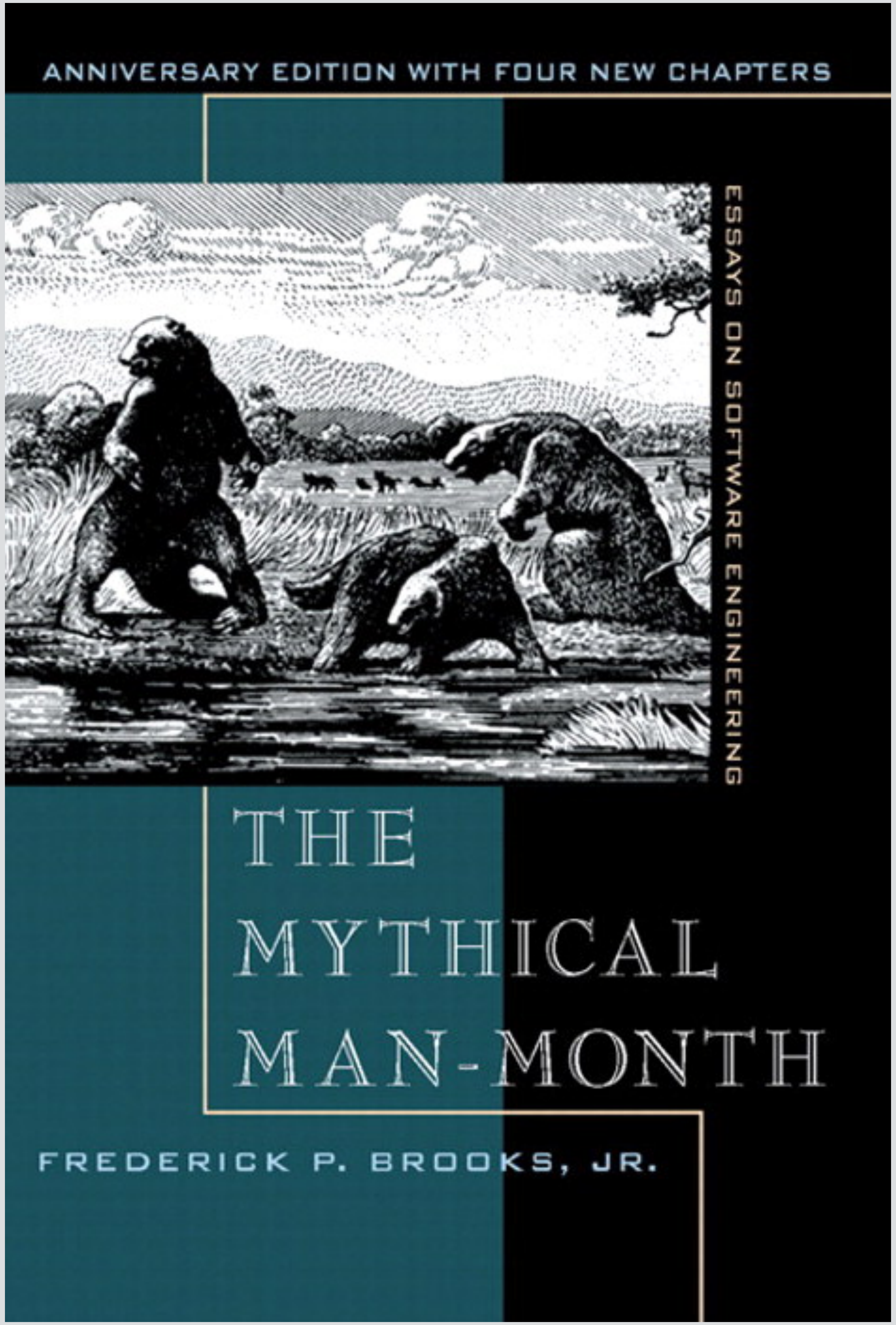
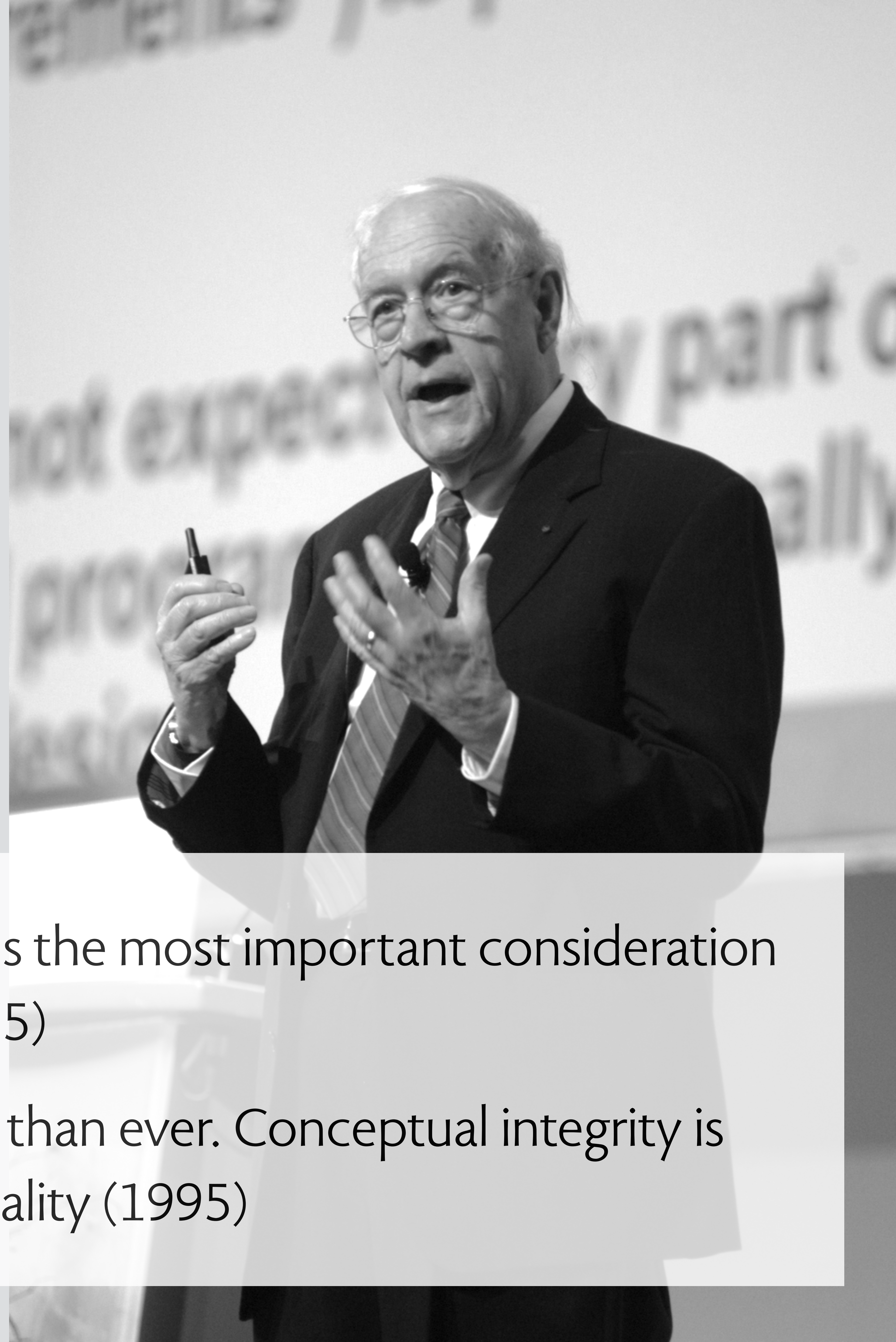
concept design

Daniel Jackson · MIT CSAIL · ER Online Summer Seminars · August 5, 2020

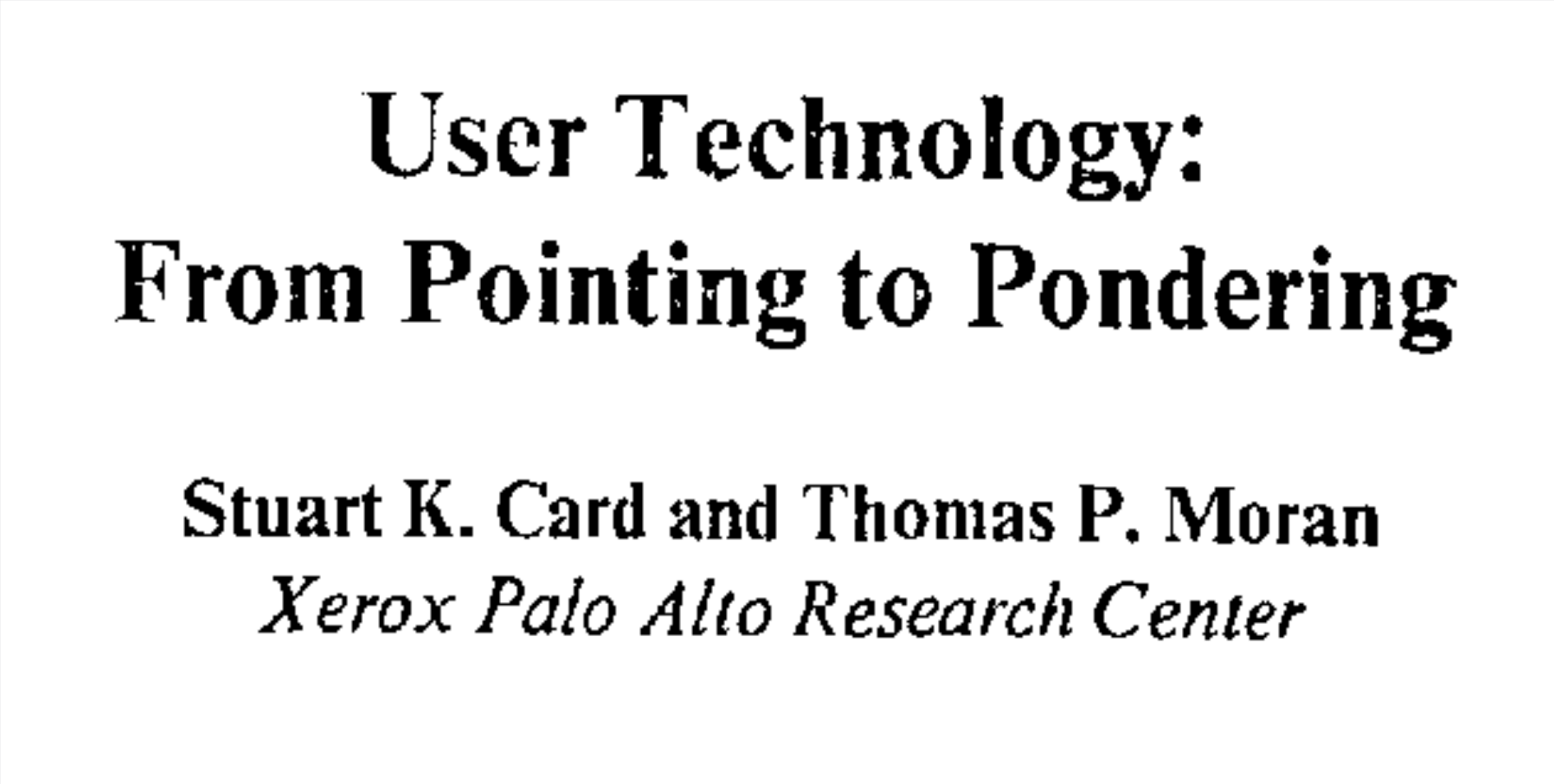
desperately seeking concepts


ANNIVERSARY EDITION WITH FOUR NEW CHAPTERS
ESSAYS ON SOFTWARE ENGINEERING
THE MYTHICAL MAN-MONTH
FREDERICK P. BROOKS, JR.



Conceptual integrity is the most important consideration in system design (1975)

I am more convinced than ever. Conceptual integrity is central to product quality (1995)

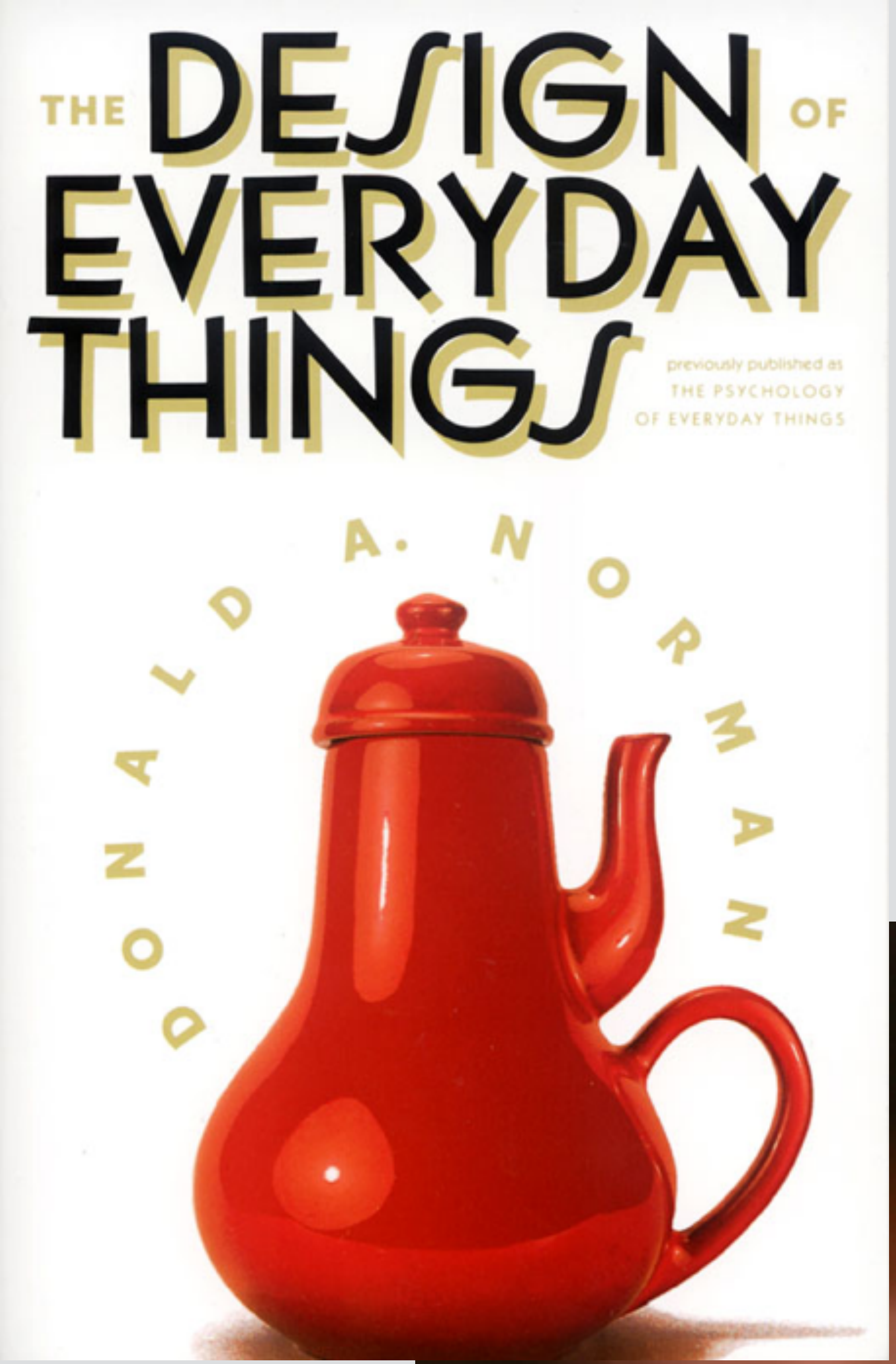


User Technology:
From Pointing to Pondering

Stuart K. Card and Thomas P. Moran
Xerox Palo Alto Research Center

1986

> It is clear that users attempt to make sense—by building mental models—of the behavior of a system as they use it. If a simple model is not explicitly or implicitly provided, users formulate their own myths about how the system works… [I]f the user is to understand the system, the system has to be designed with an explicit conceptual model that is easy enough for the user to learn. We call this the intended user's model, because it is the model the designer intends the user to learn.


THE DESIGN OF EVERYDAY THINGS
previously published as THE PSYCHOLOGY OF EVERYDAY THINGS
DONALD A. NORMAN

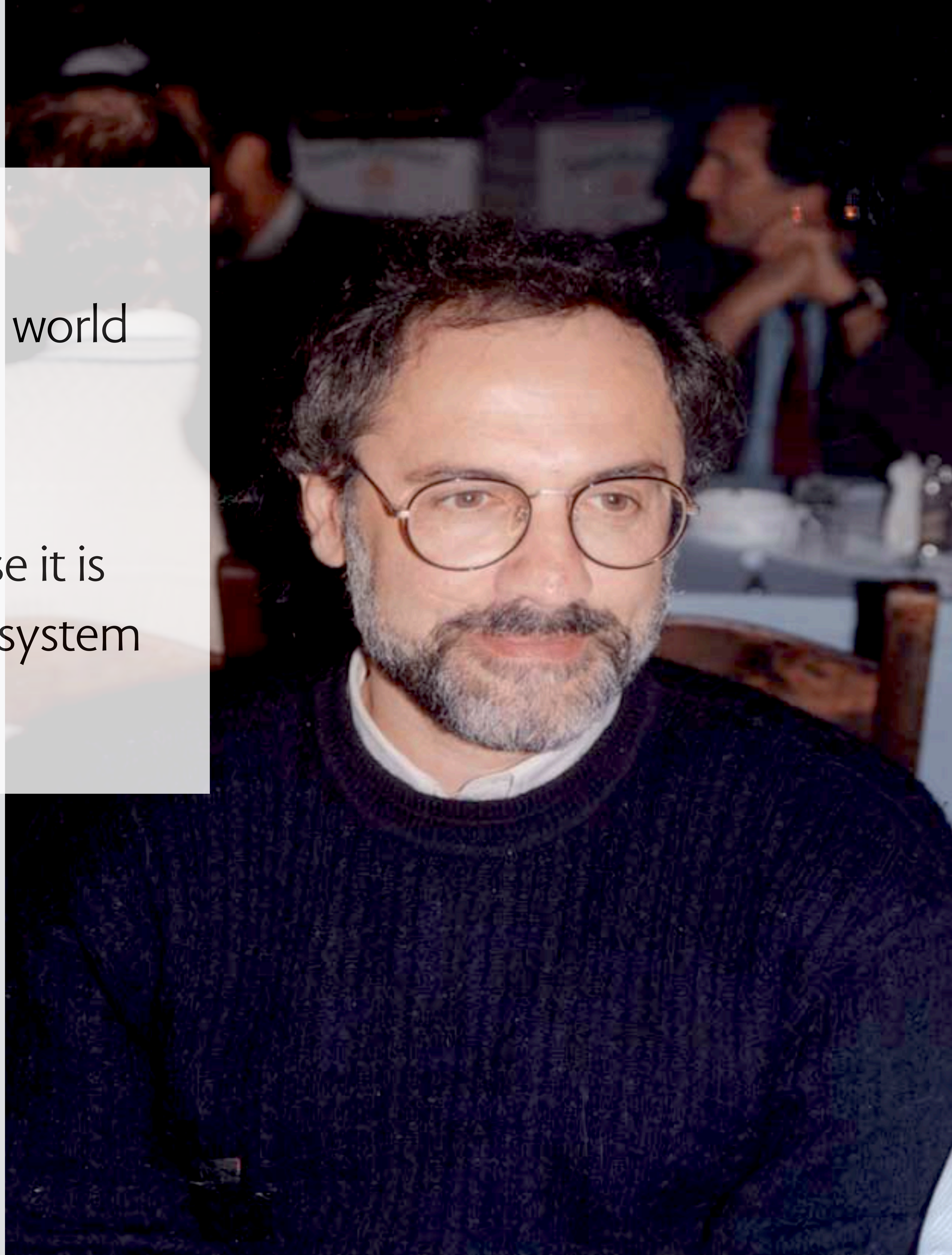
1988

When the designers fail to provide a conceptual model, we will be forced to make up our own, and the ones we make up are apt to be wrong. Conceptual models are critical to good design.

Donald Norman

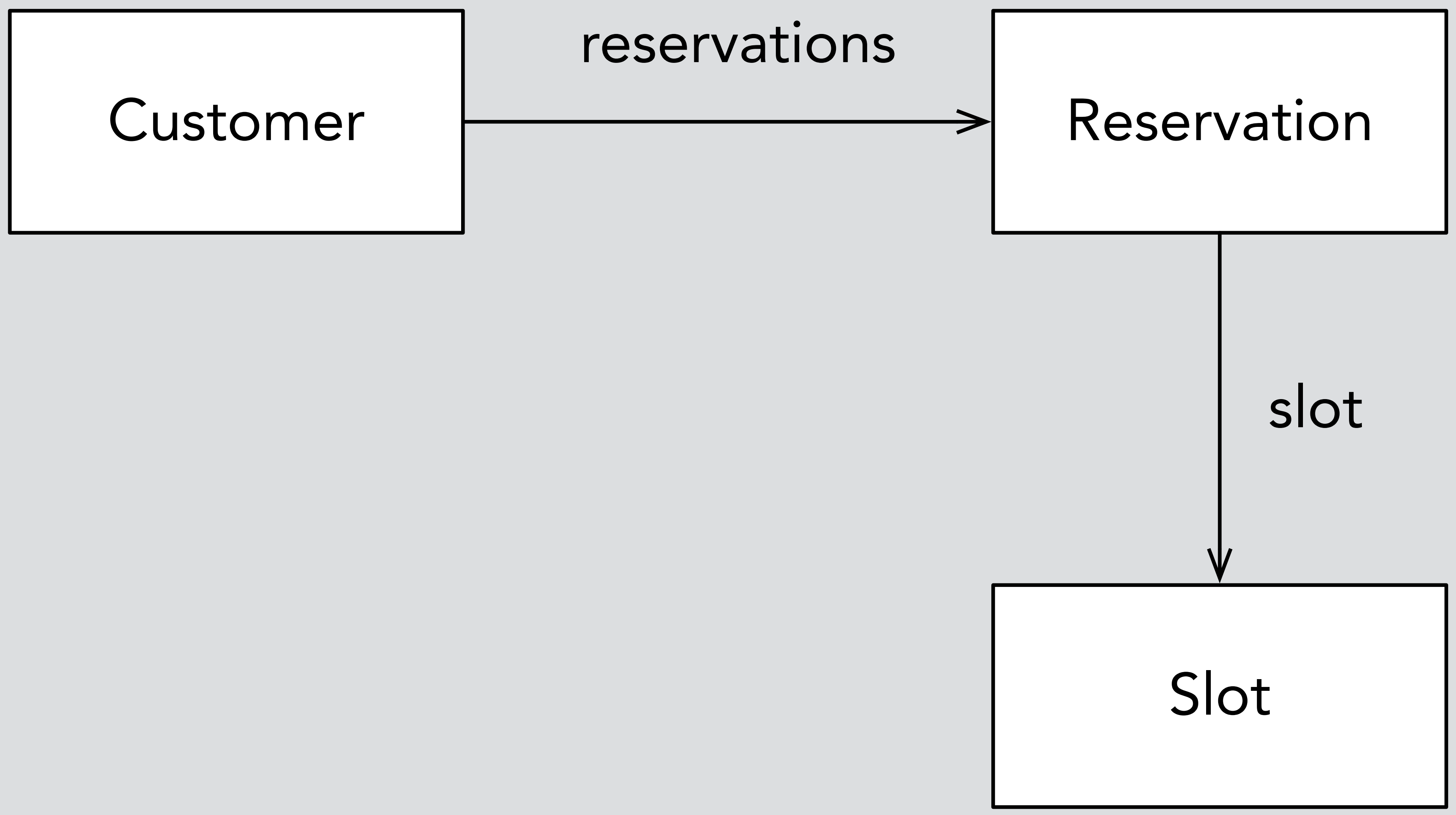
> Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication…
>
> We are interested in conceptual modelling because it is useful in rationalizing and supporting information system development.

John Mylopoulos. Conceptual modeling and Telos, 1992

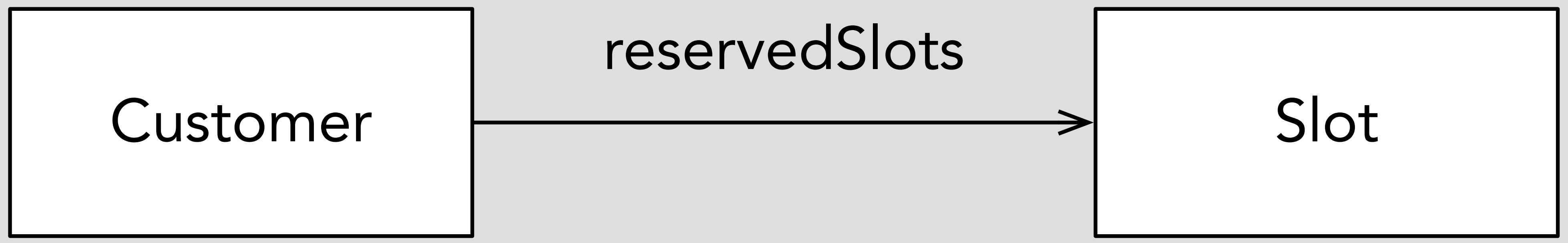


where’s the concept?

3 entities: how many concepts?



is the relation a concept?



> The conceptual modelling community not only has no clear, general agreement on what its models model, it also has no clear picture of what the available options and their implications are. **One common claim is that models represent concepts, but there is no clear articulation of what the concepts are**.

Chris Partridge, Cesar Gonzalez-Perez and Brian Henderson-Sellers. Are Conceptual Models Concept Models? 2013

why it matters

modularity is the essence of design
provides separation of concerns & structure for reuse

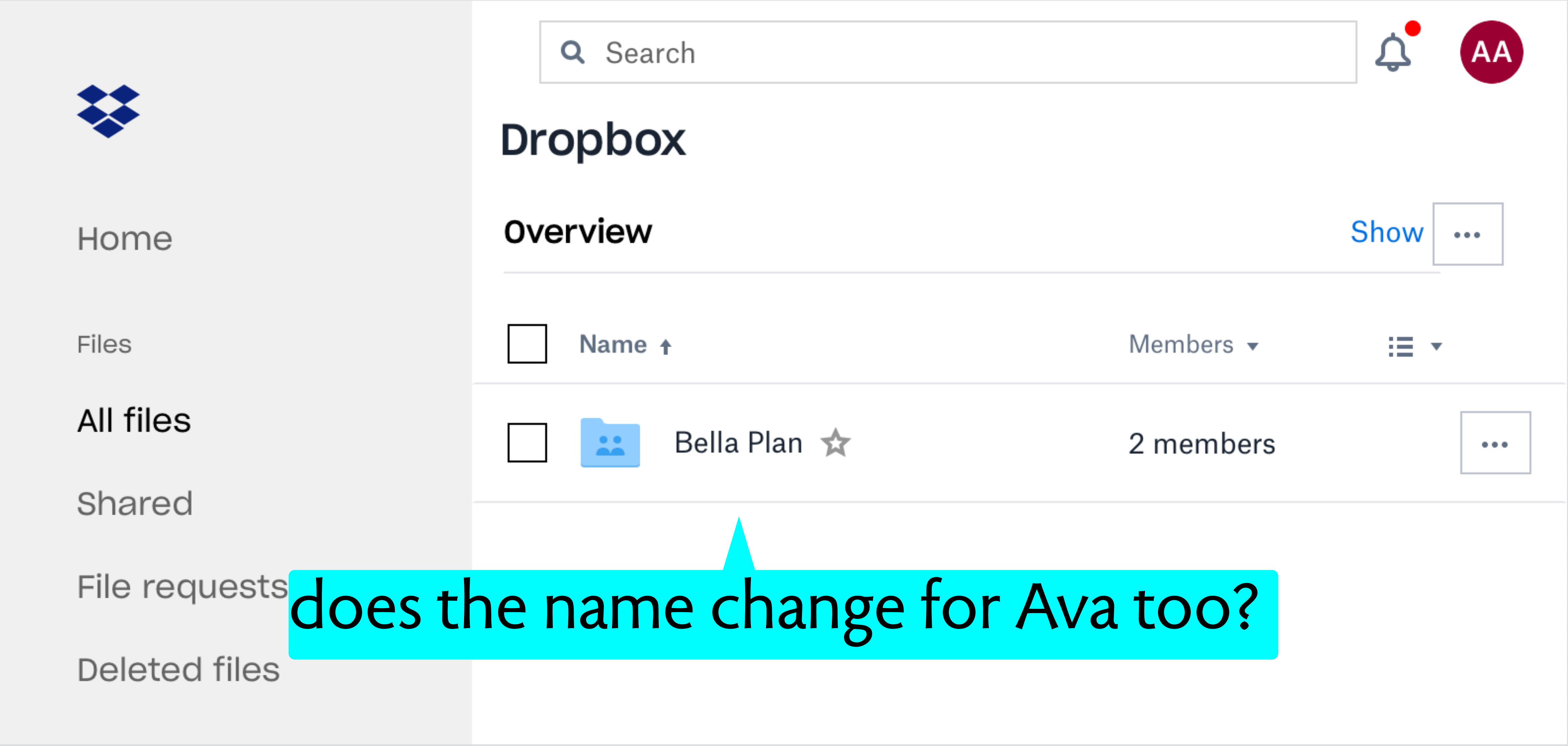
without concepts, what are conceptual models?
like formal models of a domain in Alloy (or Z, or Statecharts….)

we have an intuition that concepts are distinct
restaurant reservation app based on concept of “reservation”?

dropbox delusions

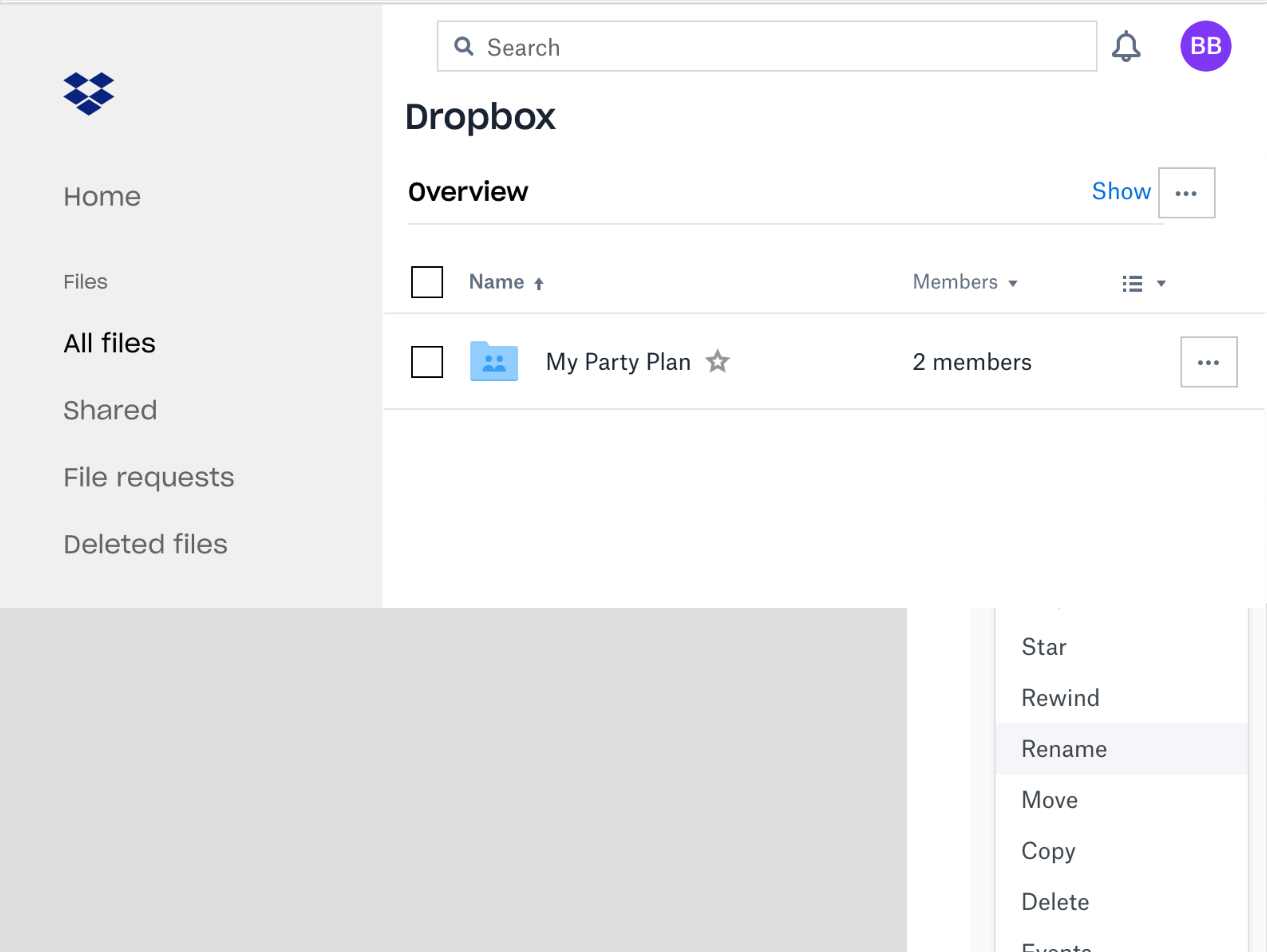
Ava is a party planner

Bella is having a party



Name	Members
Bella Plan ☆	2 members

does the name change for Ava too?



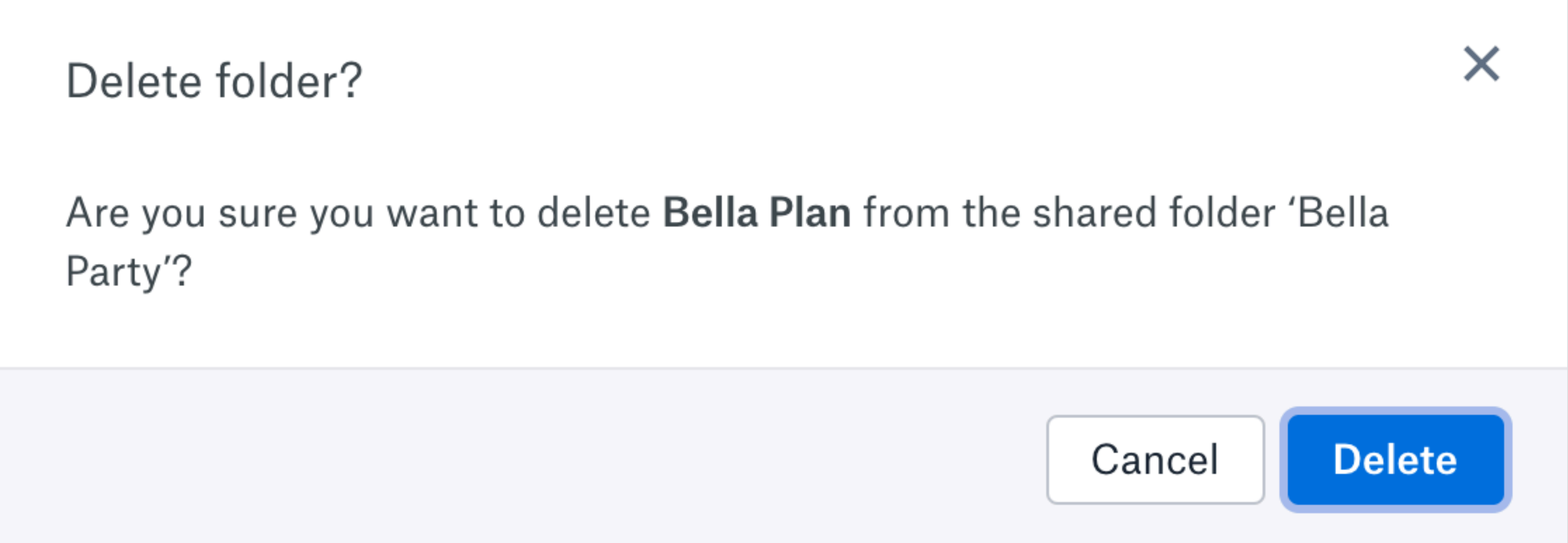
Name	Members
My Party Plan ☆	2 members

answer: it depends

if Ava just shares Bella Plan with Bella
and Bella renamed the folder, Ava sees no change

if Ava shares a folder Bella Party with Bella
containing the folder Bella Plan, and Bella renamed Bella Plan
then Ava does see the change

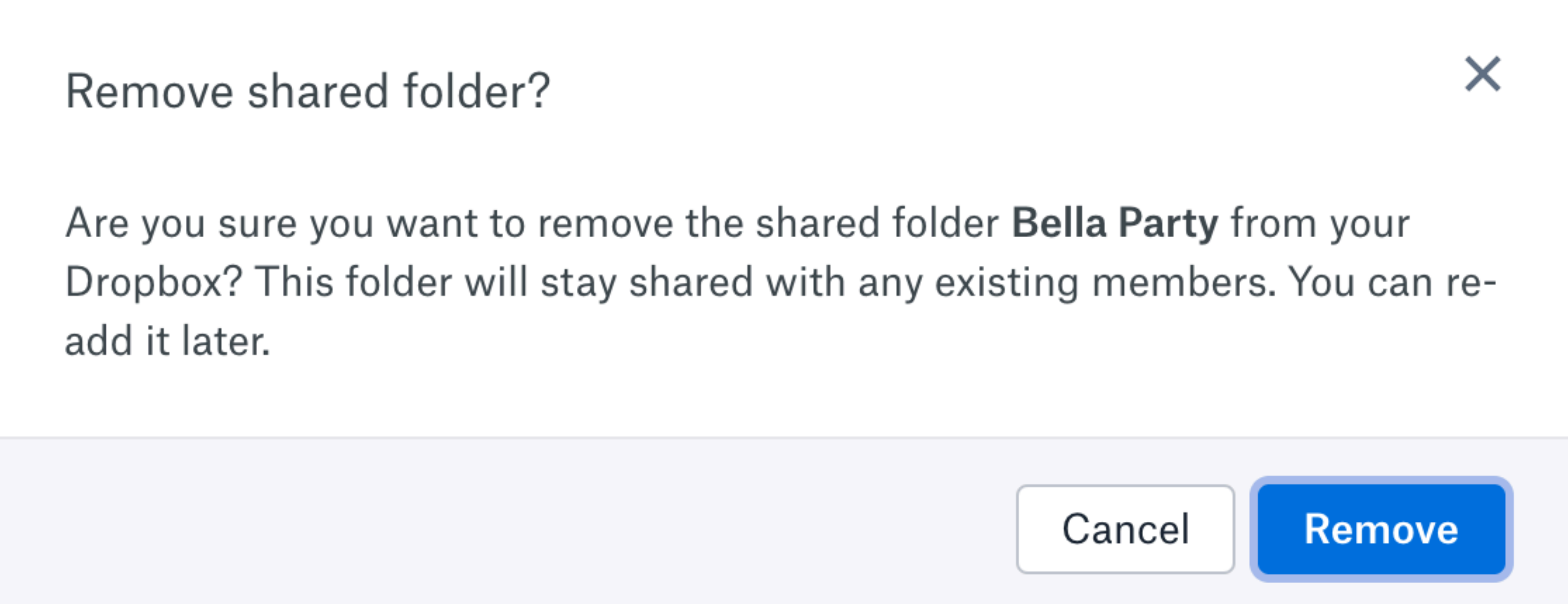
same two cases for deletion



Delete folder?

Are you sure you want to delete **Bella Plan** from the shared folder ‘Bella Party’?

Bella deletes Bella Plan from shared folder Bella Party

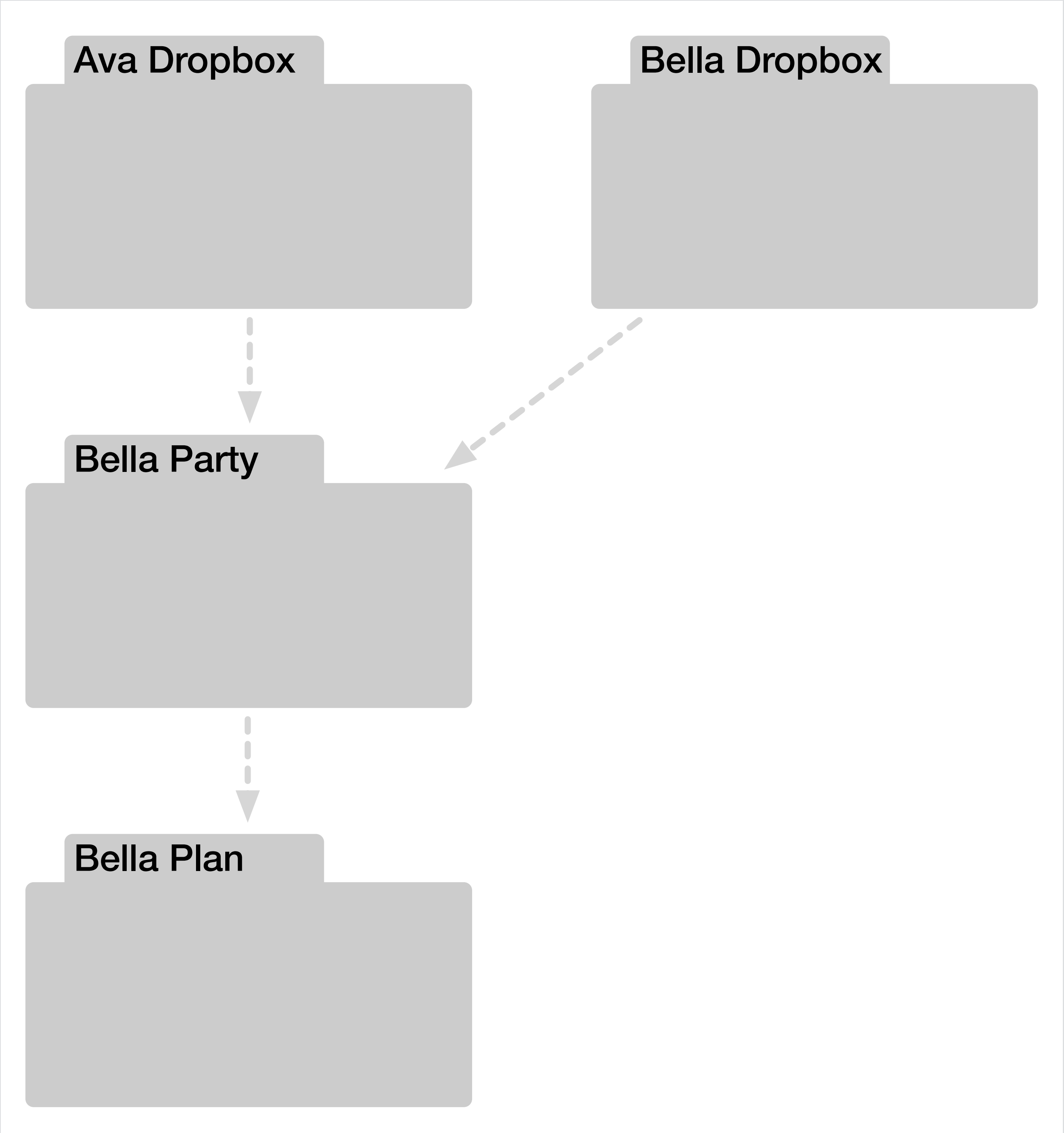


Remove shared folder?

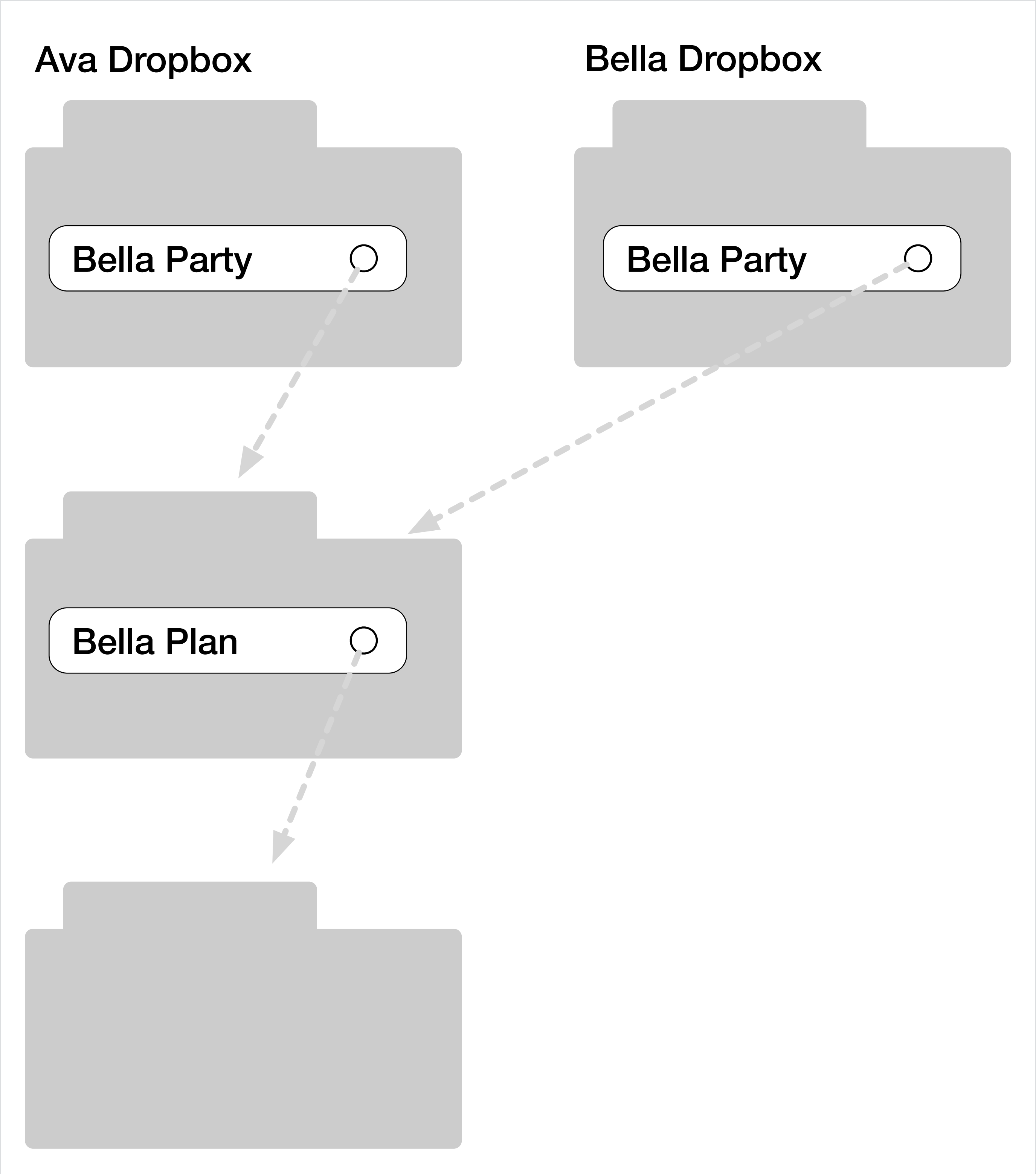
Are you sure you want to remove the shared folder **Bella Party** from your Dropbox? This folder will stay shared with any existing members. You can re-add it later.

Bella deletes shared folder Bella Party

two concepts



name follows **metadata** concept

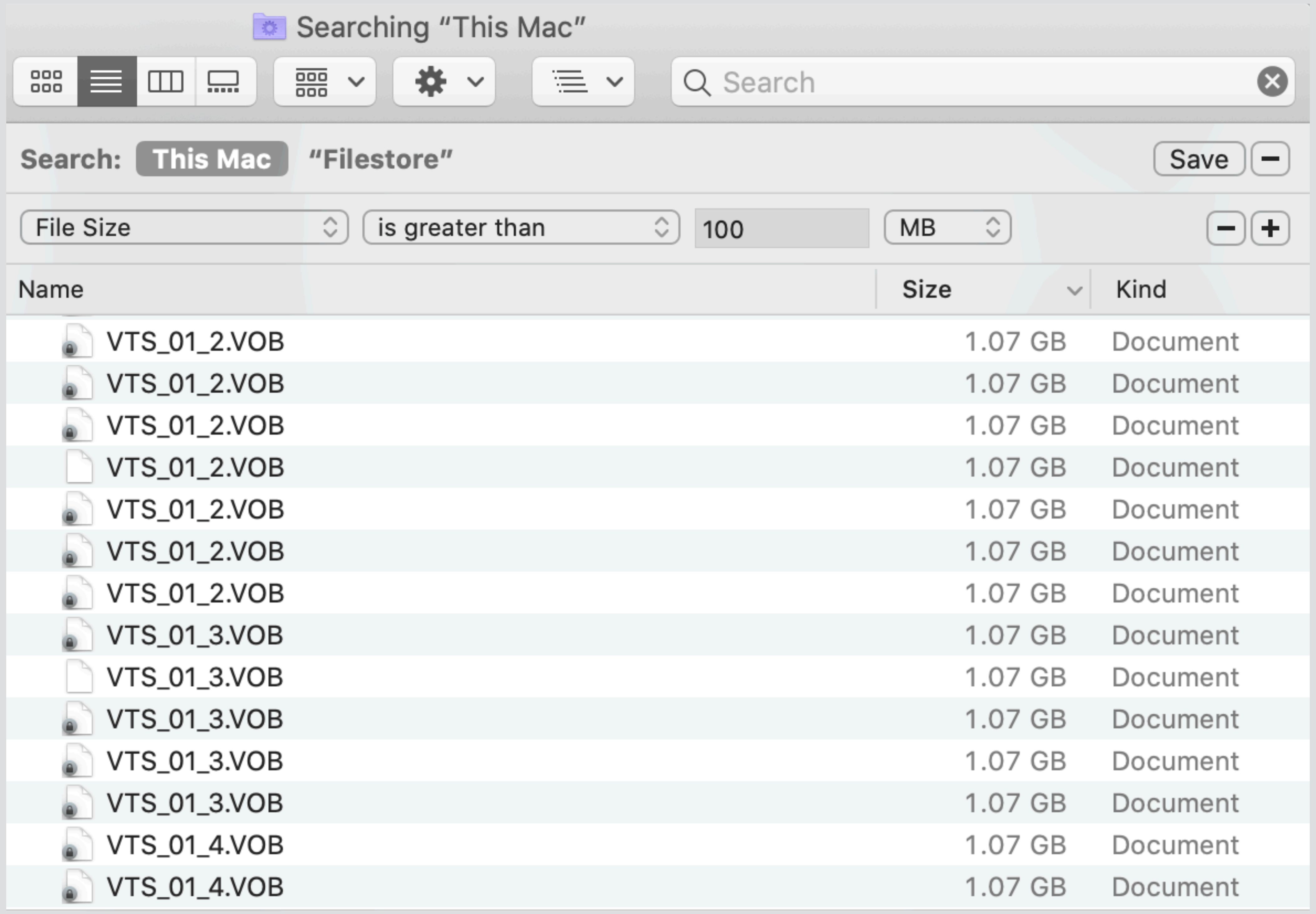


name is part of **unixFolder** concept

```
concept metadata
purpose tag items with properties for easy lookup
structure
 val: Item -> Property -> Value
actions
 define (i: Item, p: Property, v: Value)
  i.val[p] := v
 find (out is: Item, p: Property, v: Value)
  is = {i | i.val[p] = v}
 read (i: Item, p: Property, out v: Value)
  v := i.val[p]
principle
  define(i, p, v); no define(i, p,...); find(is,p,v)
  => i in is
```

```
concept unixFolder
purpose organize named items
structure
 member: Folder -> Name -> Item
actions
 add (i: Item, to: Folder, n: Name)
  to.member[n] := i
 rename (i: Item, f: Folder, old, new: Name)
  f.member := f.member - old->i  + new->i
 find (f: Folder, n: Name, out i: Item)
  i := f.member[n]
principle
  add(i, f, n); no rename(i, f,...) or add(i',f,n);
  find(f, n, i') => i' = i
```

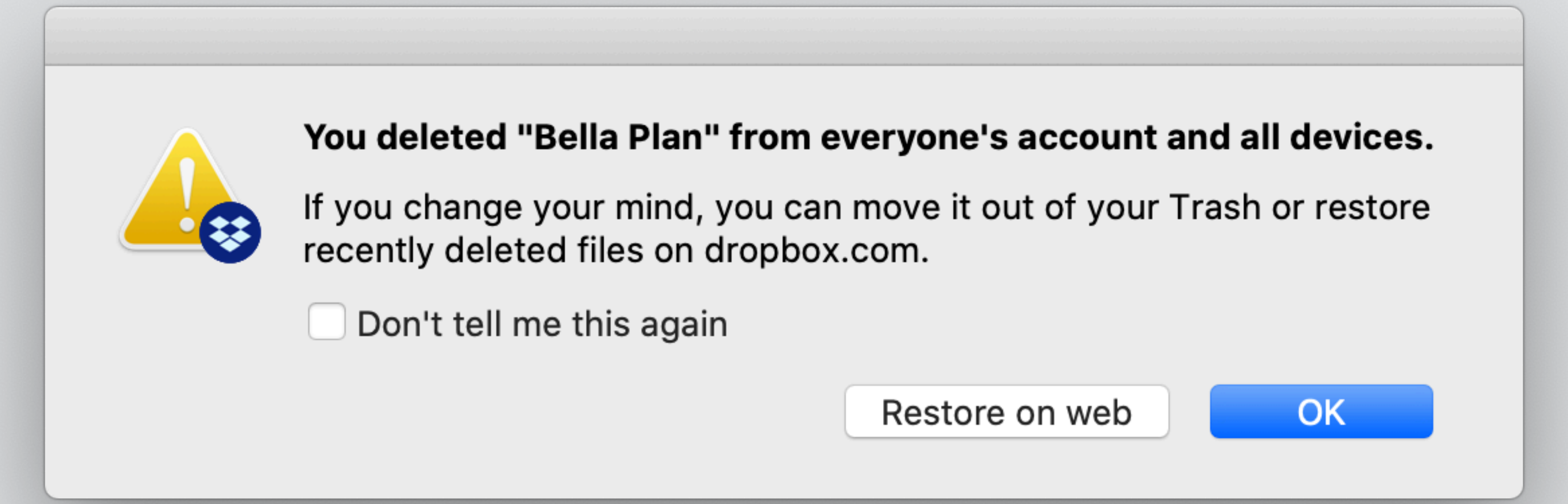
a real dropbox disaster



Search: This Mac “Filestore”

File Size is greater than 100 MB

Name	Size	Kind
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_2.VOB	1.07 GB	Document
VTS_01_3.VOB	1.07 GB	Document
VTS_01_3.VOB	1.07 GB	Document
VTS_01_3.VOB	1.07 GB	Document
VTS_01_3.VOB	1.07 GB	Document
VTS_01_3.VOB	1.07 GB	Document
VTS_01_4.VOB	1.07 GB	Document
VTS_01_4.VOB	1.07 GB	Document



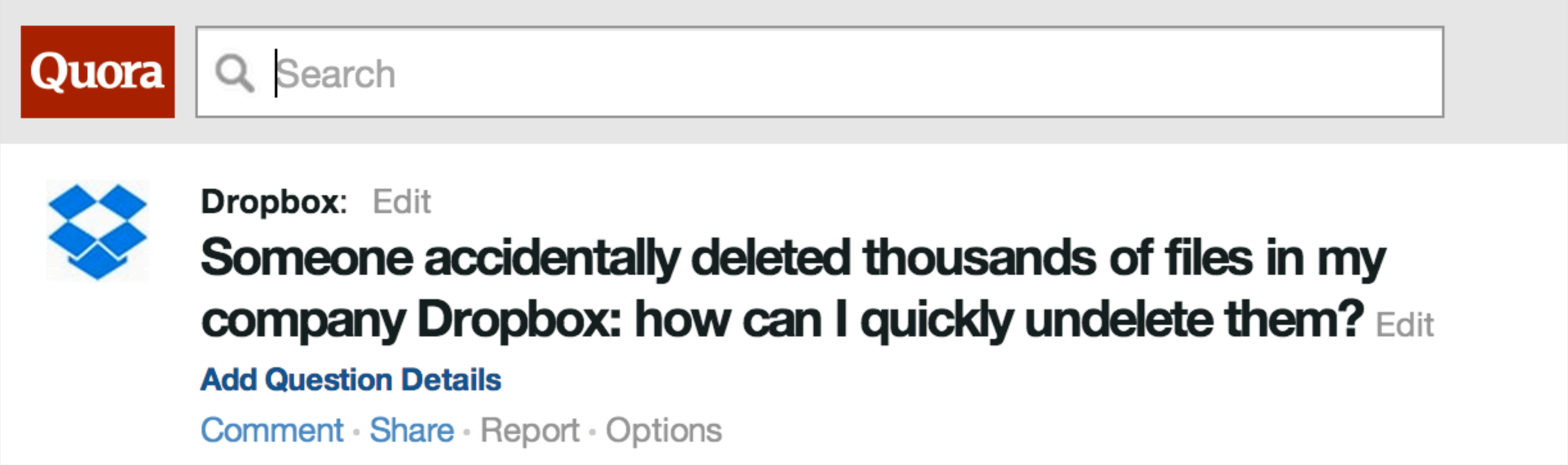
You deleted "Bella Plan" from everyone's account and all devices.

If you change your mind, you can move it out of your Trash or restore recently deleted files on dropbox.com.

□ Don't tell me this again

how to make space: find big files & delete ones you don’t recognize

a sad dropbox tale



Quora

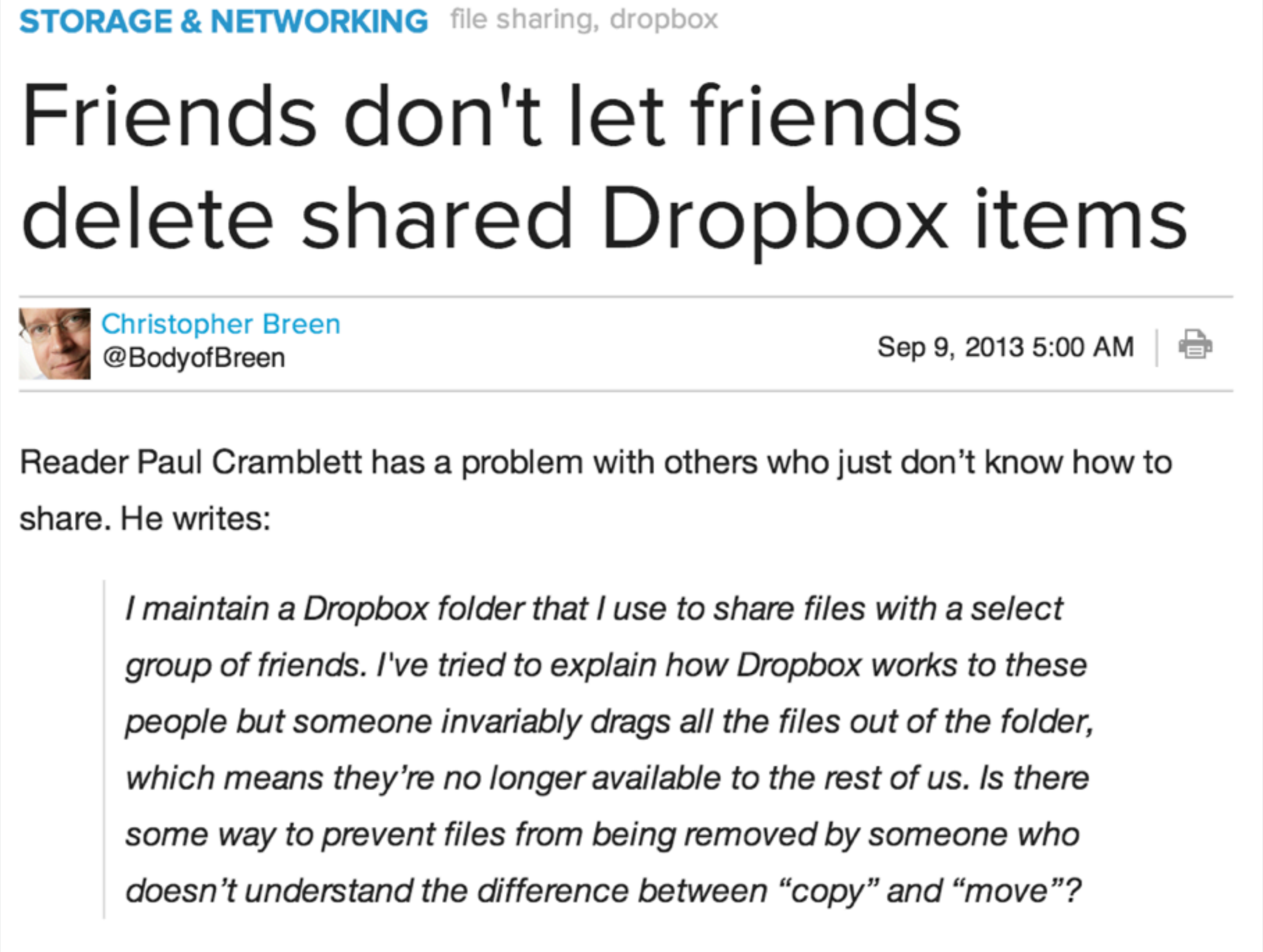
Search

Dropbox: Edit

Someone accidentally deleted thousands of files in my company Dropbox: how can I quickly undelete them? Edit

Add Question Details

Comment · Share · Report · Options



STORAGE & NETWORKING file sharing, dropbox

Friends don't let friends delete shared Dropbox items

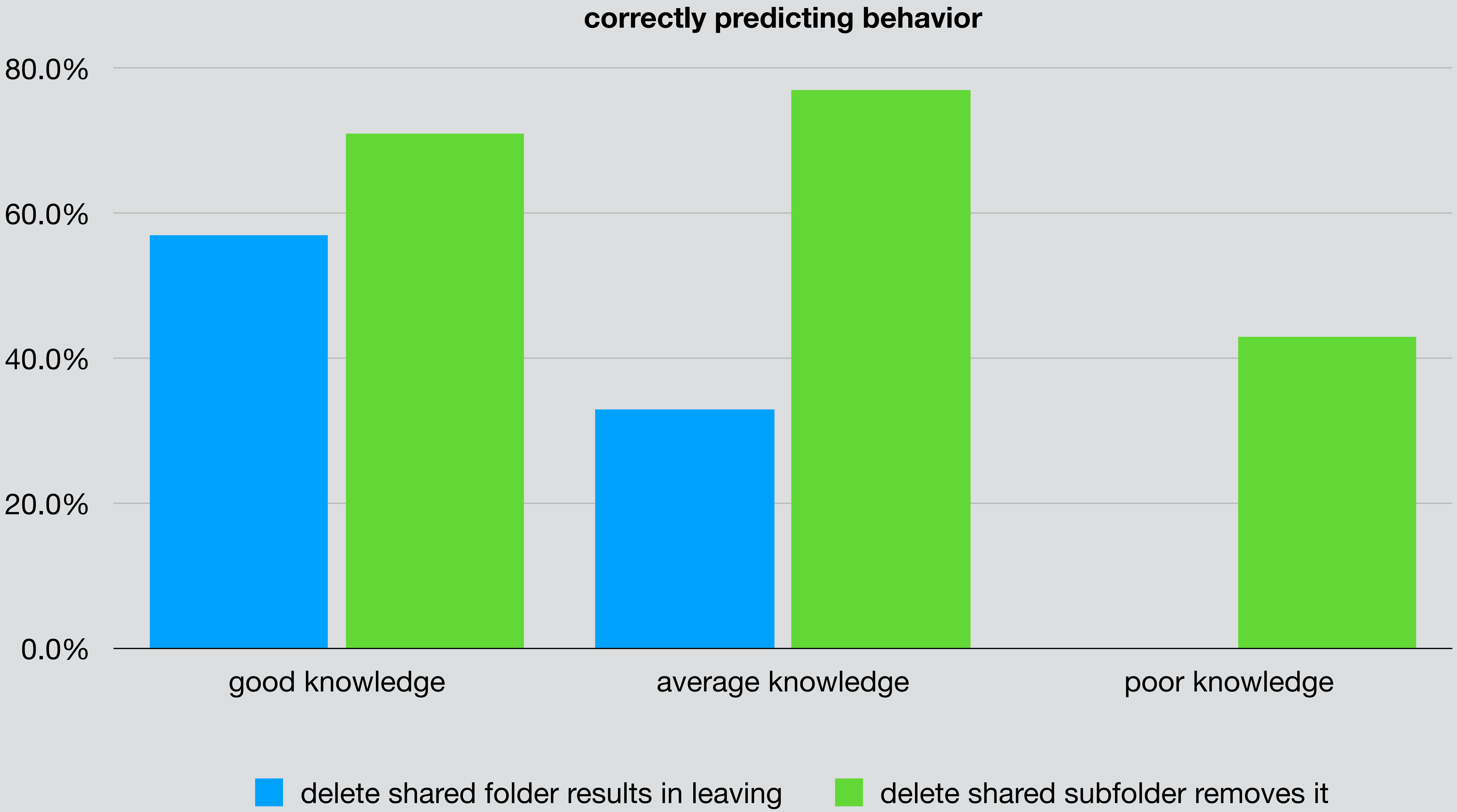
Christopher Breen
@BodyofBreen

Sep 9, 2013 5:00 AM

Reader Paul Cramblett has a problem with others who just don't know how to share. He writes:

> *I maintain a Dropbox folder that I use to share files with a select group of friends. I've tried to explain how Dropbox works to these people but someone invariably drags all the files out of the folder, which means they're no longer available to the rest of us. Is there some way to prevent files from being removed by someone who doesn't understand the difference between "copy" and "move"?*

survey of dropbox users (MIT CS undergrads)



Kelly Zhang

the big picture

what caused the dropbox problem? not these things

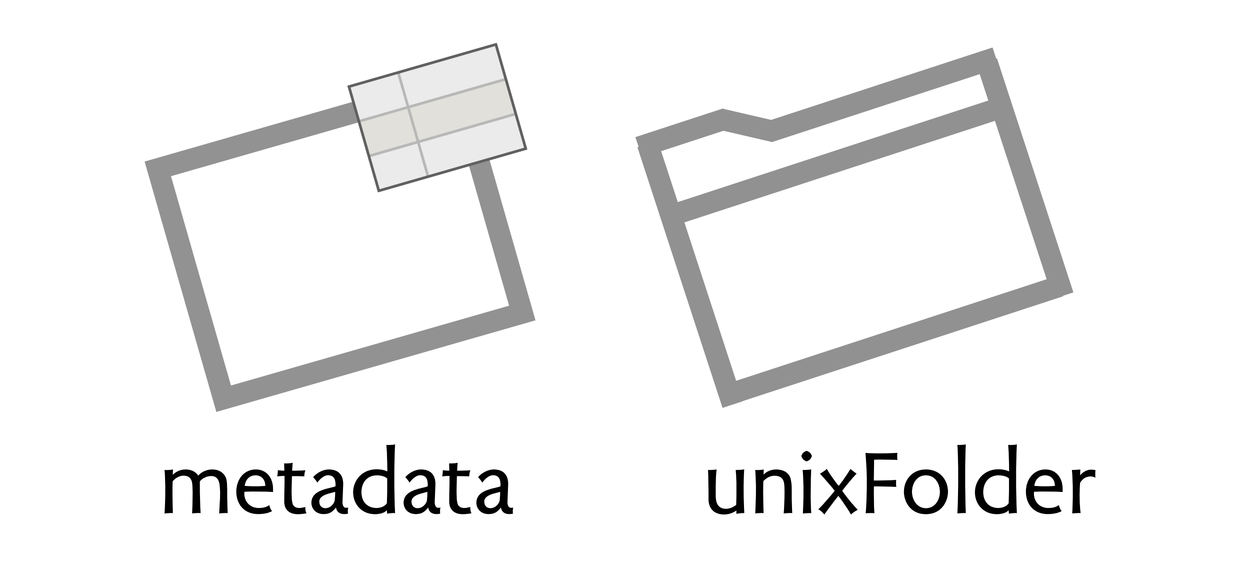
lack of technology

bugs in the code

classic UI design flaws

for robust, usable software...

get the concepts right

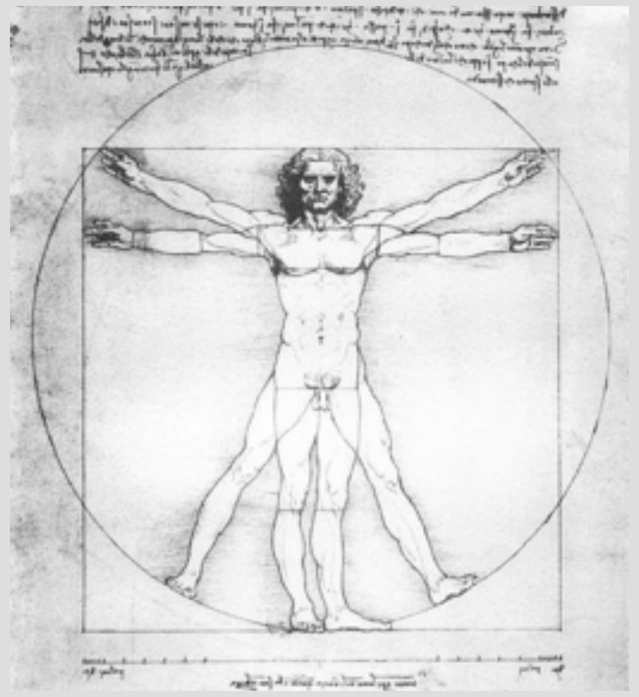


understand the user

design the user interface

avoid bugs in code

levels of UX design



physical

color, size, layout, type, touch, sound

Perceptual Fusion, Fitt's Law, Accessibility

linguistic

icons, labels, tooltips, site structure

Consistency, Info Foraging, Navigation Aids

conceptual

semantics, actions, data model, purpose

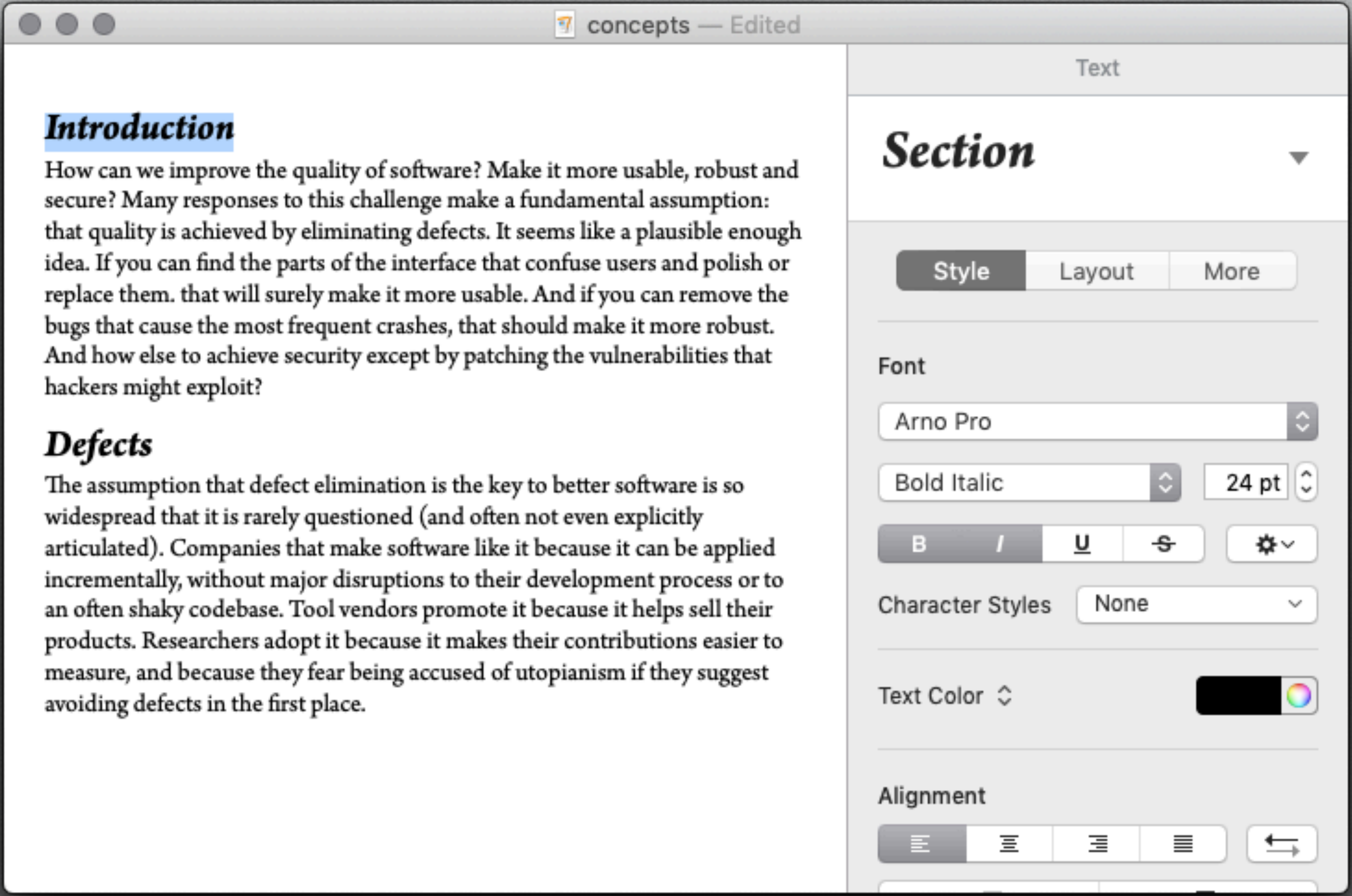
Undo, Norman's mapping, mental model alignment

concrete

abstract

a story of style

example: style concept



Introduction

How can we improve the quality of software? Make it more usable, robust and secure? Many responses to this challenge make a fundamental assumption: that quality is achieved by eliminating defects. It seems like a plausible enough idea. If you can find the parts of the interface that confuse users and polish or replace them. that will surely make it more usable. And if you can remove the bugs that cause the most frequent crashes, that should make it more robust. And how else to achieve security except by patching the vulnerabilities that hackers might exploit?

Defects

The assumption that defect elimination is the key to better software is so widespread that it is rarely questioned (and often not even explicitly articulated). Companies that make software like it because it can be applied incrementally, without major disruptions to their development process or to an often shaky codebase. Tool vendors promote it because it helps sell their products. Researchers adopt it because it makes their contributions easier to measure, and because they fear being accused of utopianism if they suggest avoiding defects in the first place.

Section

Font

Arno Pro

Bold Italic

24 pt

Character Styles: None

There is no problem in computer science that cannot be solved by introducing another level of indirection.
David Wheeler

concept style

name: essential for knowledge capture

purpose consistent formatting

purpose: why the concept exists

structure

```
defined: Style -> one Format
style: Element -> one Style
format: Element -> one Format = style.defined
```

structure: localized data model

actions

```
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s
```

actions: observable & atomic

principle

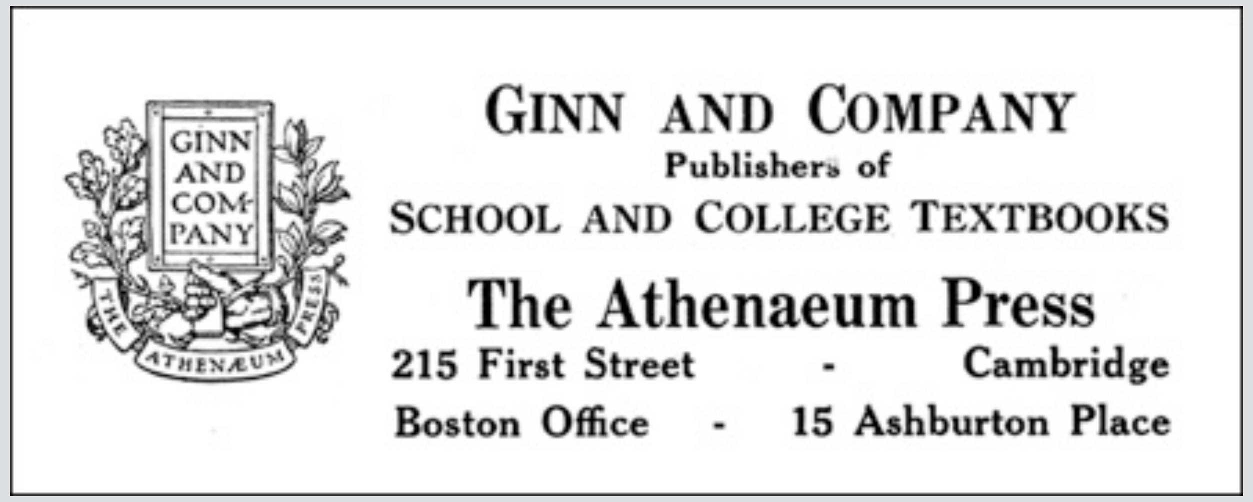
```
after define(s,f); assign(e1,s);
assign(e2,s); define(s,f’)
observe e1.format = e2.format = f’
```

OP justifies & explains design

how behavior fulfills purpose

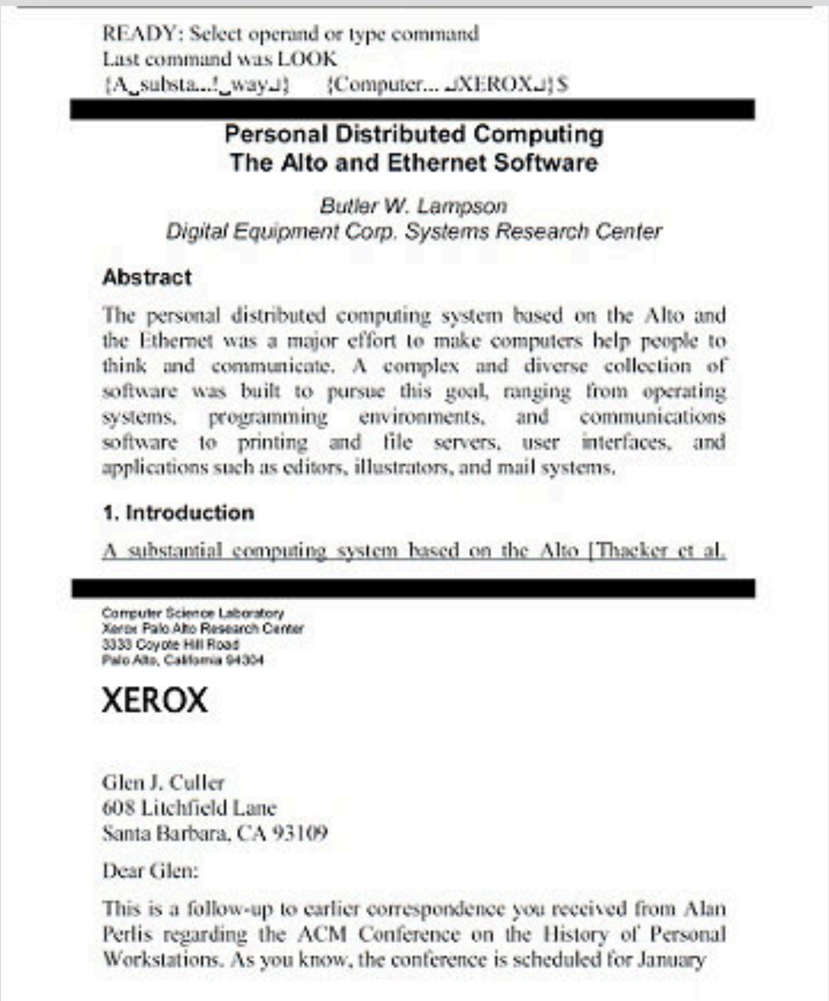
Michael Polanyi
operational principle

the invention of style


GINN AND COM-PANY
THE ATHENÆUM PRESS

GINN AND COMPANY
Publishers of
SCHOOL AND COLLEGE TEXTBOOKS
The Athenaeum Press
215 First Street - Cambridge
Boston Office - 15 Ashburton Place

Tim Mott visits Ginn in 1974
brings idea of styles to PARC



READY: Select operand or type command
Last command was LOOK
{A⌴substa...!⌴way⏎} {Computer... ⏎XEROX⏎}$

Personal Distributed Computing
The Alto and Ethernet Software

Butler W. Lampson
Digital Equipment Corp. Systems Research Center

Abstract

The personal distributed computing system based on the Alto and the Ethernet was a major effort to make computers help people to think and communicate. A complex and diverse collection of software was built to pursue this goal, ranging from operating systems, programming environments, and communications software to printing and file servers, user interfaces, and applications such as editors, illustrators, and mail systems.

1. Introduction

A substantial computing system based on the Alto [Thacker et al.

Computer Science Laboratory
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

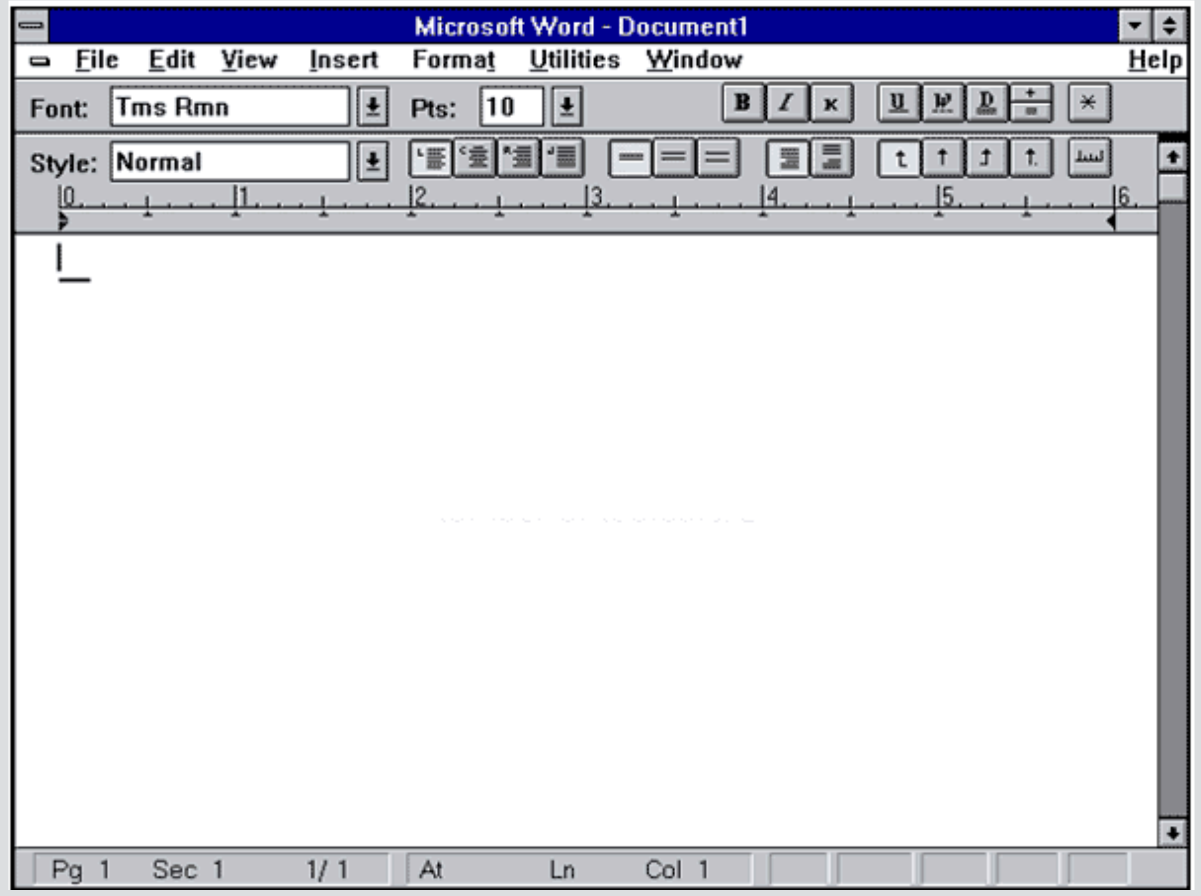
XEROX

Glen J. Culler
608 Litchfield Lane
Santa Barbara, CA 93109

Dear Glen:

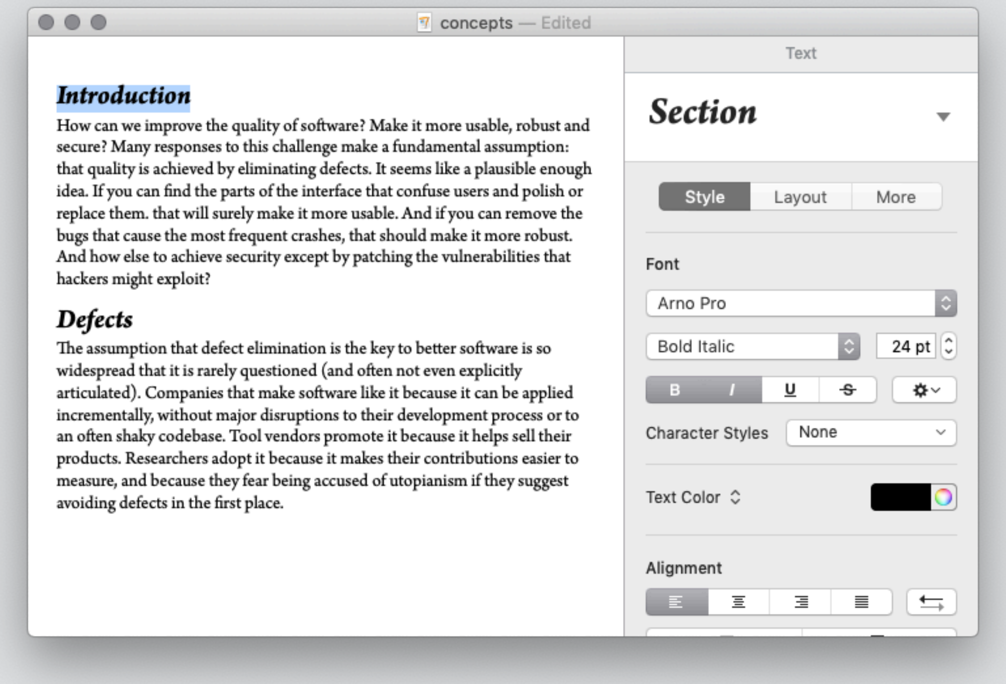
This is a follow-up to earlier correspondence you received from Alan Perlis regarding the ACM Conference on the History of Personal Workstations. As you know, the conference is scheduled for January

Charles Simonyi's team
implements style in
Bravo text editor


Microsoft Word - Document1
File Edit View Insert Format Utilities Window Help
B I K U W D
Pg 1 Sec 1 1/1 At Ln Col 1

Font: Tms Rmn Pts: 10
Style: Normal

Simonyi brings style
to Microsoft in 1983



Introduction

How can we improve the quality of software? Make it more usable, robust and secure? Many responses to this challenge make a fundamental assumption: that quality is achieved by eliminating defects. It seems like a plausible enough idea. If you can find the parts of the interface that confuse users and polish or replace them. that will surely make it more usable. And if you can remove the bugs that cause the most frequent crashes, that should make it more robust. And how else to achieve security except by patching the vulnerabilities that hackers might exploit?

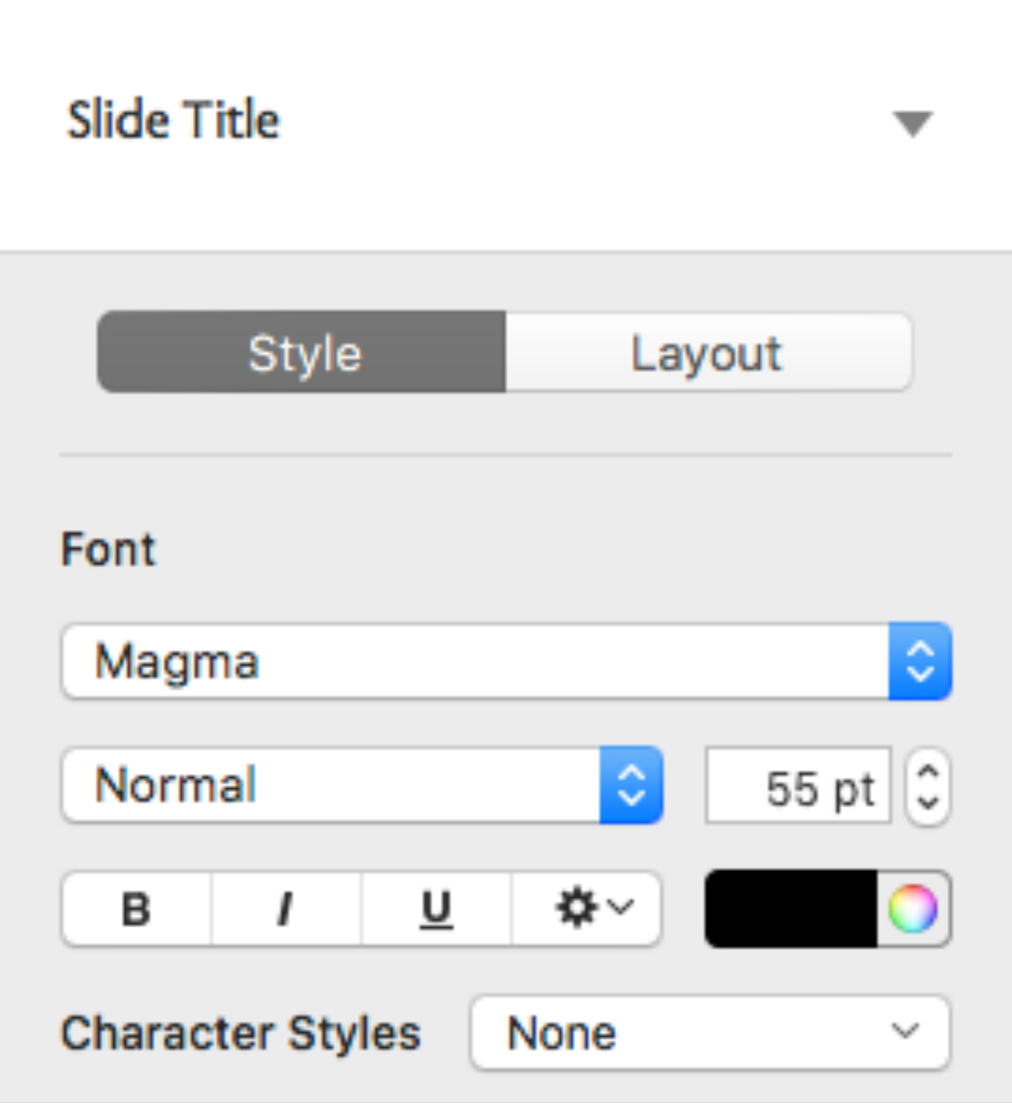
Defects

The assumption that defect elimination is the key to better software is so widespread that it is rarely questioned (and often not even explicitly articulated). Companies that make software like it because it can be applied incrementally, without major disruptions to their development process or to an often shaky codebase. Tool vendors promote it because it helps sell their products. Researchers adopt it because it makes their contributions easier to measure, and because they fear being accused of utopianism if they suggest avoiding defects in the first place.

Section

Font
Arno Pro
Bold Italic 24 pt
Character Styles None
Text Color
Alignment

Apple Pages 2005

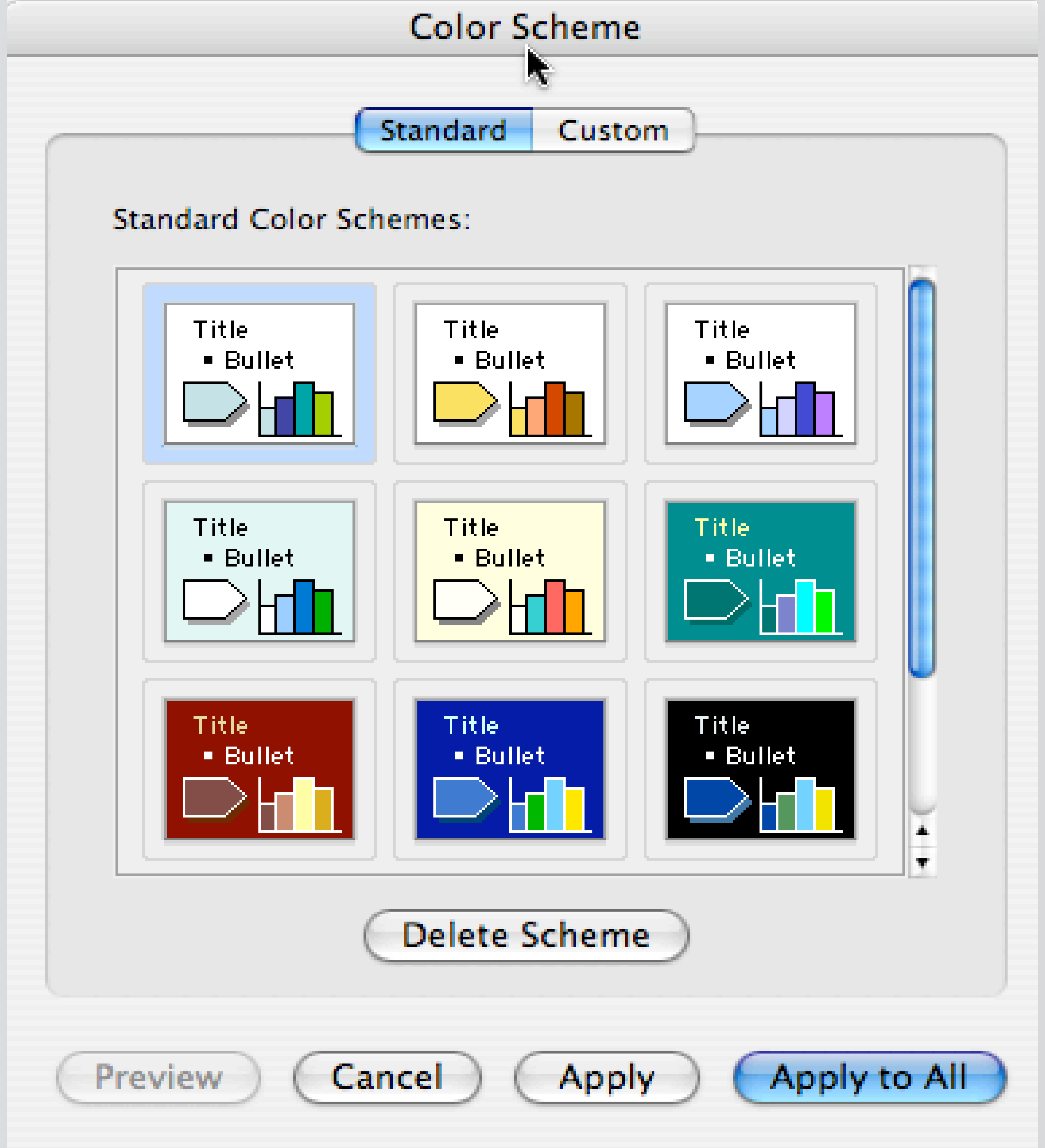


Slide Title

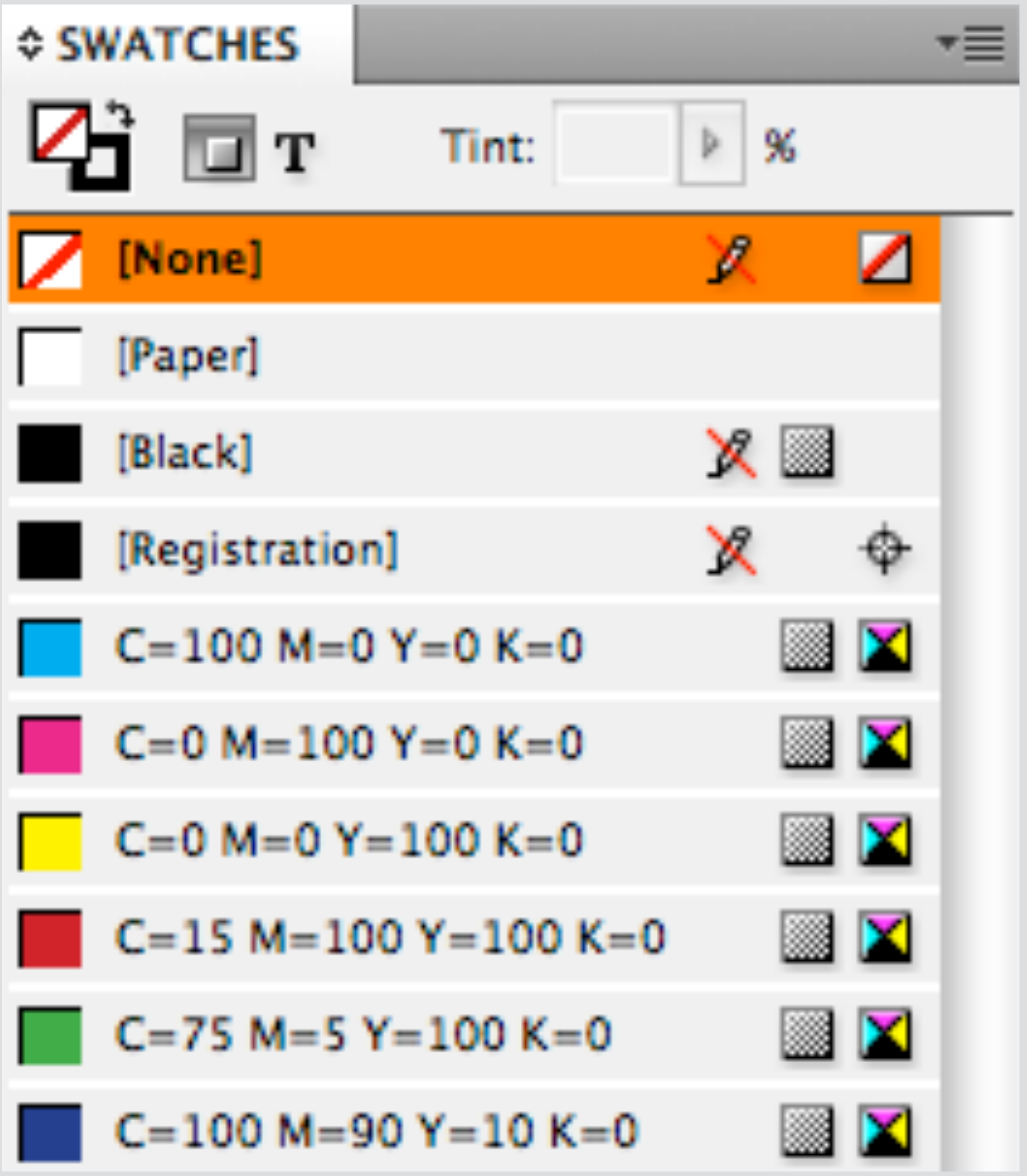
Font
Magma
Normal 55 pt
Character Styles None

Apple Keynote
adds style concept
c. 2017

other instances of style

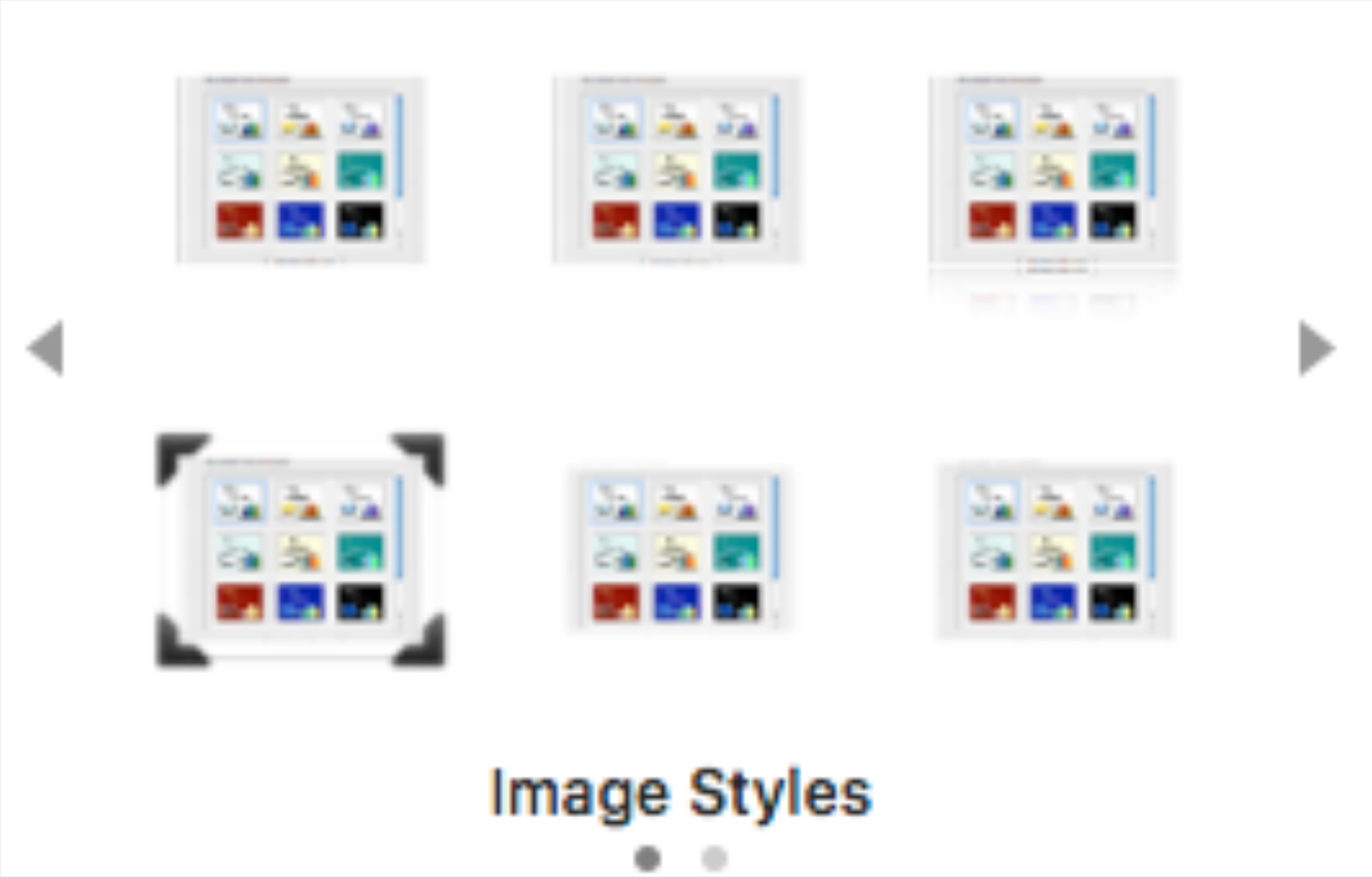


Powerpoint color schemes



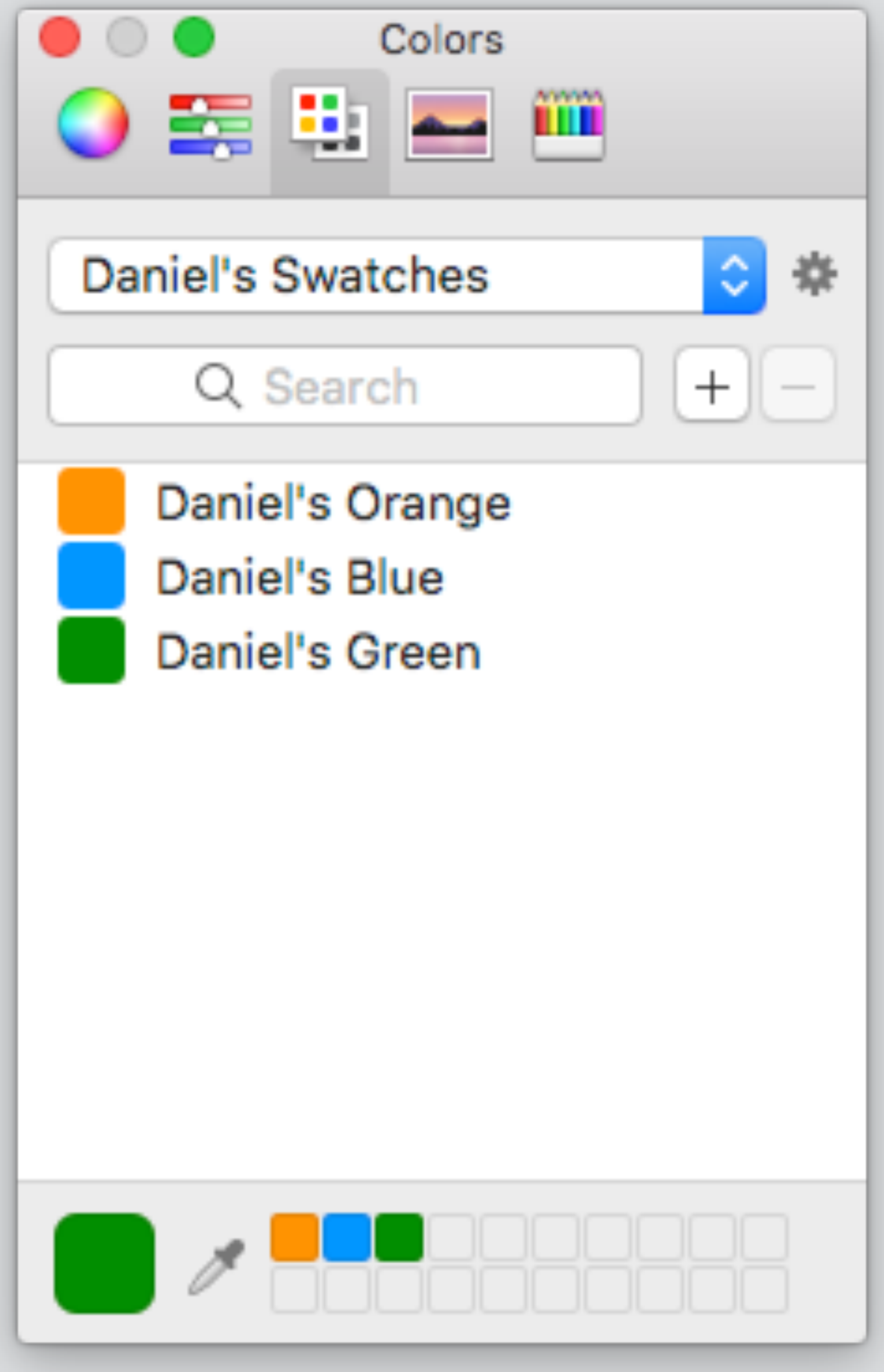
[None]
[Paper]
[Black]
[Registration]
C=100 M=0 Y=0 K=0
C=0 M=100 Y=0 K=0
C=0 M=0 Y=100 K=0
C=15 M=100 Y=100 K=0
C=75 M=5 Y=100 K=0
C=100 M=90 Y=10 K=0

Indesign swatches

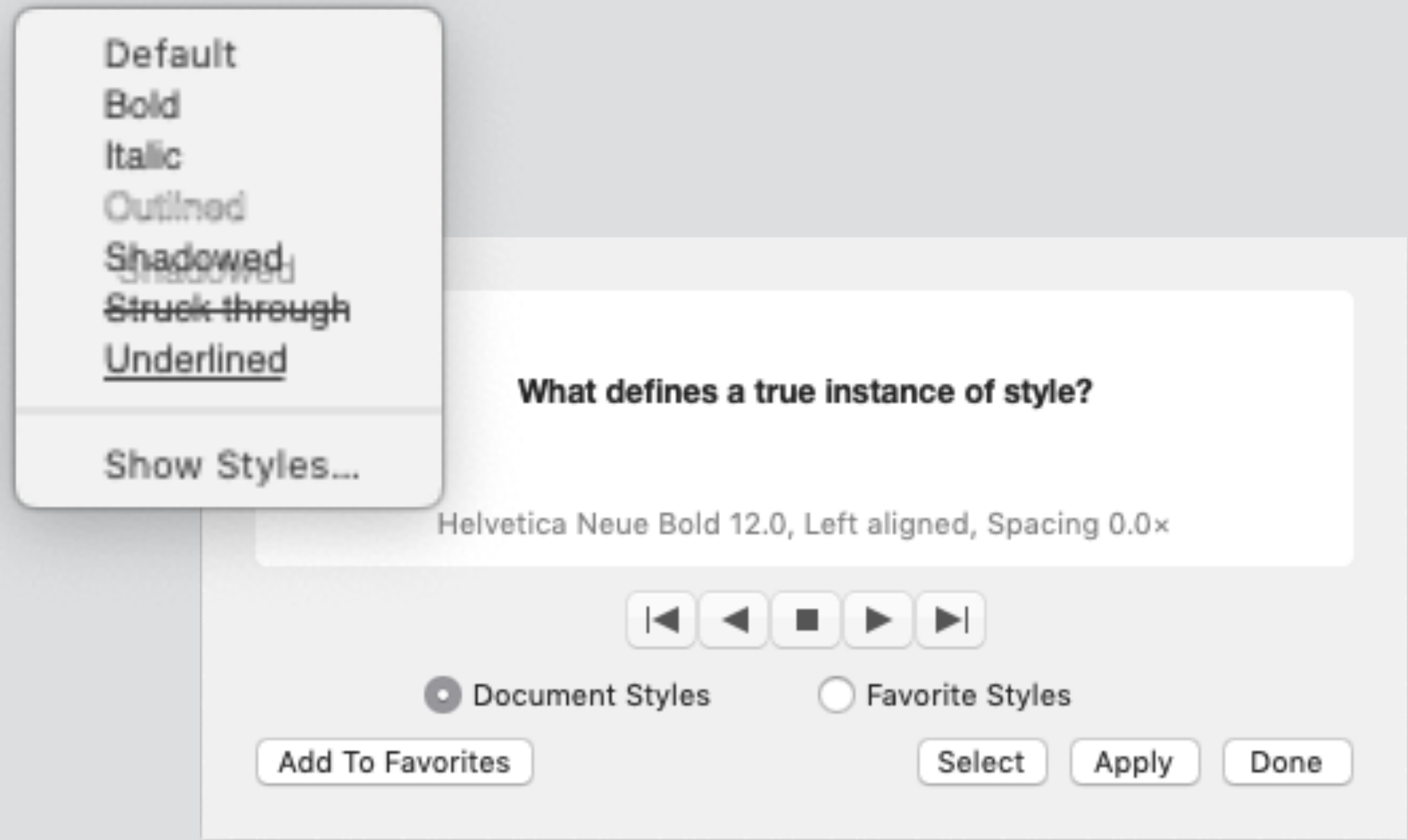


Keynote image styles

non-instances: “pseudo-style”



Apple color swatches



What defines a true instance of style?

Helvetica Neue Bold 12.0, Left aligned, Spacing 0.0×

TextEdit “styles"

a concept handbook

concepts indexed by purpose
consistent formatting:
style, template, copy settings, ...

design variants
override formats
style inheritance
next style
partial styles
shareable stylesheet

known issues
deleting styles: what happens to elements?
copying elements between documents
need for “as is” values
troublesome properties (eg, fontstyle)

typical uses
formatting paragraphs & characters
formatting graphic objects
Word, Pages, CSS, ...

often used with
paragraph
format

implementation hints
...

key properties of a concept: style as an example

inventive

style has a long history of creativity & refinement

not domain entities that are just “out there”

purposeful

for consistency of formatting, esp. in large documents

not arbitrary fragments of functionality

behavioral

“if you update the style of multiple paragraphs their formatting all changes in concert”

not data models or ontologies

self-contained

style concept independent of format, paragraph, typeface

not datatypes or modules

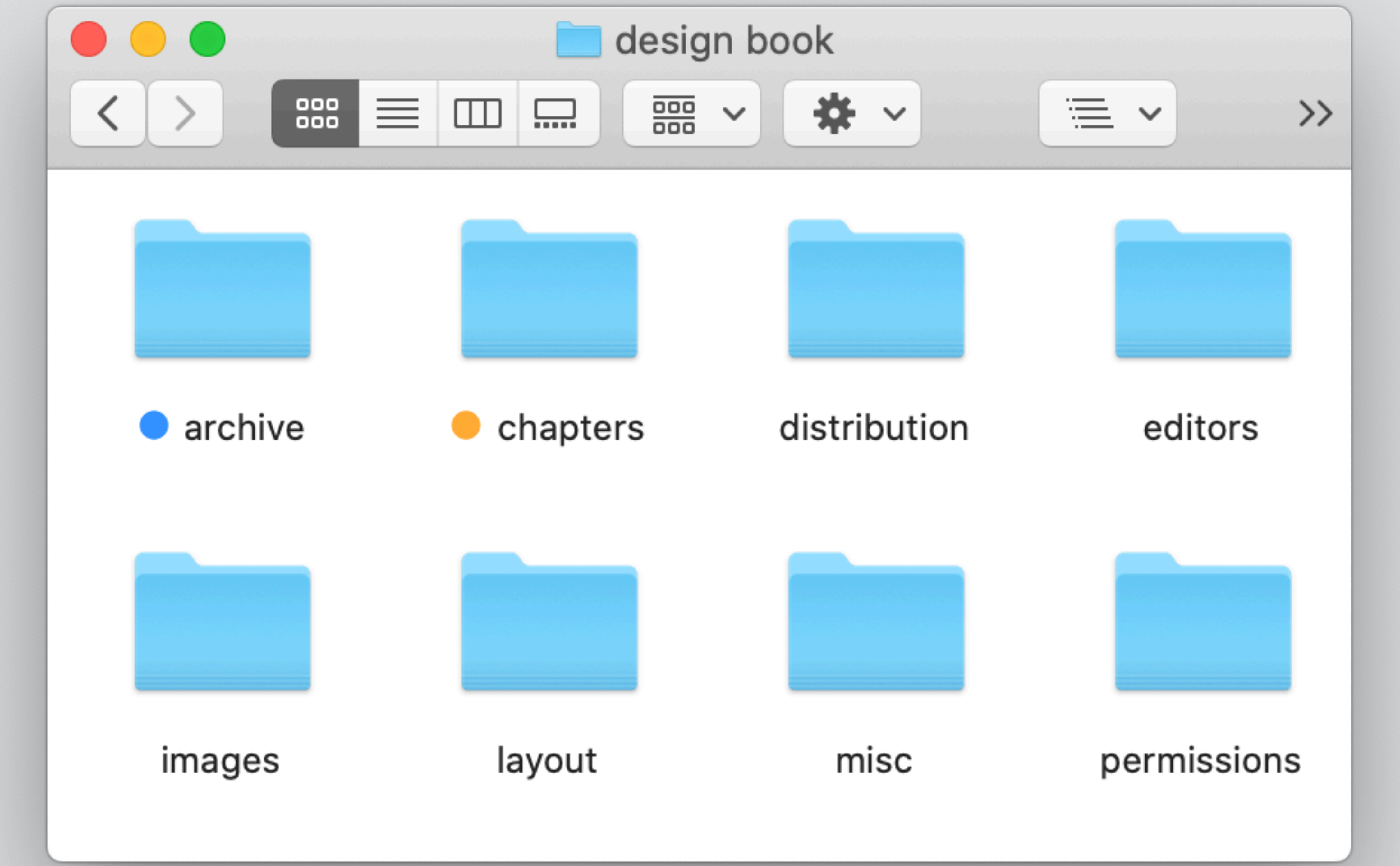
reusable

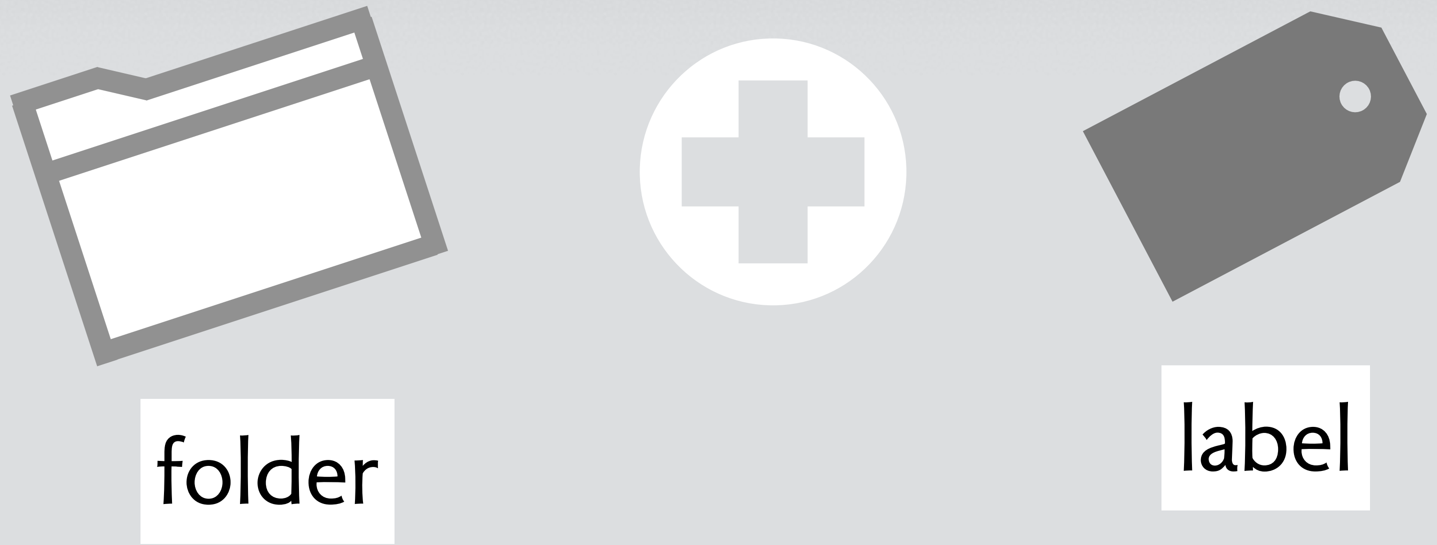
style in Keynote inspired by style in Pages, inspired by Style in Word…

often not domain-specific

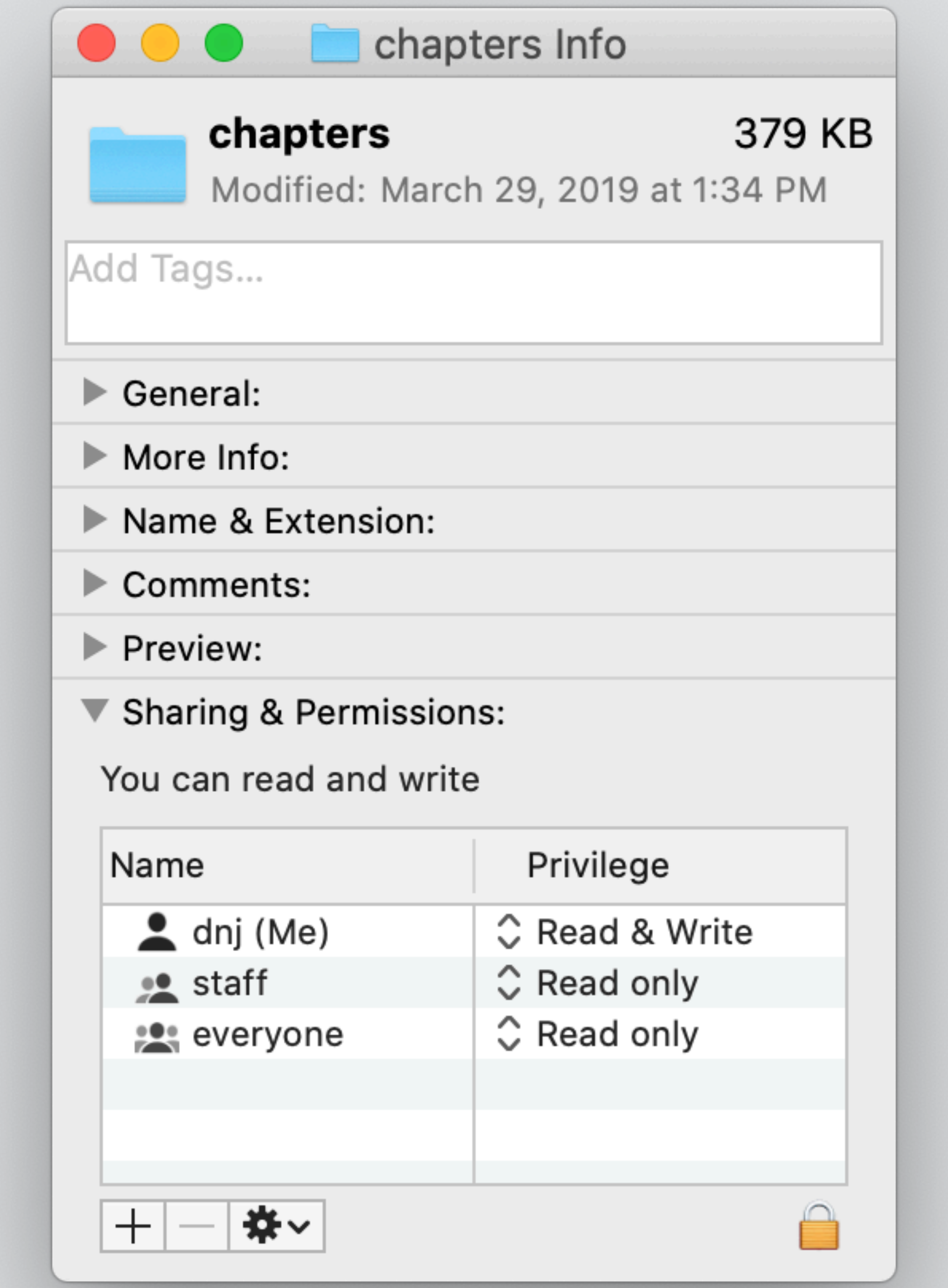
composing concepts

weakest: existence coupling





most common: action synchronization



chapters Info

chapters 379 KB

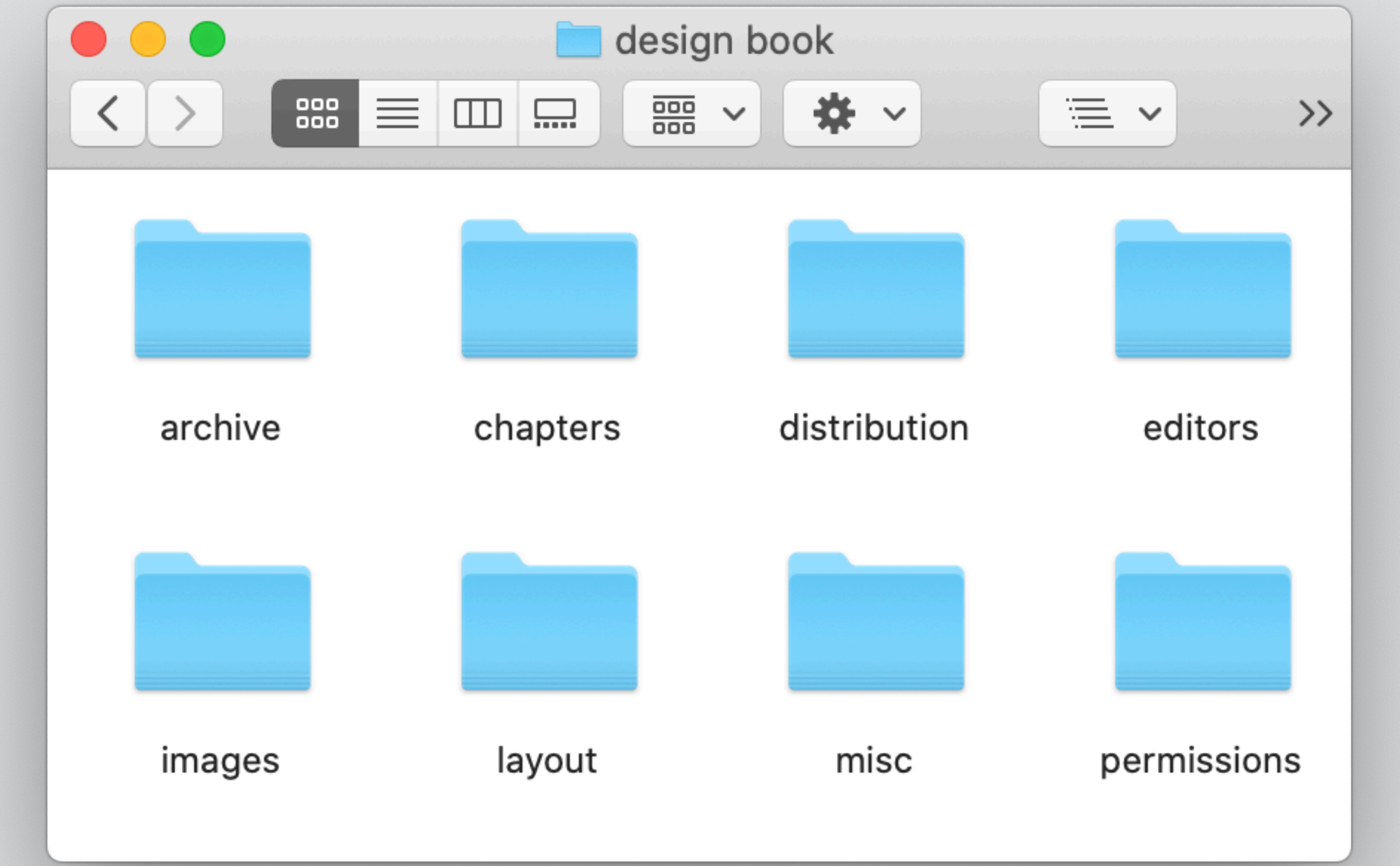
Modified: March 29, 2019 at 1:34 PM

Add Tags...

- General:
- More Info:
- Name & Extension:
- Comments:
- Preview:
- Sharing & Permissions:

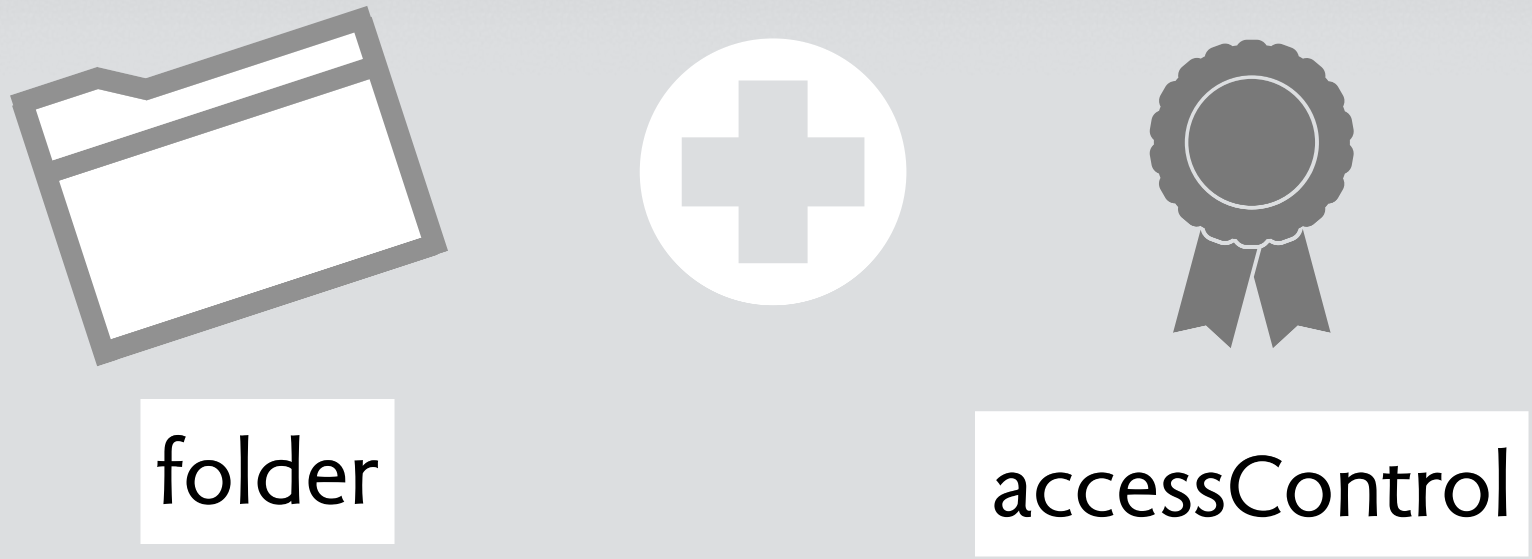
You can read and write

Name	Privilege
dnj (Me)	Read & Write
staff	Read only
everyone	Read only



design book

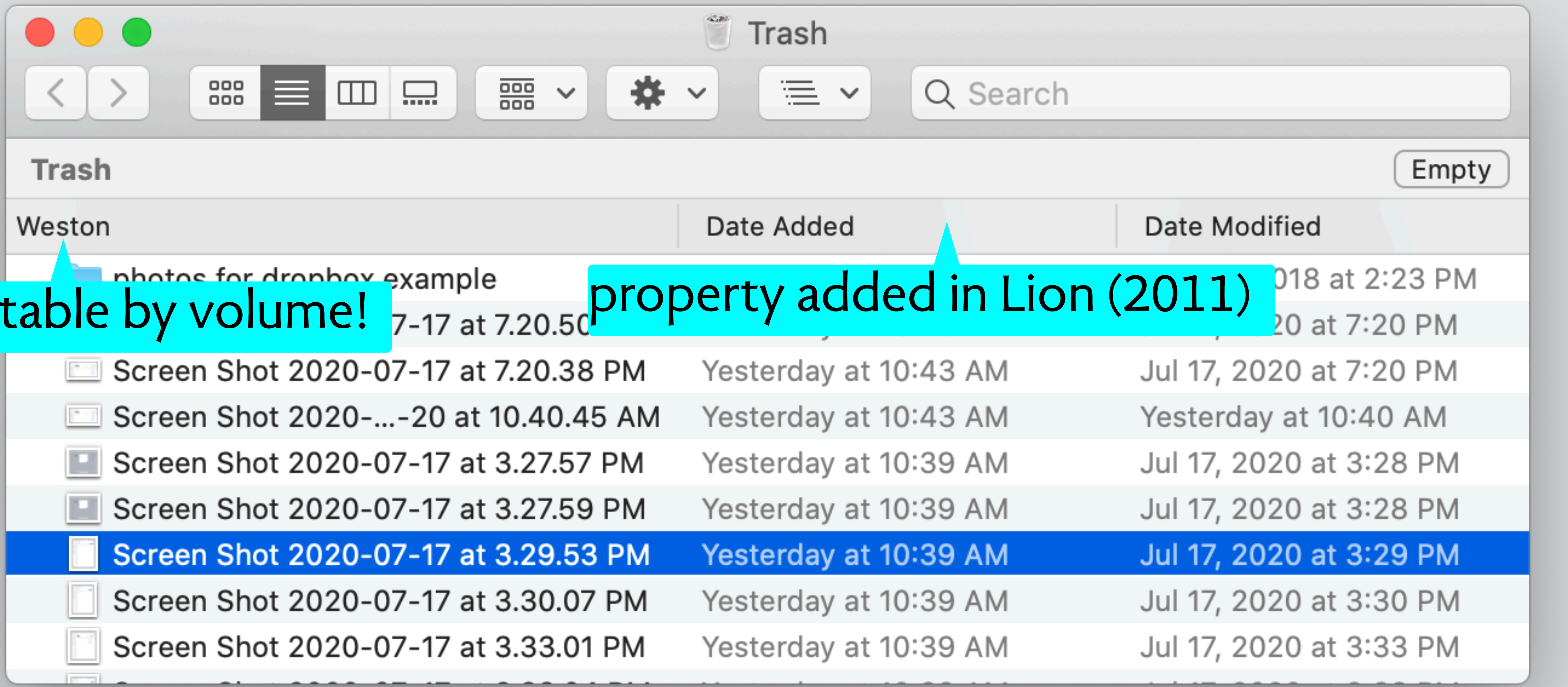
archive, chapters, distribution, editors, images, layout, misc, permissions



folder

accessControl

tightest: structure synchronization



Trash — Empty

Weston	Date Added	Date Modified
photos for dropbox example		018 at 2:23 PM
7-17 at 7.20.50		20 at 7:20 PM
Screen Shot 2020-07-17 at 7.20.38 PM	Yesterday at 10:43 AM	Jul 17, 2020 at 7:20 PM
Screen Shot 2020-...-20 at 10.40.45 AM	Yesterday at 10:43 AM	Yesterday at 10:40 AM
Screen Shot 2020-07-17 at 3.27.57 PM	Yesterday at 10:39 AM	Jul 17, 2020 at 3:28 PM
Screen Shot 2020-07-17 at 3.27.59 PM	Yesterday at 10:39 AM	Jul 17, 2020 at 3:28 PM
Screen Shot 2020-07-17 at 3.29.53 PM	Yesterday at 10:39 AM	Jul 17, 2020 at 3:29 PM
Screen Shot 2020-07-17 at 3.30.07 PM	Yesterday at 10:39 AM	Jul 17, 2020 at 3:30 PM
Screen Shot 2020-07-17 at 3.33.01 PM	Yesterday at 10:39 AM	Jul 17, 2020 at 3:33 PM

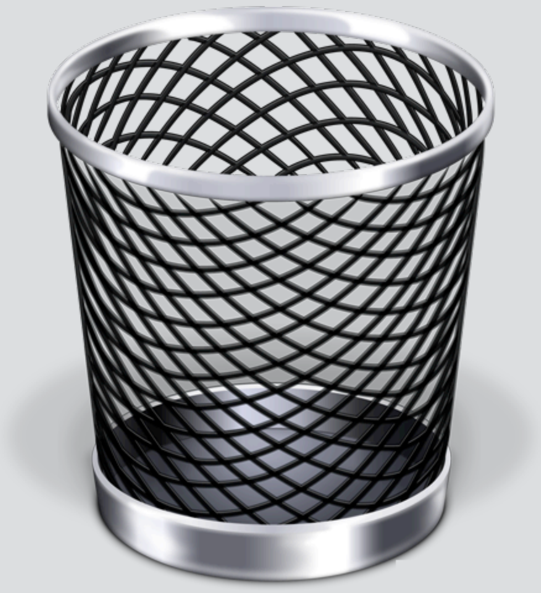
folder sortable by volume!

property added in Lion (2011)

folder
trash

designing on purpose

understanding why: the key to usability



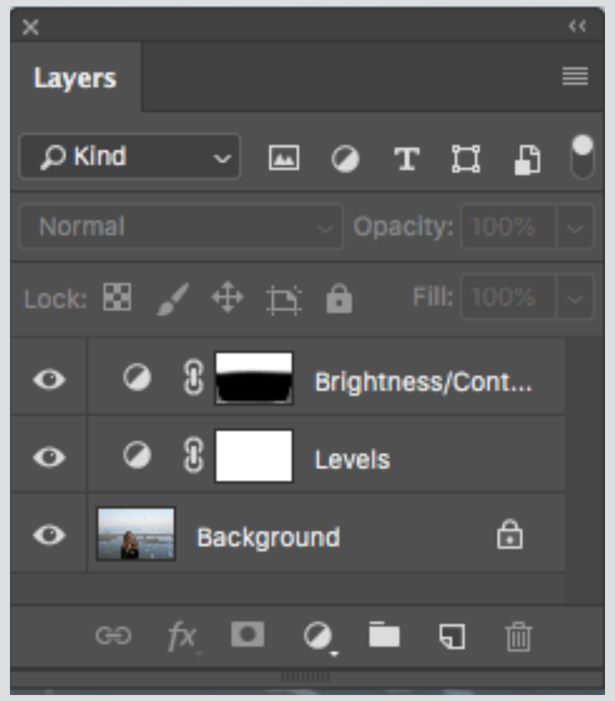
Macintosh Trash

wrong purpose

deleting things

undeleting things

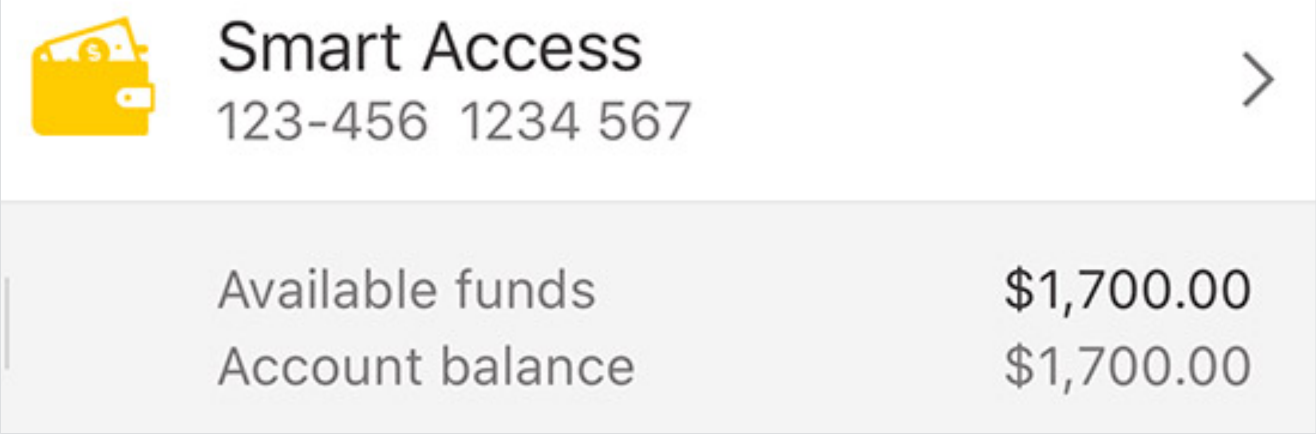
right purpose



Photoshop Layers

stacking objects

non-destructive editing



Smart Access
123-456 1234 567

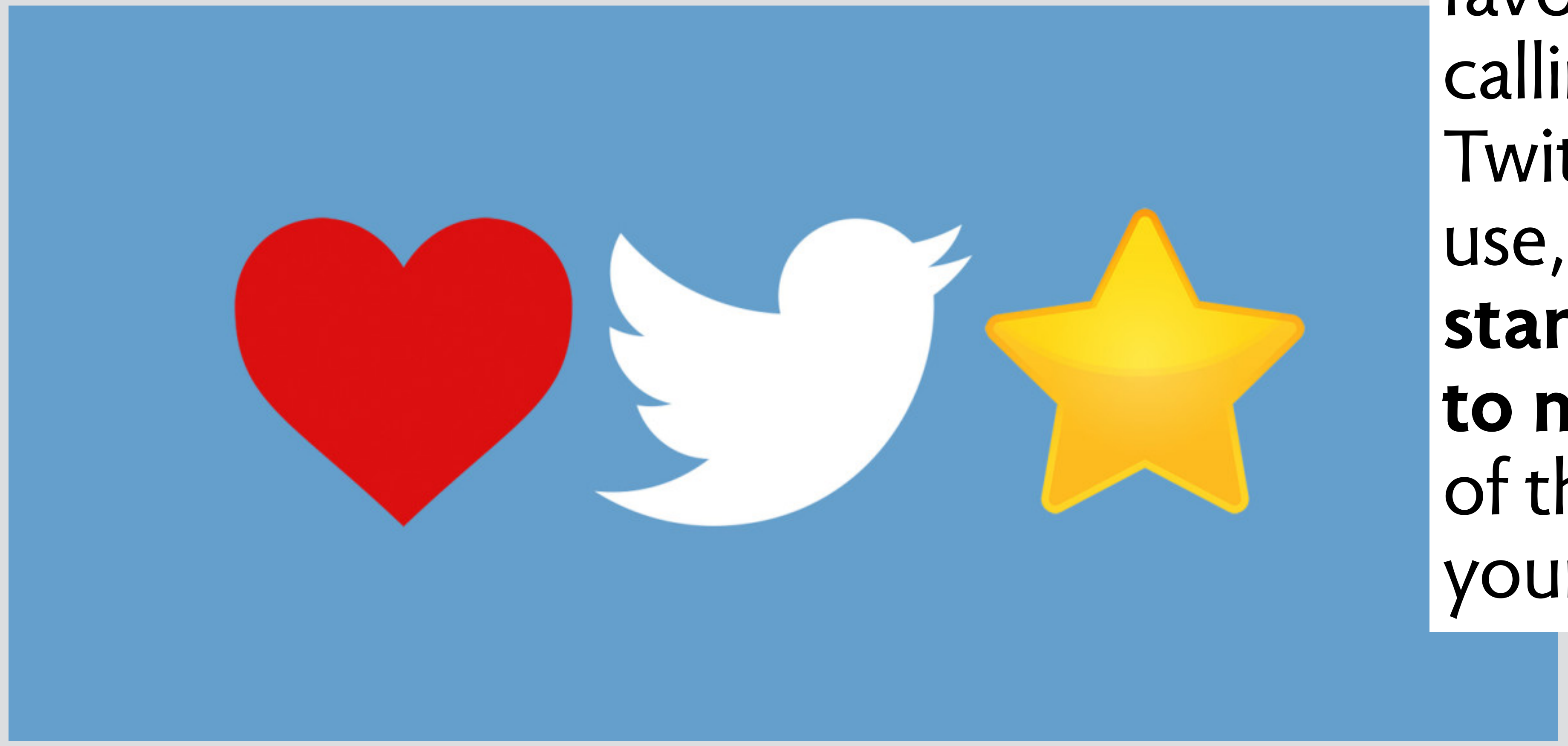
Available funds	$1,700.00
Account balance	$1,700.00

Available Funds

signal that deposits are safe

permission to use

a conceptual flaw in Twitter



We are changing our star icon for favorites to a heart and we'll be calling them likes. We want to make Twitter easier and more rewarding to use, and **we know that at times the star could be confusing, especially to newcomers**. You might like a lot of things, but not everything can be your favorite. *Twitter*

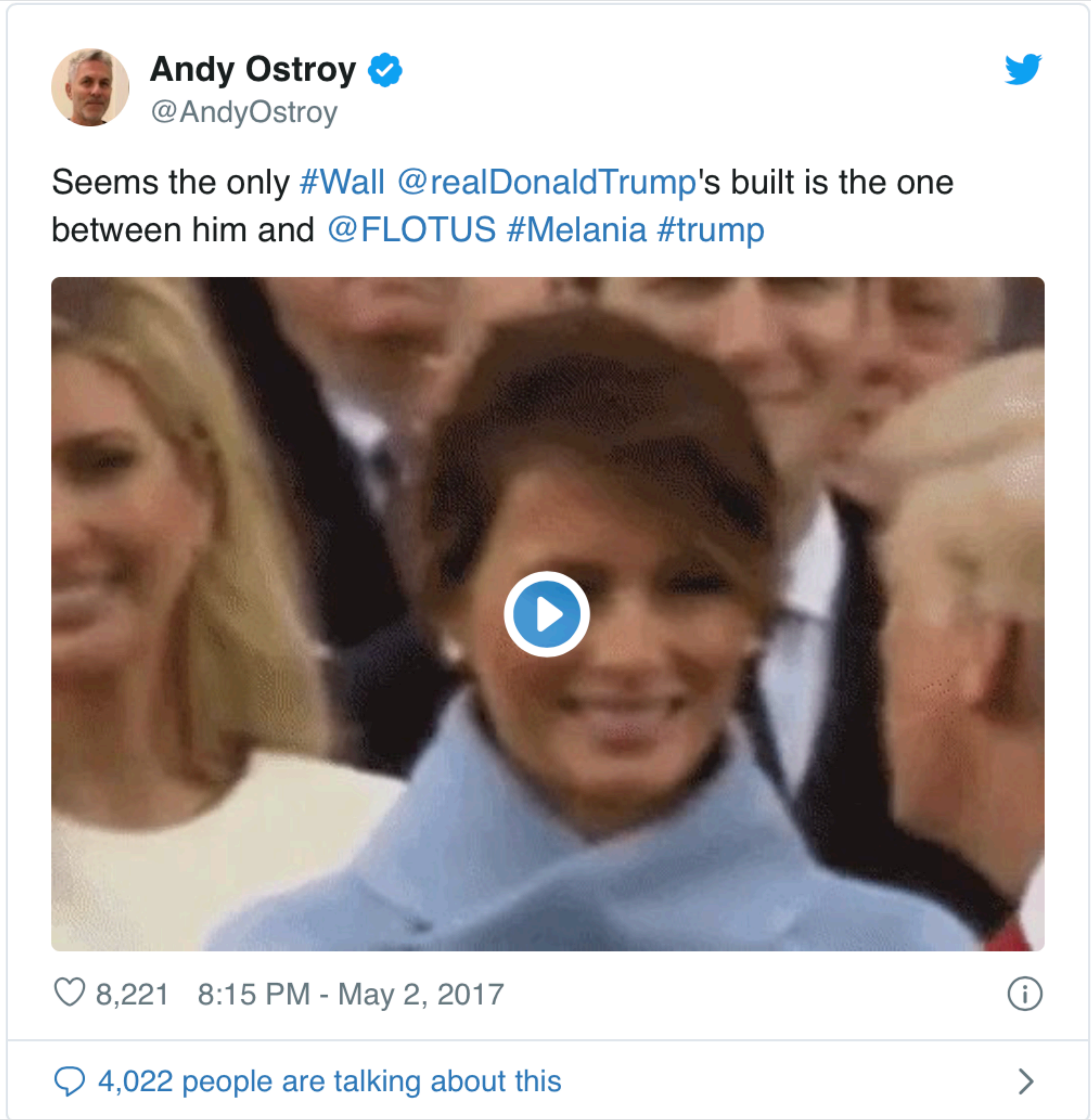
Nov 2, 2015: Twitter changes Favorite (Star) to Like (Heart)

The problem for Twitter is that the "favorite" function had developed a range of uses over time, many of which are known only to the journalists and social-media experts who spend all their time on the service. For some (including me), **clicking the star icon was a way of saving a tweet for later**, or of sending a link that was being shared to a service like Instapaper or Pocket. *Mathew Ingram*

I've favorited more than 60,000 tweets over the years, and in that time I've come to appreciate how versatile that little button is. I use it as **a kind of read receipt** to acknowledge replies; I use it whenever a tweet makes me laugh out loud; I use it when someone criticizes me by name in the hopes that seeing it's one of my "favorite" tweets will confuse and upset them. *Casey Newton*

If Twitter integrated a simple heart gesture into each Tweet, engagement across the entire service would explode. More of us would be getting loving feedback on our posts and that would **directly encourage more posting** and more frequent visits to Twitter. *Chris Sacca*

confused concepts lead to confused users

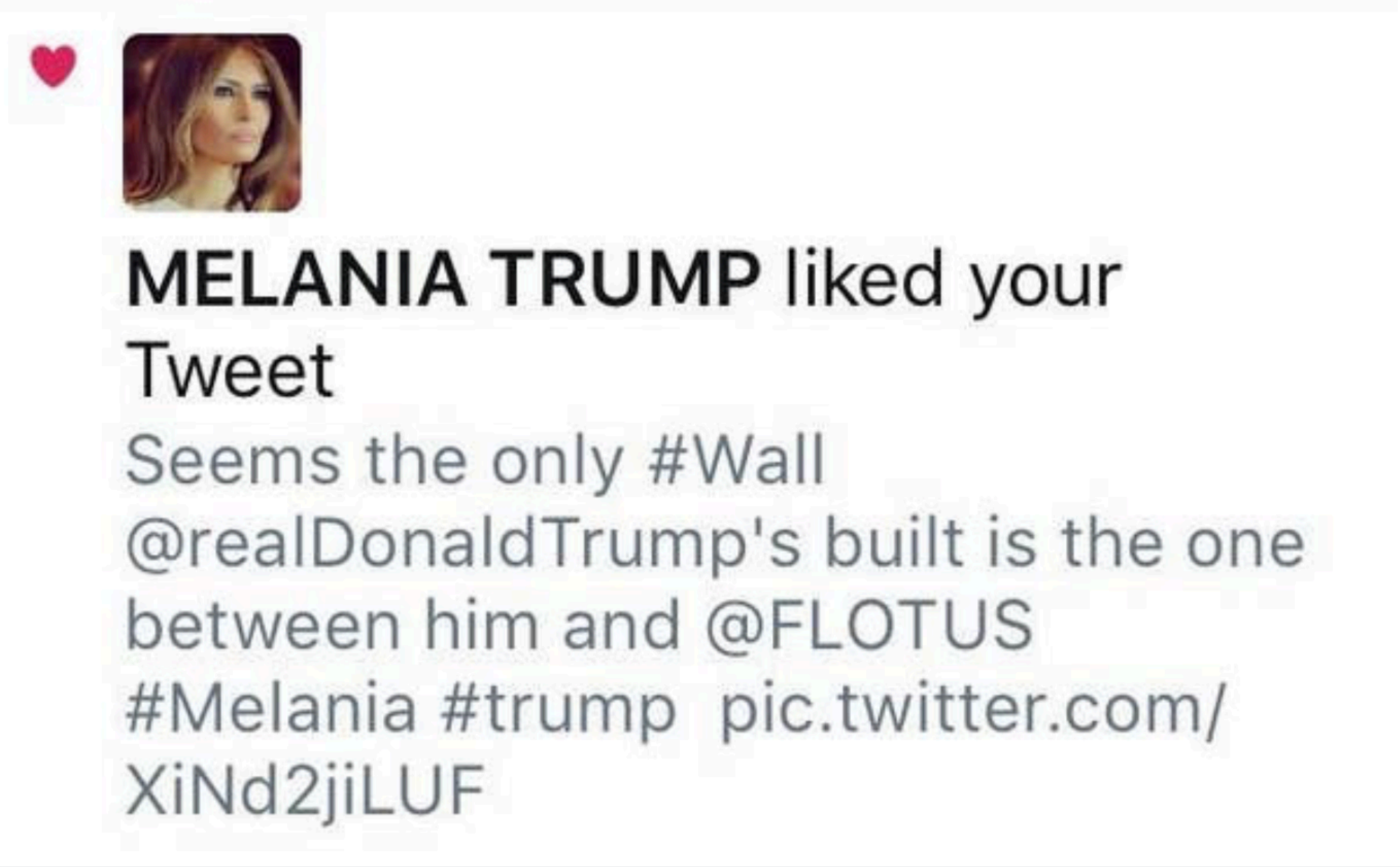


Andy Ostroy
@AndyOstroy

Seems the only #Wall @realDonaldTrump's built is the one between him and @FLOTUS #Melania #trump

♡ 8,221 8:15 PM - May 2, 2017

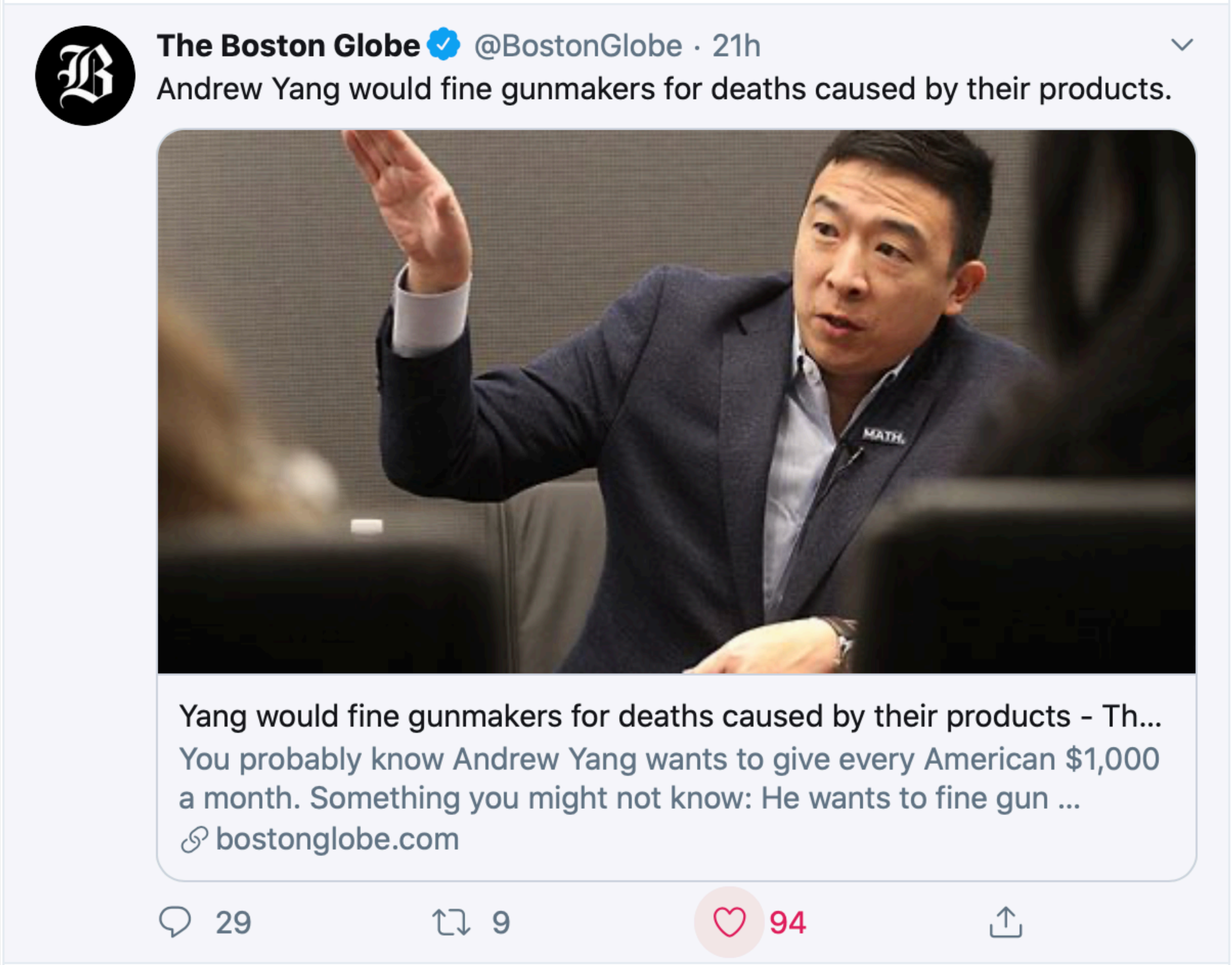
4,022 people are talking about this



MELANIA TRUMP liked your Tweet

Seems the only #Wall @realDonaldTrump's built is the one between him and @FLOTUS #Melania #trump pic.twitter.com/XiNd2jiLUF

how Twitter resolved the conceptual flaw



The Boston Globe @BostonGlobe · 21h

Andrew Yang would fine gunmakers for deaths caused by their products.

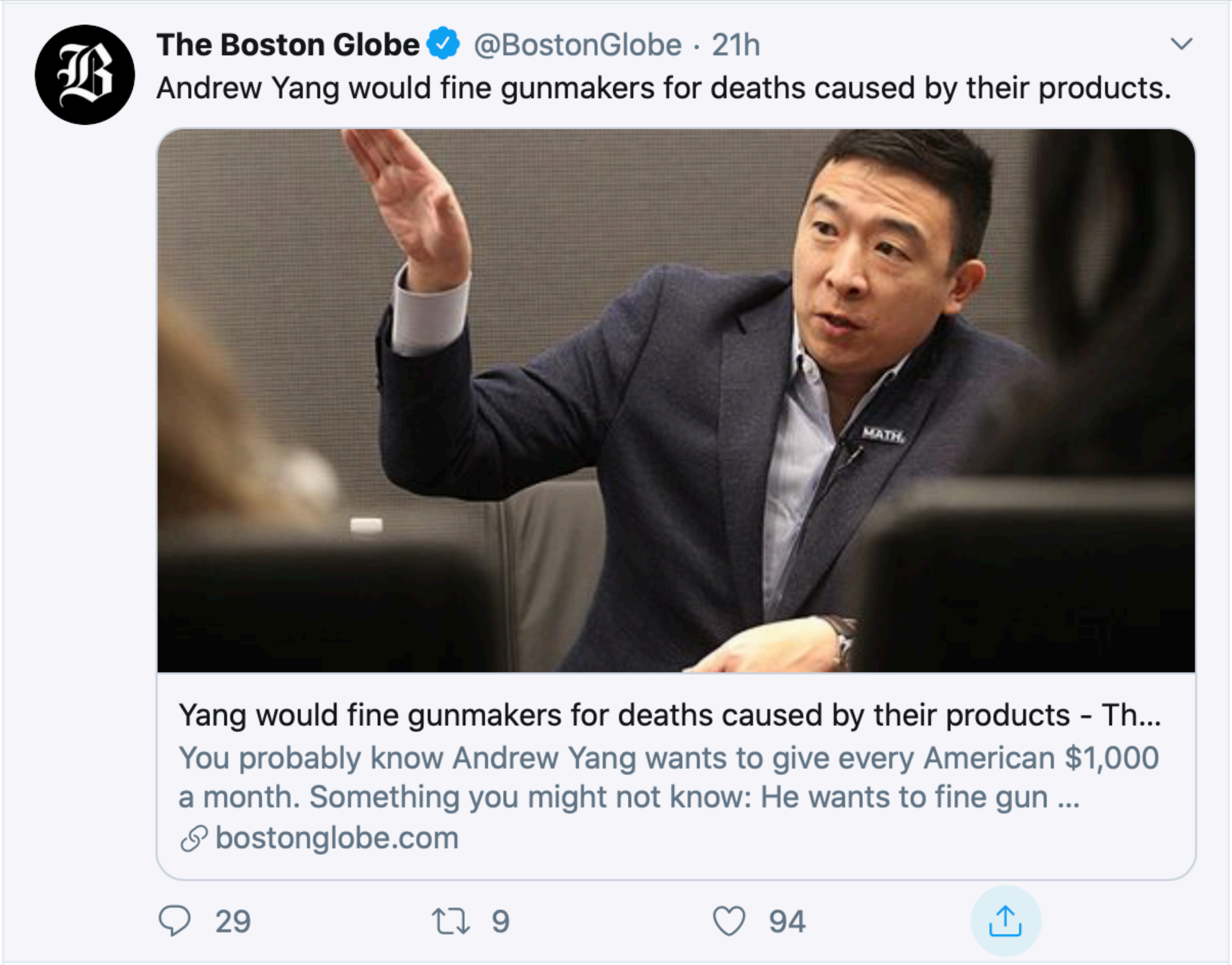
Yang would fine gunmakers for deaths caused by their products - Th...

You probably know Andrew Yang wants to give every American $1,000 a month. Something you might not know: He wants to fine gun ...

bostonglobe.com

29 9 94

Like: public



The Boston Globe @BostonGlobe · 21h

Andrew Yang would fine gunmakers for deaths caused by their products.

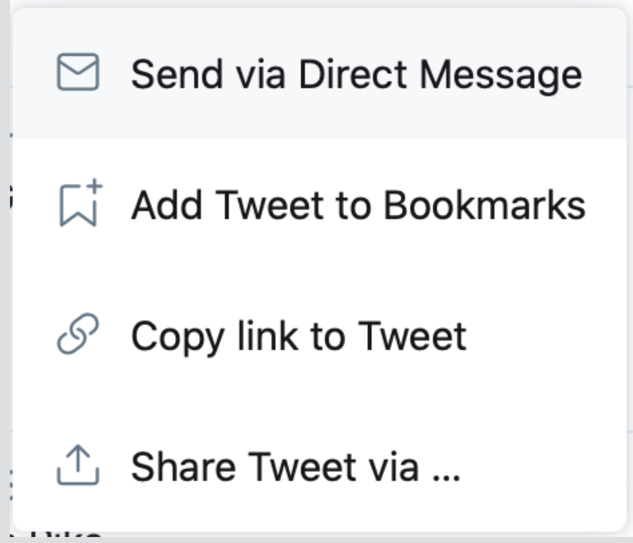
Yang would fine gunmakers for deaths caused by their products - Th...

You probably know Andrew Yang wants to give every American $1,000 a month. Something you might not know: He wants to fine gun ...

bostonglobe.com

29 9 94

Bookmark: private



Send via Direct Message

Add Tweet to Bookmarks

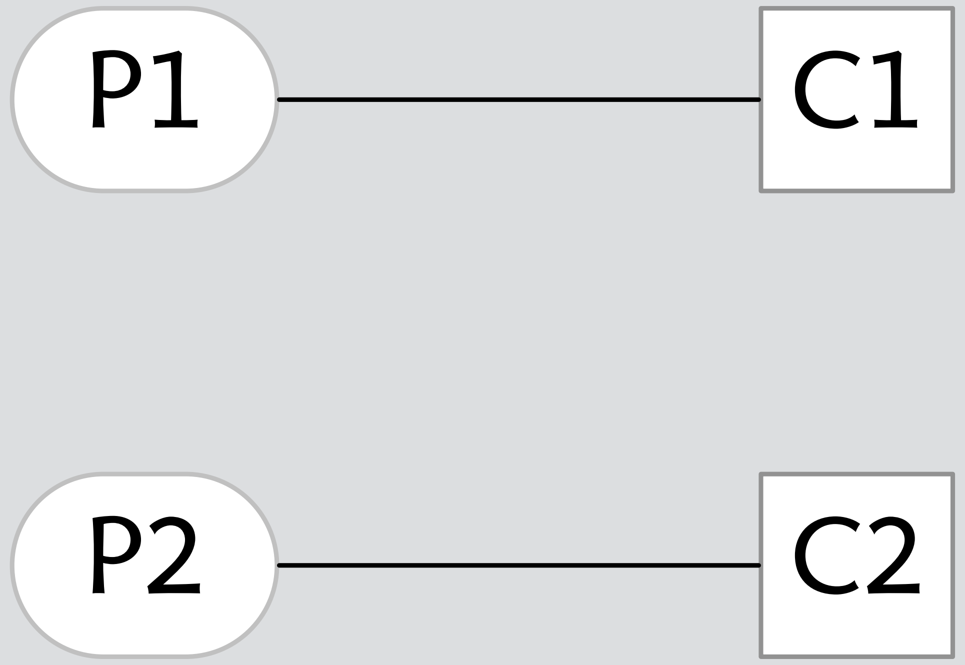
Copy link to Tweet

Share Tweet via ...

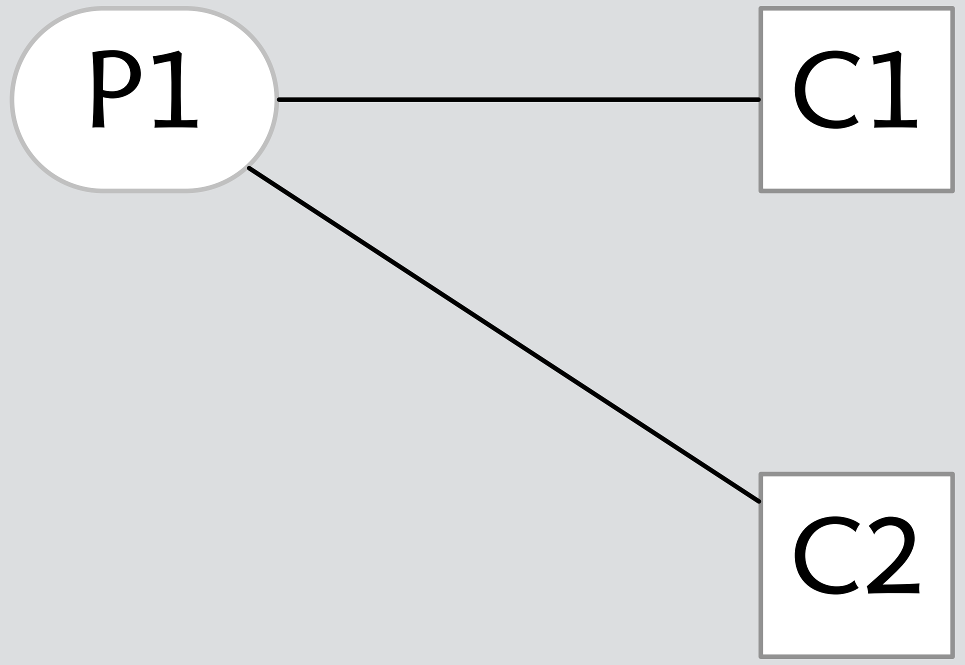
design rules

the specificity rule

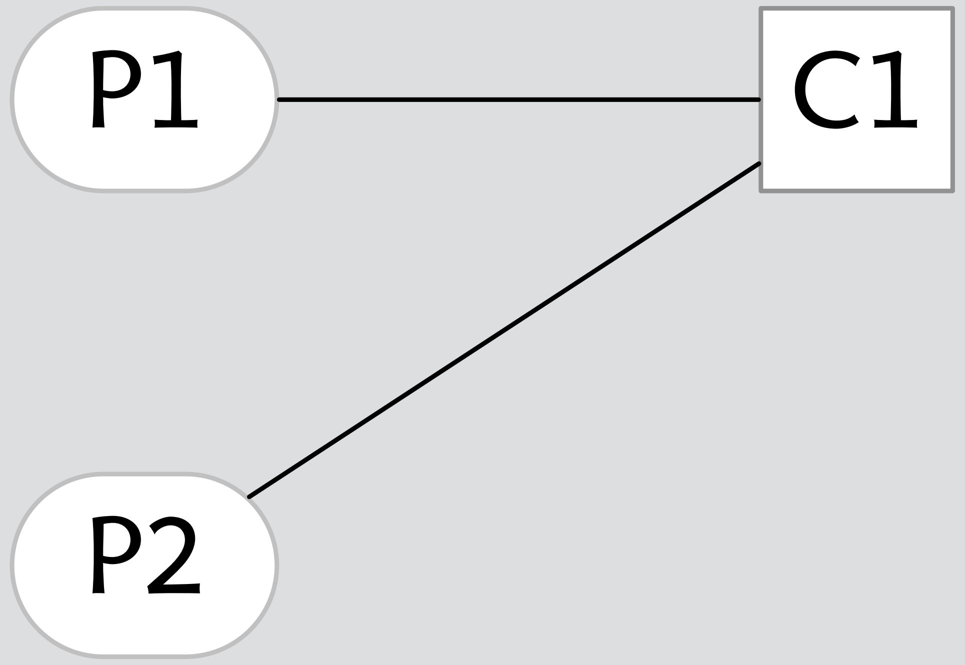
specificity
purposes:concepts are 1:1



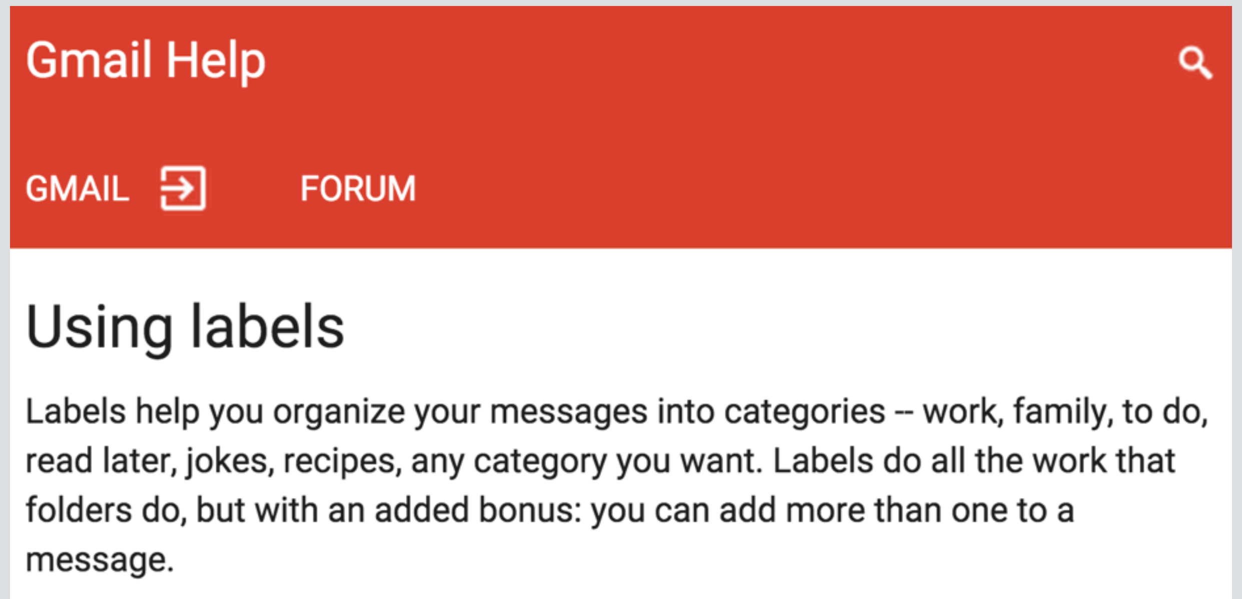
redundancy
>1 concept per purpose



overloading
>1 purpose per concept



example
category vs label in Gmail

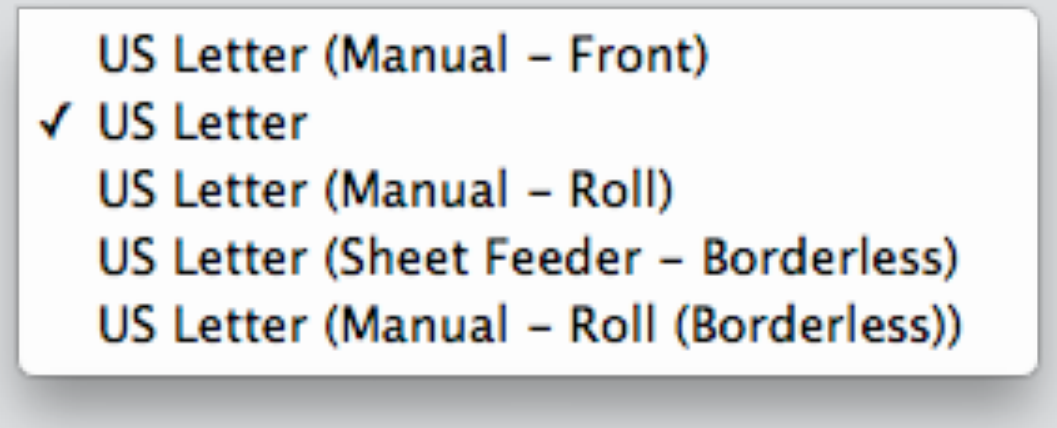

Gmail Help

GMAIL FORUM

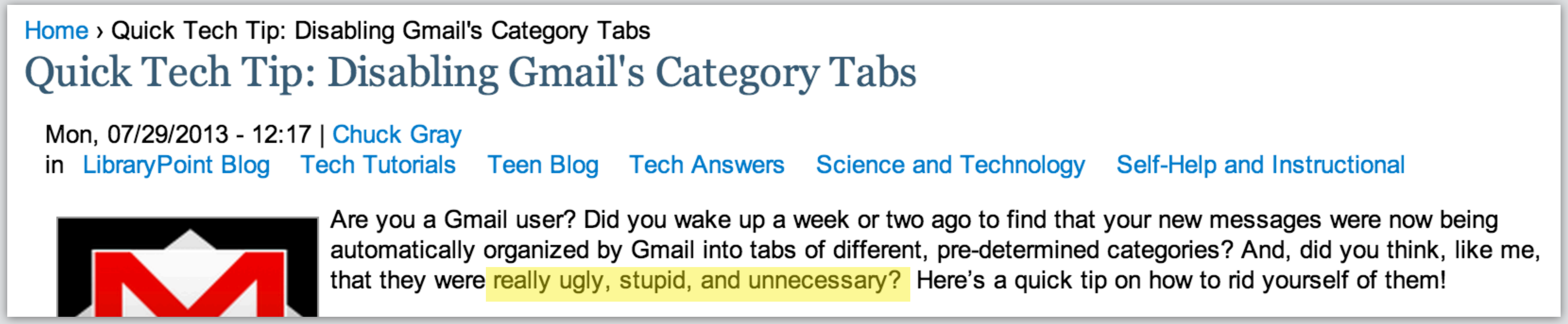
Using labels

Labels help you organize your messages into categories – work, family, to do, read later, jokes, recipes, any category you want. Labels do all the work that folders do, but with an added bonus: you can add more than one to a message.

example
page size vs feed in Epson


US Letter (Manual – Front)
✓ US Letter
US Letter (Manual – Roll)
US Letter (Sheet Feeder – Borderless)
US Letter (Manual – Roll (Borderless))

redundancy gmail categories



Home › Quick Tech Tip: Disabling Gmail's Category Tabs

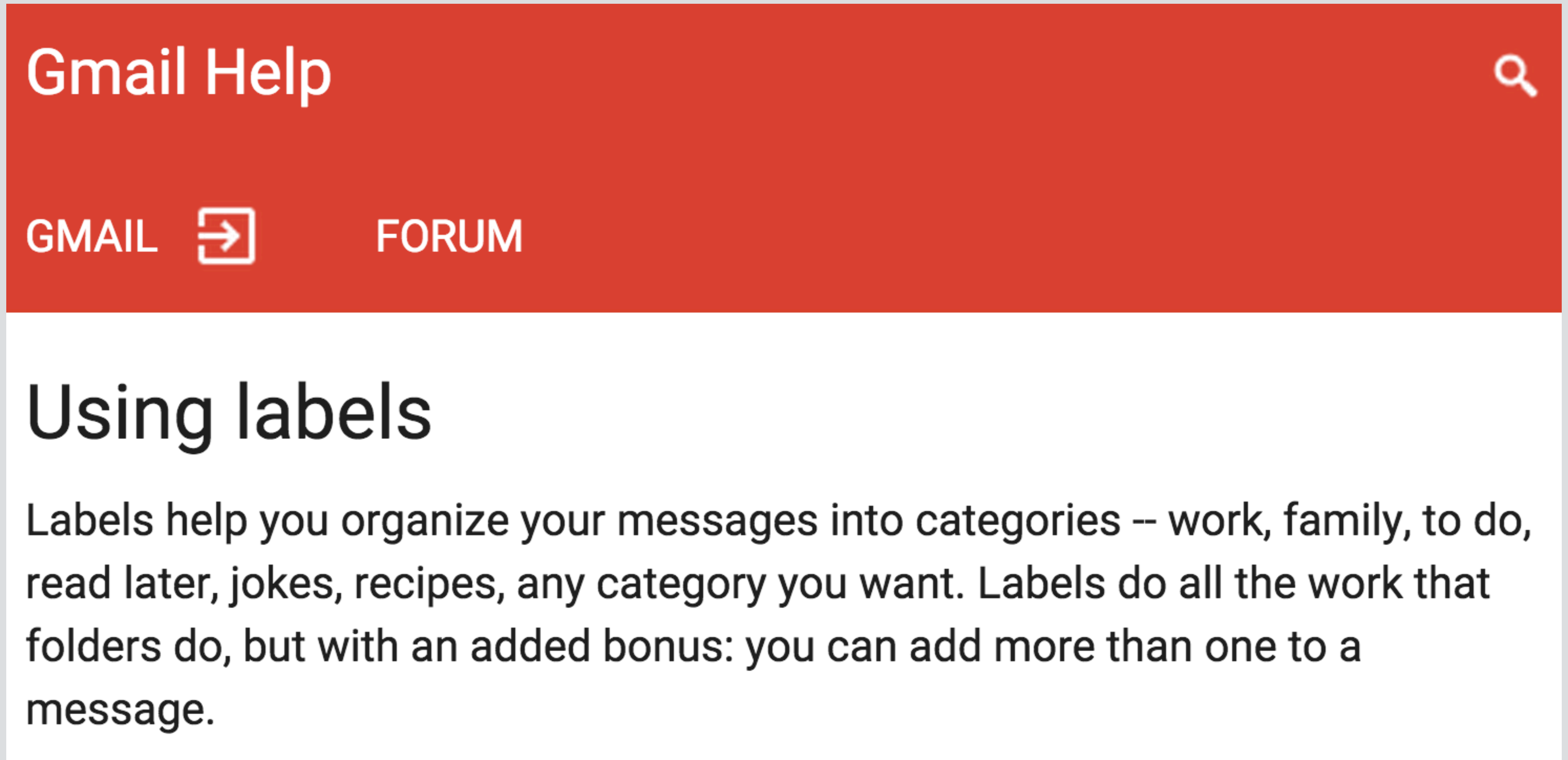
Quick Tech Tip: Disabling Gmail's Category Tabs

Mon, 07/29/2013 - 12:17 | Chuck Gray
in LibraryPoint Blog Tech Tutorials Teen Blog Tech Answers Science and Technology Self-Help and Instructional

Are you a Gmail user? Did you wake up a week or two ago to find that your new messages were now being automatically organized by Gmail into tabs of different, pre-determined categories? And, did you think, like me, that they were really ugly, stupid, and unnecessary? Here's a quick tip on how to rid yourself of them!

initial reaction to categories

redundancy gmail categories

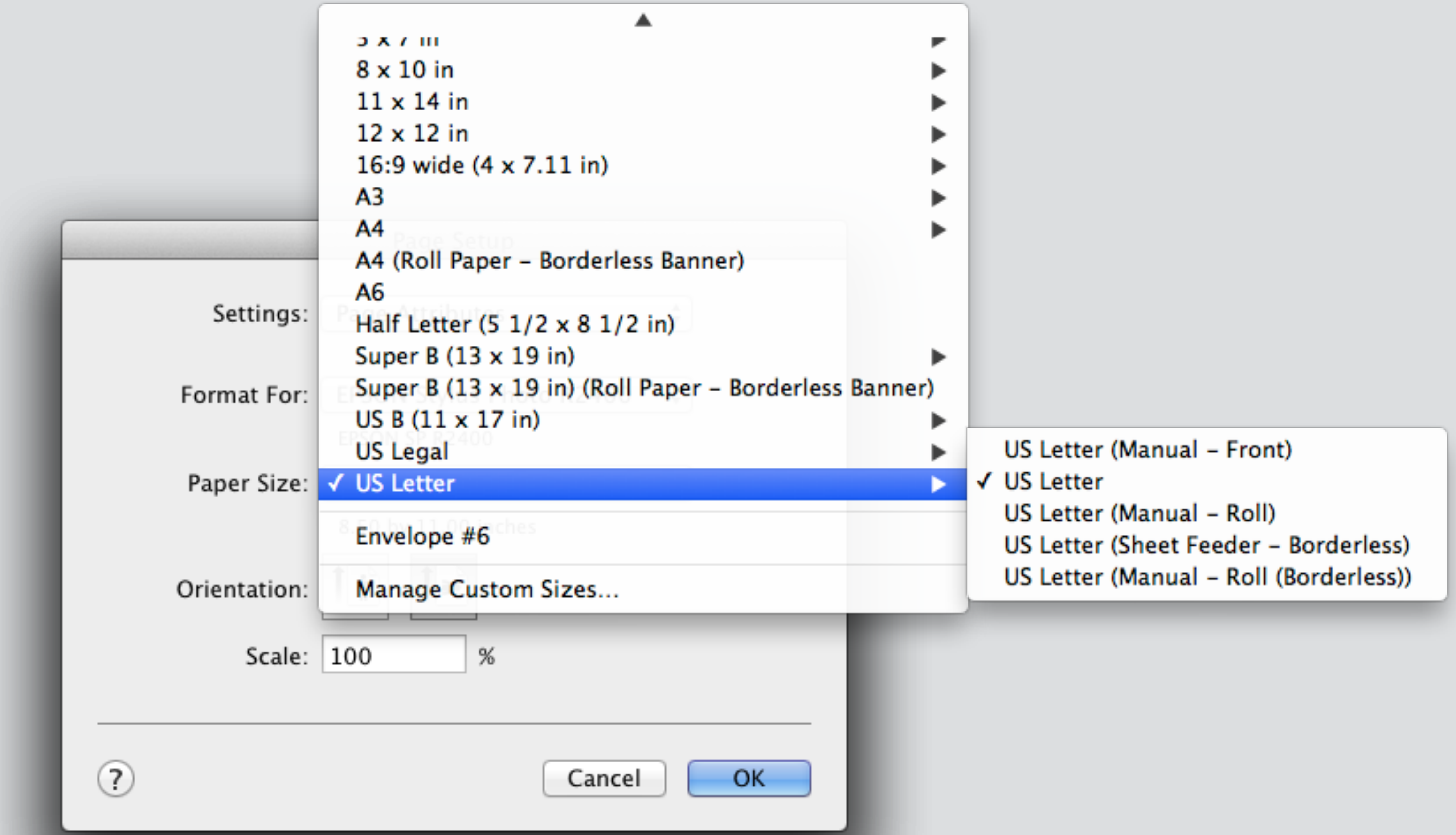


Using labels

Labels help you organize your messages into categories -- work, family, to do, read later, jokes, recipes, any category you want. Labels do all the work that folders do, but with an added bonus: you can add more than one to a message.

how Google explains labels

overloading epson driver

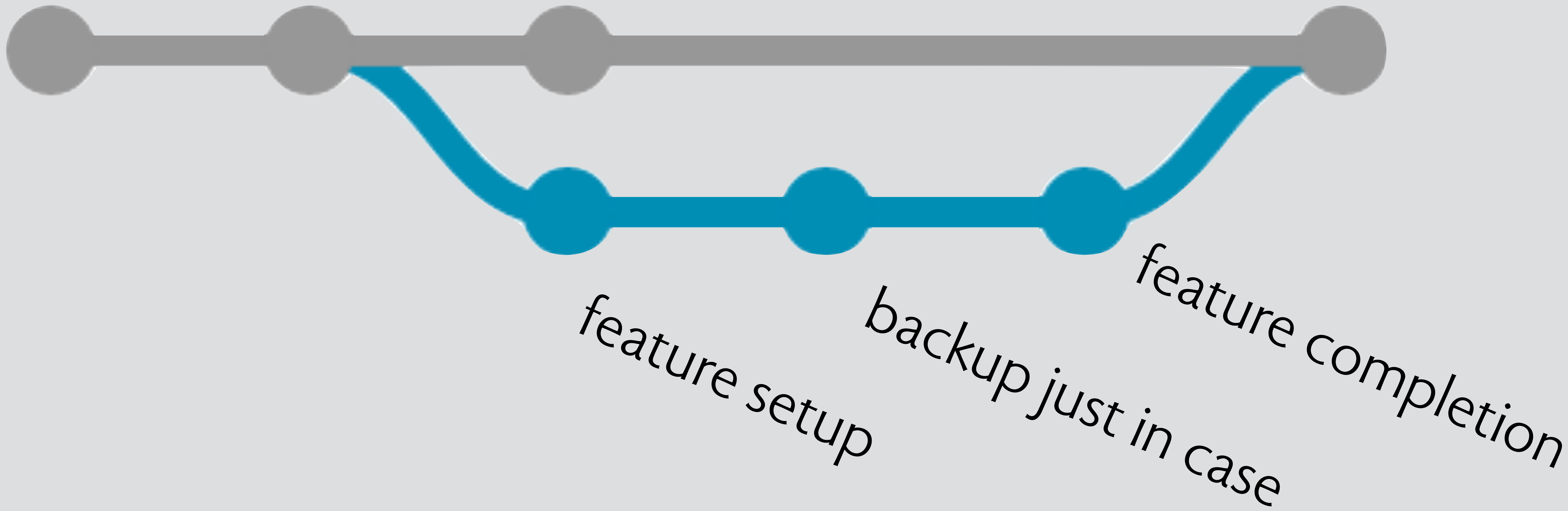

5 x 7 in
8 x 10 in
11 x 14 in
12 x 12 in
16:9 wide (4 x 7.11 in)
A3
A4
A4 (Roll Paper – Borderless Banner)
A6
Half Letter (5 1/2 x 8 1/2 in)
Super B (13 x 19 in)
Super B (13 x 19 in) (Roll Paper – Borderless Banner)
US B (11 x 17 in)
US Legal
✓ US Letter
Envelope #6
Manage Custom Sizes...
US Letter (Manual – Front)
✓ US Letter
US Letter (Manual – Roll)
US Letter (Sheet Feeder – Borderless)
US Letter (Manual – Roll (Borderless))
Settings:
Format For:
Orientation:
?
Cancel
OK

Paper Size: US Letter

Scale: 100 %

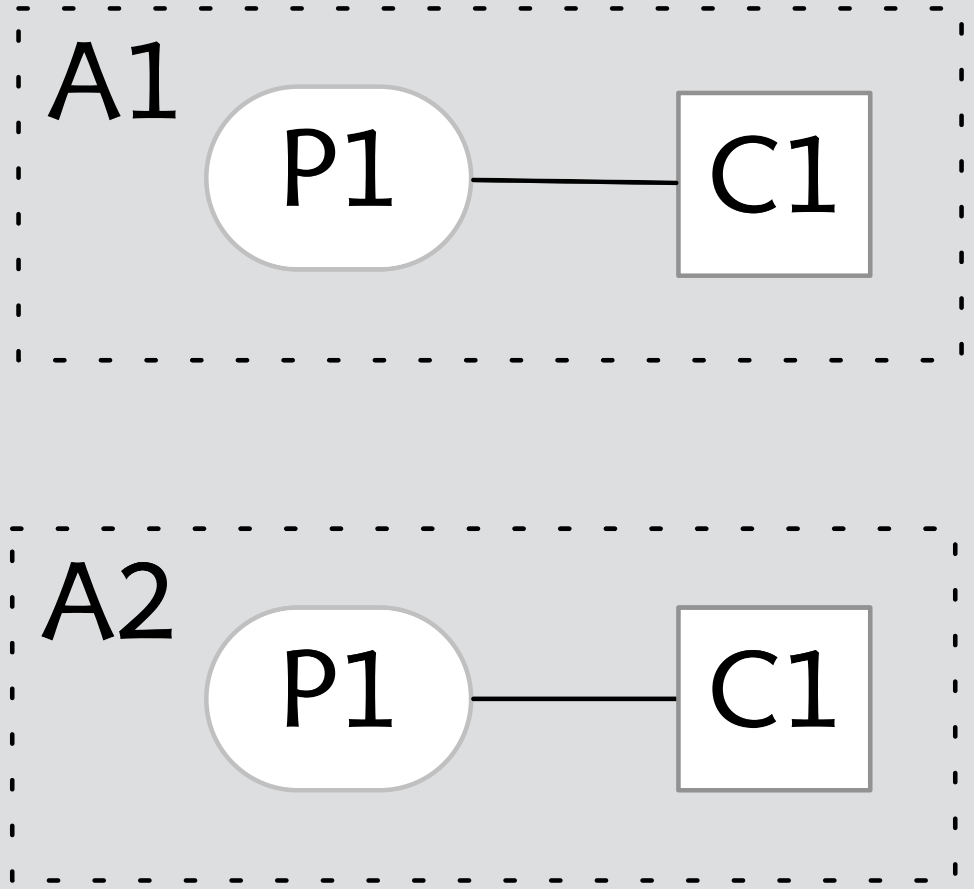
result: can’t create custom size for front loading
also, page size presets in Lightroom hold feed setting

overloading commit concept

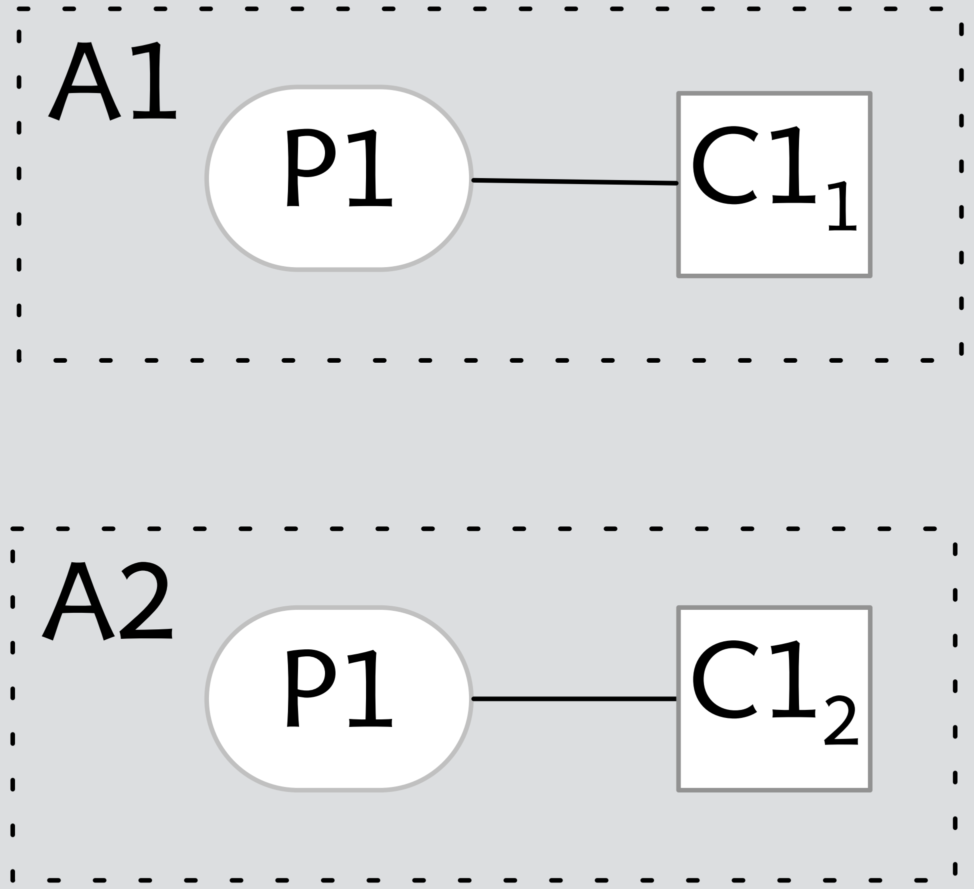


the familiarity rule

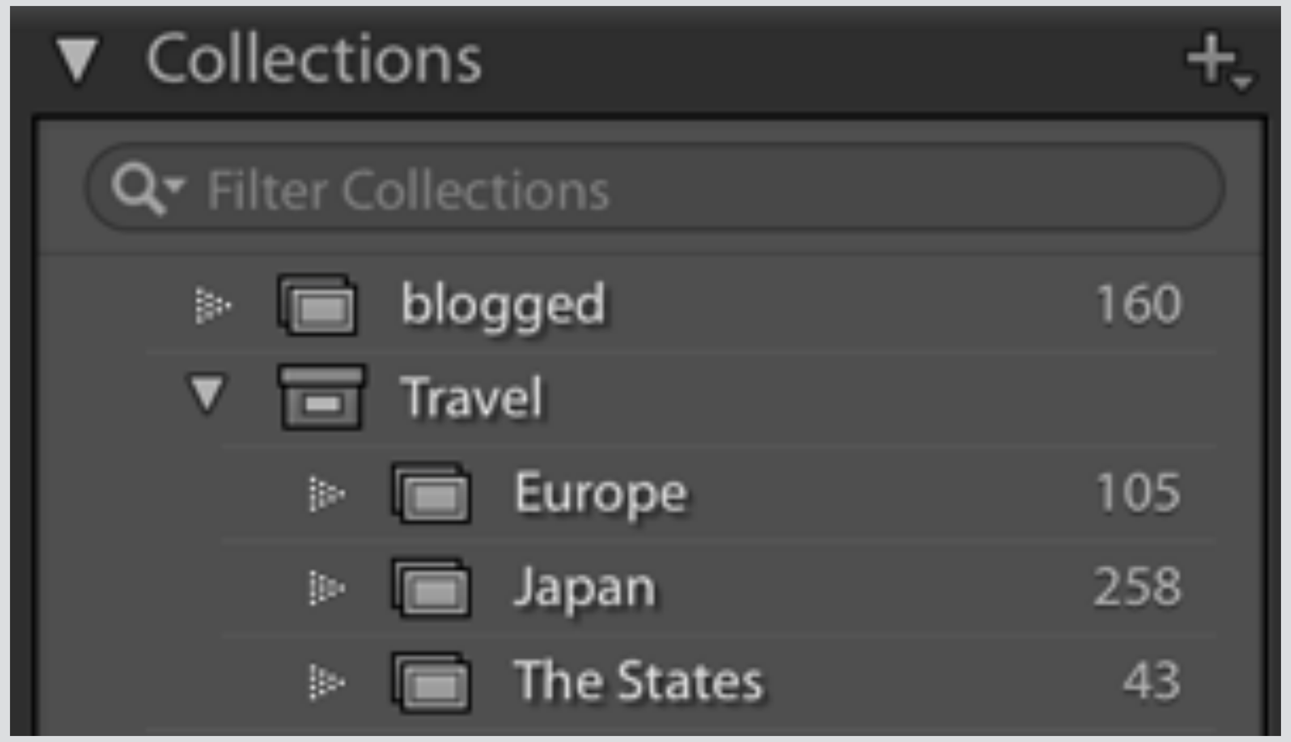
familiarity
steal, don't invent



needless specialization
custom concept, standard purpose



example
CollectionSet vs Folder in Lightroom

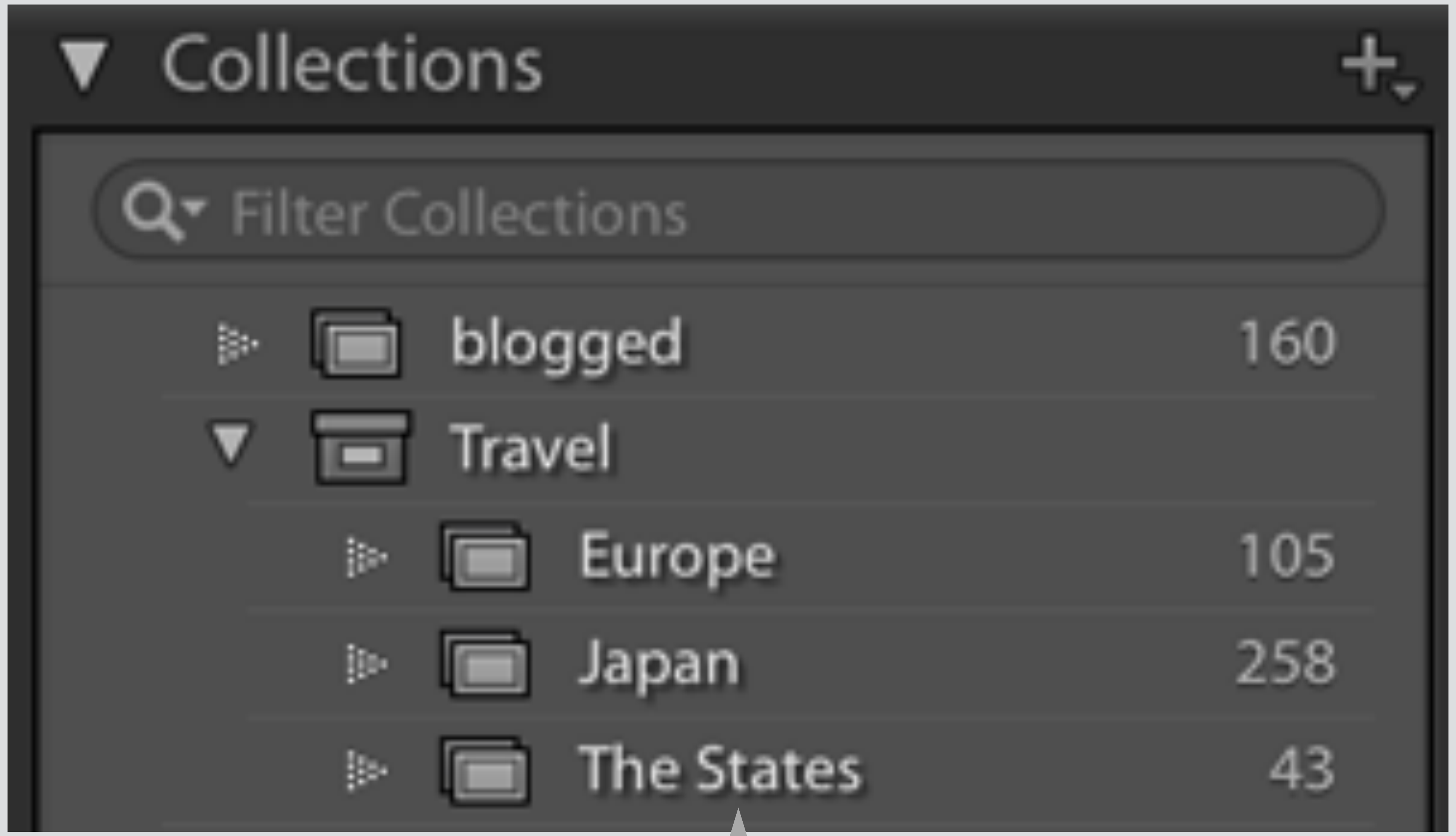


Collections

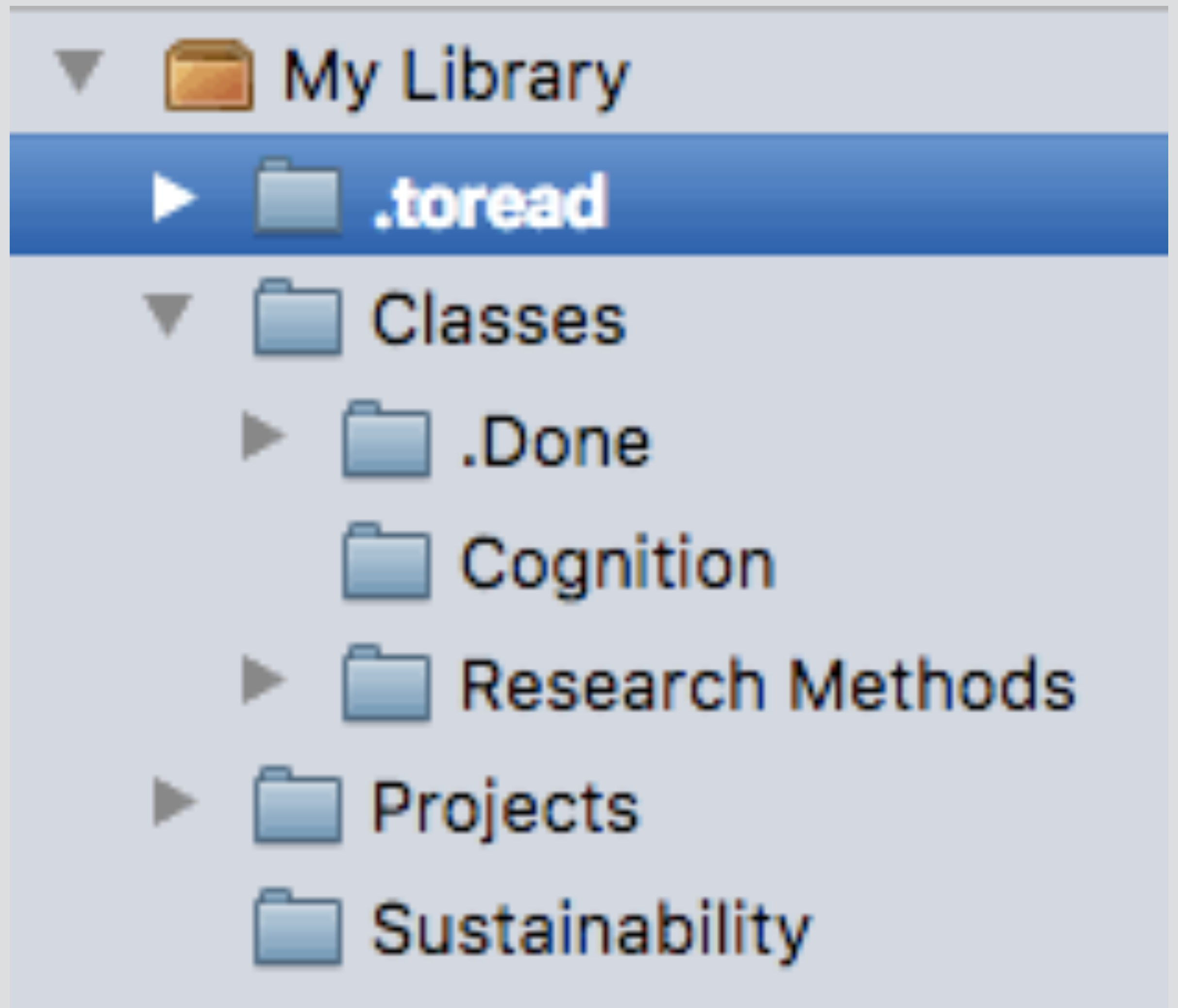
Filter Collections

blogged	160
Travel	
Europe	105
Japan	258
The States	43

familiarity Lightroom's collection (set) concept



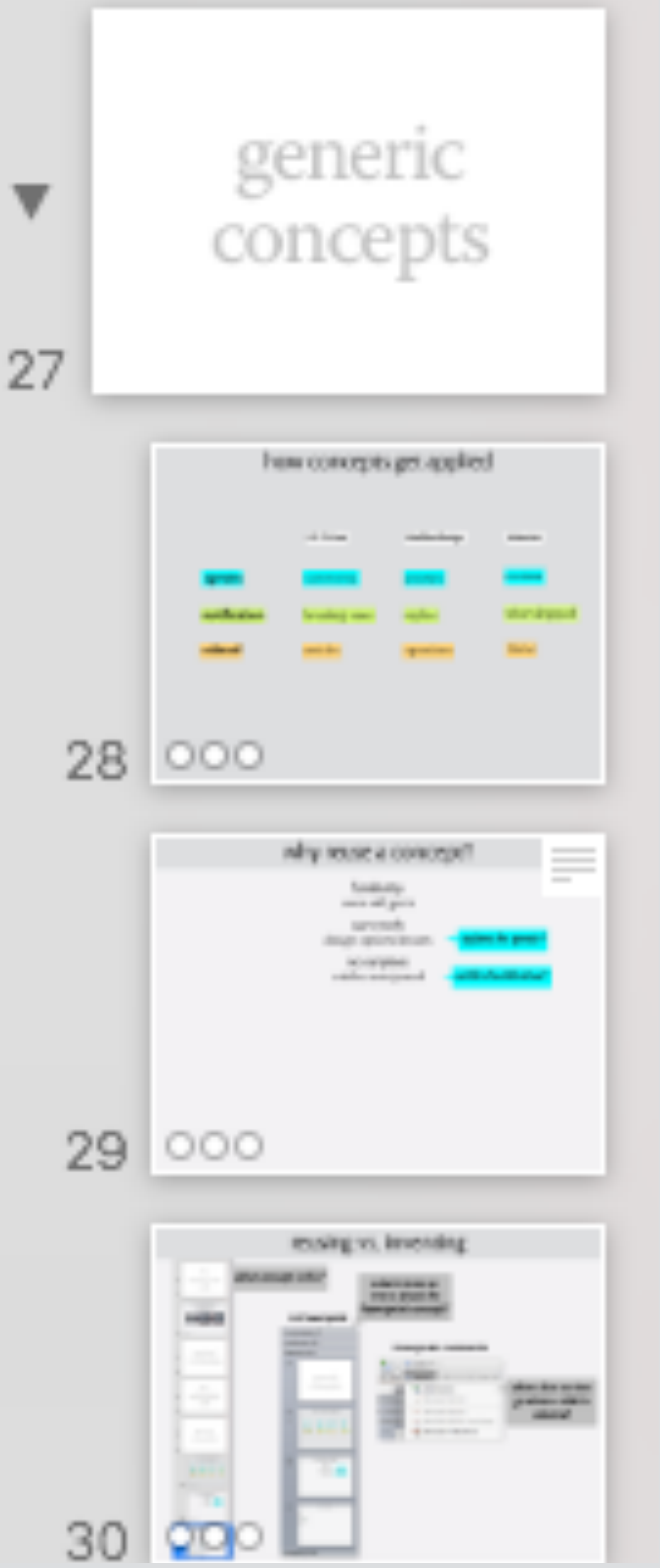
✗ Lightroom: only collection *sets* can contain collections



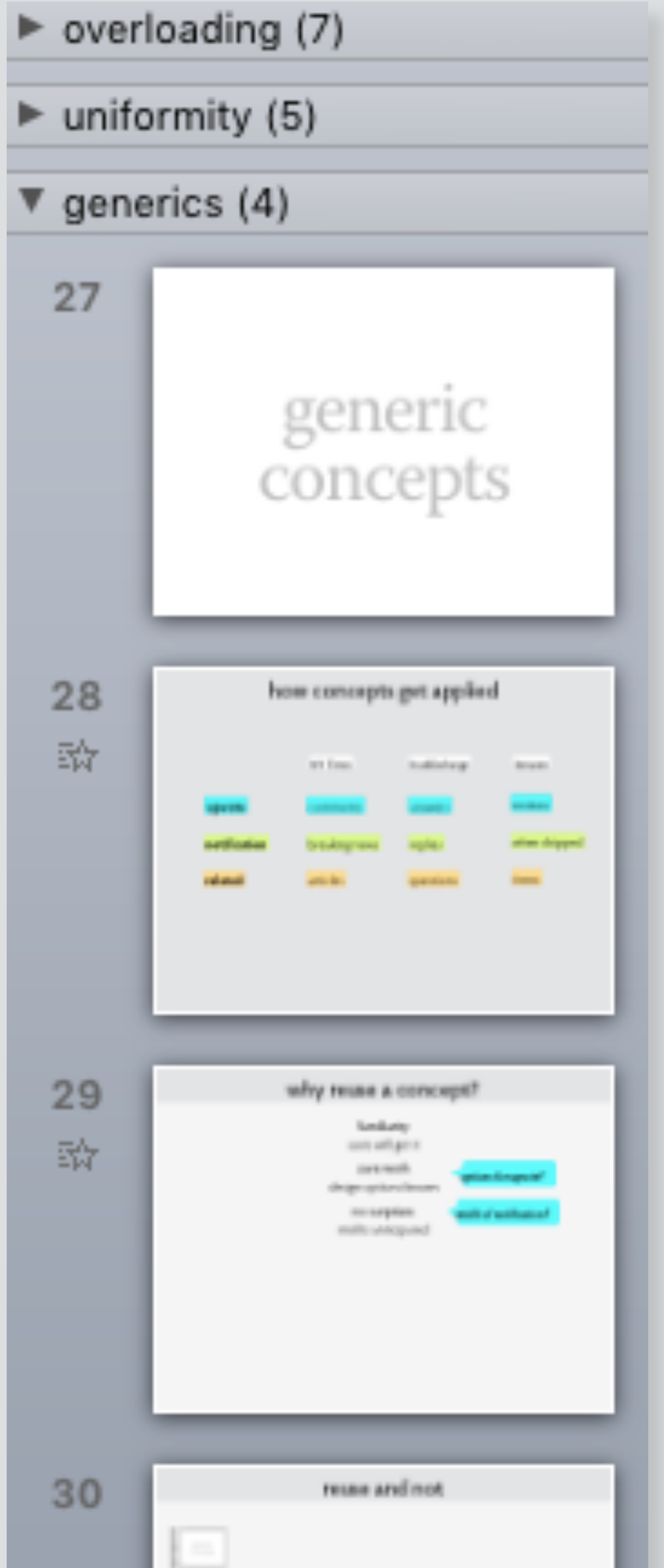
✓Zotero: collections can contain collections

familiarity Powerpoint's section concept

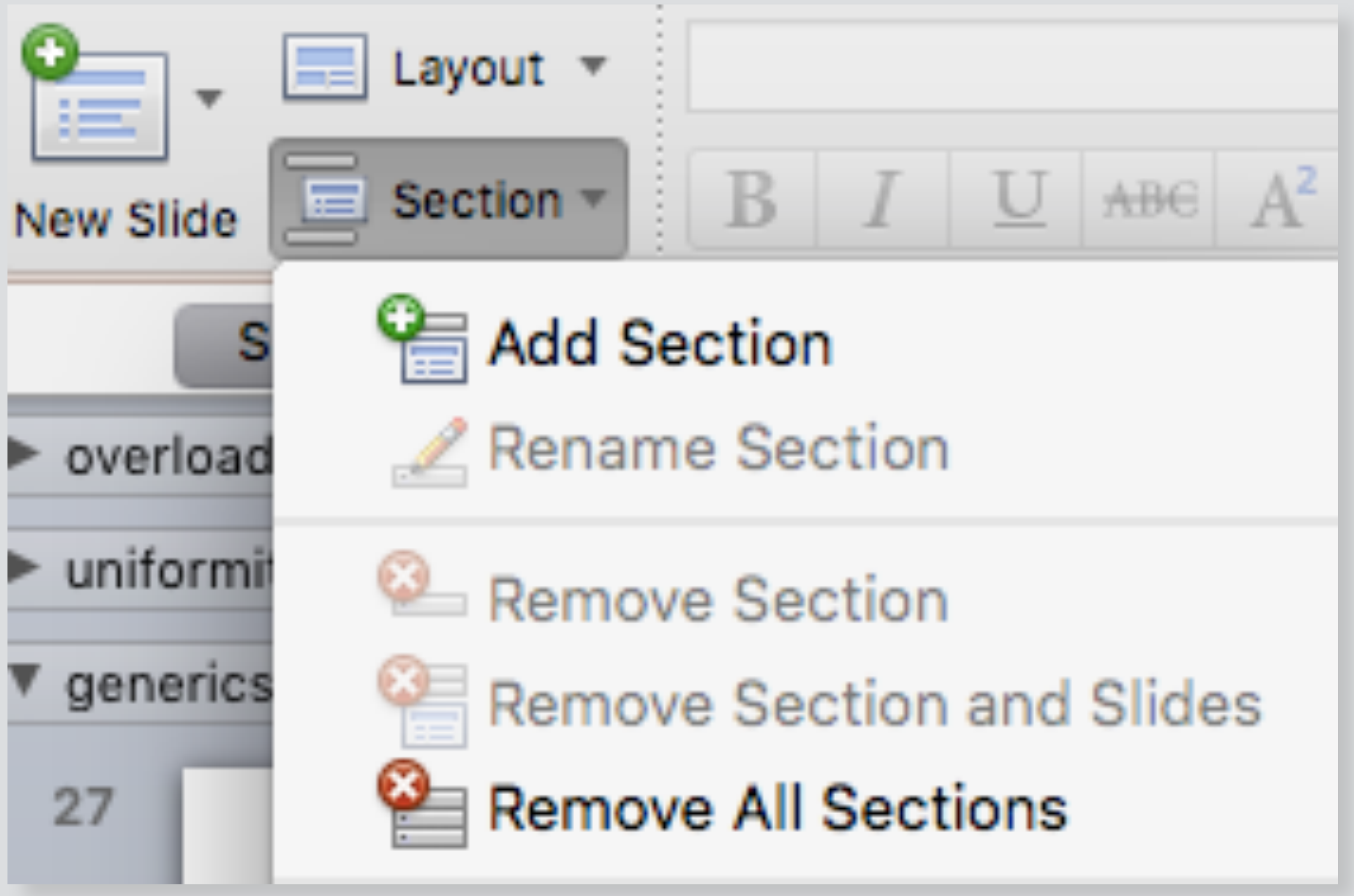
in Keynote



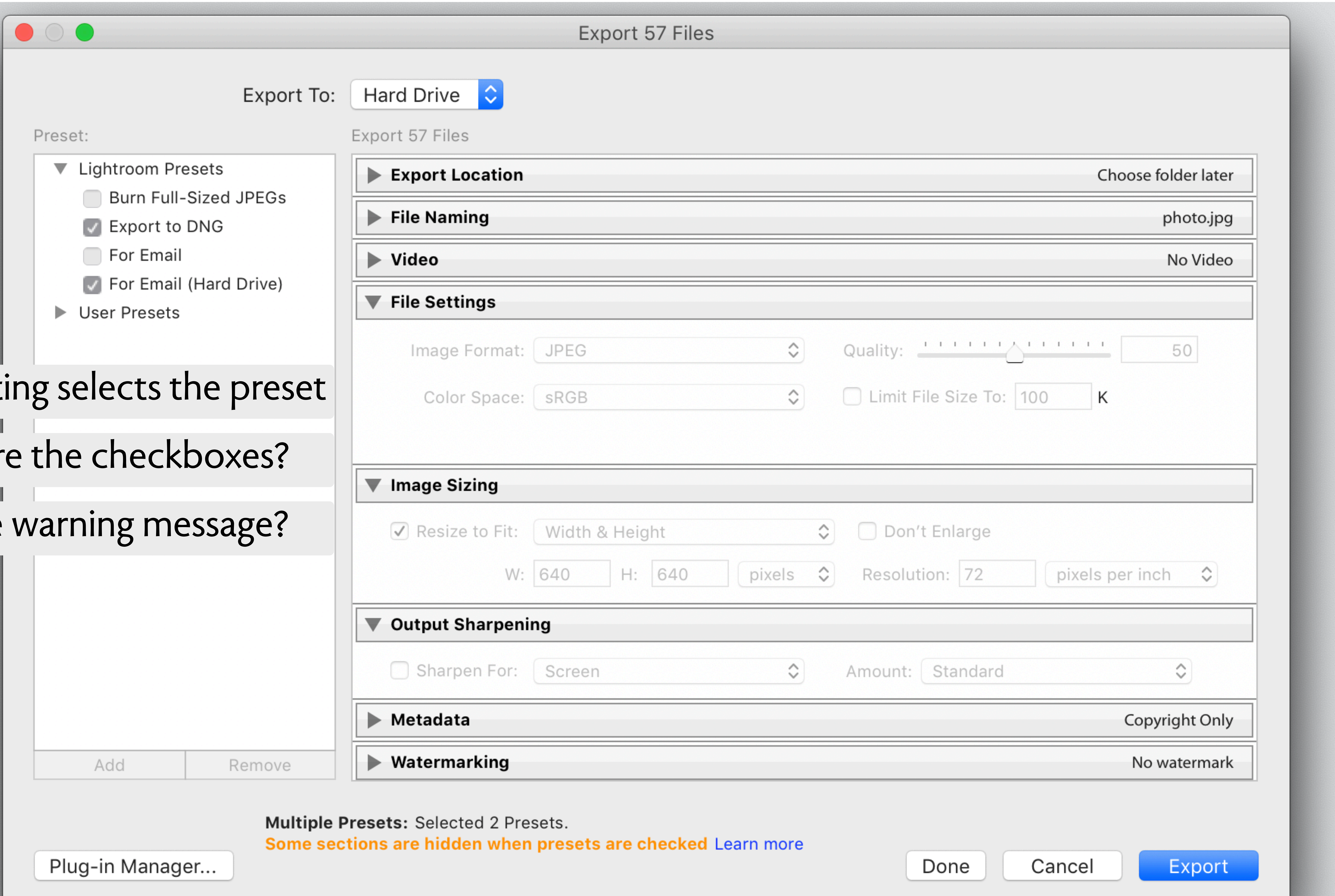
in Powerpoint



Powerpoint commands



familiarity Lightroom's export preset concept



Export 57 Files

Export To: Hard Drive

Preset:

- ▼ Lightroom Presets
  - □ Burn Full-Sized JPEGs
  - ☑ Export to DNG
  - □ For Email
  - ☑ For Email (Hard Drive)
- ▶ User Presets

Export 57 Files

▶ **Export Location** Choose folder later

▶ **File Naming** photo.jpg

▶ **Video** No Video

▼ **File Settings**

Image Format: JPEG Quality: 50

Color Space: sRGB □ Limit File Size To: 100 K

▼ **Image Sizing**

☑ Resize to Fit: Width & Height □ Don't Enlarge

W: 640 H: 640 pixels Resolution: 72 pixels per inch

▼ **Output Sharpening**

□ Sharpen For: Screen Amount: Standard

▶ **Metadata** Copyright Only

▶ **Watermarking** No watermark

Multiple Presets: Selected 2 Presets.

Some sections are hidden when presets are checked Learn more

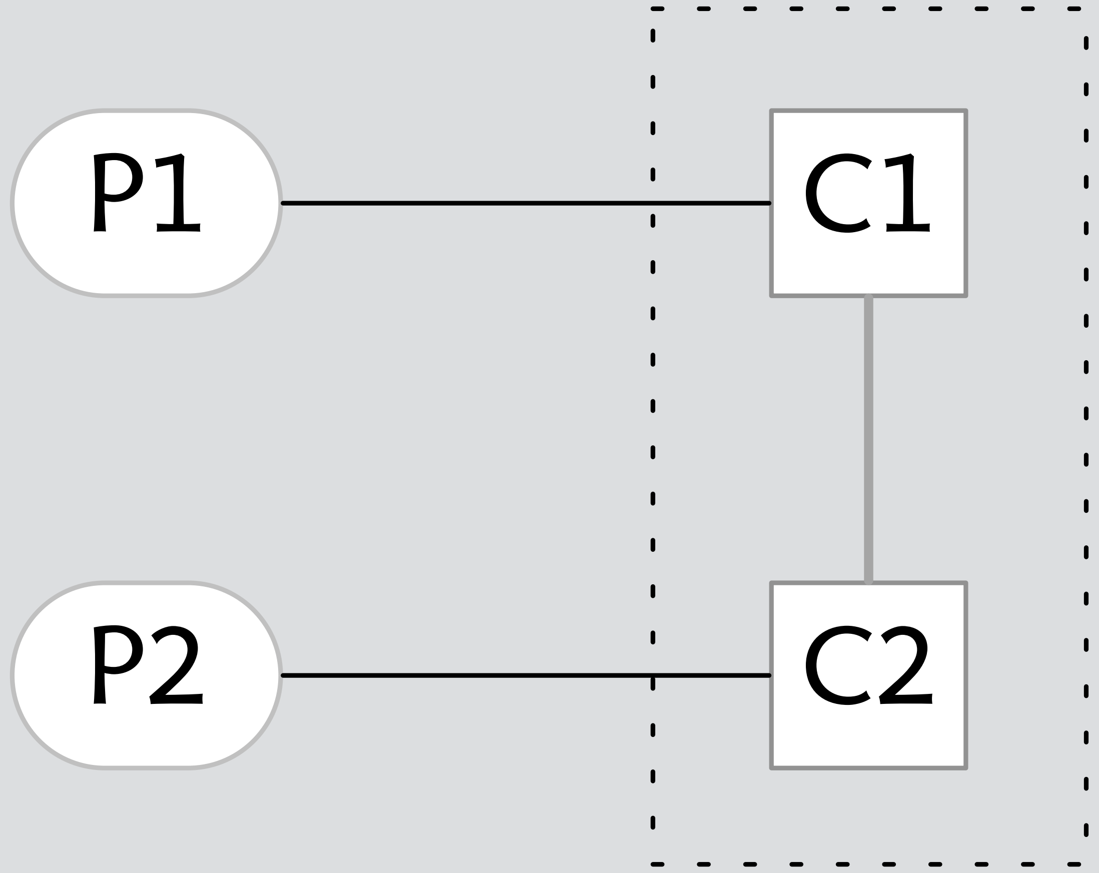
ok, highlighting selects the preset

huh, what are the checkboxes?

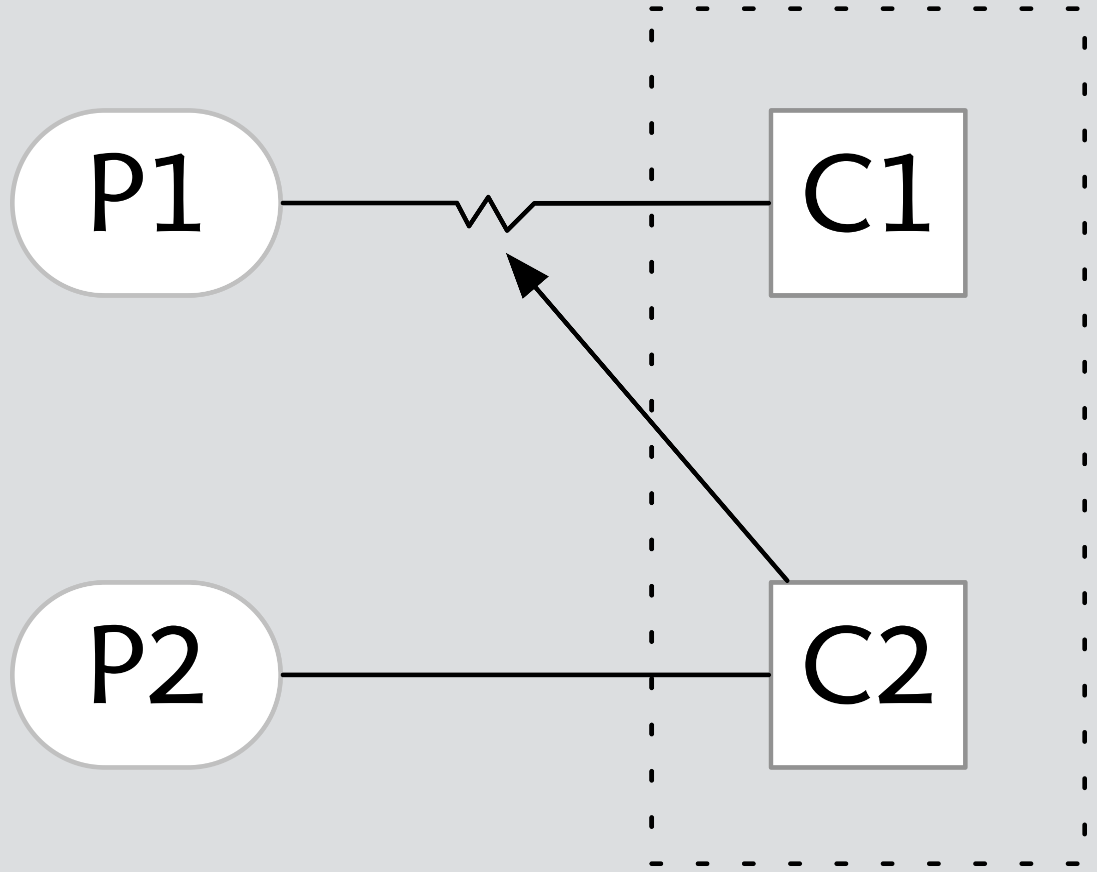
and why the warning message?

the integrity rule

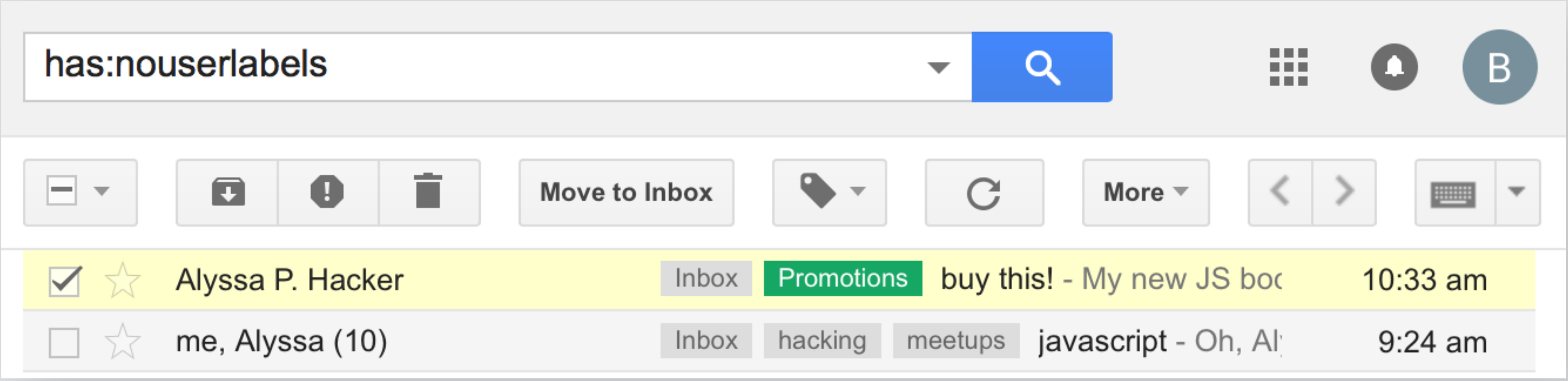
integrity
concepts safe when composed



interference
one concept breaks another



example
Label broken by Conversation in Gmail

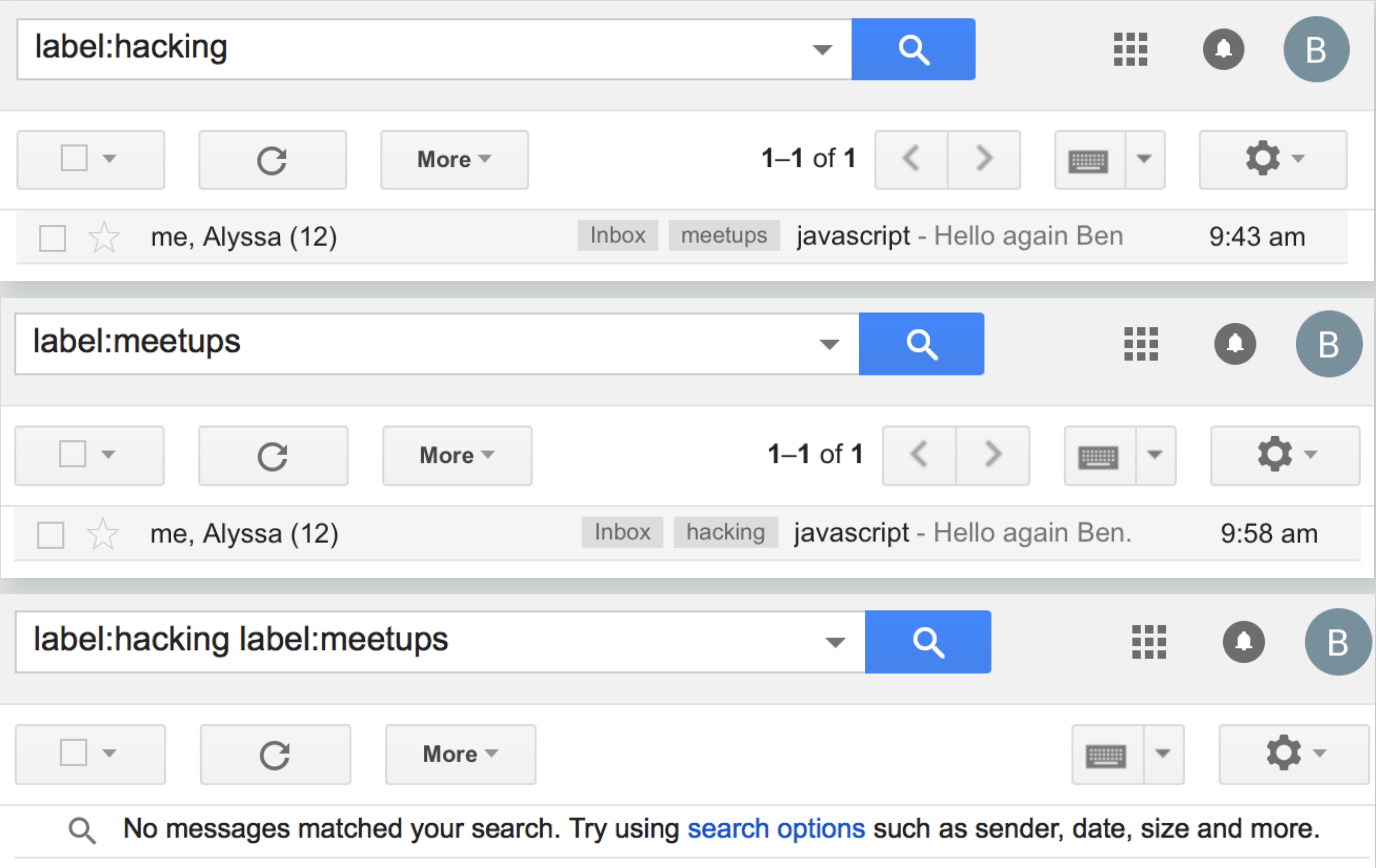


has:nouserlabels

☑ ☆ Alyssa P. Hacker Inbox Promotions buy this! - My new JS boo 10:33 am

□ ☆ me, Alyssa (10) Inbox hacking meetups javascript - Oh, Al 9:24 am

integrity Gmail conversation breaks label concept



label:hacking

1–1 of 1

□ ☆ me, Alyssa (12) Inbox meetups javascript - Hello again Ben 9:43 am

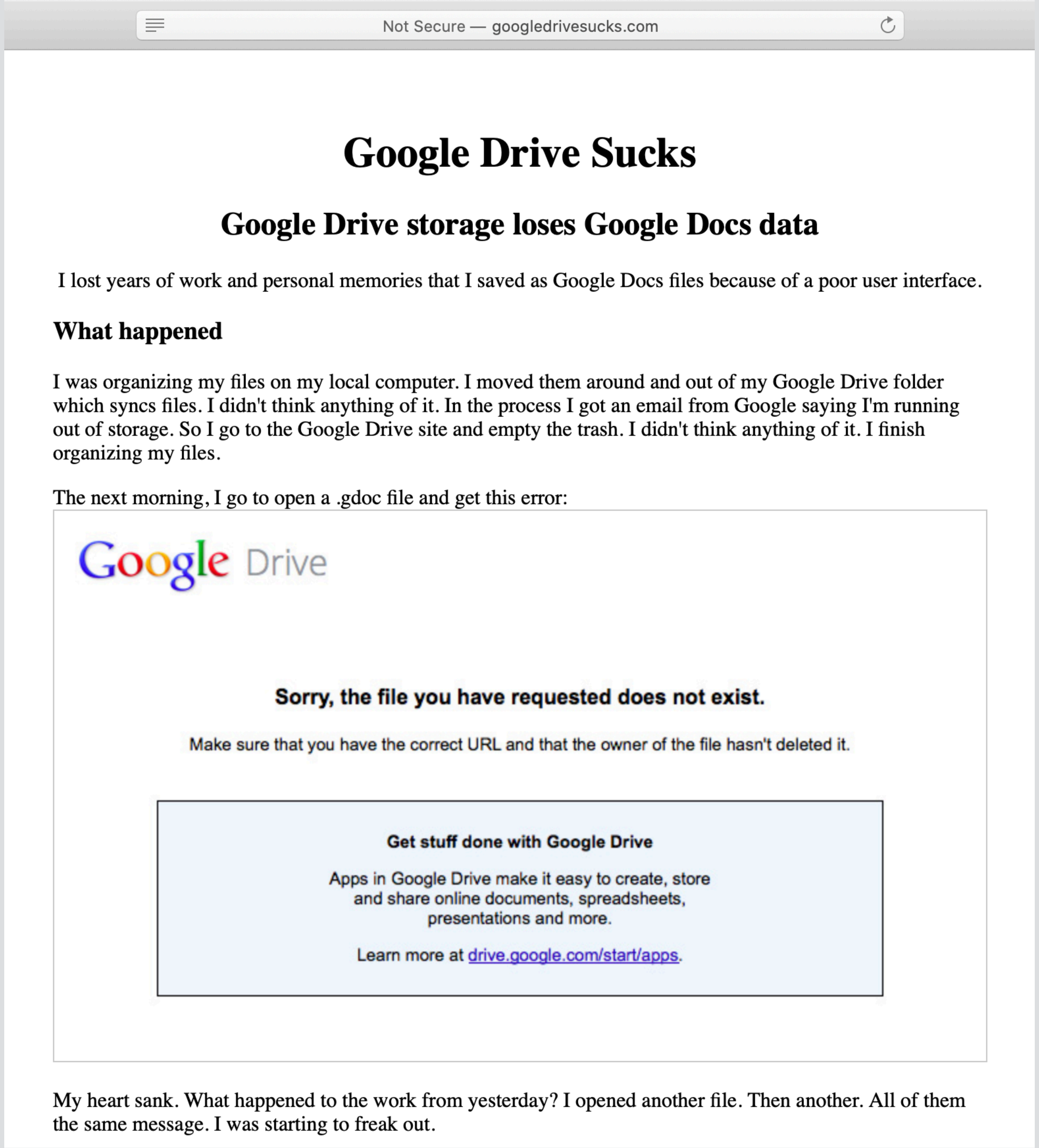
label:meetups

1–1 of 1

□ ☆ me, Alyssa (12) Inbox hacking javascript - Hello again Ben. 9:58 am

label:hacking label:meetups

No messages matched your search. Try using search options such as sender, date, size and more.



Google Drive Sucks

Google Drive storage loses Google Docs data

I lost years of work and personal memories that I saved as Google Docs files because of a poor user interface.

What happened

I was organizing my files on my local computer. I moved them around and out of my Google Drive folder which syncs files. I didn't think anything of it. In the process I got an email from Google saying I'm running out of storage. So I go to the Google Drive site and empty the trash. I didn't think anything of it. I finish organizing my files.

The next morning, I go to open a .gdoc file and get this error:

Sorry, the file you have requested does not exist.

Make sure that you have the correct URL and that the owner of the file hasn't deleted it.

Get stuff done with Google Drive

Apps in Google Drive make it easy to create, store and share online documents, spreadsheets, presentations and more.

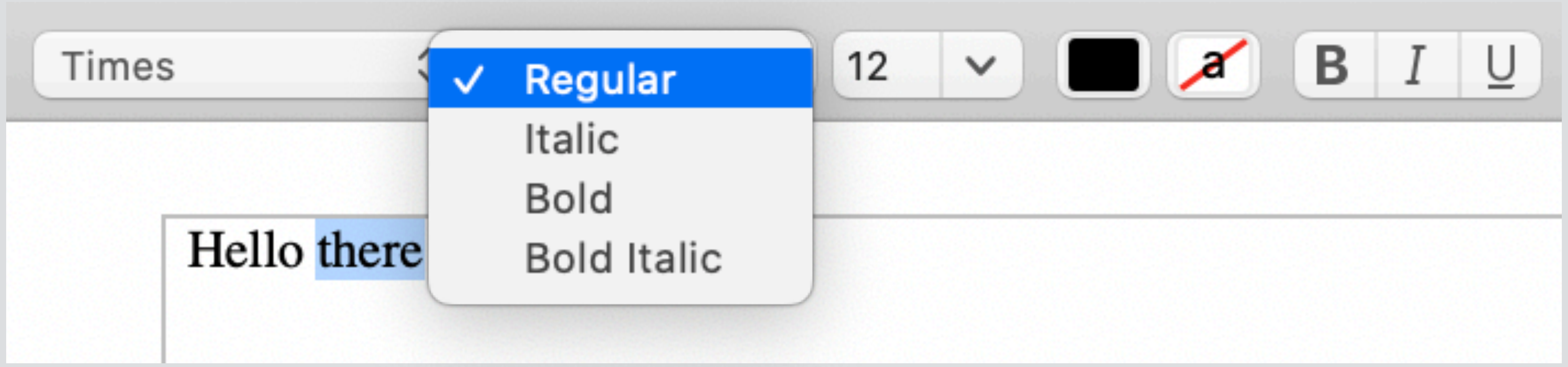
Learn more at drive.google.com/start/apps.

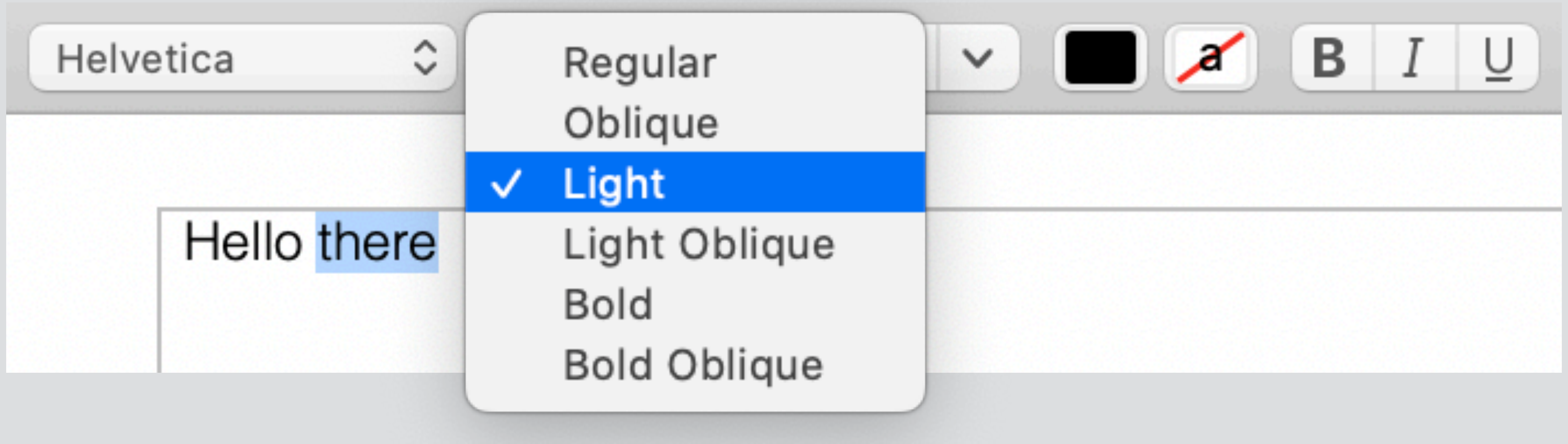
My heart sank. What happened to the work from yesterday? I opened another file. Then another. All of them the same message. I was starting to freak out.

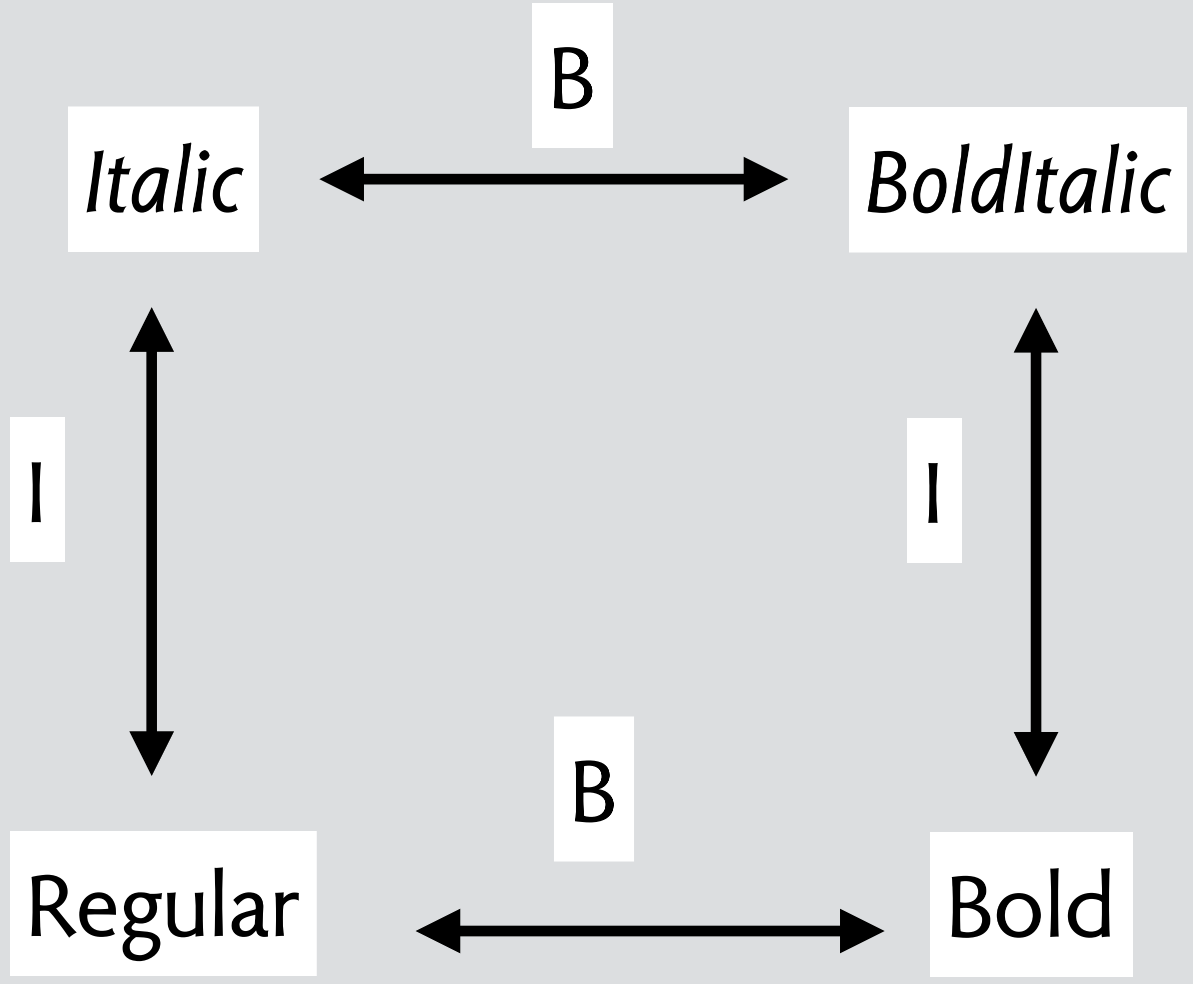
integrity cloudapp breaks sync concept

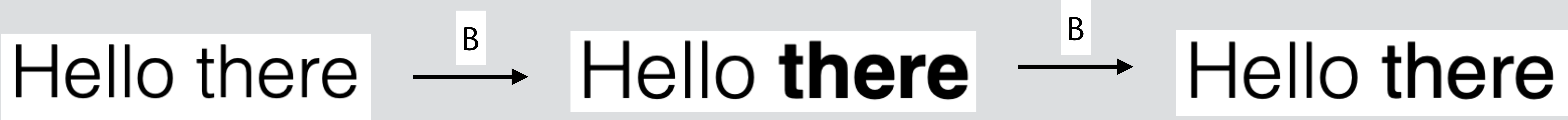
Google drive in cloud
slides.gdoc
slides.pdf
Google drive on client machine
slides.gdoc
url
slides.pdf
Another directory on client machine
move
Google drive in cloud
slides.gdoc
slides.pdf
Google drive on client machine
Another directory on client machine
slides.gdoc
url
slides.pdf
sync
Google drive in cloud
Google drive on client machine
Another directory on client machine
slides.gdoc
url
slides.pdf

integrity proFont breaks toggleFormat concept



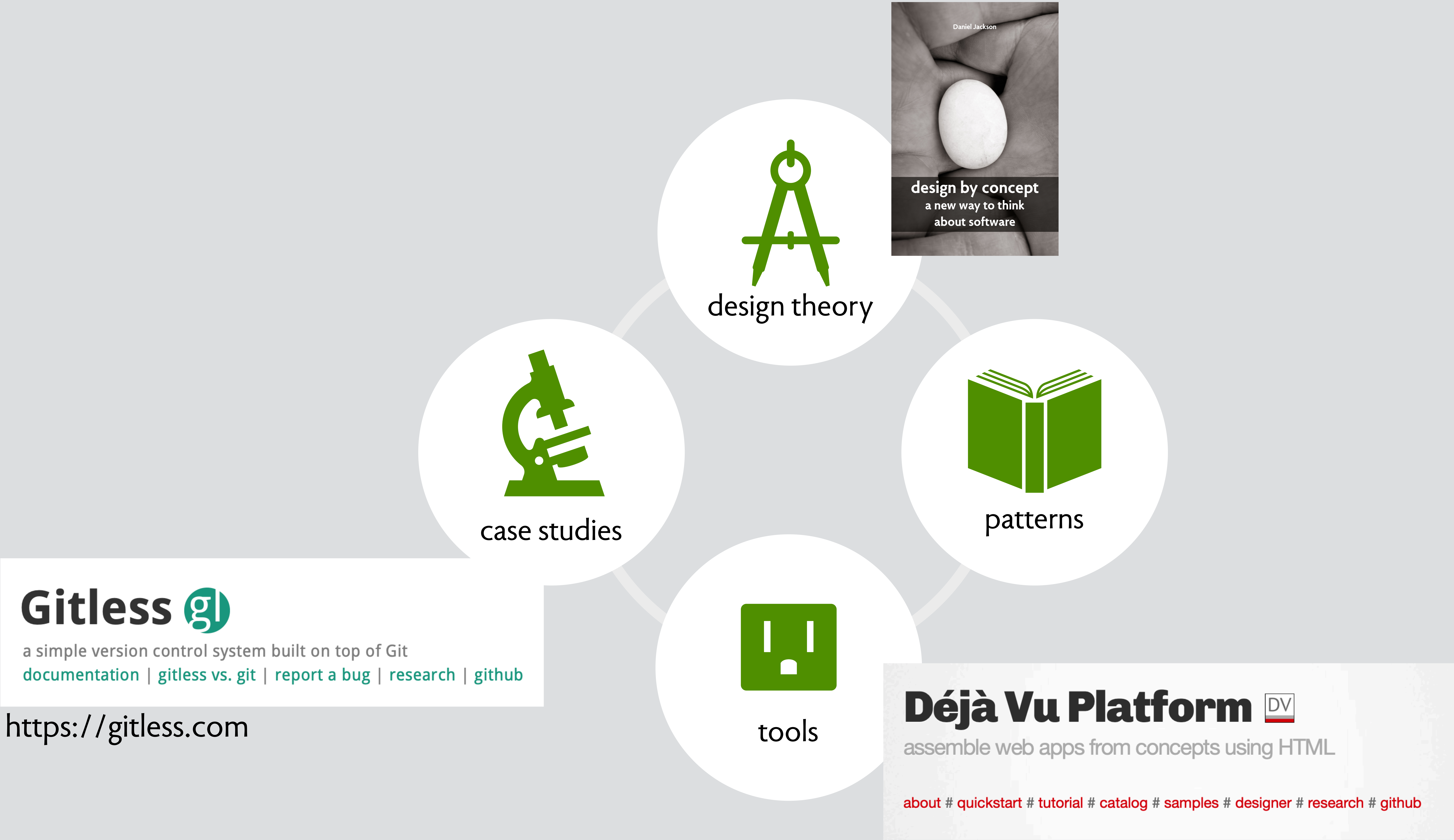






conclusions

a research & teaching program



design theory

case studies

patterns

tools

Daniel Jackson

design by concept
a new way to think
about software

Gitless

a simple version control system built on top of Git

documentation | gitless vs. git | report a bug | research | github

https://gitless.com

Déjà Vu Platform

assemble web apps from concepts using HTML

about # quickstart # tutorial # catalog # samples # designer # research # github

https://deja-vu-platform.com

some research challenges

formalizing design criteria
genericity, uniformity, decoupling

smooth transition to code
new architectures, like microservices

design language
an extension of Alloy? a logic for OPs

stay in touch!

register here for updates about the book etc:

https://tinyurl.com/conceptdesignlist

extra slides

apps = {concepts}

software app = {concepts}

Finder (1984)

folder, trash

Word (1983)

paragraph, format, style

Photoshop (1988)

pixelarray, layer, mask

Facebook (2004)

update, friend, like

Drive (2012)

synchronization, sharing

Google Doc (2009)

edit (OT), cloud file

software app class = {concepts}

text editor (eg, Emacs)

line, buffer

word processor (eg, Word)

paragraph,
format, style

desktop publisher (eg, Quark)

page, textflow

concept choices within an app class

sharing content
post/comment/repost

controlling access
friend/follow/group/channel

how you react
upvote/rating/reaction

personal organizing
favorite/bookmark

shared organizing
hashtag/mention/label

concepts for social media apps

comparing apps via concepts

	Lightroom	Photoshop	Capture One	Silver Efex
concepts for editing images?	**action** tool preset	adjustment **layer/mask** tool	adjustment **layer/mask** tool	filter/preset/style adjustment **control point**

inventory of concepts for a single app: Lightroom

Library

- Collection
- Folder
- VirtualCopy
- Metadata
- Flag
- Keyword
- Label
- Rating
- Selection

Develop

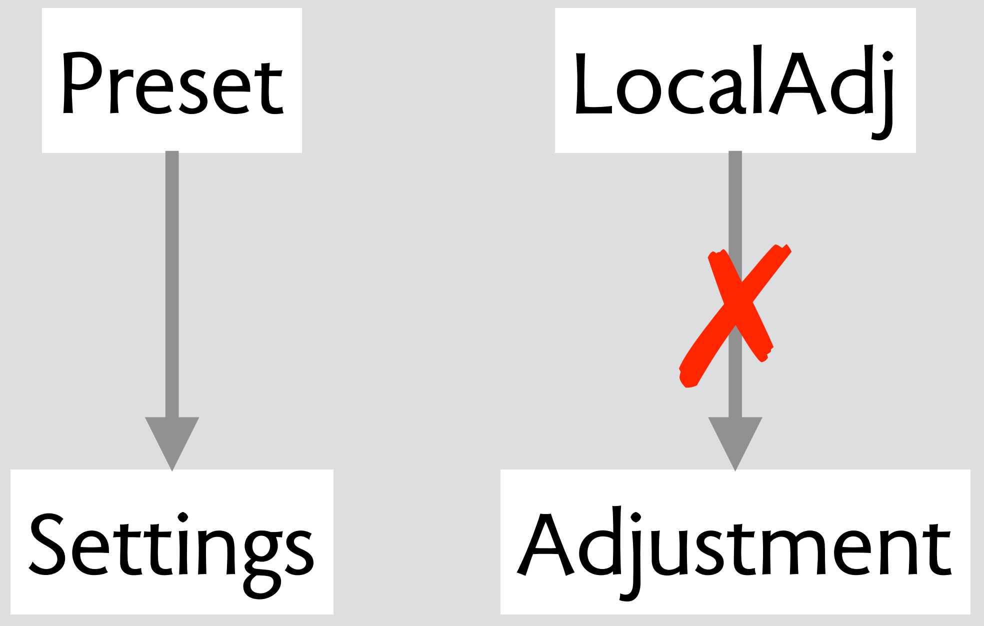
- Action
- Adjustment
- LocalAdj
- Preset
- Settings
- Snapshot

Printing

- Package
- Template
- Watermark

Undo

Preference



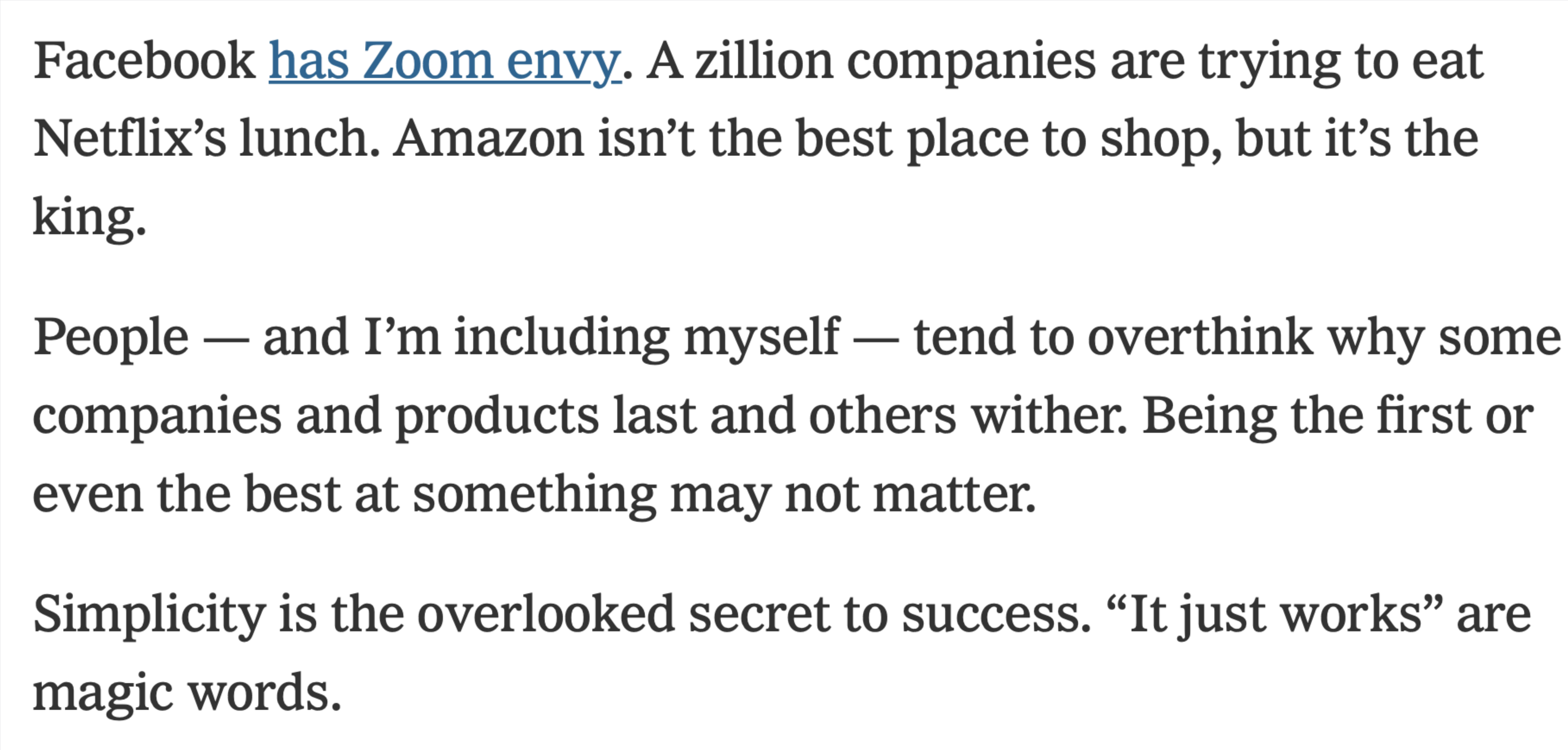
Preset → Settings

LocalAdj ✗ Adjustment

preset uses settings

software that
“just works”

software that “just works”



Facebook has Zoom envy. A zillion companies are trying to eat Netflix’s lunch. Amazon isn’t the best place to shop, but it’s the king.

People — and I’m including myself — tend to overthink why some companies and products last and others wither. Being the first or even the best at something may not matter.

Simplicity is the overlooked secret to success. “It just works” are magic words.

Shira Ovide, NYT, April 27, 2020

“just works” is not so easy

frictionless
unobtrusive
natural
learnable

predictable
robust
safe & secure
error-tolerant

powerful
capable
flexible
efficient

what it’s not about

cool technology: cloud, machine learning, blockchain

removing or preventing bugs in code

a theory of software design

structure
elements, relationships, composition

criteria
objective measures of goodness

patterns
capturing design experience

examples of theories

typography

structure

page, text block, margin
glyph, ligature, alternate
ascender, bowl, serif
justification, spacing, alignment

criteria

readability: x-height, line length
consistent color: italics not bold
avoiding widows & orphans

patterns

classic text block ratios
standard leading
serif/sans pairings

bread baking

structure

crust, interior, air pockets
fermenting & raising agents
flour varieties

criteria

shaping & elasticity
density & crumb
caramelization of crust

patterns

Lahey no-knead sourdough
Irish soda bread
pan cooked flat bread

software engineering

structure

function, module, package
closure, functional, callback
loop, iterator, stream

criteria

encapsulation of rep
simple interfaces
avoiding dependences

patterns

layered architecture
immutable datatype
model-view-controller
map/reduce/filter

concept structure & semantics

There is no problem
in computer science
that cannot be solved by introducing another level of indirection.
David Wheeler

concept Style

purpose consistent formatting

structure
defined: Style -> **one** Format
style: Element -> **one** Style
format: Element -> **one** Format = style.defined

actions
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s

principle
after define(s,f); assign(e1,s);
assign(e2,s); define(s,f’)
observe e1.format = e2.format = f’

name: essential for knowledge capture

purpose: why the concept exists

structure: localized data model

actions: observable & atomic

OP justifies design and explains it

shows how behavior fulfills purpose

```
concept Style
```

no dependences

```
purpose consistent formatting
```

```
structure
defined: Style -> one Format
style: Element -> one Style
ne Format = style.defined
```

separation of concerns

maximal polymorphism

```
actions
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s
```

```
principle
after define(s,f); assign(e1,s);
assign(e2,s); define(s,f')
observe e1.format = e2.format = f'
```

OP is an archetypal scenario

a theorem about behaviors

shows how purpose fulfilled

justifies packaging as concept

generalizes concept variants

Michael Polanyi
operational principle

meaning of a single concept

concept AuthUser

purpose identify users

structure

name, password: User -> **one** String

sessions: Client -> **set** User

actions

register(n: Name, p: String, **out** u: User)

login (n: Name, p: String, c: Client)

logout (c: Client)

auth (c: Client, **out** u: User)

principle

register(n,p,u); login(n,p,c); auth(c,u’)

=> u’ = u

meaning is set of **traces**:

```
{
< >,
<register(n0,p0,u0)>,
<register(n0,p0,u0), login(n0,p0,c0)>,
<register(n0,p0,u0), register(n1,p1,u1)>,
...
<register(n0,p0,u0), login(n0,p0,c0), auth(c0,u0)>,
...
}
```

actually, transition **histories**:

trace <register(n0,p0,u0)> is projection of history

```
<
({name={}, password={}, sessions={}},
register(n0,p0,u0),
{name={u0->n0}, password={u0->p0, sessions={}})
>
```

meaning of a single concept

concept Upvote

purpose track relative popularity

structure
votes: Item -> User

actions

```
upvote (i: Item, u: User)
   votes += i->u
count (i: Item, out k: int)
   k = #i.votes
```

principle
no upvote(i,u) **then** …
count(i, k); upvote(i,u); count(i, k’)
=> k’>k

traces:

```
{
< >, …
< count(i0, 0) >, …
< upvote(i0, u0) >, …
< upvote(i0, u0),  count(i0, 1) >, …
< count(i0, 0), upvote(i0, u0),  count(i0, 1) >, …
}
```

histories:

```
{
<>,
<({votes={}}, upvote(i0,u0), {votes={i0->u0}})>
…
}
```

formalizing transitions, histories & traces

transitions

a transition is a triple (pre-state, action-with-args, post-state)

let pre(x), action(x), post(x) be the pre-state, action and post-state of x

let inits(c) and trans(c) be the initial states and set of transitions of concept c

histories

a history is a sequence of transitions

history h is consistent if for all f, g != <>, h = f∧g implies post(last(f)) = pre(first(g))

concept histories

histories(c), the histories of a concept c include:

(1) the empty history <>

(2) any <x> where x in trans(c) and pre(x) in inits(c)

(3) any consistent history f∧<x> where f in histories(c) and x in trans(c)

concept traces

if h in histories(c), map(h, action) in traces(c)

theorems

prefix closure: if f∧g in histories(c) then f in histories(c) [and same for traces]

complete state: if h and f∧g in histories(c), h∧g in histories(c) if it's consistent

semantics of composition

Y
Hacker News
new | past | comments | ask | show | jobs | submit
login
add comment

post concept

auth concept

upvote concept

comment concept

▲ How to rewrite it in Rust (michaelfbryan.com)

173 points by FBT 5 hours ago | hide | past | web | favorite | 15 comments

▲ sorenbs 2 hours ago [-]

We did a similar thing with a Scala -> Rust rewrite for the http://prisma.io query engine.

By rewriting small components and integrating them into the existing project using Javas native interface, our small team of 5 developers were able to pull off this massive rewrite in just under a year. The resulting code base is rearchitected in a few very important ways, but mostly follows the same structure.

And because we kept and evolved our old Scala based test suite, we have a very high confidence in the rewrite.

When Async/.await finally landed, we could switch over very quickly, and it has been a joy to focus on benchmarks and performance over the last month. Spoiler: Rust is faster than Scala :-D

reply

▲ tombert 1 hour ago [-]

I promise that this is asked genuinely and isn't some sort of veiled "gotcha!" (it's tough to tell on the internet sometimes); what was the reason for a change from Scala to Rust?

I ask because Scala already has a good type system and the JVM typically has good performance nowadays, particularly with something like GraalVM, so I am actually really curious to why you felt a Rust rewrite was a good idea.

reply

making an app by composing concepts

concept Post
actions
new (a: Author, s: String, out p: Post)
edit (p: Post, s: String)
get (a: Author, out ps: set Post)

concept Comment
actions
new (a: Author, s: String, t: Target, **out** c: Comment)
get (t: Target, out cs: set Comment)

concept Upvote
actions
upvote (i: Item, u: User)
count (i: Item, out r: Int)

concept Owner
actions
register (o: Owner, i: Item)
owns (o: Owner, i: Item)

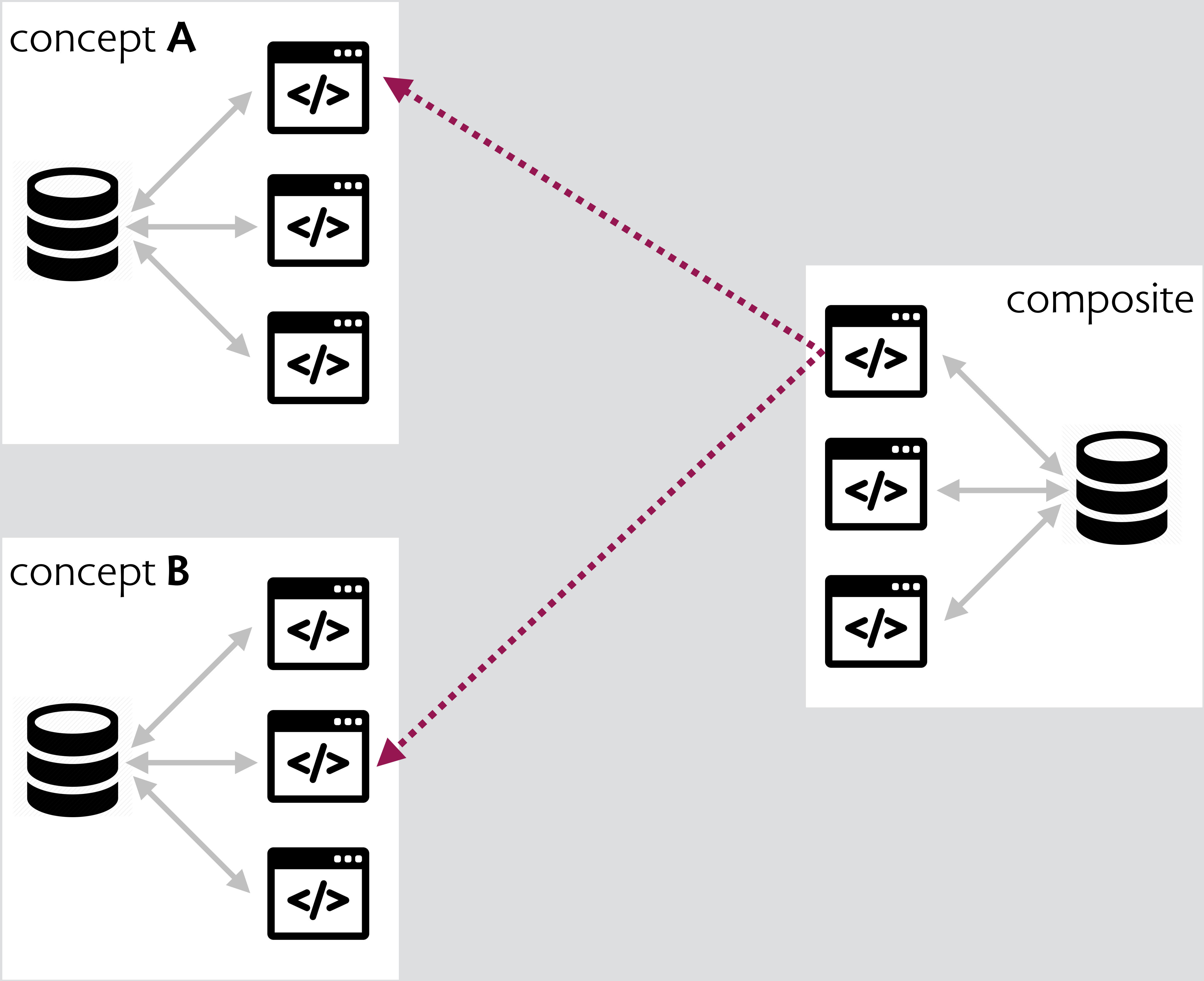
concept AuthUser
actions
register (n: Name, p: String, **out** u: User)
login (n: Name, p: String, c: Client)
logout (c: Client)
auth (c: Client, out u: User)

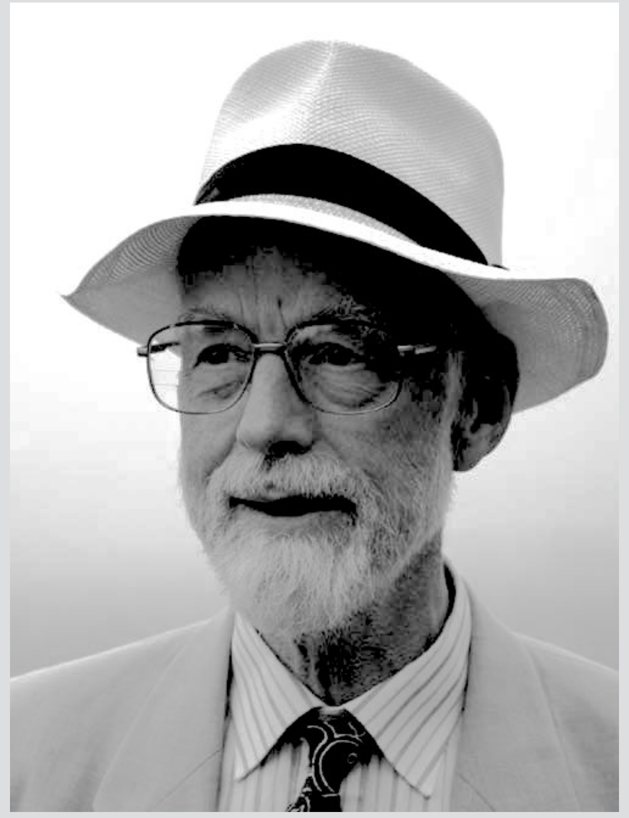
app HackerNews
includes Post, Comment, Upvote, AuthUser, Owner
synchronizes
newPost
  AuthUser.auth (c, u)
  Post.new(u, s, p)
  Owner.register(u, p)

editPost
  AuthUser.auth (c, u)
  Owner.owns(u, p)
  Post.edit(p, s)
newComment
  AuthUser.auth (c, u)
  Comment.new(u,s,p,x)
upvotePost
  AuthUser.auth (c, u)
  Upvote.upvote (p, u)
...

projecting transition

each transition in composite system
is interpreted as a transition in one of the concepts





Tony Hoare
CSP (1978)

code icon by Freepik from www.flaticon.com

check that projected transitions meet concept specifications

```
register
  AuthUser.register (n1, p1, u1)
...
login
  AuthUser.login (n1, p1, c1)
...
newPost
  AuthUser.auth (c1, u1)
  Post.new(u1, s1, p1)
  Owner.register(u1, p1)
upvotePost
  AuthUser.auth (c1, u1)
  Upvote.upvote (p1, u1)
```

concept AuthUser

AuthUser.register (n1, p1, u1)
AuthUser.login (n1, p1, c1)
AuthUser.auth (c1, u1)
AuthUser.auth (c1, u1)

✔

concept Post

Post.new(u1 ,s1, p1)

✔

concept Owner

Owner.register(u1, p1)

✔

concept Upvote

Upvote.upvote (p1, u1)

✔

formalizing composites histories & synchronizations

recall: transitions

trans(c) is the set of transitions of concept c [and trans(C) for concept set C]

composite histories

h is a composite history of an app made of concepts c in C if
every transition in h is in trans(C) and the subhistory h@c is in histories(c)

composite transitions and synchronizations

a composite transition X for concepts C is a non-empty sequence of trans(C)
a synchronization S is a set of composite transitions
an execution of S is a concatenation of some members of S

app histories

the histories of an app composed of concepts C with sync S are
the composite histories of C that are executions of S

not prefix-closed

note that the histories of an app are not generally prefix-closed
transitions of a composite transition must occur all-or-none

axes of synchronization

sync on actions alone

concept Post
actions
new (a: Author, s: String, **out** p: Post)
edit (a: Author, p: Post, s: String)
get (a: Author, **out** ps: **set** Post)

concept AuthUser
actions
register (n: Name, p: String, **out** u: User)
login (n: Name, p: String, c: Client)
logout (c: Client)
auth (c: Client, **out** u: User)

sync post (c: Client, s: String, **out** u: User, **out** p: Post)
AuthUser.auth (c, u)
Post.new (u, s, p)

sync edit (c: Client, p: Post, s: String, **out** u: User)
AuthUser.auth (c, u)
Post.edit (u, p, s)

sync on actions & pre-state

```
concept Trash
state
all, trashed: set Object
actions
create (out o: Object)
delete (o: Object)
restore (o: Object)
emptyTrash ()
```

```
concept Folder
state
contents: Folder -> (File + Folder)
static root, trash: disjoint Folder
initially contents = root -> trash
actions
newFolder (parent: Folder, out f: Folder)
newFile (parent: Folder, f: File)
move (o: File + Folder, to: Folder)
delete (f: File + Folder)
```

```
sync moveToTrash (o: File + Folder)
Folder.move (o, Folder.trash)
for x: o.*(Folder.contents) | Trash.delete (x)

sync empty ()
Trash.empty()
for x: Trash.trashed | Folder.delete(x)

sync restore (o: File + Folder, to: Folder)
{no (to + o.(Folder.parent)) & Trash.trashed}
Folder.move(o, to}
for x: o.*(Folder.contents) | Trash.restore (x)
```

sync on actions & post-state

```
concept Channel
state
rc, gc, bc: Image -> Channel
pixel: (Image + Channel) -> Coord -> Pixel
static red, green, blue: Pixel -> Pixel // color to greyscale
inv
all i: Image, c: Coord | i.pixel[c].red = i.rc.pixel[c] ...
actions
edit (x: Channel + Image, e: Coord -> Pixel)
```

```
concept Adjustment
state
pixel: Image -> Coord -> Pixel
adjFuns: Adjustment -> Param -> Pixel -> Pixel
actions
adjust (i: Image, a: Adjustment, p: Param)
```

```
sync applyAdjustment (i: Image, a: Adjustment, p: Param)
Adjustment.adjust (i, a, p)
Channel.edit (i, e)
{e = Channel.pixel[i]}
```

concept polymorphism

a fully polymorphic concept

```
concept Style

purpose consistent formatting

structure
defined: Style -> one Format
style: Element -> one Style
format: Element -> one Format = style.defined

actions
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s
```

this concept is polymorphic in the types Style and Format: they are essentially type **variables**

permuting transitions

concept Style

purpose consistent formatting

structure
defined: Style -> **one** Format
style: Element -> **one** Style
format: Element -> **one** Format = style.defined

actions
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s

typed transitions
the elements of each transition can be typed based on the decls

example
{defined={}, style={}, format={}}
define(s0: Style, f0: Format)
{defined={s0: Style->f0: Format}, style={}, format={}}

permuting a transition
given a permutation π on type T, π: $T \longrightarrow T$
permutation π (t) of transition t just lifts π over t

example
π: Style $\longrightarrow$ Style = {s0->s1, s1->s0}
π (t) =
  {defined={}, style={}, format={}}
  define(s1: Style, f0: Format)
  {defined={s1: Style->f0: Format}, style={}, format={}}

permutation invariance & polymorphism

concept Style

purpose consistent formatting

structure
defined: Style -> **one** Format
style: Element -> **one** Style
format: Element -> **one** Format = style.defined

actions

```
define (s: Style, f: Format)
  s.defined := f
assign (e: Element, s: Style)
  e.style := s
```

invariance & polymorphism

a concept C is invariant (or polymorphic) in type T iff
for any permutation π on type T, π: $T \longrightarrow T$
whenever t is a transition of C, π (t) is also

what this means

the concept just does database-like operations
similar to Tarski's notion of "logical operations"

example

Style concept is polymorphic in Style and Format

primitive types are not polymorphic

concept Upvote

purpose track relative popularity

structure
votes: Item -> User

actions

```
upvote (i: Item, u: User)
   votes += i->u
count (i: Item, out k: int)
   k = #i.votes
```

an example of a non-polymorphic type
Upvote is not polymorphic in the type int

example of non-invariant transition
π: int $\longrightarrow$ int = {0->1, 1->0}
{votes={}} count (i0:Item, 0:int) {votes={}} is a transition
{votes={}} count (i0:Item, 1:int) {votes={}} is not a transition

note
a concept may be polymorphic in a primitive type
but that indicates a specification error

special values break polymorphism

concept Format

purpose stylize text

structure

static Bold, Underline, Italic: disjoint Format

format: Text -> set Format

actions

```
apply (t: Text, f: Format)
  f in Bold + Underline + Italic
  t.format :=
    f in t.format => t.format - f, t.format + f
print (t: Text) ...
```

an example of special values

this (very simplified) Format concept defines special values

represented as variables of the state, set initially

an initialization subtlety

initial values aren't given in the spec

but they must be chosen in any implementation

so Format concept is not polymorphic in the type Format

incomplete specification

this spec does not say what print does

but implied that it italicizes text formatted as italic, etc

opaque types

call these non-polymorphic, non-primitive types "opaque"

polymorphic type ~ type variable

opaque type ~ abstract data type

implications of polymorphism

concept Post
actions
new (a: Author, s: String, out p: Post)
edit (p: Post, s: String)
get (a: Author, out ps: set Post)

concept AuthUser
actions
register (n: Name, p: String, **out** u: User)
login (n: Name, p: String, c: Client)
logout (c: Client)
auth (c: Client, out u: User)

sync
AuthUser.auth (c, u)
Post.new (u, s, p)

joining polymorphic types
polymorphic types can be joined in concept compositions
so AuthUser.User can be joined to Post.Author
this is how Deja Vu works

exposing implementation detail
AuthUser is polymorphic in String, so should be Password, say
(but if validated password, would no longer be polymorphic)

implications of opacity

concept Channel
state
rc, gc, bc: Image -> Channel
pixel: (Image + Channel) -> Coord -> Pixel
static red, green, blue: Pixel -> Pixel
actions
edit (x: Channel + Image, e: Coord -> Pixel)

concept Adjustment
state
pixel: Image -> Coord -> Pixel
adjFuns: Adjustment -> Param -> Pixel -> Pixel
actions
adjust (i: Image, a: Adjustment, p: Param)

sync
Adjustment.adjust (i, a, p)
Channel.edit (i, e)
{e = Channel.pixel[i]}

joining opaque types
if opaque types are joined, concepts must share interpretation
not truly independent of each other

example
Channel and Adjustment both have Pixel as opaque
must have common interpretation of pixel values

example: waze

concept CrowdsourcedConditionTracking

purpose track condition of a public resource

structure
reports: User -> Resource -> Condition -> Time
inferred: Resource -> Condition

actions
report (u: User, r: Resource, c: Condition, t: Time)
update () // compute inferred from reports

principle
with accurate reports and frequent updating,
inferred condition reflects reality

which types are opaque
in this concept?

concept ConditionPrediction

purpose predict future from past conditions

structure
history: Resource -> Time -> **one** Condition
predicted: Resource -> TimeSlot -> **one** Condition
slot: Time -> **one** TimeSlot

actions
report (r: Resource, t: Time, c: Condition)
update () // compute inferred from reports

principle
with accurate reports and frequent updating,
inferred condition reflects reality

which types are opaque
in this concept?

example: group

group concept

concept Group

purpose control access to shared assets

structure

```
members: Group -> User
assets: Group -> Asset
```

actions

```
join (u: User, g: Group)
   g.members += u
contribute (u: User, g: Group, a: Asset)
  u in g.members
  g.assets += a
access (u: User, a: Asset)
   a in (members.u).assets
```

principle

if you join a group and some contributes an asset,
you can access it

invitation concept

concept Invitation

purpose grant optional access to resource

structure

```
pending, accepted: set Invitation
from, to: Invitation -> one User
for: Invitation -> Resource
```

actions

```
invite (inviter, invitee: User, r: Resource, out i: Invitation)
  i not in pending + accepted
  pending += i
  i.from := inviter; i.to := invitee; i.resource :- r
accept (invitee: User, i: Invitation)
  i in pending and i.from = invitee
  accepted += i; pending -= i
access (u: User, r: Resource)
  some i: accepted | i.to = user and i.for = r
```

synchronizing group and invitation

Group
join (u: User, g: Group)
contribute (u: User, g: Group, a: Asset)
access (u: User, a: Asset)

Invitation
invite (inviter, invitee: User, r: Resource, out i: Invitation)
accept (invitee: User, i: Invitation)
access (u: User, r: Resource)

sync
join (u, g) || accept (u, i) where Invitation.for[i] = g

purpose as design criterion

OP as a criterion for being a concept

if you can formulate a compelling OP, you have a concept

what's compelling?
intricate protocol
non-trivial outcome

what's not?
entity with CRUD
can't stand alone

social media
upvote: when you upvote, post ranked higher
friend: when you become friend, can access updates
post: after submitting post, people can read it
user account: when login, authenticated as particular user
user profile: : just a data structure without an OP
edit post: : just an action
timeline: an action? (show posts chronologically by author?)

image editing
image-local: when you edit pixels with local adjustment, get new image
image-global: when you apply global adjustment, image changes
image-channel: when you edit channel, whole image changes
channel, pixel, etc (alone): just data structures without an OP
brush, gradient, etc: just an action

why does this matter?
guides granularity,
structure of design

some design criteria for reusability & simplicity

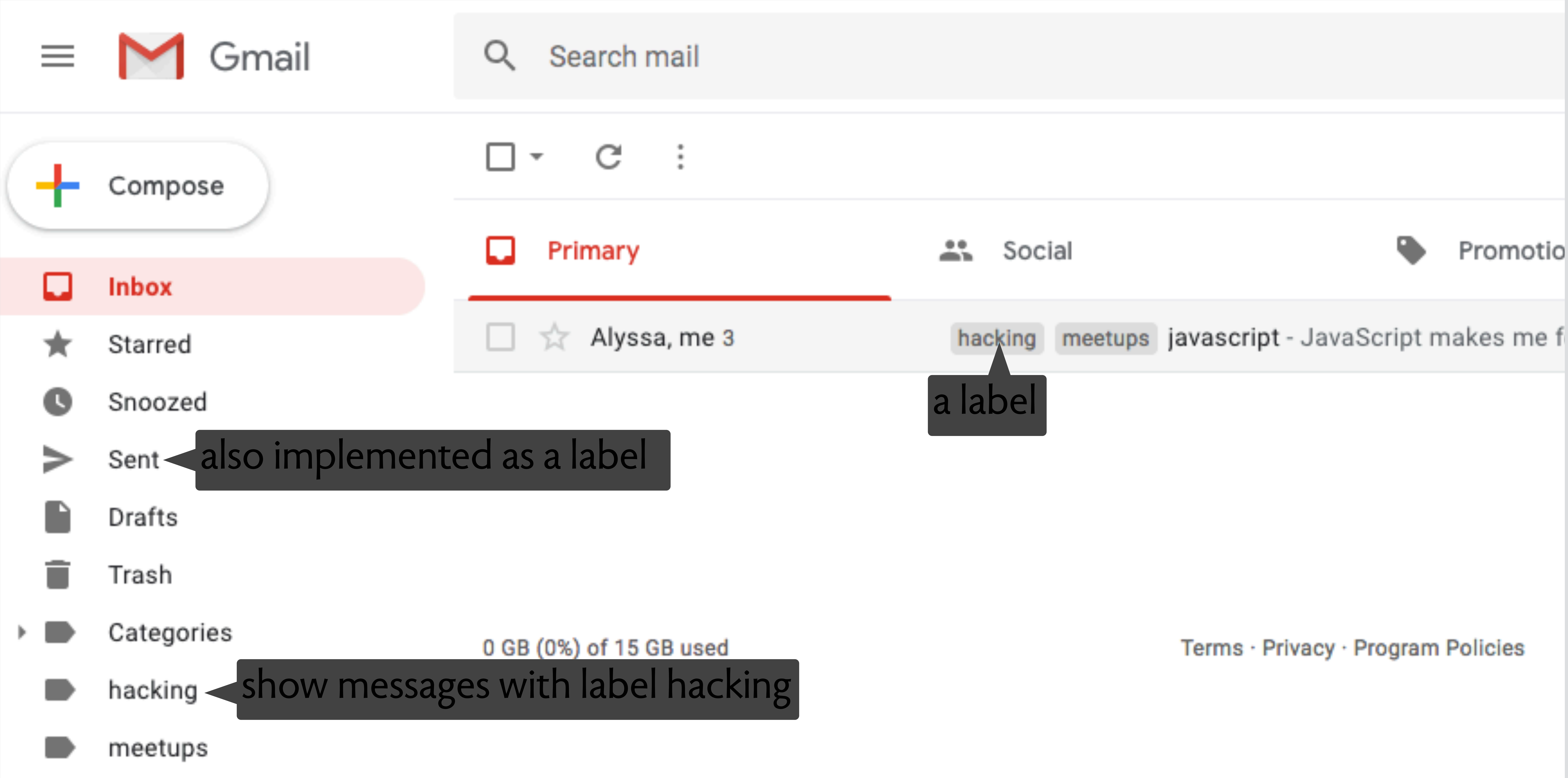
make concepts as polymorphic as possible
example: Group should not include user profiles (opaque)

break into smallest concepts you can
example: separate Invitation from Group

but not so small that OP is lost
example (good): Group
example (bad): Pixel
example (on the edge): UserProfile

gmail design issues

using labels to organize messages



Alyssa, me 3 hacking meetups javascript - JavaScript makes me f

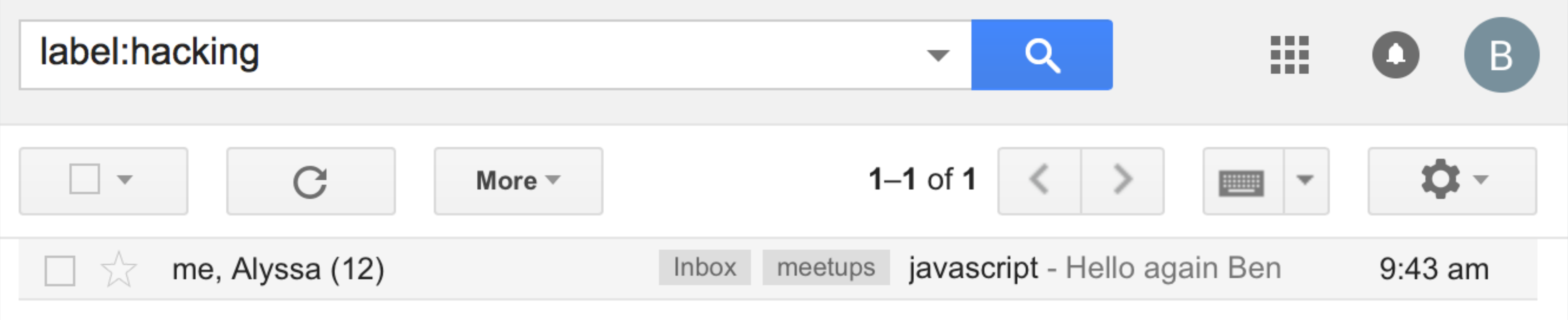
0 GB (0%) of 15 GB used

a label

also implemented as a label

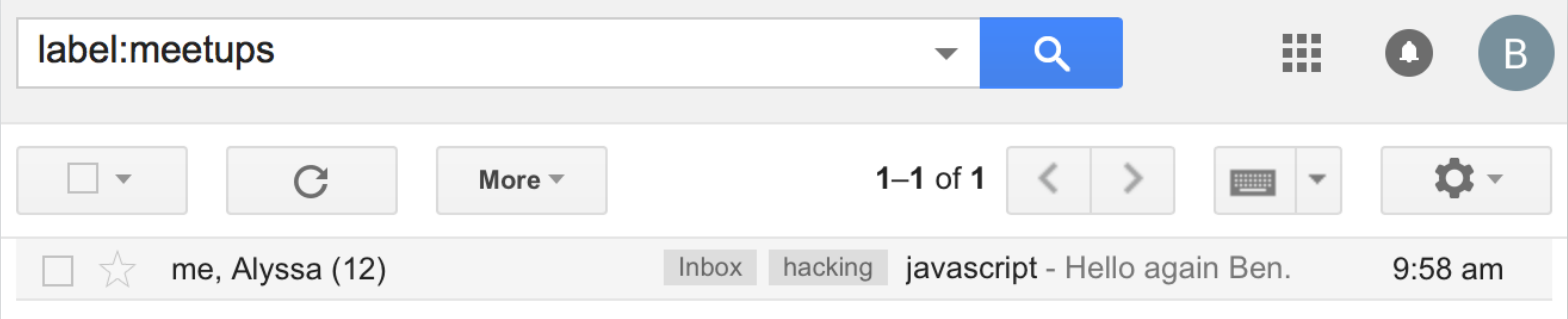
show messages with label hacking

a surprising behavior



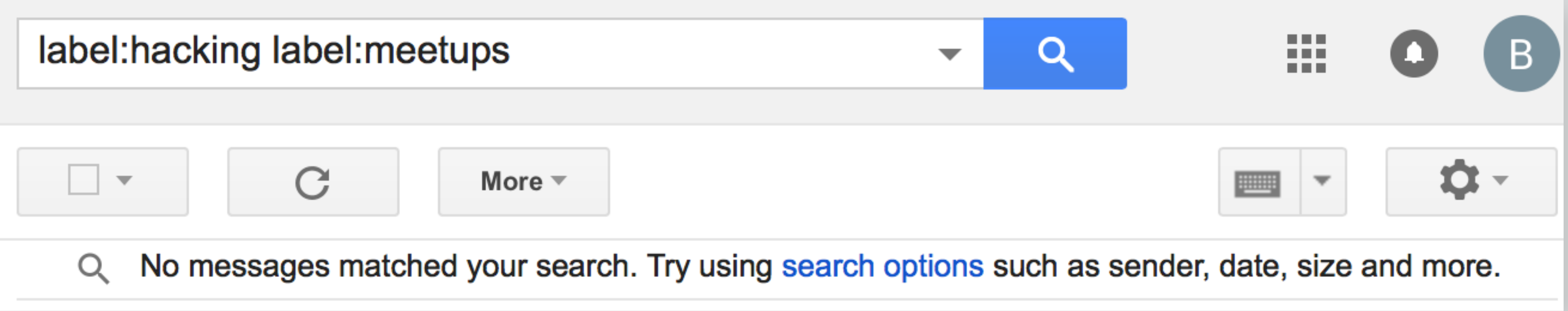
label:hacking

□ ☆ me, Alyssa (12) Inbox meetups javascript - Hello again Ben 9:43 am



label:meetups

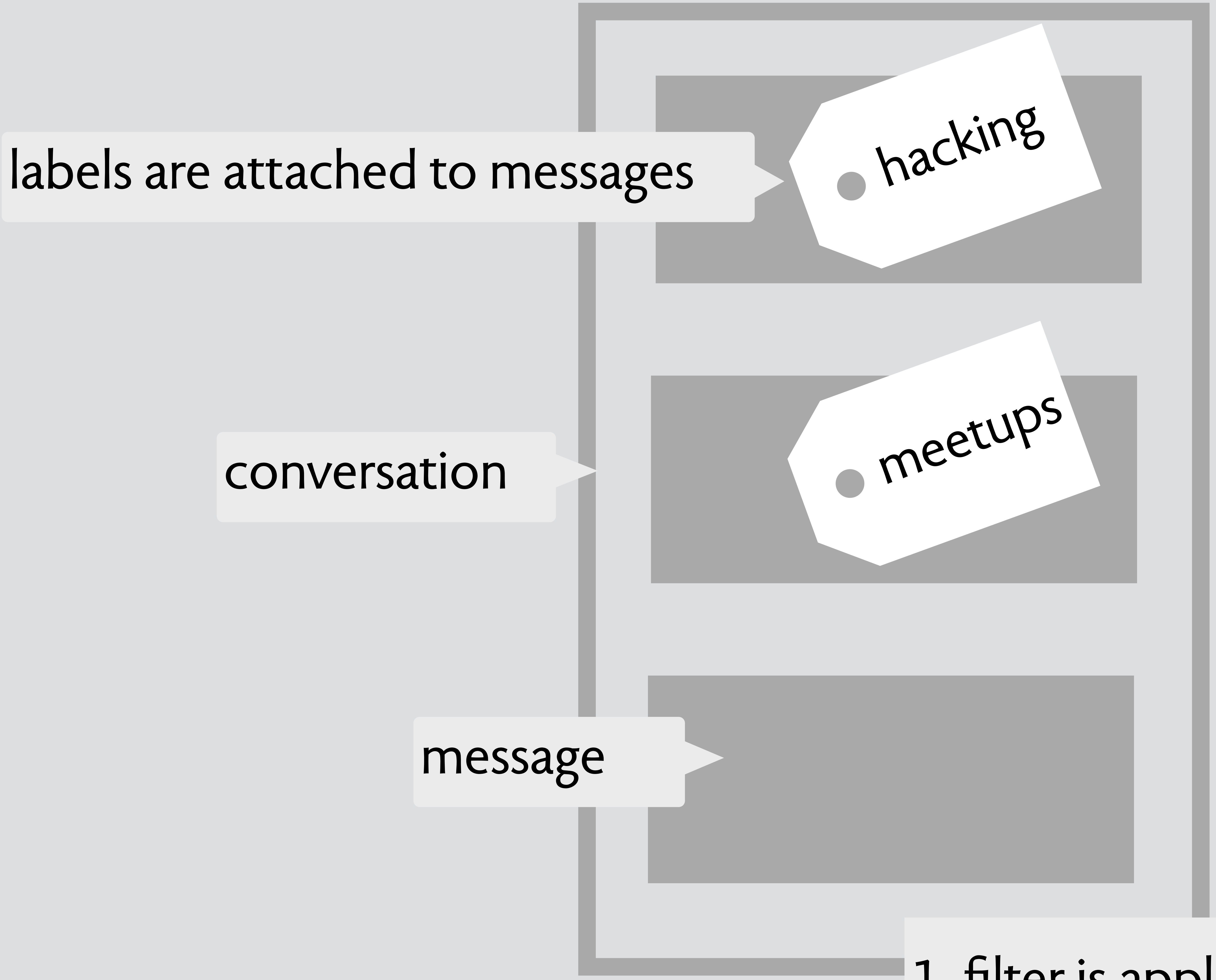
□ ☆ me, Alyssa (12) Inbox hacking javascript - Hello again Ben. 9:58 am



label:hacking label:meetups

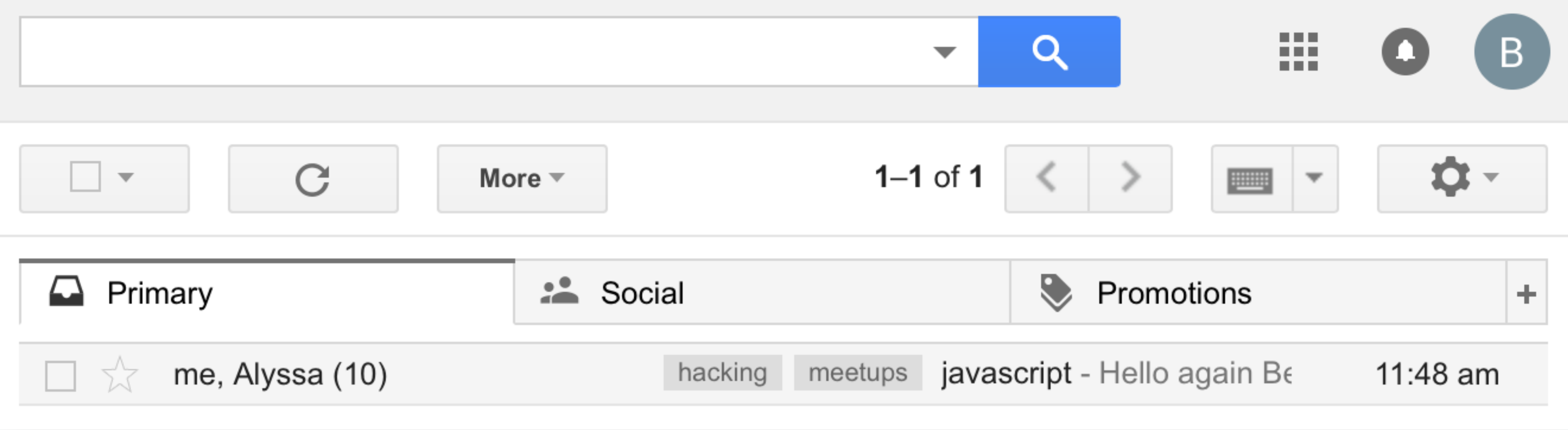
No messages matched your search. Try using search options such as sender, date, size and more.

what's going on?

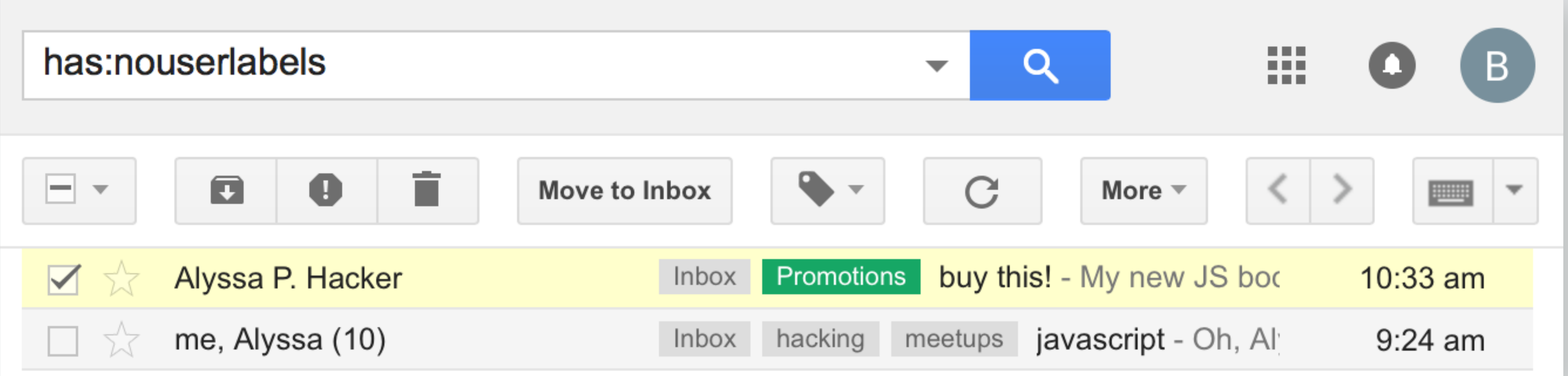


1. filter is applied to set of messages: some match
2. conversation appears if it includes a matched message

so this is not a surprise



☐ ☆ me, Alyssa (10) hacking meetups javascript - Hello again Be 11:48 am

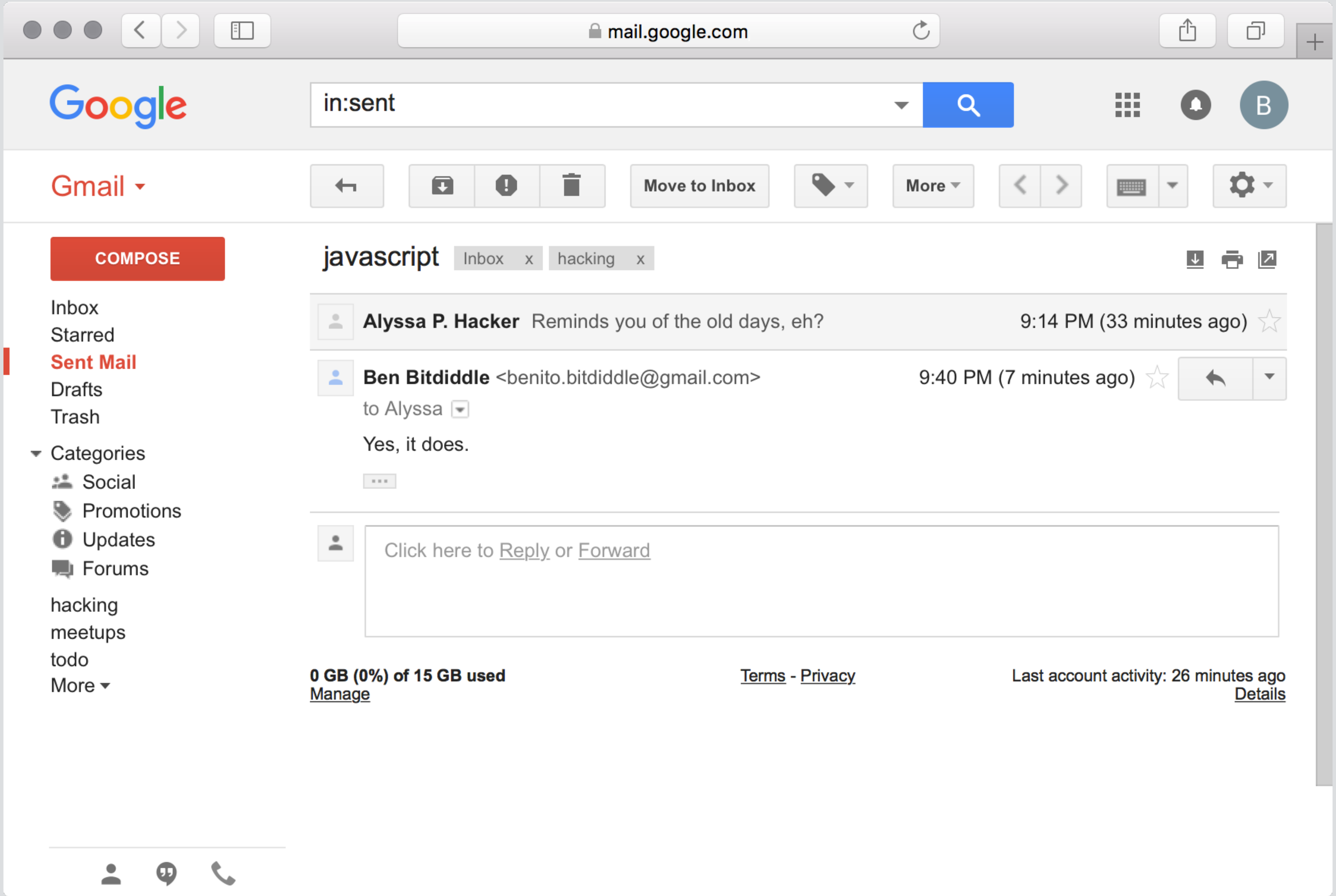


has:nouserlabels

☑ ☆ Alyssa P. Hacker Inbox Promotions buy this! - My new JS boo 10:33 am

☐ ☆ me, Alyssa (10) Inbox hacking meetups javascript - Oh, Al 9:24 am

and this makes sense too (but order is special)



javascript Inbox x hacking x

Alyssa P. Hacker Reminds you of the old days, eh? 9:14 PM (33 minutes ago)

Ben Bitdiddle <benito.bitdiddle@gmail.com> 9:40 PM (7 minutes ago)
to Alyssa

Yes, it does.

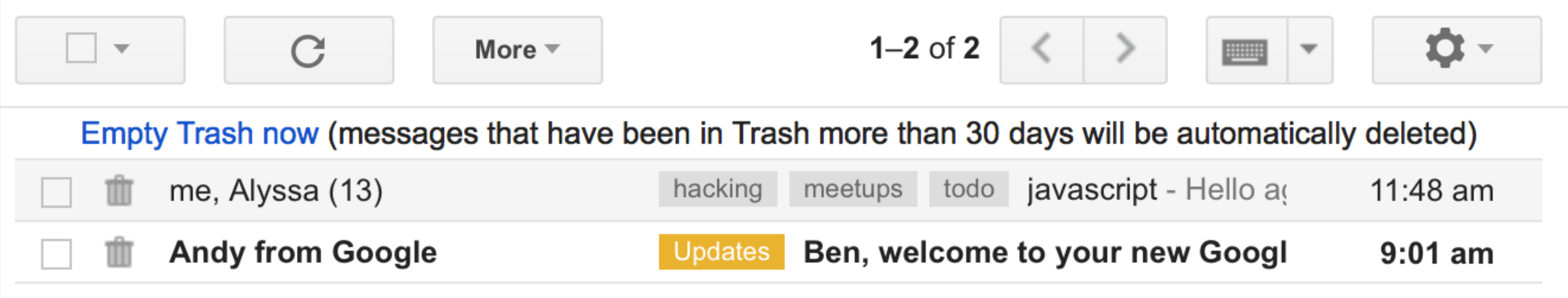
Click here to Reply or Forward

0 GB (0%) of 15 GB used
Manage

Terms - Privacy

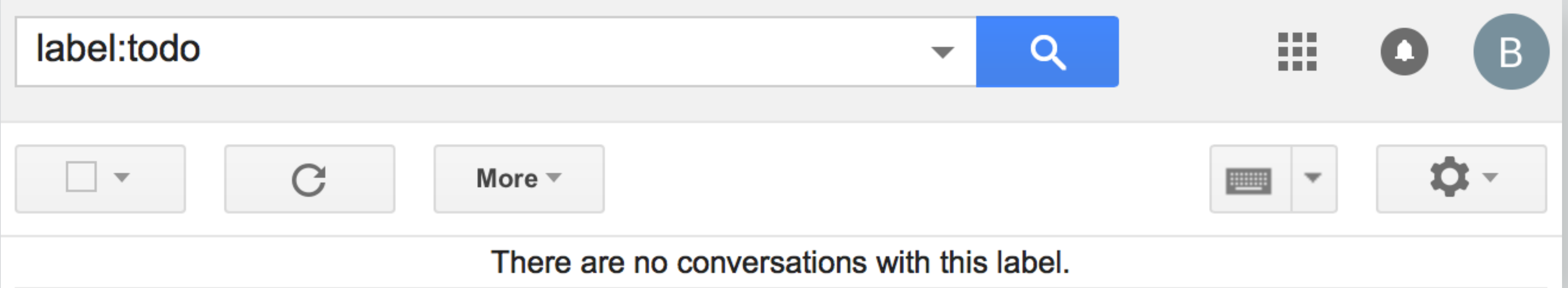
Last account activity: 26 minutes ago
Details

and this almost makes sense



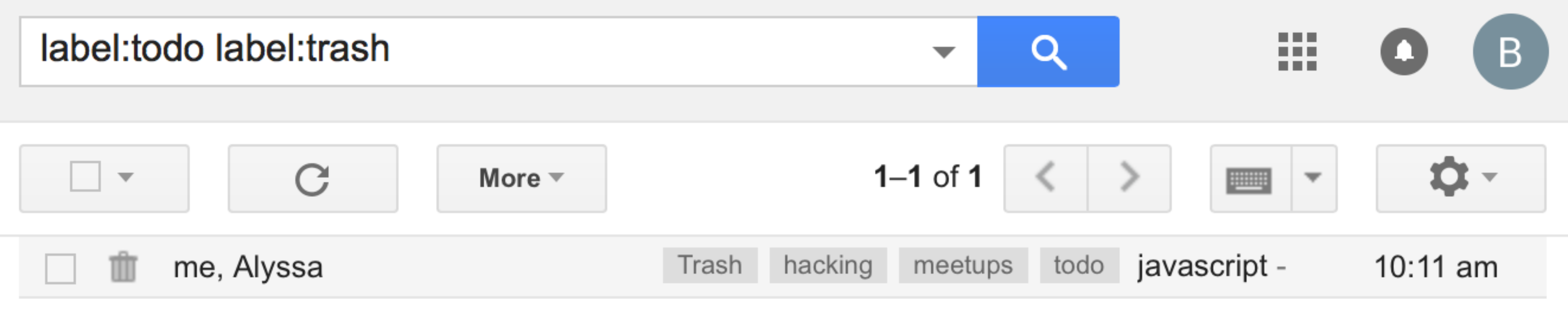
Empty Trash now (messages that have been in Trash more than 30 days will be automatically deleted)

me, Alyssa (13)	hacking meetups todo	javascript - Hello ag	11:48 am
Andy from Google	Updates	**Ben, welcome to your new Googl**	**9:01 am**



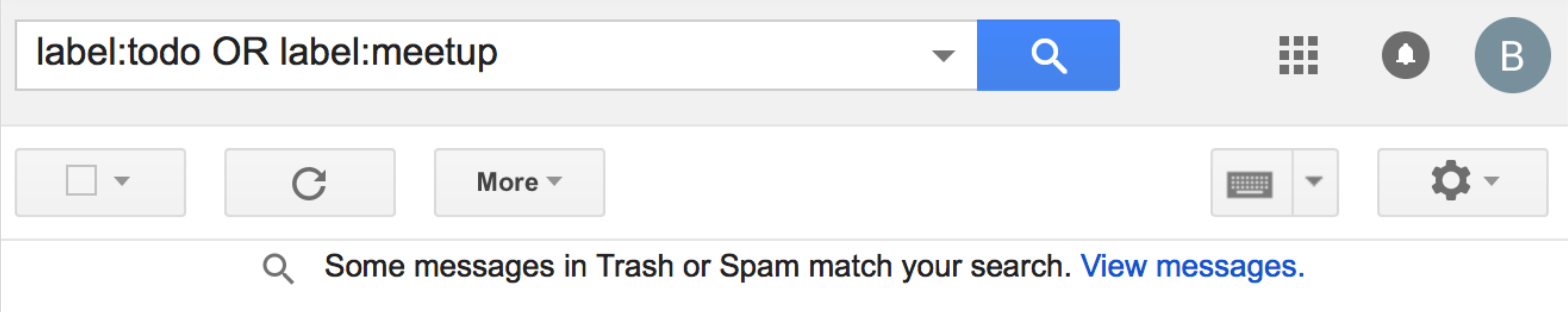
label:todo

There are no conversations with this label.



label:todo label:trash

me, Alyssa	Trash hacking meetups todo	javascript -	10:11 am



label:todo OR label:meetup

Some messages in Trash or Spam match your search. View messages.

the label concept

concept Label

purpose organize items for easy retrieval

structure

label: Item -> one String

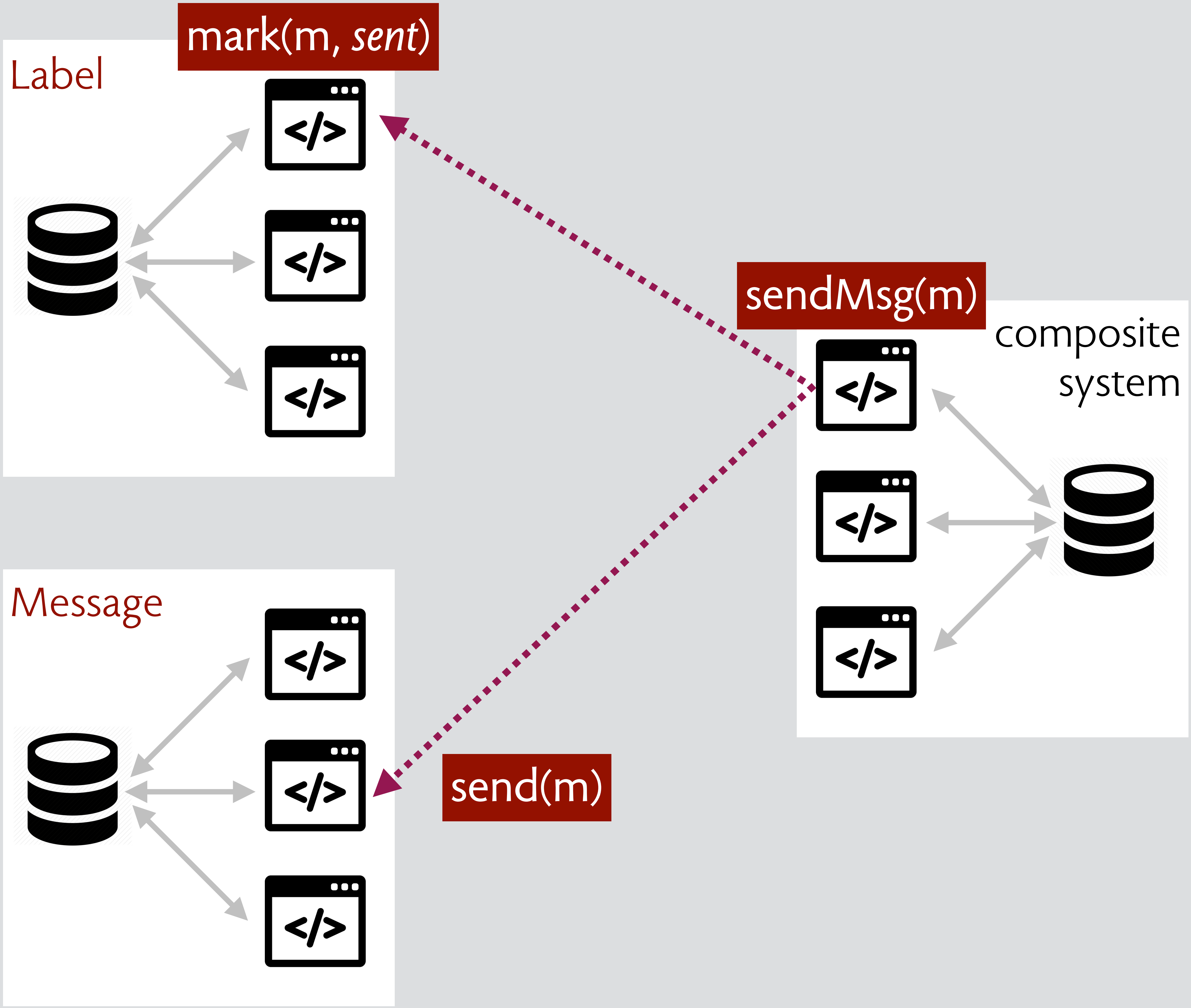
actions

```
mark (i: Item, p: Label)
  i.label += p
unmark (i: Item, p: Label)
  i.label -= p
find (ps: set Label): set Item
  result = {i | ps in i.labels}
```

story

if mark(i,p); find(p):is then i in is

if no mark(i,p); find(p):is then i !in is



in:sent
1 of 1

javascript Inbox × hacking × meetups ×

Alyssa P. Hacker <alyssa.pure.hacker@gmail.com> Tue, May 8, 9:14 PM
to me

Reminds you of the old days, eh?

Ben Bitdiddle <benito.bitdiddle@gmail.com> Tue, May 8, 9:40 PM
to Alyssa

Yes, it does.

Alyssa P. Hacker Mon, Jul 30, 1:24 PM
JavaScript makes me feel nostalgic for Scheme.

Ben Bitdiddle <benito.bitdiddle@gmail.com> 1:15 PM (1 minute ago)
to Alyssa

Is JavaScript just Scheme with prototypes and some hacky coercions?

when message m is sent
Label.mark(m, 'sent')
occurs implicitly

when Sent link is clicked
Label.find('sent'):ms
occurs

*but **ms** includes*
messages never marked

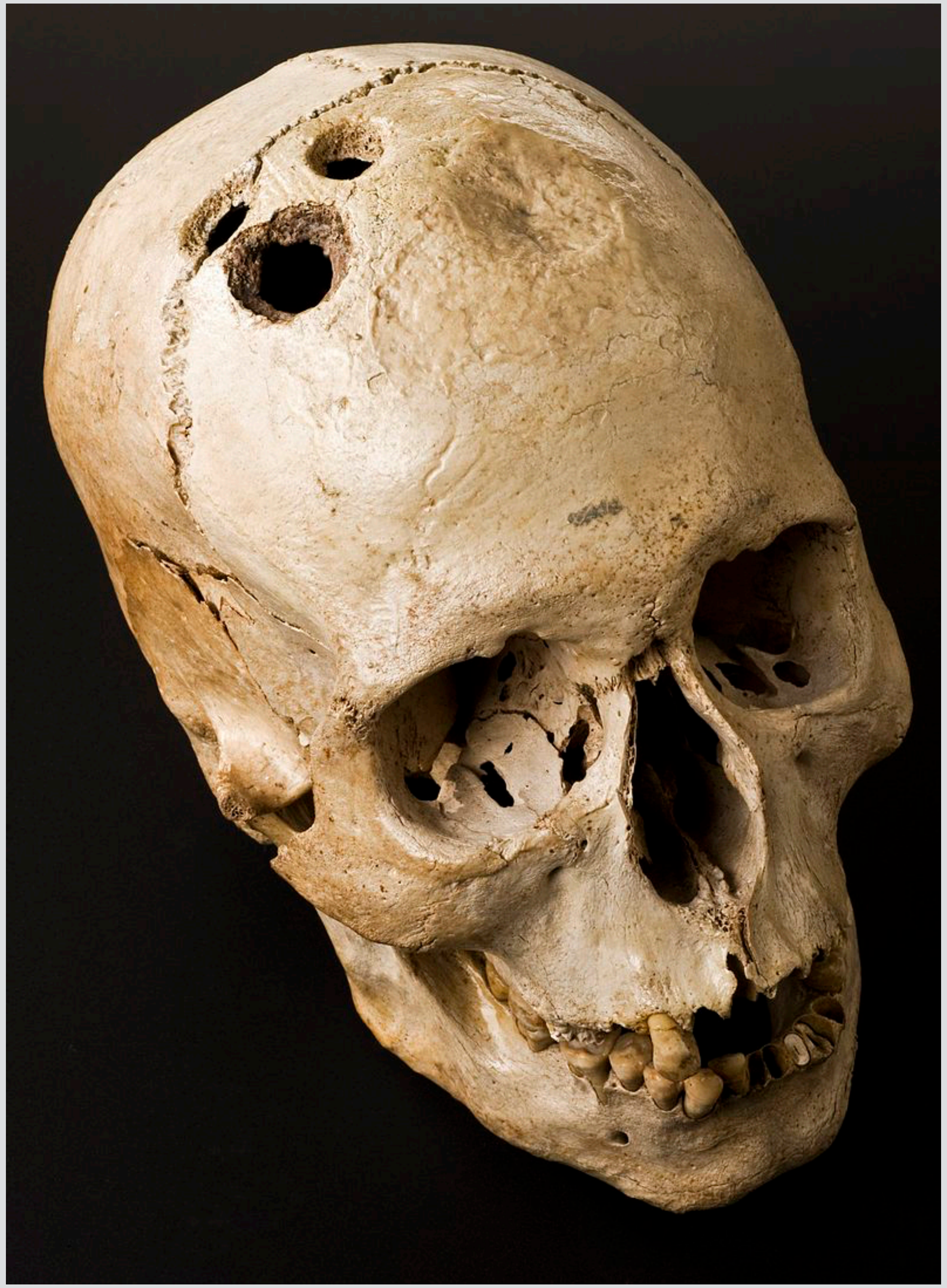
why pick on gmail?

not a strawman!
about 1.5B users
20% of global market
27% of all email opens

do these nitpicks matter?

“The details are not the details; they make the product”—Charles and Ray Eames

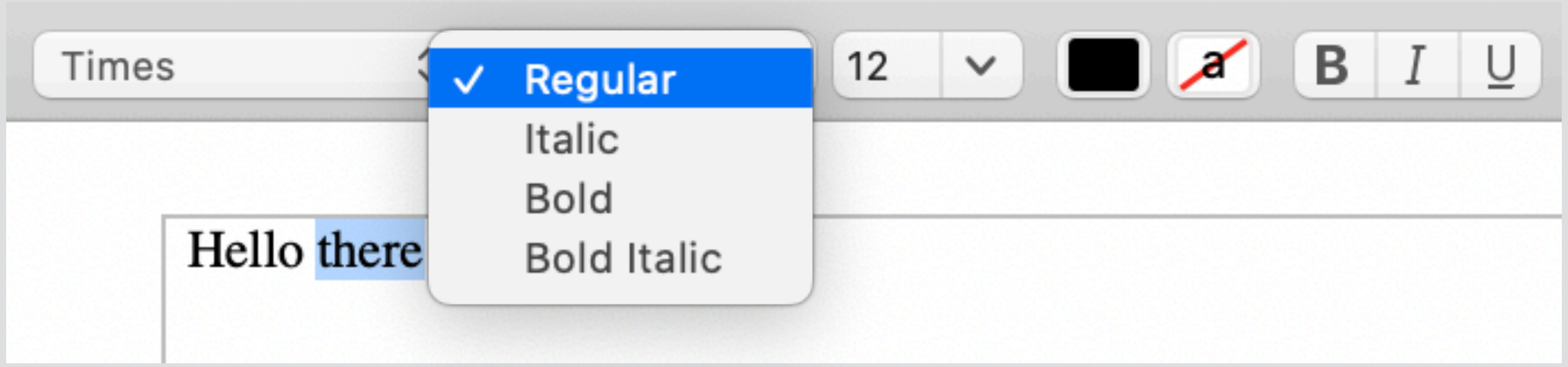
trepanning: small symptoms of major surgery



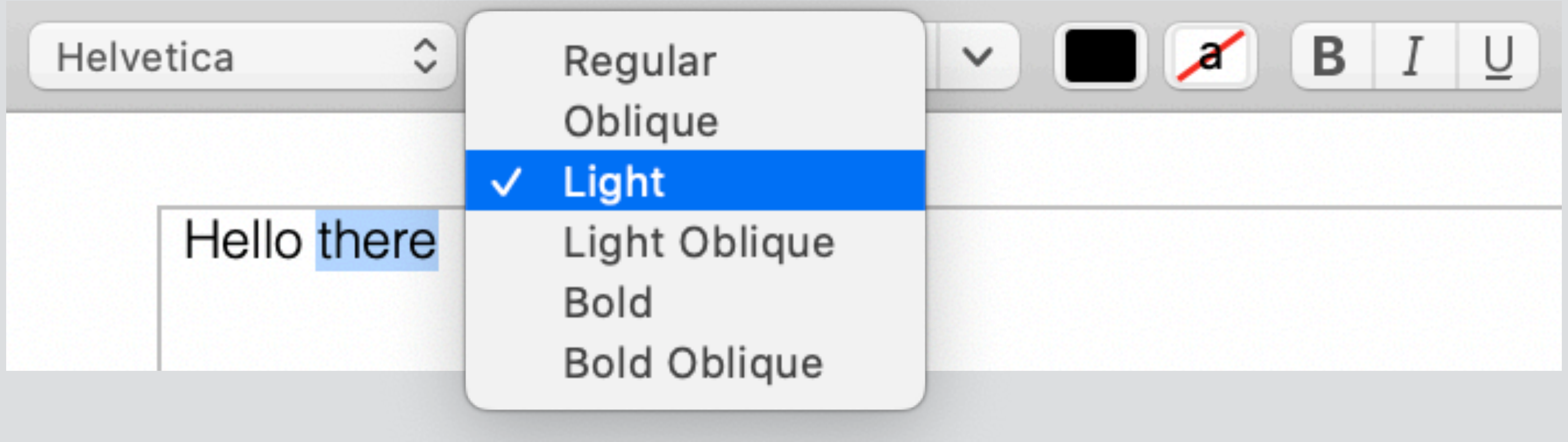
Bronze Age skull with evidence of trepanning

The Extraction of the Stone of Madness, Hieronymus Bosch font integrity example

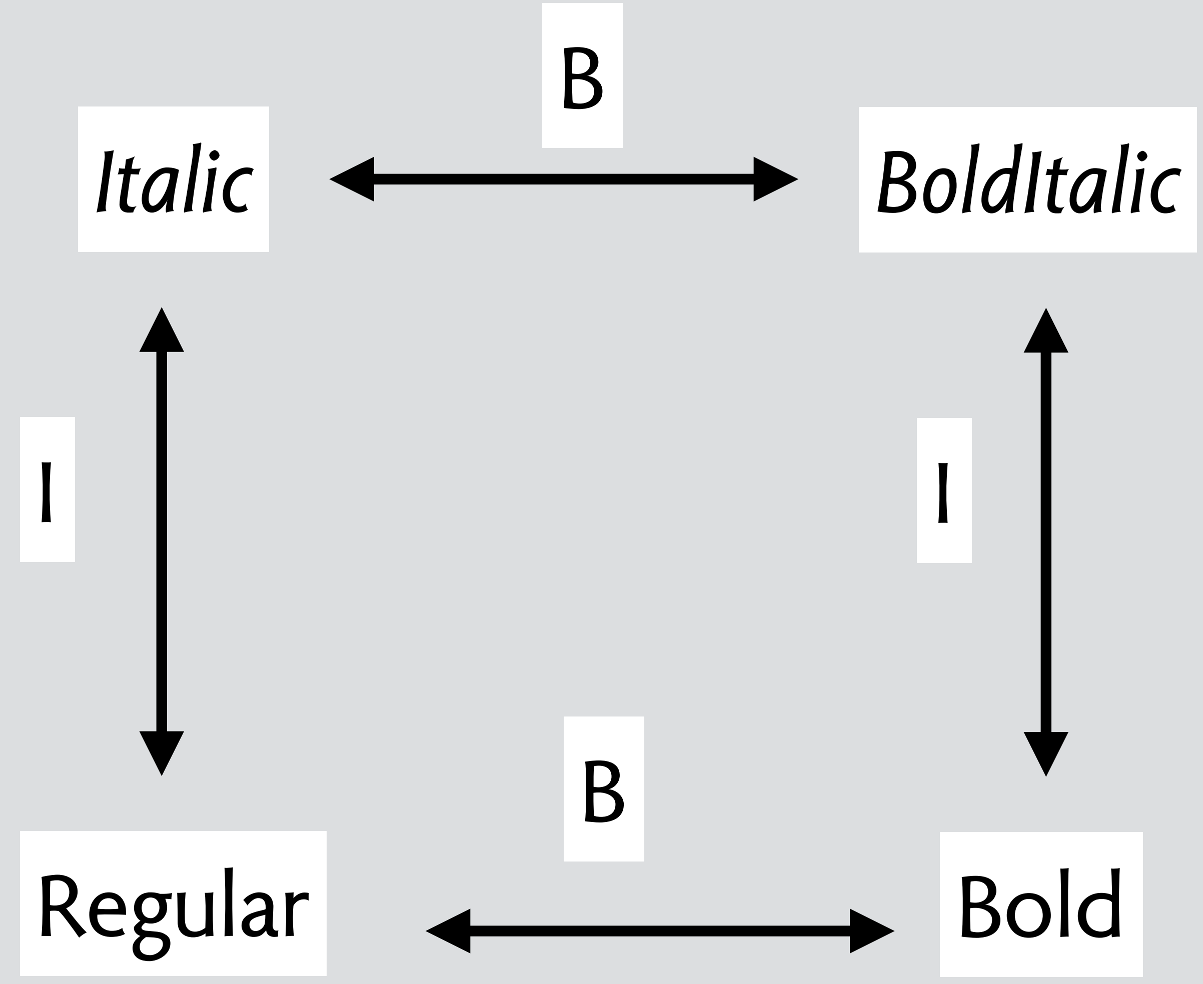
pro fonts break integrity of format concept

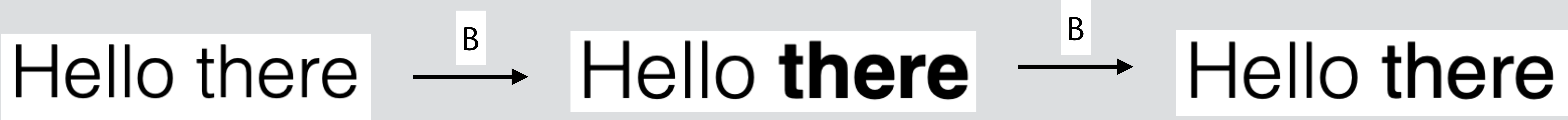


Hello there



Hello there





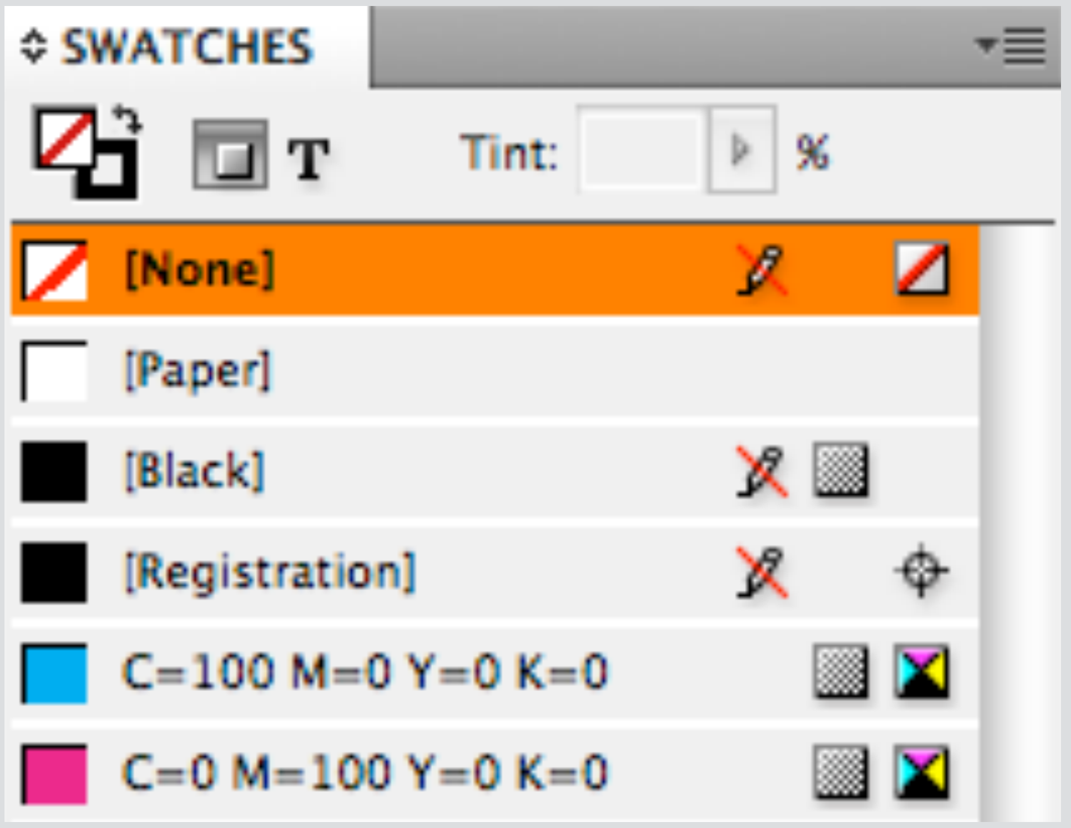
Hello there

Hello **there**

Hello **there**

synergy examples

what is design?



[None]

[Paper]

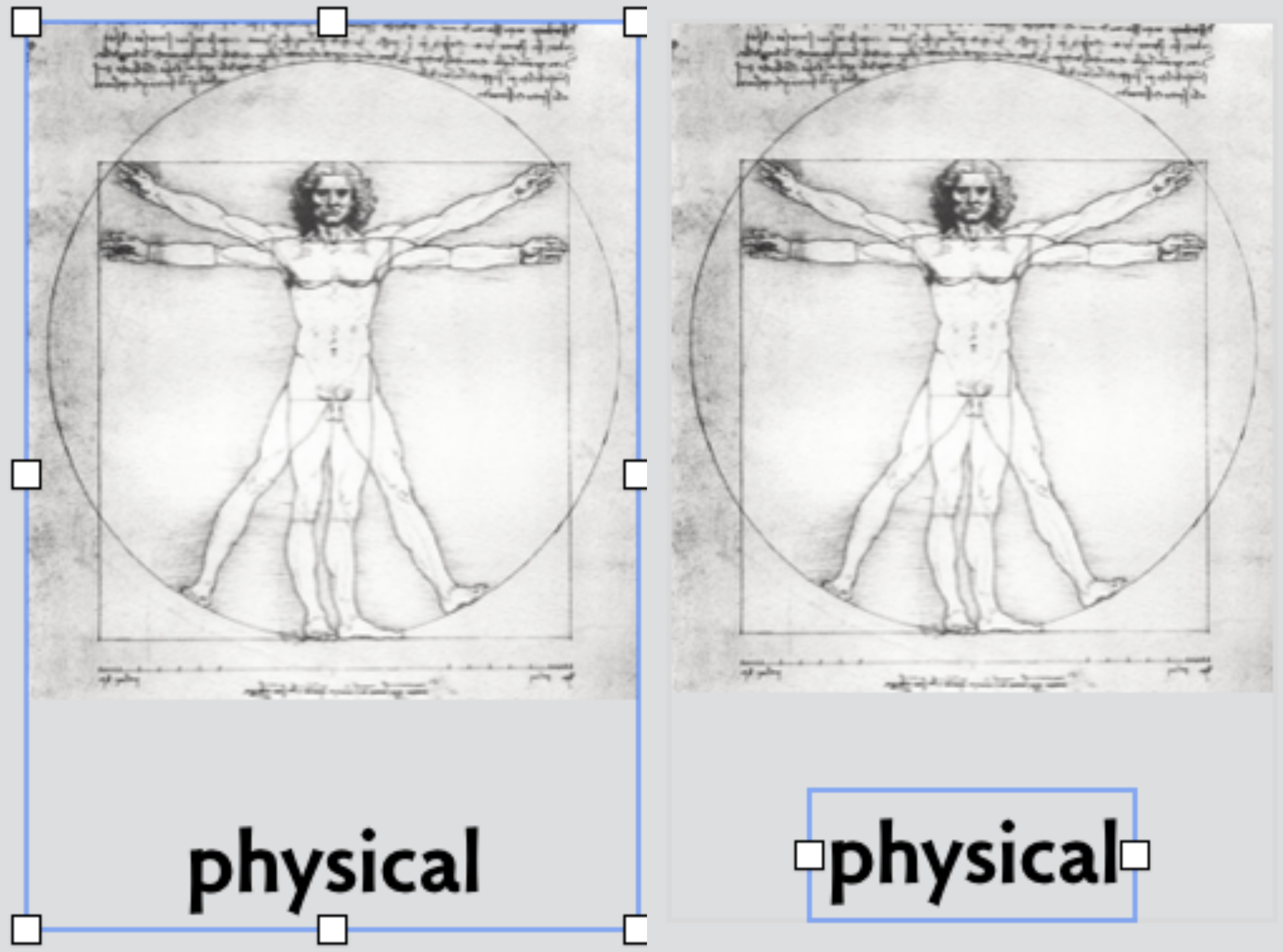
[Black]

[Registration]

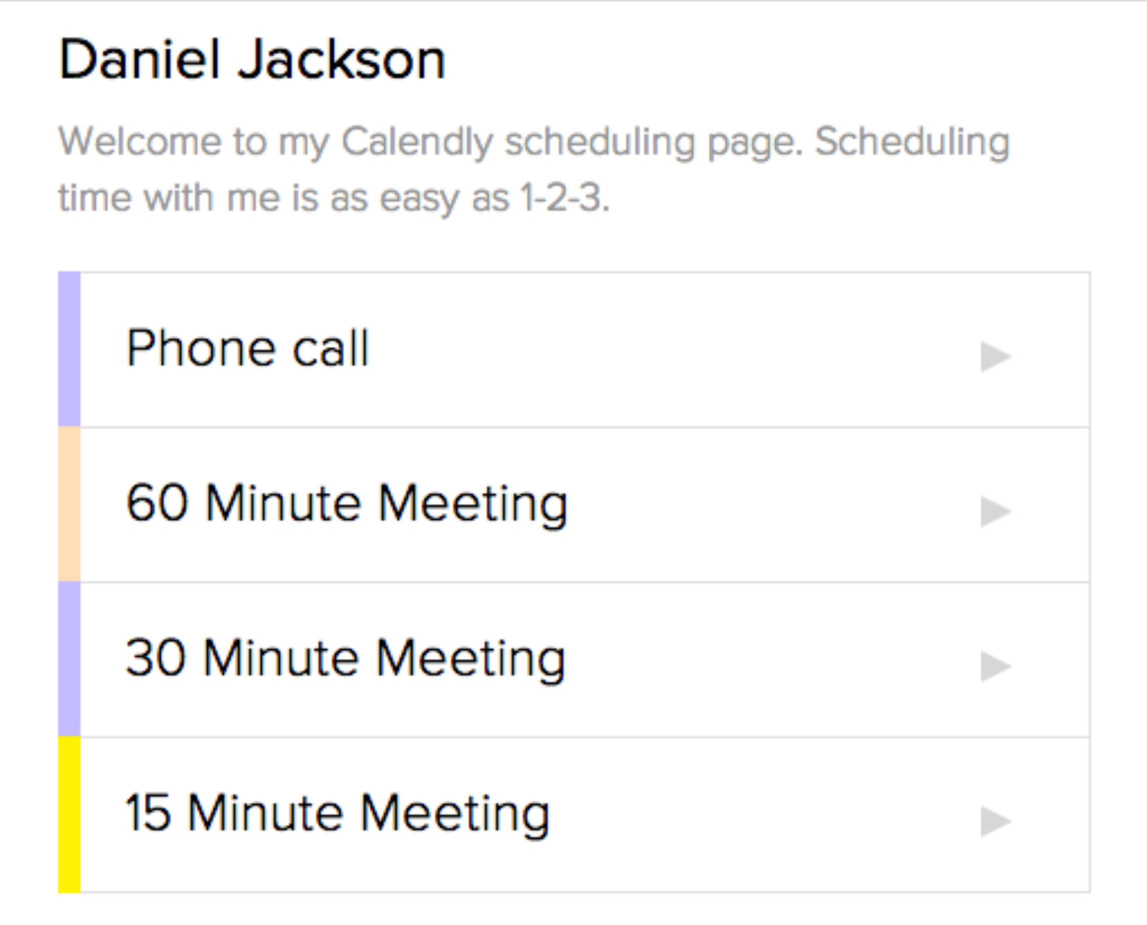
C=100 M=0 Y=0 K=0

C=0 M=100 Y=0 K=0

reusing concepts
using Style for color swatches



refining concepts
click to select Group elements

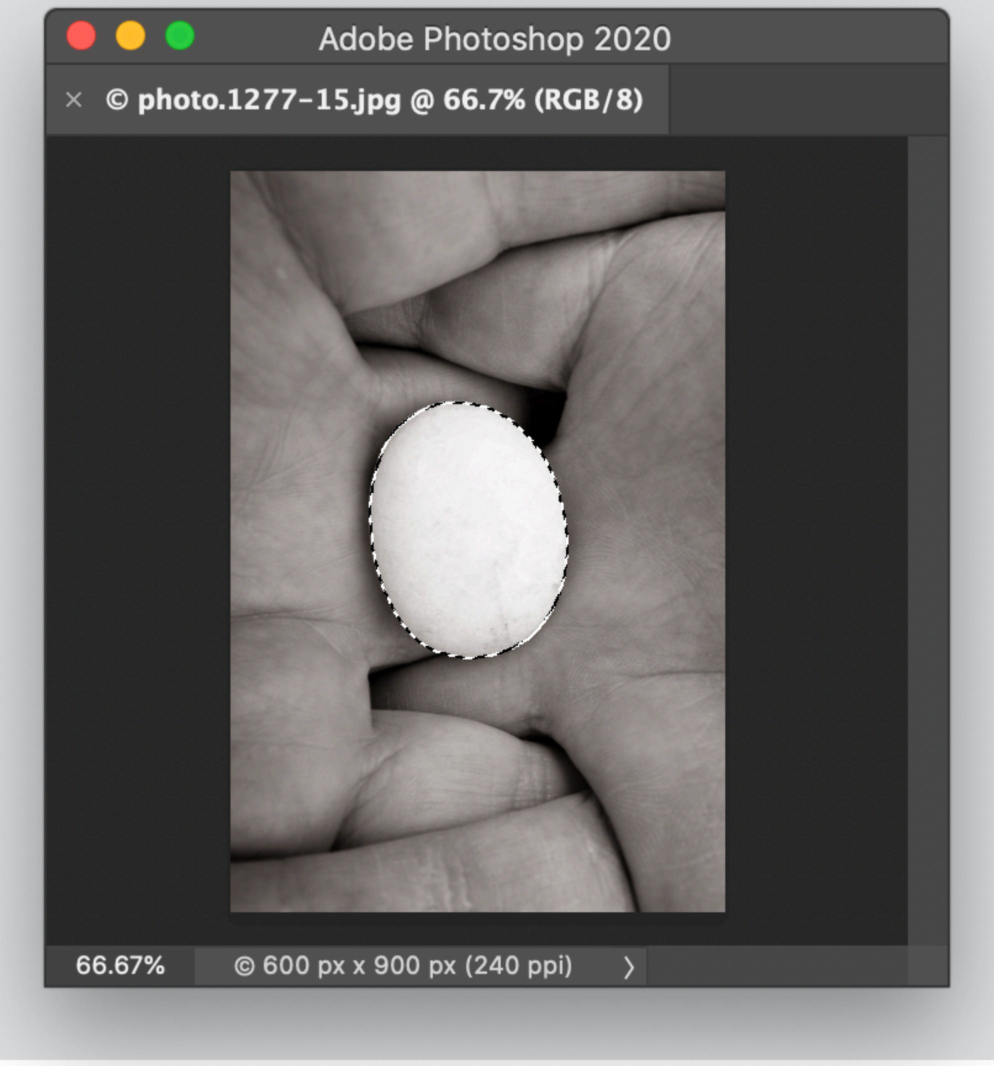


Daniel Jackson

Welcome to my Calendly scheduling page. Scheduling time with me is as easy as 1-2-3.

- Phone call ▶
- 60 Minute Meeting ▶
- 30 Minute Meeting ▶
- 15 Minute Meeting ▶

inventing concepts
Event Type in Calendly



synergy: merging concepts
channels in Photoshop

the trash concept & its history

```
concept Trash
purpose undo deletion
structure
  all, inTrash: set Object
actions
  delete (o: Object)
  empty ()
  restore (o: Object)
  new (o: Object)
  exists (o: Object, out b: bool)
story
  delete(o); restore(o); exists(o, true)
  delete(o); empty(); exists(o, false)
```

Apple Lisa (1982): “Wastebasket”
Apple Macintosh (1984): “Trash”
Microsoft MS-DOS 6 (1993): “DeleteSentry”
Apple vs. Microsoft (1994): Apple lost, but ©Trash
Windows 95 (1995): “Recycle Bin”

holds files not folders, so can’t recover structure

merging two concepts

concept Trash

purpose undo deletion

structure
all, inTrash: **set** Object

actions
delete (o: Object)
empty ()
restore (o: Object)
new (o: Object)
exists (o: Object, **out** b: bool)

story
delete(o); restore(o); exists(o, true)
delete(o); empty(); exists(o, false)

concept Folder

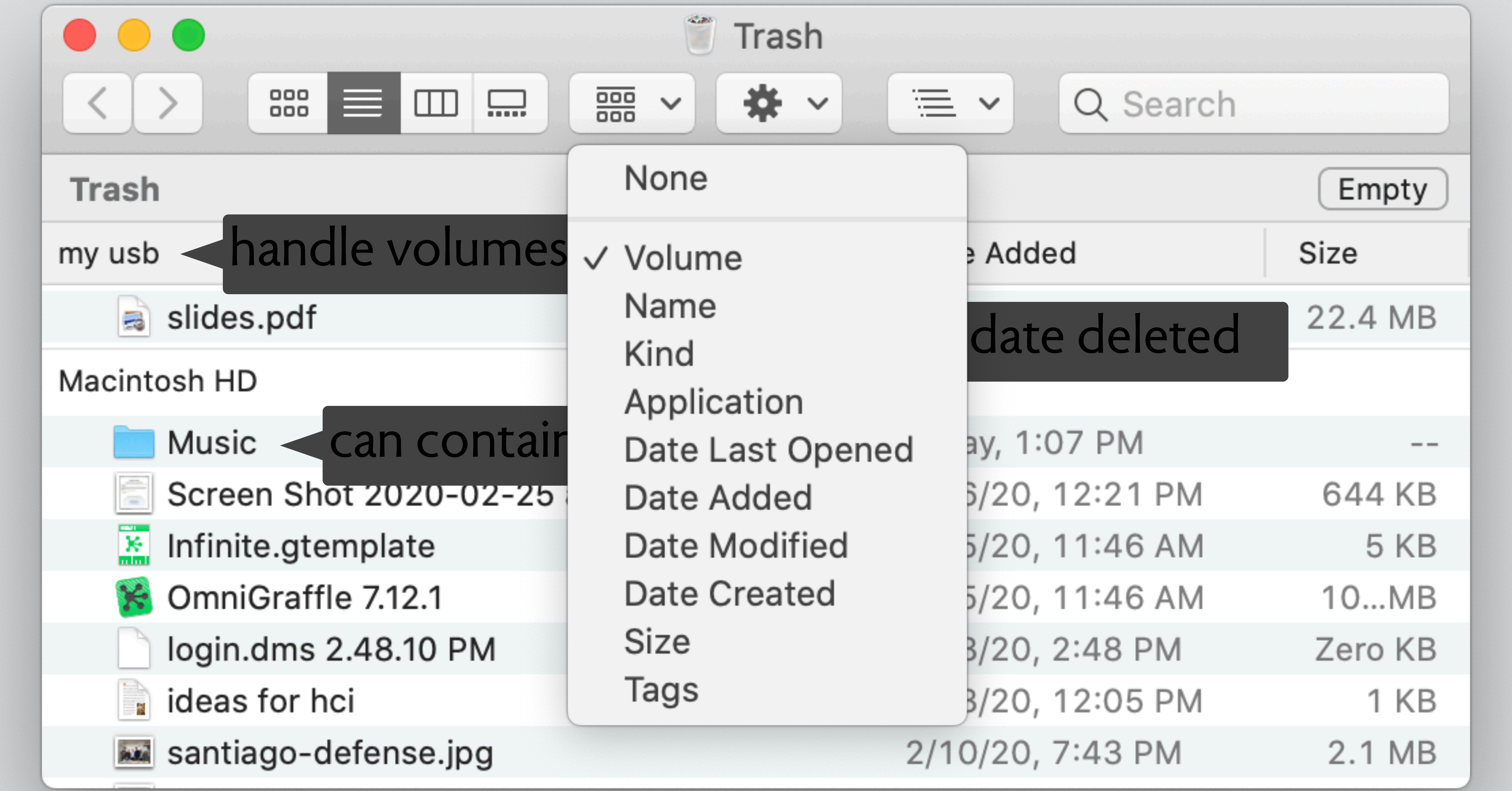
purpose local organization

structure
root: Folder
contents: Folder -> **set** (Folder + Object)

actions
move (o: Object + Folder, to: Folder)
new (p: Folder, **out** f: Folder)
list (f: Folder, **out** os: **set** Object)
delete (f: Folder)
root (**out** f: Folder)

story
list(f, os); move(o, to); list(f, os’)
=> **if** o **not in** os **and** to != f **then** os = os’

trash x folder



	e Added	Size
my usb		
slides.pdf		22.4 MB
Macintosh HD		
Music	ay, 1:07 PM	--
Screen Shot 2020-02-25	6/20, 12:21 PM	644 KB
Infinite.gtemplate	5/20, 11:46 AM	5 KB
OmniGraffle 7.12.1	5/20, 11:46 AM	10...MB
login.dms 2.48.10 PM	3/20, 2:48 PM	Zero KB
ideas for hci	3/20, 12:05 PM	1 KB
santiago-defense.jpg	2/10/20, 7:43 PM	2.1 MB

handle volumes

date deleted

can contain

trash x folder

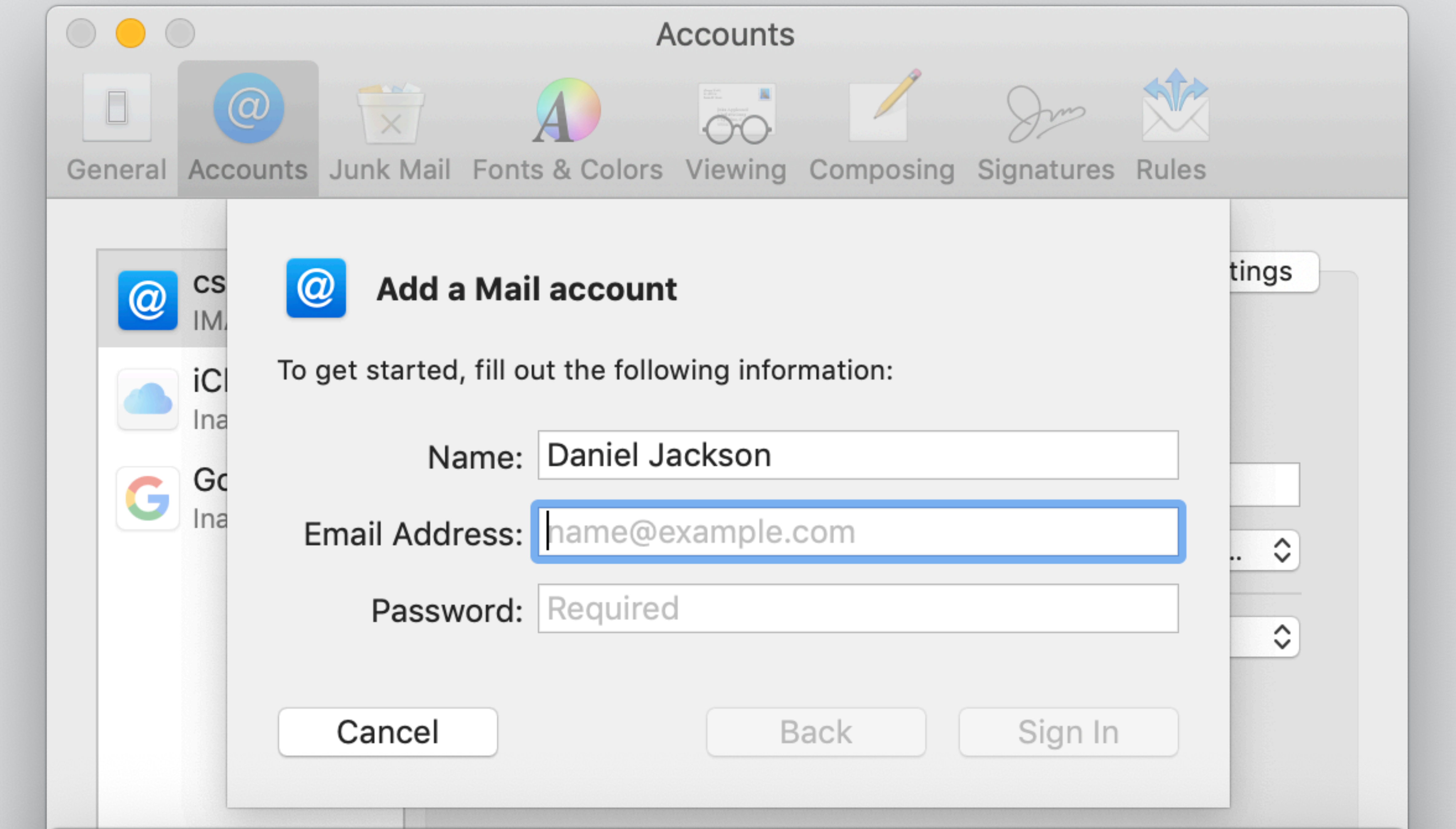
purpose: undo deletion

purpose: local organization

synergies
trash is not a special thing
all folder tools apply
can put folder in trash
move to trash = delete
move from trash = restore
date added = date deleted

anomalies
trash contains objects from >1 volume
in trash folder, can group by volume
delete immediately allows partial emptying
trash folder has no path (path concept)
can't move trash folder or delete it

email x server account



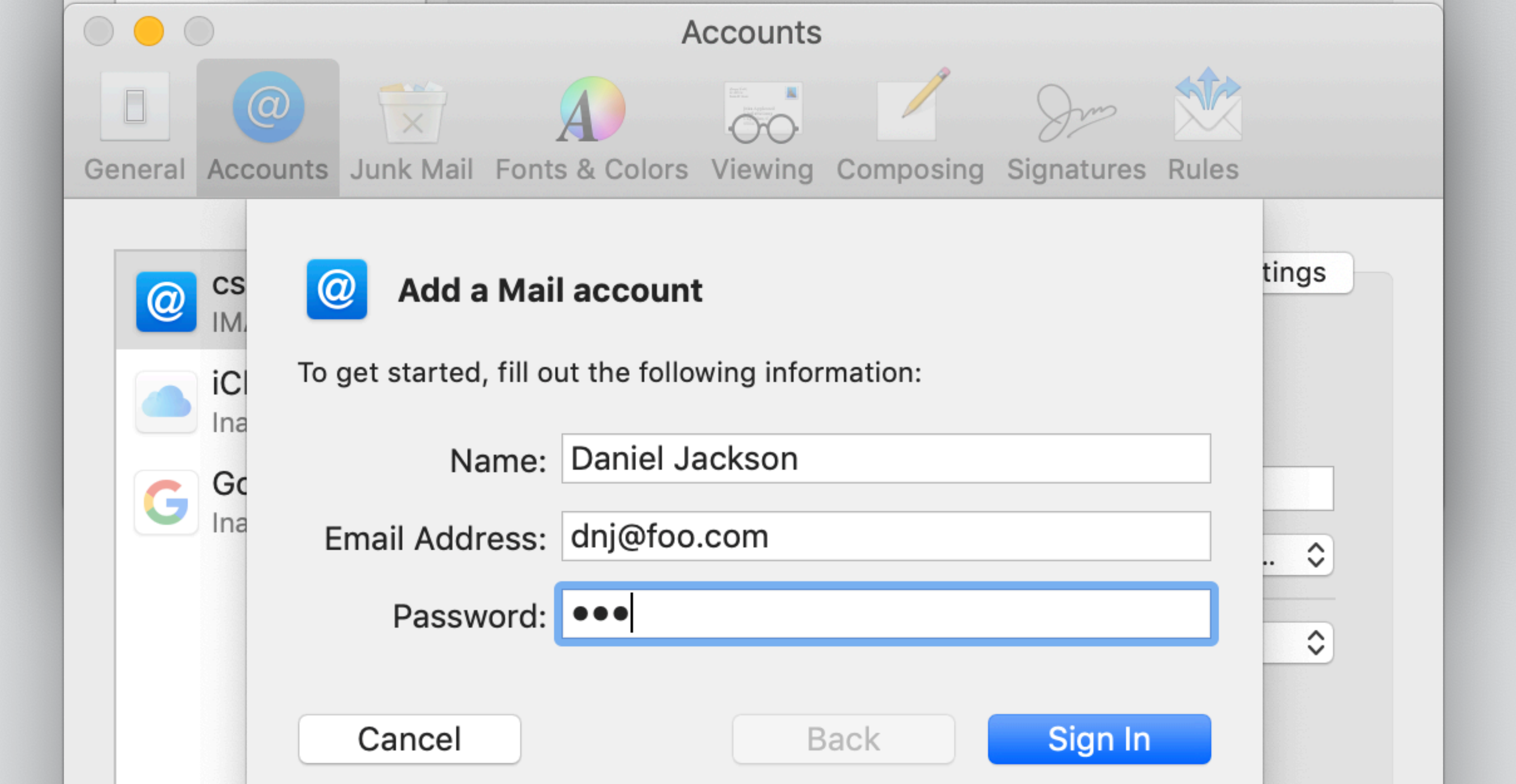
Add a Mail account

To get started, fill out the following information:

Name: Daniel Jackson

Email Address: name@example.com

Password: Required



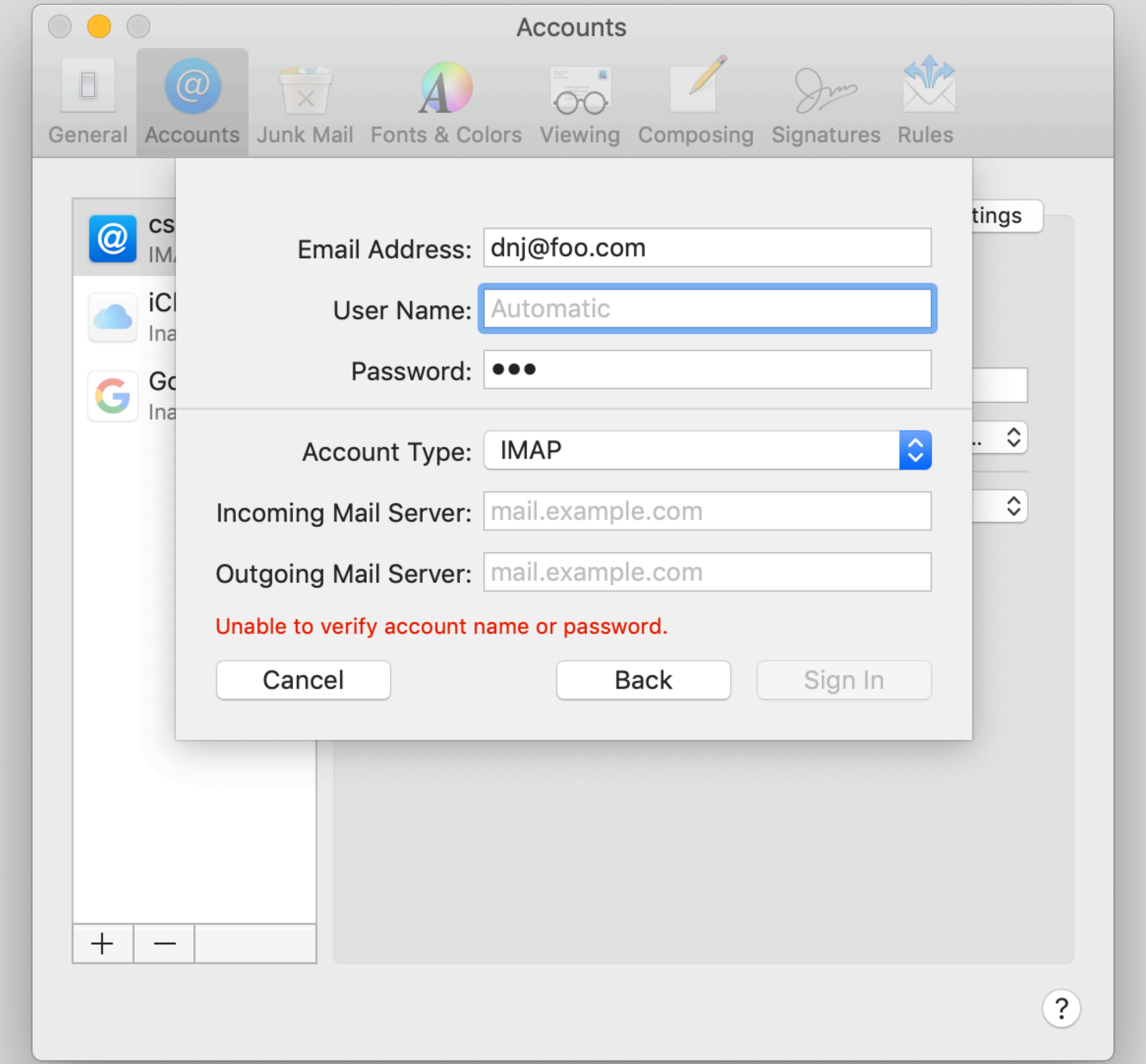
Add a Mail account

To get started, fill out the following information:

Name: Daniel Jackson

Email Address: dnj@foo.com

Password: •••



Email Address: dnj@foo.com

User Name: Automatic

Password: •••

Account Type: IMAP

Incoming Mail Server: mail.example.com

Outgoing Mail Server: mail.example.com

Unable to verify account name or password.

style/toc synergy

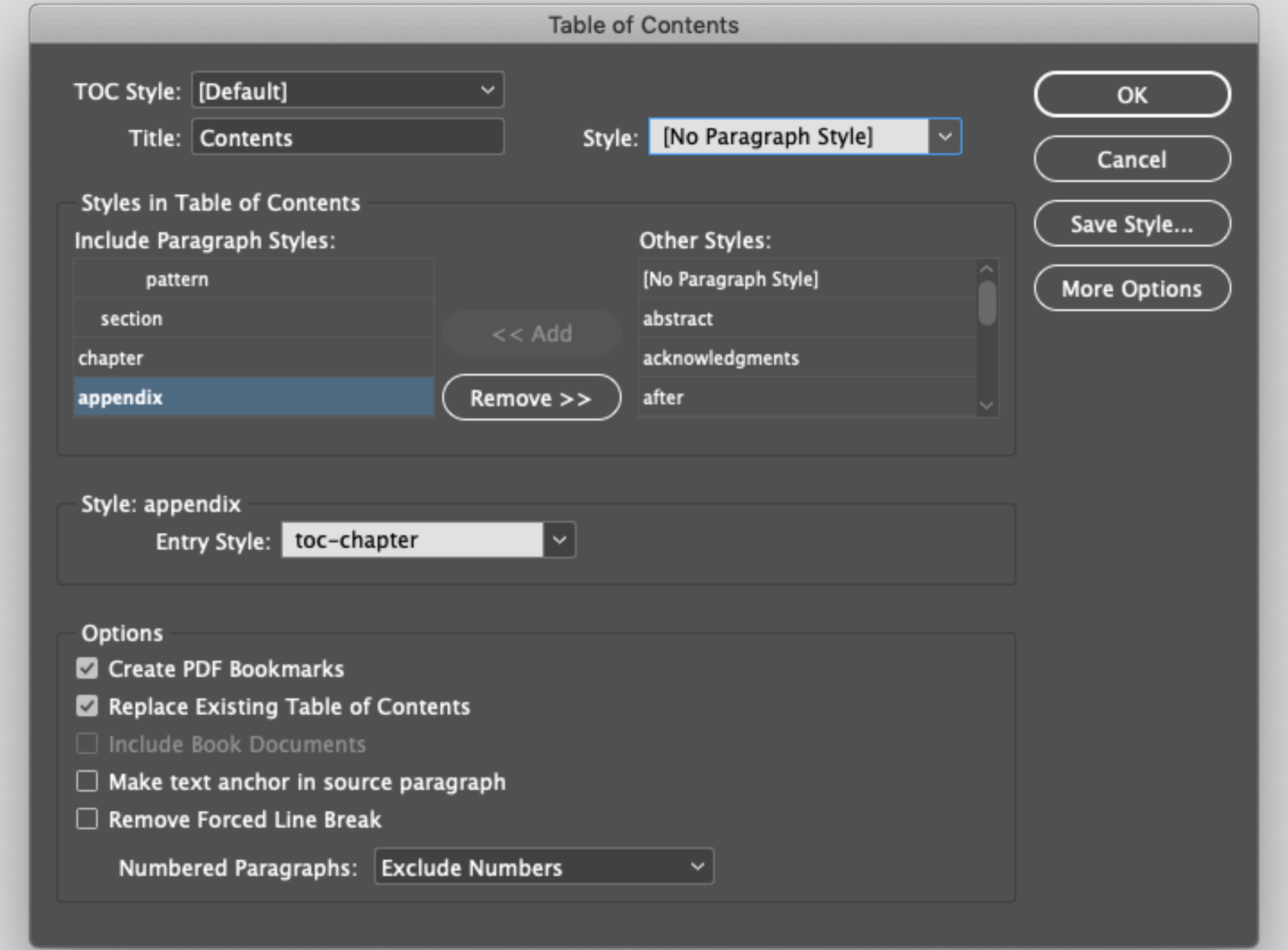


Table of Contents

TOC Style: [Default]

Title: Contents

Style: [No Paragraph Style]

Styles in Table of Contents

Include Paragraph Styles:

- pattern
- section
- chapter
- appendix

Other Styles:

- [No Paragraph Style]
- abstract
- acknowledgments
- after

Style: appendix

Entry Style: toc-chapter

Options

☑ Create PDF Bookmarks

☑ Replace Existing Table of Contents

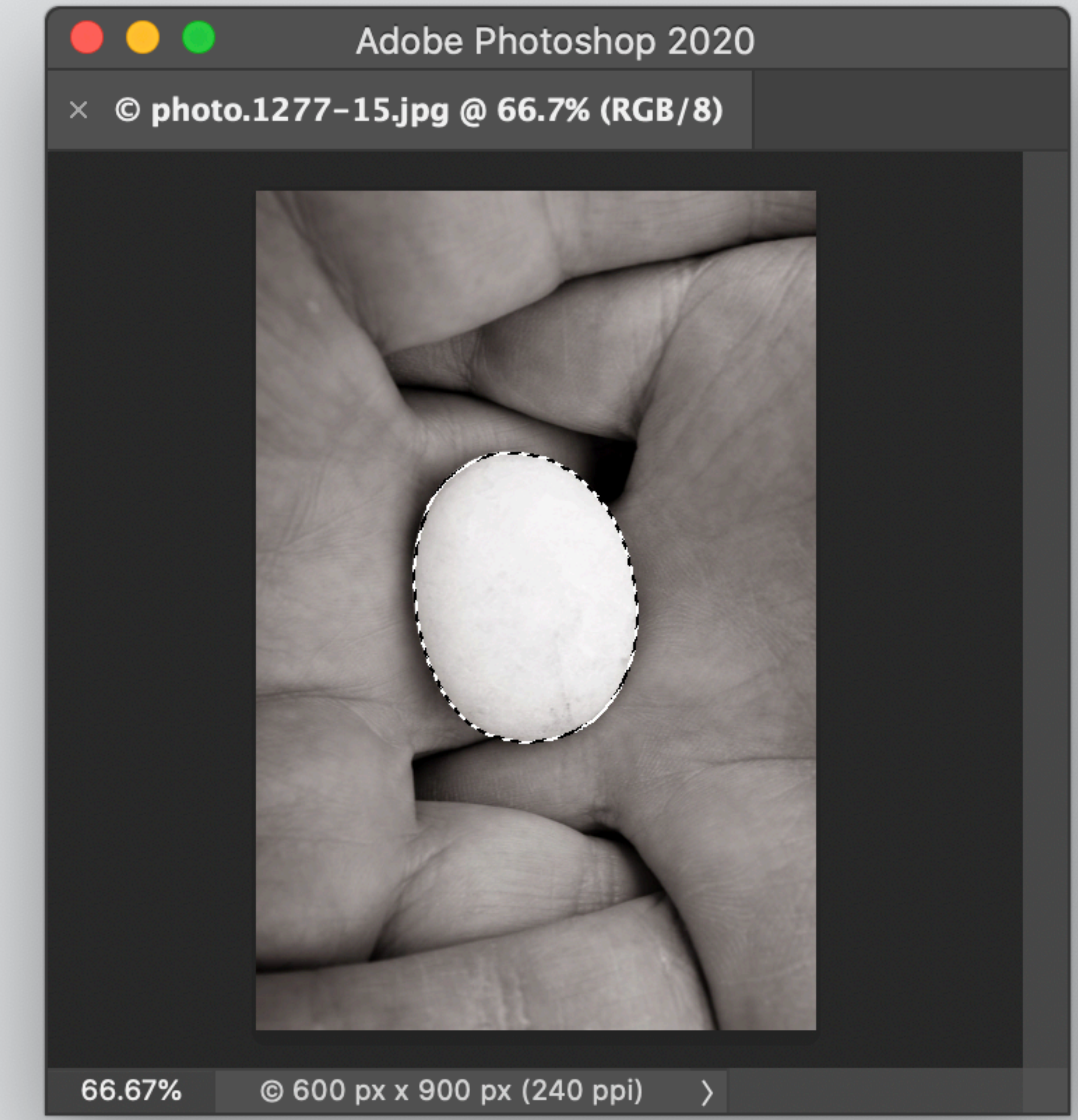
☐ Include Book Documents

☐ Make text anchor in source paragraph

☐ Remove Forced Line Break

Numbered Paragraphs: Exclude Numbers

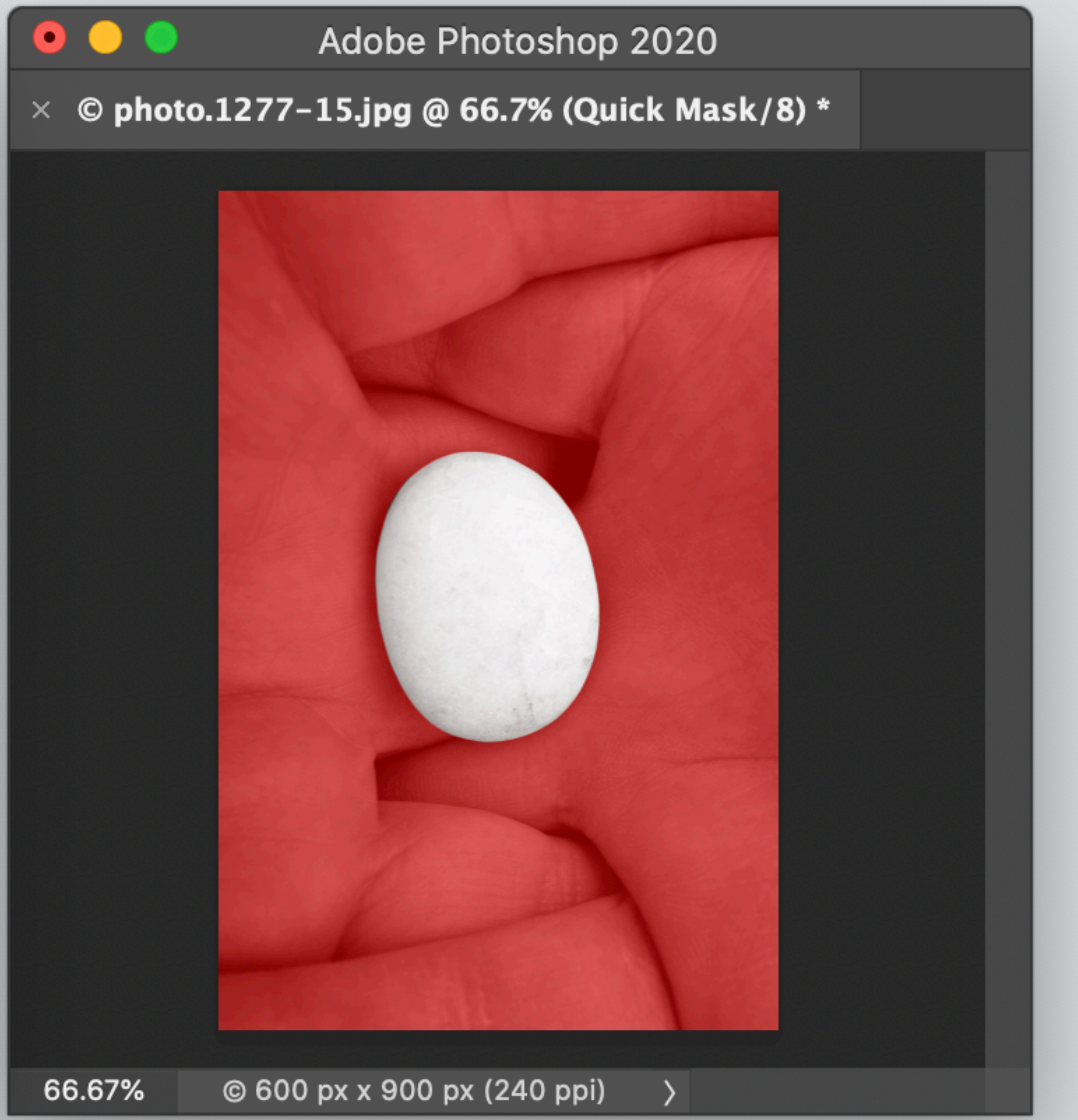
photoshop synergies



Adobe Photoshop 2020

© photo.1277-15.jpg @ 66.7% (RGB/8)

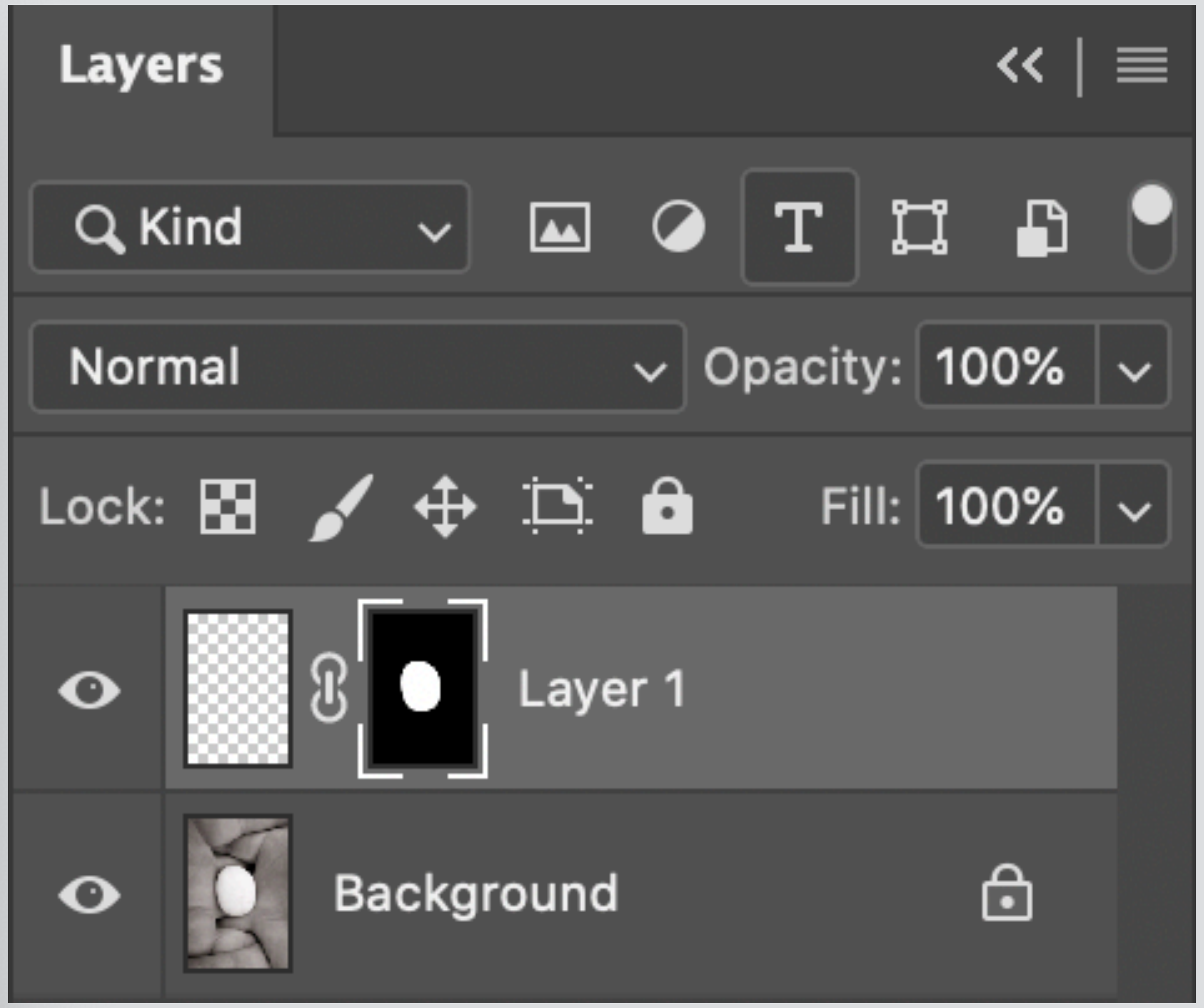
66.67% © 600 px x 900 px (240 ppi)



Adobe Photoshop 2020

© photo.1277-15.jpg @ 66.7% (Quick Mask/8) *

66.67% © 600 px x 900 px (240 ppi)



Layers

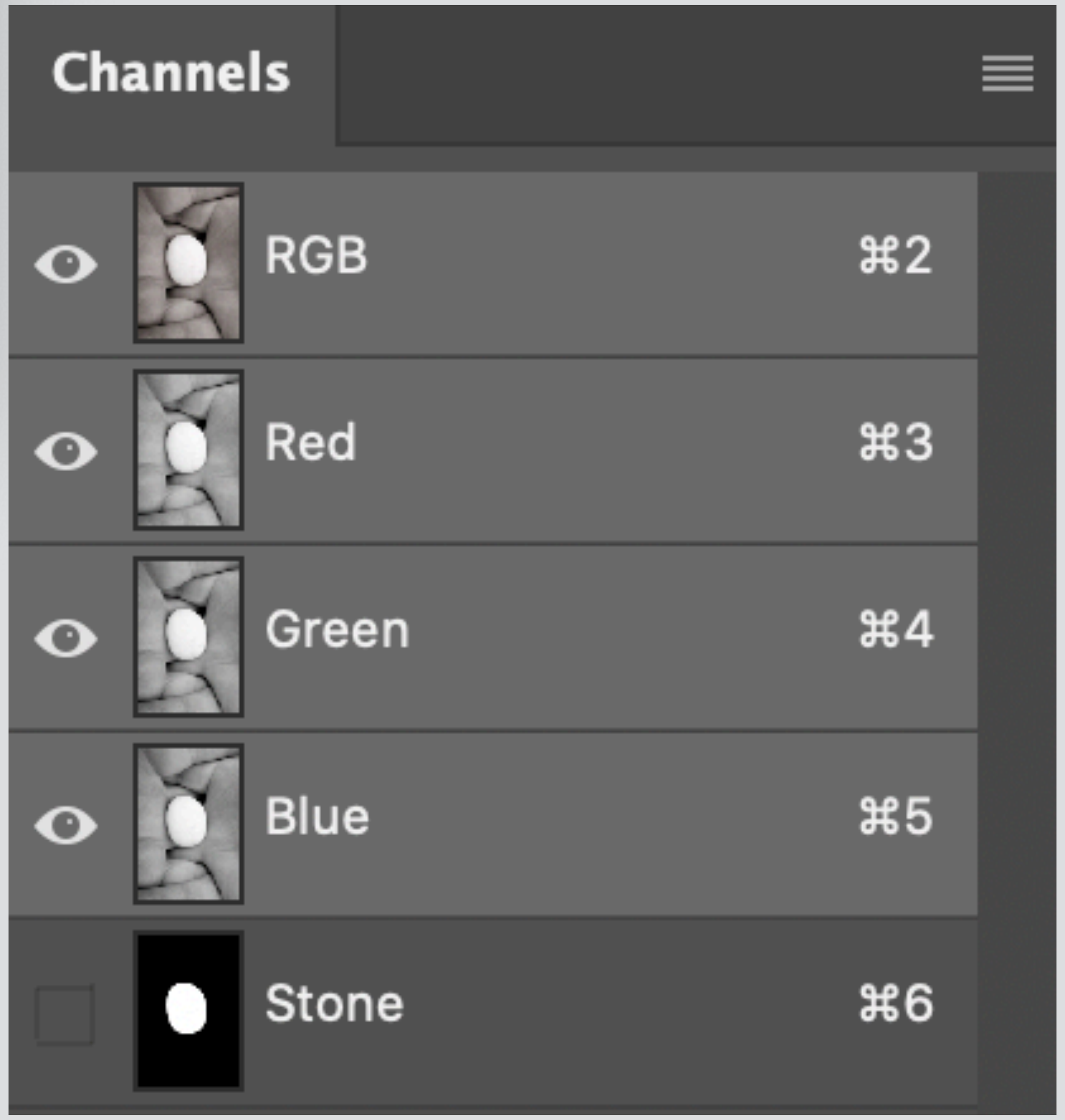
Kind

Normal Opacity: 100%

Lock: Fill: 100%

Layer 1

Background



Channels

RGB	⌘2
Red	⌘3
Green	⌘4
Blue	⌘5
Stone	⌘6

selection = mask = channel = image

the crazy power of photoshop

how to sharpen an image using an edge mask

select channel with greatest contrast
duplicate selected channel
apply Filter > Stylize > Find Edges

treat channel as image

apply Image > Adjustments > Invert
apply Filter > Other > Maximum
apply Filter > Noise > Median
apply Image > Adjustment > Levels
apply Filter > Blur > Gaussian Blur
right-click to make channel a selection

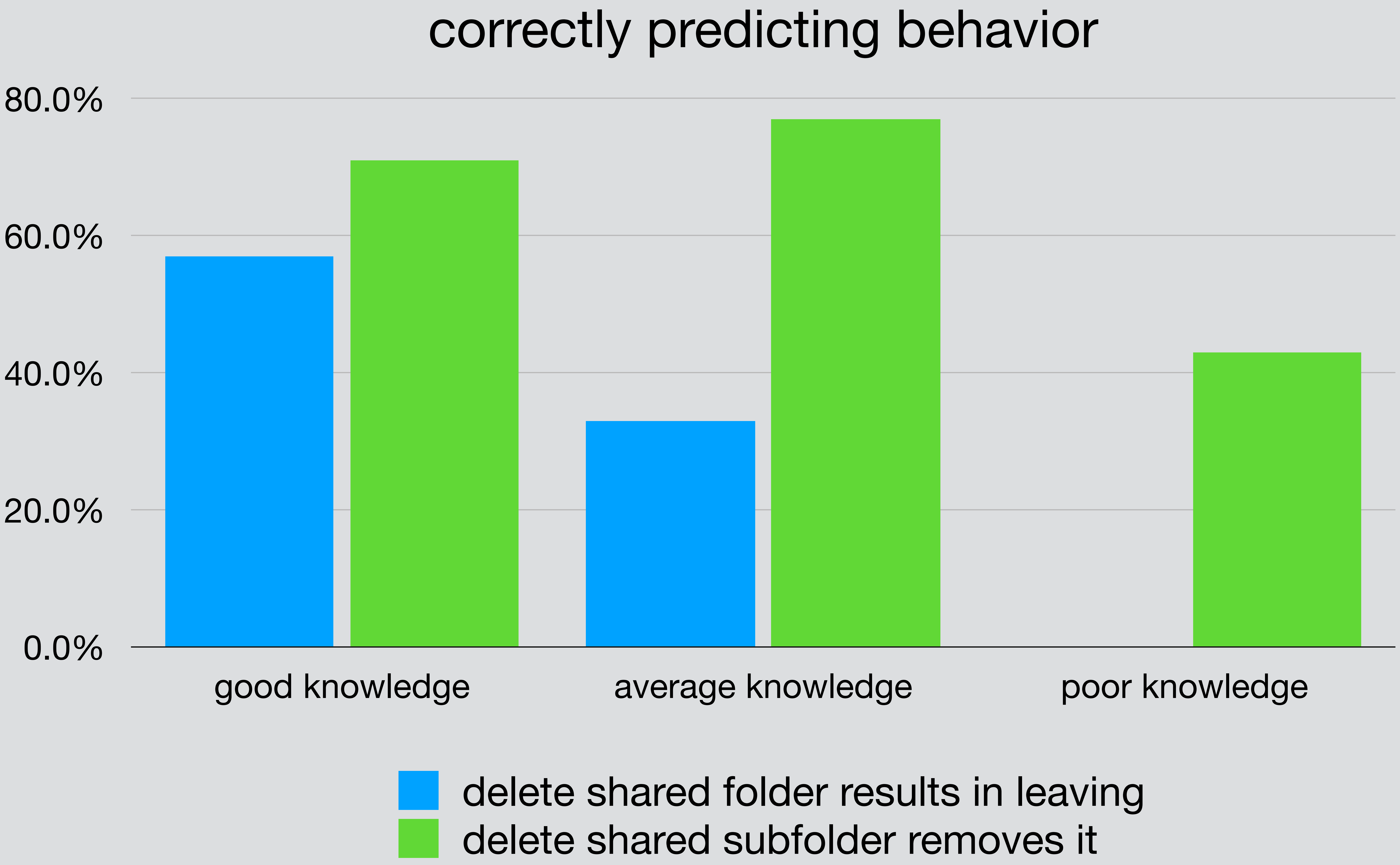
make selection from channel

select image layer
apply Select > Inverse
apply Filter > Sharpen > Unsharp Mask

apply filter using selection as mask

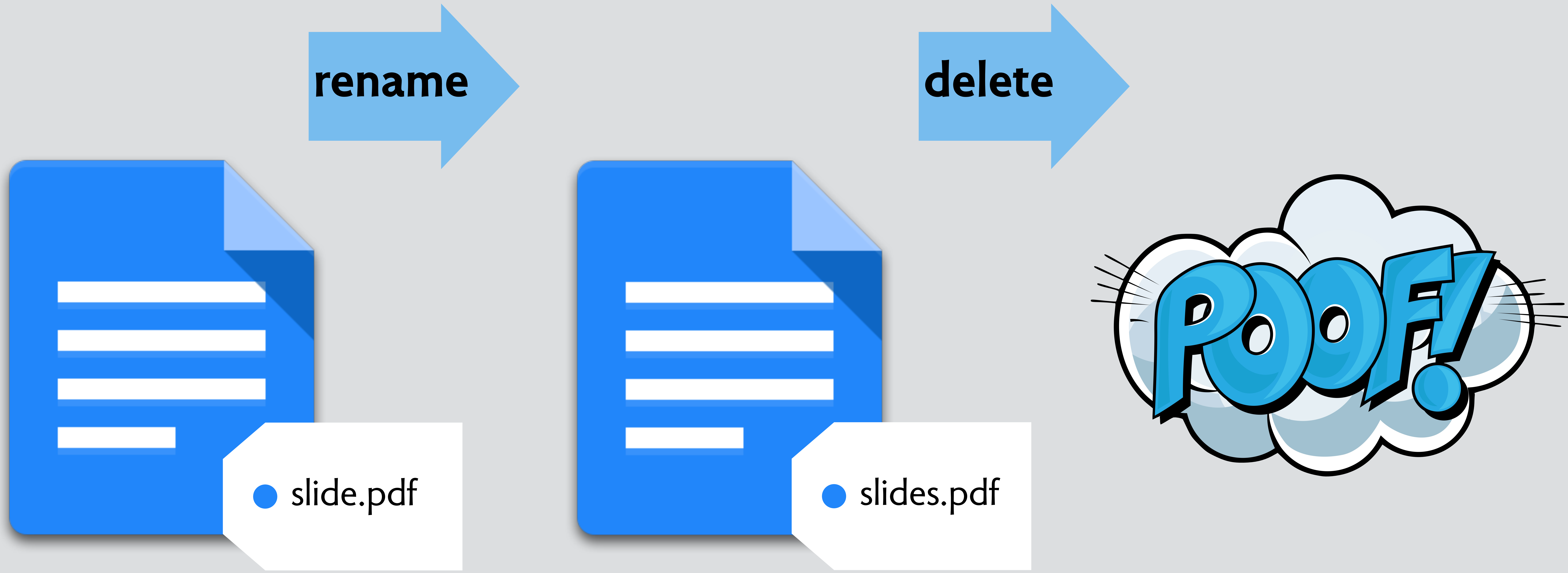
dropbox filename example

survey of dropbox users (MIT CS undergrads)



Kelly Zhang

a conceptual model of file names and deletion



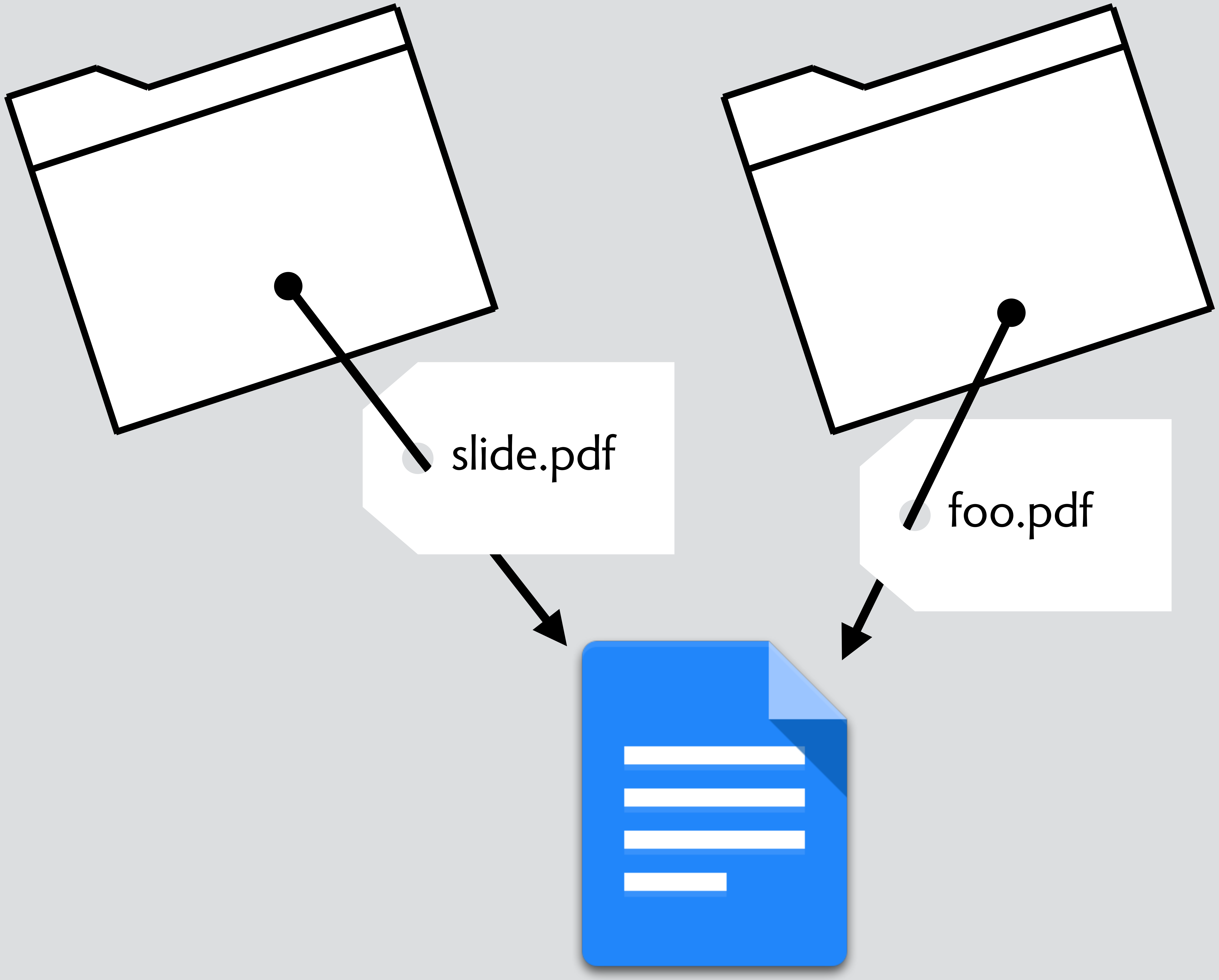
rename

delete

● slide.pdf

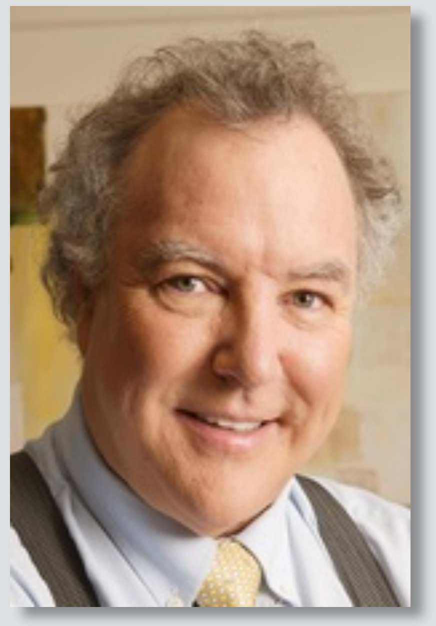
● slides.pdf

the actual model, courtesy of multics (1963-69!)



tog: conceptual models

Principle: Choose metaphors that will enable users to instantly grasp the finest details of the conceptual model

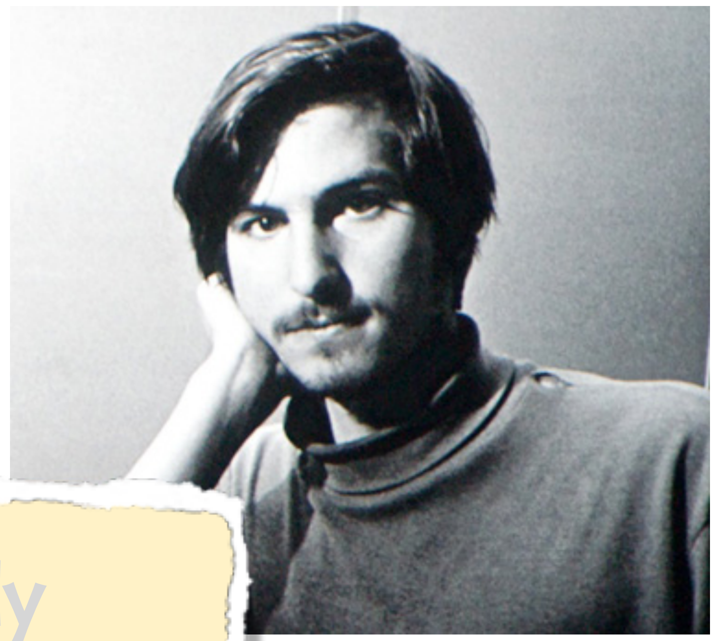


Bruce Tognazzini
First Principles of Interaction Design

brooks essence and accident

[T]o see what rate of progress one can expect in software technology, let us examine the difficulties of that technology. Following Aristotle, I divide them into **essence**, the difficulties inherent in the nature of software, and **accidents**, those difficulties that today attend its production but are not inherent.

The **essence of a software entity is a construct of interlocking concepts**: data sets, relationships among data items, algorithms, and invocations of functions. This essence is abstract in that such a conceptual construct is



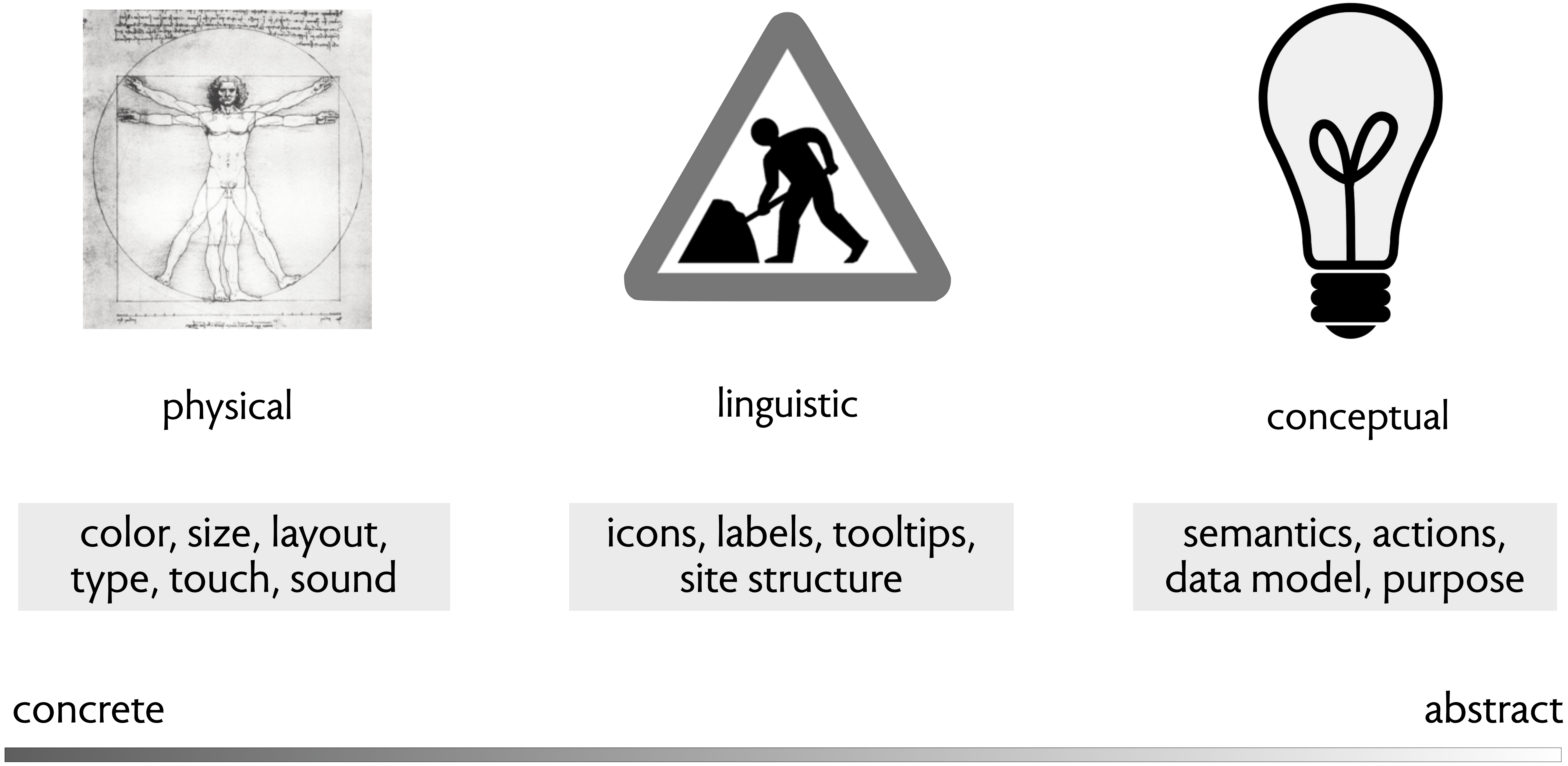
To design something really well, you have to get it. You have to really grok what it's all about. It takes a passionate commitment to really thoroughly understand something, chew it up, not just quickly swallow it. Most people don't take the time to do that.

hoare simplicity

Almost anything in software can be implemented, sold, and even used given enough determination… But there is one quality that cannot be purchased in this way—and that is reliability.

The price of reliability is the pursuit of the utmost simplicity. It is a price which the very rich find most hard to pay.

levels of UX design (export diagram)



physical

linguistic

conceptual

color, size, layout, type, touch, sound

icons, labels, tooltips, site structure

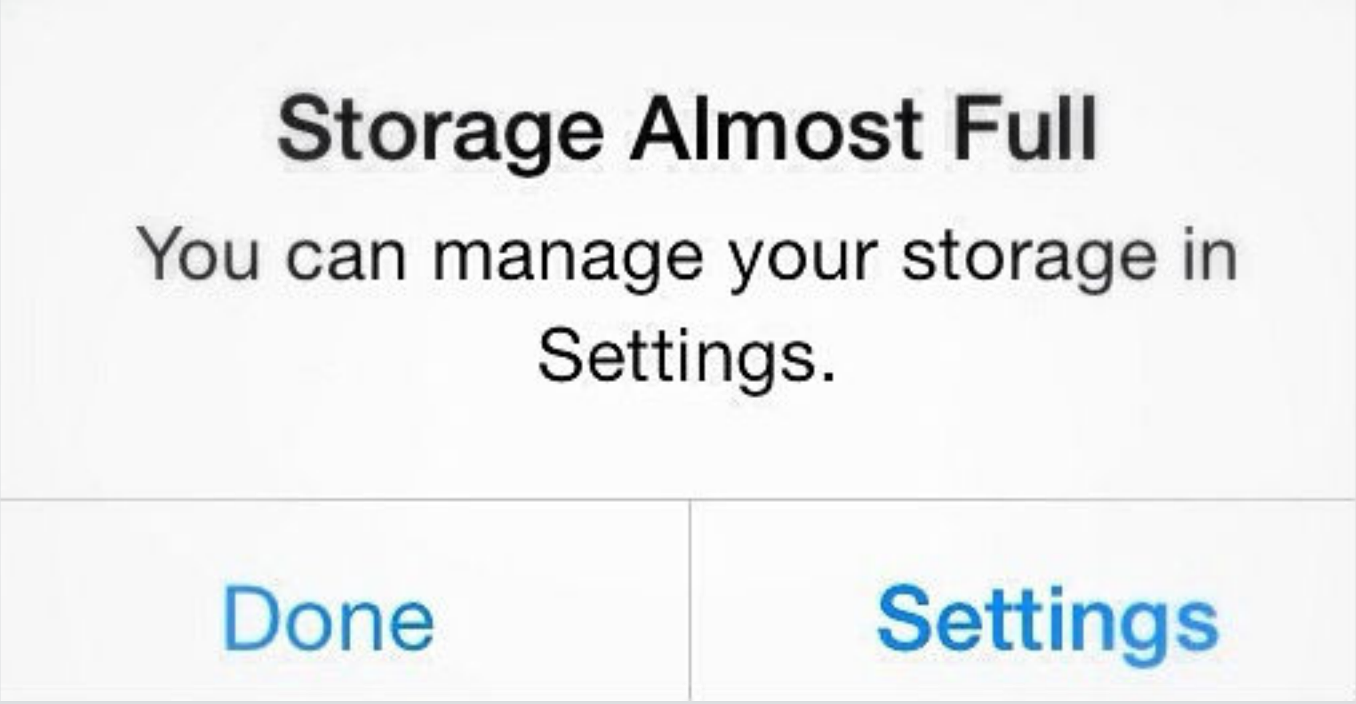
semantics, actions, data model, purpose

concrete

abstract

quality beyond correctness

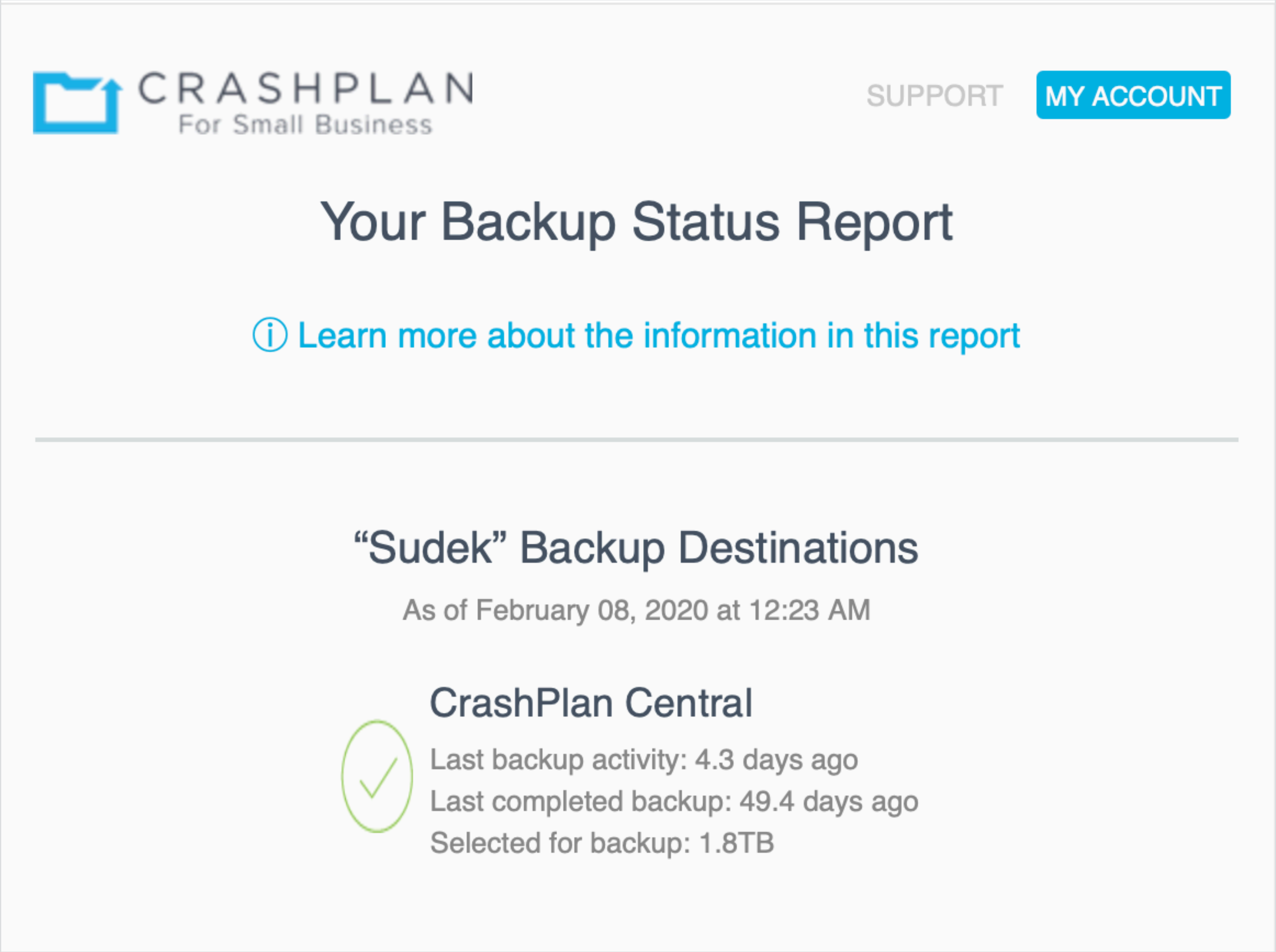
"it's not a bug, it's a feature"



Storage Almost Full

You can manage your storage in Settings.

iPhone: storage catch-22



Your Backup Status Report

ⓘ Learn more about the information in this report

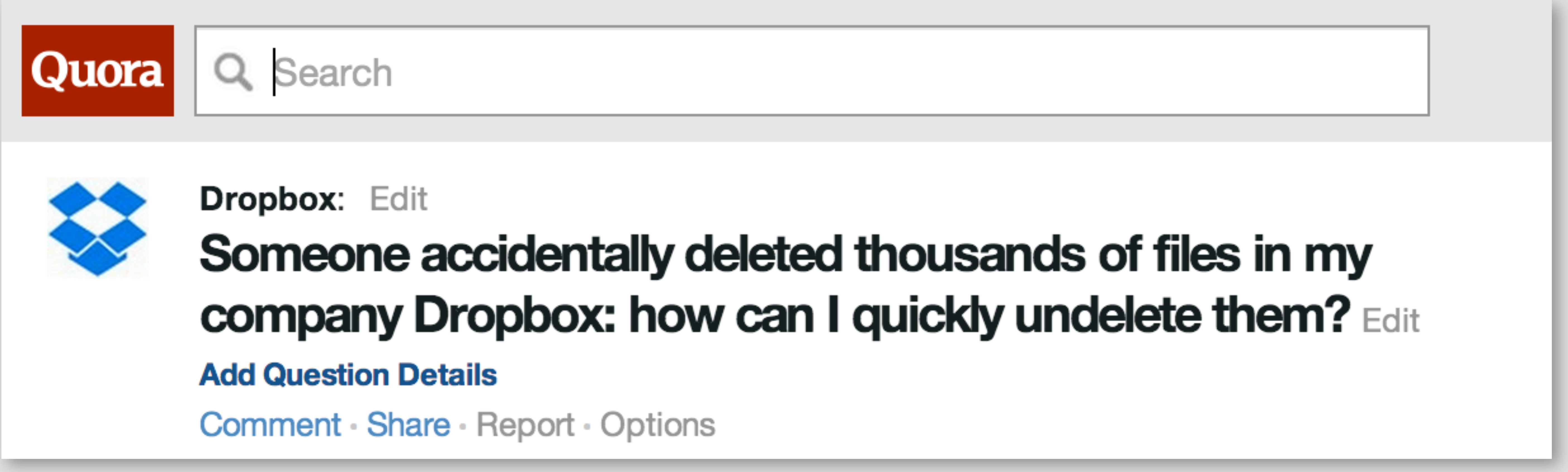
"Sudek" Backup Destinations

As of February 08, 2020 at 12:23 AM

CrashPlan Central

Last backup activity: 4.3 days ago
Last completed backup: 49.4 days ago
Selected for backup: 1.8TB

crashplan: this is success?



Dropbox: Edit

Someone accidentally deleted thousands of files in my company Dropbox: how can I quickly undelete them? Edit

Dropbox: deleting shared files

```
concept trash
```

```
purpose undo deletion
```

rationale for designer & motivation for user

```
structure
  objects, trashed: set Object
```

data model, but encapsulated

```
actions
  delete (o: Object)
    o in objects - trashed => trashed += o
  empty ()
    objects -= trashed; trashed := none
  restore (o: Object)
    o in trashed => trashed -= o
  new (o: Object)
    o !in objects => objects += o
```

succinct & precise behavior

```
principle
… delete(o); restore(o) {o in objects - trashed}
… delete(o); empty() {o !in objects}
```

archetypal scenario, explains essence of design

concept reservation

purpose consistent formatting

structure

```
 slots: Owner -> Slot
 holds: User -> Slot
```

actions

```
 create (o: Owner, s: Slot)
   no slots.s => slots += o -> s
 reserve (u: User, o: Owner, s: Slot)
   no holds.s and o -> s in slots => holds += u -> s
 cancel (u: User, s: Slot)
   u -> s in holds => holds -= u -> s
 use (u: User, o: Owner, s: Slot)
   u -> s in holds and o -> s in slots =>
```

principle

```
 if create and reserve and not cancel then can use
```

name: essential for knowledge capture

purpose: why the concept exists

structure: localized data model

actions: observable & atomic

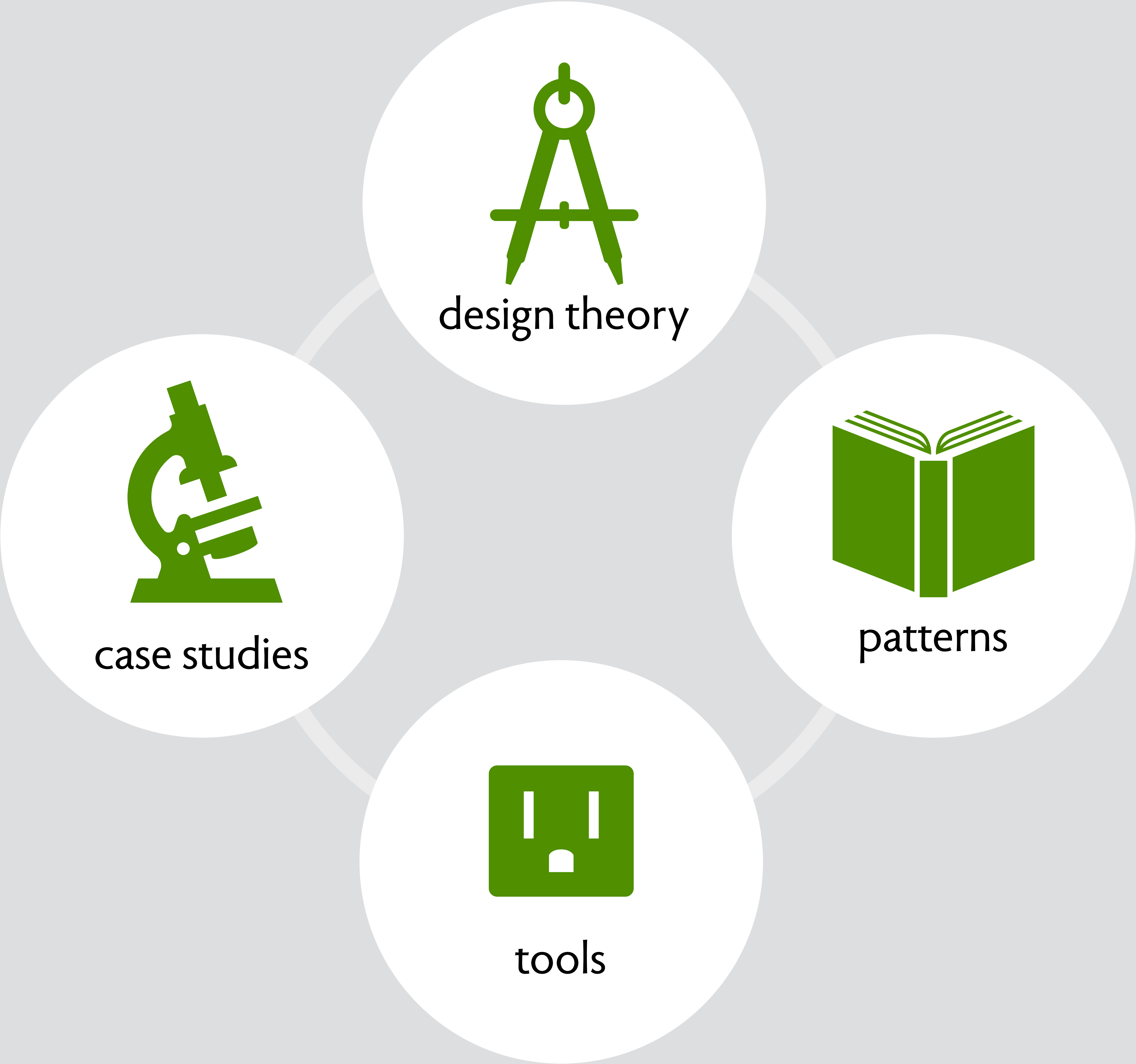
OP justifies design and explains it

shows how behavior fulfills purpose

elements of a **concept design method**

structure: how to express & combine concepts	separation of concerns: easier to focus, divide labor
principles: applicable distillation of experience	avoiding predictable pitfalls, speeding up design
patterns: handbook of known concepts & issues	capturing expertise and experience for better design
tools: exploit computing for analysis & synthesis	catching subtle flaws, reducing manual effort

a research & teaching program



design theory

case studies

patterns

tools

Daniel Jackson

design by concept

a new way to think about software

principle: make concepts modular

concepts have **no dependences**

✓ trash does not “use” deleted labels

concepts **encapsulate decisions**

✓ labels independent of folder structure

✗ Facebook tags change access control

concepts are **polymorphic**

✓ label items not folders

✗ Twitter tweet content determines if reply or not

modularity groups

simple group functionality

user can create a new group
other users can request to join
users can contribute posts to the group
and can read other user's posts

modularity group, most granular concepts

concept Group
state
owner, members: Group -> User
assets: Group -> Asset
actions
create (owner: User, **out** g: Group)
join (u: User, g: Group)
contribute (u: User, g: Group, a: Asset)
access (u: User, a: Asset)

concept Request
state
owns, requested, granted, denied: User -> Resource
actions
register (owner: User, r: Resource)
request (u: User, r: Resource)
respond (o, u: User, r: Resource, answer: bool)

concept Post
state
author: Post -> Author
content: Post -> String
actions
new (a: Author, s: String, out p: Post)
edit (p: Post, s: String)
get (a: Author, out ps: set Post)

sync newGroup (o: User, **out** g: Group)
Request.register(o, g)
Group.create(o, g)

sync requestJoin (u: User, g: Group)
Request.request(u, g)

sync join (o, u: User, g: Group)
Request.respond (o, u, g, true)
Group.join (u, g)

sync post (u: User, g: Group, s: String, **out** p: Post)
Post.new (u, s, p)
Group.contribute (u, g, p)

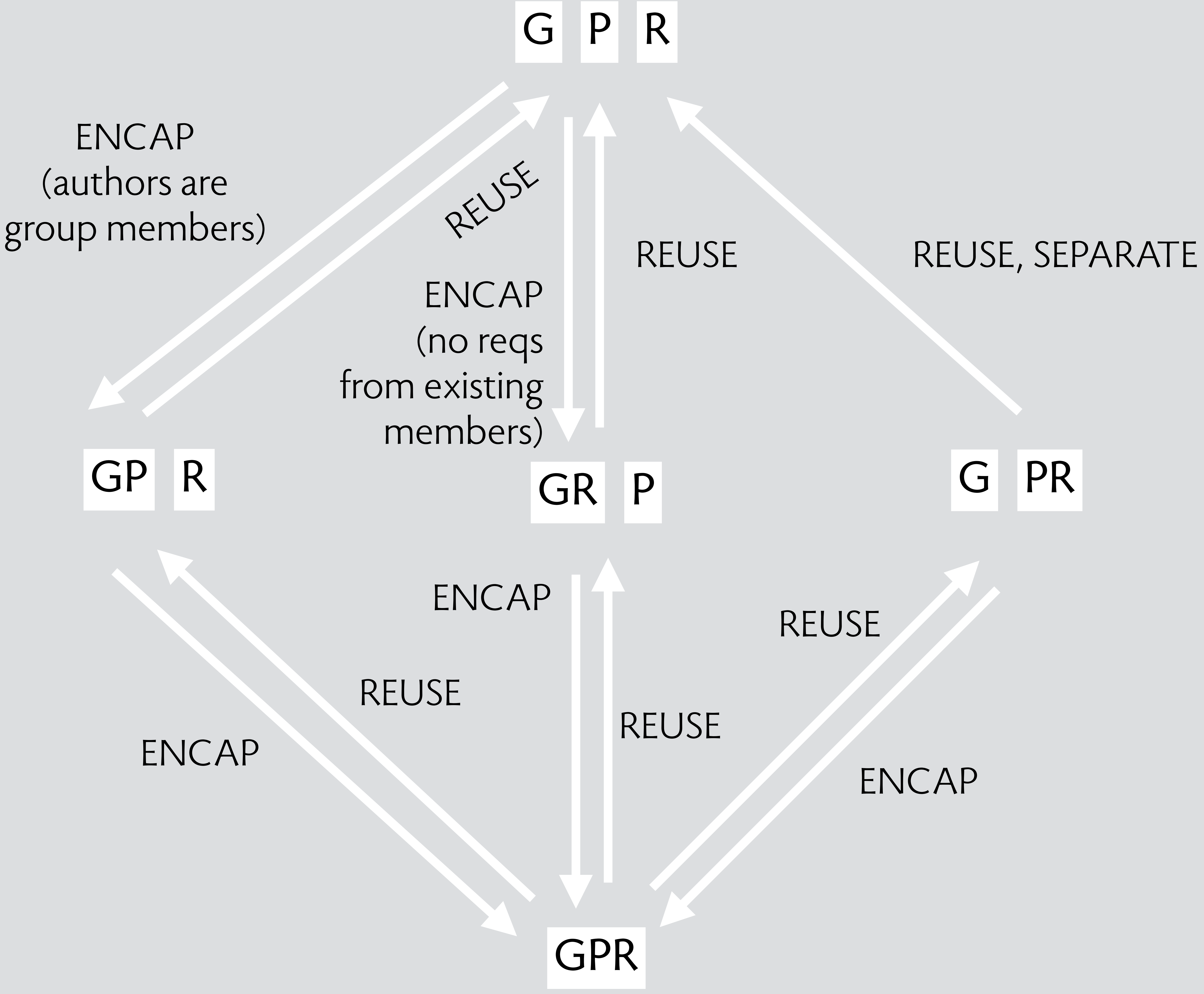
modularity design moves

REUSE
what: break into concepts that can be used independently
when: new concept is more focused, stands alone, and usable in other contexts

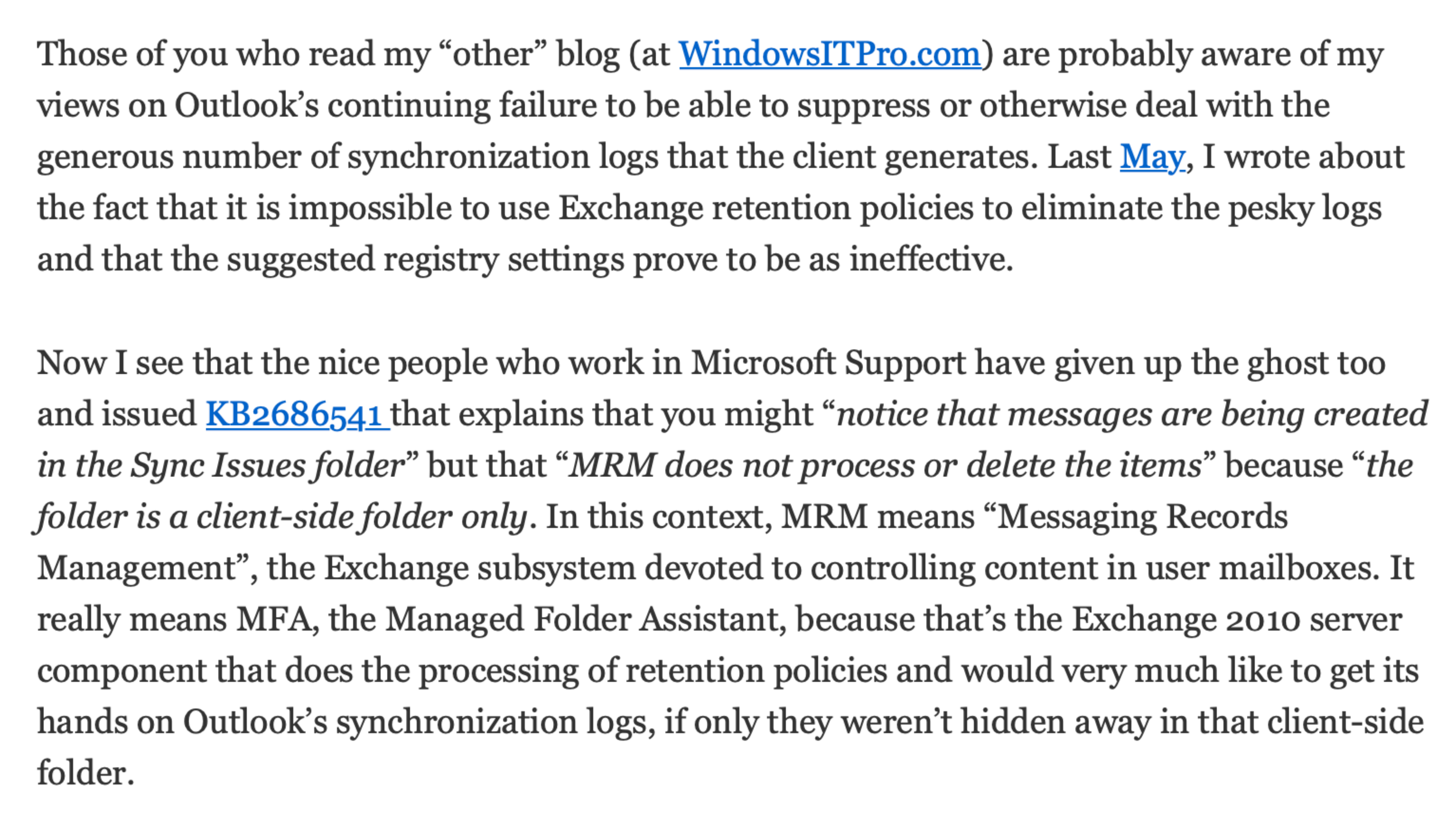
SEPARATE
what: factor out disjoint functionalities into separate concepts
when: some subsets of actions and states are decoupled; unclear purpose

ENCAPSULATE
what: bring functionality together to localize design decisions
when: invariants and couplings cross concept boundaries, and complicate sync

modularity design moves for group/post/request concepts



overloading outlook sync issues



Those of you who read my “other” blog (at WindowsITPro.com) are probably aware of my views on Outlook’s continuing failure to be able to suppress or otherwise deal with the generous number of synchronization logs that the client generates. Last May, I wrote about the fact that it is impossible to use Exchange retention policies to eliminate the pesky logs and that the suggested registry settings prove to be as ineffective.

Now I see that the nice people who work in Microsoft Support have given up the ghost too and issued KB2686541 that explains that you might “*notice that messages are being created in the Sync Issues folder*” but that “*MRM does not process or delete the items*” because “*the folder is a client-side folder only*. In this context, MRM means “Messaging Records Management”, the Exchange subsystem devoted to controlling content in user mailboxes. It really means MFA, the Managed Folder Assistant, because that’s the Exchange 2010 server component that does the processing of retention policies and would very much like to get its hands on Outlook’s synchronization logs, if only they weren’t hidden away in that client-side folder.

synchronization logs are stored as messages in email folders
naturally, not sync’d with server
but create storage leak and can’t be accessed by admins

https://thoughtsofanidlemind.com/2012/08/29/outlook-sync-issue/