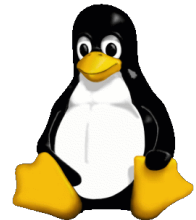


How The Command-Line Can *Empower* You

```
[13:05:18 ~]
[13:06:04 ~]
[13:06:05 ~]
[13:06:07 ~] curl 'https://en.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exintro&explaintext&redirects=1&titles=empowerment' | gron | rg \\.
extract | cut -c 301-357
It is the process of becoming stronger and more confident
[13:07:16 ~]
[13:07:19 ~] rg command-line def.txt
1:A command-line interface or command language interpreter (CLI),
2:also known as command-line user interface, console user interface[1]
[13:07:23 ~]
```

Eric Fung • @gnufmuffin



- ▶  /remind me Lunch for Gemma at noon
- ▶  inspect(document.body);
- ▶  efung@host:~\$ ls -l
- ▶  C:\>dir /w

Command-line interface: a way of interacting with a computer by giving it commands using text

But...  why would I?



Image credit:  burst.shopify.com



Why should I use the command line?

- ▶ Gets you results more **quickly**
- ▶ Lets you **control** the output
- ▶ Be less reliant on **others**
- ▶ Makes you more **versatile**
- ▶ Gives you **transferable** skills



What we'll cover

- ▶ Basic command-line setup
- ▶ Typing less
- ▶ Navigating folders
- ▶ Working with Finder and clipboard
- ▶ Searching files
- ▶ Integrating with remote APIs
- ▶ Dealing with JSON data
- ▶ Manipulating images

The basics

- Terminal
- Shell
- Package manager

a

b

c

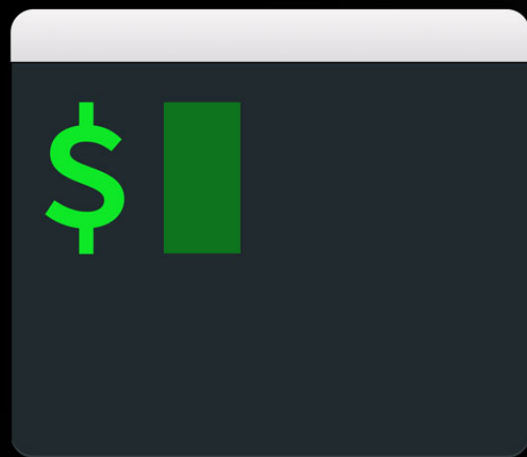
d

e

f

Terminal: iTerm2

- ▶ Extensive features and customization
- ▶ Favorites:
 - ▶ ⌘-click to open URLs and files
 - ▶ Highlights all search results
 - ▶ ⌘-; autocompletes from anything on screen



iTerm2

iTerm2 is a terminal emulator for macOS that does amazing things.



Shell: bash

- ▶ macOS default shell
- ▶ Widely used → lots of support
- ▶ May want to install newer version



Homebrew

The missing package manager for macOS

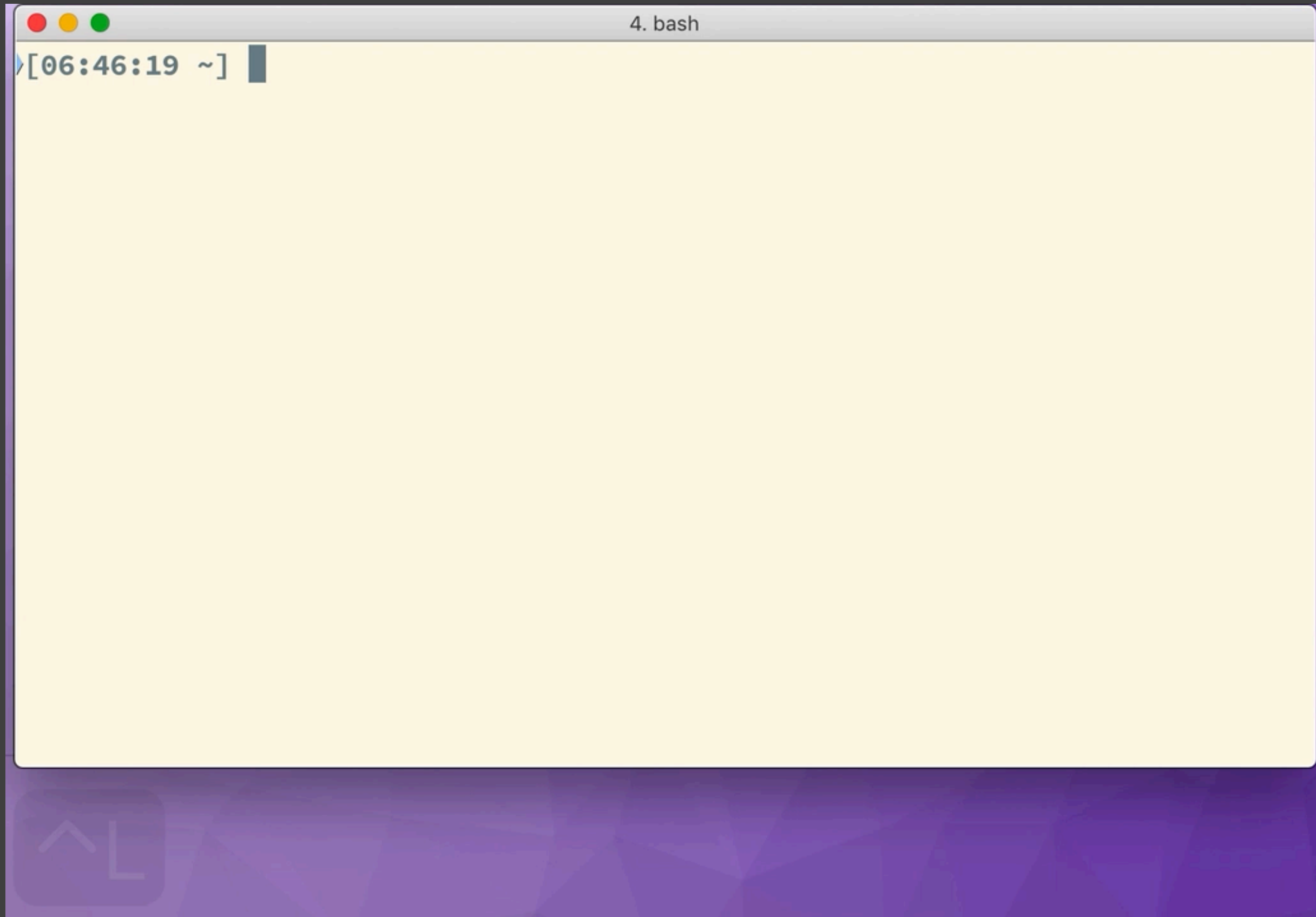
Package manager: Homebrew

- ▶ Many packages available (over 4000+)
- ▶ Software, fonts, plugins, GUI apps
- ▶ Self-healing

Typing less

Tab completion

- ▶ Autocompletes commands, options, files, directories
- ▶ Context-sensitive
- ▶ Installed with many Homebrew packages



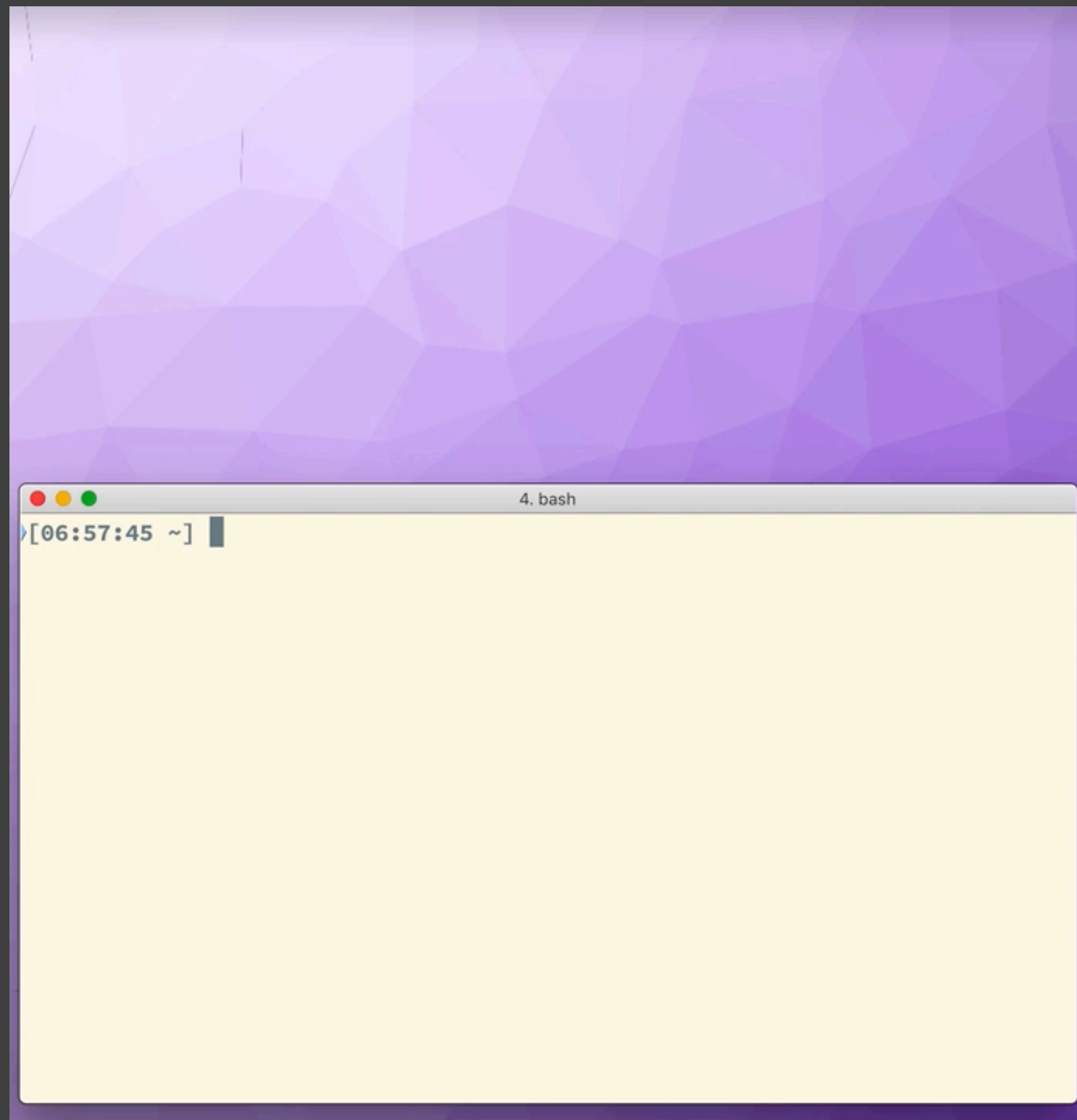
```
4. bash
[22:55:17 (source) gnufmuffin.com] git diff --git public/css/hyde.css public/css/hyde.css
index 860922a9..f3e95293 100644
--- public/css/hyde.css
+++ public/css/hyde.css
@@ -41,7 +41,7 @@ html {
 }
 @media (min-width: 58rem) {
   html {
-    font-size: 18px;
+    font-size: 20px;
   }
 }

[22:56:07 (source) gnufmuffin.com]
```

Working with Finder

From shell to Finder

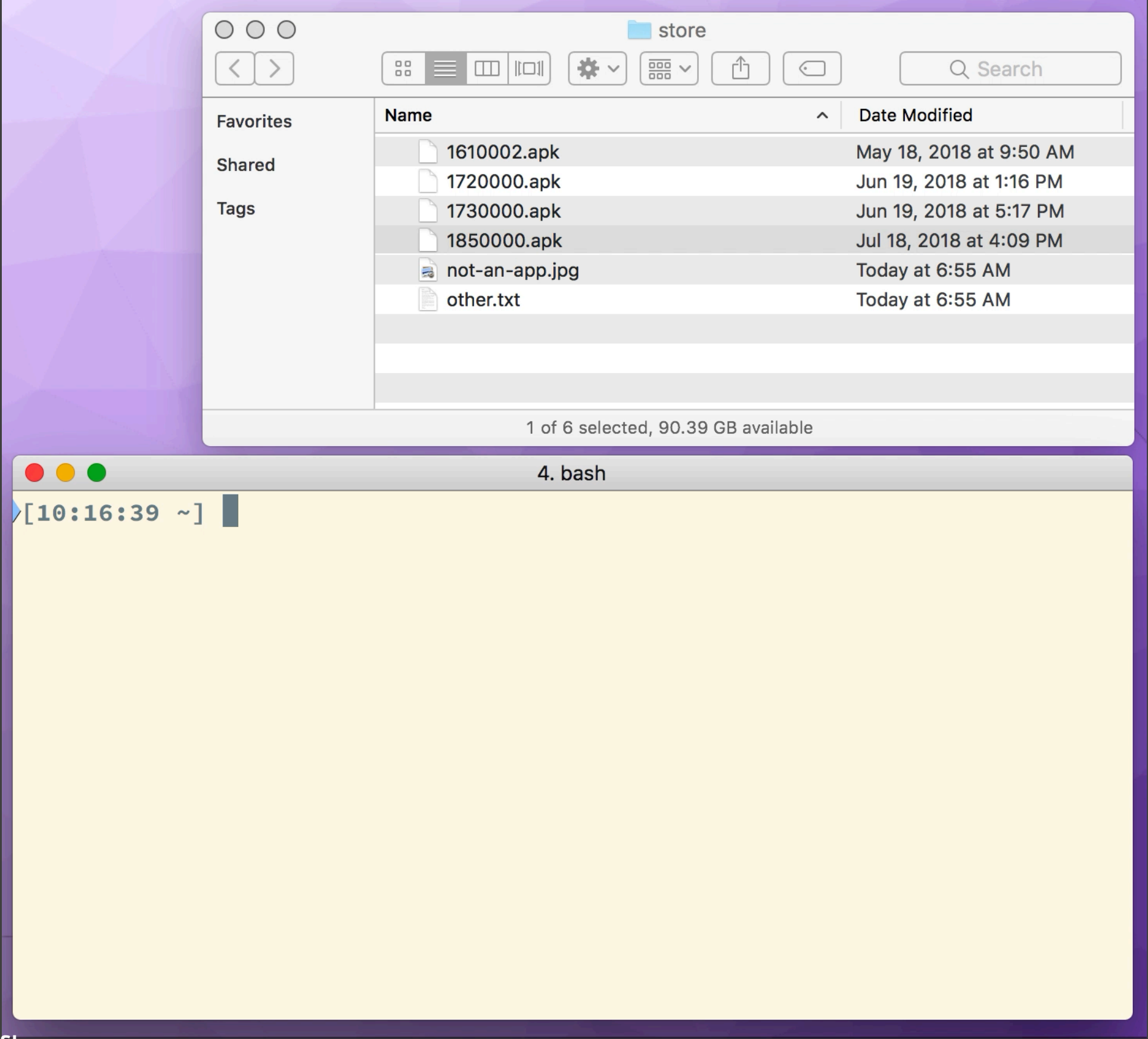
- ▶ `open <file>`
 - ▶ Equivalent to double-clicking on `<file>`
- ▶ `open .`
 - ▶ Opens Finder with current directory
- ▶ `open <url>`
 - ▶ Opens default browser to `url`



Demo: macOS open

From Finder to shell

- ▶ Drag-and-drop file(s) onto Terminal
 - ▶ Pastes their filename(s), space-separated



Demo: Drag files to shell → paste filename

Navigation



cd 101

- ▶ `cd app/src/main/res/layout`
- ▶ `cd ../../java`
- ▶ `cd -`
 - ▶ Go to previous directory



4. bash

▶ [11:00:56 ~] j



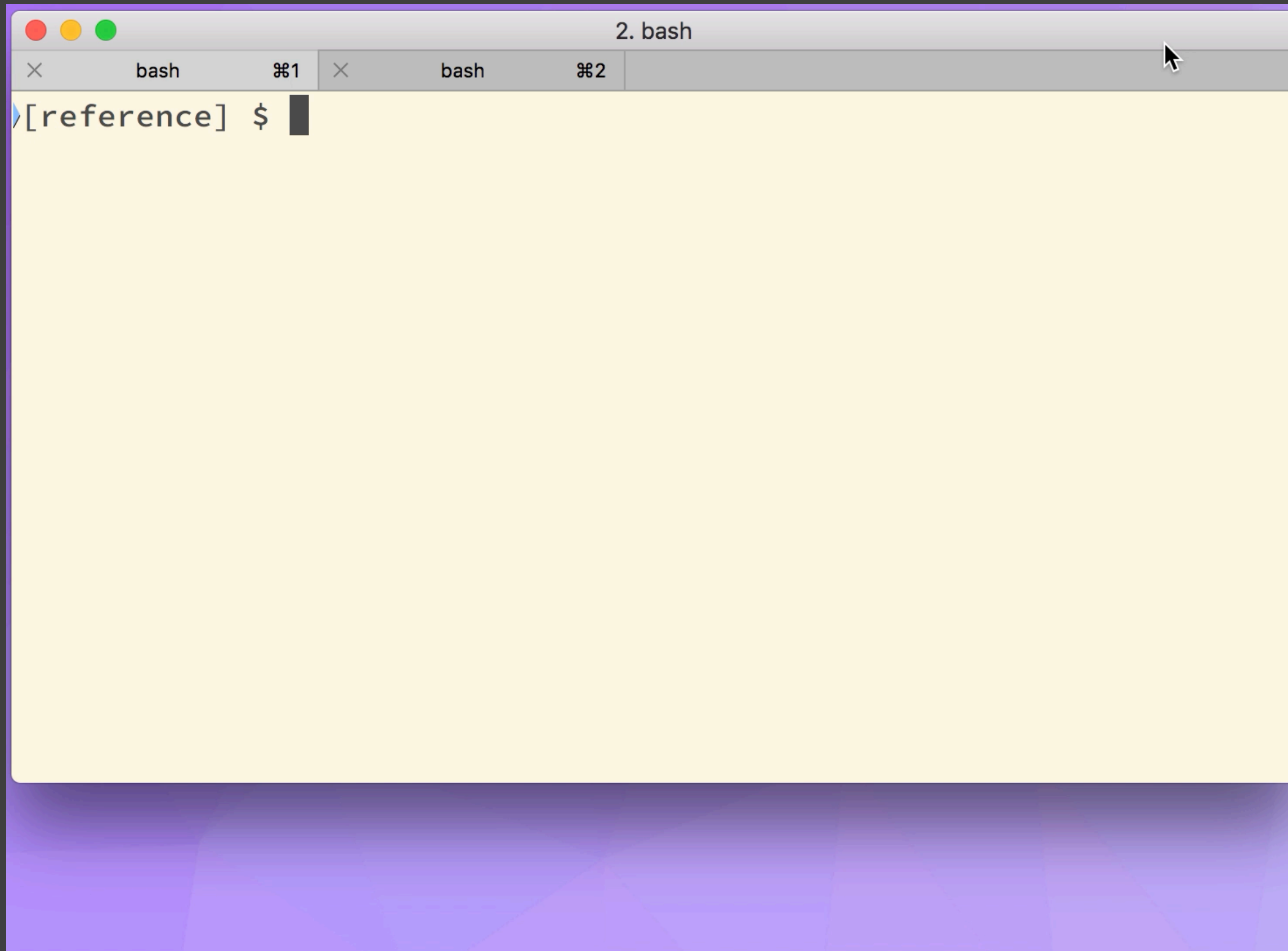
autojump

- ▶ Provide fast access to directories further away
- ▶ Selects directories based on usage
- ▶ Caveat: must be visited first

autojump

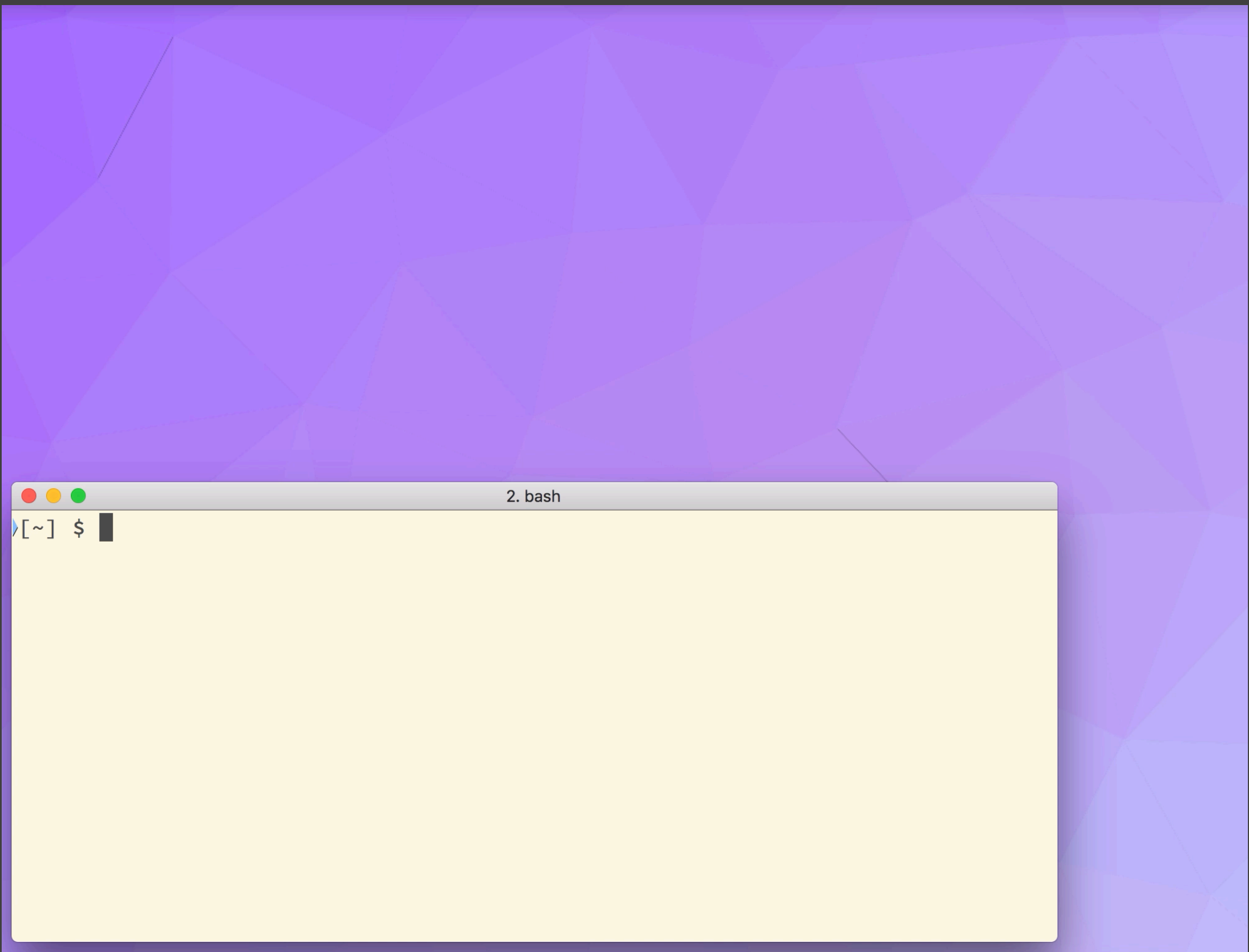
- ▶ `j name`
 - ▶ Go to most-used directory containing name
- ▶ `jc name`
 - ▶ Go to a subdirectory containing name
- ▶ `j name <Tab>`
 - ▶ Offers choices when there are multiple matches

Searching



ripgrep

- ▶ Supports `grep` options
- ▶ *Much* faster
- ▶ Knows about file types
- ▶ Respects revision control ignores
- ▶ Support for emoji 🐱



I/O



I/O 101

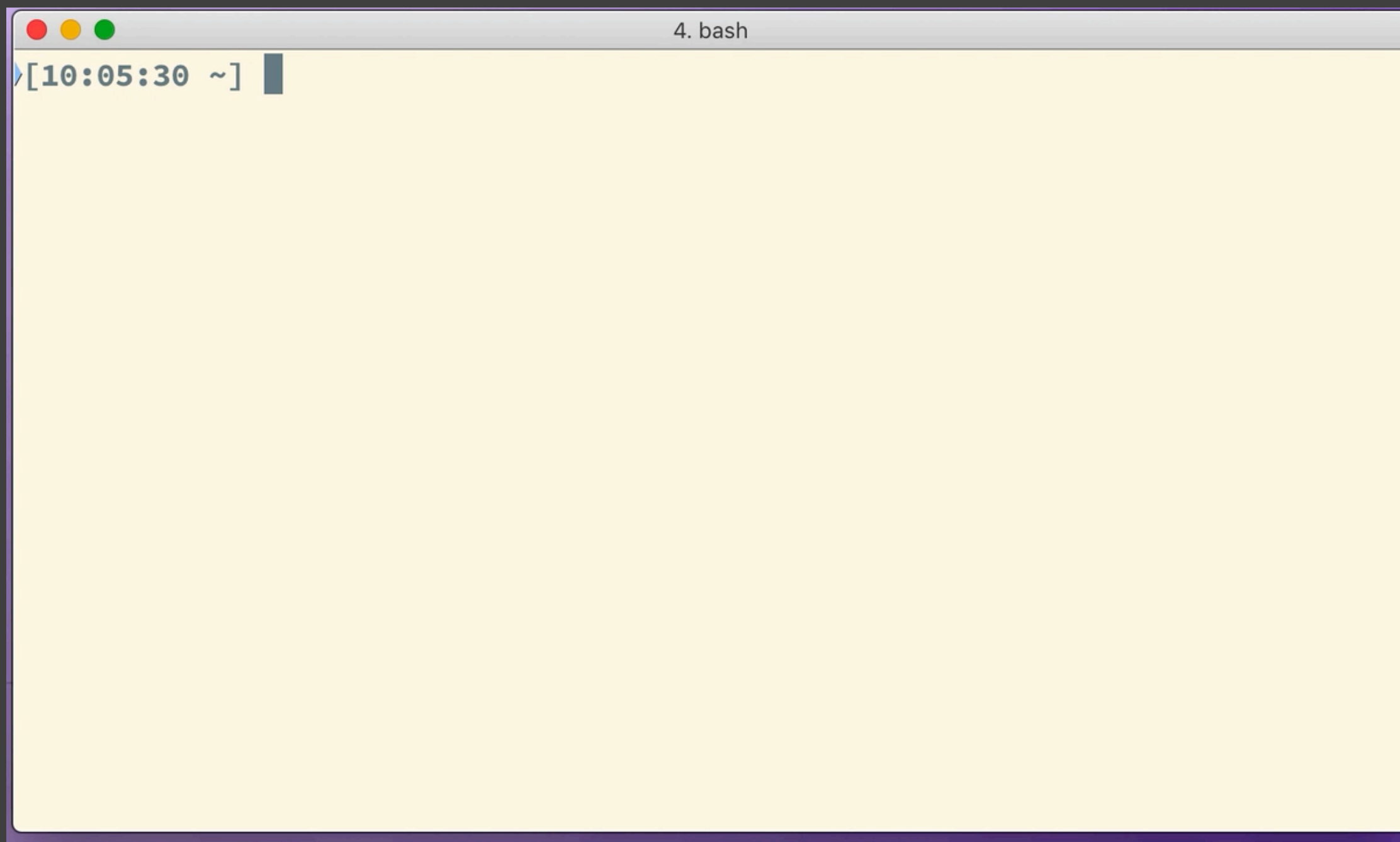
- ▶ Streams: `stdin` and `stdout`
- ▶ Redirection: `>` and `<`
- ▶ Pipelines: `|`

Working with Clipboard

- ▶ `pbcopy` (`⌘-C` for the shell)
 - ▶ Reads from `stdin`, copies to clipboard
- ▶ `pbpaste` (`⌘-V` for the shell)
 - ▶ Copies contents of clipboard, writes to `stdout`

The Web





curl



- ▶ Comprehensive client for working with Internet protocol transfers
 - ▶ *"the number of features will make your head spin!"*
- ▶ Supports HTTP, SMTP, IMAP, RTSP, and lots more
- ▶ Designed to be unattended

Debugging endpoints: --include

```
$ curl 'https://search.icons8.com/api/iconsets/v4/search' --include
HTTP/2 200
server: keycdn-engine
date: Thu, 09 Aug 2018 19:39:52 GMT
content-type: application/json; charset=utf-8
content-length: 96
cache-control: max-age=604800
expires: Thu, 16 Aug 2018 19:39:52 GMT
x-cache: HIT
x-shield: active
x-edge-location: usch
access-control-allow-origin: *
accept-ranges: bytes
```



POSTing data to an API

```
$ curl -X POST \
  --header "Content-Type: application/json" \
  https://jsonplaceholder.typicode.com/posts \
  --data-ascii @body
```

JSON



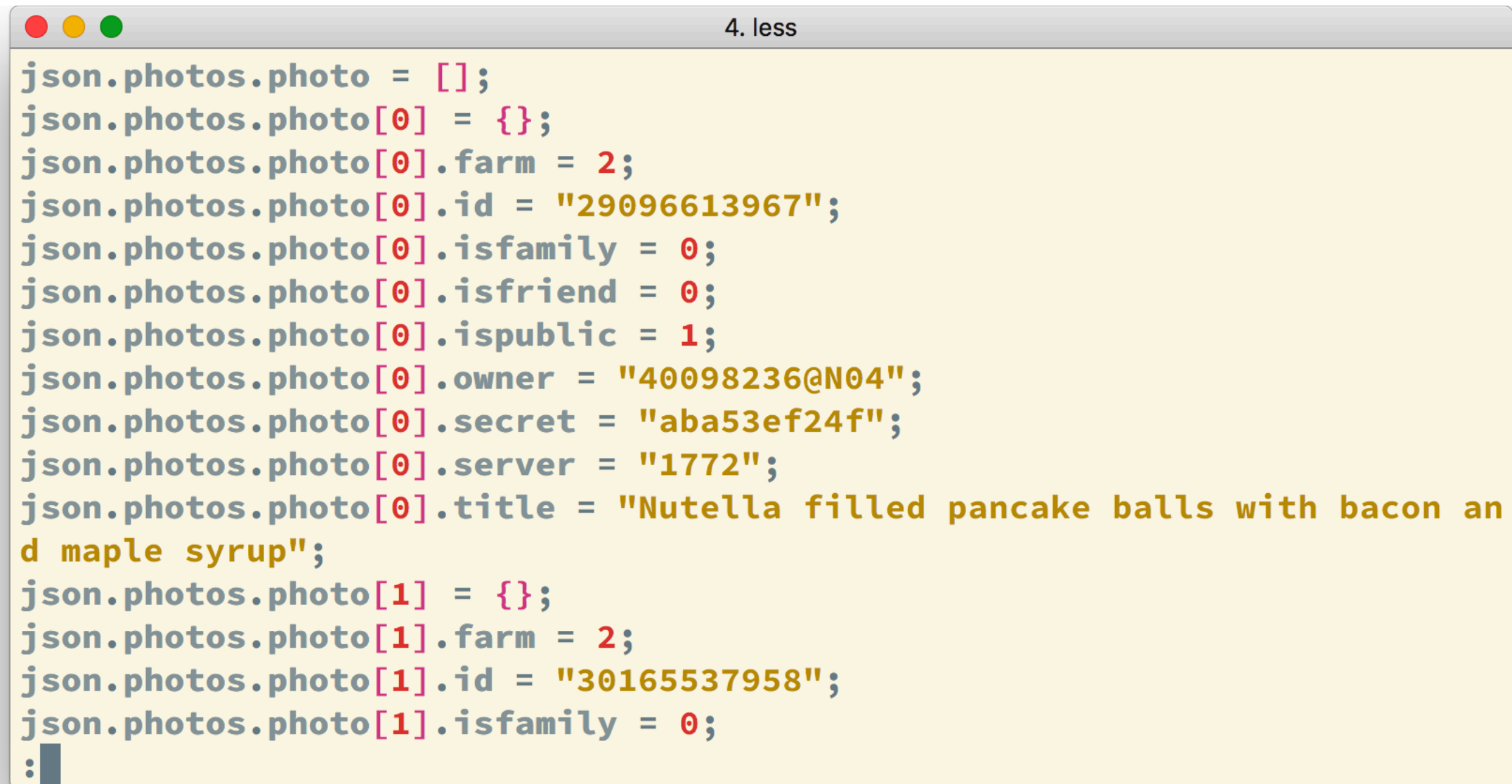
4. bash

```
[~] $ curl "https://api.flickr.com/services/rest/?method=flickr.photos.getRecent&api_key=${FLICKR_KEY}&per_page=8&format=json&nojsoncallback=1"
```

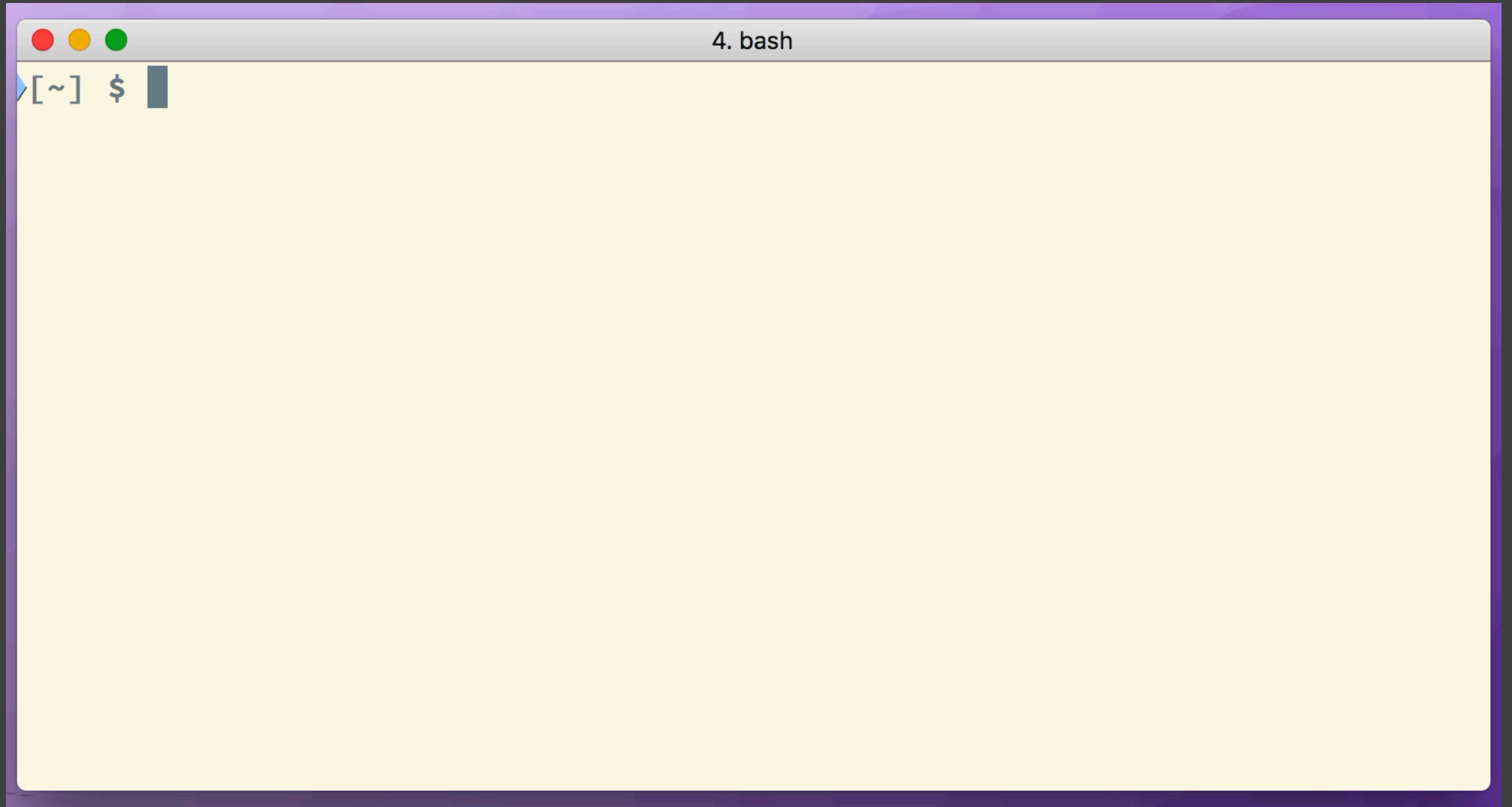
gron

- ▶ Turns JSON into line-oriented data
- ▶ Then, search with tools like ripgrep
- ▶ Identify individual array elements

Example gron output



```
4. less
json.photos.photo = [];
json.photos.photo[0] = {};
json.photos.photo[0].farm = 2;
json.photos.photo[0].id = "29096613967";
json.photos.photo[0].isfamily = 0;
json.photos.photo[0].isfriend = 0;
json.photos.photo[0].ispublic = 1;
json.photos.photo[0].owner = "40098236@N04";
json.photos.photo[0].secret = "aba53ef24f";
json.photos.photo[0].server = "1772";
json.photos.photo[0].title = "Nutella filled pancake balls with bacon and maple syrup";
json.photos.photo[1] = {};
json.photos.photo[1].farm = 2;
json.photos.photo[1].id = "30165537958";
json.photos.photo[1].isfamily = 0;
:
```



jq

- ▶ Powerful, flexible processor
- ▶ Transform data you have, to data you *want*
 - ▶ JSON, CSV, URI, plain text
- ▶ Concise syntax
- ▶ Handles **JSON Lines**, e.g. log files

Example JSON data

```
{  
  "photos": {  
    "page": 1,  
    "pages": 125,  
    "total": 100,  
    "photo": [  
      {  
        "id": "2348183",  
        "title": "Fuzzy cat",  
        "url": "https://example.com/1.jpg"  
      }, ...  
    ]  
  }  
}
```

jq example

```
$ jq -r '.photos.photo' flickr.json
```

```
[
  {
    "id": "2348183",
    "title": "Fuzzy cat",
    "url": "https://example.com/1.jpg"
  },
  {
    "id": "7937001",
    "title": "Good boi",
    "url": "https://example.com/2.jpg"
  },
  ...
]
```

jq example

```
$ jq -r '.photos.photo | .[]' flickr.json
```

```
{  
  "id": "2348183",  
  "title": "Fuzzy cat",  
  "url": "https://example.com/1.jpg"  
}  
{  
  "id": "7937001",  
  "title": "Good boi",  
  "url": "https://example.com/2.jpg"  
}  
...
```

Extract values from array of objects

```
$ jq -r '.photos.photo | .[] | .title' flickr.json
```

AJH-20180814-467-Edit.jpg

East Garston series: quintessential English countryside

IMG_20180723_215912_323

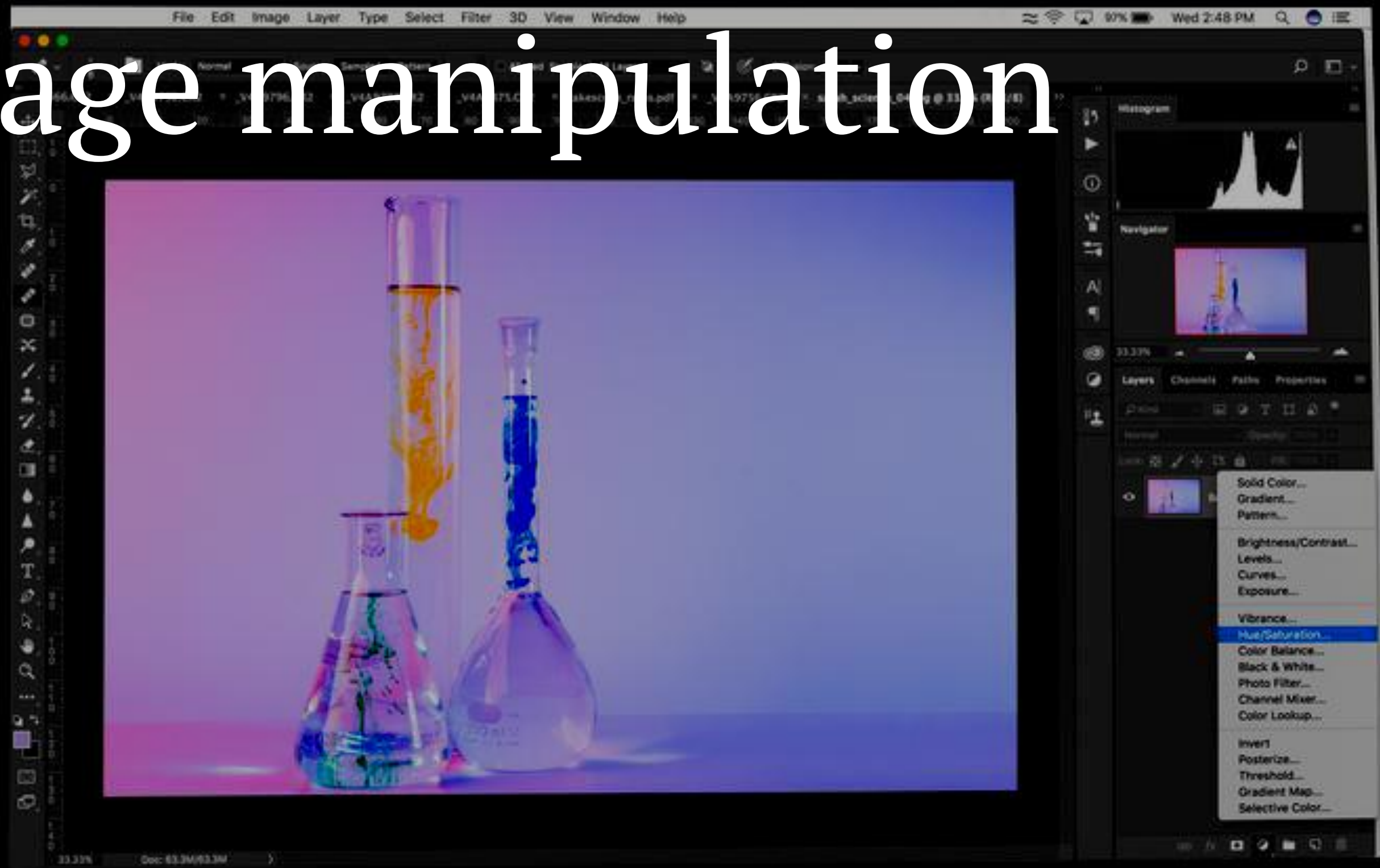
P1030365.jpg

Frankrijk 2018

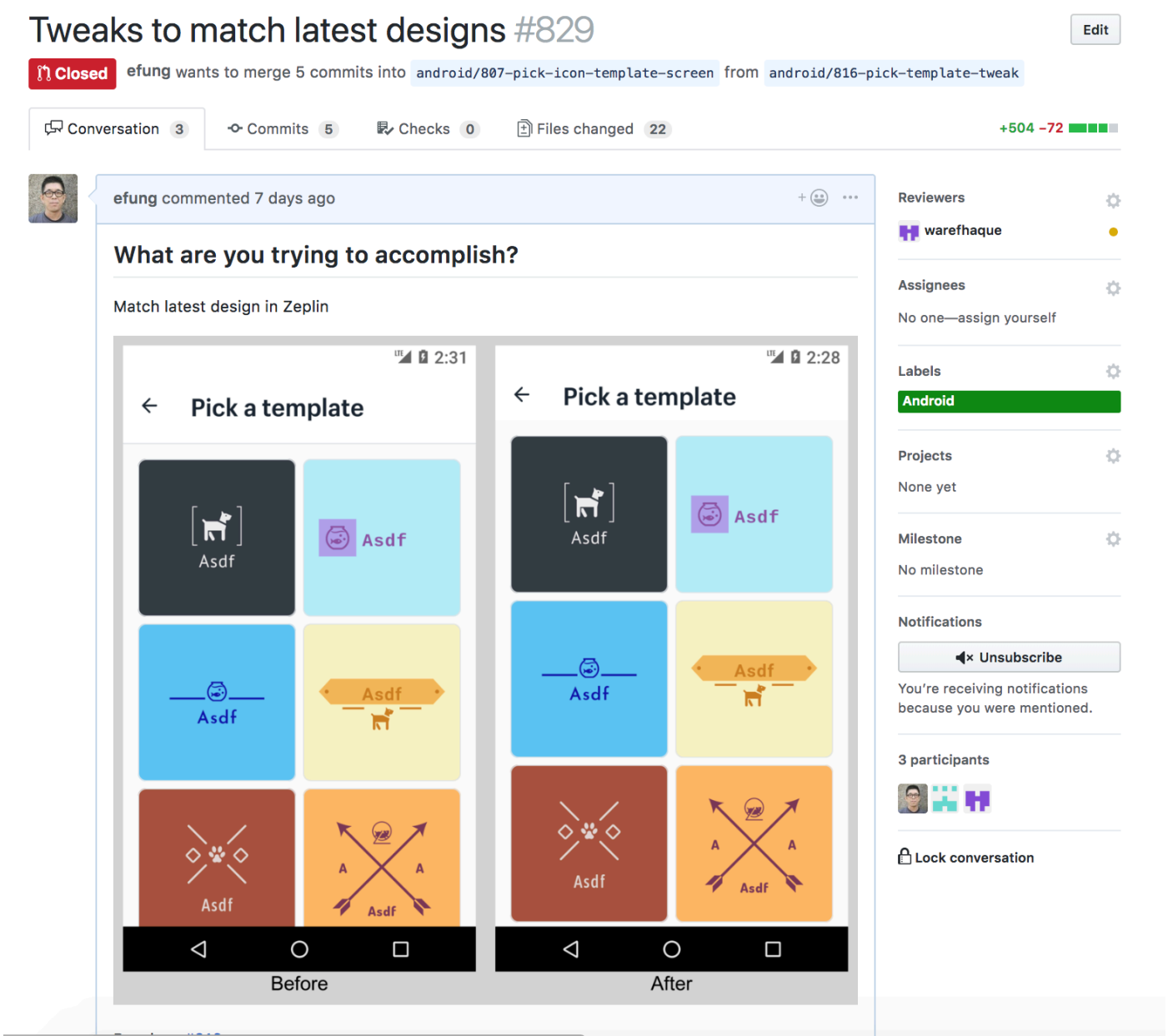
Shirts on Sale (+6 03 6143 5225)Contact Us 03 6143 5225 <https://t.co/dBTffHWW99>

Tratamiento Alternativo Para El Colon Irritable

Image manipulation



Compositing images

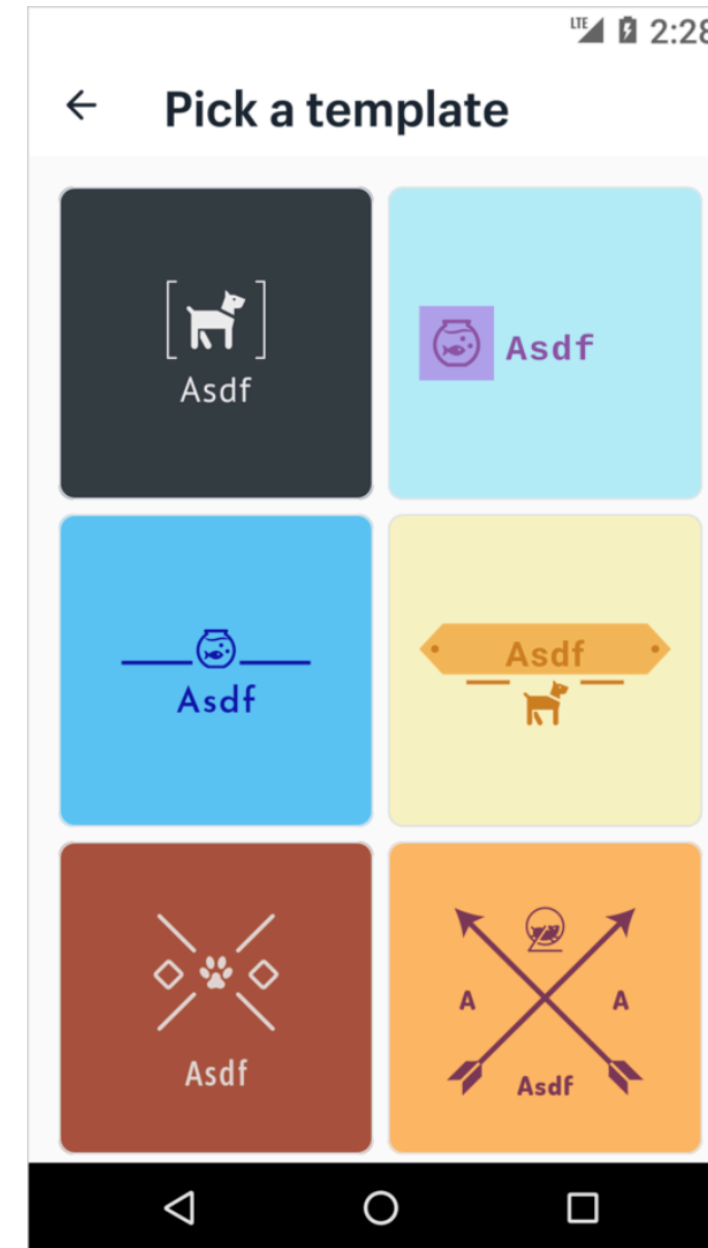
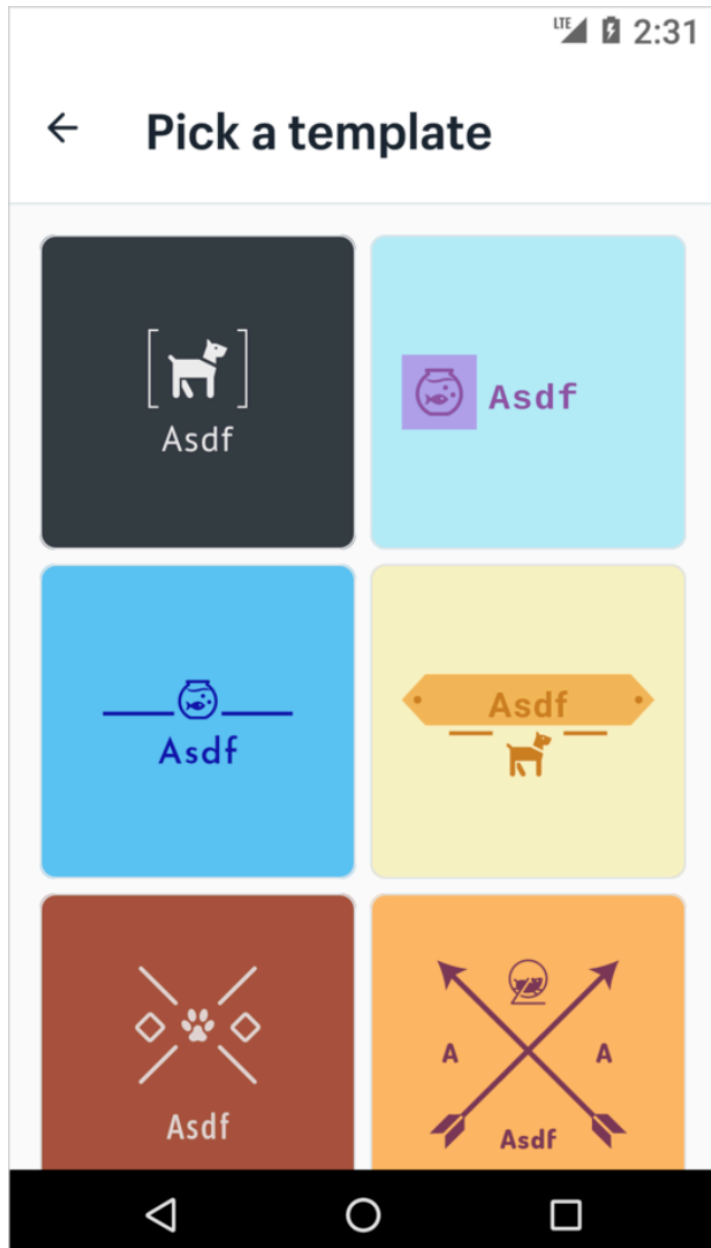


ImageMagick

- ▶ Suite of command-line utilities
- ▶ Create, edit, compose, or convert bitmaps
- ▶ Read/write 200 image formats
- ▶ Handles really large images



Example: Composite images



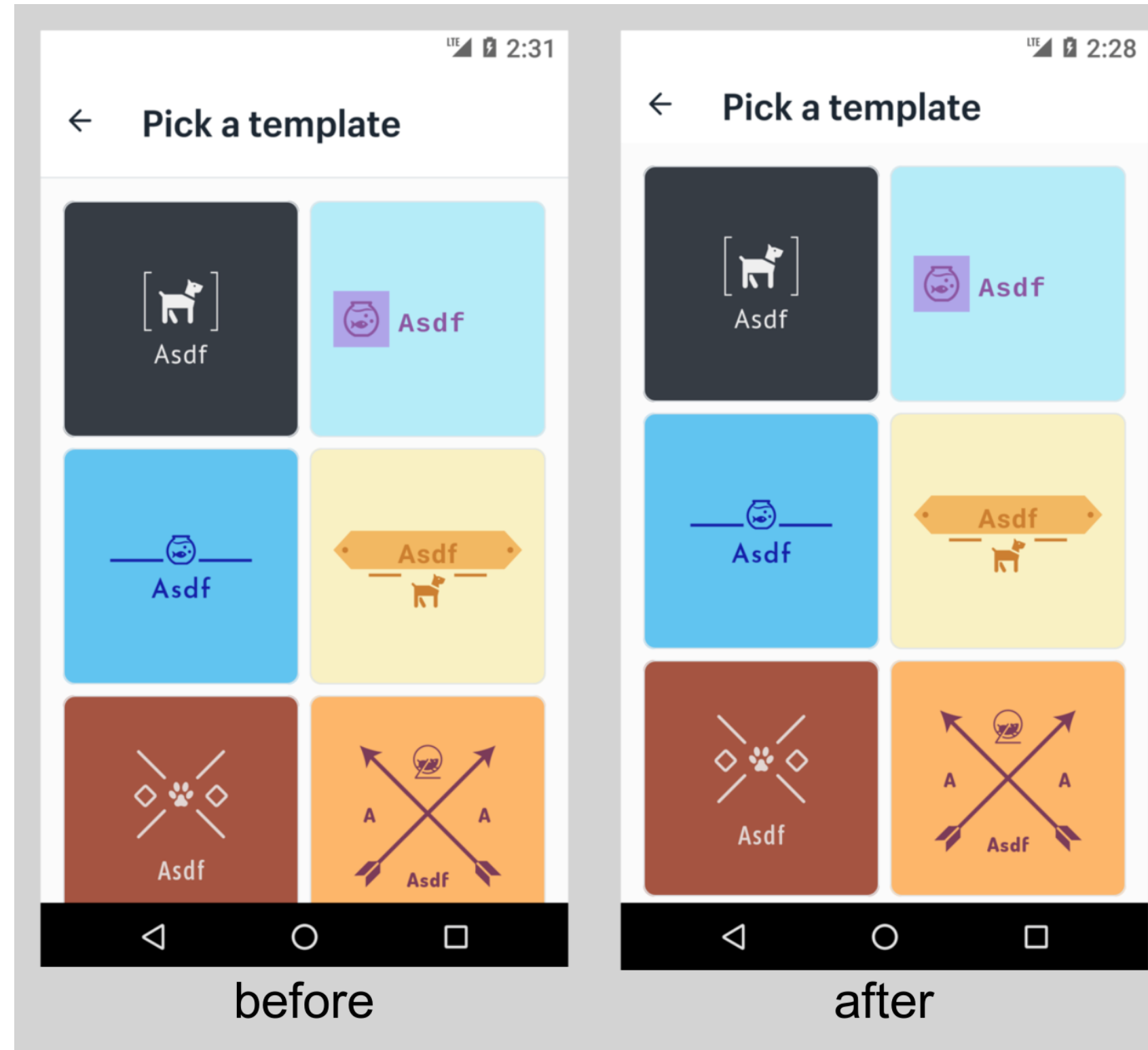
bash function using IM's montage

```
side_by_side() {  
    montage -background lightgray -pointsize 60 \  
        -label '%t' \  
        $* \  
        -tile x1 -geometry '+30+30' \  
        combined.png  
}
```

- ▶ `$*` is all the function's arguments
- ▶ `-label '%t'` is IM syntax defining label from each argument's base filename

Usage

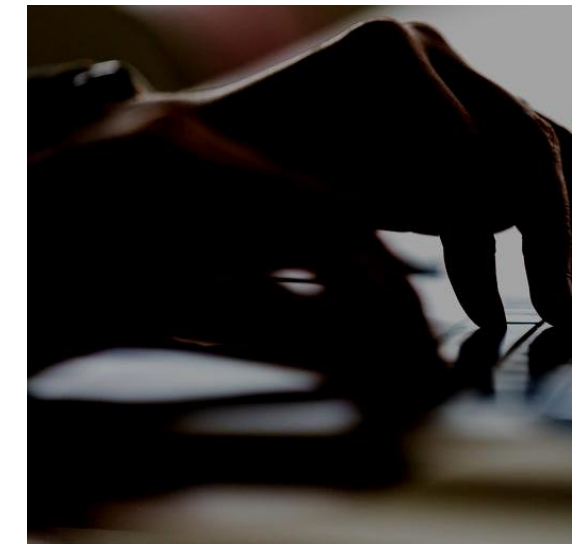
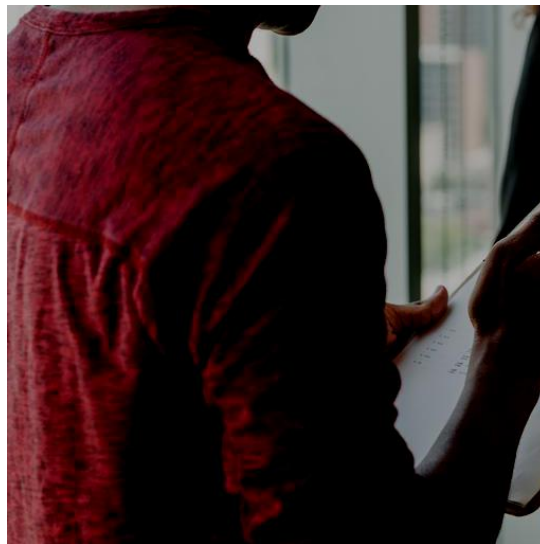
```
$ side_by_side before.png after.png
```



Example: Batch processing



Example: Batch processing



bash loop with IM's convert

```
for f in image1.jpg image2.jpg image3.jpg; do
    convert -brightness-contrast -30 \
           -gravity SouthWest \
           -crop 512x512+0+0 \
           $f \
           `basename $f .jpg`.cropped.jpg
done
```

bash loop with IM's convert

```
for f in image1.jpg image2.jpg image3.jpg; do
    convert -brightness-contrast -30 \
           -gravity SouthWest \
           -crop 512x512+0+0 \
           $f \
           `basename $f .jpg`.cropped.jpg
done
```

- ▶ for loop assigns each filename to variable f
- ▶ gravity defines origin
- ▶ basename removes extension, backticks inline result



Summary

What we've learned

- ▶ Command-line utilities are powerful tools
- ▶ Can be combined in different ways to get stuff done
- ▶ They can make you more productive
- ▶ Excellent additions to your developer toolbelt

End

- ▶ **Twitter**

[@gnufmuffin](#) • [@ShopifyEng](#)

- ▶ **Webinar**

[Recording on Zoom](#) ← with demo videos

- ▶ **Slides**

speakerdeck.com/efung ← static deck

Appendix: How to install (1/2)

- ▶ **iTerm2**: full-featured terminal
- ▶ **Homebrew**: package manager
- ▶ bash completion: completions for macOS builtins
 - ▶ 🍺¹ bash-completion
- ▶ **autojump** (j): navigate directories faster
 - ▶ 🍺 autojump
- ▶ **ripgrep** (rg): fast, intelligent search tool
 - ▶ 🍺 ripgrep

¹ 🍺 denotes Homebrew package name

Appendix: How to install (2/2)

- ▶ **curl**: transfer data with URLs
 - ▶ 🍺 curl
 - ▶ everything curl **book**
- ▶ **gron**: make JSON line-oriented and searchable
 - ▶ 🍺 gron
- ▶ **jq**: transform and query JSON
 - ▶ 🍺 jq
 - ▶ **jq manual**
 - ▶ **jq cookbook**
- ▶ **ImageMagick**: utilities for image manipulation
 - ▶ 🍺 imagemagick
 - ▶ **usage examples**