

# HTTP Can Do That?!

A collection of bad ideas

by Sumana Harihareswara

@brainwane

Changeset Consulting

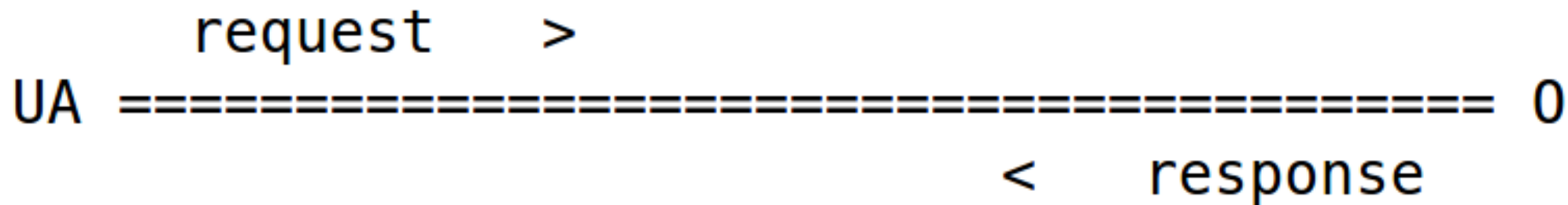
# HTTP

Hypertext

Transfer

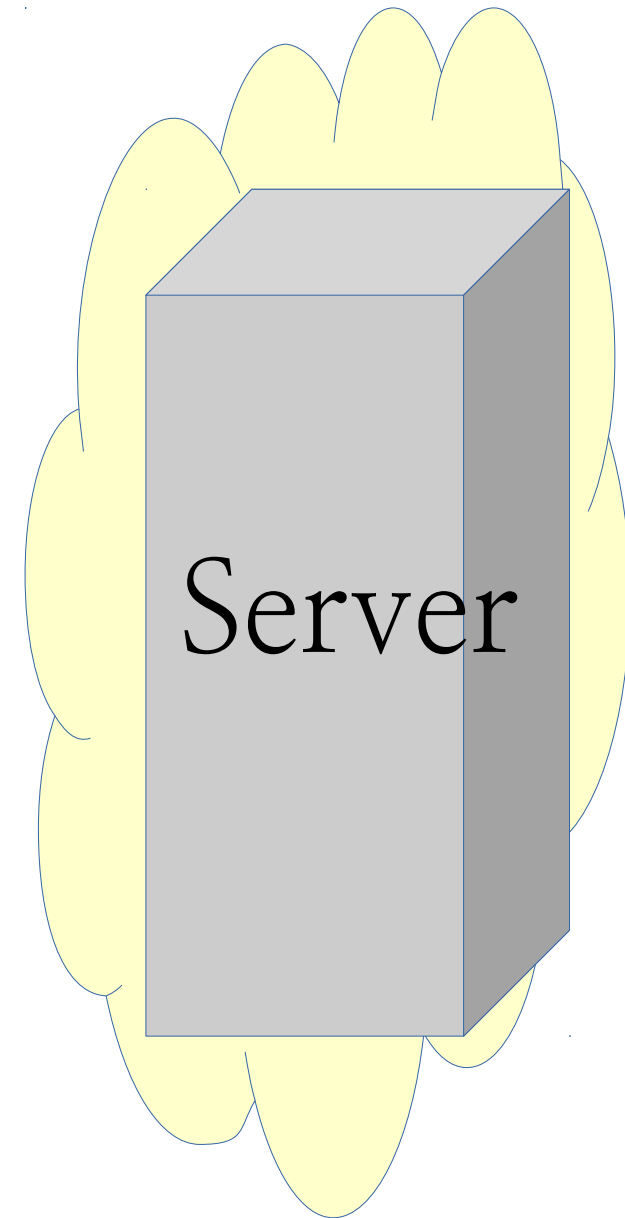
Protocol

# Diagrams!

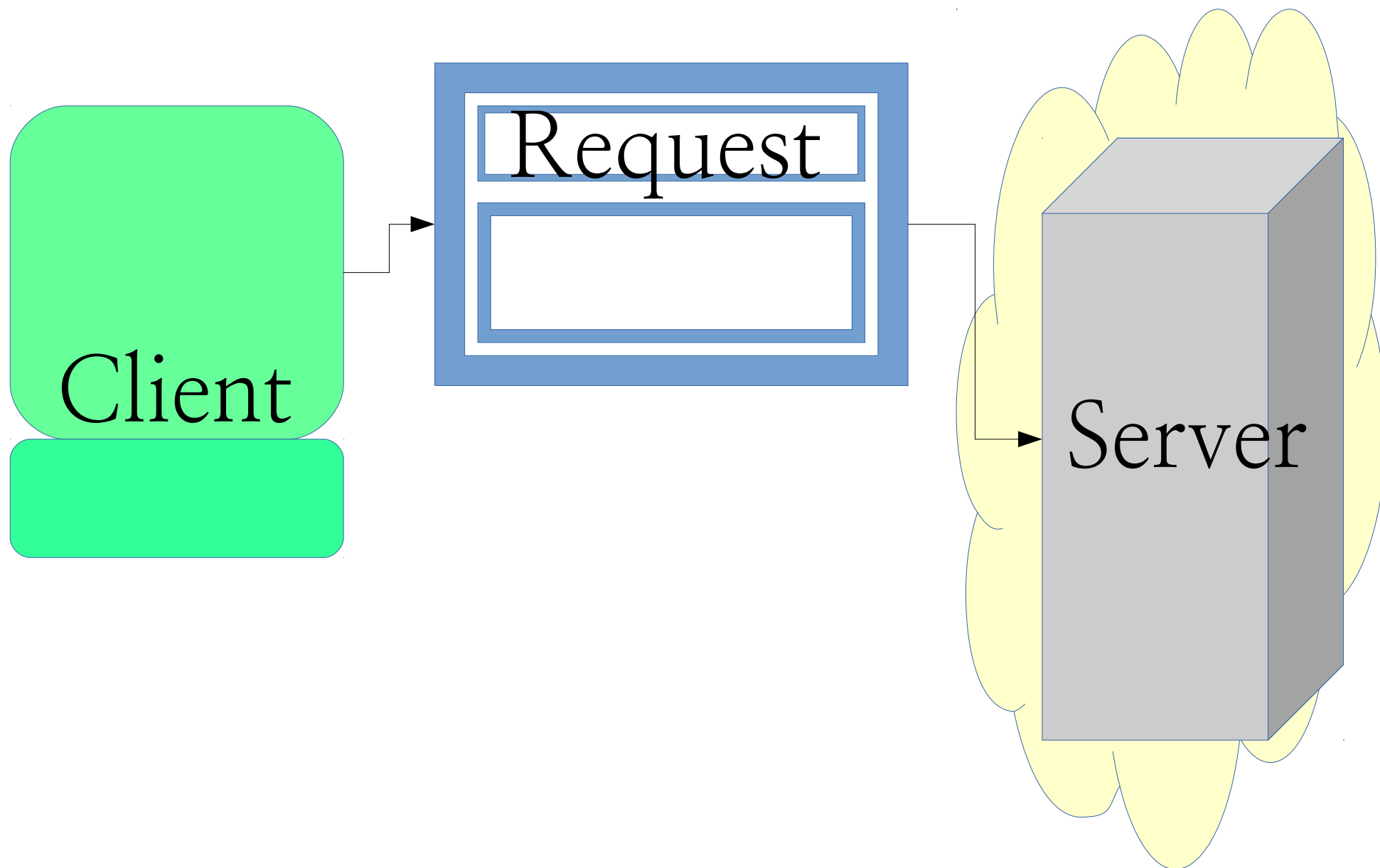


- Internet Engineering Task Force (IETF) RFC 7230  
Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

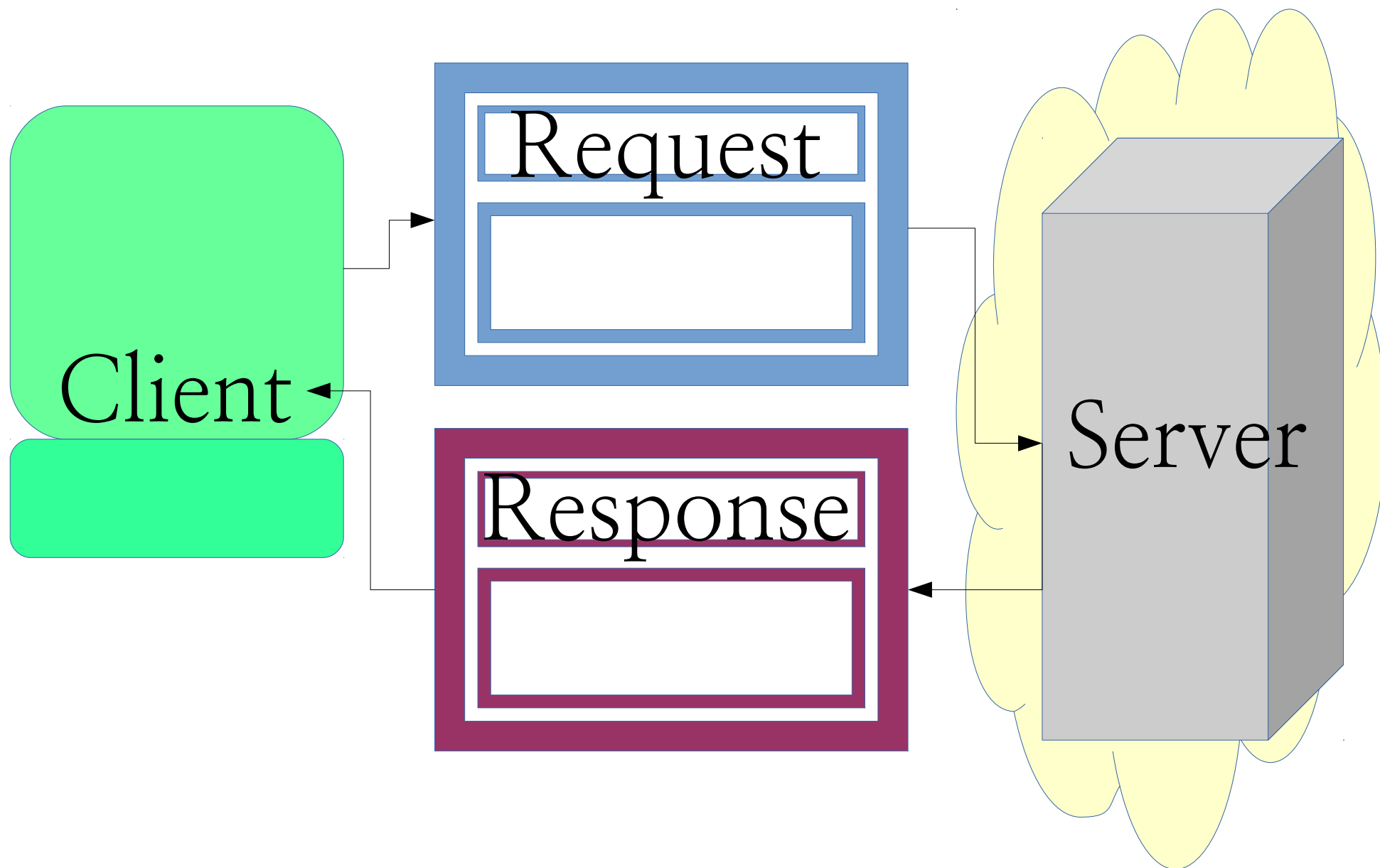
# HTTP: crash course



# HTTP: crash course



# HTTP: crash course



# An HTTP Message (Request or Response)

## START-LINE

HTTP version (1.1)

Request method (**GET**, **POST**)

Response status code (200, 404, 500)



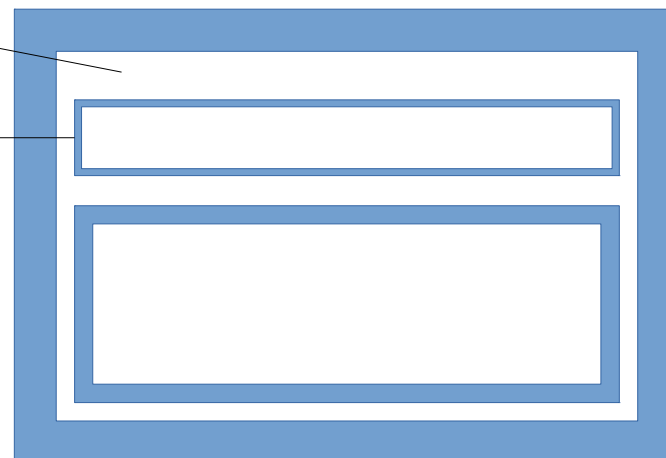
# An HTTP Message (Request or Response)

## START-LINE

HTTP version (1.1)  
Request method (`GET`, `POST`)  
Response status code (200, 404, 500)

## HEADERS

Content-Type  
Content-Length  
.....





# An HTTP Message (Request or Response)

## START-LINE

HTTP version (1.1)

Request method (`GET`, `POST`)

Response status code (200, 404, 500)

## HEADERS

Content-Type

Content-Length

.....

## BODY



# Example Request

START-LINE

GET / HTTP/1.1

HEADERS

Host: www.sumana.biz  
Accept: text/html  
User-Agent: ScraperBot

BODY



# Example Response

## START-LINE

HTTP/1.1 200 OK

## HEADERS

Content-Type: text/html

Content-Length: 203

Date: Tue, 16 Jun 2015 16:21:56  
GMT

Last-Modified: Tue, 16 Jun 2015  
13:27:14 GMT



## BODY

```
<html>
<head>
<title>Welcome to
Sumanaville</title>
</head>
<body><center><h1>Ro
ckin'</h1>
<p>This is a pretty
rockin' site. I'm
```

# Methods

# Popular request methods ("verbs")

- **GET**

gimme

- **POST**

here you go

# First bad idea: **POST** but not **GET**

```
class APIHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):  
    def do_POST(self):
```

more:

<https://gitlab.com/http-can-do-that/secureapi>

# POST but not GET: use cases

letters to Santa Claus

# POST but not GET: use cases

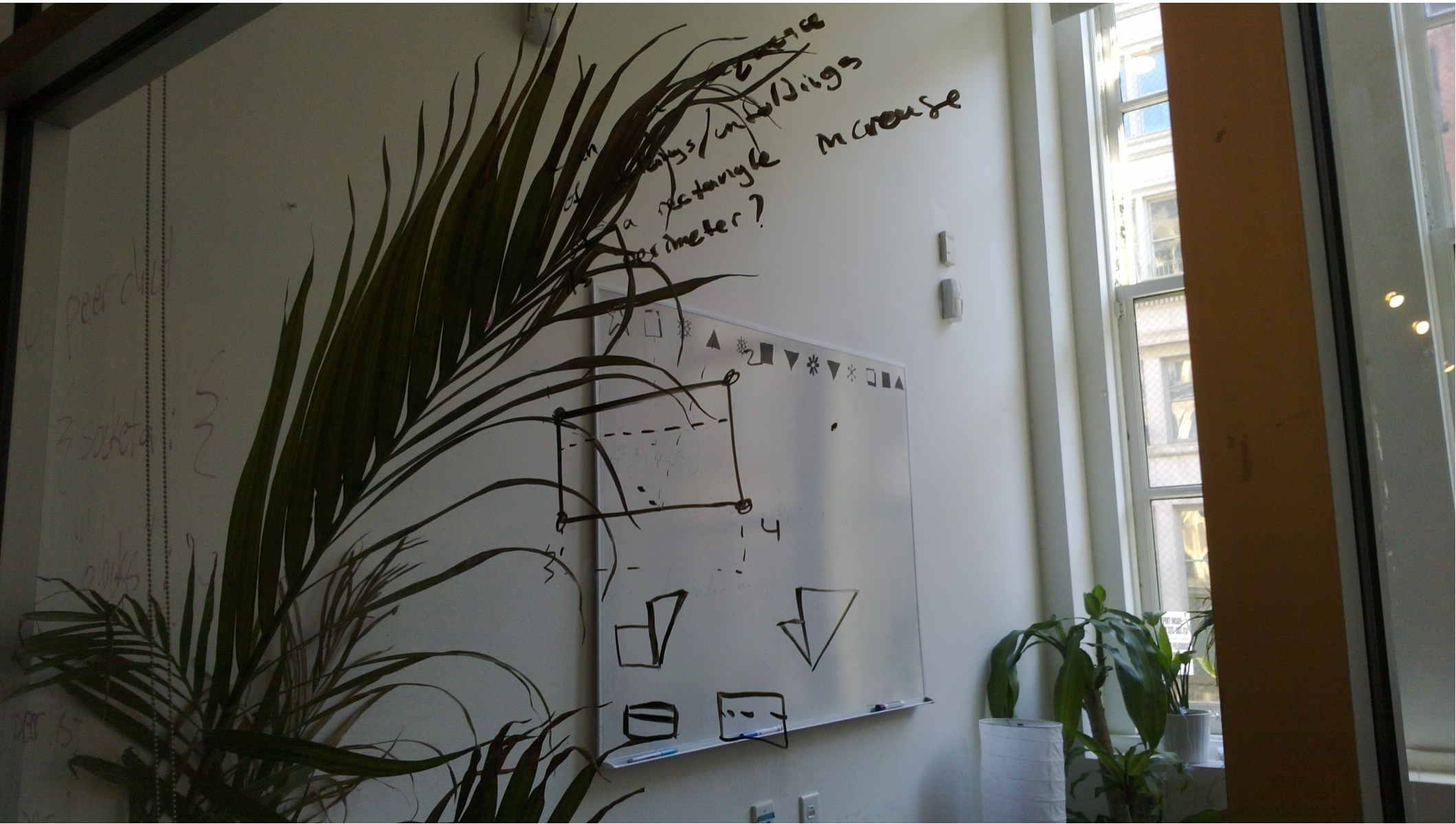
employee suggestion box



# POST but not GET: use cases

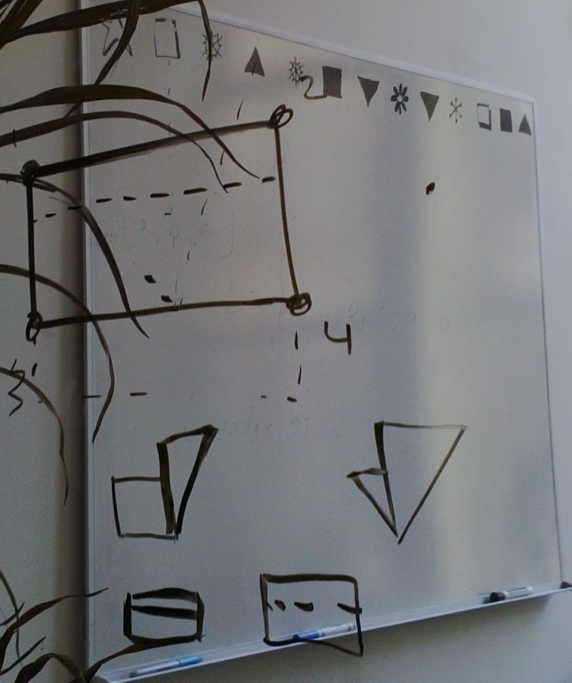
extremely moderated blog comments

(a logistical note)



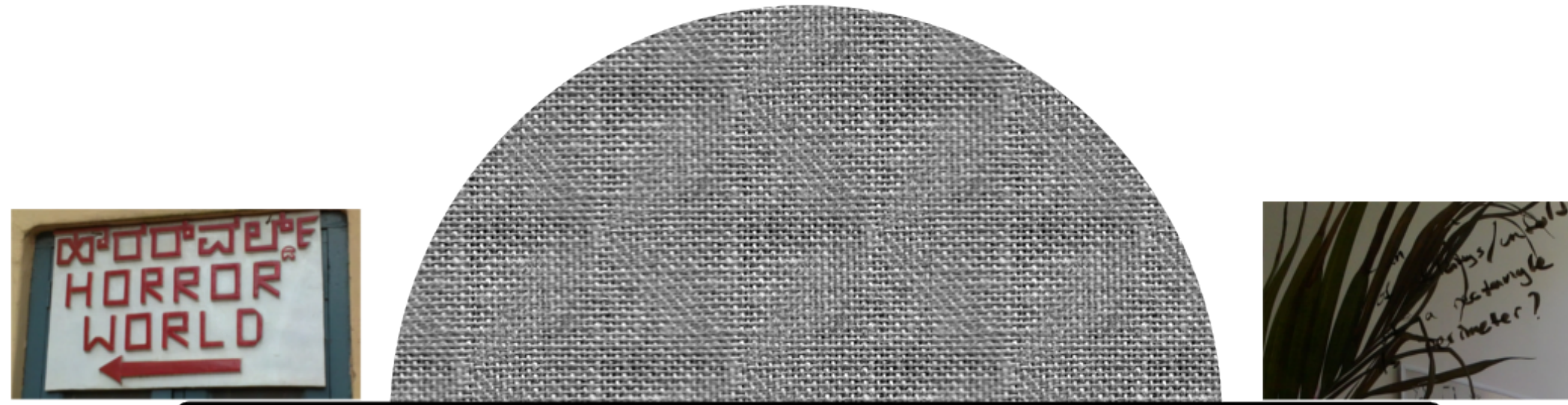
peer dist  
3-5000  
2015  
2015

of m... in buildings  
a rectangle  
diameter?  
increase

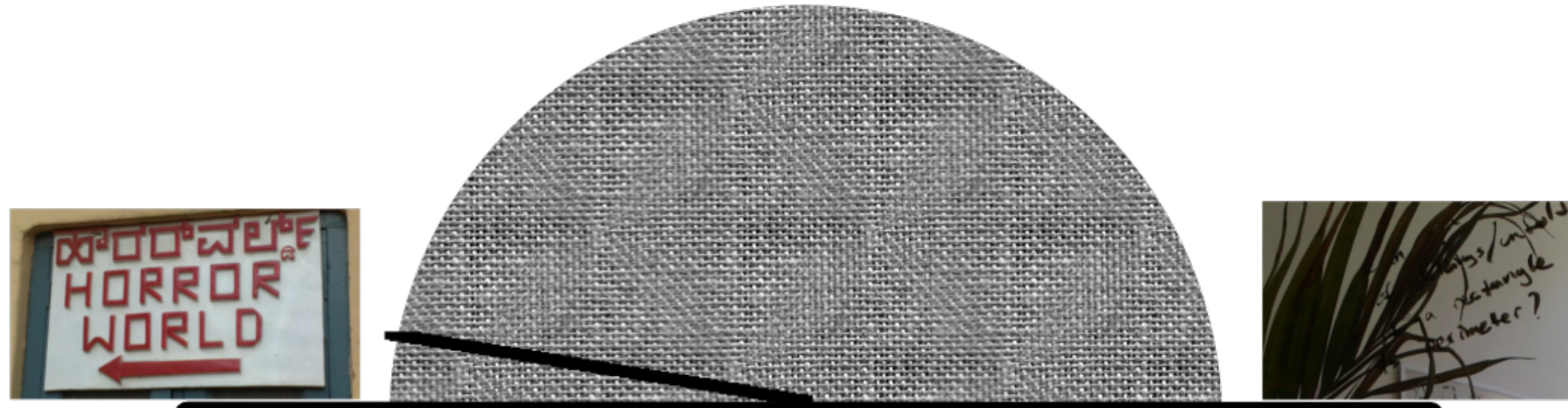




# Bad Idea Scale



Giving client no way to **GET** –  
bad idea



# Remember “CRUD”?

Create

Read

Update

Delete

# Remember “CRUD”?

Create

**POST**

Read

**GET**

Update

**POST**

Delete

**POST**



# Remember “CRUD”?

Create

**POST**

Read

**GET**

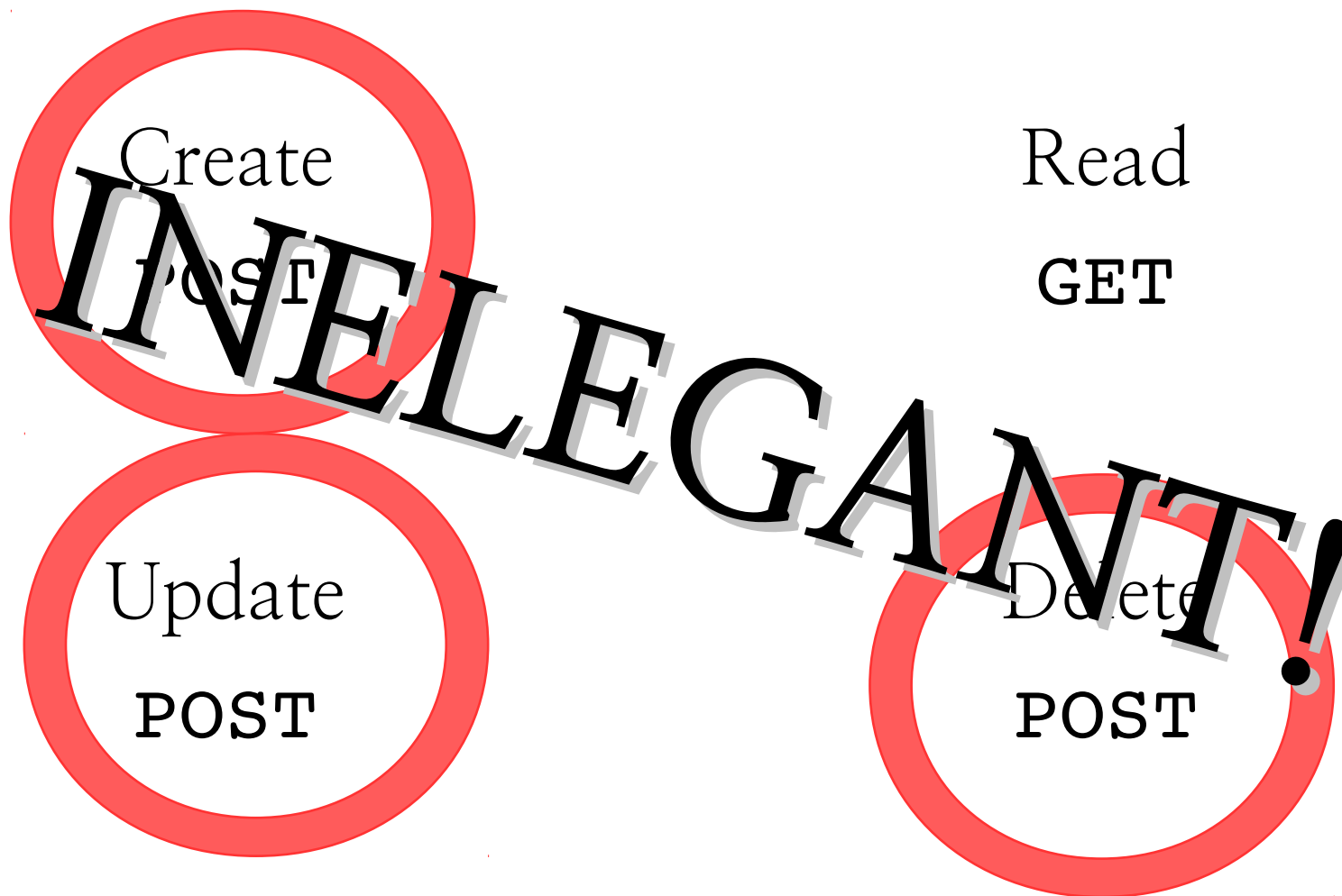
Update

**POST**

Delete

**POST**

# Remember “CRUD”?



# Underappreciated methods

**DELETE**

delete a resource!

# Implementing DELETE

```
sumanah@perspective ~/test/http $ python serverwdelete.py  
serving at port 8000
```

# Implementing DELETE

```
sumanah@perspective ~/test/http $ python serverwdelete.py  
serving at port 8000
```

```
>>> import requests  
>>> r = requests.get("http://localhost:8000")  
>>> █
```

# Implementing DELETE

```
sumanah@perspective ~/test/http $ python serverwdelete.py  
serving at port 8000
```

```
>>> import requests  
>>> r = requests.get("http://localhost:8000")  
>>> █
```

```
sumanah@perspective ~/test/http $ python serverwdelete.py  
serving at port 8000  
127.0.0.1 - - [18/Jun/2015 19:53:49] "GET / HTTP/1.1" 200 -  
█
```

# Implementing DELETE

```
>>> import requests
>>> r = requests.get("http://localhost:8000")
>>> █
```

```
sumanah@perspective ~/test/http $ python serverwdelete.py
serving at port 8000
127.0.0.1 - - [18/Jun/2015 19:53:49] "GET / HTTP/1.1" 200 -
█
```

# Implementing DELETE

```
class APIHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):  
  
    def do_DELETE(self):  
        def delete_file(name):  
            self.send_response(204)  
            os.remove(name)  
            self.end_headers()  
        if self.path == '/':  
            self.send_response(403)  
            self.end_headers()  
        elif self.path == '/FileToDelete.txt':  
            delete_file("FileToDelete.txt")  
        else:  
            self.send_response(404)  
            self.end_headers()
```



# Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*  
Thu Jun 18 19:55:49 EDT 2015  
FileToDelete.txt
```

# Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*  
Thu Jun 18 19:55:49 EDT 2015  
FileToDelete.txt
```

```
>>> d = requests.delete("http://localhost:8000/FileToDelete.txt")
```

# Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*  
Thu Jun 18 19:55:49 EDT 2015  
FileToDelete.txt
```

```
>>> d = requests.delete("http://localhost:8000/FileToDelete.txt")
```

```
sumanah@perspective ~/test/http $ python serverwdelete.py  
serving at port 8000  
127.0.0.1 - - [18/Jun/2015 19:53:49] "GET / HTTP/1.1" 200 -  
127.0.0.1 - - [18/Jun/2015 19:56:52] "DELETE /FileToDelete.txt HTTP/1.1" 204 -  
█
```

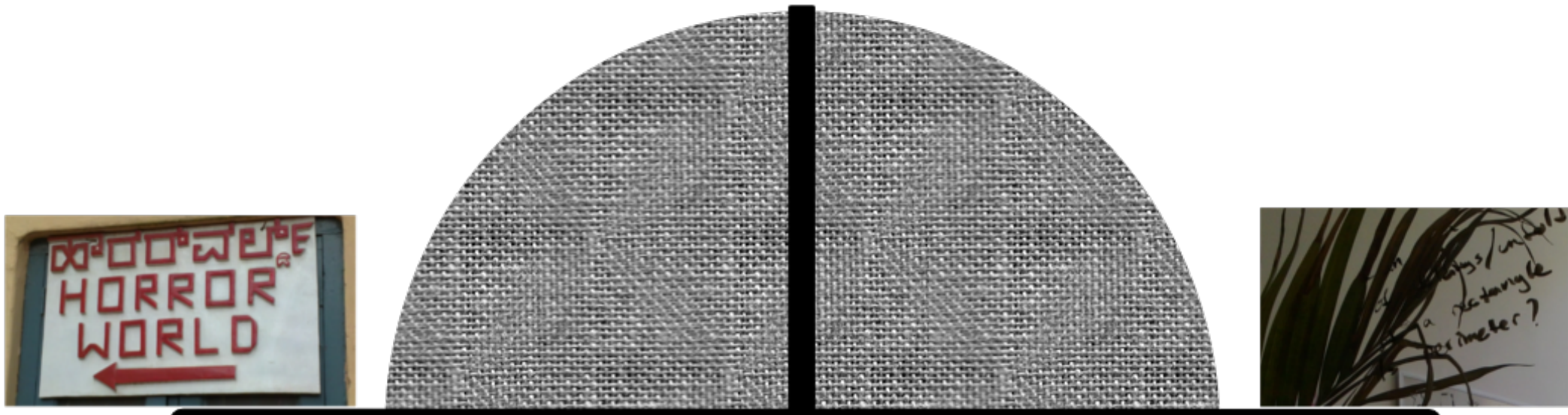
# Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*  
Thu Jun 18 19:55:49 EDT 2015  
FileToDelete.txt  
sumanah@perspective ~/test/http $ date && ls FileToDelete*  
Thu Jun 18 19:56:54 EDT 2015  
ls: cannot access FileToDelete*: No such file or directory  
sumanah@perspective ~/test/http $ █
```

# Implementing DELETE

```
>>> d.status_code
204
>>> d.reason
'No Content'
```

# DELETE – good idea?



# Underappreciated methods

**PUT**

“here you go”

# Wait

I thought **POST** meant “here you go”



# So what is **POST**, anyway?

The standard says it means:

“Above our pay grade; take this to the boss”

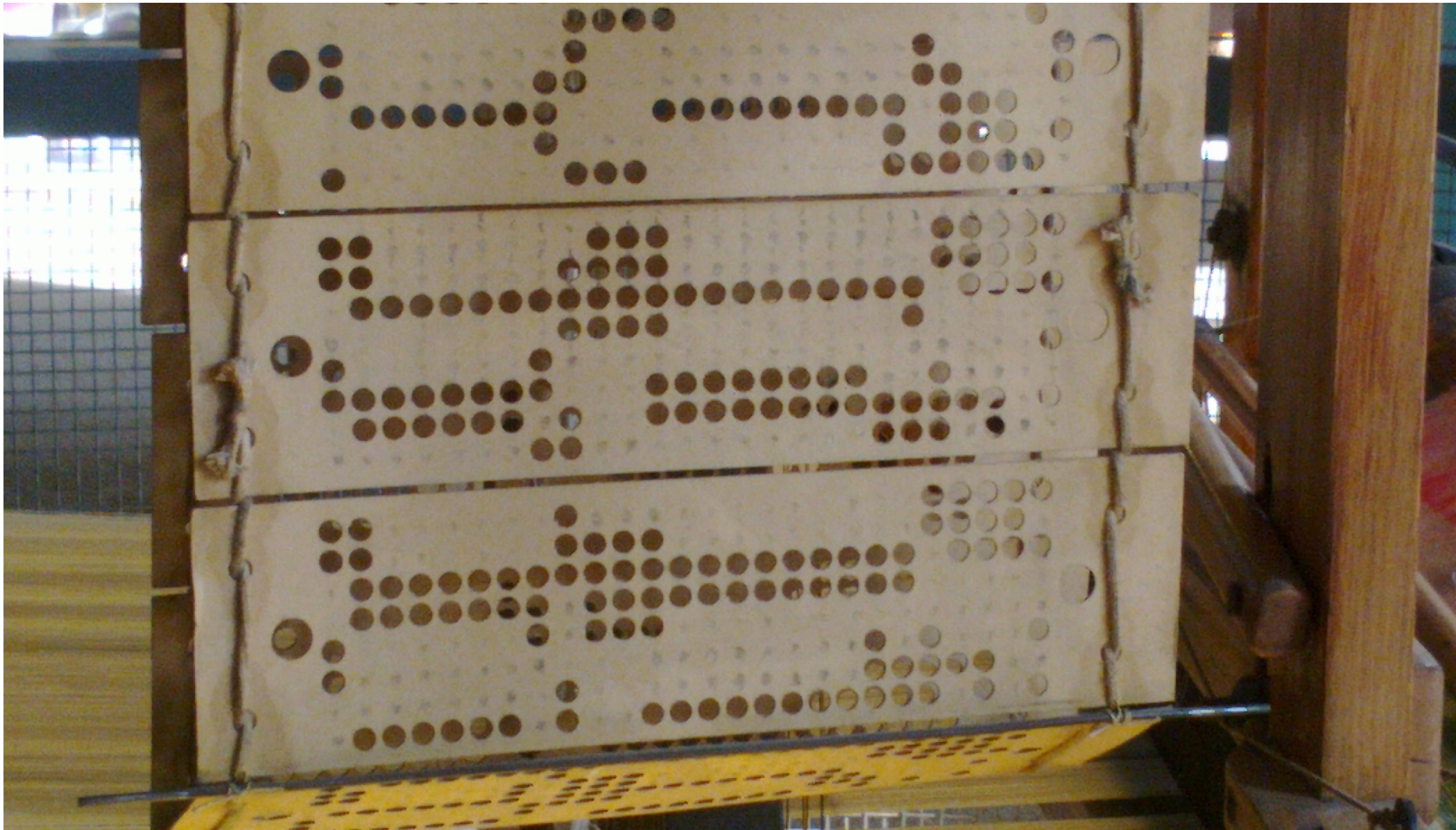
a.k.a. Overloaded **POST**

So what is **POST**,  
anyway?

Often, we use it for:

“Create a new item in this set”

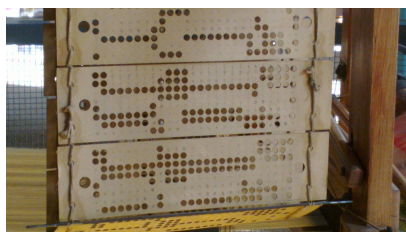
a.k.a. **POST**-to-append



# PUT vs. POST

**PUT** /cards /5

Body:

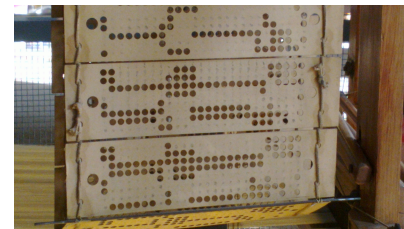


Means:

“Put this picture at /cards /5 .”

**POST** /cards /5

Body:



Means:

“Tell the webapp that this picture applies to /cards /5 somehow – figure it out.”

# “CRUD” & HTTP verbs

Create

**PUT**

Read

**GET**

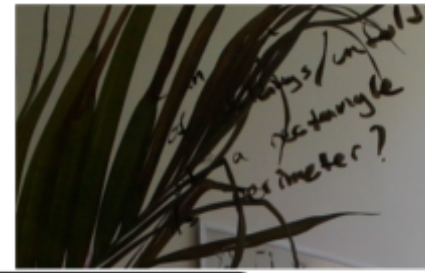
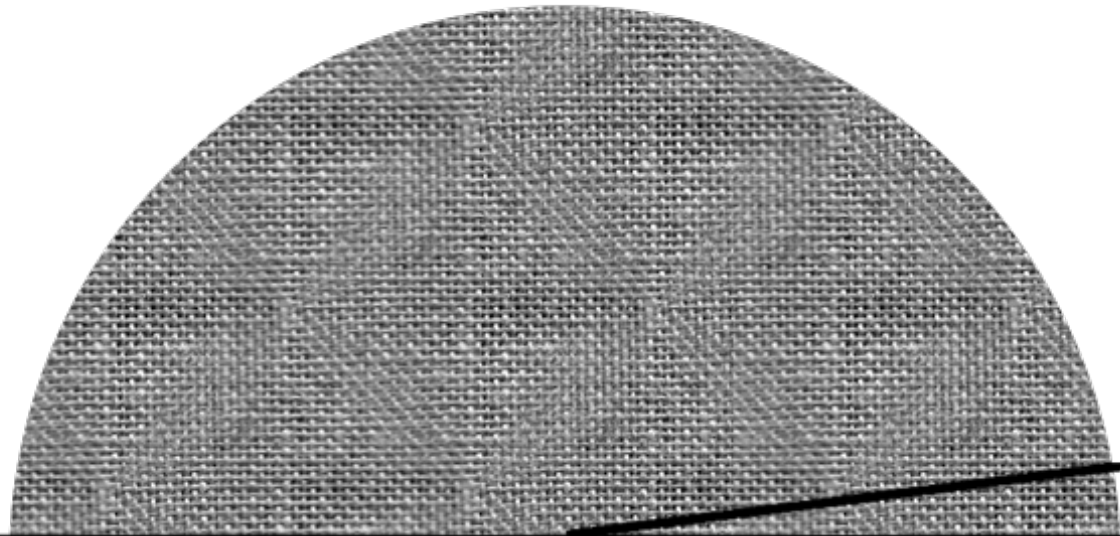
Update

**PUT**

Delete

**DELETE**

# PUT – good idea?

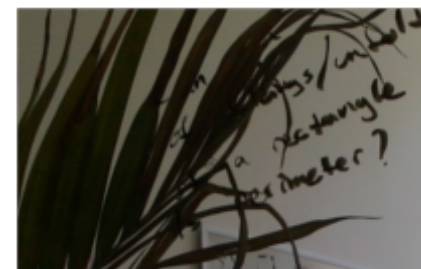
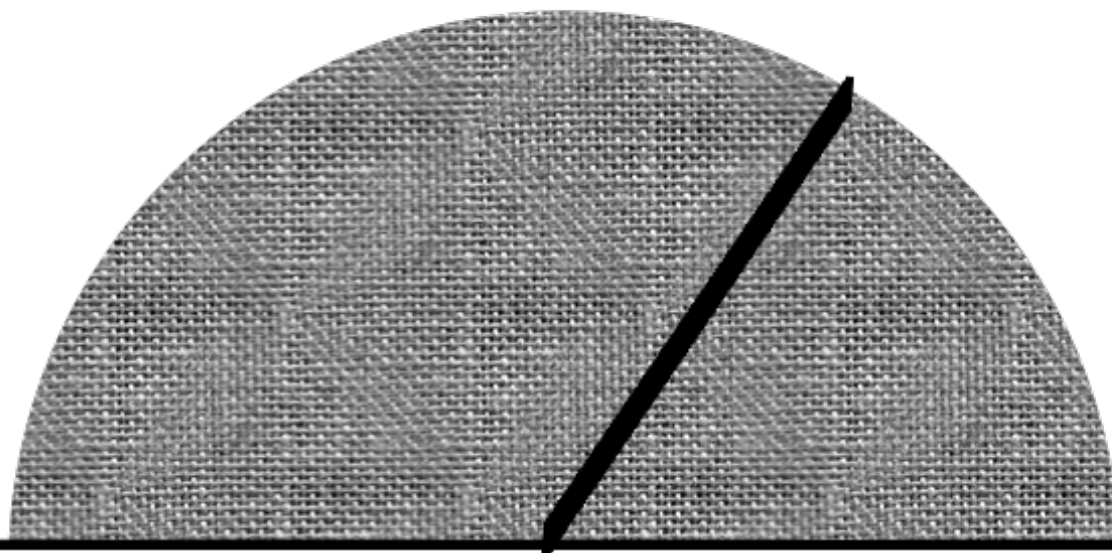


# More underused methods

- **PATCH**

update just part of this document/resource

# PATCH - good idea?





# More underused methods

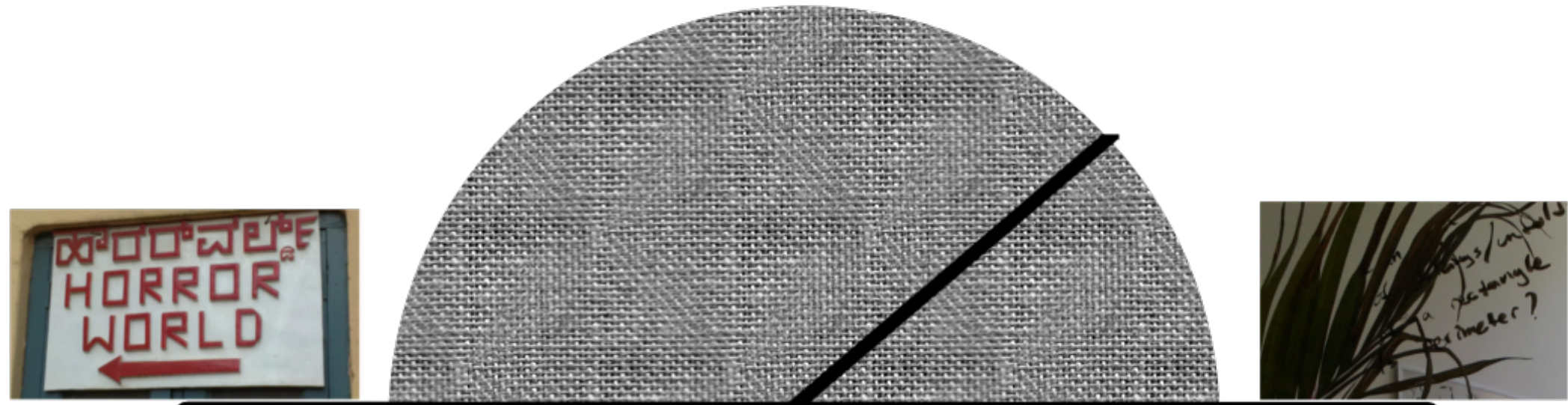
- **PATCH**

update just part of this document/resource

- **OPTIONS**

ask what verbs the client's allowed to use (for a specific path, or server-wide)

# OPTIONS – good idea?



A super-cool method

**HEAD**

like **GET**, but just for metadata

# GET vs. HEAD

Request:

**GET** / HTTP/1.1

Request:

**HEAD** / HTTP/1.1

Response:

- Start-line
- Headers
- Body

Response:

- Start-line
- Headers

# HEAD saves time

```
sumanah@perspective ~/test/http $ ipython
```

# HEAD saves time

```
sumanah@perspective ~/test/http $ ipython
```

```
In [1]: import requests
```

```
In [2]: uri = "https://upload.wikimedia.org/wikipedia/commons/3/3c/Lisebergskaninen_liseberg_goteborg_s  
weden_20100718.jpg"
```

```
In [3]: %timeit rfull = requests.get(uri)  
1 loops, best of 3: 1.33 s per loop
```

```
In [4]: %timeit rhead = requests.head(uri)  
1 loops, best of 3: 163 ms per loop
```

# HEAD saves time

```
In [3]: %timeit rfull = requests.get(uri)
1 loops, best of 3: 1.33 s per loop
```

```
In [4]: %timeit rhead = requests.head(uri)
1 loops, best of 3: 163 ms per loop
```

# You don't need the body to check:

Does it exist?

Do I have permission to **GET** it?

**Content-Length**

**Last-Modified**

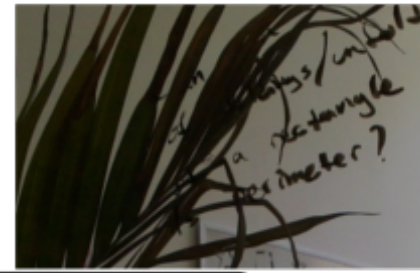
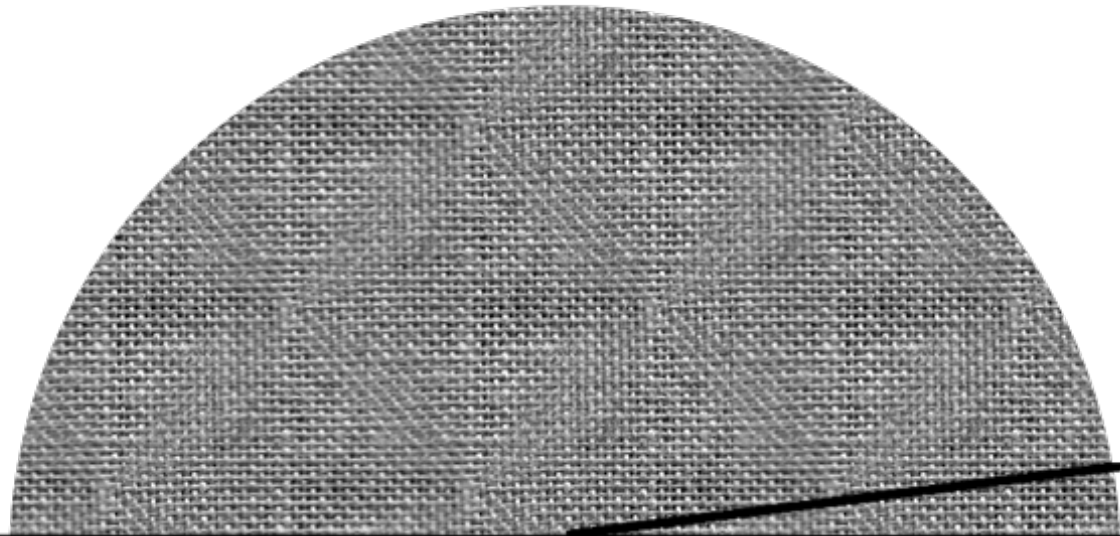
**Content-Type**

**ETag**

**Retry-After**



# HEAD – good idea?



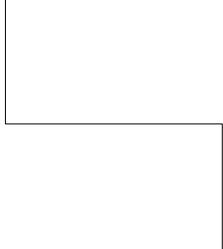
# Headers

# Popular headers include:

Content-Type  
Content-Length

# Popular headers include:

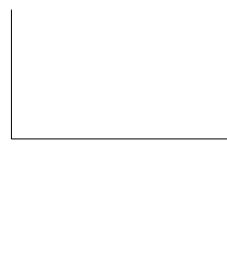
Also known as MIME or Mime



**Content-Type**  
**Content-Length**

# Popular headers include:

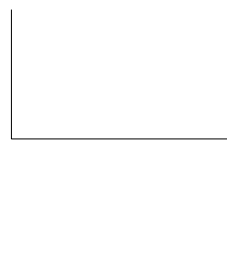
text/\*



Content-Type  
Content-Length

# Popular headers include:

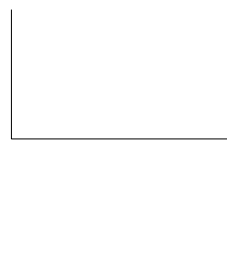
application/\*



Content-Type  
Content-Length

# Popular headers include:

chemical/\*



Content-Type  
Content-Length

# Popular headers include:

Content-Encoding

Accept-Encoding

Content-Language

Accept-Language



# More headers

ETag

If-Match

If-None-Match

# More headers

If-Modified-Since  
If-Unmodified-Since  
Last-Modified  
Cache-Control

# A popular header

User-Agent

# An unpopular header

**From**

The email address  
of the person making the request

# Uses for **From**

Really bad auth

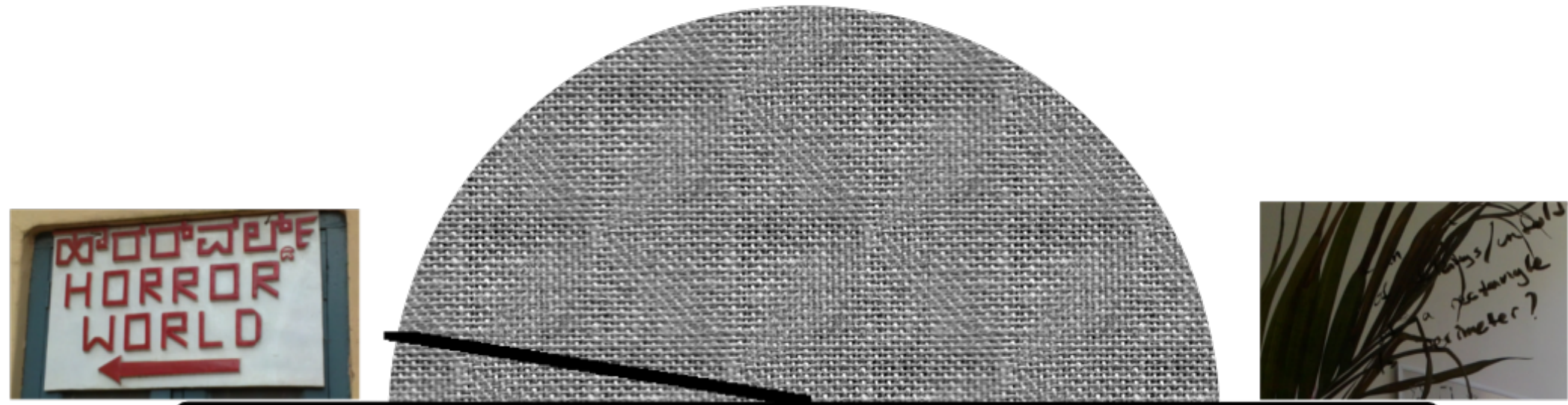
# Uses for **From**

“Yes, I saw your site launch”

# Uses for **F**rom

Coded messages  
meant for network surveillor

# From - bad idea





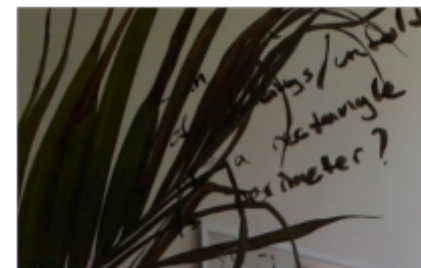
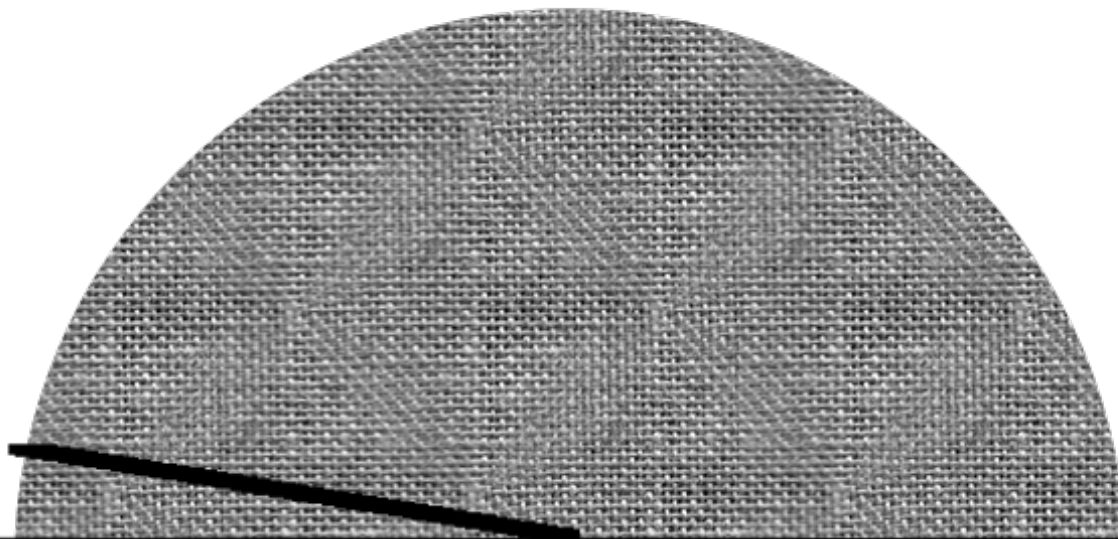
# Another spy trick

“Each header field consists of a case-insensitive field name followed by a colon (":")...”

– Internet Engineering Task Force (IETF) RFC 7230  
Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

So: vary the case of the headers you send!!!

# Header casing as secret info channel – bad idea



# A popular header

Host

# A required header

**Host**

required in request messages

# Host & path work together

```
$ netcat myhostname.tld 80
```

```
GET /bicycle HTTP/1.1
```

```
Host: myhostname.tld
```

# Host & path work together

The screenshot shows a web browser displaying the Astoria Bookshop website. The address bar shows the URL `www.astoriabookshop.com/event/storytime-73`. The page content includes the Astoria Bookshop logo, a navigation menu, and a section for the 'Storytime!' event. The event details are: Event date: Thursday, June 25, 2015 - 11:00am to 12:00pm; Event address: 31-29 31st Street. A shopping cart icon shows 0 items.

The browser's developer tools are open, showing the Network tab. The selected request is 'storytime-73'. The Headers tab is active, displaying the following information:

- General**
  - Remote Address: 162.242.246.191:80
  - Request URL: `http://www.astoriabookshop.com/event/storytime-73`
  - Request Method: GET
  - Status Code: 200 OK
- Response Headers (16)**
- Request Headers** (view parsed)
  - GET /event/storytime-73 HTTP/1.1
  - Host: `www.astoriabookshop.com`
  - Connection: keep-alive
  - Accept: `text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8`
  - User-Agent: `Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/38.0.2125.81 Chrome/43.0.2357.81 Safari/537.36`

# Host & path work together

 `www.astoriabookshop.com/event/storytime-73`

× Headers Preview Response Cookies Timing

## ▼ General

**Remote Address:** 162.242.246.191:80

**Request URL:** http://www.astoriabookshop.com/event/storytime-73

**Request Method:** GET

**Status Code:**  200 OK

## ▶ Response Headers (16)

## ▼ Request Headers view parsed

GET /event/storytime-73 HTTP/1.1

Host: www.astoriabookshop.com

# Host & path work together

The screenshot shows a web browser displaying a BBC news article. The address bar shows the URL `www.bbc.com/news/magazine-33139568`. The page content includes the BBC logo, navigation menus for News, Sport, Weather, Shop, Earth, Travel, and More, and a search bar. Below the navigation is a red banner with the word "NEWS" and sub-menus for Home, Video, World, US & Canada, UK, Business, Tech, Science, Magazine, and More. An advertisement placeholder is visible below the navigation. The main article title is "Do butterflies hold the answer to life's mysteries?" dated 16 June 2015. The browser's developer tools are open, showing the Network tab with a list of requests. The selected request is for `magazine-33139568`. The Headers pane shows the following request headers:

```
GET /news/magazine-33139568 HTTP/1.1
Host: www.bbc.com
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/43.0.2357.81 Chrome/43.0.2357.81 Safari/537.36
DNT: 1
Referer: http://www.bbc.com/news
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```



# Host & path work together

 [www.bbc.com/news/magazine-33139568](http://www.bbc.com/news/magazine-33139568)

---

▼ Request Headers      view parsed

```
GET /news/magazine-33139568 HTTP/1.1  
Host: www.bbc.com
```

# A popular header

**Host**

(wait –  
why do we need to repeat this?  
It's in the URL!  
right?)

# How **Host** helps

HTTP  
is separate from  
the Domain Name System

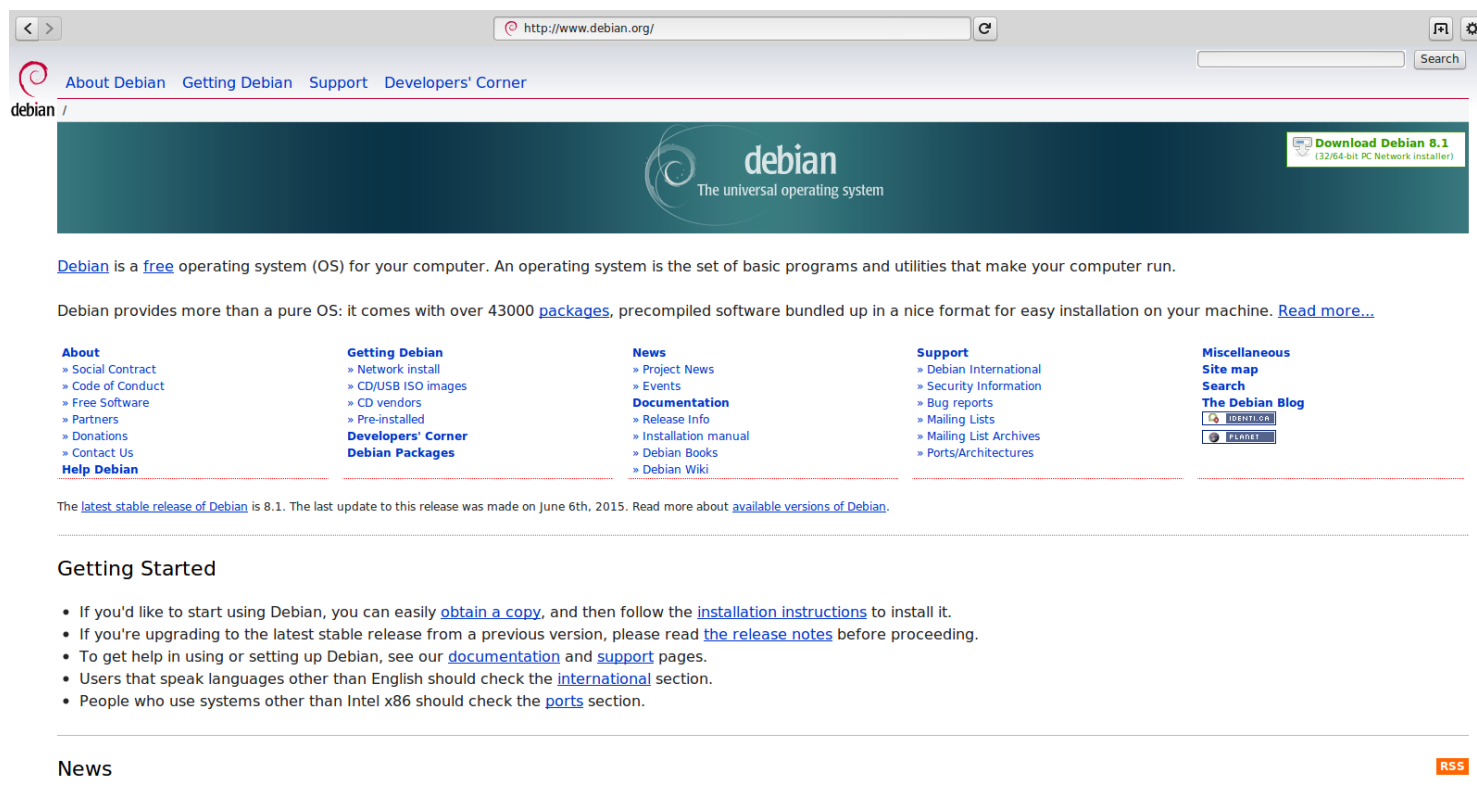
# How **Host** helps

## **Host**

helps route requests  
among different domains  
that sit on the same server

# Examples of virtual hosts

www.debian.org



The screenshot shows the Debian website homepage in a browser window. The address bar displays 'http://www.debian.org/'. The page features a navigation menu with links for 'About Debian', 'Getting Debian', 'Support', and 'Developers' Corner'. A search bar is located in the top right corner. The main content area has a dark green header with the Debian logo and the tagline 'The universal operating system'. A 'Download Debian 8.1' button is visible in the top right of the header. Below the header, the text states: 'Debian is a [free](#) operating system (OS) for your computer. An operating system is the set of basic programs and utilities that make your computer run. Debian provides more than a pure OS: it comes with over 43000 [packages](#), precompiled software bundled up in a nice format for easy installation on your machine. [Read more...](#)'

The page is organized into five columns of links:

- About**
  - » Social Contract
  - » Code of Conduct
  - » Free Software
  - » Partners
  - » Donations
  - » Contact Us
  - Help Debian**
- Getting Debian**
  - » Network install
  - » CD/USB ISO images
  - » CD vendors
  - » Pre-installed
  - Developers' Corner**
  - Debian Packages**
- News**
  - » Project News
  - » Events
  - Documentation**
  - » Release Info
  - » Installation manual
  - » Debian Books
  - » Debian Wiki
- Support**
  - » Debian International
  - » Security Information
  - » Bug reports
  - » Mailing Lists
  - » Mailing List Archives
  - » Ports/Architectures
- Miscellaneous**
  - Site map**
  - Search**
  - The Debian Blog**
  - IDENTICON
  - PLANET

Below the columns, a notice states: 'The [latest stable release of Debian](#) is 8.1. The last update to this release was made on June 6th, 2015. Read more about [available versions of Debian](#).'

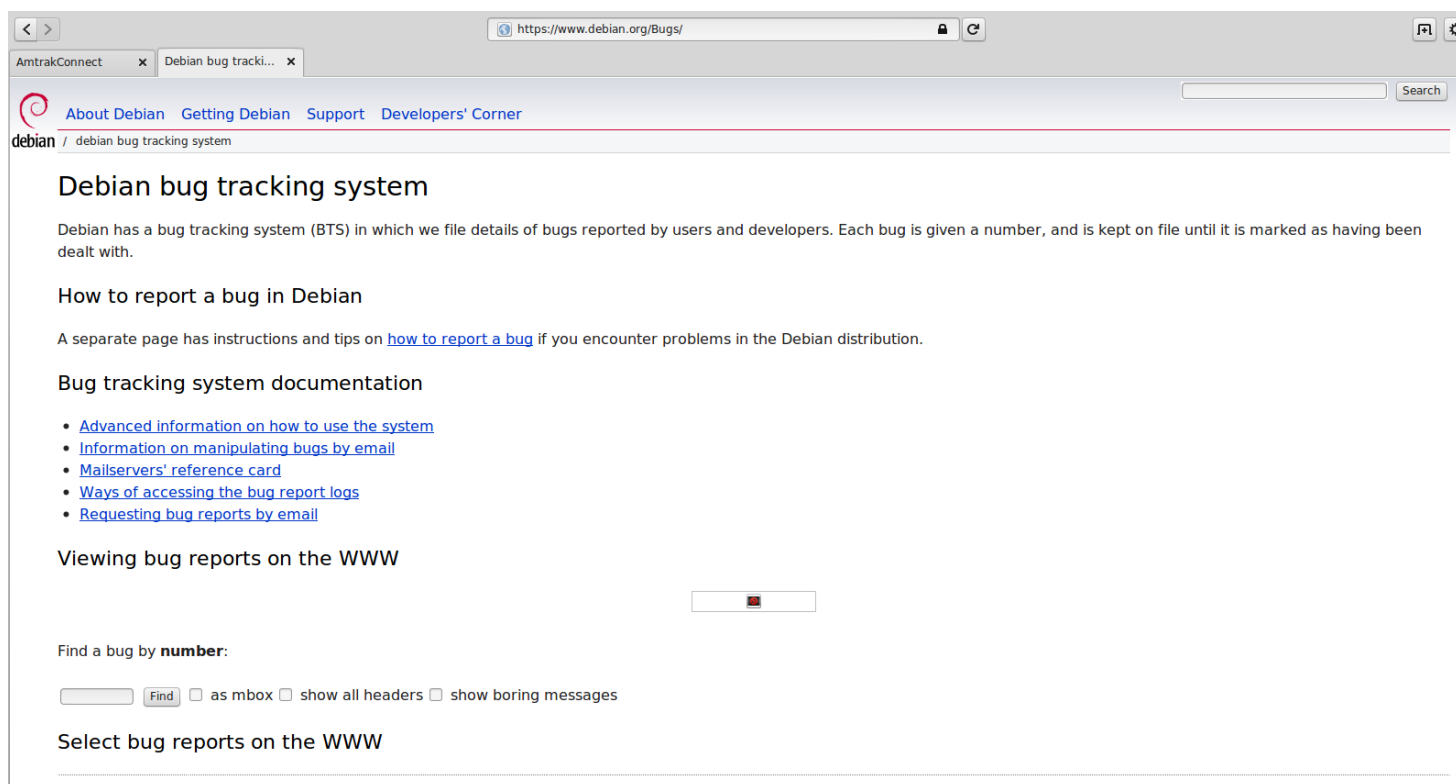
The 'Getting Started' section contains the following list:

- If you'd like to start using Debian, you can easily [obtain a copy](#), and then follow the [installation instructions](#) to install it.
- If you're upgrading to the latest stable release from a previous version, please read [the release notes](#) before proceeding.
- To get help in using or setting up Debian, see our [documentation](#) and [support](#) pages.
- Users that speak languages other than English should check the [international](#) section.
- People who use systems other than Intel x86 should check the [ports](#) section.

The footer includes the word 'News' on the left and an 'RSS' icon on the right.

# Examples of virtual hosts

bugs.debian.org



The screenshot shows a web browser window with the URL `https://www.debian.org/Bugs/`. The page title is "Debian bug tracking system". The navigation menu includes "About Debian", "Getting Debian", "Support", and "Developers' Corner". The main content area contains the following sections:

- Debian bug tracking system**

Debian has a bug tracking system (BTS) in which we file details of bugs reported by users and developers. Each bug is given a number, and is kept on file until it is marked as having been dealt with.
- How to report a bug in Debian**

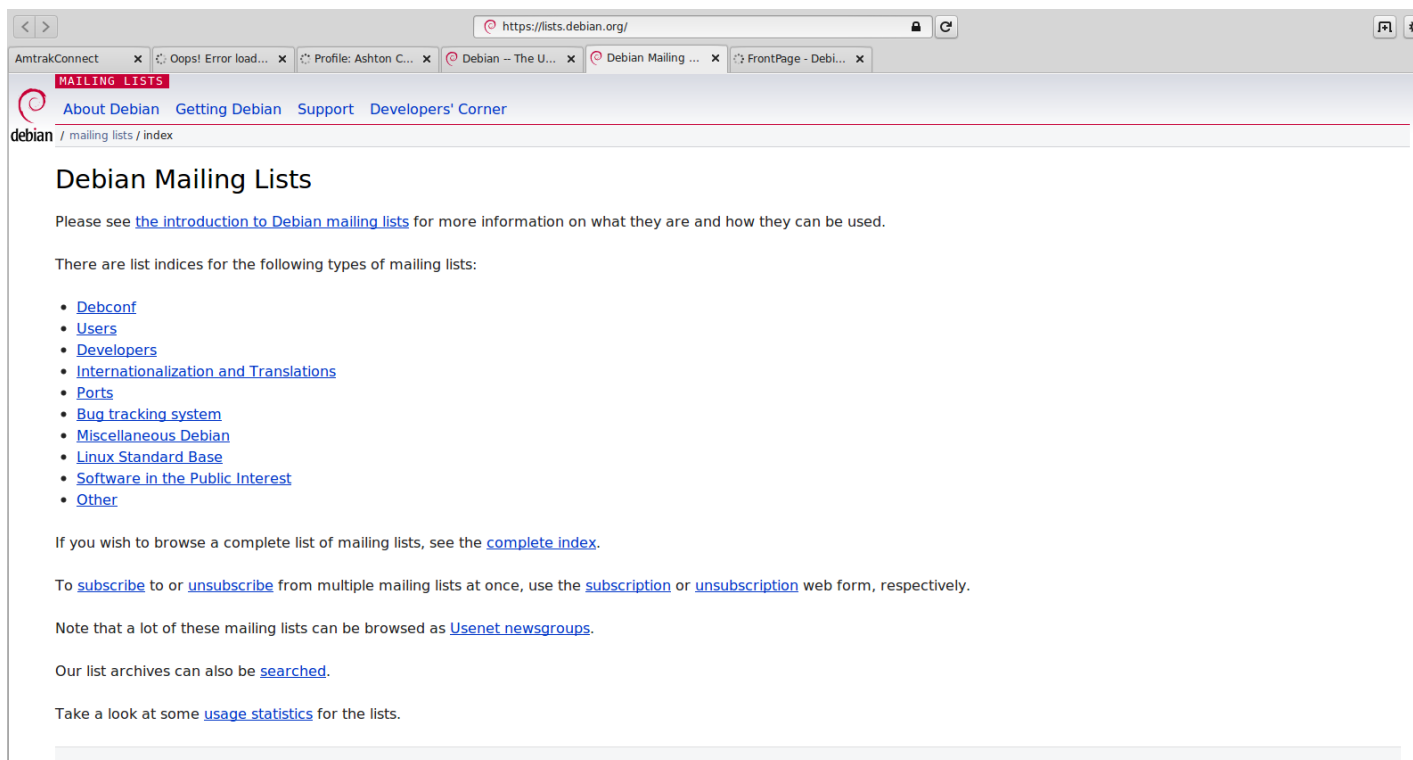
A separate page has instructions and tips on [how to report a bug](#) if you encounter problems in the Debian distribution.
- Bug tracking system documentation**
  - [Advanced information on how to use the system](#)
  - [Information on manipulating bugs by email](#)
  - [Mailservers' reference card](#)
  - [Ways of accessing the bug report logs](#)
  - [Requesting bug reports by email](#)
- Viewing bug reports on the WWW**

Find a bug by **number**:

as mbox  show all headers  show boring messages
- Select bug reports on the WWW**

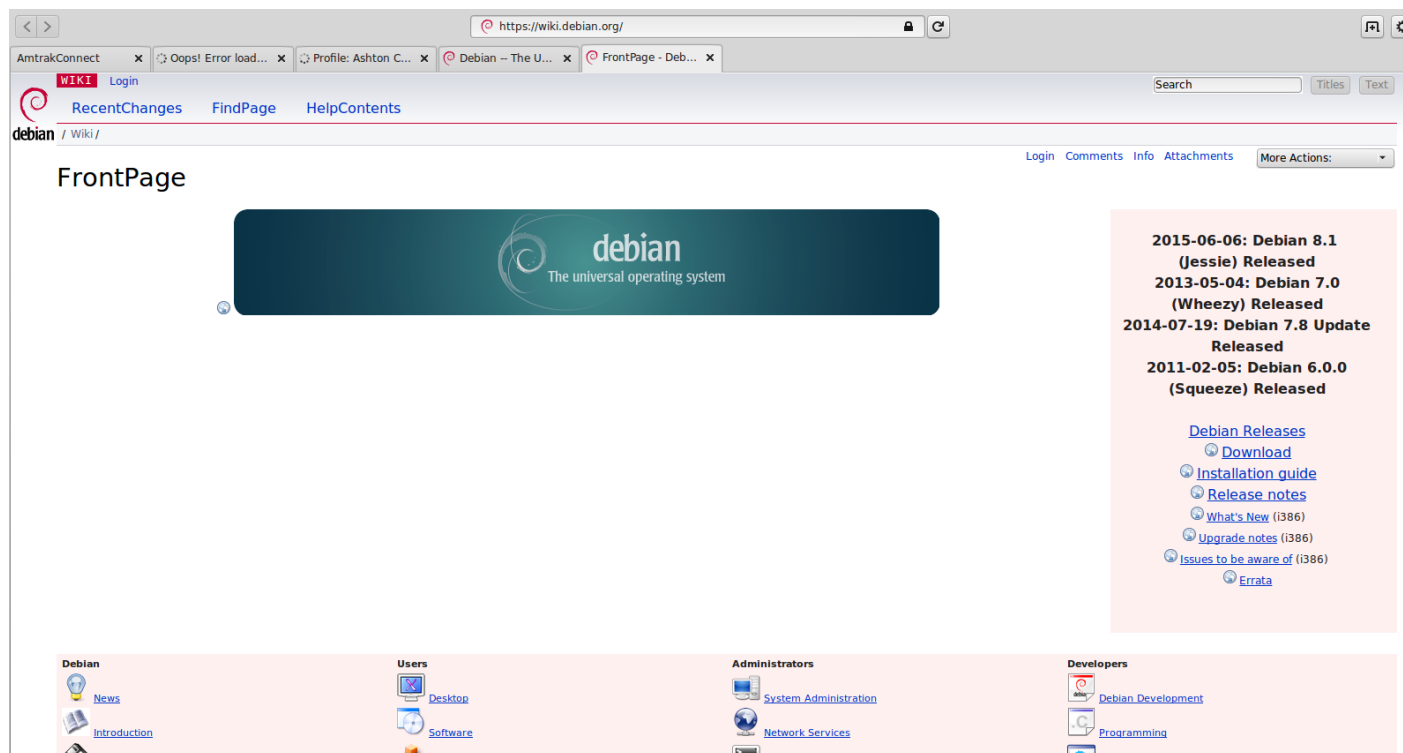
# Examples of virtual hosts

lists.debian.org



# Examples of virtual hosts

wiki.debian.org





# But watch out...

```
>>> import requests
>>> headers = {"Host": "sumanariffic"}
>>> r = requests.get("http://www.debian.org", headers=headers)
```

# But watch out...

```
>>> r.text
'<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">\n<HTML>\n<HEAD>\n  <TITLE>We
lcome to mirror-csail!</TITLE>\n</HEAD>\n<BODY>\n\n\n<H1>Welcome to mirror-csail!<
/H1>\n\n\nThis is mirror-csail, a system run by and for the <a href="https://www.d
ebian.org/">Debian Project</a>.\nShe does stuff.\nWhat kind of stuff and who our
kind sponsors are you might learn on\n<a href="https://db.debian.org/machines.c
gi?host=mirror-csail">db.debian.org</a>.\n\n\n<P>\n<HR NOSHADE />\n<FONT size="-1"
>DSA</FONT>\n\n\n</BODY>\n</HTML>\n'
```

# A spam story

# A spam story

My **404** logs (Drupal admin console):

TYPE page not found

DATE Thursday, October 9, 2014 – 10:46

USER Anonymous (not verified)

LOCATION <http://myphishingsite.biz/http://myphishingsite.biz>

REFERRER

MESSAGE [ttp://myphishingsite.biz](http://myphishingsite.biz)

SEVERITY warning

HOSTNAME *[IP address]*

# A spam story

My 404 logs (Drupal admin console):

TYPE page not found

DATE Thursday, October 9, 2014 – 10:46

USER Anonymous (not verified)

LOCATION <http://myphishingsite.biz/http://myphishingsite.biz>

REFERRER

MESSAGE ttp://myphishingsite.biz

SEVERITY warning

HOSTNAME *[IP address]*

# A spam story

My access logs:

```
[IP address] - -  
[09/Oct/2014:10:46:09 -0400]  
"GET http://myphishingsite.biz  
HTTP/1.1" 404 7574 "-" [User-Agent]
```

# A spam story

Legit mistakes would look like:

```
[IP address] - -
```

```
[09/Oct/2014:10:46:09 -0400]
```

```
"GET /http://berkeley.edu HTTP/1.1"
```

```
404 7574 "-" [User-Agent]
```

# A spam story

Intentionally malformed your request!

```
$ netcat myhostname.tld 80
```

```
GET http://spam.com HTTP/1.1
```

```
Host: spam.com
```

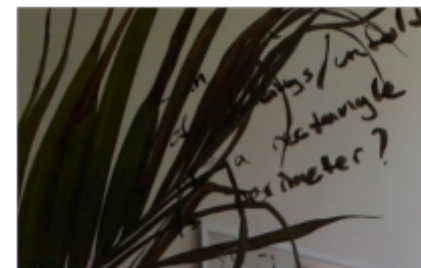
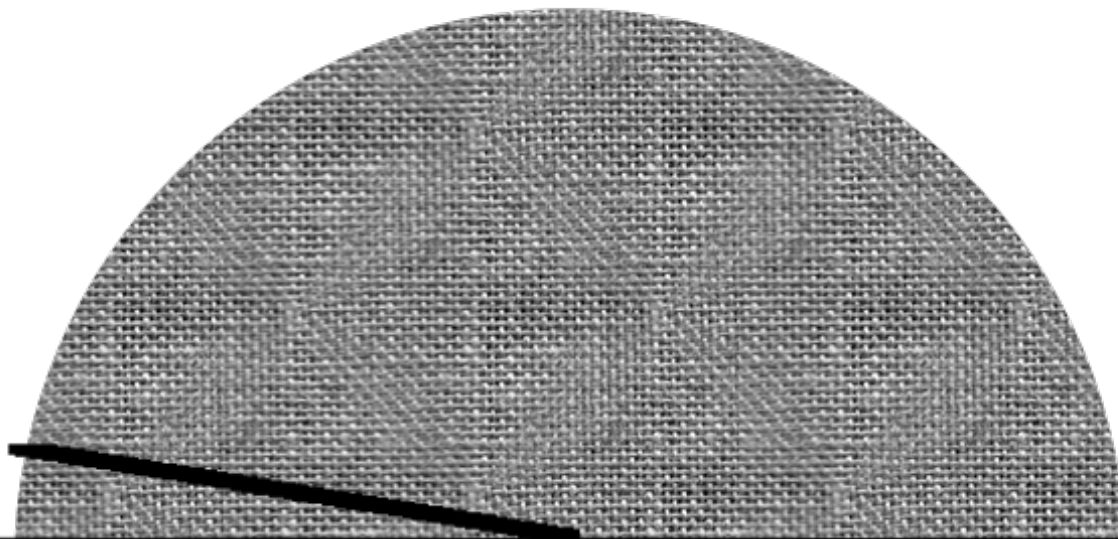


# A spam story

Intentionally malformed your request!

```
$ netcat myhostname.tld 80  
GET /viagra-bitcoin HTTP/1.1  
Host: spam.com
```

# 404 spamming – bad idea



# Define your own header!

“Header fields are fully extensible: there is no limit on the introduction of new field names, each presumably defining new semantics, nor on the number of header fields used in a given message.”

–(RFC 7230)

# Define your own header!

X-blah-blah-blah

# Define your own header!

`X-Wikimedia-Debug`

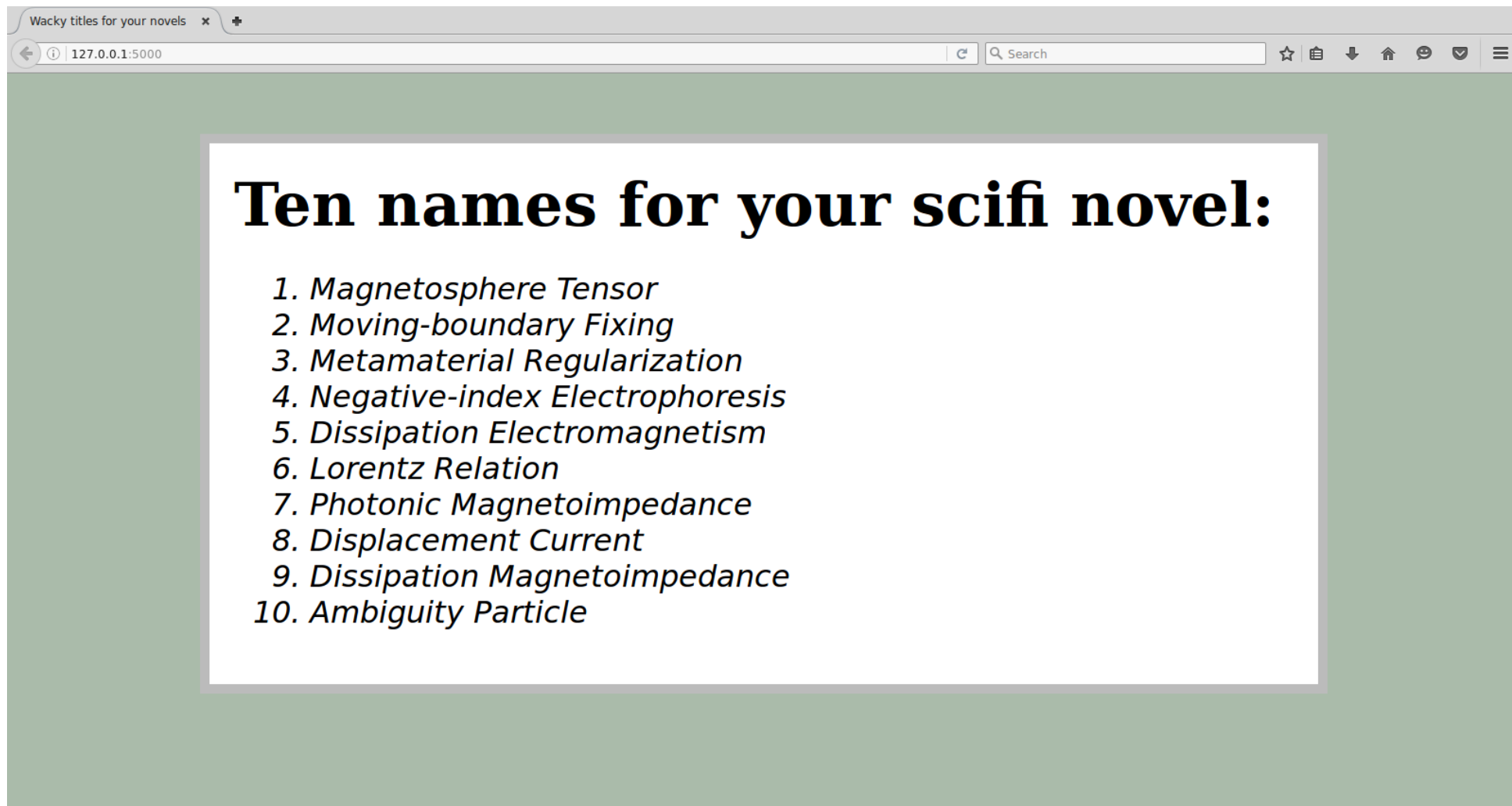
# Define your own header!

`X-Wikimedia-Debug`  
an HTTP request header

- Backend selection (Varnish)
- Caching behavior
- Request profiling (record a trace)
- Debug logs
- Read-only mode
- Browser extensions

More: <https://wikitech.wikimedia.org/wiki/X-Wikimedia-Debug>

# Define your own header!



## **Ten names for your sci-fi novel:**

- 1. Magnetosphere Tensor*
- 2. Moving-boundary Fixing*
- 3. Metamaterial Regularization*
- 4. Negative-index Electrophoresis*
- 5. Dissipation Electromagnetism*
- 6. Lorentz Relation*
- 7. Photonic Magnetoimpedance*
- 8. Displacement Current*
- 9. Dissipation Magnetoimpedance*
- 10. Ambiguity Particle*

# Define your own header!

```
import random
import requests
-from flask import Flask,render_template
+from flask import Flask,render_template,make_response
app_fl = Flask(__name__)

@app_fl.route('/')
@@ -18,7 +19,9 @@ def index_pg():
    pagetitles = getarticletitles()
    adj, noun = massagelist(pagetitles)
    sugg = wackytitles(adj, noun)
-    return render_template('titles.html',title=sugg)
+    response = make_response(render_template('titles.html',title=sugg))
+    response.headers['X-Sumana-Is-Amazing'] = 'Indeed, Verily So'
+    return response
```



# Define your own header!

```
response = make_response(render_template('titles.html',title=sugg))
response.headers['X-Sumana-Is-Amazing'] = 'Indeed, Verily So'
return response
```

# Define your own header!

The screenshot shows a web browser window with a single tab titled "Wacky titles for your novels". The address bar shows "127.0.0.1:5000". The main content area displays a white box with the heading "Ten names for your sci-fi novel:" followed by a numbered list of ten sci-fi related terms:

1. *Magnetosphere Tensor*
2. *Moving-boundary Fixing*
3. *Metamaterial Regularization*
4. *Negative-index Electrophoresis*
5. *Dissipation Electromagnetism*
6. *Lorentz Relation*
7. *Photonic Magnetoimpedance*
8. *Displacement Current*
9. *Dissipation Magnetoimpedance*
10. *Ambiguity Particle*

The browser's developer tools are open to the Network tab. The network log shows two requests:

Status	Method	File	Domain	Type	Transferred
200	GET	/	127.0.0.1:5000	html	0.52 KB
200	GET	basic.css	127.0.0.1:5000	css	0.24 KB

The selected request (the root page) is expanded to show the Headers sub-tab. The request details are:

- Request URL: http://127.0.0.1:5000/
- Request method: GET
- Remote address: 127.0.0.1:5000
- Status code: 200 OK
- Version: HTTP/1.0

The response headers are:

- Content-Length: "530"
- Content-Type: "text/html; charset=utf-8"
- Date: "Tue, 31 May 2016 06:50:44 GMT"
- Server: "Werkzeug/0.11.10 Python/2.7.6"
- X-Sumana-Is-Amazing: "Indeed, Verily So"

At the bottom of the developer tools, a summary bar indicates "2 requests, 0.76 KB, 0.70 s".

# Define your own header!

▼ Response headers (0.191 KB)

**Content-Length:** "530"

**Content-Type:** "text/html; charset=utf-8"

**Date:** "Tue, 31 May 2016 06:50:44 GMT"

**Server:** "Werkzeug/0.11.10 Python/2.7.6"

**X-Sumana-Is-Amazing:** "Indeed, Verily So"

<https://gitlab.com/http-can-do-that/novel-titles>

# Define your own header!

▼ Response headers (0.191 KB)

**Content-Length:** "530"

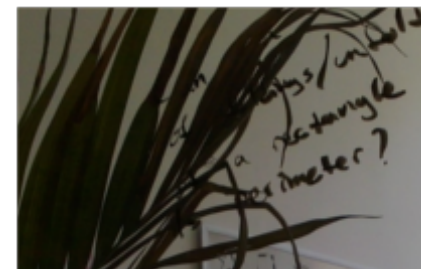
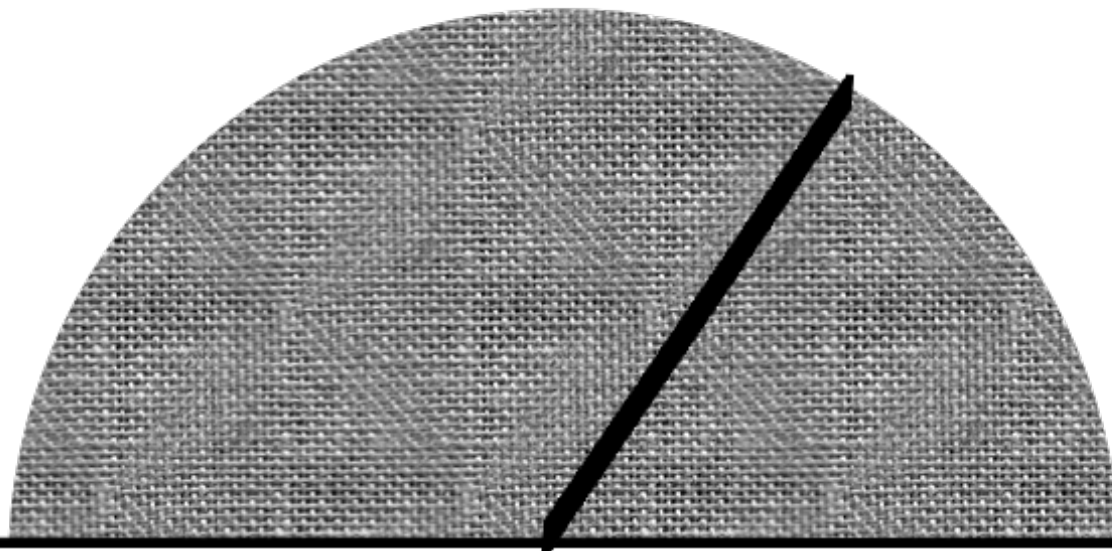
**Content-Type:** "text/html; charset=utf-8"

**Date:** "Tue, 31 May 2016 06:50:44 GMT"

**Server:** "Werkzeug/0.11.10 Python/2.7.6"

**X-Sumana-Is-Amazing:** "Indeed, Verily So"

# Defining your own headers – good idea?



# Status codes

# Status codes

100 & 101: Informational

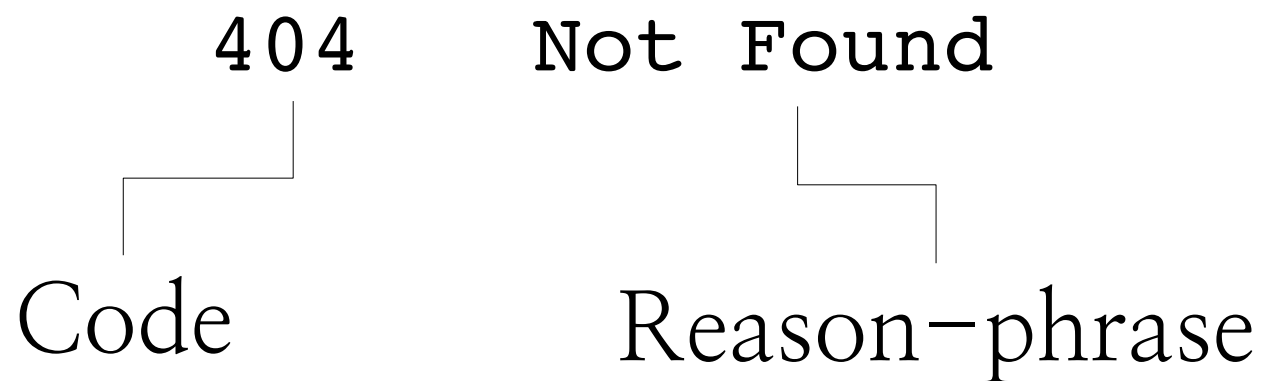
2xx: Successful

3xx: Redirection

4xx: Client error

5xx: Server error

# Status (response) codes





# Status (response) codes

“A client SHOULD ignore  
the reason–phrase content.”

# Heard of these?

- **410 Gone**

It was here, but now it's not.

- **304 Not Modified**

You said, 'GET this, if it's been modified since [date]'. It hasn't been.

# 451 Unavailable For Legal Reasons

Server is legally required  
to reject client's request

# 451 Unavailable For Legal Reasons

Can't let you see that; it's censored.

# 451 Unavailable For Legal Reasons

“This is considered a client-side error even though the request is well formed and the legal requirement exists on the server side. After all, that representation was censored for a reason. There must be something wrong with you, citizen.”

–RESTful Web APIs,

Leonard Richardson & Mike Amundsen

# 451 Unavailable For Legal Reasons

Internet Engineering Task Force (IETF)  
Request for Comments: 7725  
Category: Standards Track  
ISSN: 2070-1721

T. Bray  
Textuality  
February 2016

## An HTTP Status Code to Report Legal Obstacles

### Abstract

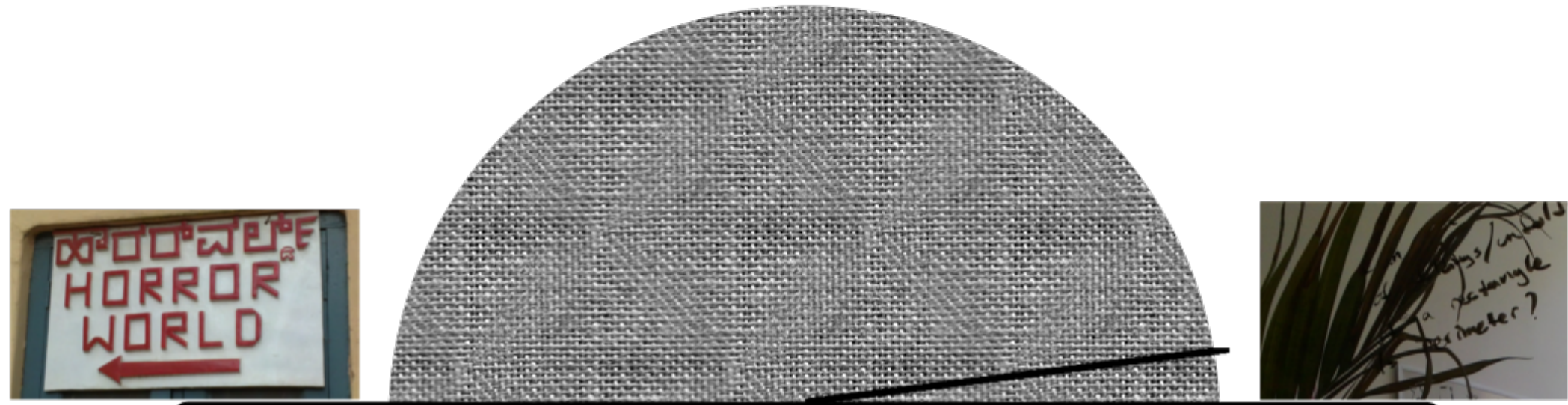
This document specifies a Hypertext Transfer Protocol (HTTP) status code for use when resource access is denied as a consequence of legal demands.

### Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

451 – good idea?



# WTF responses

All of these  
were found in the wild



# WTF responses

Code: 126

Reason: Incorrect key file for table  
'/tmp/mysqltmp/#sql\_13fb\_2.MYI'; try to repair it  
SQL=SHOW FULL COLUMNS FROM  
'y4dnu\_extensions'

# WTF responses

Code: 301

Reason: explicit\_header\_response\_code

# WTF responses

Code: 403

Reason: You've got to ask yourself one question:  
Do I feel lucky?

# WTF responses

Code: 403

Reason: can't put wasabi in bed

# WTF responses

Code: 404

Reason: HTTP/1.1 404

# WTF responses

Code: 404

Reason: Not Found"); ?> <?php Header("cache-  
Control: no-store, no-cache, must-revalidate

# WTF responses

Code: 200

Reason: Forbidden

# WTF responses

Code: 404

Reason: Apple WebObjects



# WTF responses

Code: 404

Reason: forbidden

# WTF responses

Code: 434

Reason: HTTP/1.1 434

# WTF responses

Code: 451

Reason: Unknown Reason-Phrase

# WTF responses

Code: 503

Reason: Backend is unhealthy

# WTF responses

Code: 520

Reason: Origin Error

# WTF responses

Code: 525

Reason: Origin SSL Handshake Error

# WTF responses

Code: 533

Reason: mtd::http: Unknown: Banned

# WTF responses

Code: 732

Reason:

[http://www.\[hostname\].com/intro/copyright.php](http://www.[hostname].com/intro/copyright.php)



# WTF responses

Code: 999

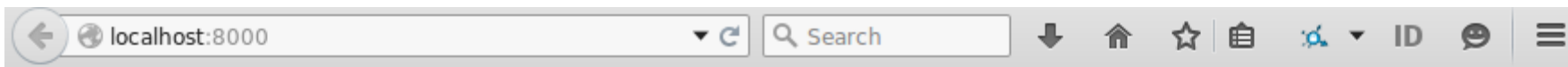
Reason: Request denied

# Changing Reason-Phrases

```
import http.server

class OddHeaderHTTPRequestHandler(http.server.SimpleHTTPRequestHandler):
    responses = dict(http.server.SimpleHTTPRequestHandler.responses)
    responses[200] = ('Oll Korrekt', 'Oh Kay')
    responses[404] = ('I Know Nothing', 'Nothing here by that name')
```

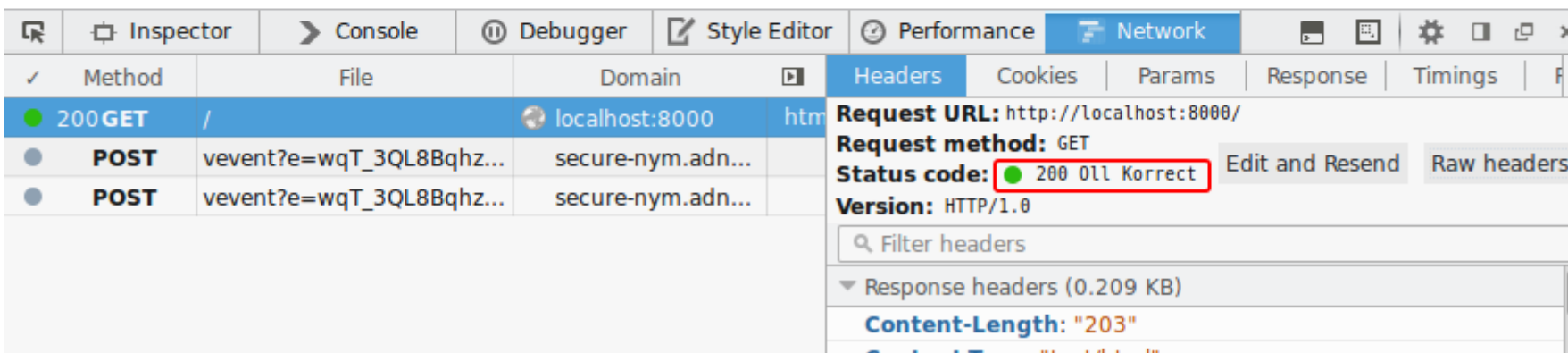
more at <https://gitlab.com/http-can-do-that>



# Rockin'

This is a pretty rockin' site. I'm glad you decided to visit.

# Wheee!



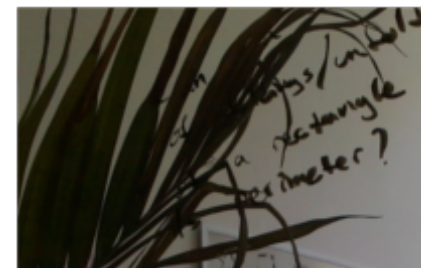
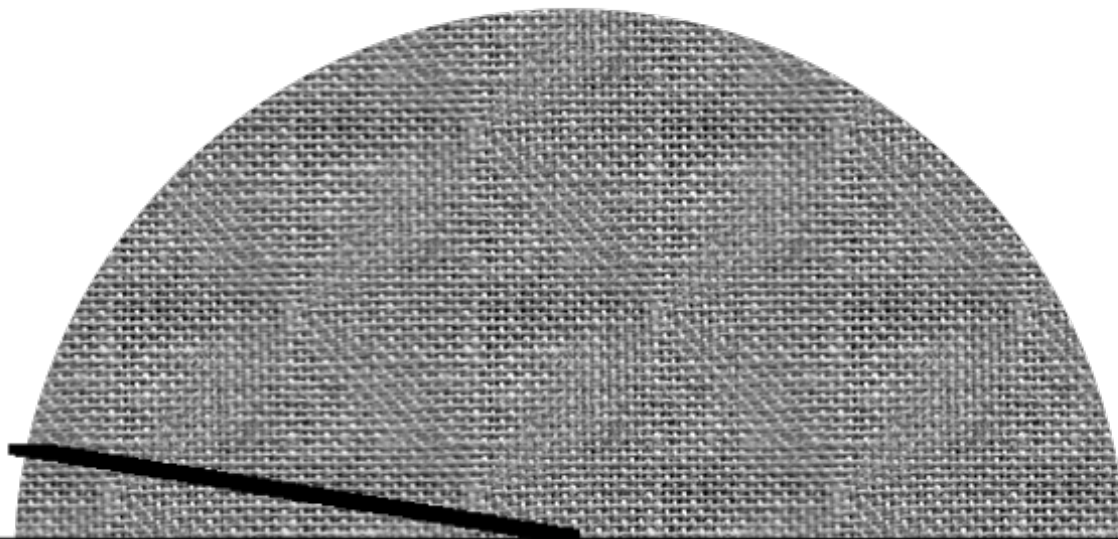
**Request URL:** http://localhost:8000/

**Request method:** GET

**Status code:** ● 200 OK Korrekt

Edit and Resend

# Bespoke status codes/reasons – good idea?



# Conclusion

# There's so much more

- “Don't cache this”
- **Pragma** – pass instructions to server/client
- **CONNECT, TRACE, LINK, & UNLINK** methods
- **409 Conflict**
- Look-before-you-leap requests
- Resources at HTTPS vs. HTTP URLs can differ
- “q” and preference ranking in the Accept header
- Content-Disposition (e.g. “attachment”)

The feeling of power

The sense of wonder



What might the web have been?

What might it still be?

# Read & play

- RFCs 7230–7235
- `requests`
- `netcat`, `wget`, `netstat`, `telnet`
- basic HTTP servers (in your favorite language)
- <https://gitlab.com/http-can-do-that>

# Thanks

Leonard Richardson

Greg Hendershott

Zack Weinberg

The Recurse Center

Clay Hallock

Paul Tagliamonte

Open Source Bridge

Julia Evans, Allison Kaptur, Amy Hanlon, and  
Katie Silverio

# Thank you

Sumana Harihareswara

<http://changeset.nyc>

@brainwane

<https://gitlab.com/http-can-do-that>

[sumanah@panix.com](mailto:sumanah@panix.com)