Using Vim

- Jerry Peng

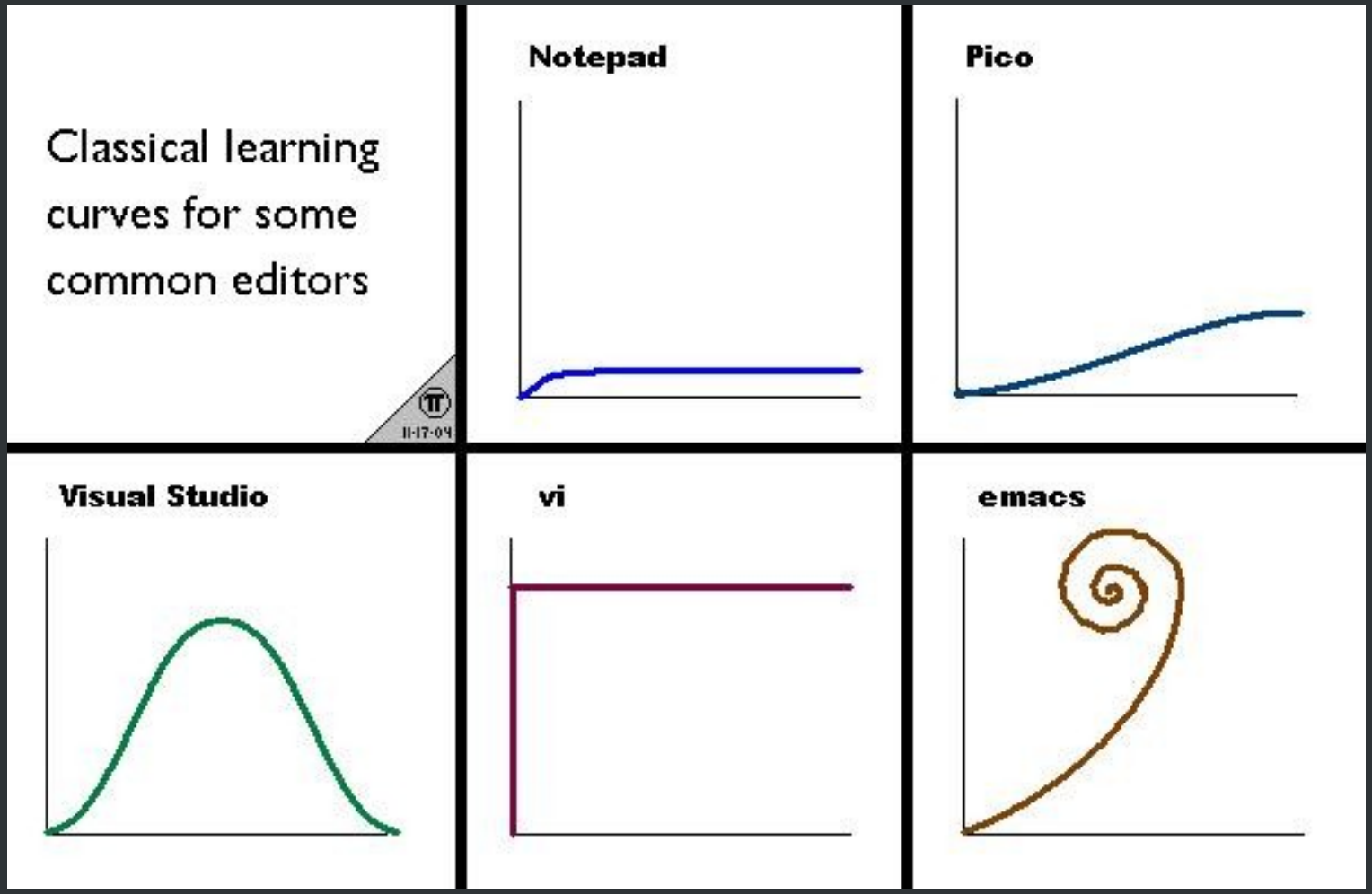
About Me

- Jerry Peng，彭睿
- 沃克斯科技 Java 开发者
- Linux, C, Python 爱好者
- Ubuntu, Arch, Gentoo

Why Vim?

- 模式编辑器
  - 缩短命令的按键序列
  - 普通、可视模式基本无须 Ctrl， Alt 等辅助键
- 脚本、正则表达式、多语言、 CLI 、 GUI……
- 不会 Vim 的 geek 都是伪 geek...
  - 无意挑起同 Emacs 党的争论
  - 适合自己的才是最好的

Before We Start...



Before We Start...

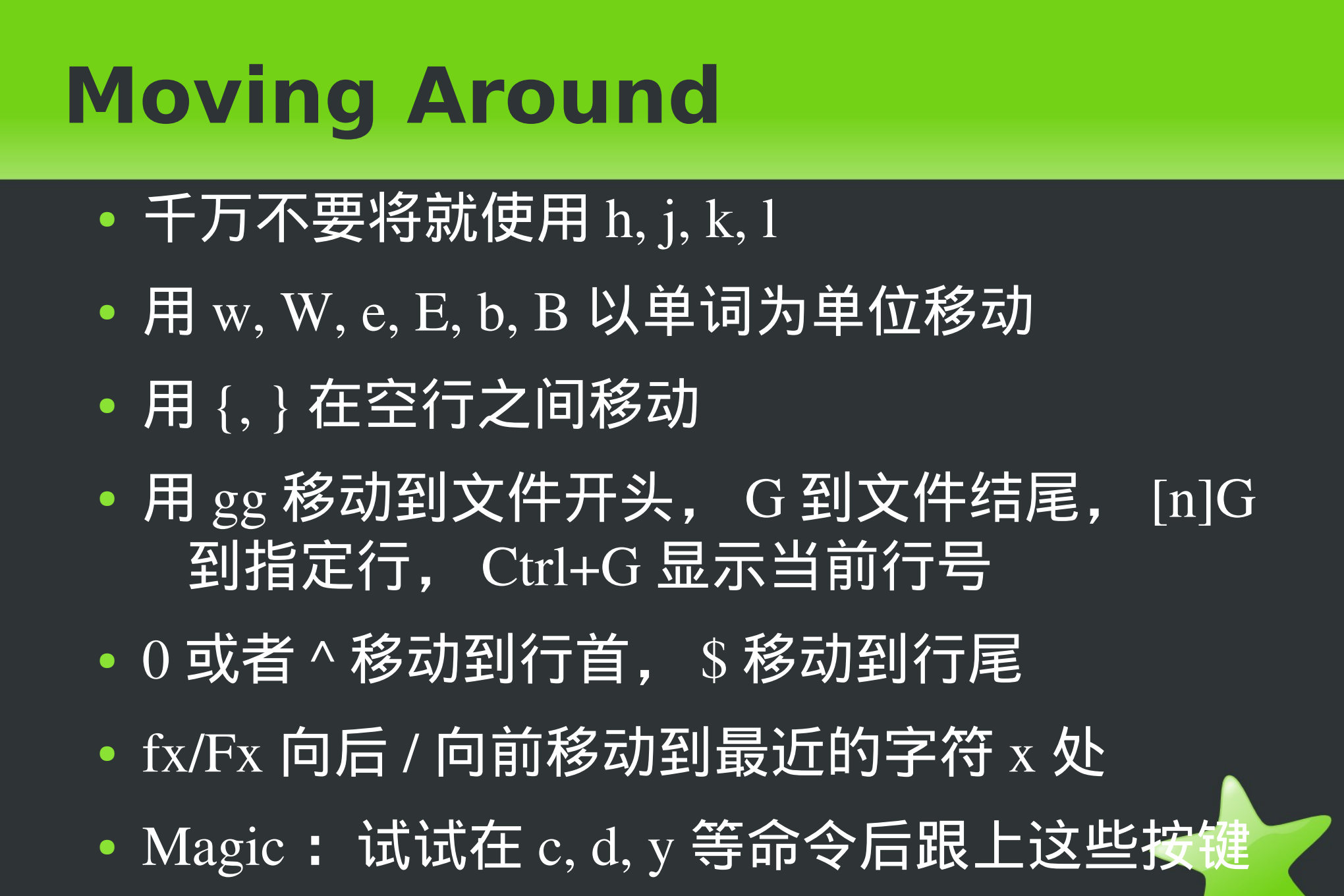
- Vim 是个给力的编辑器，但 ...
  - 陡峭的学习曲线，要愿意付出学习代价
  - 不要将就使用基本的功能而不深入学习
  - 每天都使用它，探索它！
- 善用 Vim 的帮助
  - :help 或者 F1 打开帮助
  - 十分详细，建议仔细阅读

Common Vim Tricks

- Moving Around
- Buffers
- Editing in Insert mode
- Copy & Paste
- Text Objects
- Searching and Replacing
- Using Your Own .vimrc

Moving Around

- 千万不要将就使用 h, j, k, l
- 用 w, W, e, E, b, B 以单词为单位移动
- 用 {, } 在空行之间移动
- 用 gg 移动到文件开头， G 到文件结尾， [n]G 到指定行， Ctrl+G 显示当前行号
- 0 或者 ^ 移动到行首， $ 移动到行尾
- fx/Fx 向后 / 向前移动到最近的字符 x 处
- Magic： 试试在 c, d, y 等命令后跟上这些按键



Moving Around

- Ctrl+F/Ctrl+B 来向下 / 向上翻页
- Ctrl+D/Ctrl+U 来向下 / 向上滚动半页
- Ctrl+O 后退
- :help navigation

Buffers

- 同时编辑多个文件
  - :e 打开文件到新 buffer 中
  - :bd 关闭当前 buffer
  - 不可见，所以不直观
  - 通过插件显示 buffer 列表
    - minibufexpl

Buffers

- 常用操作
  - 用 :bn/:bp 移动到下一个 / 上一个 buffer
  - :buffers 查看已打开的 buffer 列表
  - :b n 移动到第 n 个 buffer
  - 考虑映射成别的按键
- :help buffers

Editing in Insert Mode

- 强力必杀技 Ctrl+N
  - 补全当前 buffer 中已经出现过的单词
  - 任意文件类型都适用
  - 如果是程序源代码，还包含 tags 等等
  - 在补全列表中用 Ctrl+N/Ctrl+P 向后 / 向前移动
- 粘贴： Ctrl+R
  - 稍候详述
- :help Insert

Copy & Paste

- 用 y 来复制
  - 普通模式通过 text-objects 按键来指定复制范围
  - vistual 模式复制已选中的文本
- 用 p/P 来粘贴
  - 普通模式用 p/P 粘贴到光标后 / 前
  - Visual 模式会替代已选中的文本
- 插入模式用 Ctrl+R 加寄存器名来粘贴
  - 粘贴系统剪贴板中的内容是 Ctrl+R +

Copy & Paste – Registers

- 用 " 跟上寄存器名来指定操作的寄存器
  - c, d, y, p 等命令都可用
- 0-9 是堆栈结构
  - 让人迷惑，个人通常用命名寄存器
  - Yankring 插件
- 被复制、替换或者删除的文本都会存入寄存器
- 系统剪贴板对应的是 "+ 寄存器
- :help registers

Text Objects

- 帮助理解 Vim 命令按键的规律
- 一旦理解并熟悉就快要步入 " 运指如飞 " 的境界了 ...
  - 只是个人经验
- 有规律可循，容易理解和记忆
- :help text-objects

What's Text Objects?

- 简短的按键序列
- 代表 buffer 中的文本对象
- 用途：
  - 普通模式：放在命令之后来指定操作对象
  - 可视模式：选定文本
- 以 a 和 i 开头：
  - a： a(n) xxx object，包含空格等
  - i： inner xxx object，不包含空格等

Text Objects

- Text Objects 实例
  - aw/iw： 单词（前者包含单词之后的空格）
  - a{/i{ ： 大括号之间的文本（前者包含大括号）
    - 也可以用 aB/iB
  - a"/i" ： 一对双引号（前者包含引号）
  - at/it ： 一对 XML tag 之间的内容（前者包含 tag ）
- A lot more
  - :help text-objects

Text Objects

- 普通模式使用实例：
  - yiw： 复制光标所在单词
  - ciB： 修改光标所在处前后的 {} 之间的内容
    - 对 C/C++, Java, JS 等超级有用 ...
  - dab： 删除光标所在处前后的 () 之间的文本，包括 ()
  - yat： 复制光标所在的 XML 块，包含标签
- 发挥想象力吧

Vim Command Pattern

- 是时候总结 Vim 命令的按键模式了
  - 仅针对普通模式
- [register][repeats]cmd[cmd/txt_obj/nav_key]
  - 重复按命令键两次是针对当前行，如 dd 删除一行， cc 更改一行， yy 复制一行
- "m3dd 从光标开始删除 3 行并存入寄存器 m
- 4dat 从光标处开始删除 4 级 XML 标签

Vim Command Pattern

- 绝大多数操作都能以这套模式完成
- 多加使用，很快就能熟悉
  - 千万不要用 h, j, k, l 或上下左右移动到一个位置，按 i 进入 Insert 模式，用 Backspace 删除一堆字符然后再输入！
  - 要以尽可能减少按键次数为荣！
  - 要以单调的重复按键为耻！

- VimGolf - real Vim ninjas count every keystroke!
  - 见附录链接

Text Objects

- 可视模式的使用
  - 选择光标所在处的指定文本对象
  - 如果是嵌套的对象，重复按键能选择更大范围
    - 参考实例
- 发挥想象力吧

Searching & Replacing

- 用 / 和 ? 开始输入要搜索的字符串
  - / 向后，? 向前
- 用 n 和 N 来搜索下一个
  - n 向后，N 向前
- 超级实用：用 * 和 # 搜索光标处的单词
  - * 向后，# 向前
- 硬功夫：正则表达式

Searching & Replacing

- 打开增量搜索
  - :set incsearch
- 高亮所有搜索结果
  - :set hlsearch
- 相关配置写到自己的 .vimrc 中

Searching & Replacing

- 简单替换
  - 普通模式 r 替换光标处的字符
  - 普通模式 R 进入替换模式，输入的字符会替换已有内容
    - 和 Windows 下 Insert 键功能类似

Searching & Replacing

- 复杂替换
  - :{range}s/regex/replacement/{flags}
  - 常用： :%s/regex/replacement/g
  - :help :s
- 正则表达式让其威力大增
  - 还是硬功夫

Using Your Own .vimrc

- 强烈推荐用 github 存放个人配置文件
  - 连同插件、配色等一起放到 github 上
- 学习 .vimrc 配置语言
- 考虑从别人的 .vimrc 开始
  - 强烈推荐 amix.dk 的 vim 配置（见附录）
- 用 pathogen 来管理插件
  - 见实例

Vim for Programmers

- Code Completion
- Quickfix
- Ctags & Cscope
- Fuzzyfinder
- SnipMate
- Zencoding

Vim for Programmers

- 简单介绍，实际演示
- 大部分通过插件实现
  - 寻找适合自己口味的插件
  - 关注 vim.org ，寻找需要的插件

Code Completion

- 都在插入模式触发
- 配置正确的情况下 Ctrl + N 能完成大部分补全
- Tag 补全
  - 用 Ctrl + X Ctrx +] 触发
- Omni 补全
  - 用 Ctrl + X Ctrl+O 触发
  - Vim 仅仅提供框架来调用自定义函数
  - 实际功能需要插件提供，通常更强大

quickfix

- 加速 " 编辑 -> 编译 / 运行 -> 改错 " 的过程
  - 显示错误列表
  - 跳转到上一个 / 下一个错误
- :make 命令
  - 默认调用 make
  - 用 :set makeprg=xxx 来替换成别的命令
- 调用 makeprg 后 vim 会记录一个错误列表

quickfix

- :cc 显示当前错误
- :cw 打开 quickfix 窗口显示所有错误
- :cn 显示下一个错误并跳转到代码相应位置
- :cp 显示上一个错误并跳转到相应位置
- :help quickfix

ctags & cscope

- 通过 ctags 程序生成 tag 数据文件
- 查找、补全 tag
- Ctrl+] 跳转到光标处的 tag， Ctrl+O 跳转回去
- 支持多种语言
- fuzzyfinder, taglist 等插件
- Cscope 更加强大，但仅支持 C/C++
- :help tags

fuzzyfinder

- 模糊查找
- 即时显示匹配结果
  - Ctrl + N 移动到下一个结果
  - Ctrl + P 移动到上一个结果
  - Enter 打开
- 查找文件 , tag, buffer, quickfix...
- 禁用不常用功能，将常用的映射到快捷键上

snipMate

- TextMate 的必杀 feature
  - TextMate 是 Mac OS X 下的强力编辑器
  - 大大提升代码编辑效率
- 自定义代码片段和对应的触发字符串
- 输入触发器串，按 Tab 补全成完整片段
- 按 Tab 在插入点之间切换
- 言语难以表达，见示例

zencoding

- 为 HTML/XML 而生
- 一种微型语言
  - It's like magic...
- 多种 IDE/ 编辑器下都有实现
- 言语难以表达，见示例

What's Next?

- 每天都使用它，探索它！
- 认清 Vim 学习曲线陡峭的事实，坚持学习
  - 用好 Vim 帮助，其内容十分详细
- 与人交流
  - Vim-CN Google Group
- 工欲善其事，必先利其器
  - 整理出自己的一套 Vim 配置和插件，并管理好

Q & A

References

- Vim官网
- vi/vim使用进阶
- Vim官网插件目录
- amix.dk的终极Vim配置
- Why, oh WHY, do those #?@! nutheads use vi?
- VimGolf - real Vim ninjas count every keystroke!
- 提到的各色插件，请 Google 之