

# Documentation is King

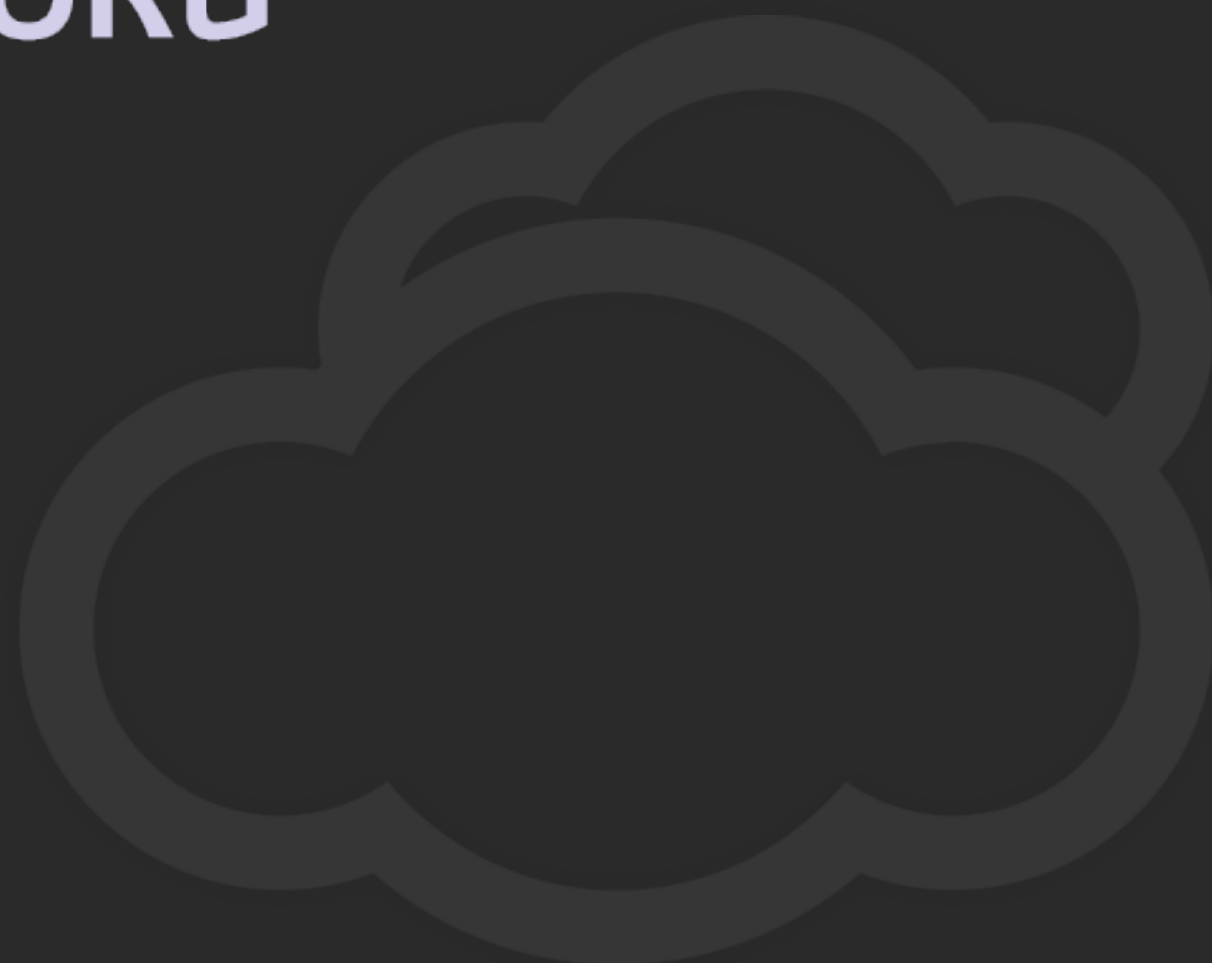


*Kenneth Reitz*

Hi.

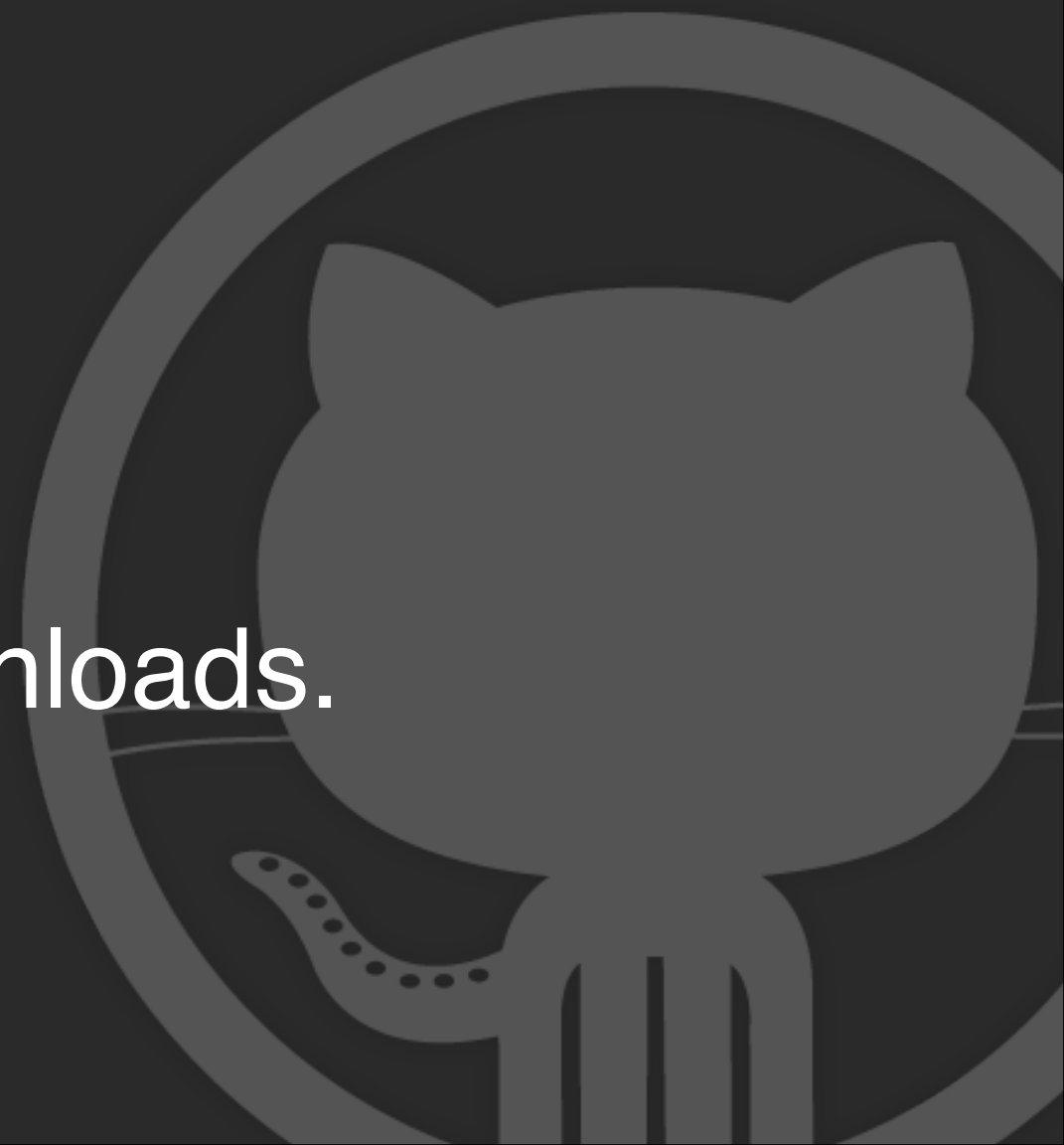
@kennethreitz





# github.com/**kennethreitz**

- ~18 serious projects.
- 100+ experiments.
- OSX-GCC-Installer: 56TB of downloads.
- Requests: 28.8+ million downloads.



# Other Interests...

- Street Photography
- Synthesizers and Music Production
- World Travel (~150,000 miles last year)
- Public Speaker (29 events last year)
- Classic Video Games!

# The Best Things

Prime Lenses, Monophonic  
Synths, Handheld Games...

Pen and paper.

Mechanical watch.

A single carry-on.

CONSTRA

PROPS

CREATI

VITY



# The Simple Things

Simple → Prime Lenses, Monophonic  
Synths, Handheld Games... Simple!

Simple Pen and paper Simple!

Simple → Mechanical watch.

A single carry-on. Simple!

pra•gmat•ic |prag'matɪk|, *adj*:

Dealing with things sensibly and realistically in a way that is based on practical rather than theoretical considerations

# Requests

## HTTP for Humans

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf-8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type": "User" ...}'
>>> r.json
{u'private_gists': 419, u'total_private_repos': 77, ...}
```



The API is all that matters.

Everything else is secondary.

People are going to be  
spending two or three hours a  
day  
with these machines — more  
than they spend with a car.

— Steve Jobs, 1983

Software design must be given  
at least as much consideration  
as we give automobiles today  
— if not a lot more.

— Steve Jobs, 1983

That worked for  
Apple.

Developers spend 8+ hours a day with APIs.

Why are they treated differently?



# Requests Success

- Python is a language built for Humans.
- Why should HTTP be non-trivial?
- I explored and discovered what I really needed, and built it.
- I had a real problem that I solved for myself.

# Requests Success

- At first, Requests was far from powerful.
- But, it deeply resonated with people.
- Features grew over time, but the API was never compromised.

Developers spend 8+ hours a day with APIs.

Build for yourself—a developer.

How?

Write the Docs.

- Before any code is written, write the README — show some examples.
- Write some code with the theoretical code that you've documented.

# Paradigm Shift

- Instead of engineering something to get the job done, you interact with the problem itself and build an interface that reacts to it.
- You discover it. You respond to it.

# Sculptures, Etc.

- Great sculptures aren't engineered or manufactured—they're discovered.
- The sculptor studies and listens to the marble. He identifies with it.
- Then, he responds.
- Setting free something hidden



# Responsive Design

- It's not about a design that will “work” on a phone, tablet, and desktop.
- It's about making something that identifies itself enough to respond to the environment it's placed in.
- Free of arbitrary constraints.

Readme-Driven Development?  
Responsive API Design.

# Complex Code is Bad

- Tight coupling, monolithic codebases.
- Lurking, growing technical debt.
- Maintenance burden is high.
- Self-serving instead of problem-solving.

# Simple Code is Good

- Code solves problems created by humans.
- The less code, the less to maintain.
- Negative diffs are the best diffs.
- Small, sharp, distributed services.

Simplicity is always  
better than functionality.

— Pieter Hintjens

# What is Open Source?

- Transparent groups of distributed developers working together to make software and projects that make the world a better place.

# Open Source is Epic

- We have a unique opportunity to take part in a powerful social movement, creating the tools that are fundamentally changing the world around us.
- Social Media, Elections, Journalism, Wikileaks, etc...

Documentation is the glue that  
makes open source possible.



# Bad Open Source

(GitX, Facebook SDK, httpplib2, hubcap, oauth2, &c)

- Appears unmaintained (20+ pull requests).
- Fails to solve a clear problem.
- Has unclear expectations.

# Great Open Source

(Jenkins, Python, Django, Pip, Bundler, &c)

- Solves a clear problem.
- Communicates well with users.
- Manages expectations realistically.

# Great Open Source

(Jenkins, Python, Django, Pip, Bundler, &c)

- Solves a clear problem.
- Communicates well with users.
- Manages expectations realistically.

**Documentation!**



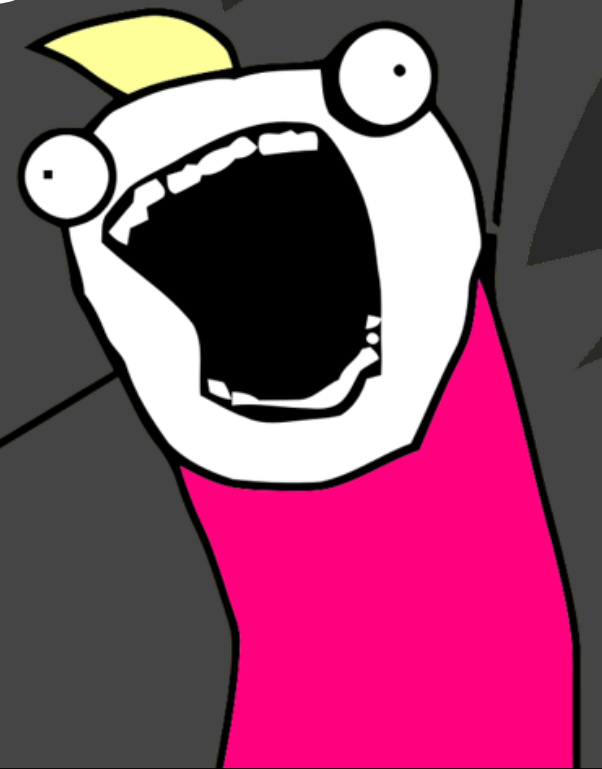
# Internal Codebase Patterns

- Components are tightly coupled.
- Broad tribal knowledge is required.
- Iterative change of components difficult.
- Technical debt has a tendency to spread.
- Little documentation (if any).

# Pretend it's Open Source

- Components become concise & decoupled.
- Concerns separate themselves.
- Best practices emerge (e.g. no creds in code).
- Documentation and tests become crucial.

Document  
All The Things!



# Documentation = Better Code

- Documentation is more important than tests.
- It changes the way we think about problems.
- Specifically, explaining concepts to users and fellow developers helps uncover asymmetry in APIs.

A decorative, symmetrical frame in a light gray color. It features ornate, swirling lines and small circular accents, resembling a stylized floral or scrollwork design. The frame is centered on the page and encloses the text.

Asymmetry?



# Python 2.5

Bytes        '42'

Unicode     u'42'

# Python 2.5

# Python 3.0

Bytes

'42'

b'42'

Unicode

u'42'

'42'

# Python 2.6

# Python 3.1

Bytes '42', b'42'

b'42'

Unicode u'42'

'42'

# Python 2.7

# Python 3.2

Bytes	'42', b'42'	b'42'
Unicode	u'42'	'42', ?

```
graph LR; subgraph Python_2_7 [Python 2.7]; B2["Bytes: '42', b'42'"]; U2["Unicode: u'42'"]; end; subgraph Python_3_2 [Python 3.2]; S3["str: '42'"]; Q3["?"]; end; B2 --> S3; U2 --> S3;
```

# Python 2.7

# Python 3.3

Bytes '42', b'42'

b'42'

Unicode u'42'

'42', u'42'

## Symmetrical.

Just as writing tests helps  
encourage composable code,  
writing documentation  
encourages consistent code.

# Documentation = Better Workplace

- Every design decision should be documented.
- Reduces process locks and sync points.
- Automates the onboarding process.
- Employees can hop from project to project.
- Deploy to production without worry.

Imagine never having to tap on a  
coworker's shoulder again.



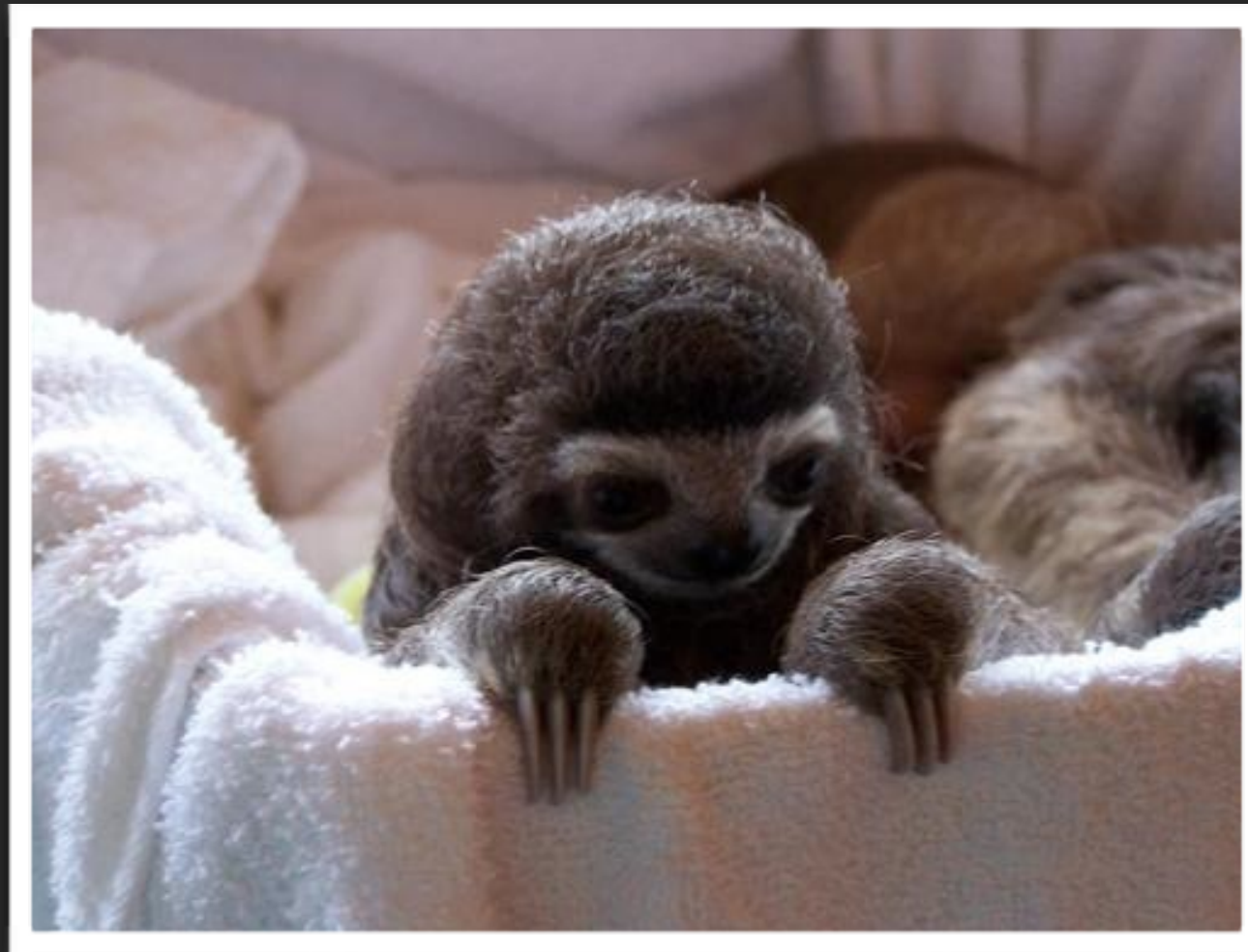
Imagine never getting interrupted  
by a coworker again.

# Documentation = Better Lifestyle

- Documentation enables asynchronous workflows.
- Increased autonomy leads to a happier life.
- Fewer interruptions, fewer misunderstandings.

Documentation makes  
the world a better place.

# Write the Docs



...or the sloth will find you.

Questions?

[github.com/kennethreitz](https://github.com/kennethreitz)

