

# JS

DEBUG & OUTIL DE DEV.

Nicolas Fazio  
[@FazioNico](#)

# PROGRAMME

- Les bases
- Gestion des erreurs
- Outils de développement

A yellow square logo with the letters 'JS' in a dark gray, bold, sans-serif font.

JS

**LES BASES**

# LES ERREURS EN JS

Par default, les erreurs JS ne s'affiche pas dans votre page web.

Votre code peut donc contenir des erreurs de syntaxe ou des erreurs logiques, qui sont du coup difficiles à diagnostiquer.

Souvent, lorsque le code JavaScript contient des erreurs, rien ne se passera. Il n'y a aucun message d'erreur, et vous obtiendrez aucune indication où chercher les erreurs.

Mais alors comment faire pour afficher les erreurs et ainsi les corriger?

# JAVASCRIPT DEBUGGER

Debugger une application n'est pas facile. Mais heureusement, tous les navigateurs modernes ont un débogueur intégré.

Les débogueurs intégrés peuvent être activés et désactivés.

Avec un débogueur, vous pouvez également définir des points d'arrêt (lieux où l'exécution de code peut être arrêté), et examiner les variables.



# GESTION DES ERREURS AVEC CHROME DEVTOOLS

# PRÉSENTATION

Ouvrir Chrome DevTools avec `cmd+alt+i`

**Elements:** permet de voir la page Web que le navigateur voit. En utilisant l'outil Eléments, vous pouvez voir le HTML brut, les styles CSS et plus encore.

**Console:** affiche les erreurs JS et permet d'exécuter du code JavaScript et d'interagir avec votre page.

**Network:** liste les composants de votre page, requêtes http, combien de temps ces demandes prennent, etc...

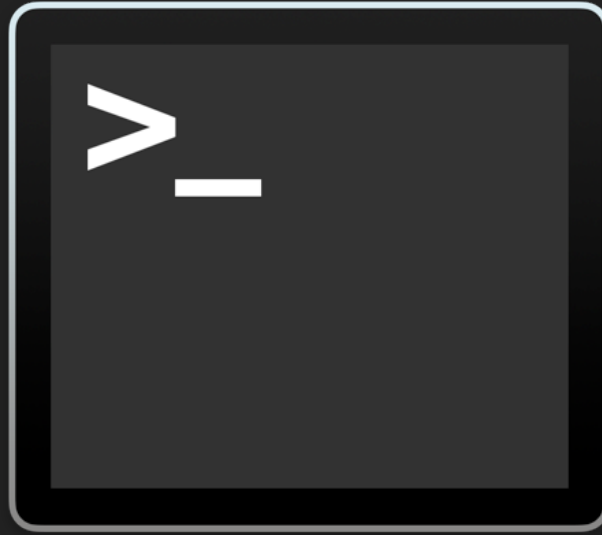
**Sources:** liste tous les fichiers de ressources et propose un debugger step by step.

**DEMO!**



# TP

- ouvrir projet demo-devtools
- trouver la ligne du fichier qui fait bugger l'application
- utiliser les breakpoint pour connaître la valeurs de variable et identifier d'ou vient le problème
- debugger l'application



# OUTILS DE DEVELOPPEMENT

## CLI

# CLI : COMMAND LINE INTERFACE

Un outil rebutant de prime abord mais qui est incontournable en programmation.

```
Terminal - carl@manjaro:~
File Edit View Terminal Go Help
[carl@manjaro ~]$ sudo pacman -S mate mate-extras
:: There are 31 members in group mate:
:: Repository mate
   1) libmate  2) libmatecanvas  3) libmatecomponent  4) libmatecomponentui
   5) libmatekbd  6) libmatekeyring  7) libmatenotify  8) libmateui
   9) libmateweather 10) libmatewnck 11) mate-backgrounds 12) mate-common
  13) mate-conf 14) mate-control-center 15) mate-corba 16) mate-desktop
  17) mate-dialogs 18) mate-doc-utils 19) mate-file-manager
  20) mate-icon-theme 21) mate-keyring 22) mate-menus 23) mate-mime-data
  24) mate-notification-daemon 25) mate-panel 26) mate-polkit
  27) mate-session-manager 28) mate-settings-daemon 29) mate-themes
  30) mate-vfs 31) mate-window-manager

Enter a selection (default=all):
:: There are 18 members in group mate-extras:
:: Repository mate
   1) mate-applets 2) mate-bluetooth 3) mate-calc 4) mate-conf-editor
   5) mate-document-viewer 6) mate-file-archiver 7) mate-file-manager-sendto
   8) mate-image-viewer 9) mate-media 10) mate-menu-editor
  11) mate-power-manager 12) mate-system-monitor 13) mate-terminal
  14) mate-text-editor 15) mate-utils 16) python-caja 17) python-corba
  18) python-mate

Enter a selection (default=all):
resolving dependencies...
looking for inter-conflicts...
:: mate-dialogs and zenity are in conflict. Remove zenity? [y/N] y
```

# CLI : COMMAND LINE INTERFACE

C'est est une interface "homme"=>"machine" dans laquelle la communication entre l'utilisateur et l'ordinateur s'effectue en mode texte.

1. l'utilisateur tape une ligne de commande, c'est-à-dire du texte au clavier pour demander à l'ordinateur d'effectuer une opération
2. l'ordinateur affiche du texte correspondant au résultat de l'exécution des commandes tapées ou à des questions qu'un logiciel pose à l'utilisateur.

# CLI : COMMAND LINE INTERFACE

Par convention, une ligne de commande est décrite avec un caractère spécial qui la précède, c'est le \$

Exemple: \$ `mkdir MonDossier`

Mais le \$ ne sera pas tapé dans le terminal, il est uniquement présent pour signifier qu'il s'agit d'une ligne de commande.

Pour ouvrir le terminal:

- cmd+space
- terminal
- double clic

# CLI : COMMAND LINE INTERFACE

Voici quelques ligne de commandes incontournables:

```
# liste les fichiers d'un dossier
$ ls

# nettoie votre fenêtre de terminal
# en reléguant tout le texte au dessus
$ clear

# se déplacer
$ cd repertoire
$ cd ..

# Créer un dossier
$ mkdir MonDossier

# Supprimer un dossier
$ rmdir MonDossier

# Créer un fichier
$ touch index.html

# Supprimer fichier
$ rm index.html

# exécuter une commande en tant qu'admin
$ sudo
```

<https://buzut.fr/101-commandes-indispensables-sous-linux/>

# TP

- exécuter les quelques lignes de commande précédemment vues



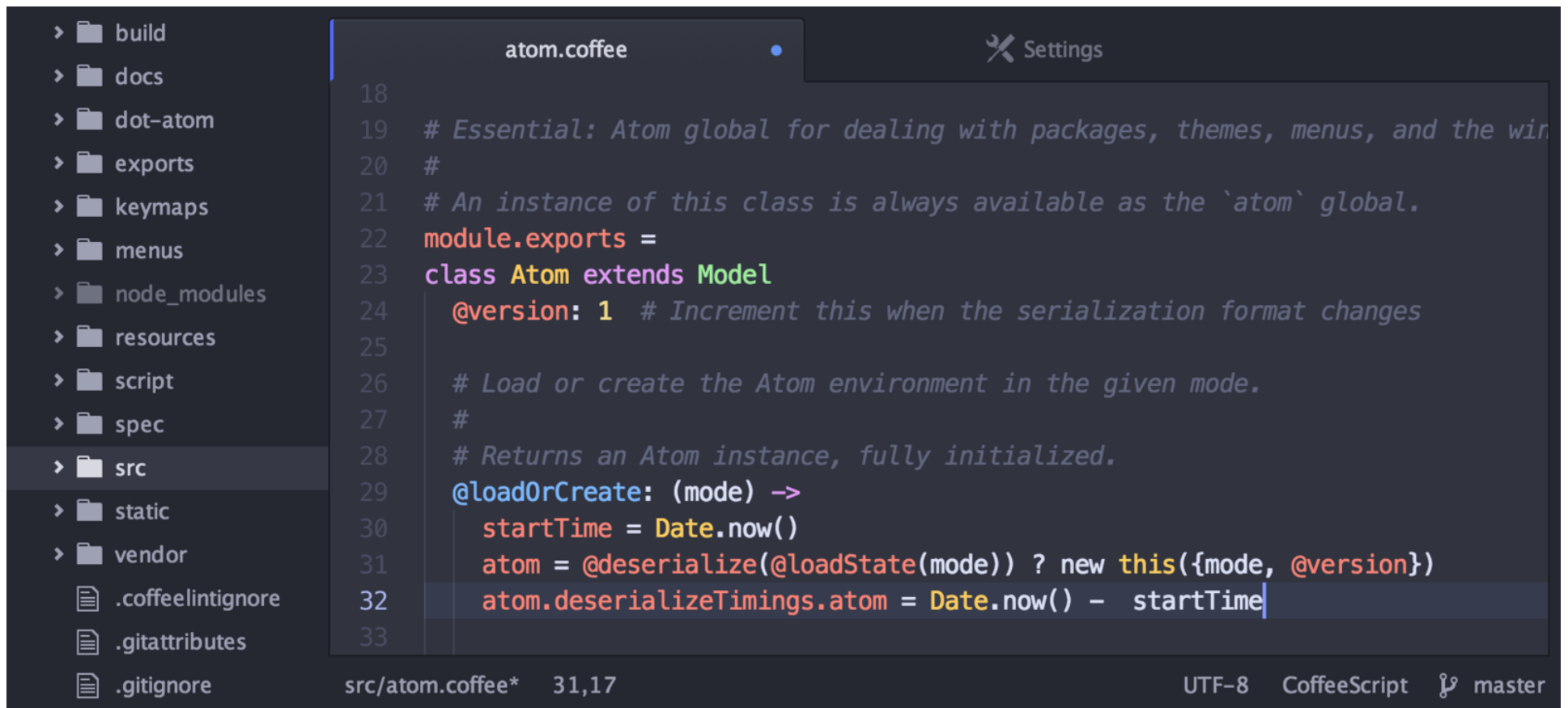
# OUTILS DE DEVELOPPEMENT ATOM



# EDITEUR DE TEXT ATOM

Télécharger et installer Atom

<https://atom.io/>



The screenshot shows the Atom text editor interface. On the left is a sidebar with a file tree containing folders like build, docs, dot-atom, exports, keymaps, menus, node\_modules, resources, script, spec, src (selected), static, and vendor, along with files like .coffeelintignore, .gitattributes, and .gitignore. The main editor area displays the 'atom.coffee' file with CoffeeScript code. The code includes comments about the Atom global and an instance, followed by a class definition for 'Atom' extending 'Model'. It features a version attribute, a comment about serialization format changes, and a '@loadOrCreate' method that initializes the Atom instance by deserializing the state or creating a new one with the current mode and version. The status bar at the bottom indicates the file path 'src/atom.coffee\*', line and column '31,17', encoding 'UTF-8', language 'CoffeeScript', and branch 'master'.

```
18
19  # Essential: Atom global for dealing with packages, themes, menus, and the win
20  #
21  # An instance of this class is always available as the `atom` global.
22  module.exports =
23  class Atom extends Model
24    @version: 1 # Increment this when the serialization format changes
25
26    # Load or create the Atom environment in the given mode.
27    #
28    # Returns an Atom instance, fully initialized.
29    @loadOrCreate: (mode) ->
30      startTime = Date.now()
31      atom = @deserialize(@loadState(mode)) ? new this({mode, @version})
32      atom.deserializeTimings.atom = Date.now() - startTime
33
```

# PLUGINS ATOM

**Lister:** 'atom->préférences->packages'

**Installer:** 'atom->préférences->install'

## LISTE DE PLUGIN:

**Emmet:** <https://atom.io/packages/emmet>

**Auto Close HTML:** <https://atom.io/packages/autoclose-html>

**Highlight Selected:** <https://atom.io/packages/highlight-selected>

**Linter:** <https://atom.io/packages/linter>

**JSlint:** <https://atom.io/packages/jslint>

**ESlint:** <https://atom.io/packages/linter-eslint>

**Atom TernJS:** <https://atom.io/packages/atom-ternjs>

**PlatformIO IDE Terminal:** <https://atom.io/packages/platformio-ide-terminal>

**Git plus:** <https://atom.io/packages/git-plus>

**DEMO!**

**DES QUESTIONS?**

# TP

- installer Atom sur votre machine
- installer la liste des plugins atom
- paramétrer les plugins pour un projet ES6
- tester les plugins

**FIN ;-)**