Class Topics

- Web Architecture
  - Why is there a reload button?
  - How does HTTP work?
  - Why is a connection to Amazon secure from eavesdroppers?
  - How does a browser work?
- Web Semantics
  - How do auctions (Ebay), recommenders (Netflix, Amazon), search engines (Google) work?
  - Basic data mining
- Web-Scale Systems
  - DNS, Akamai, Google File System, MapReduce, others

Course Info

- Lectures
  - Attend whichever you prefer
  - Lectures are also recorded
- Discussion sections
  - See class Google calendar http://eecs485.org
- Major pieces of work:
  - Midterm, Final (gCal/Syllabus for dates)
  - 5 programming assignments
- The Web and this class draw from many sub-disciplines of computer science

Programming Projects

- 5 projects, which build on each other
- By the end of class, you will build Google Web Search, circa 2004
- Lots of opportunity for custom features, bells and whistles
- Your chance to be creative and build something great

- **Project 1 starts this week**

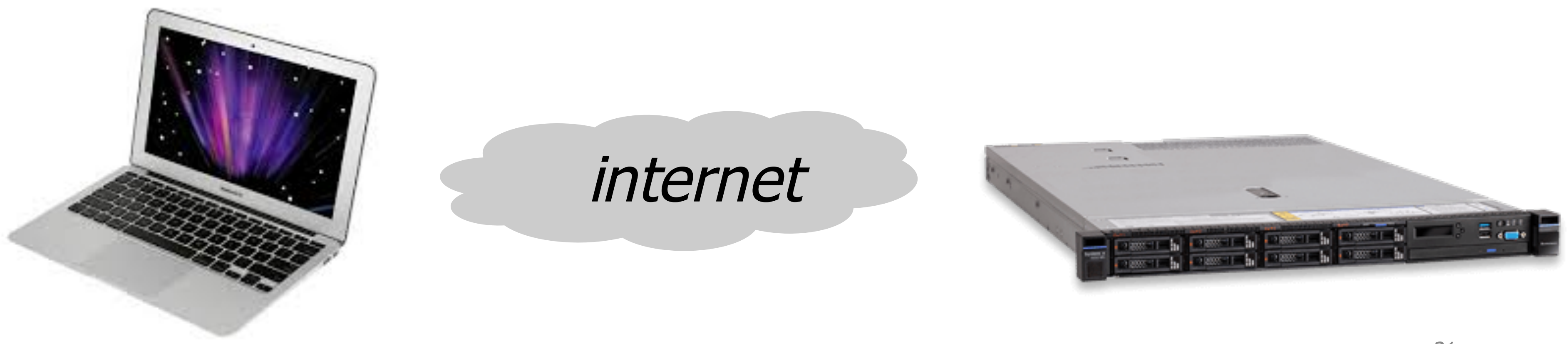
Class Mechanics

- Programming projects
  - Go out regularly, starting this week
  - Each builds on the last; don’t fall behind
  - No late days
  - Get hints from anywhere, but your code is yours

- Grade
  - 5 projects x 10% each = 50%
  - Midterm Exam 23%
  - Final Exam 27%

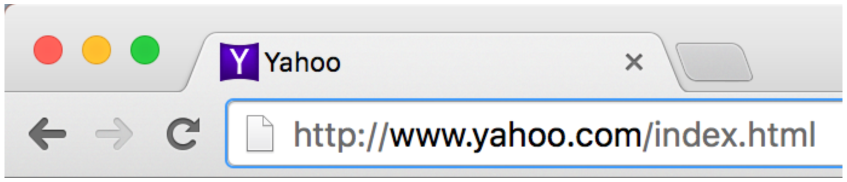
The request response cycle

- The request response cycle is how two computers communicate with each other on the web
- It's simple:
  - A client requests some data
  - A server responds to the request



The request response cycle

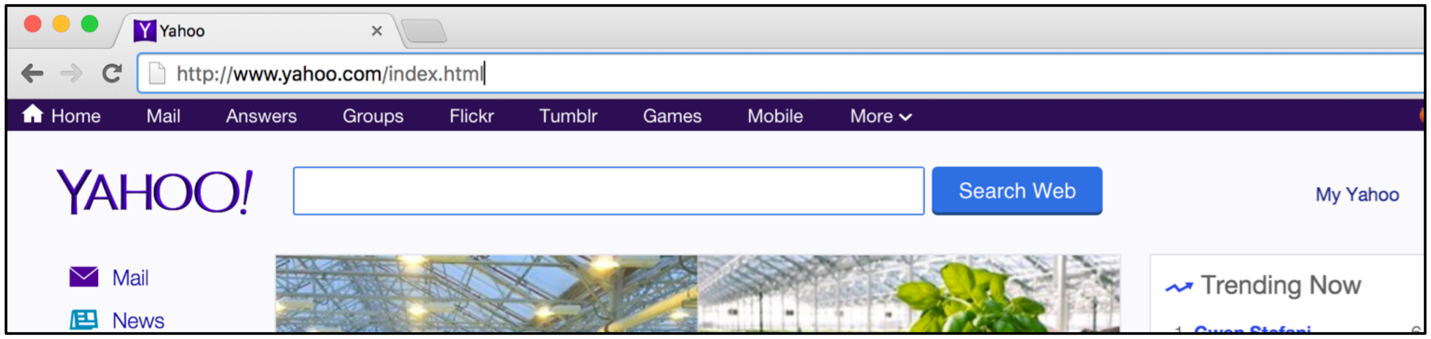
- A client requests a web page



- A server responds with an HTML file

```
<!DOCTYPE html>
...
```

- The client renders the HTML

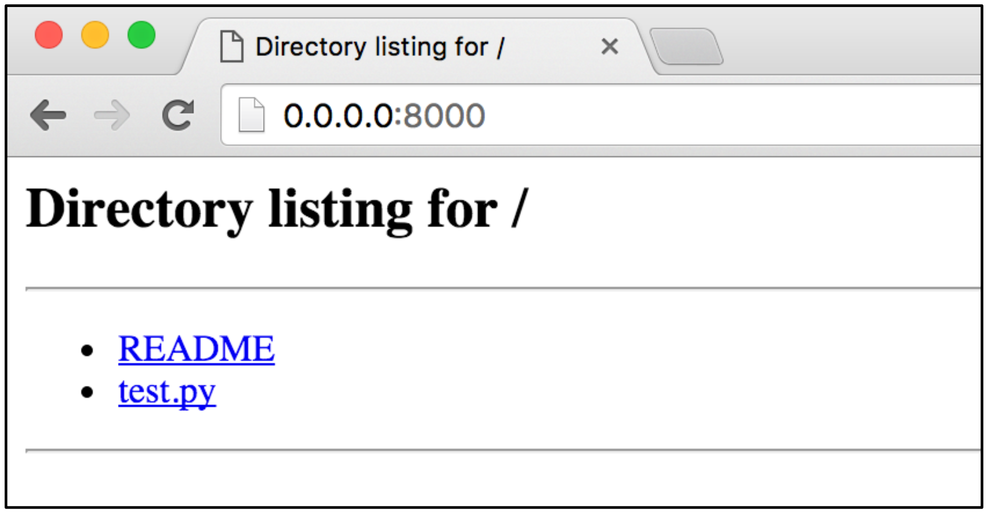


How does a server respond?

- The simplest response is loading a file from disk
- Try this:

```
$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

- Now, navigate to http://0.0.0.0:8000
  - Or http://localhost:8000



Directory listing for /

- README
- test.py

How does a server respond?

- How does `SimpleHTTPServer` work?
- Pseudo code

```
while not shutdown_request:
  if request:
    with open(request.filename) as fh:
      content = fh.read()
      copy(content, request.client)
```

Python/Flask quick start

- With our simple HTTP server example, we couldn't change the content of the files, only provide copies of them
- What if we want to produce different pages each time the web page is produced?
- Python/Flask is a library for dynamically creating web pages
- Install:

```
pip install flask
```

Python/Flask quick start

- Obligatory Hello World

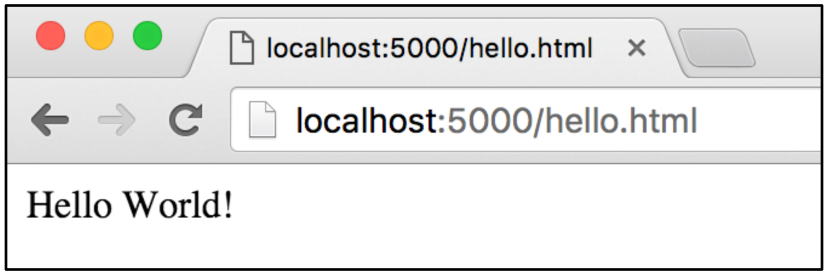
```
# hello.py

from flask import Flask
app = Flask(__name__)

@app.route('/hello.html')
def hello_world():

    return 'Hello World!'


if __name__ == '__main__':

    app.run()
```

Python/Flask quick start

- Now, run it

```
$ python hello.py
 * Running on http://127.0.0.1:5000/  (Press CTRL+C to
quit)
```

- And browse to the web page it serves



Hello World!

- NOTE: `localhost` == `127.0.0.1`

Python/Flask quick start

- What is this code doing?

```
app = Flask(__name__)
# ...
def hello_world():
    return 'Hello World!'
# ...
app.run()
```

- A lot like our previous `SimpleHTTPServer` example

```
while True:
  if request.url == '/hello.html':
    content = hello_world()
    copy(content, request.client)
```

Python/Flask quick start

- Let's produce different pages each time

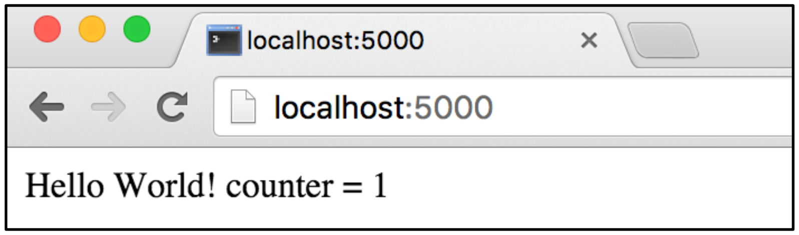
```
# hello.py
from flask import Flask
app = Flask(__name__)

@app.route('/hello.html')
def hello_world():
    hello_world.counter += 1
    return "Hello World! counter = {}".format(
      hello_world.counter)

hello_world.counter = 0 #outside function

if __name__ == '__main__':
    app.run()
```

Python/Flask quick start

- Run the new code

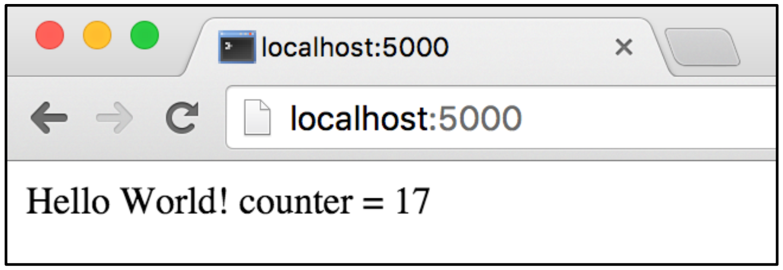
```
$ python hello.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C
to quit)
```

- Browse



Hello World! counter = 1

- Hit "refresh" a few times



Hello World! counter = 17

What's the point

- What's the point? Who cares about counters?
- Example: everybody who goes to http://facebook.com sees something different
- This is how

Office hours and contact

- Professor office hours 2-3 pm, before lecture
  - Whoever lectures that day does office hours
- To contact the staff
  - eecs485staff@umich.edu
- DeOrio:
  - 2705 BBB
  - awdeorio@umich.edu
- Cafarella:
  - 4709 BBB
  - michjc@umich.edu