

2 DEVELOPMENT FRAMEWORK NODE.JS, NPM, GIT

Salvatore Rinzivillo

OBJECTIVES

- Setup a developing environment
 - Install Node.js and NPM
 - Install and configure Vue and @vue/cli
 - Configure and initialize a project
- Install and configure git
 - Create a repository and import project files
- IDEs
 - Fork, Git Desktop
 - WebStorm, VS Code

NODE.JS AND NPM

WHAT IS NODE.JS

- “An asynchronous event driven Javascript runtime”
 - Non-blocking, event-driven I/O operations
 - Lightweight and efficient for data-intensive applications
 - Distributed computation and load balancing
- Available for download at <https://nodejs.org/>

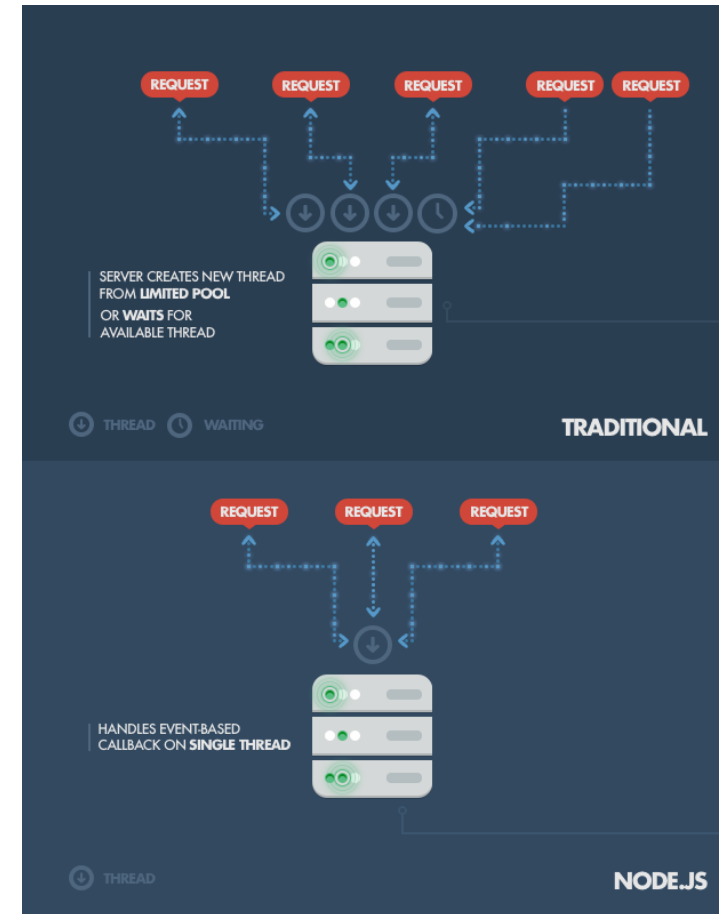


Image source: <https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js>

NPM – NODE PACKAGE MANAGER

- Node.js has a large library of public available, reusable components
- Components are available through a repository
- Manage libraries for global use and local projects
- Handle all dependencies

NPM - COMMANDS

- `npm init`
 - Initialize a project, creating a file `package.json`
- `npm install <module>`
 - Download and install module within the directory `node_modules`
 - ~~With the flag `--save`, add the module to the `package.json` list of dependencies~~
 - With the flag `--global` (or `-g`) the module is installed globally on the system

MOST USED PACKAGES

- Express: a web application development framework for node.js
- Lodash: general utilities for handling data structures in javascript
- http-server; webpack; babel
- Specifically for the course:
 - D3
 - Vega; Vega-lite
 - Bootstrap
 - Vue.js
 - ...

EXERCISE – CREATE A PROJECT WITH NODE.JS

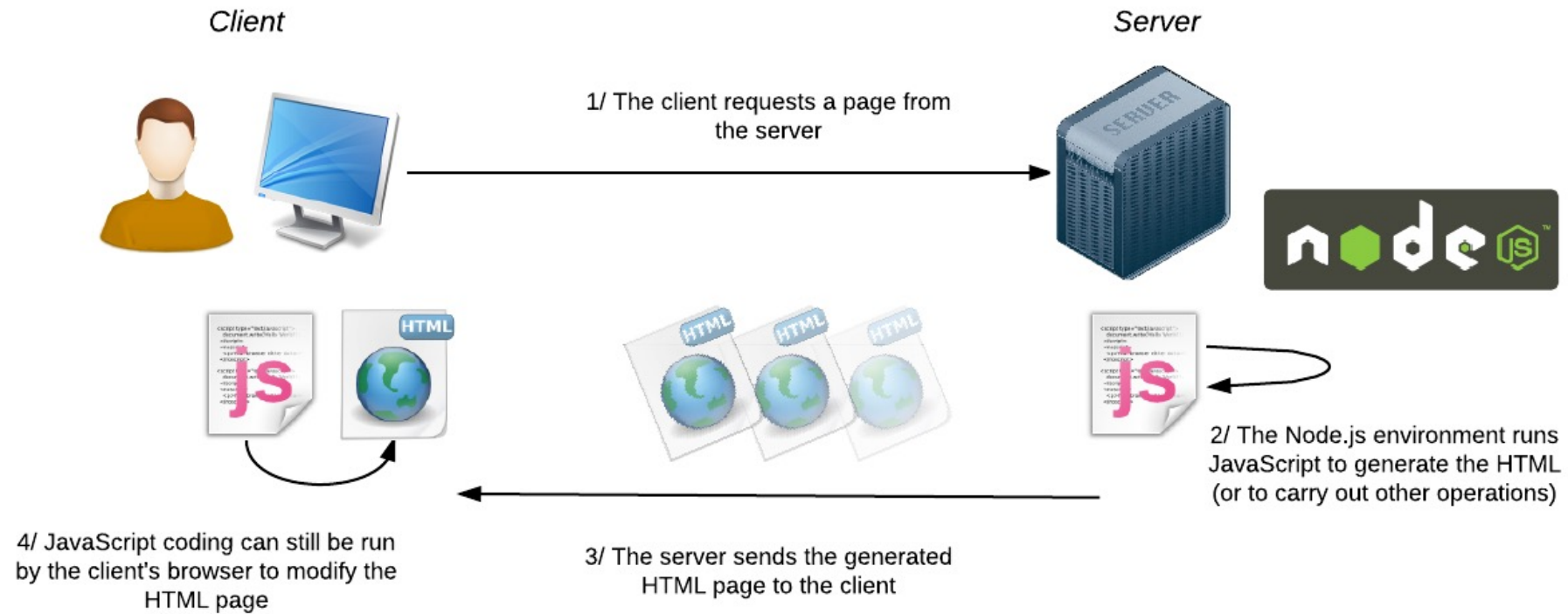
- Demo

CONFIGURING VUE.JS AND @VUE/CLI

- `npm install -g @vue/cli`
 - Create a command to manage Vue.js projects
- `npm init vue@latest`
- `cd my-project`
- `npm install`
- `npm run dev`
- These commands create a skeleton project configured with Vue.js

WEB SERVER

WEB SERVER FOR NODE.JS



WEB SERVER IN NODE.JS

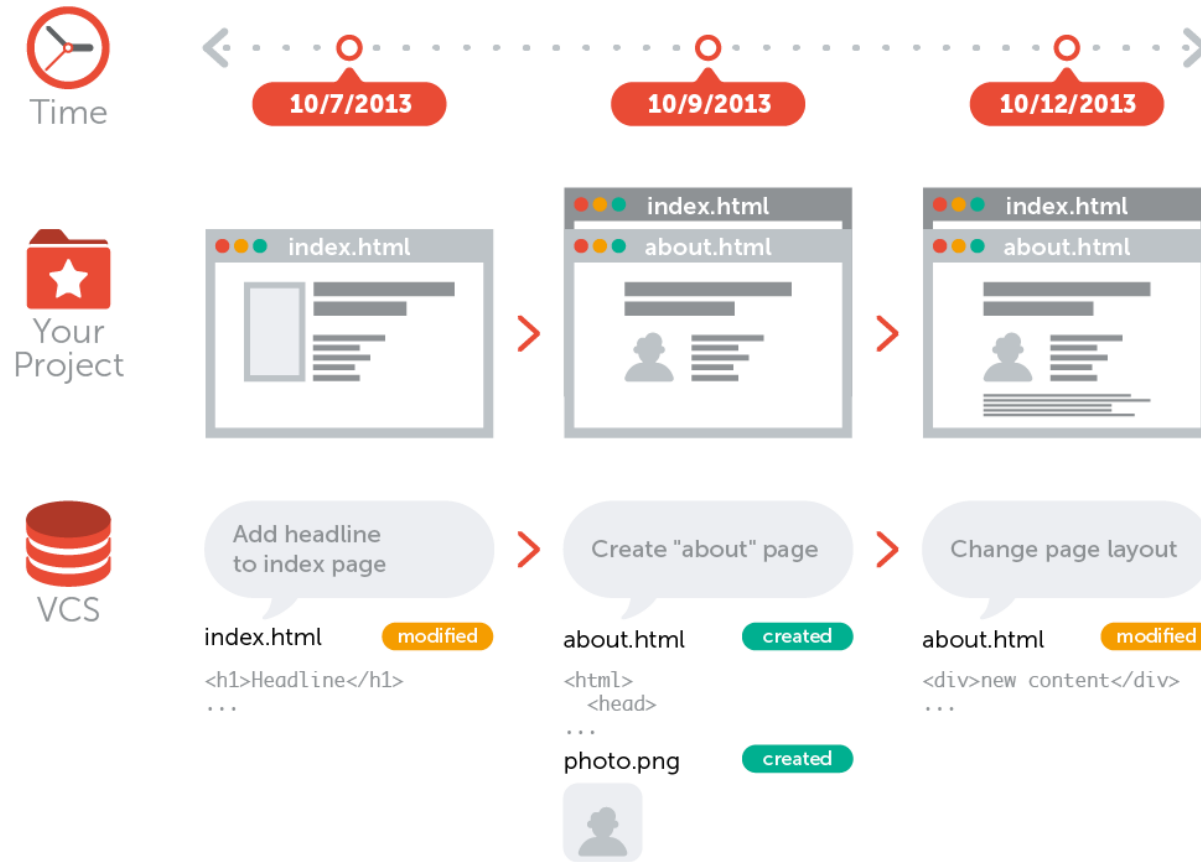
- There are several modules available for running a web server
- A very simple http server:
 - `npm install -g http-server`
- A more sophisticated application server:
 - `npm install -g express`

EXERCISE – USE HTTP-SERVER TO ACCESS OUR PROJECT

- Demo

VERSION CONTROL WITH GIT

WHAT IS VERSION CONTROL?



WHY USE A VERSION CONTROL SYSTEM?

- Collaboration
 - Any member of a team can work on any file at any time
 - Merge of contribution is handle by the VCS
- Storing versions
 - Tracking of changes through periodic saves of snapshots
 - Only one version of a project at any time
 - Other versions are packed within the VCS
- Restore previous versions
- Follow the development of the project
- Backup, when using external repositories

WHICH VCS? INTRODUCING GIT

- Download a client from public repositories
 - For example: GitHub, BitBucket
- Use clients specific for your OS
 - For example: `brew install git` (for MacOSX)
- Initial configuration
 - `git config --global user.name "rinziv"`
 - `git config --global user.email "rinzivillo@isti.cnr.it"`
 - `git config --global`

GIT – CREATING A REPOSITORY

- GIT handles two kinds of repositories
 - Local repository
 - Contained within a folder `.git` in the root of the project folder
 - Only on person access this repo
 - Remote repository
 - Located into a remote server
 - Locally stored within the `.git` folder
 - Team members work concurrently on remote repository

GIT – CREATE A LOCAL REPOSITORY

- Move within the project root directory
- Use `git init` to start versioning tracking
- The root of the project is called `working copy`
 - There is only one working copy at any moment
 - It is possible to update the current working copy with previous versions from the repository
- Some files (usually related to the OS) can be ignored for the versioning
 - Create a file called `.gitignore` in the root of the project folder
 - List the files to ignore within the file

GIT – CLONE A REMOTE REPOSITORY

- A remote repository have a URL of the form:
 - `ssh://user@server/git-repo.git`
 - `user@server:git-repo.git`
 - `http://example.com/git-repo.git`
 - `https://example.com/git-repo.git`
 - `git://example.com/git-repo.git`

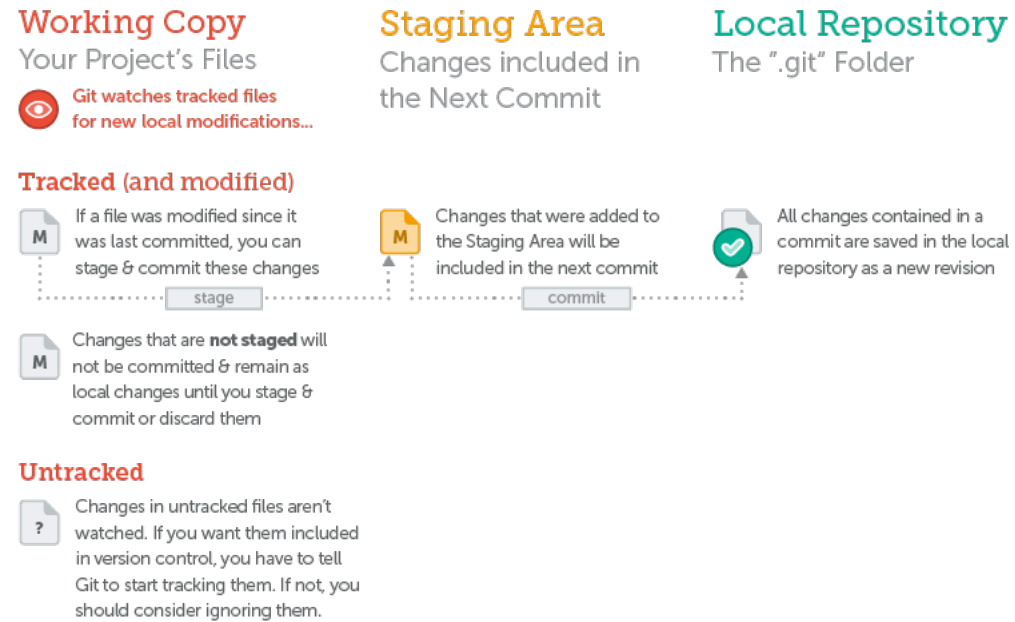
GIT - COMMIT

- Commit operation save the snapshot of the working copy on the repository
 - `git add -A`
 - `git commit -m "Initial commit"`

GIT – STATUS OF THE PROJECT

- Each file within the project have one of the following state
 - **Untracked**: the file is not under version control. GIT do not track any change on this file
 - **Tracked**: GIT reports changes on these files
- **git status** reports the list of files within the project that have changed and those that are not tracked

GIT – STAGING AREA



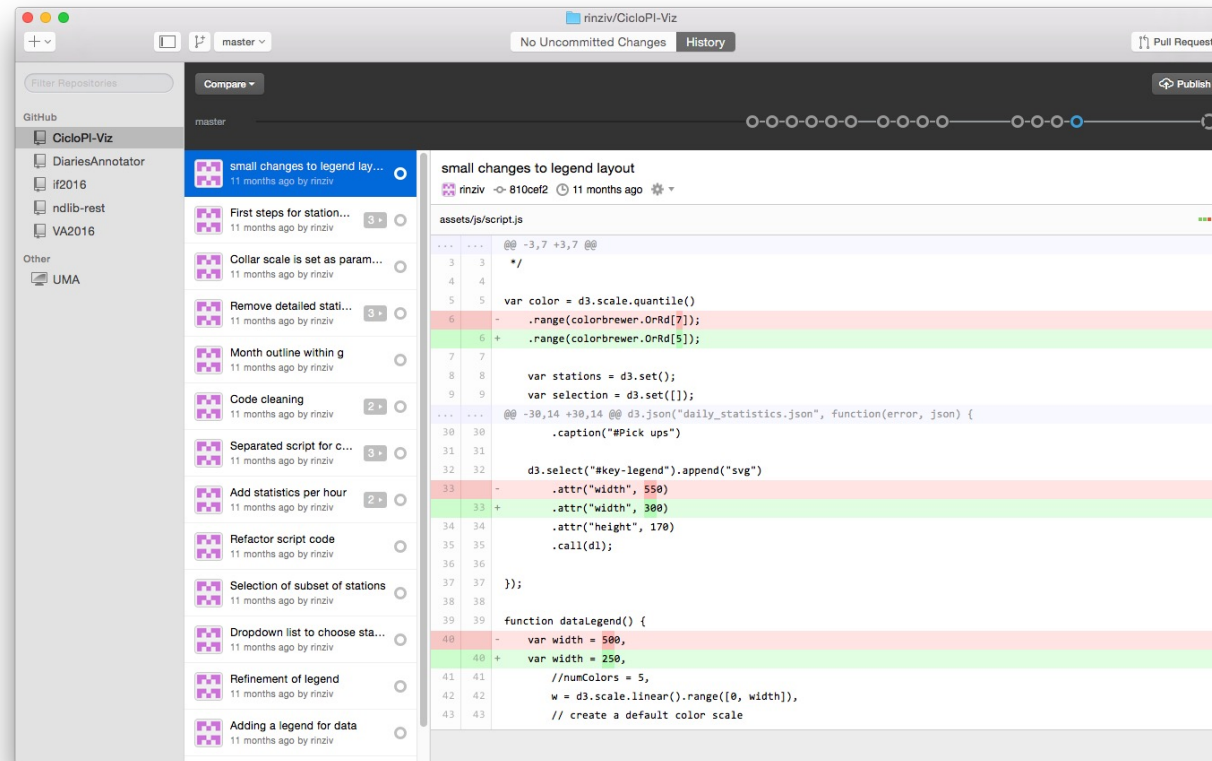
GIT – PREPARING THE STAGING AREA AND COMMIT

- Add the files to include in the staging area
 - `git add new-page.html index.html css/*`
 - `git rm error.html`
 - Check the status with: `git status`
- Commit changes
 - `git commit -m "Implement the new login box"`

WHEN TO COMMIT?

- Each commit should contains changes related to a single topic
- Save only completed work (for temporary saving use Stash)
- Test the project before committing
- Add descriptive message
- Commit often

DESKTOP GUI VERSION



FORK

The screenshot displays a GitHub repository for 'swift' with a 'master' branch. The commit history is shown in a table format:

Commit Hash	Author	Message	Date
806cf57	swift-ci	[Runtime] Make swift::swift_conformsToSwiftProtocol overridable.	28 Nov 2018 at 09:36
da772e9	Doug Gregor	[Runtime] Make swift::swift_conformsToSwiftProtocol overridable.	28 Nov 2018 at 08:37
f9797b4	Doug Gregor	[Runtime] Add dummy "module name" parameter to swift::_conformsTo...	28 Nov 2018 at 01:11
5528db1	swift-ci	Merge remote-tracking branch 'origin/master' into master-next	28 Nov 2018 at 08:30
2889a1f	Harlan Haskins	Merge pull request #20816 from harlanhaskins/the-missing-link	28 Nov 2018 at 08:12
717532e	Harlan Haskins	[ParseableInterface] Test that module-link-name is serialized and used	28 Nov 2018 at 01:12
3e15224	Harlan Haskins	[ParseableInterface] Serialize module-link-name in binary module	28 Nov 2018 at 00:08
618f3a6	swift-ci	Merge remote-tracking branch 'origin/master' into master-next	28 Nov 2018 at 07:09
e4efac5	Saleem Abdulrasool	Merge pull request #20717 from trougton/patch-2	28 Nov 2018 at 07:04
89d6cdf	Thomas Roughton	Define mappings for CF types on LLP64	25 Nov 2018 at 08:00
0b68085	Saleem Abdulrasool	Merge pull request #20813 from drodriguez/fix-android-gold-link...	28 Nov 2018 at 07:00
def1172	Daniel Rodríguez Tro...	Split link flags and link libraries lists to make gold happy.	28 Nov 2018 at 00:24

The selected commit (2889a1f) is a merge pull request from Harlan Haskins. The commit message is "[ParseableInterface] Ensure module-link-name is serialized and used". The commit details show the author as Harlan Haskins (harlan@apple.com) and the committer as GitHub (noreply@github.com). The commit SHA is 2889a1f906c481df1037b718bbb3b6289a8d6a27, and the parent commit is e4efac5 717532e. The file tree shows the file lib/Frontend/ParseableInterfaceSupport.cpp.

EXERCISE – CREATE A REPOSITORY FOR OUR PROJECT

- Demo

CLASSROOM REPOSITORY

<https://github.com/va602aa-master>