vim 解决问题的方式

技巧 1 结识 .命令

- .命令可以重复上次的修改
  - >g 会增加从当前行到文档末尾处的缩进层级
  - 每次进入插入模式也会形成一次修改。从进入插入模式的那一刻起，直到返回普通模式为止，vim 会记录每一个按键操作。
  - .命令是一个微型的宏

技巧 2 不要自我重复

- a 命令在当前光标之后添加内容
- a 命令在当前行的结尾添加内容

技巧 3 以退为进

- f{char}命令让 vim 查找下一处指定字符出现的位置，如果找到了，就直接把光标移动到那里
- ;命令会重复上次 f 命令所查找的字符
- s 命令先删除光标下的字符，然后进入插入模式

技巧 4 执行、重复、回退

- @:可以重复任意 ex 命令
- &重复上次的:substitute 命令

目的	操作	重复	回退
做出一个修改	{edit}	.	u
在行内查找下一指定字符	f{char}/t{char}	;	,
在行内查找上一指定字符	f{char}/t{char}	;	,
在文档中查找下一处匹配项	/pattern<cr>	n	n
在文档中查找上一处匹配项	?pattern<cr>	n	n
执行替换	:s/target/replacement	&	u
执行一系列修改	qx{changes}q	@x	u

技巧五 查找并手动替换

- *命令可以查找当前光标下的单词

技巧六 结识.范式

普通模式

技巧 7 停顿时请移开画笔

技巧 8 把撤销单元切成块

技巧 9 构造可重复的修改

- 如果在插入模式中使用了<up>、<down>、<left>、<right>这些光标键，将会产生一个新的撤销块。会对.命令的操作产生影响
- 使用更为精准的 aw 文本对象

技巧 10 用次数做简单的算术运算

- <c-a>和<c-x>命令对数字执行加和减操作。在不带数字执行时，它们会逐个加减，但如果带一个次数前缀，那么就可以用它们加减任意整数。
  - 如果光标不在数字上，会把当前光标之上或之后的数值加上[count]
  - vim 会把 0 开头的数字解释为八进制值，可以加入 set nrformats=将所有数字设为十进制

技巧 11 能够重复，就别用次数

技巧 12 双剑合璧，天下无敌

命令	用途
c	修改
d	删除
y	复制到寄存器
g~	反转大小写 （可视化模式下）
gu	转为小写
gu	转为大写
>	增加缩进
<	减小缩进
=	自动缩进
!	使用外部程序过滤{motion}所跨越的行

- >>缩进当前行
- guap 把整段文字转换为大写

- vim 的语法只有一条额外的规则，即当一个操作符命令被连续调用两次时，它会作用于当前行
  - gUgU 或 gUU 可以作用于当前行

插入模式

技巧 13 在插入模式中可即时更正错误

- <C-h>删除前一个字符（同退格键）
- <C-w>删除前一个单词
- <C-u>删至行首

技巧 14 返回普通模式

按键操作	用途
Esc>	切换普通模式
<C-[	切换到普通模式
<C-o>	切换到插入-普通模式

- 插入-普通模式 在此模式中，可以先执行一个普通模式，执行完后，马上回到插入模式
- zz 让当前行显示在窗口正中；<C-o>zz 在插入-普通模式中触发这条命名让正中显示后直接 j 回到插入模式

技巧 15 不离开插入模式，粘贴寄存器中的文本

- <C-r>{register}插入寄存器中的内容，但是当 textwidth 或 autoindent 选项被激活了的话，那么最终会出现不必要的换行或额外的缩进
- <C-r><C-p>{register}会更智能些，它会按照原意插入寄存器中的文本，并修正任何不必要的缩进

技巧 16 随时随地做运算

- 表达式寄存器允许我们做一些运算，并把运算结果直接插入到文档中
- 在插入模式中，输入 <C-r>=就可以访问这一寄存器。这条命令会在屏幕的下方显示一个提示符，我们可以在其后输入要执行的表达式。输入表达式后敲一下<CR>，Vim 就会把执行的结果插入到文档的当前位置了。

技巧 17 用字符编码插入非常用字符

- 在插入模式下输入<C-v>{code}其中{code}是要插入字符的编码

- 把光标移动到字符上面并按 ga 命令，下方就会列出编码信息（十进制和十六进制）

按键操作	用途
<C-v>{123}	以十进制字符编码插入字符
<C-v>u{1234}	以十六进制字符 c 编码插入字符
<C-v>{nondigit}	按原义插入非数字字符
<C-j>{char1}{char2}	插入以二合字母{char1}{char2}表示的字符

技巧 18 用二合字母插入非常用字符

技巧 19 用替换模式替换已有文本

- 虚拟替换模式 gR

可视化模式

技巧 20 深入理解可视化模式

- viw
- <C-g>可以在可视化模式及选择模式间切换，在选择模式中输入任意可见字符，此字符会替换所选内容并切换到插入模式

技巧 21 选择高亮选区

命令	用途
v	激»面向字符的可视模式
V	激»面向行的可视模式
<C-v>	激»面向列块的可视模式
gv	重选上次的高亮选区
o	切换高亮选区的活动端

技巧 22 重复执行面向行的可视命令

- 当使用 . 命令重复对高亮选区所做的修改时，此修改会重复作用于相同范围的文本(验证失败)

技巧 23 只要可能，最好用操作符命令，而不是可视命令

- 可视化模式下可以使用 U 命令来大写所选字符
- vit 选择标签内文本
- 当一条可视模式命令被重复执行时，它会影响相同数量的文本

- 在这个模式下 . 命令有时会有一些异常的表现。

技巧 24 面向列块的可视模式编辑表格数据

技巧 25 修改列文本

- 列块可视化模式中，删除操作会影响所有被选中的，但插入操作只影响顶行，而且 `a i o` 等命令无效
    - 因为在可视化模式下，`a i` 被当作一个文本对象的组成部分，而 `o` 有其他的作用

技巧 26 在长短不一的高亮块后添加文本

- `I A` 命令

命令行模式

技巧 27 结识 Vim 的命令行模式

- 有些命令在插入模式和命令行模式中可以通用。例如，可以用 `<C-w>` 和 `<C-u>` 分别删除至上个单词的开头及行首，也可以用 `<C-v>` 或 `<C-k>` 来插入键盘上找不到的字符，还可以用 `<C-r>{register}` 命令把任意寄存器的内容插入到命令行

技巧 28 在一行或多个连续行上执行命令

- 用行号作为地址
    - 如果只输入一条只包含数字的 `Ex` 命令，那么 Vim 就把这个数字解析成一个地址，并把光标移动到该数字所指定的行上
        - `:1` 跳转到第一行，`:$`跳转到最后一行
    - `:p` 或者`:print` 在命令行输出当前行内容
        - `:3p` 光标移动到第三行，然后显示该行内容
        - `:3d` 光标移动到第三行，并删除此行
- 用地址指定一个范围
    - `:2,5p` 打印从第二行到第五行之间的每一行内容（包含第二和第五行），运行完后，光标停留在第五行
    - `.`代表当前行
    - `%`代表文件中所有的行

- 用高亮选区指定范围
  - 可视化模式下按下:键，命令行上会预先填充一个范围 :'<,'>，然后输入一条 Ex 命令，使它作用在每个被选中的行上
  - '<是代表高亮选区首行的位置标记
  - '>是代表高亮选区的最后一行
- 用模式指定范围
  - :/<html>/,/\/html>/p 也符合 :{start},{end}
    - {start} 是 /

      /，而{end} 地址是 /</html>/
- 用偏移对地址进行修正
  - :{address}+n 如果 n 被省略，缺省值为 1
    - :.,.+3p 等于 当前行到下三行

技巧 29 使用 :t 和 :m 命令复制和移动

- :copy 及其 :t 可以把一行或者多行从文档的一部分复制到另一部分
  - :{range}t{address}
- :move 可以把一行或多行文档移到文档其他地方

技巧 30 在指定范围上执行普通模式命令

- :%normal A; 会在全文每行末尾添加分号

技巧 31 重复上次的 EX 命令

- . 命令可以重复上次的普通模式命令。然而，如果想重复上次的 Ex 命令的话，我们得使用 @: 才行。
- : 寄存器总是保存着最后执行的命令行命令

技巧 32 自动补全 Ex 命令

技巧 33 把当前单词插入命令行

- 在 Vim 的命令行下，<C-r><C-w>映射项会复制光标下的单词并把它插入到命令行中。
- *命令等效于输入 /\<<C-r><C-w>\><CR>

技巧 34 回溯历史命令

- <UP> <Down>

命令	动作
q/	打开查找命令历史的命令行窗口
q:	打开 Ex 命令历史的命令行窗口
<C-f>	从命令行模式切换到命令行窗口

技巧 35 运行 shell 命令

- !叹号前缀
- <C-z>挂起 Vim 所属的进程，并把控制权交还给 bash
- jobs 查看当前作业列表
- fg 命令唤醒一个被挂起的作业

管理多个文件

技巧 36 用缓冲区列表管理打开的文件

- :ls 列出所有被载入内存中的缓冲区的列表
- bnext
- % 符号指明哪个缓冲区在当前窗口中可见，而 # 符号则代表轮换文件。按<C-^>可以在当前文件和轮换文件间快速切换
- :bprev 和 :bnext 在列表中反向或正向移动，每次移动一项；而 :bfirst 和 :blast 则分别跳到列表的开头和结尾
- 用:buffer N 命令直接凭编号跳转到一个缓冲区
- :bufdo 命令允许我们在 :ls 列出的所有缓冲区上执行 Ex 命令
  - :argdo 更加实用
- 删除缓冲区，可以用 :bdelete 命令
  - :bdelete N1 N2 N3
  - :N,M bdelete

技巧 37 用参数列表将缓冲区分组

- 用 Glob 模式指定文件

- :args commnads Vim 会在 shell 中执行反撇号括起来的命令，然后将结果输出作为:args 的参数

技巧 38 管理缓冲区列表

- 修改但未保存的文件会有 + 号

技巧 39 将工作区切分成窗口

- 用 s 命令可以水平切分此窗口，使之成为两个高度相同的窗口；或者可以用 v 命令对其进行垂直切分，这样会产生两个宽度相同的窗口。
- w h j k l
- close 关闭活动窗口
- on 只保留活动窗口，关闭其他所有窗口

技巧 40 用标签页将窗口分组

- :lcd {path} 命令让我们可以设置当前窗口的本地工作目录
  - :lcd 只影响当前窗口，而非当前标签页
- 用 :windo lcd {path} 命令为所有这些窗口设置本地工作目录。

打开及保存文件

技巧 41 用 :edit 命令打开文件

- `:pwd` 打印工作目录
- `:edit %<Tab>` %符号代表缓冲区的完整文件路径
  - `:edit %:h<Tab>` :h 修饰符会去除文件名
- cnoremap %% getcmdtype( ) == ':' ? expand('%:h').'/' : '%%'
  - 当你在 Vim 的命令行提示符后输入 %% 时，它就会被自动展开为活动缓冲区所在目录的路径

技巧 42 使用 :find 打开文件

- 需要预设 path
  - set path+=路径
  - /** 匹配该目录下所有子目录

技巧 43 使用 netrw 管理文件系统

- `:edit .` 打开文件管理器，并显示当前工作目录

- `:Explore` 打开文件管理器，并显示活动缓冲区所在目录
- 除:Explore 外，netrw 还提供了:Sexplore 及 :Vexplore 命令，这两条命令分别在一个水平切分窗口及垂直切分窗口里打开文件管理器。

技巧 44 把文件保存到不存在的目录中

- `:!mkdir -p %:h` -p 参数 创建任何不存在的中间目录

技巧 45 以超级用户权限保存文件

- `:w !sudo tee % > /dev/null`
  - `:write !{cmd}` 命令会把缓冲区的内容作为标准输入传给指定的{cmd}

用动作命令在文档中移动

技巧 46 让手指保持在本位行

- h j k l

技巧 47 区分实际行与屏幕行

- gj gk 是按照屏幕行向下及向上移动
  - 当一行显示不下，会有多个屏幕行的时候，可以在屏幕行之间移动

命令	光标动作
j	向下移动一个实际行
gj	向下移动一个屏幕行
k	向上移动一个实际行
gk	向上移动一个屏幕行
0	移动到实际行的行首
g0	移动到屏幕行的行首
^	移动到实际行的第一个非空白字符
g^	移动到屏幕行的第一个空白字符
$	移动到实际行的行尾
g$	移动到屏幕行的行尾

技巧 48 基于单词的移动

命令	光标动作
w	正向移动到下一单词的开头
g	反向移动到当前单词/上一单词的开头

e　　正向移动到当前单词/下一单词的结尾

ge　　反向移动到上一单词的结尾

- 一个单词由字母、下划线、数字，或其他非空白字符的序列组成，单词间以空白字符为间隔
- 一个字串由空白字符序列组成，字串间以空白字符分隔

技巧 49 对字符进行查找

- `f{char}` 在光标位置与当前行尾政治家查找指定字符
  - Vim 会记录上次执行过的 f{char} 命令，随后用 ; 命令就可以重复该命令了
  - 用 , 命令就可以再跳回来

命令	用途
f{char}	正向移动到下一个{char}所在之处
F{char}	反向移动到上一个{char}所在之处
t{char}	正向移动到下一个{char}所在之处的前一个字符上
T{char}	反向移动到上一个{char}所在之处的前一个字符上

技巧 50 通过查找进行移动

- 查找一个开动作

技巧 51 用精确的文本对象选择区