Dev problem that don’t help progress

There are a lot of debates between devs. Some of them are useful because after some time some clear winner emerge. But some are just a matter of either personal preferences, or even worse, won’t change the outcome.

Why such debate continu to live years after years is just a matter of friction. Because my personal choices influence yours. For example, if I chose to use some editor code style by using spaces and you prefer to use tabs. We have a problem, and as each one of us want to keeps its habits, we might try to rationalize our choices. While really it is just a matter of personal preference. And so, in the end we should decide which one win.

So if you don’t want to lose your time by searching to optimize your life here are the conclusions before the debate.

Trivial Debates

Tools & Habits

vim vs emacs vs any editor

matter of personal preferences, I switched to vim keybinding mostly to prevent hand problems, and text editing might be slighly faster at the cost of a long training

font choice

1. Edit code

   Simply chose a monospaced font that make a clear distinction between:

   - ‘0‘ and ‘O‘
   - ‘1‘ and ‘l‘
   - “‘ and ‘’‘
   - ‘”‘ and ‘”‘
   - ‘1‘ and ‘i‘
   - ‘8‘, ‘B‘, ‘6‘ and ‘0‘
   - ‘2‘ and ‘Z‘
   - ‘5‘ and ‘S‘

    - ‘|‘ and ‘¡
    - ‘()‘ and ‘{}‘
    - ‘:‘, ‘;‘ and ‘i‘
    - ‘.‘ and ‘,‘
2. Website design

   If you’re not a designer don’t over think about it. Just chose one preferred Sans serif and Serif font.

color scheme choice

You should really use a low contrast colortheme if you want to minimize headhache and there are good chances you’ll end up preferring dark themes.

tabs vs spaces

Spaces appear to win slightly because the file size is not really important and most people don’t care.

Smart tabs have still some issues with alignment.

OS choice for working

Matter of personal preference

Typed (static) vs Unityped (dynamic) programming language

I’ve got a long answer here, but if you are a proponent to unityped programming (dynamic typed programming) then you might not know language with great typing system.

If you are a proponent to static typing programming then know you can live using unityped programming.

The long answer being. Types are another abstraction. So as all abstraction, it has some benefits and some costs. I tend to believe that once you have finished your Proof of Concept Prototype, Types provide a lot more benefits than drawbacks. You can think about them as free unit testing. In fact with a complex enough type system you can think about them as an infinite number of unit tests for free. But just know that event with advanced typing system doesn’t prevent you to write tests. But the opposite is also false, you can’t simulate easily a typing system with only tests, even generative testing.

My Choices

editor: spacemacs (best of vim and emacs)

font choice: Menlo (on OS X, Hack on other OS)

color scheme: solarized dark (each time I try to change I came back to it)

tabs vs spaces: spaces (no configuration pb, file size doesn't matter today)

Mac OS X: best for working, better focus, minimal configuration, setup time (would love a *Nix env)

configurations/dotfiles: yadm

CVS: git with github (it's a social network)

typed vs untyped: typed help think right, but untyped is not <u>that</u> bad.

Todo list, timers, note taking, thought orgnaiser: 'org-mode'

Solved but not known enough

REST (not RESTful)

- Why REST: least surprise

Encoding

- Use UTF-8 Everywhere http://utf8everywhere.org

Readability:

- lenght of line (33em)