Vim

A Not-So-Brief Introduction

What is Vim?

Vim is a multiplatform, massively configurable, extensible, modal **text editor**

- *Not* an IDE
- A **console application** (in its best and truest form)
- Vi improved

Why is Vim?

- Employ muscle memory to edit code "**at the speed of thought**"
- Edit code, configuration files, log files, and much more *identically*
- Make it the perfect IDE for **you**
- Integration with command line
- Feel superior to other people!
- Fun!

Why is Vim hard?

- A *modal* interface
- Obtuse hotkeys and commands
- Enormous set of commands
- Easy to hit the wrong key when learning

Why is Vim not actually that hard?

- A *modal* interface
- Obtuse hotkeys and commands
- Enormous set of commands
- Easy to hit the wrong key when learning

but you should understand it by the end of this lecture

there's actually logic to almost every hotkey

but you only need to know a few

we'll go over incantations to fix whatever you messed up

Opening and Closing Vim

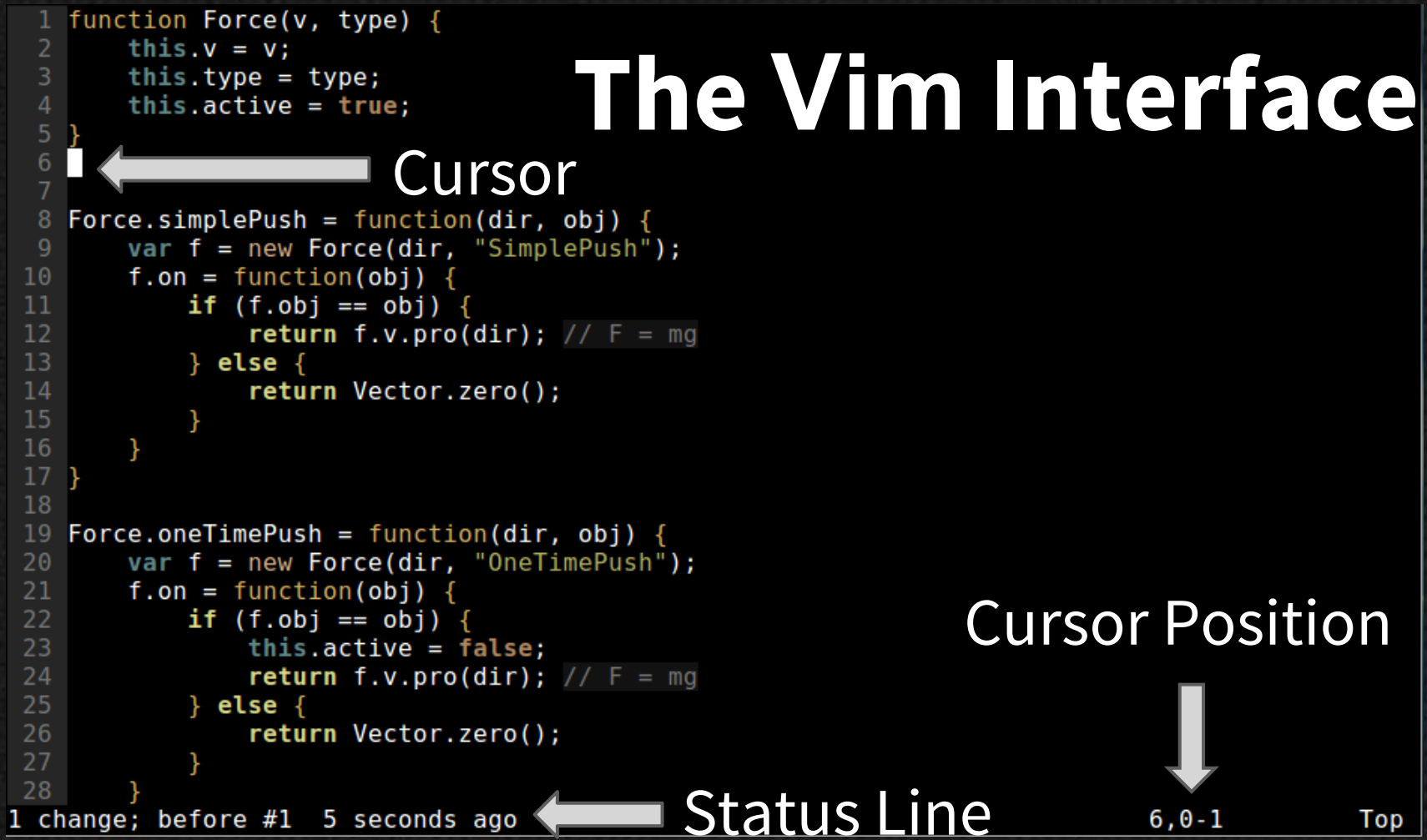
To open Vim from a terminal:

```
$ vim file.txt
```

To close Vim (and save your work):

```
:wq<Enter>
```

The Vim Interface




```
function Force(v, type) {
    this.v = v;
    this.type = type;
    this.active = true;
}

Force.simplePush = function(dir, obj) {
    var f = new Force(dir, "SimplePush");
    f.on = function(obj) {
        if (f.obj == obj) {
            return f.v.pro(dir); // F = mg
        } else {
            return Vector.zero();
        }
    }
}

Force.oneTimePush = function(dir, obj) {
    var f = new Force(dir, "OneTimePush");
    f.on = function(obj) {
        if (f.obj == obj) {
            this.active = false;
            return f.v.pro(dir); // F = mg
        } else {
            return Vector.zero();
        }
    }
1 change; before #1  5 seconds ago                    6,0-1         Top
```

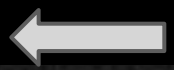
The Vim Interface

```
1  function Force(v, type) {
2      this.v = v;
3      this.type = type;
4      this.active = true;
5  }
6
7
8  Force.simplePush = function(dir, obj) {
9      var f = new Force(dir, "SimplePush");
10     f.on = function(obj) {
11         if (f.obj == obj) {
12             return f.v.pro(dir); // F = mg
13         } else {
14             return Vector.zero();
15         }
16     }
17 }
18
19 Force.oneTimePush = function(dir, obj) {
20     var f = new Force(dir, "OneTimePush");
21     f.on = function(obj) {
22         if (f.obj == obj) {
23             this.active = false;
24             return f.v.pro(dir); // F = mg
25         } else {
26             return Vector.zero();
27         }
28     }
-- INSERT --                                  6,1           Top
```

We’re in Insert Mode

The Vim Interface

```
1  function Force(v, type) {
2      this.v = v;
3      this.type = type;
4      this.active = true;
5  }
6
7
8  Force.simplePush = function(dir, obj) {
9      var f = new Force(dir, "SimplePush");
10     f.on = function(obj) {
11         if (f.obj == obj) {
12             return f.v.pro(dir); // F = mg
13         } else {
14             return Vector.zero();
15         }
16     }
17 }
18
19 Force.oneTimePush = function(dir, obj) {
20     var f = new Force(dir, "OneTimePush");
21     f.on = function(obj) {
22         if (f.obj == obj) {
23             this.active = false;
24             return f.v.pro(dir); // F = mg
25         } else {
26             return Vector.zero();
27         }
28     }
:%s/active/inactive/g
```



Typing a command

Modes

Why have modes?

(# of keys) × (# of modes) = More Hotkeys!

The only modes you really need to know:

Normal Mode home base / command entry

Insert Mode add text to document

Visual Mode select text

Let’s do it!

Point your browsers to

coolwanglu.github.io/vim.js/web/vim.html

Moving Around the Document

Use hjkl to move around the document:

←h ↓j ↑k →l

Basic Text Editing

Hit `i` to enter **Insert Mode**, then type.

Hit `<Esc>` to return to **Normal Mode**.

Motions

A **motion** is a hotkey for moving the cursor.

Important motions:

- Next **w**ord / **b**ack a word
- **e**nd of word
- **^** - first non-whitespace character on line
- **$** - last character on line

Verbs

A **verb**, as you learned in grade school, is an *action word*. Verbs in Vim are how we change the document.

Examples:

delete - (requires a motion argument)

yank - copy text

paste - paste text at cursor position

Verbs

A **verb**, as you learned in grade school, is an *action word*. Verbs in Vim are how we change the document.

Examples:

change - delete & enter insert mode

open new line

Sentences

The Vim poweruser uses compound actions, which can be thought of like **sentences**.

[verb][# of times to repeat][motion]

Examples:

dw = ?

y2w = ?

Sentences

The Vim poweruser uses compound actions, which can be thought of like **sentences**.

[verb][# of times to repeat][motion]

Examples:

dw = Delete word

y2w = ?

Sentences

The Vim poweruser uses compound actions, which can be thought of like **sentences**.

[verb][# of times to repeat][motion]

Examples:

dw = Delete word

y2w = Yank 2 words

Advanced Motions

Now, let's look at more sophisticated ways of moving around the document.

`f`ind[_] - move cursor to next [_] on line

`t`ill[_] - move cursor *before* next [_] on line

Examples:

`fm` = ?

`d2t.` = ?

Advanced Motions

Now, let's look at more sophisticated ways of moving around the document.

`f`ind[_] - move cursor to next [_] on line

`t`ill[_] - move cursor *before* next [_] on line

Examples:

`fm` = Move cursor to next 'm' on line

`d2t.` = ?

Advanced Motions

Now, let's look at more sophisticated ways of moving around the document.

`f`ind[_] - move cursor to next [_] on line

`t`ill[_] - move cursor *before* next [_] on line

Examples:

`fm` = Move cursor to next 'm' on line

`d2t.` = Delete until just *before* the second following '.' on the current line

Vim Logic

There are a few patterns in hotkeys that are worth noting:

For **verbs**, doubling the hotkey applies it to the entire row:

- `yy` - yank entire row
- `cc` - delete all characters on row and open **Insert Mode**

If an action works forwards, capitalize it to do the same backwards:

- `p` / `P` - paste *after* cursor / paste *before* cursor
- `o` / `O` - open line *after* current / open line *before* current
- `f` / `F` - find letter forwards / find letter backwards

Putting it all together

Here are some of my most-used compound actions

`dd` - ?

`d$` - ?

`ct;` - ?

`ci(` - ?

`dG` - ?

Putting it all together

Here are some of my most-used sentences

`dd` - delete current line

`d$` - ?

`ct;` - ?

`ci(` - ?

`dG` - ?

Putting it all together

Here are some of my most-used sentences

`dd` - delete current line

`d$` - delete until end of current line

`ct;` - ?

`ci(` - ?

`dG` - ?

Putting it all together

Here are some of my most-used sentences

`dd` - delete current line

`d$` - delete until end of current line

`ct;` - delete all characters on line until semicolon, then enter **Insert Mode**

`ci(` - ?

`dG` - ?

Putting it all together

Here are some of my most-used sentences

`dd` - delete current line

`d$` - delete until end of current line

`ct;` - delete all characters on line until semicolon, then enter **Insert Mode**

`ci(` - delete all characters `i`nside parenthesis, then enter **Insert Mode**

`dG` - ?

Putting it all together

Here are some of my most-used sentences

`dd` - delete current line

`d$` - delete until end of current line

`ct;` - delete all characters on line until semicolon, then enter **Insert Mode**

`ci(` - delete all characters `i`nside parenthesis, then enter **Insert Mode**

`dG` - delete until end of document

Copy / Paste

Anything you yank, *as well as anything you delete* is automatically copied to a **register**. You can see the current contents of the registers by typing

`:registers<Enter>`

in **Normal Mode**.

To paste *after* the cursor, hit **p** in **Normal Mode**, or hit **P** to paste *before* the cursor.

Undo / Redo

Any **action**, or command that changes the document, can be undone or redone.

Not insertion of a single letter - *everything typed between entering and exiting* **Insert Mode**.

To undo, hit `u` in **Normal Mode**. To redo, hit `Ctrl+R`.

Undo / Redo

Vim's most bizarre and beautiful hotkey:

To redo again, hit `.`

You can also provide a number argument.

Examples:

(Last action is `2dw`), `4.` = ?

(Last action is `cw`, then we typed "dog", then returned to Normal Mode), `.` = ?

Undo / Redo

Vim's most bizarre and beautiful hotkey:

To redo again, hit `.`

You can also provide a number argument.

Examples:

(Last action is `2dw`), `4.` = delete 8 more words

(Last action is `cw`, then we typed "dog", then returned to Normal Mode), `.` = ?

Undo / Redo

Vim's most bizarre and beautiful hotkey:

To redo again, hit `.`

You can also provide a number argument.

Examples:

(Last action is `2dw`), `4.` = delete 8 more words

(Last action is `cw`, then we typed "dog", then returned to Normal Mode), `.` = replace next word with "dog"

Commands

To enter commands, be in **Normal Mode**, then hit **:** to open up the command line.

Tab for autocompletion of commands and files works!

Examples:

:w(rite) - saves current document

:sav(e) la.js - saves current document to la.js instead of previous filename

:r(ead) some.cpp - inserts contents of “some.cpp” here

Commands

To enter commands, be in **Normal Mode**, then hit **:** to open up the command line.

The only one you **absolutely must know**:

:h(elp) [*topic*] - opens help page entry for *topic*

What Just Happened?

If something weird just happened, here are a few standard things to try.

Issue	Incantation
Somehow, I deleted *everything*!	Undo with `u`
Some window popped up!	`:q<Enter>`
WTF is Replace Mode?	`<Esc>`

Topics for Next Week

- Autocompletion
- Marks and Folding
- Command Line Integration
- Search / Replace
- Global commands
- Multiple Insertion with Visual Block Mode
- Vim Plugins
- Configuring Vim / Ftplugin
- Snippets!

How To Learn More

- Use :help [topic]
- Play around!
- StackOverflow for specific topics
- **vim-adventures.com**
- **vimgolf.com**
- **vim.wikia.com/wiki/Vim_Tips_Wiki**

Let’s Try Vimtutor

1. Install Vim
   a. Linux - you already have it.
   b. Windows - **ftp://ftp.vim.org/pub/vim/pc/gvim74.exe**
   c. Mac - **code.google.com/p/macvim/**
2. Open Vimtutor
   a. Linux - `$ vimtutor`
   b. Mac - `$ vim /usr/share/vim/vim74/tutor/tutor`
   c. Windows - run gVim, then open `C:\Program Files\Vim\vim74\tutor\tutor`

Thanks for coming, and happy Vimming!