Because you're a nice person

Library UX

Using abstraction towards friendlier APIs

Mali Akmanalp
(@makmanalp)

hear at the back

Who here has had to write code that other people have to consume?

bit.ly/abstraction-talk
(@makmanalp)

presenter notes

links

tidbits

UX

== User Experience

== How this makes me feel

How do I feel when I use this thing?

excited / frustrated

product companies think about UX - e.g. apple products

UX

== User Experience

== Usability

overlapping

How easy is it to learn and use this thing?

—

both come late into consideration, if ever

talk by kenneth reitz

```python
import urllib2

gh_url = 'https://api.github.com/user'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

https://www.kennethreitz.org/python-for-humans/

github API

HTTP request

custom class

```python
import requests

url = 'https://api.github.com/user'
auth = ('username', 'password')

r = requests.get(url, auth=auth)
print r.content
```

https://www.kennethreitz.org/python-for-humans/

Drastically different experience

“urllib is so terrible” || mad at me

Don't jump to conclusions / Hold that thought - Alternate theory

"For Humans"

code is for people

people have feelings

Why care about UX?

Why care about how users feel?

(nice person) many reasons

Good UX
reduces mistakes.

n

```python
import urllib2

gh_url = 'https://api.github.com/user'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

What is each parameter?

noun_verb or verbnoun?

reduces mistakes

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4173163/

```python
import urllib2

gh_url = 'https://api.github.com/user'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

What is each parameter?

noun_verb or verbnoun?

reduces mistakes

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4173163/

```python
import urllib2

gh_url = 'https://api.github.com/user'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

What is each parameter?

noun_verb or verbnoun?

reduces mistakes

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4173163/

```python
import urllib2

gh_url = 'https://api.github.com/user'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, 'user', 'pass')

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

What is each parameter?

noun_verb or verbnoun?

reduces mistakes

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4173163/

Good UX
minimizes distractions.

I don’t care about your lib's impl details

it's not about you - it's about me

Your library is a means to my end.

Just let me do my job!

Good UX
makes complex tasks routine.

“batteries included”

feeling of tools in my toolbox

boilerplate song and dance / incantations

Requests + beautifulsoup + pandas + seaborn

Good UX
drives adoption.

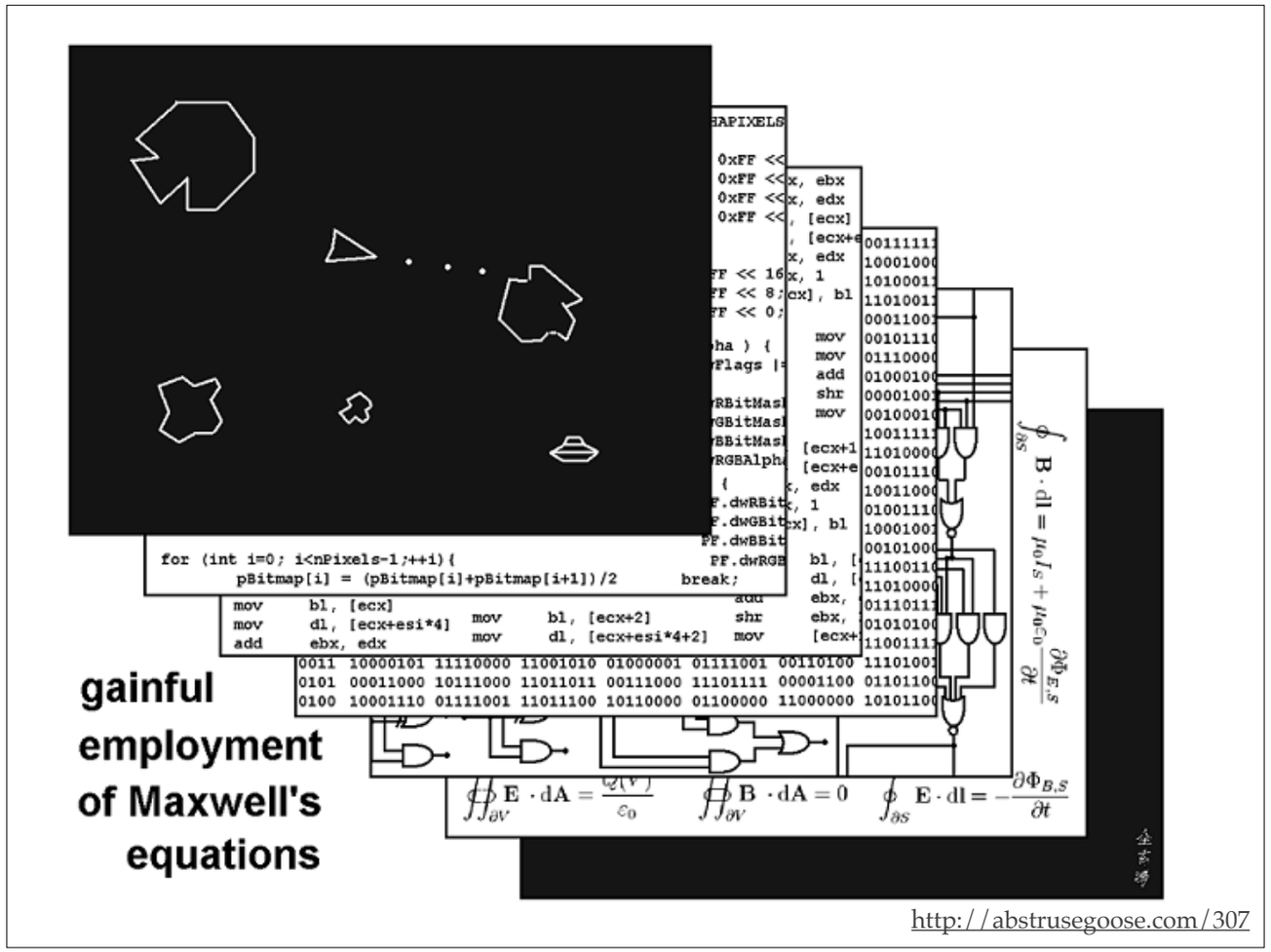
n

(sometimes Built-in) alternatives

?

how do we get better UX?

switch gears / seemingly unrelated



http://abstrusegoose.com/307

abstraction

analyze the joke to death

stack of pancakes

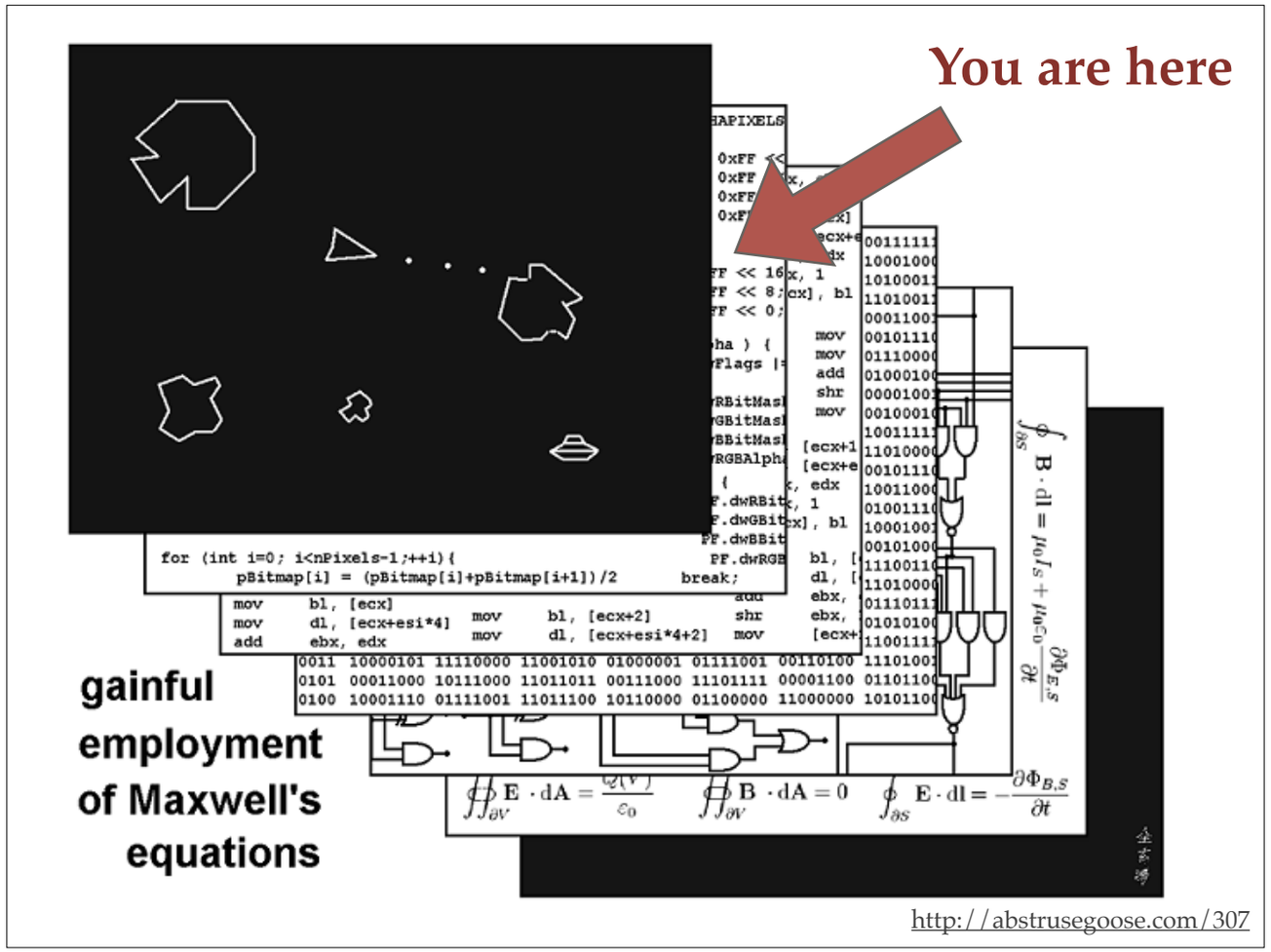
asteroids->empty void of space

Abstraction works even if the architects don't know the very blocks they're building upon

Claim:

We are primarily in the business of dealing with abstractions.

we would do well to pay more attention to them



http://abstrusegoose.com/307

Python: fall in the middle

theory Py aweosmeness: HL enough -> goal-driven needs

C

n - but what is abs really?

Abstraction is about hiding details in a controlled way.

(pause)

why hide?

Hiding details
helps reduce mistakes.

n

```python
import requests

url = 'https://api.github.com/user'
auth = ('username', 'password')

r = requests.get(url, auth=auth)
print r.content
```

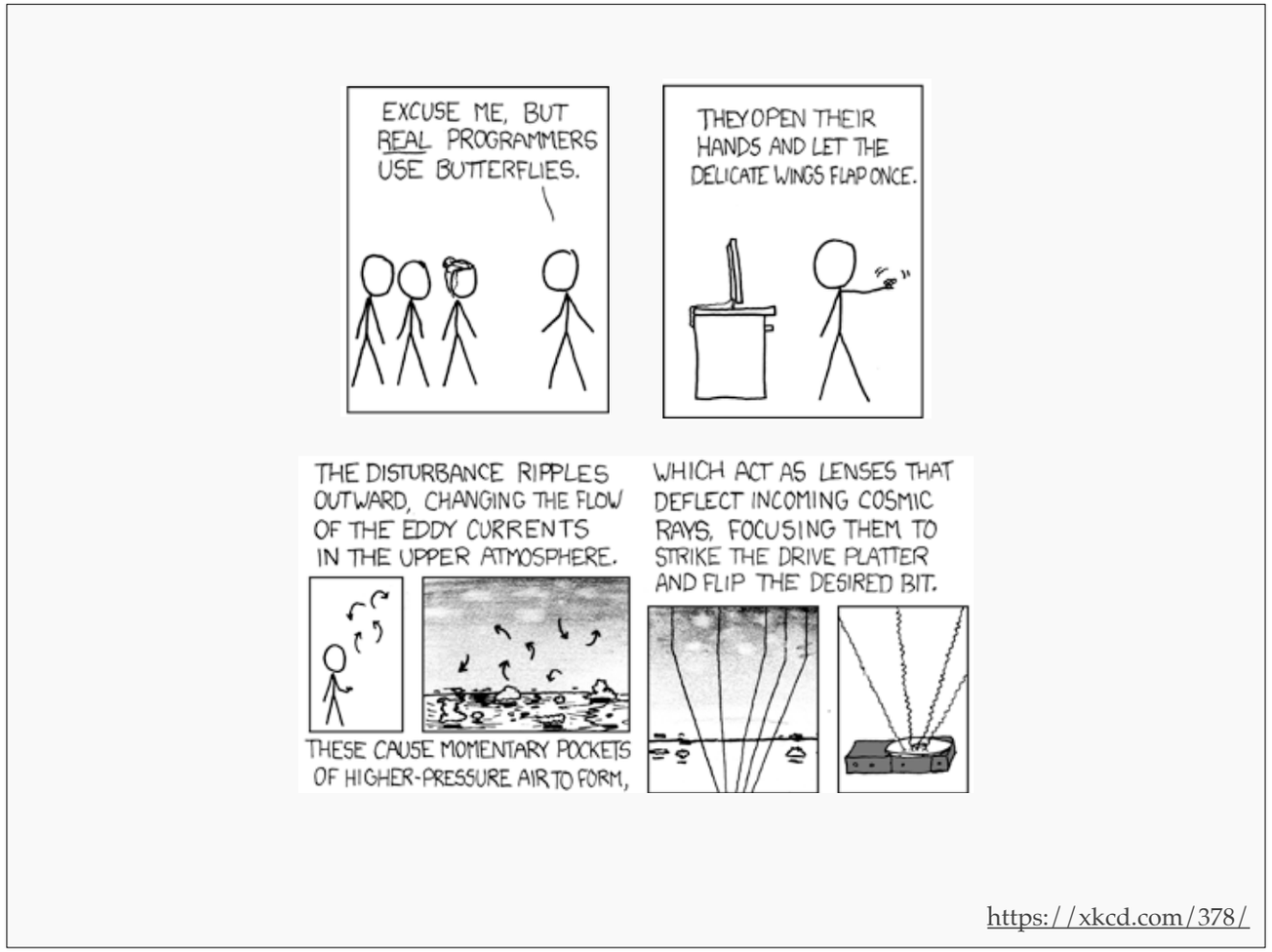
https://www.kennethreitz.org/python-for-humans/

requests is highly abstracted

less code == less errors

Hiding details
makes complex tasks routine.

Cool thing: I don’t need to know maxwell’s eqs. to make a game
helps reason about complex ideas



https://xkcd.com/378/

make fun of “real programmer”

Butterfly based disk I/O

most of your time is spent thinking about butterflies

Hiding details
provides a stable interface.

changes under the hood zero-cost for your users

testing is easier!

Claim:

Good abstraction is aligned with good UX.

mentioned things as "Good UX"

ABSTRACTION IN PYTHON

what tools do we have?

Functions

```
>>> a
array([[ 2.,  8.,  0.,  6.],
       [ 4.,  5.,  1.,  1.],
       [ 8.,  9.,  3.,  6.]])


>>> np.resize(a, (2,6))
array([[ 2.,  8.,  0.,  6.,  4.,  5.],
       [ 1.,  1.,  8.,  9.,  3.,  6.]])
```

functions abstract away the details

edge cases

blasphemies: speed

Classes

```
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String(50))
    dessert = Column(String(50))
```

another tool

sqlalchemy models

magic

Classes

```
mali = User(name="mali",
            fullname="Mali Akmanalp",
            password="pumpkin")
```

Classes

```
>>> mali.name
"mali"

>>> mali.name = "mali2"

>>> session.add(mali)
>>> session.commit()
```

classes make state explicit and organized

group state and behavior - clean

behavior that changes with state

PITFALLS

With great power comes great responsibility.

n

Leaky abstractions

when the details refuse to be hidden

http://www2.parc.com/csl/groups/sda/publications/papers/Kiczales-IMSA92/for-web.pdf

Leaky Abstractions

```
size = 1000
big_table = [list(range(size)) for _ in range(size)]
```

1000x1000

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

n

Leaky Abstractions

```
In [5]: %%timeit
   ...: for i in range(size):
   ...:     for j in range(size):
   ...:         x = big_table[j][i]
   ...:
10 loops, best of 3: 163 ms per loop
```

Leaky Abstractions

```
In [5]: %%timeit
   ...: for i in range(size):
   ...:     for j in range(size):
   ...:         x = big_table[i][j]
   ...:
10 loops, best of 3: 97.1 ms per loop
```

???

Leaky Abstractions

```
In [5]: %%timeit
   ...: for i in range(size):
   ...:     for j in range(size):
   ...:         x = big_table[i][j]
   ...:
10 loops, best of 3: 97.1 ms per loop
```

???

Leaky Abstractions

```
In [5]: %%timeit
   ...: for i in range(size):
   ...:     for j in range(size):
   ...:         x = big_table[i][j]
   ...:
10 loops, best of 3: 97.1 ms per loop
```

???

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Leaky Abstractions

```
[[ 2,  8, …,  0,  6],
 [ 4,  5, …,  1,  1],
 [ …,  …, …,  …,  …],
 [ …,  …, …,  …,  …],
 [ 8,  2, …,  5,  6],
 [ 8,  9, …,  3,  6]])
```

against the grain

"law of leaky abstractions"

acceptable compromise

Under-abstraction

n

Guts everywhere
State everywhere
Control flow everywhere

Hard to understand things

Sign: People w/ domain knowledge have a hard time understanding

Over-abstraction

watch out for the following

Coupling:

To change one thing, you must change all things.

n

Cohesion:

A thing that does too many things at the same time.

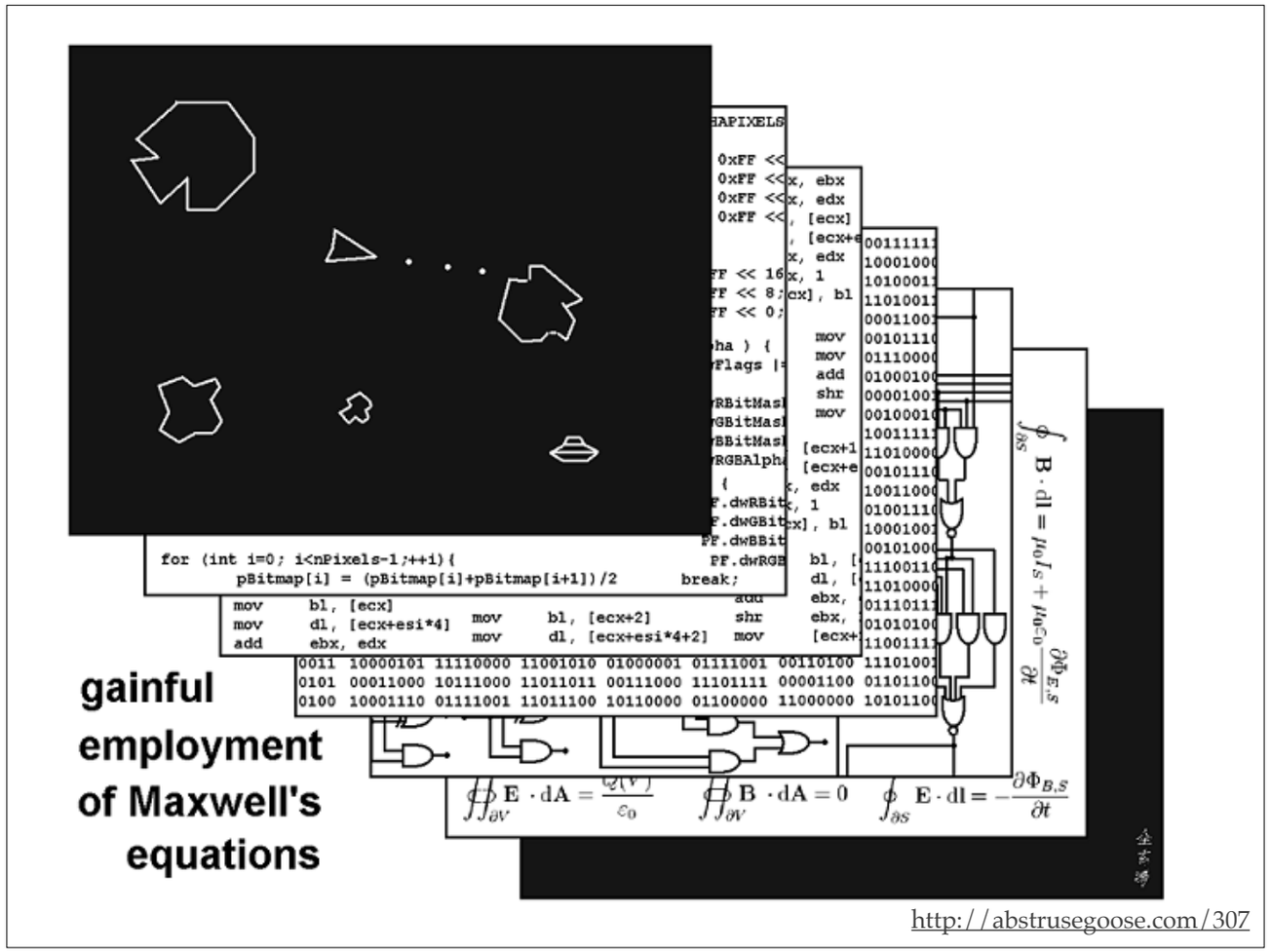
low-cohesion (contents not related to each other)

n

DECIDING ON THE LEVEL OF ABSTRACTION

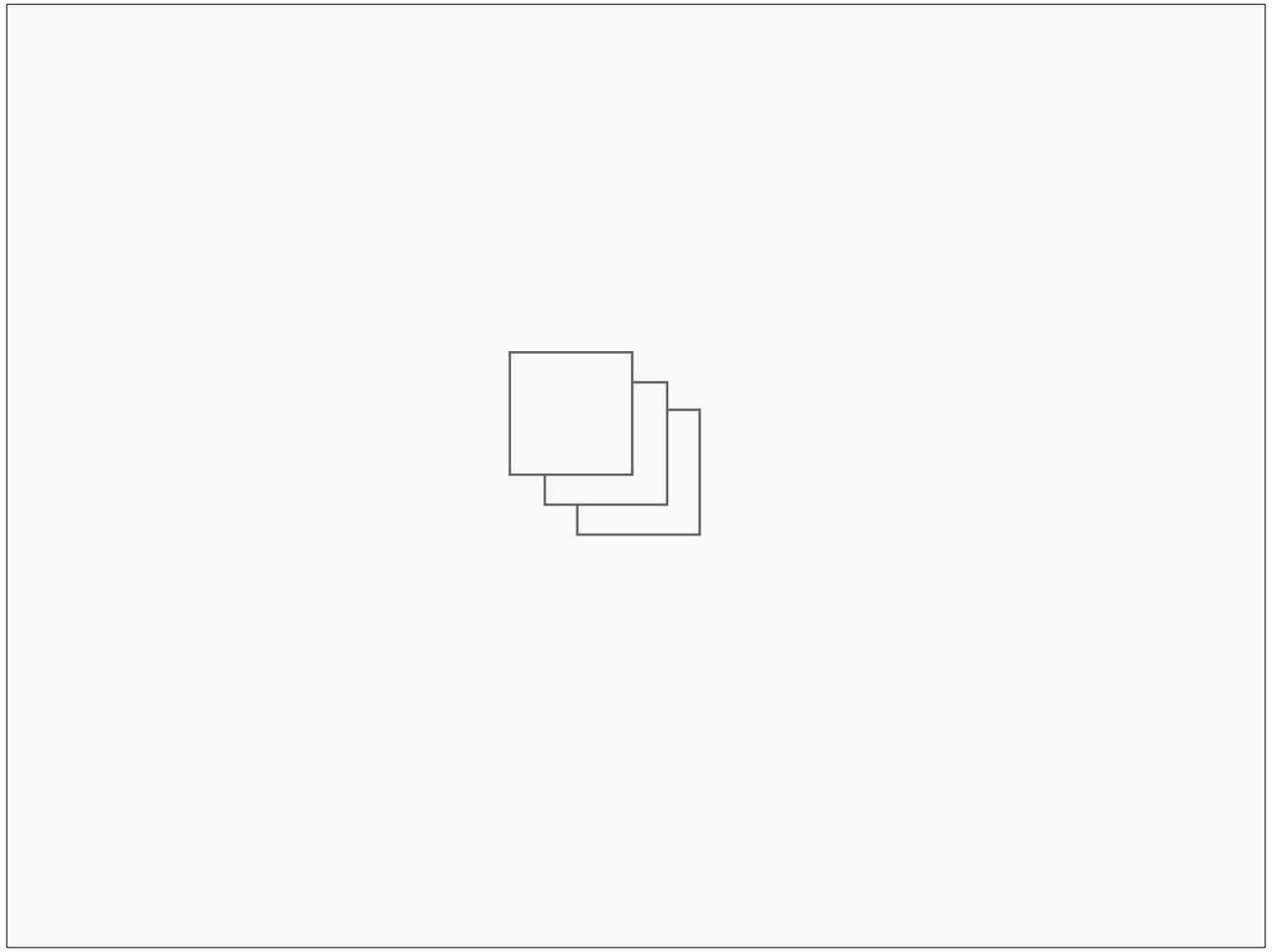
so, you're building a shiny new app

how to structure abstraction stack

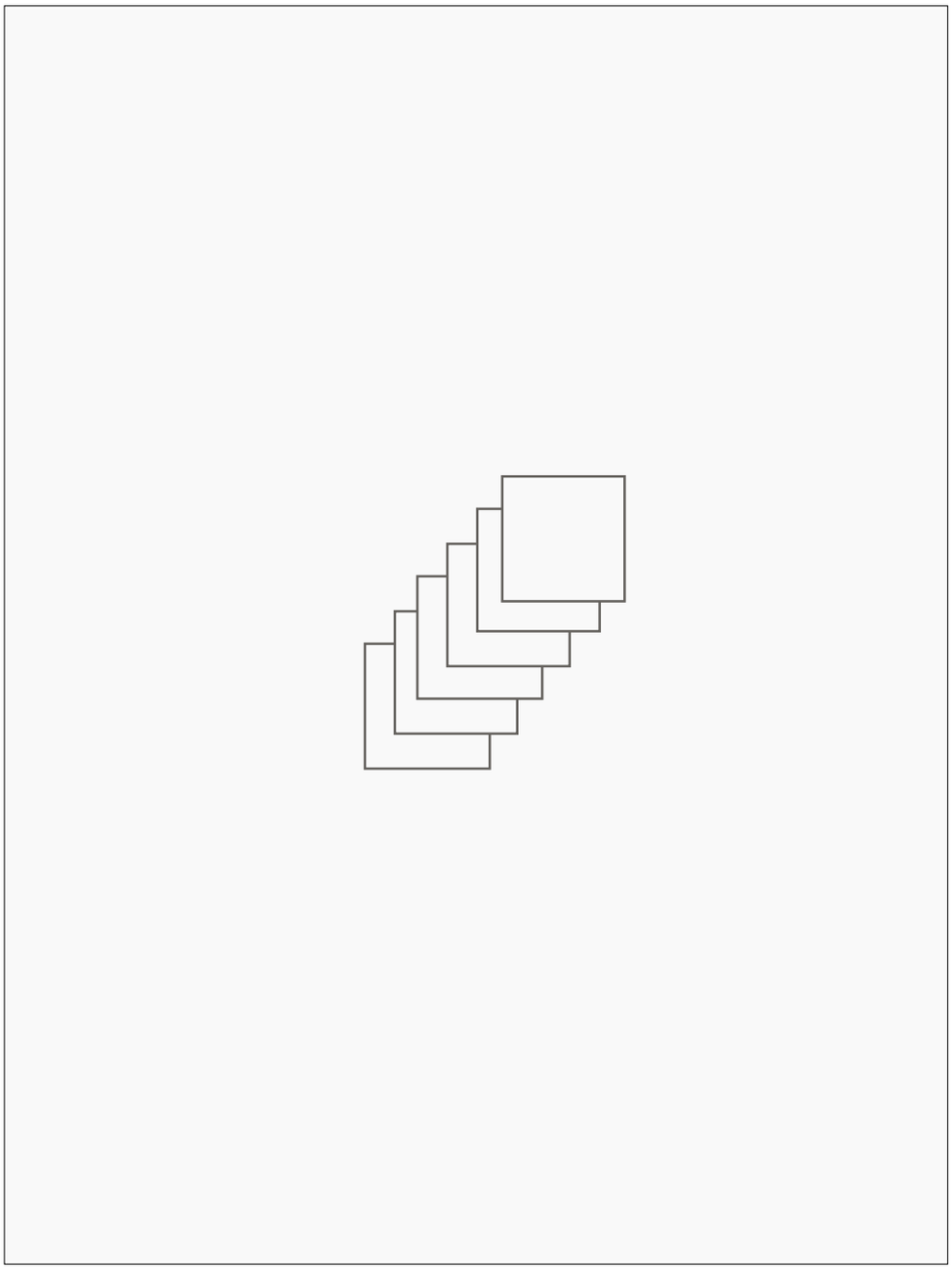


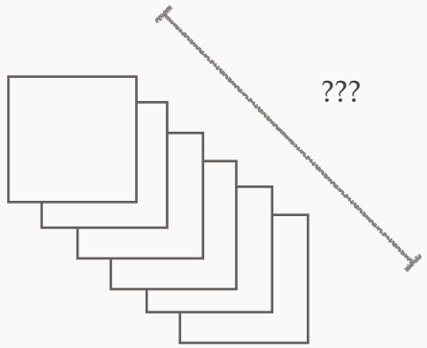
back to this model

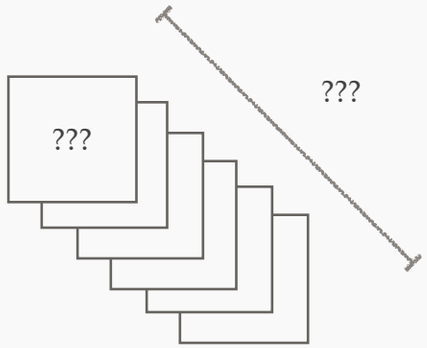
simplify a little

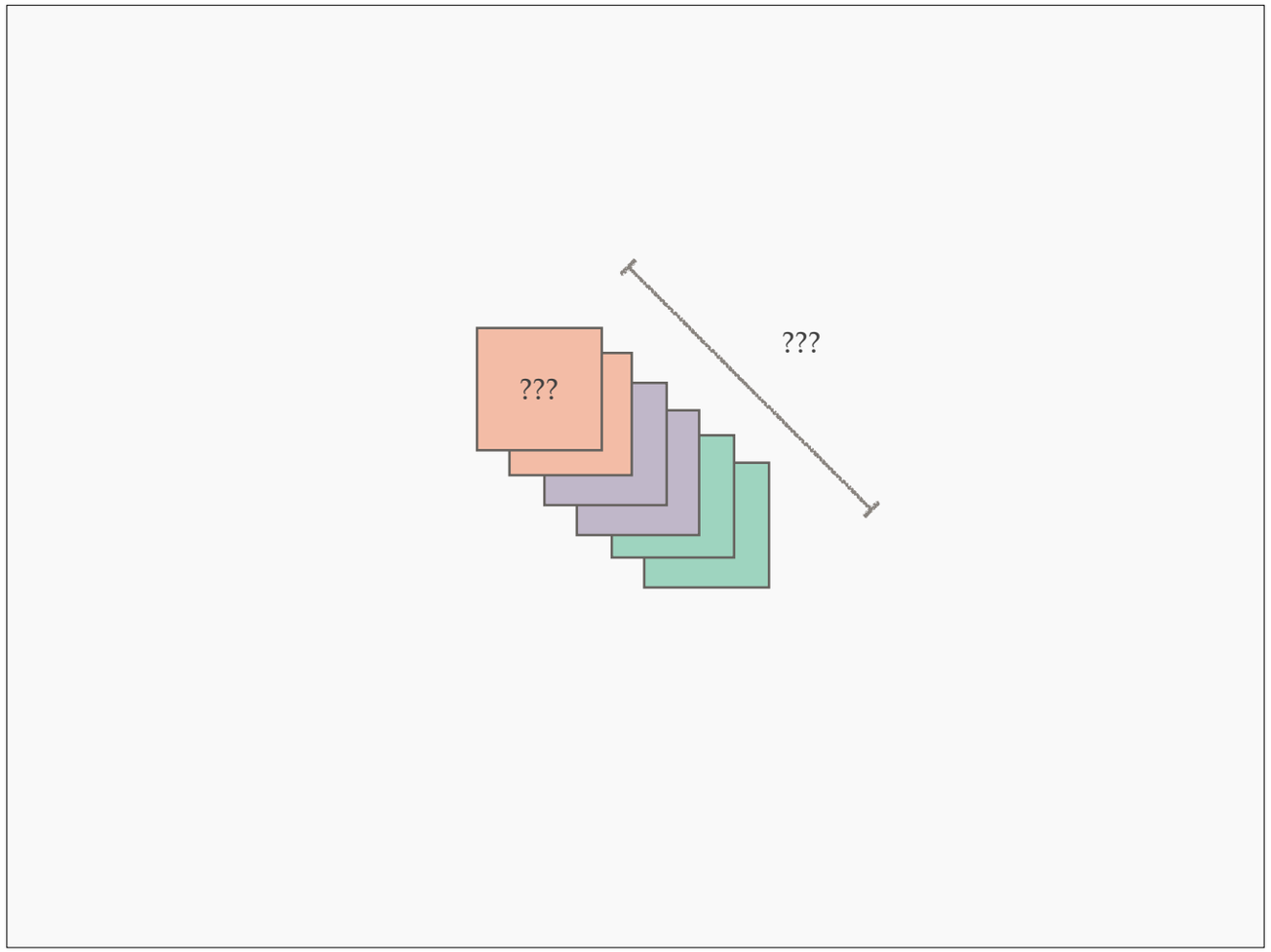


one option

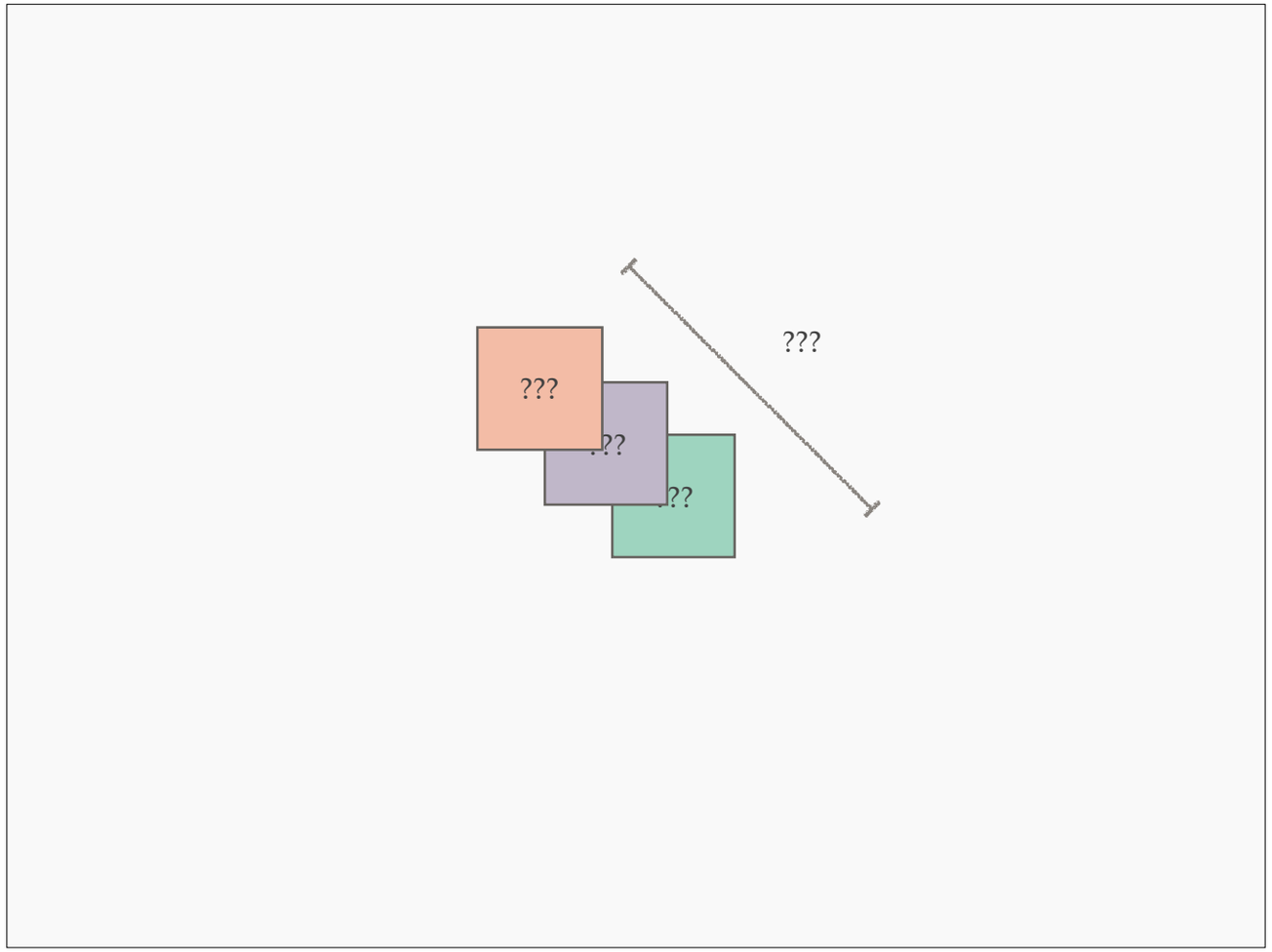








meaningful grouping



remove layers w/ preserve grouping?

?????

clearly a lot of decisions to make

advice

Press release first

PM saying:

—— next ——

"library allows developers to compose queries programmatically"

"Tired of repetitive CRUD app code? No more!!!"

"machine learning in 3 lines of code!"

"Data munging? No Problem!" - tears of joy

dask.delayed

PM saying:

—— next ——

"library allows developers to compose queries programmatically"

"Tired of repetitive CRUD app code? No more!!!"

"machine learning in 3 lines of code!"

"Data munging? No Problem!" - tears of joy

dask.delayed

PM saying:

—— next ——

"library allows developers to compose queries programmatically"

"Tired of repetitive CRUD app code? No more!!!"

"machine learning in 3 lines of code!"

"Data munging? No Problem!" - tears of joy

dask.delayed

PM saying:

—— next ——

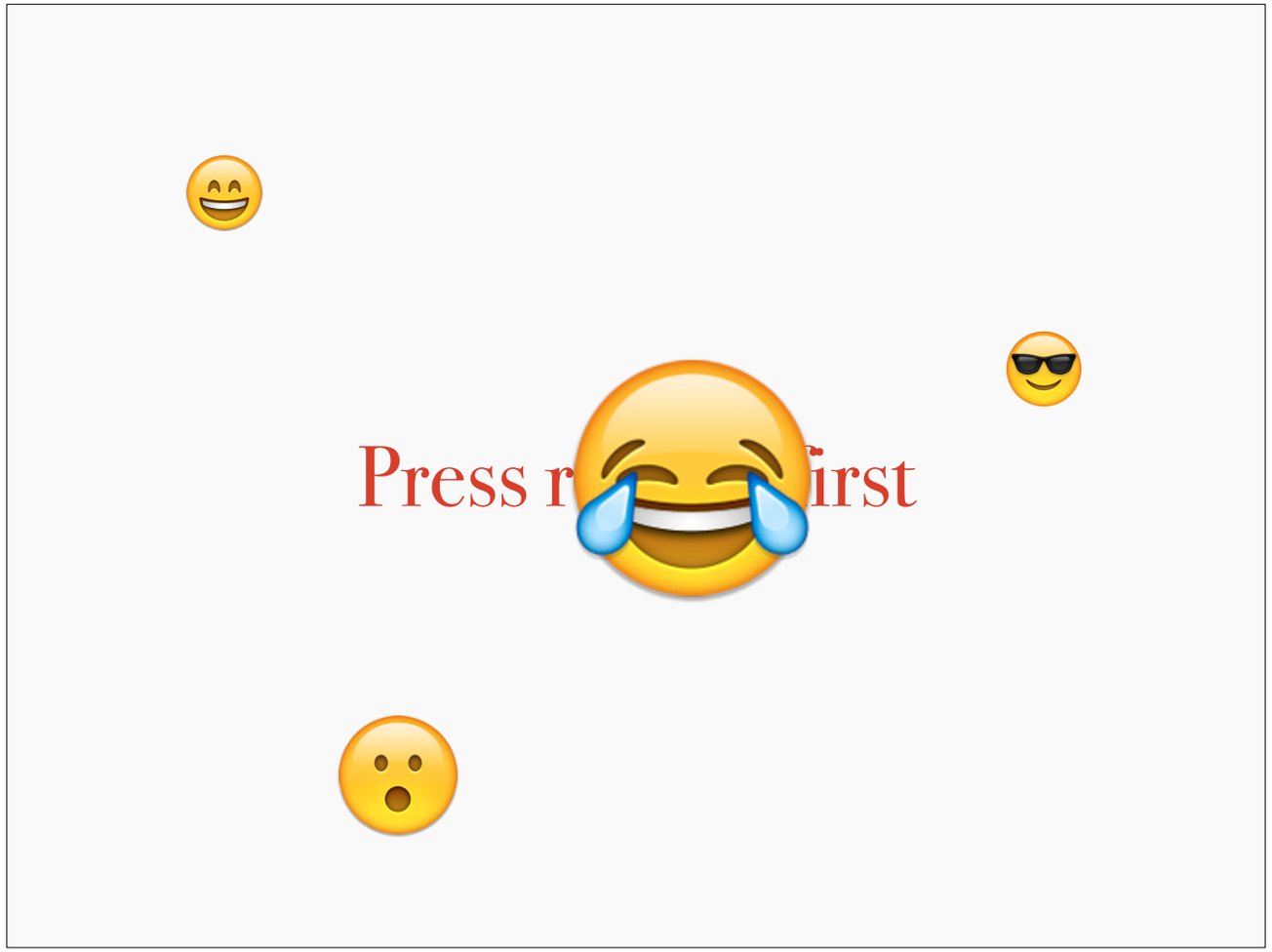
"library allows developers to compose queries programmatically"

"Tired of repetitive CRUD app code? No more!!!"

"machine learning in 3 lines of code!"

"Data munging? No Problem!" - tears of joy

dask.delayed



PM saying:

—— next ——

"library allows developers to compose queries programmatically"

"Tired of repetitive CRUD app code? No more!!!"

"machine learning in 3 lines of code!"

"Data munging? No Problem!" - tears of joy

dask.delayed

"Imaginary Code" second

play pretend

write code using your imaginary library

forces you to be specific

~~ contract-first development

Rewrite usage examples with existing libraries

how have other people tackled similar problems?

Is there an improvement?

What does it cost me?

pitfalls: cohesion, coupling

surface area

potential cost of changing / un-doing this abstraction?

How likely is this to change?

generalizations for events that won't happen

Do you really need an n-dimensional chess-board class just to make a chess game?

How does this abstraction benefit the user?

UX

reduce cognitive load and complexity

make code shorter?

Don′t
Repeat
Yourself?

explanation …

feels good

Don't
Refactor
Yet!

https://news.ycombinator.com/item?id=12528181

counterpoint

adds layers

(I didn't make this up)

"Prefer duplication over the wrong abstraction"

–Sandi Metz

https://www.sandimetz.com/blog/2016/1/20/the-wrong-abstraction

is it really that bad? sometimes yes

worth thinking about

Incremental architecture

Finally, given all these tips
what's the hurry?

Good architecture and abstraction decisions follow from domain knowledge.

n

More time on project
More domain knowledge

n

ergo …

Earlier on in the project you have less domain knowledge

n

Build less structure up front

find out that you were off about what you're building

don't fix problems before you have them

add incrementally

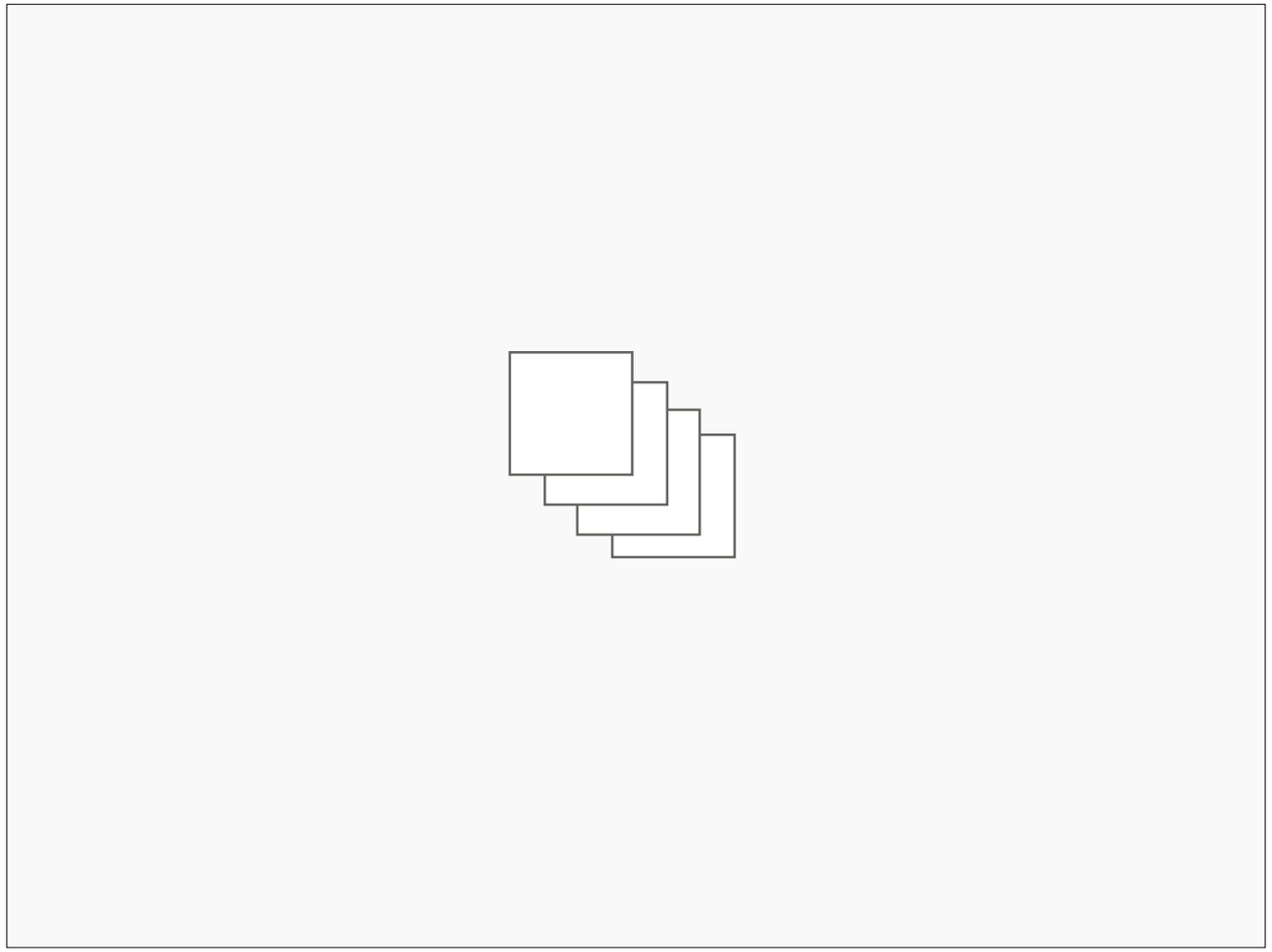
avoid "second system effect" Fred Brooks

TRICKS OF THE TRADE

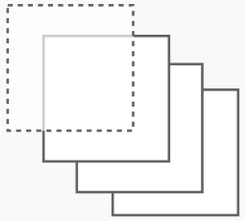
n

Trick:
Abstraction need not mean
building a wall.

just a hood, not a wall



n



solid / see-through

Flask

```
from flask import app

@app.route('/dessert')
def yum():
    return "donuts!"
```

explain!

decompose

Werkzeug under the hood

Flask

```
from flask import app

@app.route('/dessert')
def yum():
    return "donuts!"
```

explain!

decompose

Werkzeug under the hood

Flask

```
from flask import app


def yum():
    return "donuts!"

app.add_url_rule('/', 'dessert', dessert)
```

Flask

```
from flask import app


def yum():
    return "donuts!"

app.add_url_rule('/', 'dessert', dessert)
```

Flask

```
def add_url_rule(self, rule, **options, ...):
    # ...
    rule = self.url_rule_class(rule, **options, ...)
    # ...
    self.url_map.add(rule)
    # ...
```

https://github.com/pallets/flask/blob/501f0431259a30569a5e62bcce68d102fc3ef993/flask/app.py#L1071

If I look at url_map - werkzeug Map object!

Multiple access points “under the hood” - no wall

requests / urllib3 obj

Flask

```
def add_url_rule(self, rule, **options, ...):
    # ...
    rule = self.url_rule_class(rule, **options, ...)
    # ...
    self.url_map.add(rule)
    # ...
```

https://github.com/pallets/flask/blob/501f0431259a30569a5e62bcce68d102fc3ef993/flask/app.py#L1071

If I look at url_map - werkzeug Map object!

Multiple access points “under the hood” - no wall

requests / urllib3 obj

Flask

```
def add_url_rule(self, rule, **options, ...):
    # ...
    rule = self.url_rule_class(rule, **options, ...)
    # ...
    self.url_map.add(rule)
    # ...
```

https://github.com/pallets/flask/blob/501f0431259a30569a5e62bcce68d102fc3ef993/flask/app.py#L1071

If I look at url_map - werkzeug Map object!

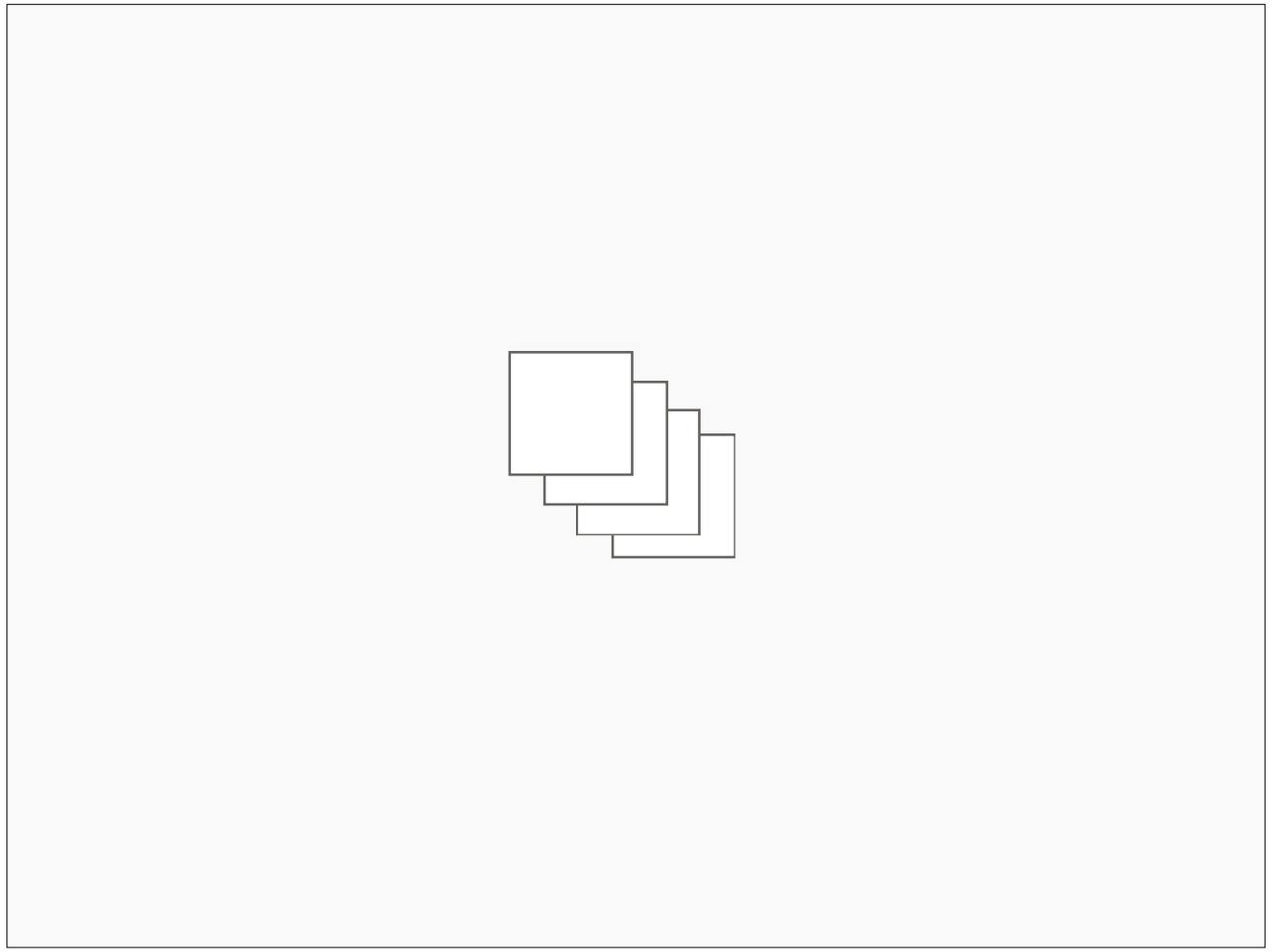
Multiple access points “under the hood” - no wall

requests / urllib3 obj

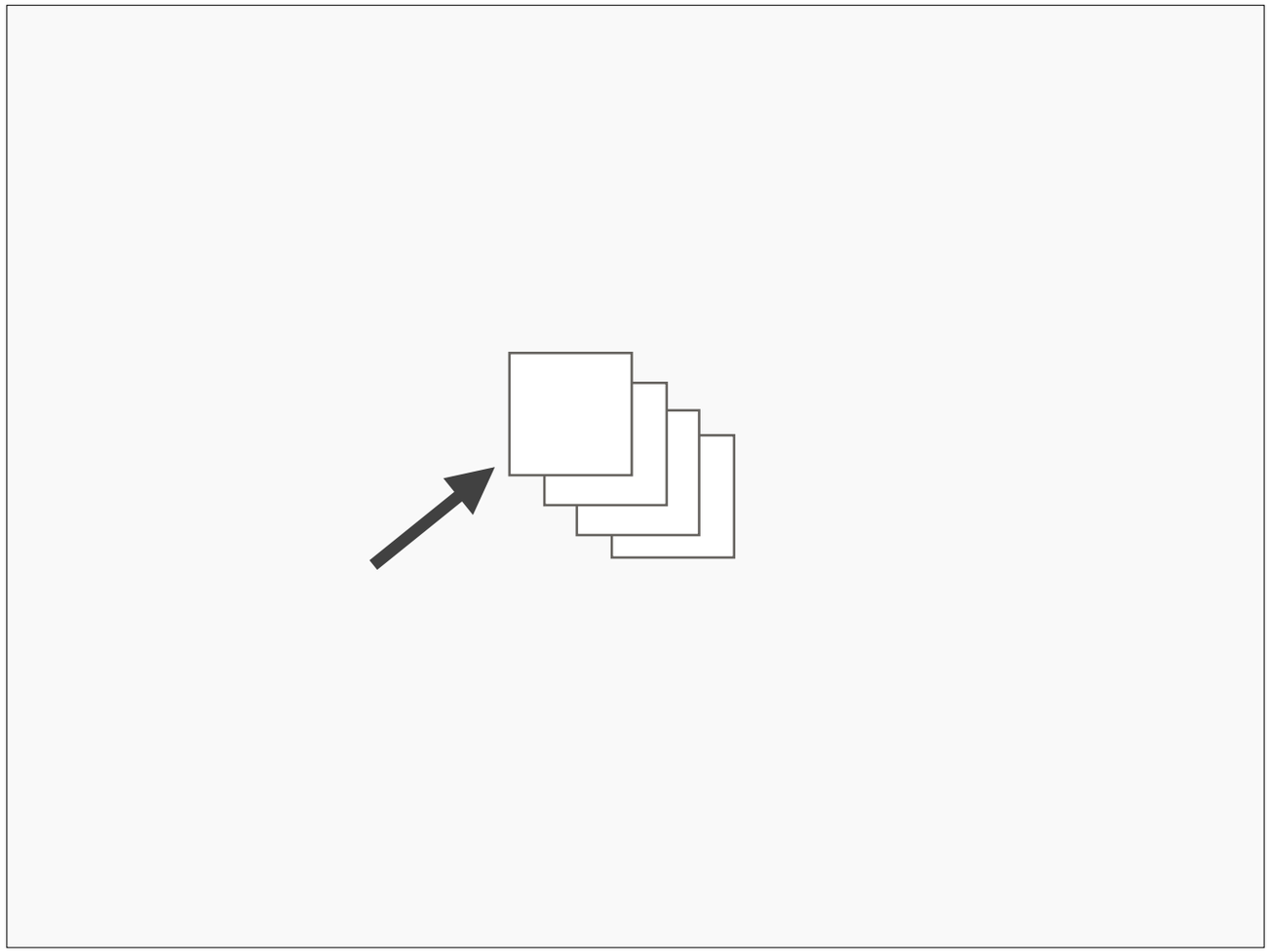
Trick:

More layers can make things cleaner.

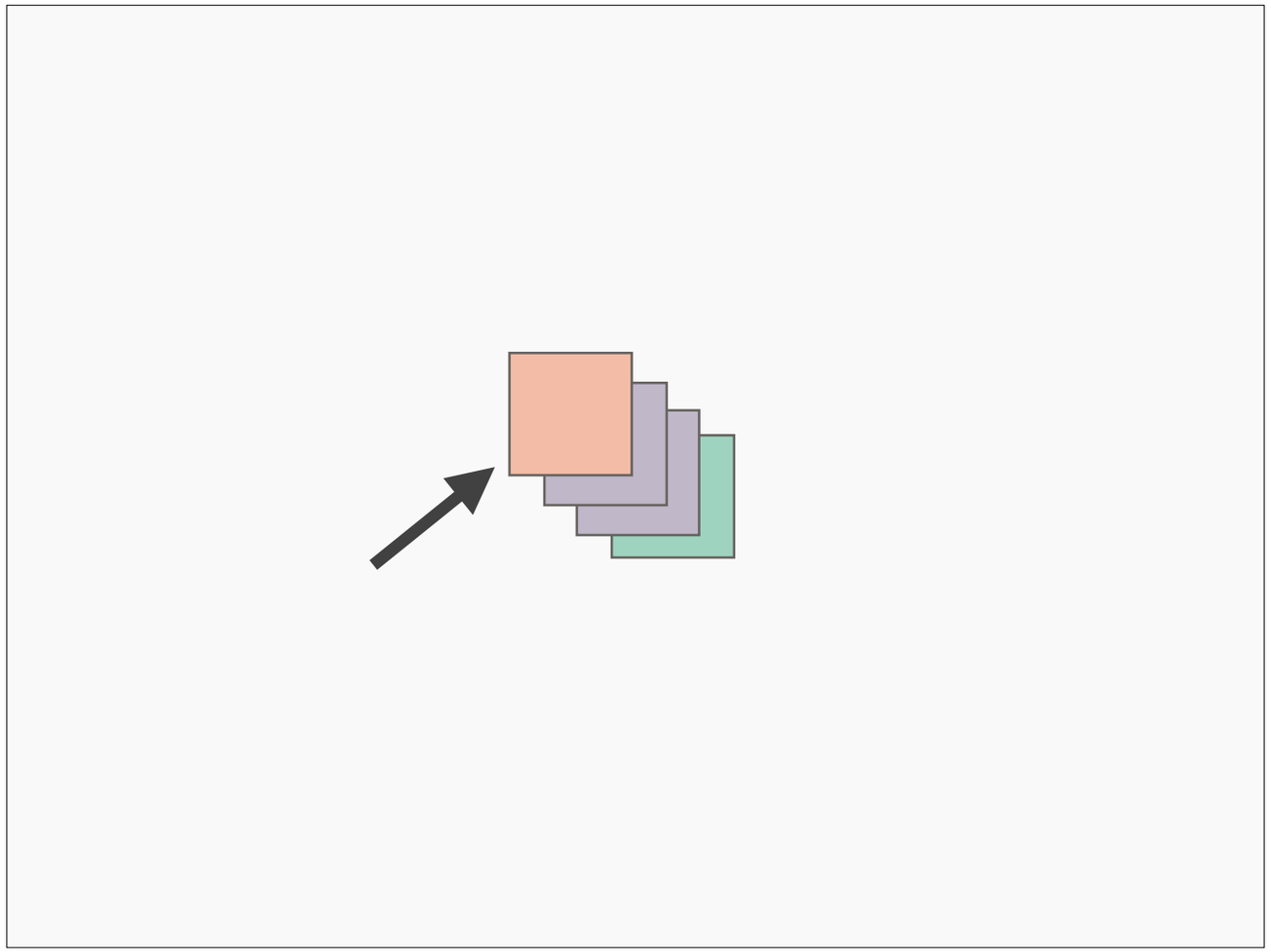
not always true / great effect



which layers should the user know about?

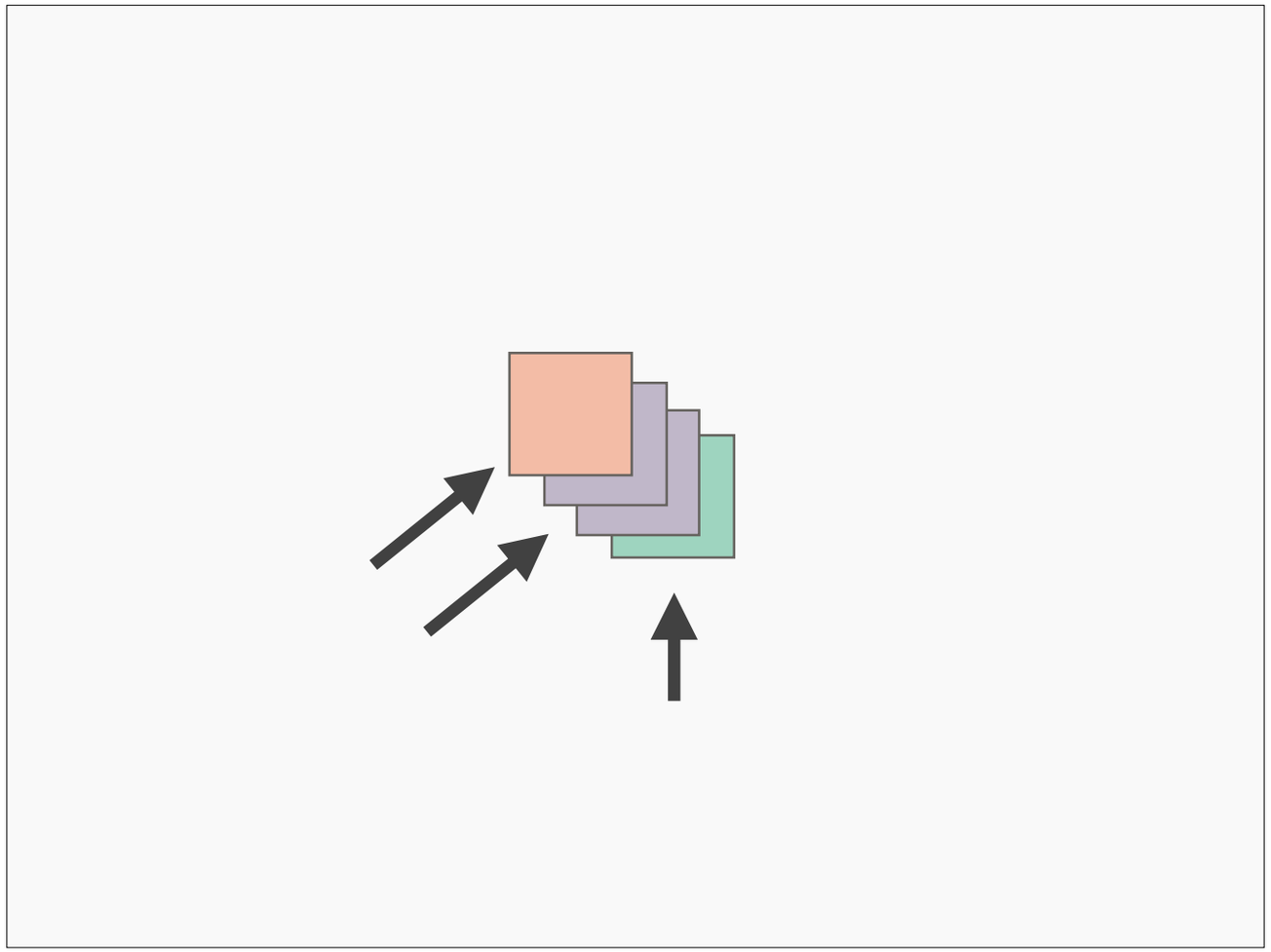


traditionally



meaningful groupings

potentially familiar?



expose each individually

Bokeh Charts

Bokeh Glyphs

Bokeh JS

interactive online visualization

create exact same chart in all 3

different levels of configurability and effort

expose all 3!

SQLAlchemy ORM
SQLAlchemy Core

Seaborn

Matplotlib

different creators

seaborn hi, MPL lo

CONCLUSION

conclude with this thought - different authors chose different levels

Claim:
The right level of abstraction
is audience-specific.

(pause)

Requests vs urllib2

```python
import requests

url = 'https://api.github.com/user'
auth = ('username', 'password')

r = requests.get(url, auth=auth)
print r.content
```

```python
import urllib2

gh_url = 'https://api.github.com/
user'

req = urllib2.Request(gh_url)

password_manager =
urllib2.HTTPPasswordMgrWithDefaultRea
lm()
password_manager.add_password(None,
gh_url, 'user', 'pass')

auth_manager =
urllib2.HTTPBasicAuthHandler(password
_manager)
opener =
urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.read()
```

is Urllib2 under-abstracted?

audience-specific

urllib “wrong” level for you, right level for library devs

http://fishsnack.deviantart.com/art/Creation-of-Adam-in-the-21st-Century-280501206

Maybe it's ok!

Some APIs are closer to the machine

Some APIs are closer to the human

We need both

Theory:
"For Humans" adds a layer we didn't know we were missing.

(pause)

we were sorely lacking it

Abstraction isn′t a goal,
It′s a tool.

can be used for good or bad

it's everywhere

it's a tool we inadvertently use

hopefully you start seeing it everywhere / make UX better

Hearing from you makes me happy!

(@makmanalp)

slides on my twitter

Classes

```
select *
from users
where name = "mali";
```

classes

any user?

Classes

```
"""
select *
from users
where name = '{}';
""".format("mali")
```

now we need to add another parameter

Classes

```
"""
select *
from users
where name = '{}'
and   dessert = '{}';
""".format("mali", "pie")
```

ugh

Classes

```
User.query\
    .filter_by(name="mali")\
    .all()
```

Classes

```
User.query\
    .filter_by(
        name=“mali”,
        dessert="cake")\
    .all()
```

.query returns a Query object.

variables! arbitrary dictionary!

Classes

```
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer,primary_key=True)
    name = Column(String(50))
    dessert = Column(String(50))
```

classes make state explicit and organized

associate and group state and behavior

hiding details makes things easy to use??

Read a lot of code!

Start with a library you use all the time

Start with a function you know

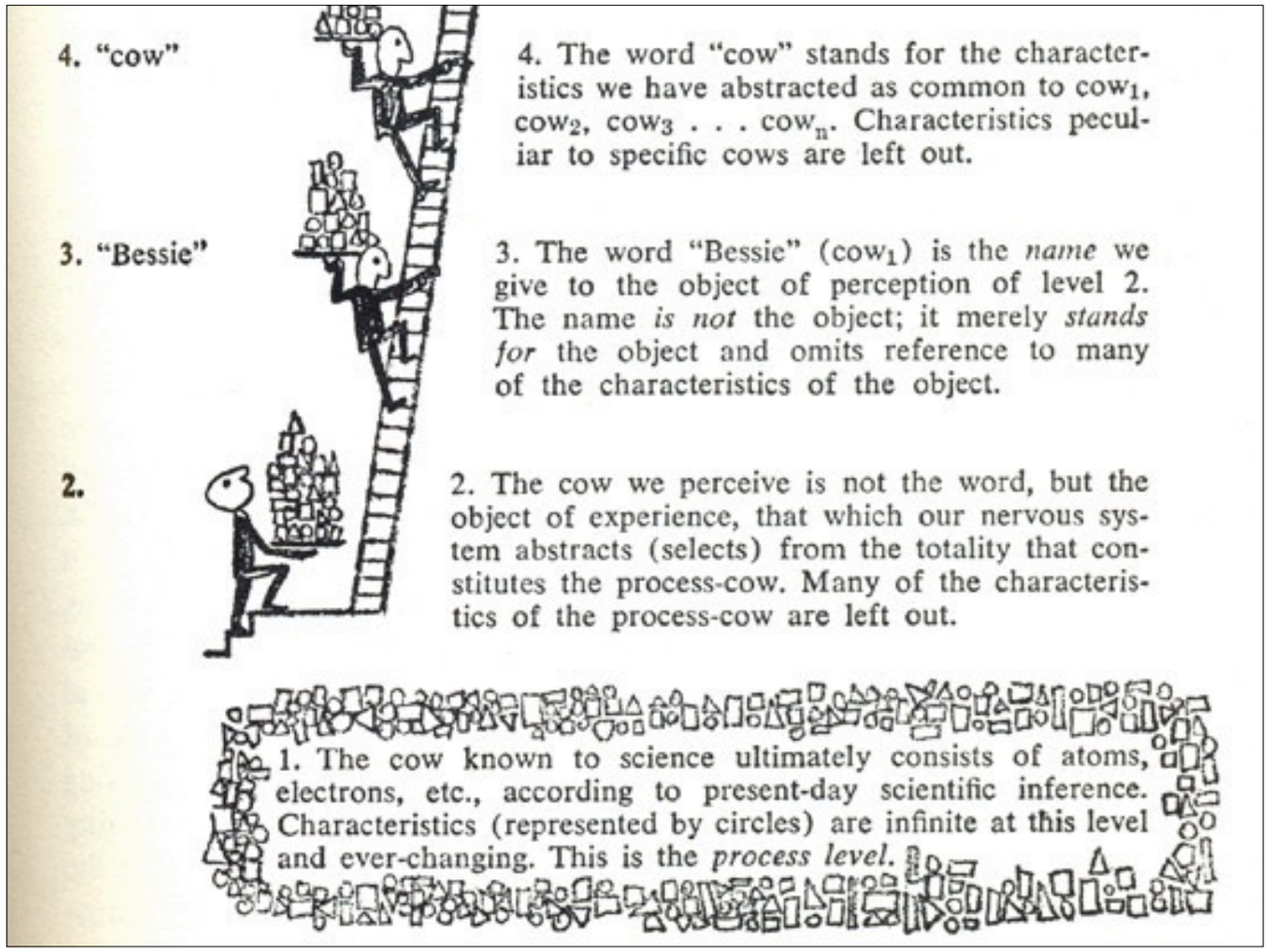
Find the code and read!

Ask for help!

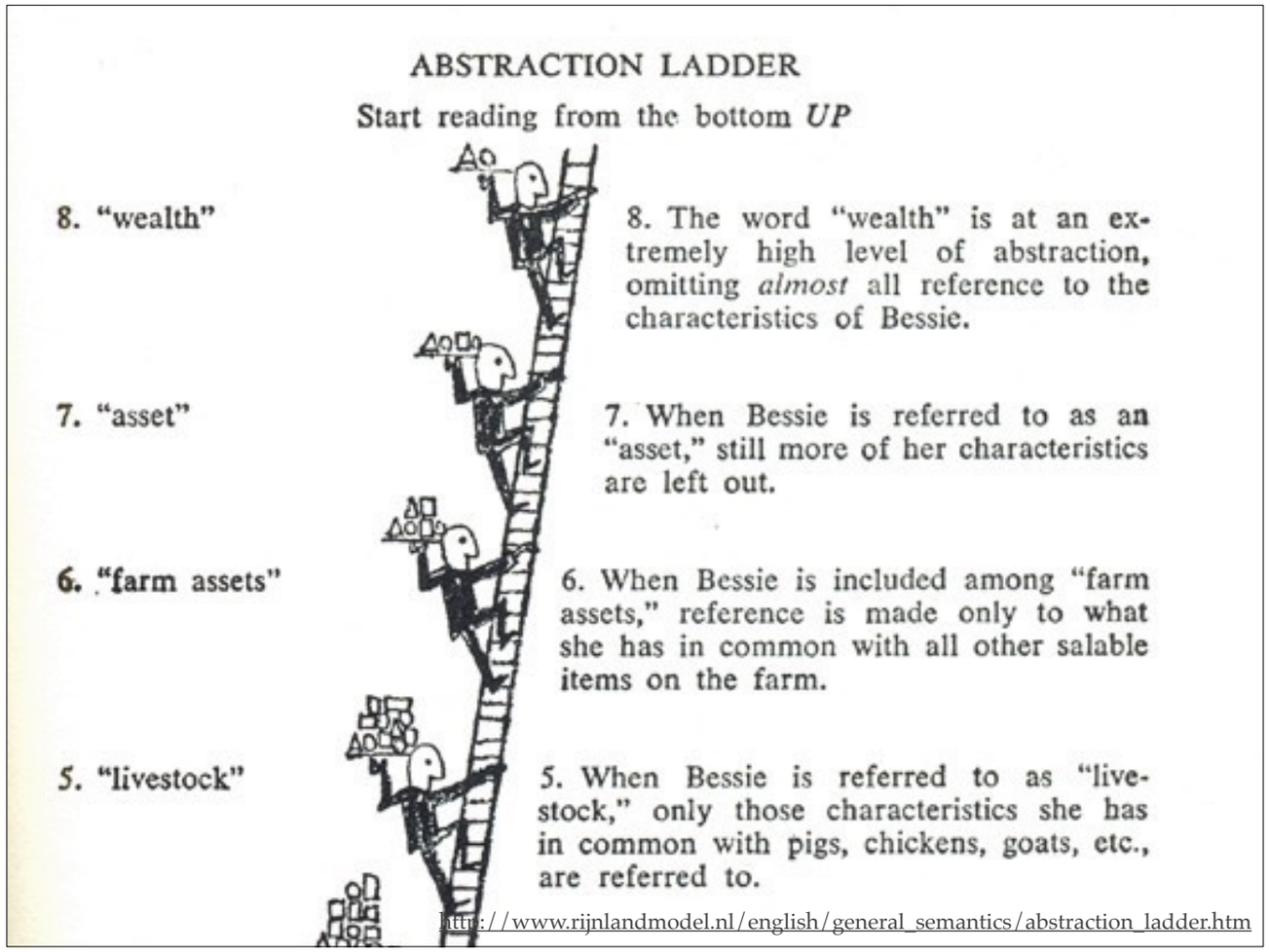
Hiding details

papers over grossness???

Cory B speaking about the guts of requests https://us.pycon.org/2017/schedule/presentation/71/



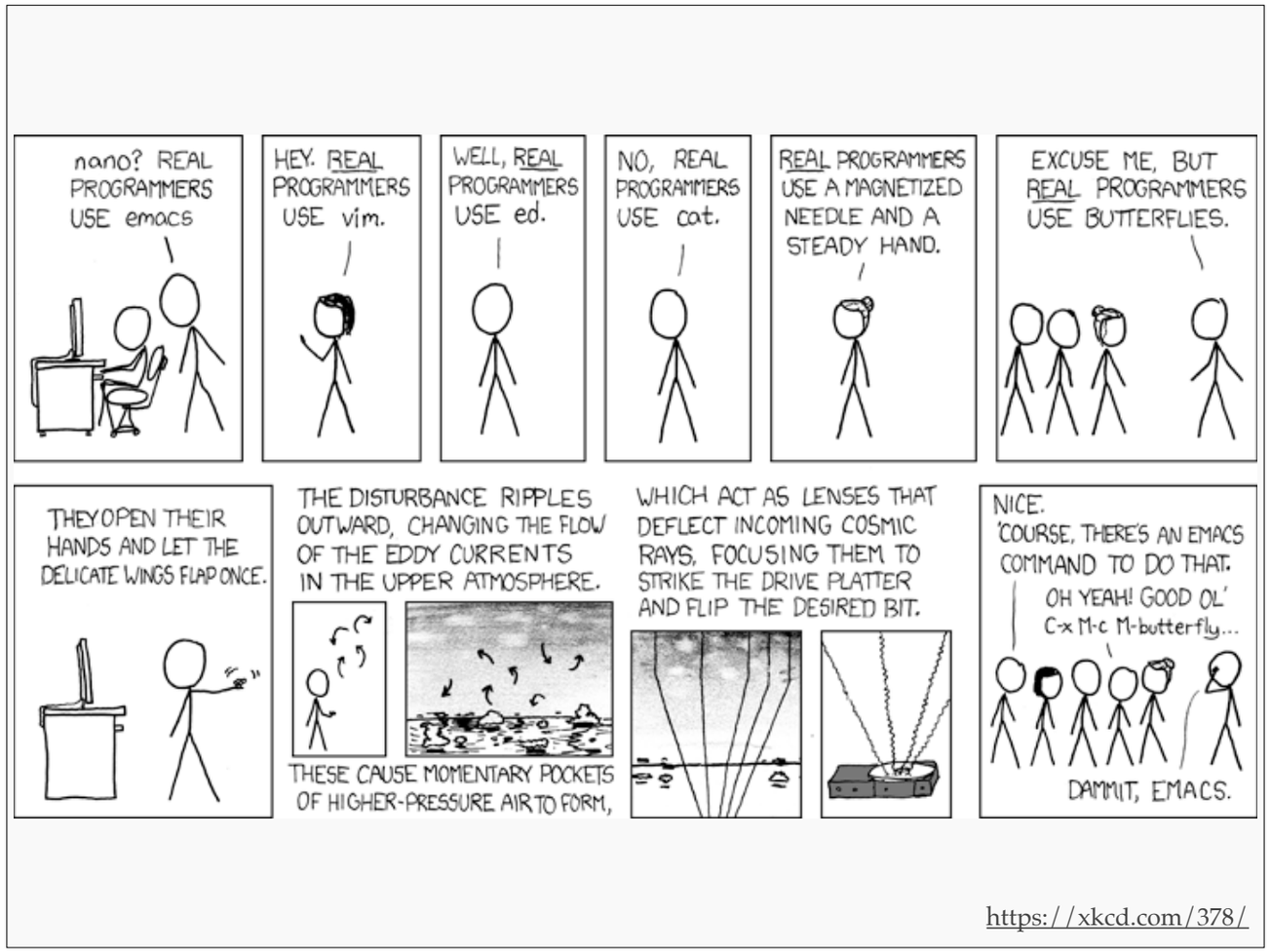
http://aminotes.tumblr.com/post/3686652856/the-process-of-abstracting-according-to-s-i


ABSTRACTION LADDER
Start reading from the bottom UP
8. "wealth"
8. The word "wealth" is at an extremely high level of abstraction, omitting almost all reference to the characteristics of Bessie.
7. "asset"
7. When Bessie is referred to as an "asset," still more of her characteristics are left out.
6. "farm assets"
6. When Bessie is included among "farm assets," reference is made only to what she has in common with all other salable items on the farm.
5. "livestock"
5. When Bessie is referred to as "livestock," only those characteristics she has in common with pigs, chickens, goats, etc., are referred to.
http://www.rijnlandmodel.nl/english/general_semantics/abstraction_ladder.htm

Hayakawa

PITFALLS

n



make fun of “real programmer”

Butterfly vs Magnet needle vs File Systems

Files aren’t “real”

it’s all zeroes and ones! to real apps