cryptographic network protocol to operate network services securely over an unsecured network
- Exchange of public & private keys to secure the exchange
  - Client has the private key
  - Server needs to have the associated public key
  - Client & server exchange messages encrypted with the keys to authenticate

THE SFTP PROTOCOL

- Send files over SSH
- ex: deploy website to a server
- SFTP apps : FileZilla, Cyberduck, WinSCP, ...

ADALTAS

GIT

- Distributed version control
- Users keep entire code & history locally, can make any change without internet
- Users create snapshots of current code (`commit`) associated to a hash code
- Users `push` commited code to the remote git server
- Multiple users can work on the same git project
- When two users modify the same code they have to `merge` the two codes

GIT COMMANDS

- `git init` : initialize a git repository
- `git status` : show the current status of the local git repo
- `git clone`: download a repository locally
- `git add [files]` : add the files to the git index
- `git commit -m "[message]"` : create a commit
- `git push -u origin master` : push commits to the distant repo
- `git pull` : pull changes from the distant repo

Git cheatsheet

GIT PLATFORMS / TOOLS

- Hosting
  - Github.com : free public repository hosting
  - Bitbucket.com : free public / private repository hosting
  - Gitlab : install your own git server anywhere
- Use
  - GitX (Mac) / GitG (Linux)
  - GitHub (Mac/Win/Linux)
  - SourceTree (Mac/Win/Linux)
  - GitKraken (Mac/Win/Linux)
  - Your terminal !

EDITORS

- As a developer, your editor shall be your best friend
- Atom editor
  - Developed by Github
  - Developed with and running on Node.JS
  - Highly customizable & lots of packages
  - http://atom.io
- Others : Sublime Text, TextWrangler, ...

PROGRAMATION PARADIGMS

- "A way of programming"
- Common paradigms :
  - Imperative - Control flow is an explicit sequence of commands
  - Functional - Computation proceeds by function calls, no global state
  - Object-oriented - Everything is an object
  - Event-driven - Control flow determined by async actions

UNIT TESTING

- Method to test individual parts of a program to show that each is correct
- Goals :
    - Find problems early in the development phase
    - Facilitate change
    - Simplifie integration
    - Documentation
- Limited, should always be coupled with other tests

ADALTAS

NODE.JS

ADALTAS

JAVASCRIPT

- Developed in 1995 at NetScape
- Shipped with IE3 in 1996 as JScript
- Standardized with EcmaScript v1 in 1997 (now v6)
- No relation to Java
- Rediscovered with Ajax around 2005 (Gmail, Maps…)
- Multi-paradigm : scripting, object-oriented, functional, imperative, event-driven
- One of the most popular languages today

NODE.JS

- Javascript runtime
- Created in 2009 by Ryan Dahl
- Uses Google’s V8 JavaScript Engine
- Package management using NPM
- Asynchronous IO
- Unix philosophy of small components
- Community
  - on Github
  - backed up by the Node.js foundation
- Current version is 6.7.0 & LTS is 4.6.0

ASYNCHRONICITY

- Threads are evil !
- Not blocking, not waisting CPU
- No memory synchronization
- Maintain multiple HTTP connections

ADALTAS

EVENT LOOP

- Event-driven paradigm
  - Central mechanisms
  - Listens for events
  - Calls a callback function
    - ex: `element.onClick()`
- The event-loop delegates blocking calls to the system
  - ex: writing, holding connections, ...

LET'S INSTALL SOME STUFF

- Build: ‘./configure; make; make install’
- Installer : https://nodejs.org/en/download/
- Package manager: ‘apt-get’, ‘yum’, ‘brew’, …
- Node management systems: ‘n’, ‘nvm’, ’nave'

HELLO WORLD !

```
// Import a module
var http = require(‘http’)

// Declare an http server
http.createServer(function (req, res) {

  // Write a response header
  res.writeHead(200, {‘Content-Type’: ‘text/plain’});

  // Write a response content
  res.end(‘Hello World\n’);

// Start the server
}).listen(1337, ‘127.0.0.1’)

// curl localhost:1337 or go to http://localhost:1337
```

MODULES ?

- Use

```
module.exports = …
```

- Export anything: a function, an array, an object …
- Import in another file:

```
var my_mod = require('/path/to/my_file.js')`
```

- That's how Node libraries are made !

ADALTAS

QUESTIONS ?

ADALTAS

NODE PACKAGE MANAGER

WHAT IS NPM ?

- Package manager for Node.JS
- Developed by Isaac Z. Schlueter
- Upload, share & download packages
- Two modes: global & local
- Modules: system I/O, networking, cryptography, framework, …
- npmjs.com

MODULE DECLARATION: PACKAGE.JSON

- Create a folder and add a ‘package.json’ file
- Add ‘name’, ‘description’ and ‘version’ fields
- Add the ‘dependencies’ and ‘devDependencies’

package.json doc

ADALTAS

MODULE INFORMATIONS: README.MD

- Written in Markdown
- Should contain :
  - Short introduction
  - Installation instructions and how to run the tests
  - Usage instruction with simple (and advanced) examples
  - Note and migration instructions
  - List of contributors

Markdown doc

ADALTAS

LET'S CREATE A MODULE !

- Convert our HTTP server into a module
- Create a ‘./src/users.js’ file
- Expose two functions: ‘save’ and ‘get’

```
module.exports = {
  save: function (user, callback){…},
  get: function (id, callback){…}
}
```

- Use this module in an ‘app.js’ file

ADALTAS

QUESTIONS ?