Making Emacs Look Nice

Emacs is an amazing text editor. And also one that is quite difficult to master. One of the very first things I noticed when I started using Emacs, was that it looked quite ugly, specially when compared to other editors (like Visual Studio, Atom, Sublime). The good news is that Emacs is extremly customizable, and with just a few lines of code and some packages, we can turn Emacs into the prettiest editor we will ever use.

1 Getting the Right Emacs Version

Let's begin by getting the proper version of Emacs for Mac. If you google "Emacs for Mac", you will probably run into the Emacs for Mac OSX page. This page offers a pure version of Emacs for Mac, but that is not the version we are looking for. Why? Because if you are using a Mac, your screen probably has a very high resolution, and that Emacs version does not support that. Another problem is that when we try to preview Latex equations inside org-mode, they will look pixelated, and we want to avoid that. So, we want to download another Emacs binary that has better support for the Mac high resolution display.

If you google "Emacs Yamamoto" you will run into the Railwaycat Homebrew Emacs Mac Port, which is a github repository that contains the right Emacs version for us. Follow this link to the download page and get the newest version. Right now I am using Emacs 25.2 Mac 6.4, but any Emacs 25 or newer should be fine for this tutorial.

Now that you downloaded Yamamoto's Emacs version and moved it to your `Applications` folder, we can start messing around with the `init.el` file. This file is located in the folder `~/.emacs.d/`, and if it does not exist, you can create it. The `.el` means that the file is going to hold a bunch of Elisp commands, which modify and extend Emacs, customizing it to your needs.

2 Getting Access to the Major Packages Repositores

In order to make Emacs look pretty, we will need to install some packages. These packages are located in repositories that Emacs can easily access. To enable this easy access, we add the following to the `init.el` file:

```
;; Add Main Repositories for Emacs packages
(package-initialize)
(setq package-archives '(("gnu" . "http://elpa.gnu.org/packages/")
                         ("marmalade" . "http://marmalade-repo.org/packages
   /")
                         ("melpa" . "http://melpa.org/packages/")))
```

In order to see what packages are available in those repositories, run the command `M-x list-packages`. If you see a package you want to install, move the cursor to the package's line and press `x`. Emacs will then prompt you for permission to install, and then you are done.

If you know the name of the package you want to install, you can skip listing all packages and go directly for the one you want. To do so, run the command `M-x package-install RET`, and then you will be prompted for the name of the package you want to install.

Installing packages is an important part of using Emacs, allowing us to use good code written by other people, making our lives much easier! Making Emacs look pretty will involve installing a handful of packages, so be ready for the `M-x package-install`!

3 Making Emacs Look Good

Now that we have access to the big repositories, we can start modifying Emacs. Modifying Emacs requires adding elisp code to the `init.el` file, and installing new packages when required. We can check whether a package has to be installed by running the code related to it. For example, after adding a line of code to your `init.el` file, move the cursor to the end of the line, and type `C-x C-e`, which will evaluate the elisp statement. If the statement produces the desired effect, then the package or feature you are modifying is already installed, and nothing else needs to be done. However, if you get an error saying the variable or function does not exist, or if nothing happens to Emacs, then the package you are trying to modify is probably not installed. Then, you should install it using `M-x package-install` or through `M-x list-packages`.

3.1 Removing the Toolbar, Scrollbar and the Menu Bar

To make Emacs look less cluttered, we begin by removing the toolbar, the scroll bar and the menu bar. Add the following to your `init.el`:

```
;; Removes toolbar, scroll bar and menu bar
(tool-bar-mode -1)
(scroll-bar-mode -1)
(menu-bar-mode -1)
```

This removes all those bars that clutter our space, making Emacs look much cleaner.

3.2 Borderless Fullscreen (Removing Title Bar)

Next, we make Emacs occupy the entire screen, but without moving it to a new Desktop area. We want to achieve a fullscreen borderless effect, so that Alt-Tabbing from one application to the other is fast. To do so, we add:

```
;; Maximizes emacs on startup and removes title bar (borderless fullscreen)
(set-frame-parameter nil 'fullscreen 'fullboth) ;; this works for the emacs
    version from Yamamoto
```

If you are not using the Emacs binary we discussed above this command will not work. If for some reason you are not willing to use that Emacs, and are using Emacs from Emacs for Mac OSX, for example, you can use this command instead:

```
;; the code below works for the emacs version from emacsformacosx.com
(add-hook 'window-setup-hook (lambda()
                        (setq ns-auto-hide-menu-bar t)
                        (set-frame-position nil 0 -24)
                        (set-frame-size nil (display-pixel-width) (
    display-pixel-height) t)))
```

Now, when you open Emacs, it should occupy the entire screen, and no menus or scrollbars should be visible.

3.3 Highlighting the Current Line

Next, we want Emacs to highlight the line where the cursor is.

```
;; Highlight Current Line
(add-hook 'after-init-hook 'global-hl-line-mode)
```

This makes jumping from line to line and quickly finding where the cursor is much faster.

3.4 Highlighting Matching Parentheses

We also want Emacs to highlight matching parentheses when the cursor is on one:

```
;; Highlight corresponding parentheses when cursor is on one
(setq show-paren-delay 0) ;; shows matching parenthesis asap
(show-paren-mode t)
```

Now it is easy to see when a parenthesis is missing in the text.

3.5 Better Line Wrapping

By default, Emacs wraps lines that are too long by using a small arrow that is displayed near the scroll bar. I find this behavior slightly ugly and not very helpful. We can use a better wrapper that just moves the overflowing content to the next line:

```
;; Proper line wrapping
(global-visual-line-mode t)
```

3.6 Making Unnecessary White Spaces Visible

When we write large files, or using programming languages that care about white spaces and indentations, it is useful to be able to see white spaces that are useless and out of place. The next line does that:

```
;; Show trailing white spaces
(setq-default show-trailing-whitespace t)
```

Now, whenever an unnecessary whitespace occurs, it will be marked in red.

Since deleting useless whitespaces by hand each time is time-consuming and annoying, we can automatically delete all of them before we save the file. This can be accomplished via:

```
;; Remove useless whitespace before saving a file
(add-hook 'before-save-hook 'whitespace-cleanup)
(add-hook 'before-save-hook (lambda() (delete-trailing-whitespace)))
```

Now, if you save a file that contains whitespaces, they are all going to be automatically deleted. This reduces file sizes and makes them neat.

3.7 Nice Icons for Emacs

Some visual modifications of Emacs require displaying icons, like an icon for a folder or a file. The default icons do not look nice and sometimes are not even available to Emacs. Fortunatelly, there is a package called `all-the-icons` that takes care of making nice icons available for Emacs and other Emacs' packages.

To install this package run `M-x package-install RET all-the-icons`, and then confirm the installation. Next, we need to install the fonts that came with this package on our Mac. To do so, run `M-x all-the-icons-install-fonts RET`. Installing all the icons fonts takes a while, but it is worth it! Now, we can make it available to other packages via the following:

```
;; Adds Nice Icons to Emacs so that other themes can use them (required for
    Doom theme below)
;; run M-x all-the-icons-install-fonts RET to install the fonts
(require 'all-the-icons)
```

3.8 Folder Explorer (File Tree)

Now we add a file explorer for Emacs called Neotree. It can be installed by running `M-x package-install RET neotree RET` and confirming the installation. To actually enable it inside Emacs, add the following:

```
;; Folder explorer (file tree) Neotree
(require 'neotree)
(Global-set-key [f8] 'neotree-toggle) ;; the F8 key toggles the file tree
```

Now, pressing F8 on the keyboard will open Neotree inside Emacs. Neotree can be used to navigate through folders and open files. It is specially useful when working on big projects since it gives a visual representation of the folders/files. You can find more about Neotree Shortcuts here.

3.9 Emacs Doom Theme

All the changes we have done so far are small quality of life improvements that affect Emacs here and there. Now, we will install a theme that will affect Emacs visually in big ways. This theme will also change how Neotree looks like, and will make use of all-the-icons package.

We will install a very nice theme package called Doom. To install it, run `M-x package-install RET doom-themes`. And to start using it, add the following to your `init.el`:

```
;; Use doom theme for Emacs buffers
(require 'doom-themes)
(load-theme 'doom-one t) ;; loads doom-one theme, there are other options:
                         ;; doom-vibrant, doom-molokai, doom-nova, and
   others
;; Use doom theme for Neotree file explorer
(doom-themes-neotree-config) ;; loads doom theme for the file tree Neotree
(setq doom-themes-enable-bold t    ; if nil, bold is universally disabled
      doom-themes-enable-italic t) ; if nil, italics is universally
   disabled
;; Enable all-the-icons beautiful icons for the Neotree doom theme
(setq doom-neotree-enable-file-icons t)
(setq doom-neotree-enable-folder-icons t)
(setq doom-neotree-enable-chevron-icons t)
;; Enables different colors for different file types for the Neotree doom
   theme
(setq doom-neotree-enable-type-colors t)
```

In order to execute all this code, move the cursor to anywhere in the `init.el` file and run `M-x eval-buffer`. Now, Emacs and the Neotree explorer are going to look completely different! The theme we are using is called `doom-one`, and looks very neat and professional. There are other themes available that you can quickly change to. To do so, just change `doom-one` to the name of some other doom-theme, like `doom-molokai`. More information is available on the doom-themes repository.

If you open Neotree, you will also notice that all the icons have changed. More details on how to further customize Neotree are available here.

3.10 A Better Mode-Line

Emacs has a mode-line on the bottom of any buffer that displays the name of the file you are working on, what major and minor modes are being used, and so on. Let's make it look better! To do so, we install the package `smart-mode-line` by running `M-x package-install RET smart-mode-line RET`, and then confirming the installation. Once installed, we add:

```
;; Smart-mode-line makes the mode-line better to read
;; first remove annoying message at startup that
;; asks if you want to run the theme lisp code
(setq sml/no-confirm-load-theme t)
;; load smart-mode-line
;; (setq sml/theme 'dark) ;; changes the theme to dark
(sml/setup) ;; automatically figures out a theme if none is specified
```

Now your mode-line will also be themed. Notice that smart-mode-line automatically detects your color scheme and chooses a nice theme for the mode-line. If you want to change the theme to something else, just uncomment the line before `(sml/setup)`. More details and other themes are available on the smart-mode-line repository.

3.11 Nice Looking LaTeX Equations in Org-Mode

Some of you might be using org-mode to take notes and write articles. If you have LaTeX configured to work with Emacs, you can use `C-c C-x C-l` to preview equations inside org-mode. However, by default Emacs will render Latex equations and display them using PNG images. PNG is a good image format, but the problem is that these renders have a very low resolution when compared to a Mac's display resolution, so they will look very bad. The solution is to use SVG images instead, which are scalable to any resolution!

Using SVG images requires Emacs to have been installed with SVG support. Unfortunately, SVG support cannot be added to Emacs like we did with packages, it has to be "pre-installed". The good news is that if you followed this tutorial and downloaded the Emacs binary mentioned at the beginning of the post, then you are good to go!

There is one extra requirement: having the program `dvisvgm` installed in your machine. If you are running the latest Latex distribution from MacTex, then you most likely already have it, and are also good to go. If you installed Latex from some other source, I highly recommend you to download the latest release from MacTex. At the time of writing, the latest release was the MacTex 2017 distribution.

To switch to SVG images just add the following to your `init.el` file:

```
;; Makes Orgmode Preview of Latex equations use SVG images instead of PNG
;; so that equations look good on higher dpi screens
(setq org-latex-create-formula-image-program 'dvisvgm)
;; to increase the size of the equations (1.0 is the default)
(setq org-format-latex-options (plist-put org-format-latex-options :scale
    1.0))
```

Now, if you run `C-c C-x C-l` you should get very sharp images that look great on Emacs.

4 Conclusion

In this tutorial we made the following changes to Emacs:

1. Removed unnecessary toolbars;
2. Made Emacs borderless fullscreen;
3. Added highlighting to the current line and matching parentheses;
4. Improved Emacs' line wrapping;
5. Made removing unnecessary white spaces quick and easy;
6. Added beautiful icons to Emacs;
7. Added a Folder Explorer to Emacs to better manage bigger projects;

8. Started using a theme that makes Emacs buffers look good, that makes Neotree look good, and also makes use of the beautiful icons we installed;

9. Changed the mode-line so that it also has a theme;

10. And changed the way Emacs render Latex equations so that they look sharp.

All these changes improved Emacs by a lot, taking it from a bland white screen, to a colorful and captivating editor that we actually want to use. I hope this post was useful to you, and if it was feel free to share it. If you have any suggestions or comments send me an email. Have a wonderful time!