Chapter 2

Introduction

When you need to write or to code, you have to choose a text editor, and a very good one. They are many text editors available out there, but very few of them are more than 40 years old. It's the case of *Emacs* (https://www.gnu.org/software/emacs/), *Vi*, and its improved successor *Vim* (https://www.vim.org). They were created in the 70's and are still used a lot nowadays. You may have already noticed that it's not thanks to the beauty of their website or the efficiency of their communication (although I must admit that the *Emacs* site has made efforts since the first version of this book). Here are some **reasons for their success**:

Forever
: You learn them once and you use them forever. In a world where languages and technologies are constantly changing, learning vim is a real chance to invest in a skill you'll be able to use forever.

Everywhere
: They are available for each and every possible platform : Mac Os X, Windows, GNU/Linux, BSD, and so on, and it's always been that way.

Efficient
: Thanks to their features (like the extensive use of the keyboard), you can edit and write text as fast as your thoughts.

For everything
: They allow you to edit everything and anything. When you'll use another programming language, or another markup language, you won't have to change your editor. Of course, this book has been written using *Vim* (and the ReST Markup).

Yet, these text editors are difficult to learn. Not that they are harder than anything else, not that you can't handle it, but rather because there is no smart way out there to learn them for now. So, here we are.

The aim of this book is to address this gap by guiding you through your discovery of *Vim*. I'll put *Emacs* aside from now and I'll focus on *Vim*. If you want to know more about this **Editor war**, be sure to check the Wikipedia page. This book doesn't claim to be a reference book about *Vim*. There are already a lot of good references on the subject like A byte of Vim. However, it claims to reduce the entry barrier to get used to *Vim*. In my opinion, the most difficult thing about learning *Vim* is not getting discouraged while finding a way to use it, learning *Vim* step by step. We all have to get things done with our text editor on a daily basis, that's why losing all your productivity when switching to *Vim* is not an option.

I'm sure you'll find a lot of people who will tell you: "Just do it cold turkey", "You'll see, it's hard at the beginning, but time will help". True, but you'll still have the problem of trying to remain productive on a daily basis. The approach of this book is the following:

- Have a modern *Vim*: syntax highlighting and nice colors.
- Use *Vim* as any other text editor: easily edit code and switch between files using the mouse.
- Learn keyboard shortcuts and go without the mouse step by step.

- Install the *best* plugins to start using *Vim* to its full potential.

Starting from bullet number 2, you'll already be able to use *Vim* on a daily basis without losing a lot of productivity. It's where the magic will happen: if you can integrate *Vim* in your daily habits, you have won. You'll then have the rest of your life to learn all the shortcuts and the tip and tricks of *Vim*.

You're tired of trying a new editor each year? You're tired of having to relearn everything from scratch every time? You're tired having to change your editor when you're using your Mac, Windows or Linux? So, just stop it, and join the community of people happy with their text editor!

2.1 What about Neovim?

A quick aside about *Neovim* https://neovim.io/ (if you don't know what it is, you can skip this part). I've decided to concentrate solely on *Vim* in this book so as not to frustrate people who can only use *Vim*.

If you're a *Neovim* user, everything in this book is still valid, as *Neovim* is compatible with *Vim*. What's more, modes, text manipulations and everything else that doesn't concern plugins is common to both *Vim* and *Neovim*: learning them for *Vim* or for *Neovim* makes no difference. In fact, you can follow this book using *Neovim* instead of *Vim* without any problems.

The special feature of *Neovim* is that it is more actively maintained than *Vim* and, among other things, uses the **Lua** programming language for plugin management. As a result, plugins written for *Neovim* are not compatible with *Vim* (the reverse is not true: plugins written for *Vim* are compatible with *Neovim*). Part of the *Vim* community has switched to *Neovim*, and plugins for *Neovim* are flourishing much faster than those for *Vim*. For the purpose of this book, this won't make any difference and you can follow it without any problem with *Vim* or *Neovim*. I may be making more *Neovim*-specific content in the future.

2.2 For who?

Every person having to produce text (code, book, reports, slideshows, …) regularly. Developers are of course concerned, but it's not only about them.

For example, if you are a:

Student

If you want to impress your future boss with your resume, it's a must. It's a proof of seriousness to see that a student took the time to learn *Vim* on his or her own. Moreover, you'll have a unique tool to write all what you'll have to write (and that you'll be able to use for the rest of your career): your LaTeX reports, your slideshows, your code (if you need Word or LibreOffice to write you reports, it's time to use LaTeX, Markdown or reStructuredText).

Friendly advice: for your slideshows, don't hesitate to use something like impress.js. It's using HTML/JS/CSS and I highly recommend that you use it to do awesome presentations based on non-proprietary technologies. You can have a look at reveal.js too, and its online editor slide.es.

Teacher

It's time to set an example for your students and to teach them a tool they will use during their entire life. *Vim* is something they'll be able to use a lot more than any programming language.

Coder

It's essential to invest time in your daily tool. You'll be learning keyboard shortcuts anyway, so you should do it for something useful. If this investment is still profitable 10 years from now, it's the perfect investment. If you put time into learning *Vim*, you will be able to use it for many decades.

System and network administrator

If you use *Emacs*, then I can forgive you. If you use nano/pico, there is nothing I can do for you. Otherwise, it's time to get some work done, folks! Remote administration of a Unix system is the perfect use case for *Vim* (a powerful text editor without the need of a graphical interface).

Writer

If you write using Markdown/reStructuredText/WikiMarkup or LaTeX, *Vim* will save you a lot of time. You'll not be able to go back to another editor after it, or you'll want to *Vimify* it at all costs.

Trust me, I have done and still do all these 5 roles, and my best investment has always been, by far, *Vim*.

2.3 What you will be learning

- How to use *Vim* as a "usual" editor first (you know, the type of text editors having syntax highlighting, allowing you to open files, to click using the mouse, …). In short, we will be demystifying *Vim* to allow you to go further.
- How to move from classical text editing to the power of *Vim*, baby step by baby step (it's where addiction begins).
- How to do without the mouse and why it's the best thing that can happen to you when you're programming/writing text.
- How you can easily deduce keyboard shortcuts with some simple rules.

To sum up: if you consider yourself a craftsman, act like one. Learn how to use your tool, once and for all.

2.4 What you will not be learning

- You will not be learning how to install and to configure *Vim* for Windows. It's doable, but I have very limited knowledge about Windows. It may happen, but not yet. Only Linux/Unix will be discussed (and by extension Mac OS X).
- You will not be learning how to use *Vi* (notice the lack of "*m*"). I'll only teach you how to be productive writing text with *Vim*, I won't be teachng you how to impress your friends with *Vi* (and anyway, *Vim* is enough for that). For those who don't get what I'm talking about, *Vi* is the "ancestor of *Vim* (which stands for *Vi - IMproved*)" and is installed by default on all Unix-like systems (even on Mac OS X).
- You will not be learning to know *Vim* by heart: this book is not a reference it's a pragmatic smart way to learn *Vim*.
- You will not learn how to pimp the colors of your *Vim*, although I will go over how to change your theme. I'll use the Solarized theme, it's the best theme for your eyes.

2.5 The hardest part is to get started

So, your are ready for the adventure? Ready to sacrifice one hour to start using *Vim*, one week to be familiar with it, and the rest of your life to be happy with your choice? So here we go! Well, almost, we need to talk a little bit before.

With *Vim* you'll have to struggle. No matter how big your willpower is, you will struggle. Be prepared. The goal of this guide is to diminish this struggle as much as possible, but be aware that you will struggle anyway. No pain, no gain. Here is the method I recommend to tame the beast:

- Try to make using *Vim* a habit. Be sure to follow this guide until the chapter about *The NERD Tree* (the file explorer). Then you'll be able to use *Vim* as you would do with Notepad++, Textmate or Sublime Text for example. You'll be using only 1% of the capacities of *Vim*, but whatever. What really matters is to use *Vim* on a daily basis.

- Be sure to have a printed sheet with all the main *Vim* shortcuts near you. The goal here is not to learn them by heart, but only to have somewhere to look when you'll ask yourself: "There must be a better way to do this".
- Keep the faith. At the beginning you'll be sceptical regarding the usefulness of learning everything from scratch with *Vim*. And then, one day, you'll have that "a-ha!" moment. You'll be asking yourself why all the software you're using can't be controlled using *Vim* shortcuts.
- Keep in mind that it's an investment for your next 20 years. As you know, investments are rarely profitable immediately.

So, enough talking, let's get started!

Chapter 3

Having a usable *Vim*

This may be a surprising approach for you, but for me, the first thing to do is to have a *Vim* usable by a normal human being. It seems that everybody agrees that *Vim* is a very **powerful editor**. And I think that you will agree too if I say that, by default, *Vim* is totally unusable. Let's be honest, without a decent minimal configuration, using *Vim* is **counterproductive**.

In my humble opinion, it's the first obstacle to tackle before anything else. This is what all the trendy editors like VSCode, TextMate, Sublime Text, Notepad++ or Netbeans are proposing: a default environment usable as it is, even if we don't use its full potential for now.

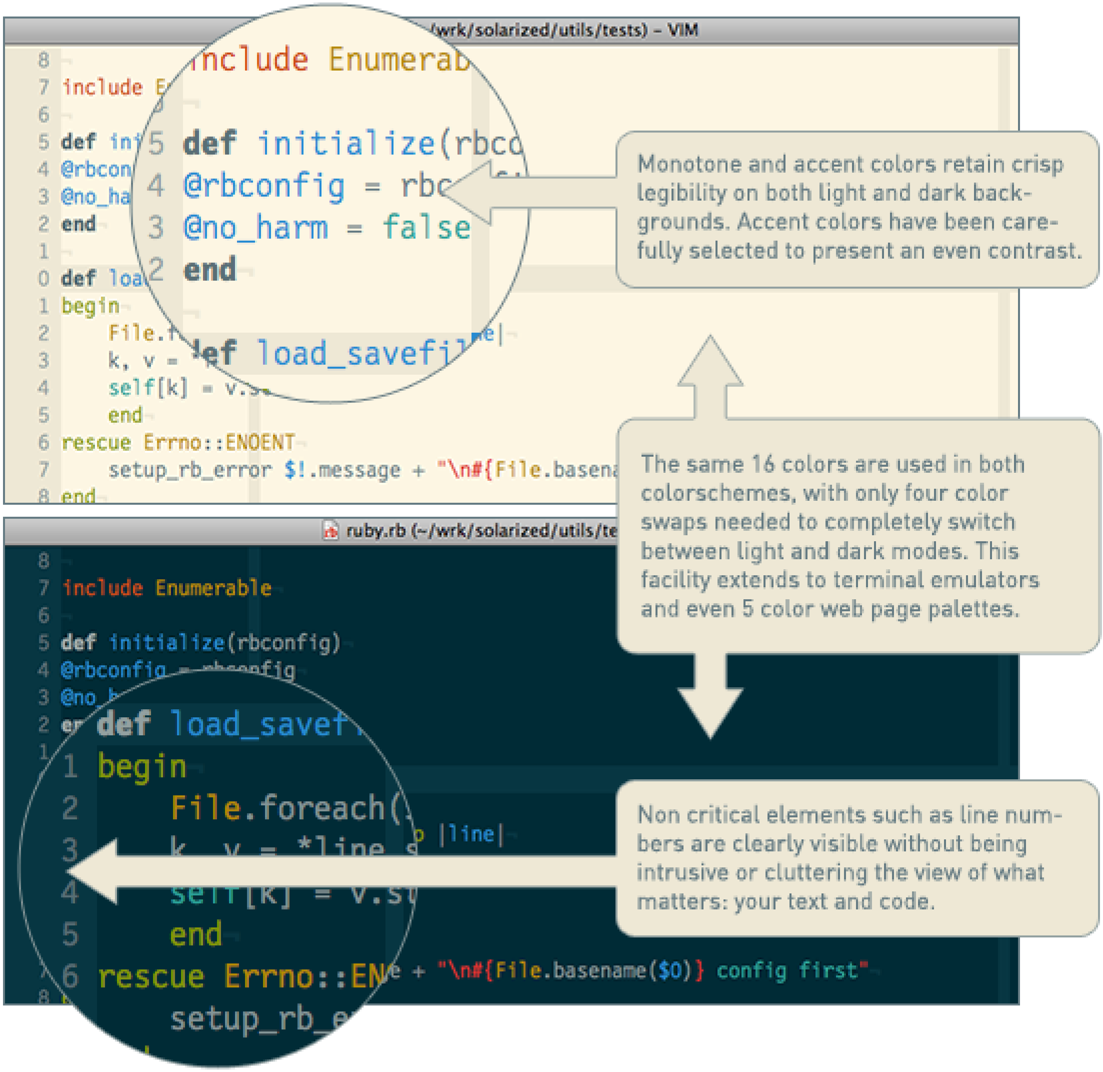
Here is what a default *Vim* is missing (and why **most of people are giving up** before they can really see the power of *Vim*):

Default configuration

You can configure *Vim* thanks to a file named `~/.vimrc`. This file is, obviously, totally empty by default. The first thing to do will be to have a `~/.vimrc` file with a minimal configuration.

Syntax highlighting

By default, *Vim* is white and ugly. To fix that, we will use the Solarized theme. If your goal is to be efficient, it's the best theme available out there (across all text editors), period. The beautiful image below will give you an idea of what it looks like (with the dark and the light theme). Personally, I'm using the dark theme.


/wrk/solarized/utils/tests) – VIM
ruby.rb (~/wrk/solarized/utils/te

Monotone and accent colors retain crisp legibility on both light and dark backgrounds. Accent colors have been carefully selected to present an even contrast.

The same 16 colors are used in both colorschemes, with only four color swaps needed to completely switch between light and dark modes. This facility extends to terminal emulators and even 5 color web page palettes.

Non critical elements such as line numbers are clearly visible without being intrusive or cluttering the view of what matters: your text and code.

File explorer

If you are using *Vim* along with a graphical interface (I suppose it's the case for 99% of you) you will by default have a `File` menu available. This menu should allow you to open a file. It is, for sure, a good start. But having a file explorer a la Netbeans or Textmate can be very handy. To mimic the same behavior, we will be using vim-fern. Be aware that, once you will have read this guide, you will not need the mouse anymore.

This chapter is mandatory if you have very few (or not at all) experience with *Vim*. By the end of the chapter, you will have a *Vim* usable on a daily basis. It should be enough to then be able to learn it gradually. Because, of course, there is no magic, you will have to practice to be able to learn *Vim*, and the sooner, the better.

However, if you are already familiar with *Vim* and don't use the mouse anymore, you can skip this chapter. But be sure to give *Solarized* a try, as you would be missing something otherwise.

3.1 Essential preamble: the insert mode

Let's be totally crazy. We will try to create the `~/.vimrc` configuration file with *Vim* itself. As I said earlier, the sooner you start to use *Vim*, the better. I told you it would be totally crazy!

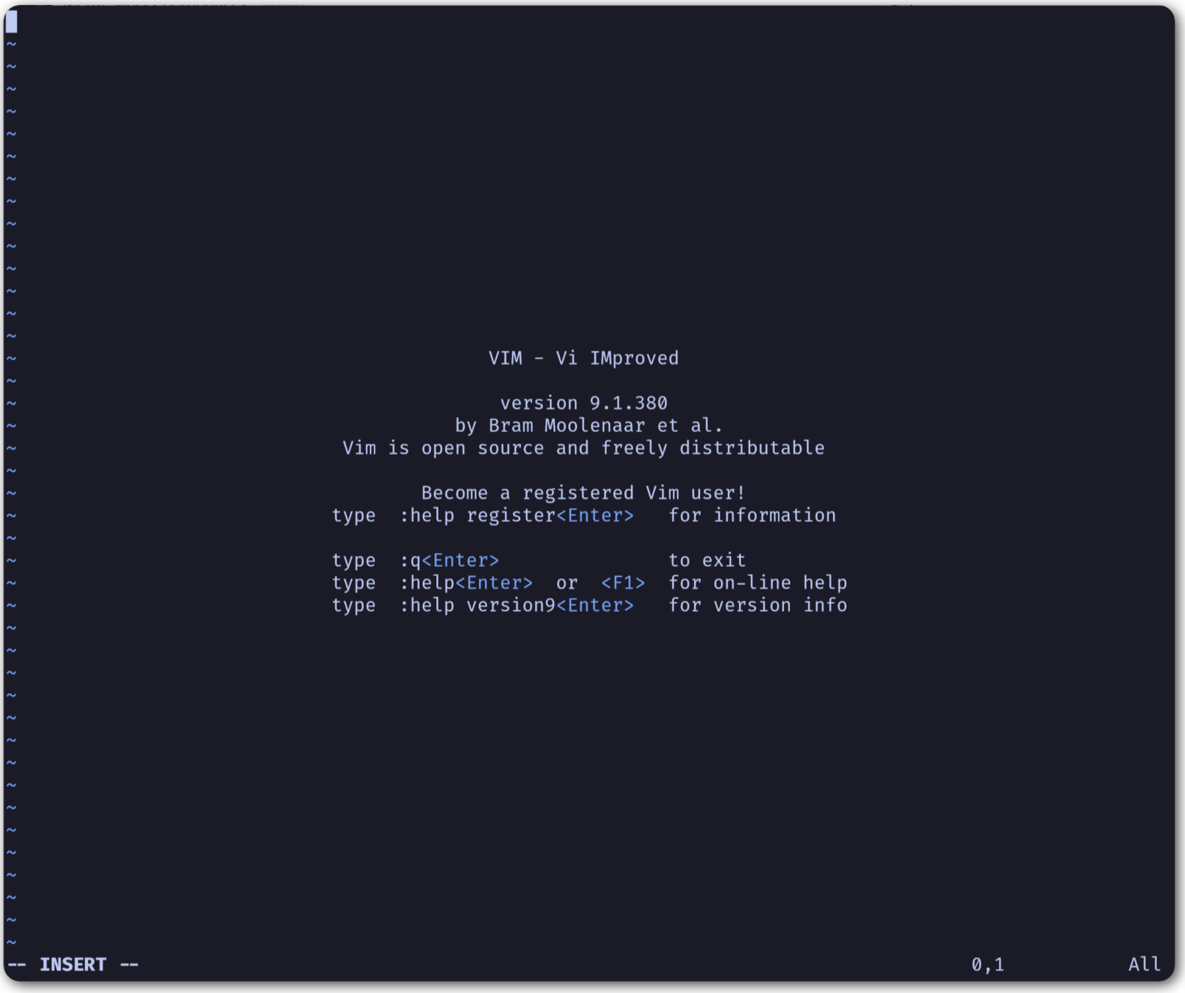
The first thing to do will certainly be to install a *Vim* version for your operating system. If you are using a Mac, give MacVim a try, it's the best *Vim* port for Mac without a doubt. If you are using GNU/Linux or any other "Unix-like" system, you should have the command *vim* or *gVim* available directly from your terminal, or at least easily installable using your package management system (for Ubuntu, the package is called *vim-gnome*). Be sure to install the ___full___ version (ie. with ruby and lua support). For Mac OS X, MacVim has already all what you need builtin. For Windows, it seems that there is a version available on the official *Vim* website (https://www.vim.org/download.php), but I haven't tested it.

Personally, I use *vim* directly on the command line, under Archlinux, in a terminal kitty with the Nerd Fonts FiraCode Nerd Font. This is the configuration used for the screenshots in this book.

When you will start *Vim*, you should see a welcome text asking you to help poor children in Uganda (or something along the lines). This text will disappear as soon as we start writing text in *Vim*.

We will start by adding a comment in the header of the file to specify the author of the document (this should be you). To be able to type text, the first thing to do will be to press the `i` key (the cursor should have changed). You should get *a page* that looks more or less like the figure below. Note the `--INSERT--` at bottom left, which indicates that we're in insertion mode (the mode in which we can enter text). For the record, my terminal's theme is a dark one, so it's possible that your Vim's colors will be different for the time being.

At the time of writing, the version of Vim I'm using is `9.1.380`.



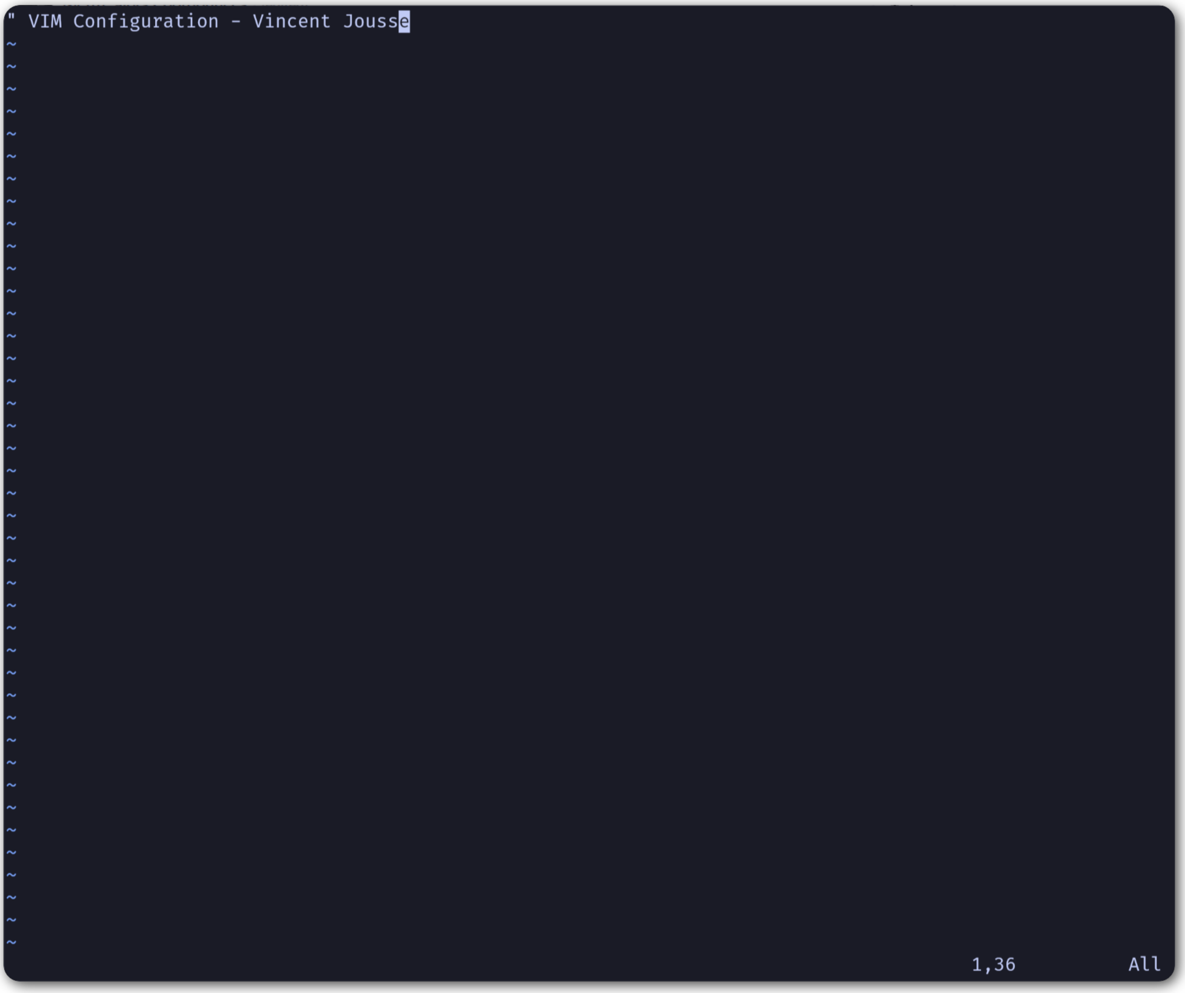
```
                              VIM - Vi IMproved

                               version 9.1.380
                           by Bram Moolenaar et al.
                   Vim is open source and freely distributable

                          Become a registered Vim user!
                  type  :help register<Enter>   for information

                  type  :q<Enter>               to exit
                  type  :help<Enter>  or  <F1>  for on-line help
                  type  :help version9<Enter>   for version info

-- INSERT --                                                   0,1           All
```

On a side note: if you don't really understand what you have done and *Vim* is displaying red messages at the bottom left or doesn't seem to react as it should when you press the `i` key, don't panic. Pressing multiples times on the `Esc` key (two times should be enough) should bring *Vim* to its default mode, the *Normal mode*. Then it should behave as you would expect again.

You should know be able to write down *the comment below* (I'll let you change my name to yours, of course 🤓):

```
" VIM Configuration - Vincent Jousse
```

You will have noticed that comments in *VimL* (the name of the *Vim* programing language) start with a `"`. Then press the `Esc` key to come back to the default mode (the normal mode) of *Vim*. That's all, you are done. Here is a screenshot of what your *Vim* should look like now:



```
" VIM Configuration - Vincent Jousse
~
~
~
                                                        1,36          All
```

I can already hear you: all that fuss for that? Well, yes. And you even don't know how to save a file. But all these things that I'm about to explain to you are logical. One of the advantages of *Vim* is that, usually, it is logical. Once you will have understood the logic behind it, all will be crystal clear for you (at least I hope so).

By default, when you start *Vim*, you are presented with its default mode. This mode is called the *Normal mode*. The purpose of this mode is not to write text (for that, you will have the *Insert mode*), but only to move the cursor and to manipulate text. The power of *Vim* is coming from the combination of these two modes (other modes exist, but it's not the topic for now). You will need some time and some practice to realize the power it's giving to you, so you will just need to trust me in the meantime.

If you are asking yourself why those modes exist, why can't we even write down some text by default, and why we are making things more complicated than they should be, the next chapter is for you.

3.2 Modes: the powerful *Vim* secrets

I suppose you will agree if I say that, if you want to learn *Vim*, it's to be more efficient when writing/-manipulating text or code. To be more efficient when writing text, there are not many solutions. There is only one actually: you need to move your hands as less as you can (even not at all) and only move your fingers.

To do so, of course, you will need to do without your mouse. In addition to being slow, the move keyboard -> mouse and then mouse -> keyboard is really bad for your joints. It's often the cause of musculoskeletal disorders. Maybe you are still young and don't know what I'm talking about, but believe me, you will have such problems one day or another (often sooner than you may think). If you are in front of your computer all day long, don't neglect those possible troubles, you may regret it someday. According to Wikipedia, it's actually the most common professional disease.

You will need to forget the movement of your right hand toward the directional keys (left/right/bottom/top) too. It's a waste of time and it's completely unneeded with *Vim*.

So what do you have the ability to do? Not a lot to be honest (but it's for your own good), you can only leave your hands on the home row *as you can see on the picture below*.

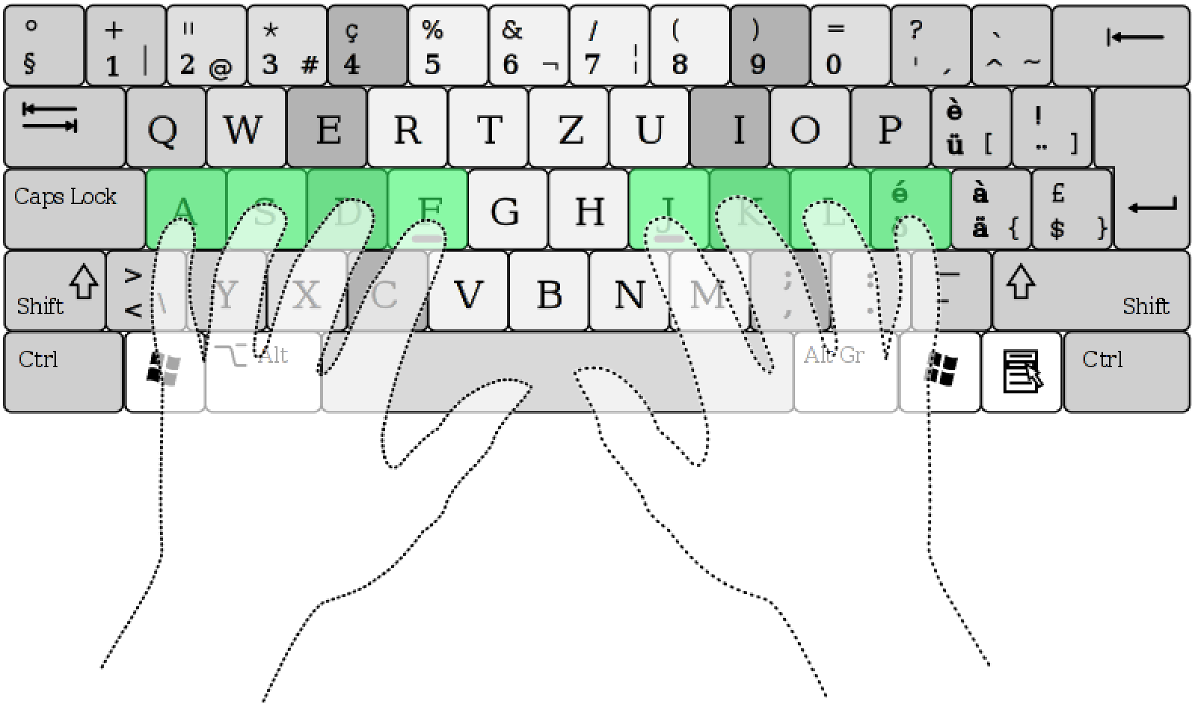


Fig. 1: Home row, QWERTY keyboard
Illustration by Cy21 - CC-BY-SA-3.0 or GFDL, via Wikimedia Commons

You will also probably find on your keyboard some marks on the letters F and J. The goal of these marks is to give a landmark for the position of your fingers (the indexes) on the home row of the keyboard.

Trying to move as less as possible the hands from the keyboard is the reason for having a *normal* mode and an *insert* mode in *Vim*. When switching from one to the other, the keys under your fingers will sometimes allow you to move the cursor and to manipulate text: copy/paste, deletion, … (it's the normal mode), sometimes they will allow you to select some text (it's the *visual mode*) and sometimes to insert some text (it's the *insert* mode). And of course, all of that is possible without the extensive use of keys combinations like *Ctrl + key* that are very bad for your fingers (*Emacs*, this one is for you).

By default, we can switch from the *insert* mode to the *normal* mode by pressing the `Esc` key, but it will be one of the first things we will change: the `Esc` key is to far from your fingers on current keyboards.

To switch from *normal* mode to *insert* mode, we can press the `i` key. We will later learn that there are other ways to do so too. For example, to enter the *insert* mode and to then create a new line below the current one (no matter where is your cursor on the line), we will use the `o` key while in *normal* mode.

I will talk again about this subject later in "*Learning how to move: the copy/paste use case*", but if you are not ready, at some point, to do without your mouse and the directional keys to edit text, I would recommend you to stop learning *Vim* right now. It's as simple as that. You can leverage the full power of *Vim* only by getting rid of the mouse and by moving your hand as little as possible.

If you want to go further, you can buy an orthogonal keyboard like TypeMatrix or Voyager ZSA. It's the keyboard I'm currently using, and my fingers are thanking me everyday.

The ultimate change would be to switch your keyboard layout to a more efficient one like Colemak, but that's another story.

3.3 The lifesaver default configuration

Let's get serious and try to have a usable *Vim*. We will start by editing the default configuration file `~/.vimrc` and by entering default values that any sane person would love to find in it.

You have to place this file in your home directory. It should be */home/your_user/.vimrc* if you are using Linux, */Users/your_user/.vimrc* if you are using Mac OS X. Generally speaking, it should be in your home directory under *~/.vimrc*. If you are using Windows, you'll need to create a file named *_vimrc* that you have to put in your *%HOME%* directory. This directory is obviously not the same across the different Windows versions. Usually, it's the directory just before your *My Documents* directory. More information is available on Wikipedia if you want.

I've directly commented all the lines in the code itself. Nothing fancy here, you should just be asking yourself why all of this is not available by default.

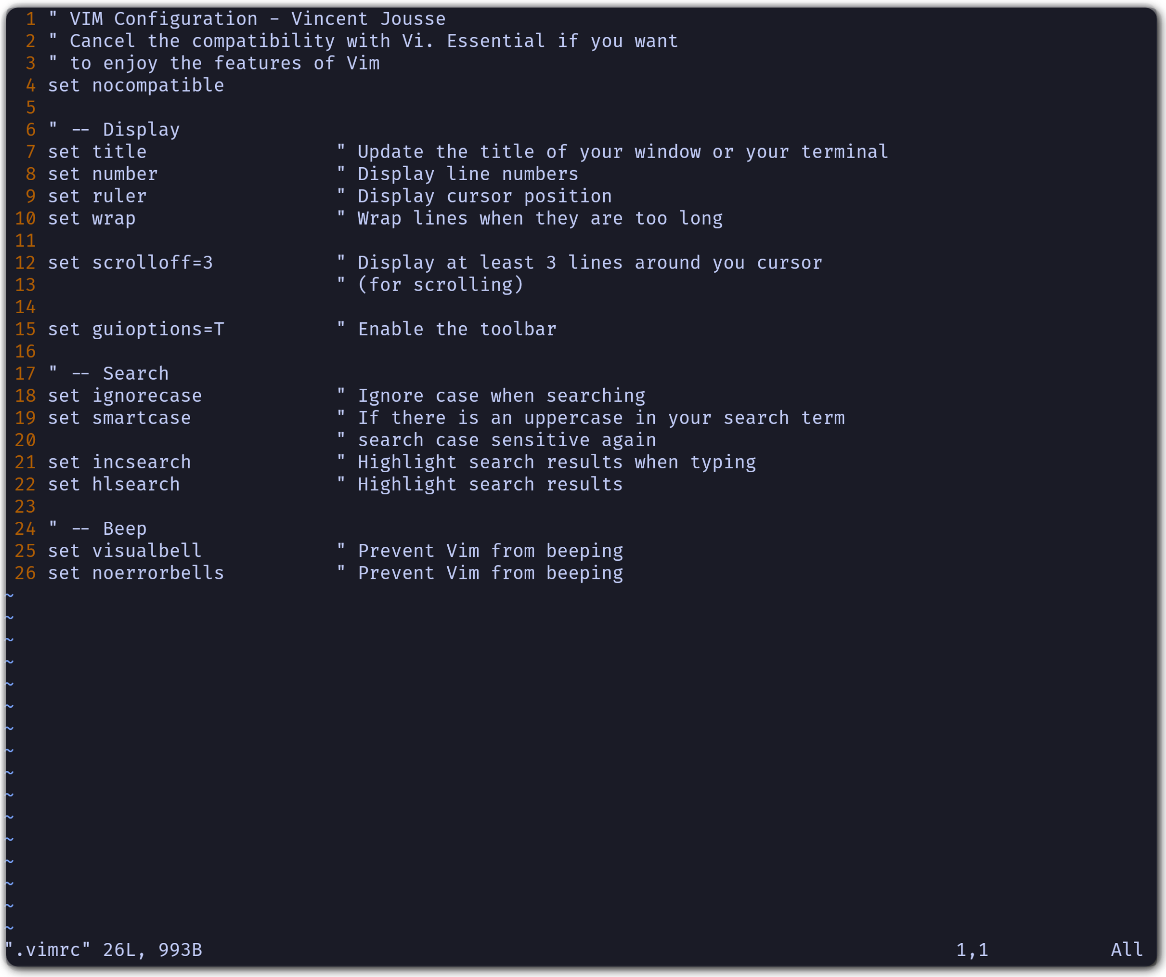
```
" VIM Configuration - Vincent Jousse
" Cancel the compatibility with Vi. Essential if you want
" to enjoy the features of Vim
set nocompatible

" -- Display
set title                 " Update the title of your window or your terminal
set number                " Display line numbers
set ruler                 " Display cursor position
set wrap                  " Wrap lines when they are too long

set scrolloff=3           " Display at least 3 lines around you cursor
                          " (for scrolling)

set guioptions=T          " Enable the toolbar

" -- Search
set ignorecase            " Ignore case when searching
set smartcase             " If there is an uppercase in your search term
                          " search case sensitive again
set incsearch             " Highlight search results when typing
set hlsearch              " Highlight search results

" -- Beep
set visualbell            " Prevent Vim from beeping
set noerrorbells          " Prevent Vim from beeping
```

For those who have done a copy/paste, you just have to save your newly created file. We want to put it in our home directory, so you have to save it as `~/.vimrc`. When using Mac OS X and Linux, `~` is the home directory of the current user. But be careful, when using Linux and Mac OS X the files starting with a `.` are hidden files. Don't be surprised when you don't `~/.vimrc` in your file explorer by default.

To save it with Vim, after pressing the `Esc` key to return to *Normal mode*, simply type `:w ~/.vimrc`. To save your next changes, type `:w` in *Normal mode*. To save and exit `:wq ~/.vimrc`. To exit `:q` and to exit without saving (force exit) `:q!`.

I have uploaded this configuration file directly on *Github*. You can download or copy/paste it directly from: https://vimebook.com/link/v2/en/firstconfig.

This is what *Vim* should look like *after your first configuration*.



```
" VIM Configuration - Vincent Jousse
" Cancel the compatibility with Vi. Essential if you want
" to enjoy the features of Vim
set nocompatible

" -- Display
set title                  " Update the title of your window or your terminal
set number                 " Display line numbers
set ruler                  " Display cursor position
set wrap                   " Wrap lines when they are too long

set scrolloff=3            " Display at least 3 lines around you cursor
                           " (for scrolling)

set guioptions=T           " Enable the toolbar

" -- Search
set ignorecase             " Ignore case when searching
set smartcase              " If there is an uppercase in your search term
                           " search case sensitive again
set incsearch              " Highlight search results when typing
set hlsearch               " Highlight search results

" -- Beep
set visualbell             " Prevent Vim from beeping
set noerrorbells           " Prevent Vim from beeping
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
".vimrc" 26L, 993B                                                   1,1           All
```

Fig. 2: *Vim* after your first configuration.

Notice the addition of line numbers on the left.

Well, it's a good start, but we now need more colors. Let's go!

3.4 And now, the color!

First, we need to enable syntax highlighting in the configuration file. Add these lines at the end of your `~/.vimrc` configuration file

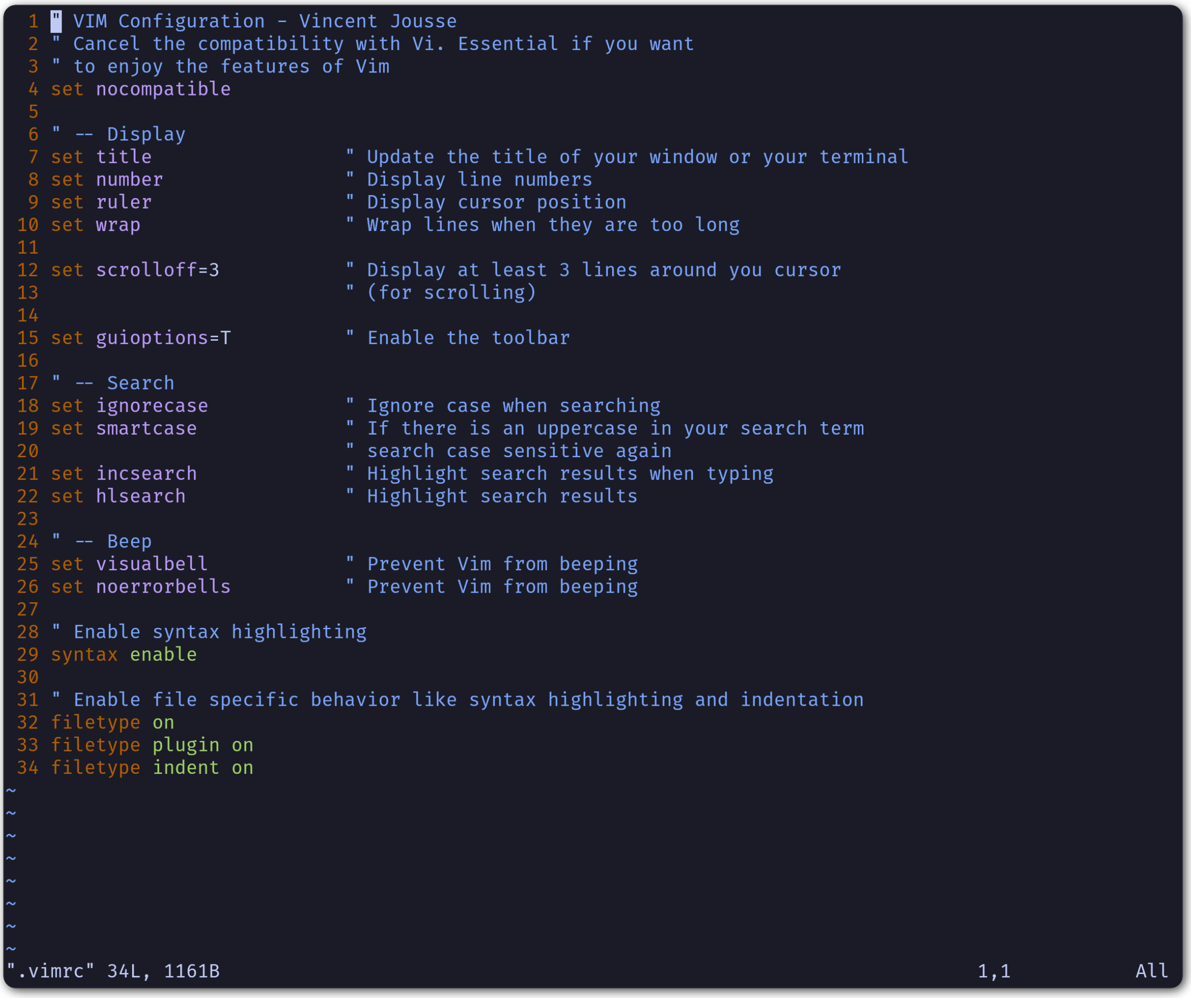
```
" Enable syntax highlighting
syntax enable

" Enable file specific behavior like syntax highlighting and indentation
filetype on
```

(continues on next page)

(continued from previous page)

```
filetype plugin on
filetype indent on
```

You should have a *Vim* looking like the picture below.



```
" VIM Configuration - Vincent Jousse
" Cancel the compatibility with Vi. Essential if you want
" to enjoy the features of Vim
set nocompatible

" -- Display
set title                 " Update the title of your window or your terminal
set number                " Display line numbers
set ruler                 " Display cursor position
set wrap                  " Wrap lines when they are too long

set scrolloff=3           " Display at least 3 lines around you cursor
                          " (for scrolling)

set guioptions=T          " Enable the toolbar

" -- Search
set ignorecase            " Ignore case when searching
set smartcase             " If there is an uppercase in your search term
                          " search case sensitive again
set incsearch             " Highlight search results when typing
set hlsearch              " Highlight search results

" -- Beep
set visualbell            " Prevent Vim from beeping
set noerrorbells          " Prevent Vim from beeping

" Enable syntax highlighting
syntax enable

" Enable file specific behavior like syntax highlighting and indentation
filetype on
filetype plugin on
filetype indent on
~
~
~
~
~
~
~
~
".vimrc" 34L, 1161B                                                                   1,1           All
```

Fig. 3: Default syntax highlighting.

For the time being, the easiest way to test the modifications you made to your `~/.vimrc` file is to restart *Vim*. If you want to use *Vim* like a boss right now, you can type in normal mode `:so ~/.vimrc` or `:so $MYVIMRC` . It will reload the configuration without the need to restart *Vim*. `:so` being a shortcut for `:source` .

This is a good first step, but now it's time to start using a theme.

Themes will allow you to have a nicer *Vim* than the default one. A theme will change the background color of *Vim* and the colors used for the syntax highlighting. As I said earlier, we will use the *Solarized* theme[1] https://ethanschoonover.com/solarized (with dark or light background, it will be up to you).

To install it, you will first need to create a directory called *.vim* in the same directory than your `~/.vimrc` (that is to say, in your home directory). Note that when using Windows, the *.vim* directory is called *vimfiles*. Each time I'll be speaking of the *.vim* directory, it will be the *vimfiles* directory for people using Windows. In this *.vim* directory, create a sub directory named *colors*. Then, download the *Solarized* theme file https://raw.githubusercontent.com/ericbn/vim-solarized/master/colors/solarized.

[1] Please note that we'll be using a modernized version of *Solarized* for vim and not the original version available on the author's site. This more recent version will enable it to run correctly on modern terminals. We'll install it from this fork https://github.com/ericbn/vim-solarized.

vim (it's the same file for the light and the dark version) and copy it in your *vim/colors/* directory. Under Linux you can do all this with the following commands:

```
mkdir -p ~/.vim/colors
wget -P ~/.vim/colors https://raw.githubusercontent.com/ericbn/vim-solarized/master/colors/solarized.vim
```

Your *.vim* directory should look like this:

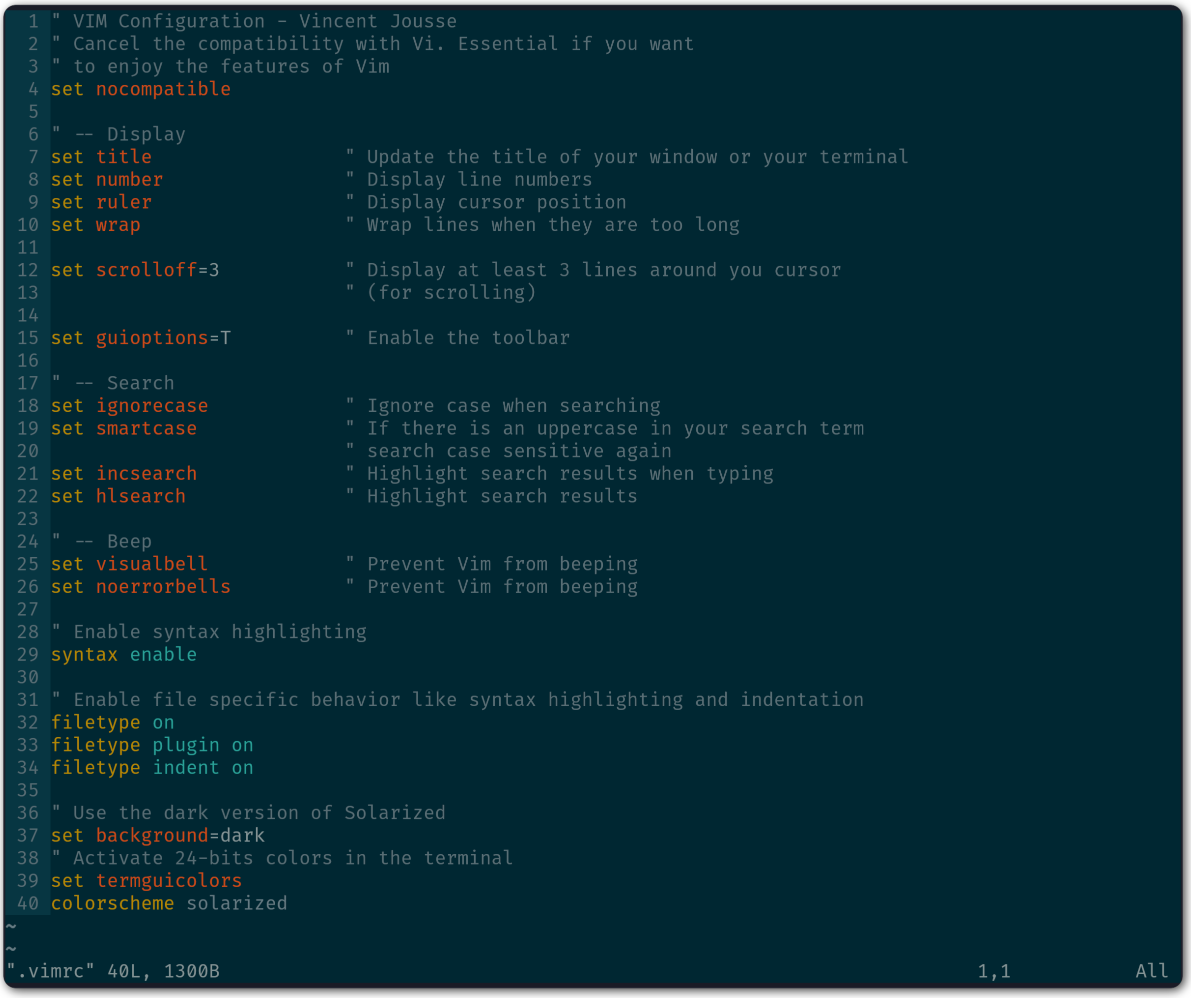
```
.vim
└── colors
    └── solarized.vim
```

Then enable the Solarized theme in your `~/.vimrc` like shown in the code below.

```
" Use the dark version of Solarized
set background=dark
" Activate 24-bits colors in the terminal
set termguicolors
colorscheme solarized
```

To test the light theme, you just have to change *dark* with *light* (for the *background* property).

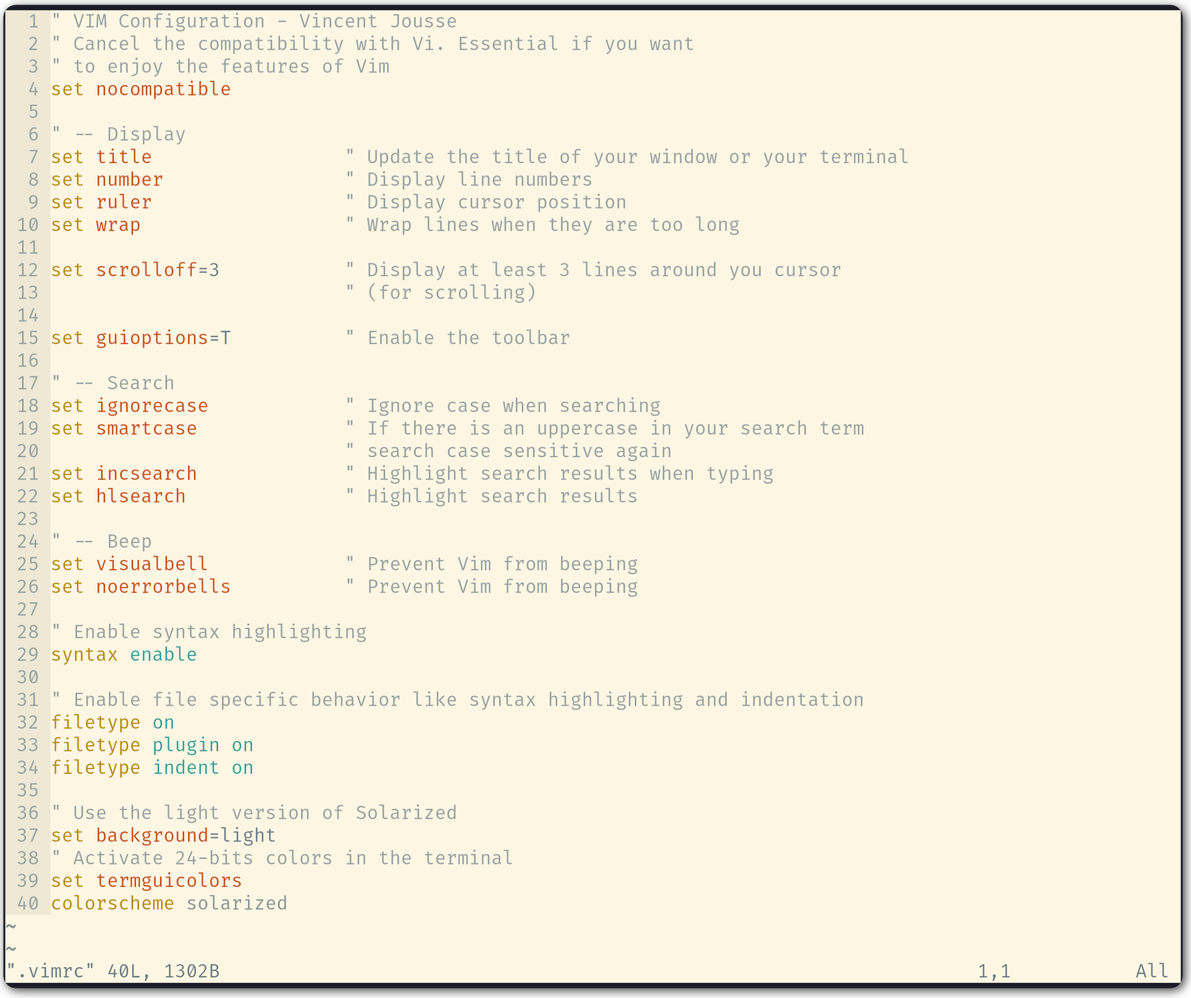
Here is a preview of the two versions (personally, I prefer the dark one).



```
" VIM Configuration - Vincent Jousse
" Cancel the compatibility with Vi. Essential if you want
" to enjoy the features of Vim
set nocompatible

" -- Display
set title                  " Update the title of your window or your terminal
set number                 " Display line numbers
set ruler                  " Display cursor position
set wrap                   " Wrap lines when they are too long

set scrolloff=3            " Display at least 3 lines around you cursor
                           " (for scrolling)

set guioptions=T           " Enable the toolbar

" -- Search
set ignorecase             " Ignore case when searching
set smartcase              " If there is an uppercase in your search term
                           " search case sensitive again
set incsearch              " Highlight search results when typing
set hlsearch               " Highlight search results

" -- Beep
set visualbell             " Prevent Vim from beeping
set noerrorbells           " Prevent Vim from beeping

" Enable syntax highlighting
syntax enable

" Enable file specific behavior like syntax highlighting and indentation
filetype on
filetype plugin on
filetype indent on

" Use the dark version of Solarized
set background=dark
" Activate 24-bits colors in the terminal
set termguicolors
colorscheme solarized
~
~
".vimrc" 40L, 1300B                                                      1,1           All
```

Fig. 4: The dark *Solarized* theme.

A bonus (if you don't use *Vim* directly in your terminal) would be to choose a font that suits your needs a little bit better. This is of course optional, but I suppose that some of you may wish to do this.



```
" VIM Configuration - Vincent Jousse
" Cancel the compatibility with Vi. Essential if you want
" to enjoy the features of Vim
set nocompatible

" -- Display
set title                 " Update the title of your window or your terminal
set number                " Display line numbers
set ruler                 " Display cursor position
set wrap                  " Wrap lines when they are too long

set scrolloff=3           " Display at least 3 lines around you cursor
                          " (for scrolling)

set guioptions=T          " Enable the toolbar

" -- Search
set ignorecase            " Ignore case when searching
set smartcase             " If there is an uppercase in your search term
                          " search case sensitive again
set incsearch             " Highlight search results when typing
set hlsearch              " Highlight search results

" -- Beep
set visualbell            " Prevent Vim from beeping
set noerrorbells          " Prevent Vim from beeping

" Enable syntax highlighting
syntax enable

" Enable file specific behavior like syntax highlighting and indentation
filetype on
filetype plugin on
filetype indent on

" Use the light version of Solarized
set background=light
" Activate 24-bits colors in the terminal
set termguicolors
colorscheme solarized
~
~
".vimrc" 40L, 1302B                                                          1,1           All
```

Fig. 5: The light *Solarized* theme.

If you are using Mac OS X, I recommend the *Monaco* font that is quite friendly. Add the following lines to your `~/.vimrc` to use it:

```
set guifont=Monaco:h13
set antialias
```

You can of course change *h13* with *h12* if you want a smaller font (or with *h14* if you want a bigger one).

Under Linux I am using the *DejaVu Sans Mono* font:

```
set guifont=DejaVu\ Sans\ Mono\ 10
set antialias
```

You can of course change the font size as you wish. To have the list of all the available fonts for your system type `:set guifont:*` in normal mode.

You will find the full version of the configuration file for this chapter online https://vimebook.com/link/v2/en/syntaxhlconfig. I will not spend more time talking about the fonts as it's dependant of your operating system and not of *Vim*.

3.5 Our first plugin: the file explorer

Here we are, we have a nice *Vim* that we can actually use with pretty colors. Now we need to be able to open files, which could come in handy! This will be a good opportunity to install our first plugin. We're going to do this in two steps: first, install a plugin manager to prevent your plugins from getting too messy, then install the appropriate plugin to explore a file directory.

3.5.1 Plugin manager: vim-plug

vim-plug is typically the kind of plugin that you discover after having already configured your *Vim*. Then you ask yourself, "*Why didn't I start this way?*". Fortunately, I have a good news for you: we will be starting the right way.

First of all, let's start with a little explanation about how to install plugins using *Vim*. Plugins are installed by copying files (most of the time with the **.vim* extension) in subdirectories of your *~/.vim* directory. By the way, we've already created a subdirectory called *colors* that contains our first coloration plugin using the Solarize theme.

The main problem with this approach is that the plugins are not isolated. So you will have to copy files from different plugins in the same directory and you will soon not be able to know from what plugin a file is coming from. As a result, when you will want to remove or update a plugin, it will be a nightmare to know where the files are located.

That's why *vim-plug* is especially useful, it will allow each plugin to be located in a separate directory. Here is an example of a *~/.vim* directory before and after the usage of *vim-plug*:

Listing 1: .vim before *vim-plug*

```
.vim-
├── autoload
│   └── phpcomplete.vim
├── colors
│   └── solarized.vim
└── syntax
    ├── php.vim
    └── sql.vim
```