- 2 - Vim quickstart
  -- Modes --
1) Normal Mode: everything that's typed is a (case sensitive) command.
2) Insert Mode: enter the insert mode with "i". Exit with ESC.
3) Line Mode (Command Line Mode): enter the line mode with ":".

- 3 - Vim Essentials
  -- Essential Navigation Commands --
j, k, h, l (Move the cursor around)
CTRL-f (Move 1 page forward)
CTRL-b (Move 1 page back)
w (Move forward 1 word)
b (Move back 1 word)
W (Move forward 1 word irrespective of punctuation)
B (Move back 1 word irrespective of punctuation)
gg (Go to the start of the file)
G (Go to the end of the file)
<n>gg (Go to the nth line)
:44 ENTER (In line mode, goes to the 44th line)
:$ ENTER (In line mode, goes to the last line)
CTRL-G (Shows document informations)
:set ruler (In line mode, enable the ruler)
:set noruler (In line mode, disables the rules)
:set ruler! (In line mode, if the ruler is enabled, it disables it.)
g CTRL-G (Show more document informations)
z ENTER (Keeps the cursor at the current position, but reposition the view)
0 (Goes to the first column of the line)
<n> (j, k, l, h moves n characters in the selected direction)
^ (Moves to the first character of the line)
$ (Goes to the end of the line)
  -- Deleting text and thinking in Vim --
x (deletes the character at the right of the cursor)
X (deletes the character at the left of the cursor)
dw (deletes an entire word starting from the right of the cursor - d for the delete operation, w for the word)
dW (like the previous command but punctuation ignored)
d$ (deletes all the text from the right of the cursor)
d0 (deletes all the text from the left of the cursor)
db (deletes an entire word starting from the left of the cursor)
dB (like the previous command but ignores punctuation)
dd (deletes the whole line)
. (repeats the last command)

- 4 - The Vim Help System
:h[elp] (Open up the help system in a new read-only window)
:q[uit] (Close the help window)
CTRL-] on a certain word, opens the help for that command.
CTRL-o return back to the previous position in the documentation
CTRL-i goes forward to the position in the documentation
CTRL-ww switch window and position the cursor in the other window outside the documentation.
CTRL-d shows a list of commands that completes the command we're writing
TAB shows a wildmenu (:set wildmenu) for the possible completion of the command.

- 5 - Deleting, Yanking and Putting
Cut, Copy and Paste in vim are called Kill, Yank and Put.
Killing and yanking are operations stored in various types of registers:
1. Unnamed register

2. Numbered registers
3. Named registers
4. Black Hole register.

Registers are prefixed with "
Unnamed register: ""
Numbered registers: "0, "1, "2, etc.
Named registers: "a, "b, etc. (there are 26, from a to z)
Blackhole register: "_

"" contains text from the operations d, c, s, x, and y.
"0 holds the last yanked text (y)
"1 holds the last deleted (d) or changed (c) text.
Subsequent numbered registers shift with each d or c operation.
"_ the blackhole register stores text on demand without affecting other registers.

The d series of commands do not delete text physically. They kill the text instead.
For example, dd will kill the whole line and puts the line inside the unnamed register and in "1 register.

:reg (Inspect the content of the registers)
:reg 14 (Shows the content of registers "1 and "4)
u (Undo the last command)
CTRL-r (Redo the last command)
y (Yank (copy) text. This command accepts all previous variations od the 'd' command, yw, yy, etc).
p (Put (paste) the killed or yanked text in the line under the cursor)
P (Put the killed or yanked text in the line above the cursor)
"r d [y] (Kill or yank text in the specified register r)
"n p (Put the text stored in the register n)
"a y (Yank text in the named register a
"A y (Yank other text appending it in the named register a)

- 6 - Transforming and substituting text
I (Move the cursor to the first non-blank character of a line and enters insert mode)
a (Open insert mode at the right of the cursor)
A (Goes to the end of the line and enters insert mode)
o (Insert a new line and enter insert mode)
O (Insert a new line above the line and enters insert mode)
[n]i text ESC (Repeats the insertion command for n times, eg. 80i * ESC produces a line with 80 stars)
R (Enters replace mode. Everything inserted will substitute the text under the cursor)
r (Enters replace mode to substitute just one carachter and goes back to normal mode immediately)
cw (Change mode for a word, deletes the word and enters insert mode for substitution)
"r cw (Saves the deleted text in a named register r)
c$ (Like before, but replaces all the text from the cursor to the end)
cc (Like before, but applys to the whole line)
~ (swap a character at the right of the cursor from lowercase to uppercase and viceversa)
g~w (swap characters from lowercase to uppercase for a whole word)
g~~ (swap characters from lowercase to uppercase for a whole line)
g~$ (swap characters form lowercase to uppercase for the whole line from the current cursor position)

gUw (force all characters uppercase for a word, regardless of the capitalization)
gU$ (force all characters uppercase for a whole line, regardless of the capitalization)
guw (force all characters lowercase for a word, regardless of the capitalization)
gu$ (force all characters lowercase for w whole line, regardless of the capitalization)
J (Joins two consecutive lines. It appends spaces intelligently)
[n]J (Joins n consecutive strings appending a space for each joined line)
gJ (Joins two consecutive lines without appending spaces)

-- Search, Find and Replace 1
f c (In a line, search for the first occurrence of the character c. Repeat the same search along the line with :, in the opposite direction with ,)
F c (In a line, back-search for the first occurrence of the character c. Repeat with : while repeating in the opposite direction with ,)
t c (In a line, search for the first occurrence of a character and position the cursor on the character before it)
T c (In a line, back-search for the first occurrence of the character c.)
dtw (Search commands are motions, they can be concatenated with other commands. This deletes all the characters from the cursor to the first occurrence of w)
/word (Search the whole file for the word word. Repeat the search with 'n')
:set is [nois] or [is?] to see if active (Enables or disables incremental search. When starting a search, characters are highlighter)
:set hls [nohls] or [hls?] to see if active (Enables or disables highlight search. When enabled, all the searches in the page are highlighted)
/word ENTER, command (eg. cw and the text to replace) ESC (to search for a work, execute a command, then press 'n' to go the next search and press . to repeat the command. N search for the reverse order)
?word (back-search for a word, 'n' to repeat the search, N to reverse the order)
* (keep pressing asterisk) (Search for the word under the cursor, n to repeat the command and continuing searching)
# (keep pressing #) (Back-search for the word under the cursor, n to repeat the command, N to reverse the order)

-- Search, Find and Replace 2
:s/old/new/ (Substitute old with new in the linei, only first occurrence)
:s/old/new/[flags] (with flag g, all occurrence in the line will be substituted)
:[range]s/old/new/[flags] (A range is one or more lines, separated by a , or ; The simplest range is a line number)
:1s/is/isn't/g (Will substitute all occurrences of is with isn't on line 1)
:1,5s/for/FOR/g (Will substitute all occurrences of for with FOR in the first 5 lines)
Other range symbols are:
. --> Current line
$ --> Last line
% --> All lines (entire file)
:.,$s/old/new/g (Substitute all occurrences of old with new from the current line to the end)
:/Global/,/Local/s/net/org/g (In the range from the line containing the word Global to the word Local, substitute all occurs of net with org)

:/Global/,$s/net/org/g (From the line containing Global to the end of the file, substitute all occurs of net with org)
In order to substitute words containing characters that are also part of the command, we need to quote them. For example:
to substitute /var/spool with /usr/local:
:s/\/var\/spool/\/usr\/local/
There is an easier way: using another symbol as the character separator, for example the character # (others can be used, like +, |, etc)
:s#/var/spool#/usr/local# (Substitute /var/spool with /usr/local, using a different character separator)
:set nu [nonu] [nu!] [nu?] (Set line numbers in the editor)

- 7 - Text Objects and Macros

-- Text Objects
When operating with text objects, the command operate on the entire object regardless of the cursor position.
daw (delete a word - deletes an entire word, including the whitespace after it. Word in this case is considered a text object)
diw (delete inner word - deletes an entire word preserving the whitespace after it)
Patterns for text objects are:
{operator}{a}{object}
{operator}{i}{object}
a and i are punctuation and whitespaces delimited, A and I don't consider punctuation.
eg. daw (delete a word) or ciw (change inner word)

-- Sentences and paragraph text objects
A sentence boundary is a period.
To delete full sentences, 'is' or 'as' are used in conjuction with the operator.
das (deletes a sentence)
dis (deletes an inner sentence preserving the whitespace at the end)

A paragraph boundary is a blank line.
To delete full paragraph, 'ap' or 'ip' are used in conjunction with the operator.
dap (deletes a paragraph)
dip (deletes an inner paragraph preserving the whitespace at the end)

-- Custom text objects
In programming languages, delimiters for text are of various types.
It can be specified after the 'a' or 'i' operator.

ca] (changes all the text enclosed between [and], including the brackets)
ci] (changes all the text enclosed between [and], preserving the brackets which are the boundaries)
da) or dab (changes all the text enclosed between (and), including the brackets, also called blocks with the specific command dab)
di) or dib (changes all the text enclosed between (and), preserving the brackets, also called blocks with the specific command dib)
da} or daB (changes all the text enclosed between { and }, including the brackets, also called blocks with the specific command daB)
di} or diB (changes all the text enclosed between { and }, preserving the brackets, also called blocks with the specific command diB)
This series of commands also works with characters <>, "", '', ``
b are blocks enclosed in round parentheses, B are blocks enclosed in curly braces.

-- Tag objects
In HTML for example, text constitutes tags. Vim has specific tag text objects, represented by the operator 't'.
Each text enclosed between angular brackets are considered tags in Vim.
dat (deletes all the text enclosed between two tags, including the tags)
dit (deletes all the text enclosed between two tags, preserving the tags)
-- Macros
Macros are a recorded sequence of keystokes. Keystrokes are recorded into registers and played back when requested.
qa (Start recording a macro inside register a)
qA (Start recording an updated macro at register a appending to it)
q (stops recording the macro when done)
@a (plays the macro at register 'a')
@@ (replays the last played macro)
2@a (repeats the macro in the register a 2 times)
:15,27 normal @a (Executes the macro in the range of lines starting from 15 to 27)

Before recording macros, it's good practice to:
- Normalize the cursor position using '0'
- Record commands
- Go to the next line using j

- 8 - Visual Mode
There are three types of visual modes.
1. Characterwise visual mode
2. Linewise visual mode
3. Blockwise visual mode

v (Enters characterwise visual mode)
V (Enters linewise visual mode)
CTRL+v (Enters blockwise visual mode)
o (Switch selection in the opposite direction)
O (In visual block mode, it moves the cursor in the opposite position of the line)
gv (If exiting the visual mode for some reason, it re-enters it preserving selections made in it)
In Visual Mode, operation that can be done on the selected text are:
- ~ - Switch case
- c - Change
- d - Delete
- y - Yank
- r - Replace
- x - Delete
- I - Insert
- A - Append
- J - Join
- u - Make lowercase
- U - Make uppercase
- > - Shift text right (Settings involved :shiftwidth, :tabstop, :expandtab)
- < - Shift text left

Selection doesn't involve only motion commands, but also text objects.
Selection can be done using text objects without the command. For example:
iw (Selects in visual mode an inner word)
ap (Selects in visual mode the whole paragraph)
is (Selects in visual mode the inner statement)

Traditional selection commands for text objects are also possible, for example 'vyap' will work.
In visual block mode, there are times that a line longer than its successor is longer. It won't be selected using traditional motions. Using the character "$" it will be selected.
When lines are selected in visual mode and the user enters command line mode with :, a set of characters are presented:
:'<,'> (Represents the boundaries for the visual selection in visual mode)
Substitution command can be entered like this:
:'<,'>s/United States of America/USA/g
:'<,'>ce (Center the selection with 80 columns format)
:'<,'>ce 40 (Center the selection with 40 columns format)
:'<,'>le (Align left the selection)
:'<,'>le5 (Align left with 5 space padding)
:'<,'>ri5 (Align right with 5 space padding)

- 9 - Vim Settings, Preferences and Customizations
:set (Shows all the options set with the set command)
:set command& (Will reset the value set by the command to its default value)
:h option-list (Shows the list of configurable options for the .vimrc file)
:option (Shows the list of configurable options for the .vimrc file among with their documentation)
:set history=1000 (Set the history for the commands)
" (Represents a comment in the .vimrc file)
:set ruler (Show the cursor position)
:set showcmd (Shows incomplete commands when digiting them in command mode)
:set wildmenu (Shows a menu during tab completion of commands)
:set scrolloff=5 (Defaults to 0. Influence the z ENTER command. It will leave 5 rows when scrolling the text)
:set hlsearch (Enables highlight searching)
:set incsearch (Enables incremental searching)
:set ignorecase
:set smartcase
:set number (Shows line numbers)
:set backup (Saves a backup file everytime a file is saved, eg. ~filename.txt)
:set bex=something (Set the backup extension)
:set autoindent [ai] (Enables autoindenting)
:set smartindent [si] (Enables smart indenting, useful for source code)
:set bg=dark [=light] (Set the background color)
:color colorscheme (Set the color scheme. CTRL-D for seeing the possible completions)
:map (prints the key mapping, the different ones from the defaults)
:map KEY KEYSTROKES (Assign a keystroke to a key)
:source .vimrc (Reloads the .vimrc file or any file specified)
When overriding commands, it is possible to use a namespace to avoid to overwrite existing commands. This is a special character called leader key, and its default is \.
map <leader>w :w!<CR>
When you press \w, the executed command is the defined one.
The leader key can be redefined with:
let mapleader="," (Redefines the leader key from \ to ,)
:mkvimrc (Creates a new .vimrc file)

- 10 - Vim Buffers and Windows
   Buffers

Buffers can have three states, visible in the buffer list:
a - active and visible
h - hidden, loaded in memory but not visible
i - inactive, not loaded in memory and not displayed.
% means current buffer, # means alternate buffer.
+ means the buffer has unsaved modifications.
:set hidden (Makes Vim unload buffers that are not visible and reload them once they're visible)

:e file (Edit another file in the Vim buffer and opens it in the current buffer)
:buffers (Shows the available open buffers)
:b3 (Opens buffer number 3)
:b buffer_name (Open buffer named buffer_name)
:ls (like :buffers)
:bn (Open the next buffer)
:bp (Open the previous buffer)
:bf (Open the first buffer)
:bl (Open the last buffer)
CTRL-^ (Open the alternate buffer [the buffer before the current one])
:qall! (Closes all buffers without saving)
:wall (Saves the content of all buffers)
:badd <filename> (Open a new buffer with the file content)
:bd (Deletes (unload) the buffer from the list)
:1,3bd (Unloads buffers 1 to 3)
:%bd (Unloads all buffers)
:bufdo set nu (Executes the command set nu for all buffers)
:bufdo %s/#/@/g | w (Replaces for all buffers the # with @, then save, then proceed for the next buffer)
:E (Opens the Explore window)
   Windows
:sp or CTRL-w s (Splits window horizontally)
:vs or CTRL-w v (Splits window vertically)
:sp buf-name (Splits horizontally and puts the content of the specified buffer)
CTRL-w q (Quits the window)
:on or CTRL-w o (Closes all the windows except the current one)
CTRL-ww (Moves the cursor to the next window. It cycles from left to the right)
CTRL-w j, k, h, l (Moves the cursor to down, up, left or right window to the cursor)
CTRL-w + or - (Increase or decrease the height of the window)
CTRL-s > or < (Increase or decrease the width of the window)
CTRL-w _ (Maximize the height of the window)
CTRL-w | (Maximize the width of the window)
CTRL-w = (Makes all windows of equal size)
CTRL-w r or R (Rotates windows horizontally to the right or left)
CTRL-w J, K, H, L (Moves windows in the specified direction)
:ba (Opens all the buffer each in its own window)
:windo <command> (Applies the command to all the windows)