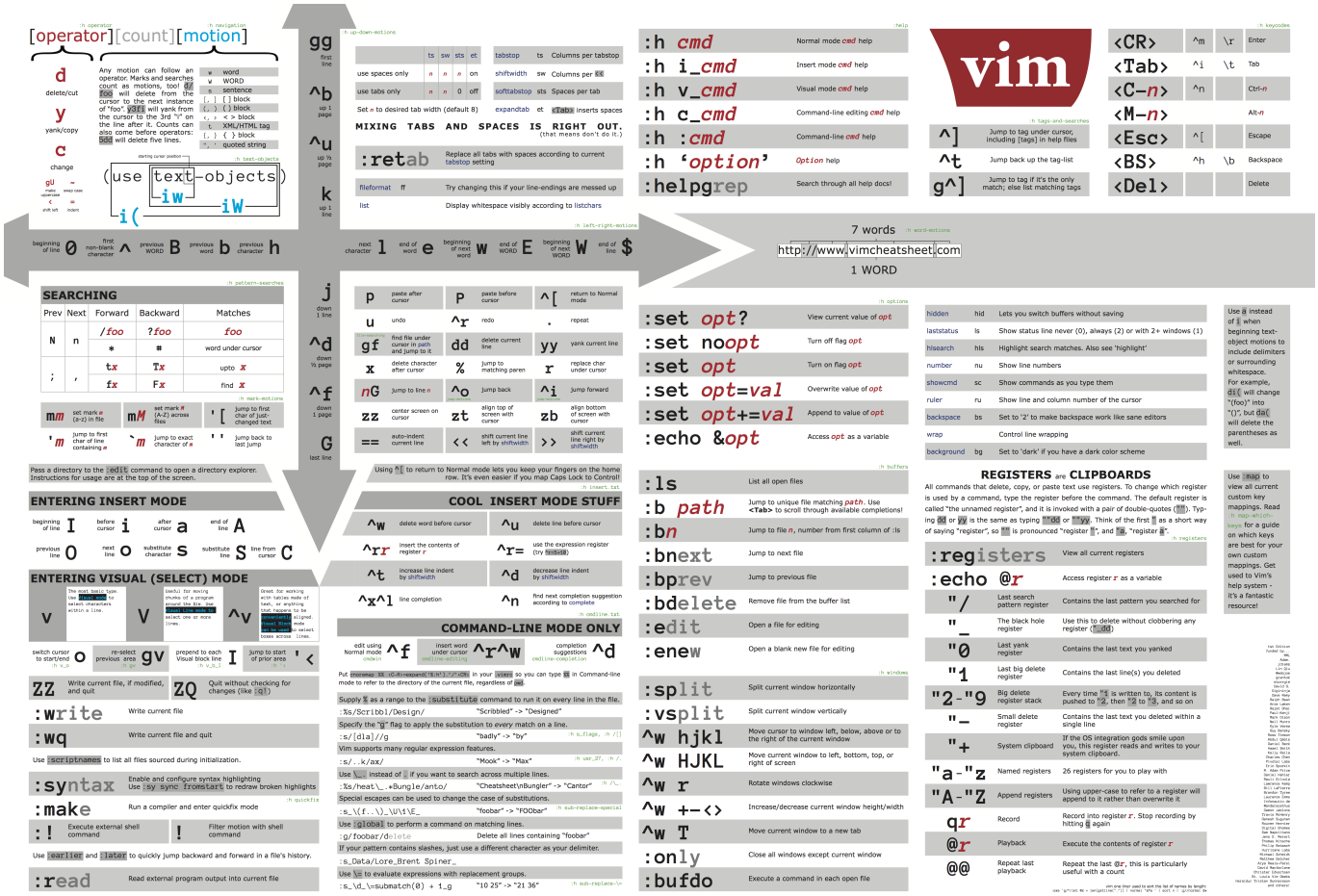
VIM Cheat Sheet




```
Command-----
도움말      :help {keyword}
다른 이름으로 저장      :saveas {file}
현재 영역 닫기      :close
커서가 위치한 단어에 대한 맨페이지 열기      K
새 버퍼에서 파일 편집      :e {file}
다음 버퍼로 이동      :bnext or :bn
이전 버퍼로 이동      :bprev or :bp
버퍼 삭제(파일 닫기)      :bd
열린 버퍼 모두 나열      :ls
새 버퍼에 파일 열고 상하로 창 분할      :sp {file}
서 버퍼에 파일 열고 좌우로 창 분할      :vsp {file}
상하로 창 분할      Ctrl + ws
창 전환      Ctrl + ww
창 닫기      Ctrl + wq
좌우로 창 분할      Ctrl +wv
오른쪽 창으로 이동(좌우 분할)      Ctrl + wh
왼쪽 창으로 이동(좌우 분할)      Ctrl + wl
아래 창으로 이동(상하 분할)      Ctrl + wj
```

위 창으로 이동(상하 분할) Ctrl + wk
새 탭에서 파일 열기 :tabnew or :tabnew {file}
힌재 분할 창을 새로운 탭으로 이동 Ctrl + wT
다음 탭으로 이동 gt or :tabnext or :tabn
이전 탭으로 이동 gT or :tabprev or :tabp
#번 탭으로 이동 {#}gt
현재 탭을 #번째로 이동(0부터 시작) :tabmove {#}
현재 탭과 그 안의 창들 닫기 :tabclose or :tabc
현재 탭 이외의 모든 탭 닫기 :tabonly or :tabo
모든 탭에서 commnad 실행하기 :tabdo {command} 예) 열린 모든 탭 닫기 :tabdo q)
저장하고, 나가지는 않기 :w
sudo로 현재 파일 저장 :w !sudo tee %
저장하고 나가기 :wq or :x or ZZ
나가기(저장하지 않은 변경 사항 있으면 실패) :q
나가기(저장하지 않은 변경 사항 버림) :q! or ZQ
열어 놓았던 모든 탭을 다 저장하고 나가기 :wqa

찾기와 바꾸기-----
패턴 찾기 /{pattern}
패턴 역방향 찾기 ?{pattern}
'마법'패턴 : 영문/숫자가 아닌 문자는 정규표현식 심볼로 해석(이스케이프 불필요) \v{pattern}
같은 방향으로 찾기 반복 n
반대 방향으로 찾기 반복 N
파일 내 모든 old를 new로 바꾸기 :%s/{old}/{new}/g
파일 내 모든 old를 new로 확인하며 바꾸기 :%s/{old}/{new}/gc
찾기 강조 표시 없애기 :noh
여러 파일에서 패턴 찾기 :vimgrep /pattern/ {file} 예) :vimgrep /foo/ **/*
다음 일치 항목으로 점프 :cn
이전 일치 항목으로 점프 :cp
일치 목록을 새창으로 열기 :copen

이동-----
왼쪽으로 이동
  한 칸 h
  처음 단어 시작으로 점프 w
  이전 단어 시작으로 점프 b
  이전 단어 시작으로 점프 B
아래로 이동
  한 칸 j
  화면 하단 L
가운데로 이동
  화면 가운데 M
위로 이동
  한 칸 k
  화면 상단 H
오른쪽으로 이동
  한 칸 l

```
  다음 단어 시작으로 점프 W
  다음 단어 끝어로 점프 E
현재 괄호와 맞는 짝의 괄호로 이동  %

행
  행의 처음으로 점프  0
  행의 시작 문자로 점프  ^
  행의 끝으로 점프  $
  행의 끝 문자로 점프     g_

문서
  문서 마지막 행으로 점프     G
     x번째 행으로 점프     {x}G
  커서 위치 기준, 오른쪽에서 가장 가까운 문자 x로 이동     f{x}
  커서 위치 기준, 오른쪽에서 가장 가까운 문자 x의 한 칸 뒤(왼쪽)으로 이동     t{x}
  커서 위치 기준 왼쪽에서 가장 가까운 문자 x로 이동     F{x}
  커서 위치 기준 왼쪽에서 가장 가까운 문자 x의 한 칸 앞(오른쪽)으로 이동     T{x}

가장 최근의 f, t, F, T 명령을 다시 실행  ;
가장 최근의 f, t, F, T 명령을 반대 방향으로 다시 실행  ,
다음 단락(또는 함수/블록)으로 이동  }
이전 단락(또는 함수/블록)으로 이동  {
커서가 있는 행을 중간으로 하도록 화면 이동     zz
커서를 움직이지 않고 화면을 한 줄 아래로 이동     Ctrl + e
커서를 움직이지 않고 화면을 한 줄 위로 이동     Ctrl + y
한 화면 위로     Ctrl + b
한 화면 아래로     Ctrl + f
    Ctrl + d
    Ctrl + u

** 커서 이동 명령 앞에 숫자를 붙이면 그 수만큼 반복. 예) 4j는 4행 아래로 이동



삽입-----
커서 앞에 삽입     i
행 시작에 삽입     I
커서 뒤에 추가     a
행 끝에 추가     A
현재 행 아래에 새 행 추가     o
현재 행 위에 새 행 추가     O
단어 끝에 추가     ea
삽입모드 종료     ESC

편집----
한 글자 바꾸기     r
현재 행과 다음 행을 연결     J
현재 행과 다음 행을 둘 사이에 공백 없이 연결     gJ
reflow paragraph     gwip
행 전체를 새로 쓰기     cc
```

change (replace) to the end of the line C
행 끝까지를 새로 쓰기 c$
change (replace) entire word cw
한 글자 삭제하고 삽입 모드 시작 s
행 삭제하고 텍스트 입력 (cc와 동일) S
두 문자 위치 바꾸기(잘라내기 붙여넣기 조합) xp
실행 취소 u
다시 실행 Ctrl + r
마지막 명령 반복 .
행 복사 yy
2줄 복사 2yy
커서 위치에서 다음 단어 시작까지 복사 yw
행 끝까지 복사 y$
커서 위치 뒤에 붙여넣기 p
커서 위치 앞에 붙여넣기 P
행 잘라내기 dd
2줄 잘라내기 2dd
커서 위치에서 다음 단어 시작까지 잘라내기 dw
행 끝까지 잘라내기 D
행 끝까지 잘라내기 d$
한 글자 잘라내기 x
Ctrl+c로 복사한 텍스트 붙여넣기 "+p (따옴표,+,p 세 개의 키)

비주얼 모드(텍스트 선택)-----
선택모드 시작. 텍스트 선택해서 명령 수행(가령 y로 복사) v
행 단위 선택 모드 시작 V
선택 영역의 반대쪽 끝으로 이동 o
블록 선택모드 시작 Ctrl + v
블록의 반대쪽 모서리로 이동 O
단어 선택 aw
()블럭 선택 ab
{}블럭 선택 aB
()블럭의 내부 선택 ib
{}블럭의 내부 선택 iB
선택 모드 종료 ESC

선택 모드 명령-----
텍스트를 오른쪽으로 이동 >
텍스트를 왼쪽으로 이동 <
선택한 텍스트 복사 y
선택한 텍스트 삭제 d
대소문자 반전 ~

레지스터-----
레지스터 내용을 표시 :reg
레지스터 x로 복사 "{x}y
레지스터 x의 내용물 붙여넣기 "{x}p

** 레지스터들은 ./viminfo에 저장되며 다음 번 vim 재시작 때 다시 읽어들임.
** 0번 레지스터에는 항상 최근 복사 명령의 값이 들어있음.

표시-----
표시 항목 표시 :marks
현재 위치를 a로 설정 m{a}
표시 a의 위치로 점프 '{a}
표시 a의 위치까지 복사 y'{a}

매크로-----
매크로 a 기록 시작 q{a}
매크로 기록 중지 q
매크로 a 실행 @{a}
마지막 실행한 매크로 재실행 @@

VIM TUTOR

커서 움직이기; hjkl
끝내기(저장하지 않음); :q!
한 글자 지우기; x
텍스트 입력; i
커서가 위치한 줄의 맨 끝에 텍스트 추가; A
저장하고 종료; :wq

단어 삭제; dw
줄의 끝까지 삭제; d$
dw(delete word with space), de(delete word without space), d$(delete till end of line)
두 단어 뒤로 이동(앞글자); 2w
세 단어 뒤로 이동(끝글자); 3e
문장의 시작으로 이동; 0
두 단어 삭제; d2w [횟수],명령,대상 | 명령,[횟수],대상
현재 줄 전체 삭제, 2줄 전체 삭제; dd, 2dd
마지막 명령 취소, 한 줄에서 수정한 것을 모두 취소, 취소 한 것 다시 실행; u, U, Ctrl+R

put; p
replace; r
한 단어 전체 change; ce
[횟수],c,대상 | c,[횟수],대상, w(단어), $(줄의 끝)

파일 내에서의 현재 위치와 파일 상태 보기, 파일의 마지막으로 이동, 파일의 시작으로 이동, 줄번호로 이동; Ctrl+G, G, gg, [줄번호]G
텍스트 검색 -> 다음, 이전, 원래 자리로 돌아가기, 다시 뒤로 가기; /[검색할텍스트] -> n, Shift+N, ?, Ctrl+O, Ctrl+I
(),[],{}의 짝 찾기; %
치환(subtitute); :/s/[기존값]/[새값](/g), :#,#s/old/new/g, :%s/old/new/g, :%s/old/new/gc

```
외부 명령어 실행; :![외부명령어]
파일 이름으로 저장; :w [FILENAME]
비주얼 모드로 선택 -> 선택한 것으로 파일이름 저장; v -> w: [FILENAME]
파일이름을 불러와 붙이기; :r [FILENAME]

커서 아래에 줄을 만들고 편집, 커서 위에 줄을 만들고 편집; o, O
커서 다음에 입력(append); a
다른 버전의 치환(Replace); R
비주얼모드 -> 복사(yank(copy), 붙이기(paste), 한 단어 복사, 다음 문장 끝으로 이동; v -> y, p, yw, j$
대소문자 구분 안하는 세팅, hlsearch와 incsearch 옵션 설정, 대소문자 구분 끄기, ;:set ic, :set hls is, :set noic, :nohlsearch, :/ignore\c

<HELP> | <F1> | :help -> :q, Ctrl+W Ctrl+W, :q
:e ~/.vimrc | :e ~/_vimrc -> :r $VIMRUNTIME/vimrc_example.vim, :w
Ctrl+d, <TAB>, :set nocp
```

VI 명령어

http://www.abyul.com/zbxe/aMST_ETC/70940

unix_linux_vi_commands_20091110.xls

▲ UNIX의 VI 명령어 정리 From. http://hbesthee.tistory.com/494

☆ 삽입 명령

명령어	설명
a	커서 뒤에 입력
A	라인 끝에 입력
i	커서 앞에 입력
I	라인시작 부분에 입력
o	커서 있는 라인 밑에 입력
O	커서가 있는 라인 위에 입력

☆ 커서 이동 명령

명령어	설명
h	왼쪽으로 커서 한 칸 이동
H	화면의 처음으로 이동
L	오른쪽으로 한 칸 이동
L	화면 끝으로 이동
e	다음 단어의 마지막으로 이동
E	커서를 공백으로 구분된 다음 단어 끝으로 이동
b	한 단어 뒤로
B	커서를 공백으로 구분된 이전 단어로 이동

명령어	설명
w	커서를 한 단어 뒤로
W	커서를 공백으로 구분된 다음 단어로 이동
k	커서를 한 라인 위로
j	커서를 한 라인 아래로 이동
O	커서를 라인의 시작으로 이동
$	커서를 라인의 끝으로 이동
Enter	커서를 다음 라인 시작으로 이동
-	커서를 전 라인의 시작으로 이동
Ctrl + F	다음 화면으로 이동
Ctrl + D	화면의 반만 앞으로 이동
Ctrl + B	전 화면으로 이동
Ctrl + U	화면의 반만 뒤로 이동
G	커서를 텍스트 마지막 라인으로
숫자G	커서를 숫자 라인만큼 이동
M	커서를 화면 중간 라인으로 이동
“	커서를 전 위치로 이동
(	문장의 시작으로 이동
{	문단의 시작으로 이동
)	문장 끝으로 이동하여 다음 단어의 시작으로 커서 이동
}	문단 끝으로 이동

☆ 방향키를 이용한 커서 이동 명령

명령어	설명
←, Del	왼쪽으로 커서 한 칸 이동
PageUp	화면 위로 이동
→, Space	오른쪽으로 한 칸 이동
PageDown	화면 아래로 이동
↑	윗 줄로 커서 이동
Enter	다음 줄 첫 칸으로 이동
↓	아래 줄로 커서 이동
Esc	다음 줄 첫 칸으로 이동
Home	줄 처음 칸으로 이동

☆ 삭제 명령

명령어	설명
x	커서가 있는 문자 삭제
X	커서가 있는 문자 앞에 있는 문자 삭제
dw	커서가 있는 단어 삭제
db	커서가 앞에 있는 단어 삭제
dW	공백으로 구분된 뒷 단어 삭제
dB	공백으로 구분된 앞 단어 삭제
dd	커서가 있는 라인 삭제
D	커서가 있는 라인의 나머지 삭제
d)	문장의 나머지 삭제

명령어	설명
d}	문단의 나머지 삭제
dG	파일의 나머지 삭제
dH	화면의 시작까지 삭제
dL	화면의 나머지 삭제
J	커서와 다음 단어의 공백을 모두 삭제

☆ 바꾸기 명령

명령어	설명
r	커서에 있는 문자 대치
R	입력 모드로 한 문자씩 덮어씀
s	커서가 있는 문자 삭제 후 입력 모드로 전환
S	커서가 있는 줄을 삭제 후 입력 모드로 전환
cb	커서가 있는 앞 문자 삭제 후 입력 모드
cW	공백으로 구분된 뒷 단어를 삭제 후에 입력 모드
cB	공백으로 구분된 앞 단어 삭제 후 입력 모드
cc	커서가 있는 라인을 삭제하고 입력 모드
C	커서가 있는 라인의 나머지를 삭제하고 입력 모드로 전환
cO	커서에서부터 라인의 시작까지 텍스트 바꾸기
c	특정 텍스트 바꾸기
c)	문장의 나머지 바꾸기
c}	문단의 나머지 바꾸기
cG	파일의 나머지 바꾸기
cm	표시까지 모든 것 바꾸기
cL	화면의 나머지 바꾸기
ch	화면의 시작까지 바꾸기

☆ 복사

명령어	설명
yw	커서가 있는 단어를 복사
yb	커서가 있는 앞 단어를 복사
yW	공백으로 구분된 뒷 단어 복사
yB	공백으로 구분된 앞 단어를 복사
y	특정한 다음 텍스트 복사
yy	커서가 있는 라인을 복사, 커서가 가리키는 곳으로 라인을 이동
y)	문자의 나머지 복사
y}	문단의 나머지 복사
yG	파일의 나머지 복사
yH	화면의 시작까지 복사
yL	화면의 나머지 복사

☆ 텍스트 이동

명령어	설명
p	삭제나 복사된 텍스트를 커서가 있는 문자나 라인 뒤에 삽입
P	삭제나 복사된 텍스트를 커서가 있는 문자나 라인 앞에 삽입
dw p	커서가 있는 단어를 삭제한 후 이를 원하는 곳 커서 뒤로 삽입
dw P	커서가 있는 단어를 삭제한 후 이를 변경한 커서가 있는 곳으로 삽입
d p	지정한 다음 텍스트로 삭제한 후 커서가 가리키는 곳으로 이동
d) P	문장의 나머지로 이동
d} p	문단의 나머지로 이동
dG P	파일의 나머지로 이동
dH P	화면 시작 부분으로 이동
dL P	화면의 나머지를 이동

☆ vi 에디터 종료 마치기 명령

명령어	설명
:q	그대로 종료하기
:q!	변경된 내용을 저장하지 않고 강제로 종료하기
:wq	변경된 내용을 저장하고 종료하기
:x	:wq와 동일한 명령
ZZ	:wq와 동일한 명령

☆ 검색

명령어	설명
/pattern	텍스트에서 앞으로 패턴 검색
>pattern	텍스트에서 뒤로 패턴 검색
n	앞 또는 뒤로 이전 검색 반복
N	반대 방향으로 이전 검색 반복
/	전 검색을 앞으로 반복
?	전 검색을 뒤로 반복

☆ 문자열 치환

명령어	설명
:s/old/new	현재 행의 처음 old를 new로 교체
:s/old/new/g	현재 행의 모든 old를 new로 교체
:10,20s/old/new/g	10행부터 20행까지 모든 old를 new로 교체
:-3,+4s/old/new/g	현재 커서 위치에서 3행 위부터 4행 아래까지 old를 new로 교체
:%s/old/new/g	문서 전체에서 old를 new로 교체
:%s/old/new/gc	문서 전체에서 old를 new로 확인하며 교체
:g/pattern/s/old/new/g	Pattern이 있는 모든 행의 old를 new로 교체
:g/pattern/s//new/g	:%s/old/new/g와 동일

☆ 옵션

옵션	기능	약어	기본값
autoindent	들여 쓰기 가능, 탭으로 들여 쓰기 범위 지정	ai	off
autoprint	줄이 바뀔 때 현재 줄을 화면상에서 출력	ap	on

옵션	기능	약어	기본값
errobells	명령 에러가 발생시 삑 소리나게 함	ed	off
number	줄 번호를 나타나게 함	nu	off
report	편집시 메시지를 보낼 편집 변화 크기 지정	report	5
showmatch	가로 닫기 괄호를 사용할 때 일치하는 가로 열기 괄호를 보여줌	sm	off
wam	저장하지 않고 vi 종료할 때 경고 메시지를 뿌려 줌	wam	on
ignorecase	검색 패턴에 사용되는 대소문자 구별하지 않음	ic	on
tabstopp=n	탭 공백을 n 수만큼 지정	ts=n	8
wrapmargin=n	텍스트 오른쪽 여백을 n 수만큼 지정	wm=n	0

☆ Mark 사용

명령어	설명
mx	현재 위치를 x 이름의 마크로 저장
``	이전에 마크한 위치로 이동
`x	마크한 위치(행, 열)로 이동
''	이전에 마크한 줄로 이동
'x	마크한 줄로 이동

☆ Named Buffer 사용

명령어	설명
"ayy	현재 줄을 "a 버퍼에 복사
"Ayy	기존의 버퍼에 현재 줄을 버퍼에 추가
"ap	"a 버퍼에 복사된 데이터를 붙여 넣기 \\ ☞'a' 부터 'z' 까지 사용가능

☆ 여러 문서 편집 (vi filename1, filename2 ... 로 실행 ; 여러 파일 열기)

명령어	설명
:n	vi로 open한 여러 파일중 다음 파일로 전환
:N	vi로 open한 여러 파일중 이전 파일로 전환
:4n	여러 파일중 4개 파일 skip후 파일 Open
:args	현재 열린 모든 파일중 현재 편집중인 파일 표시

_gvimrc

```
set nocompatible              " be iMproved, required
filetype off                  " required
set shellslash


"-------------------------------------------------------------
" Specify a directory for plugins
" - For Neovim: stdpath('data') . '/plugged'
" - Avoid using standard Vim directory names like 'plugin'
```

```
call plug#begin('~/.vim/plugged')

" Make sure you use single quotes

" Shorthand notation; fetches https://github.com/junegunn/vim-easy-align
Plug 'junegunn/vim-easy-align'

" Any valid git URL is allowed
Plug 'https://github.com/junegunn/vim-github-dashboard.git'

" Multiple Plug commands can be written in a single line using | separators
Plug 'SirVer/ultisnips' | Plug 'honza/vim-snippets'

" On-demand loading
Plug 'scrooloose/nerdtree', { 'on':  'NERDTreeToggle' }
Plug 'tpope/vim-fireplace', { 'for': 'clojure' }

" Using a non-default branch
Plug 'rdnetto/YCM-Generator', { 'branch': 'stable' }

" Using a tagged release; wildcard allowed (requires git 1.9.2 or above)
Plug 'fatih/vim-go', { 'tag': '*' }

" Plugin options
Plug 'nsf/gocode', { 'tag': 'v.20150303', 'rtp': 'vim' }

" Plugin outside ~/.vim/plugged with post-update hook
Plug 'junegunn/fzf', { 'dir': '~/.fzf', 'do': './install --all' }

" Unmanaged plugin (manually installed and updated)
Plug '~/my-prototype-plugin'

" Plugin airline
Plug 'vim-airline/vim-airline'
Plug 'vim-airline/vim-airline-themes'

Plug 'scrooloose/syntastic'
Plug 'posva/vim-vue'
Plug 'altercation/vim-colors-solarized'

" Initialize plugin system
call plug#end()
"------------------------------------------------------------
" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()
" alternatively, pass a path where Vundle should install plugins
"call vundle#begin('~/some/path/here')

" let Vundle manage Vundle, required
Plugin 'VundleVim/Vundle.vim'
```

```
" The following are examples of different formats supported.
" Keep Plugin commands between vundle#begin/end.
" plugin on GitHub repo
Plugin 'tpope/vim-fugitive'
" plugin from http://vim-scripts.org/vim/scripts.html
" Plugin 'L9'
" Git plugin not hosted on GitHub
Plugin 'git://git.wincent.com/command-t.git'
" git repos on your local machine (i.e. when working on your own plugin)
"Plugin 'file:///home/gmarik/path/to/plugin'
" The sparkup vim script is in a subdirectory of this repo called vim.
" Pass the path to set the runtimepath properly.
Plugin 'rstacruz/sparkup', {'rtp': 'vim/'}
" Install L9 and avoid a Naming conflict if you've already installed a
" different version somewhere else.
" Plugin 'ascenator/L9', {'name': 'newL9'}

Plugin 'vim-airline/vim-airline'
Plugin 'vim-airline/vim-airline-themes'
Plugin 'scrooloose/syntastic'
Plugin 'posva/vim-vue'
Plugin 'altercation/vim-colors-solarized'


" All of your Plugins must be added before the following line
call vundle#end()            " required
filetype plugin indent on    " required
" To ignore plugin indent changes, instead use:
"filetype plugin on
"
" Brief help
" :PluginList       - lists configured plugins
" :PluginInstall    - installs plugins; append `!` to update or just
:PluginUpdate
" :PluginSearch foo - searches for foo; append `!` to refresh local cache
" :PluginClean      - confirms removal of unused plugins; append `!` to
auto-approve removal
"
" see :h vundle for more details or wiki for FAQ
" Put your non-Plugin stuff after this line
"-------------------------------------------------------------
" An example for a gvimrc file.
" The commands in this are executed when the GUI is started, after the vimrc
" has been executed.
"
" Maintainer:  Bram Moolenaar <Bram@vim.org>
" Last change: 2016 Apr 05
"
" To use it, copy it to
"         for Unix:  ~/.gvimrc
```

```
"        for Amiga:  s:.gvimrc
"   for MS-Windows:  $VIM\_gvimrc
"        for Haiku:  ~/config/settings/vim/gvimrc
"      for OpenVMS:  sys$login:.gvimrc

" Make external commands work through a pipe instead of a pseudo-tty
"set noguipty

" set the X11 font to use
" set guifont=-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1

set ch=2        " Make command line two lines high

set mousehide       " Hide the mouse when typing text

" Make shift-insert work like in Xterm
map <S-Insert> <MiddleMouse>
map! <S-Insert> <MiddleMouse>

" Only do this for Vim version 5.0 and later.
if version >= 500

  " Switch on syntax highlighting if it wasn't on yet.
  if !exists("syntax_on")
    syntax on
  endif

  " For Win32 version, have "K" lookup the keyword in a help file
  "if has("win32")
  "  let winhelpfile='windows.hlp'
  "  map K :execute "!start winhlp32 -k <cword> " . winhelpfile <CR>
  "endif

  " Set nice colors
  " background for normal text is light grey
  " Text below the last line is darker grey
  " Cursor is green, Cyan when ":lmap" mappings are active
  " Constants are not underlined but have a slightly lighter background
  highlight Normal guibg=grey90
  highlight Cursor guibg=Green guifg=NONE
  highlight lCursor guibg=Cyan guifg=NONE
  highlight NonText guibg=grey80
  highlight Constant gui=NONE guibg=grey95
  highlight Special gui=NONE guibg=grey95

endif


set guifont=D2Coding:h12:cHANGEUL:qDEFAULT
" set guifont=Source_Code_Pro:h12:cANSI:qDEFAULT
```

```
if has("gui_running")
  set encoding=utf-8

  source $VIMRUNTIME/mswin.vim
  behave mswin
  if has('win32')
    set keymodel=startsel
    " set guifont=Source\ Code\ Pro:h16:cANSI:qDEFAULT
    " set guifont=JetBrains\ Mono\ Regular:h18:cANSI:qDEFAULT
    " set guifont=D2Coding\ ligature:h16:cHANGEUL:qDEFAULT
      set renderoptions=type:directx,
       \gamma:1.0,
       \contrast:0.5,
       \level:1,
       \geom:1,
       \renmode:5,
       \taamode:1

      " restore Ctrl-F to Page down
    unmap <C-F>
  elseif has('gui_macvim')
    set guifont=JetBrains\ Mono\ 16
    set antialias
  else
    set guifont=JetBrains\ Mono\ 16
  endif

  " Turn off toolbar
  set guioptions-=T
  " Turn on menu
  set guioptions+=m
end

set smartindent
set tabstop=2
set expandtab
set shiftwidth=2

" colorscheme darkblue

"set statusline+=%#warningmsg#
"set statusline+=%{SyntasticStatuslineFlag()}
"set statusline+=%*

"let g:syntastic_always_populate_loc_list = 1
"let g:syntastic_auto_loc_list = 1
"let g:syntastic_check_on_open = 1
"let g:syntastic_check_on_wq = 0

"let g:airline#extensions#tabline#enabled = 1
```

```
let g:airline_powerline_fonts = 1

syntax enable
set background=dark
colorscheme solarized

set encoding=utf-8
set
rop=type:directx,gamma:1.0,contrast:0.5,level:1,geom:1,renmode:4,taamode:1

"let g:airline_section_z = airline#section#create(['windowswap', '%3p%% ',
'linenr', ':%3v'])

set laststatus=2

set nu
set clipboard=unnamed
```

References

- vim 에디터 이쁘게 사용하기
- vim 메뉴 및 폰트 설정
- windows 용 vim 폰트(font) 및 렌더링(redering) 옵션 설정
- vundle 플러그인 관리자 설치 및 유용한 플러그인
- Vim Plug
- 밤앙개 블로그 - 디자인과 개발
- [Linux] Vim 화면 분할,(상/하/좌/우 분할 방법 정리)
- VIM 기본 설정과 추천 플러그인 및 사용법 정리
- 지금 사용하는 Vim 플러그인 100개

From:
https://theta5912.net/ - **reth**

Permanent link:
https://theta5912.net/doku.php?id=public:computer:vim_cheat_sheet&rev=1647850850

Last update: **2022/03/21 17:20**