Server-Side Web Programming: CGI (Part 2)

Robert M. Dondero, Ph.D.

Princeton University

Objectives

- We will cover…
  - CGI using the HTTP GET method (review)
  - CGI using the HTTP POST method
  - The GET method vs. the POST method

Agenda

- **CGI: GET Method (review)**
- CGI: POST Method
- GET vs. POST

CGI: GET Method (Review)

- **Common Gateway Interface (CGI)**
  - The oldest and simplest way to implement a Web app that generates content dynamically

CGI: GET Method (Review)

- To command browser to issue a HTTP **GET** request:
  - (1) Enter URL at top of browser
  - (2) Click on page link
  - (3) Submit a form whose method="get"

CGI: GET Method (Review)

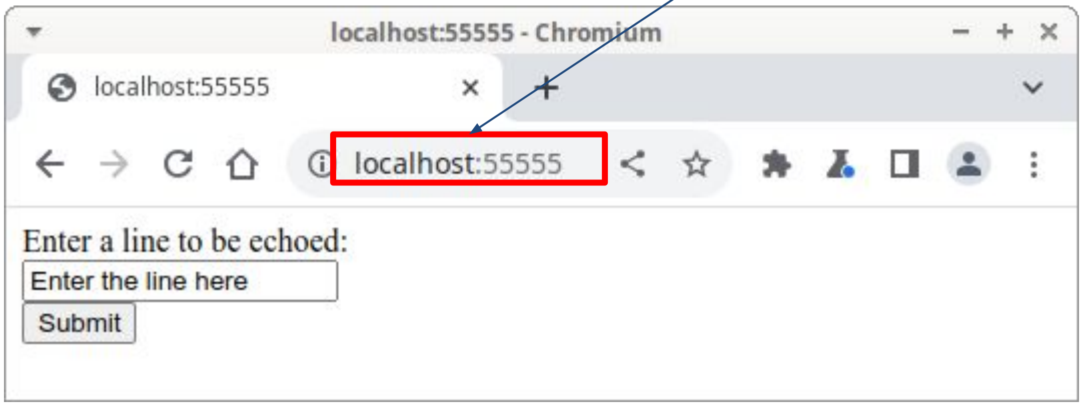
- See **EchoGet** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

CGI: GET Method (Review)

- See **<u>EchoGet</u>** app (cont.)

Default protocol is HTTP
Default page is index.html

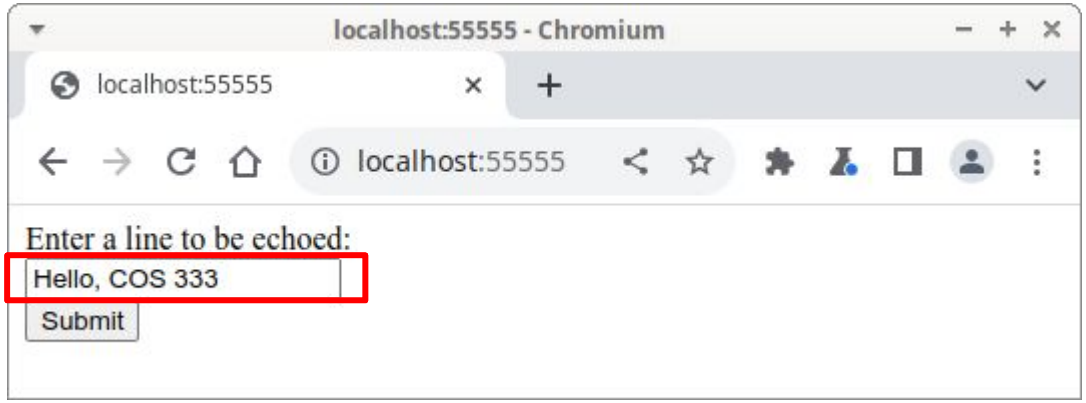


Enter a line to be echoed:

Enter the line here

CGI: GET Method (Review)

- See **EchoGet** app (cont.)

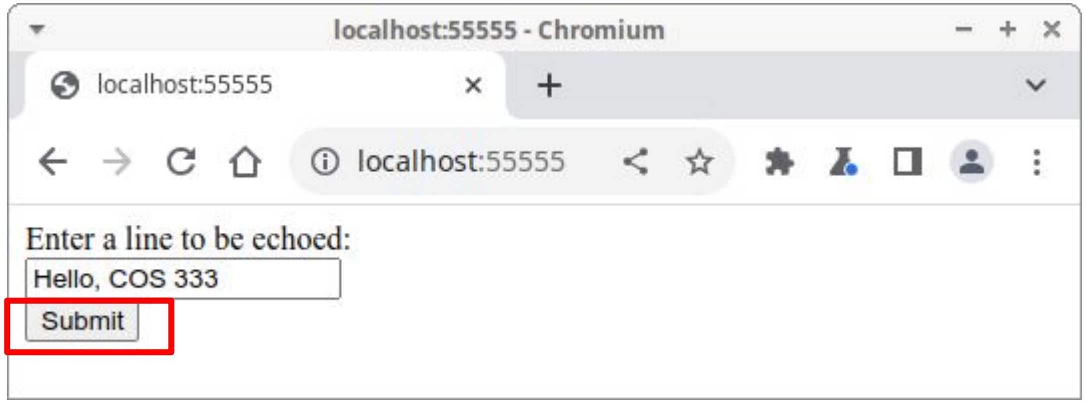


Enter a line to be echoed:

Hello, COS 333

CGI: GET Method (Review)

- See **EchoGet** app (cont.)

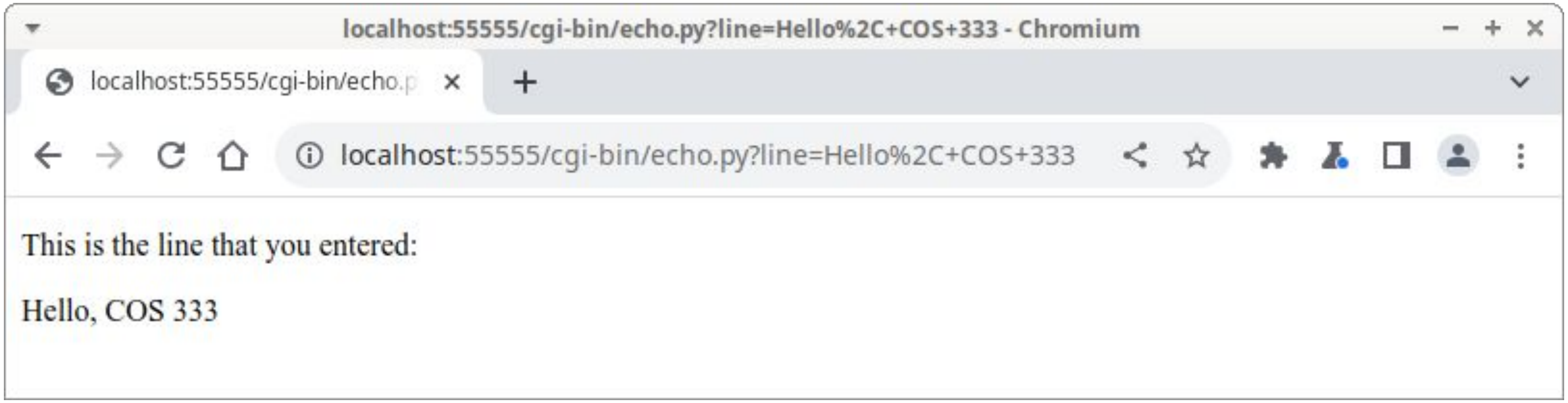


Enter a line to be echoed:

Hello, COS 333

CGI: GET Method (Review)

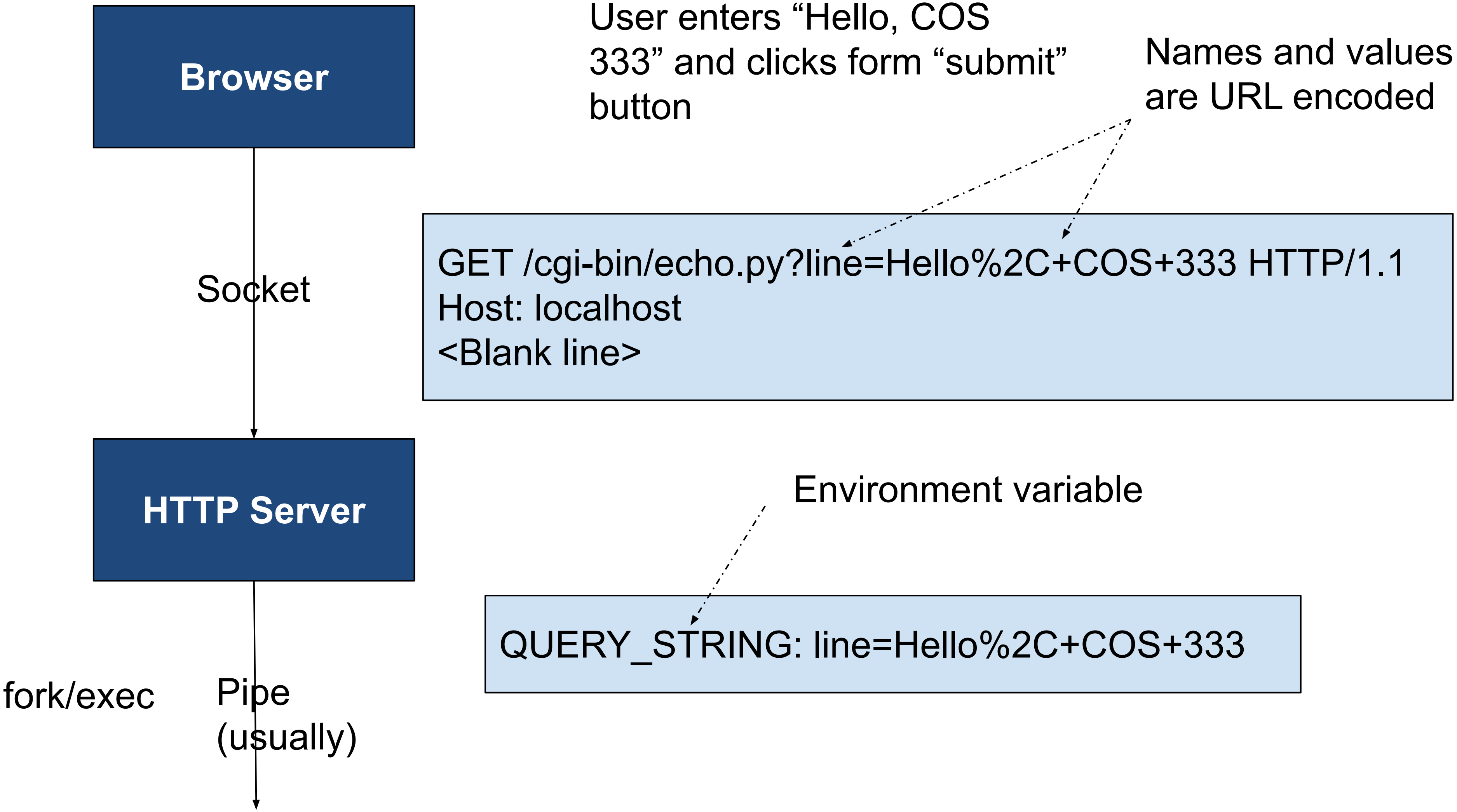
- See **EchoGet** app (cont.)


localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333 - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333

This is the line that you entered:

Hello, COS 333

CGI: GET Method (Review)



Browser

User enters “Hello, COS 333” and clicks form “submit” button

Names and values are URL encoded

Socket

```
GET /cgi-bin/echo.py?line=Hello%2C+COS+333 HTTP/1.1
Host: localhost
<Blank line>
```

HTTP Server

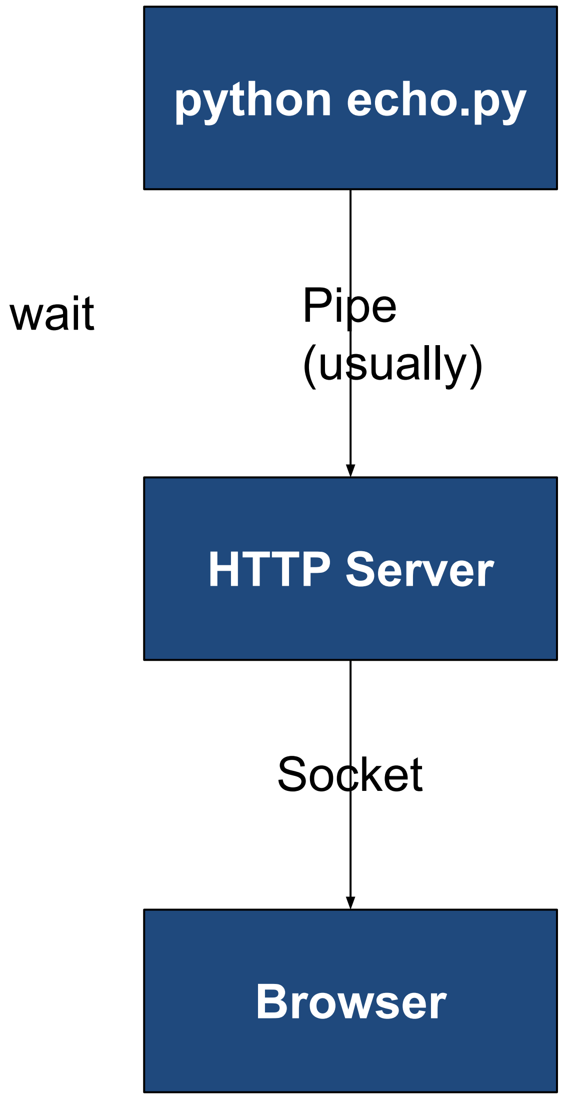
Environment variable

```
QUERY_STRING: line=Hello%2C+COS+333
```

fork/exec

Pipe (usually)

CGI: GET Method (Review)



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

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

Uses QUERY_STRING
Writes to stdout

```
HTTP/1.1 200 OK
…
Content-Type: text/html;charset=utf-8

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

There are some additional headers

CGI: GET Method (Review)

- See **EchoGet** app (cont.)
  - **runserver.py**
  - **index.html**
  - **cgi-bin/parseargs.py**
  - **cgi-bin/echo.py**

Agenda

- CGI: GET Method (review)
- **CGI: POST Method**
- GET vs. POST

CGI: POST Method

- To command browser to issue a HTTP **POST** request:
  - (1) Submit a form whose method=“post”

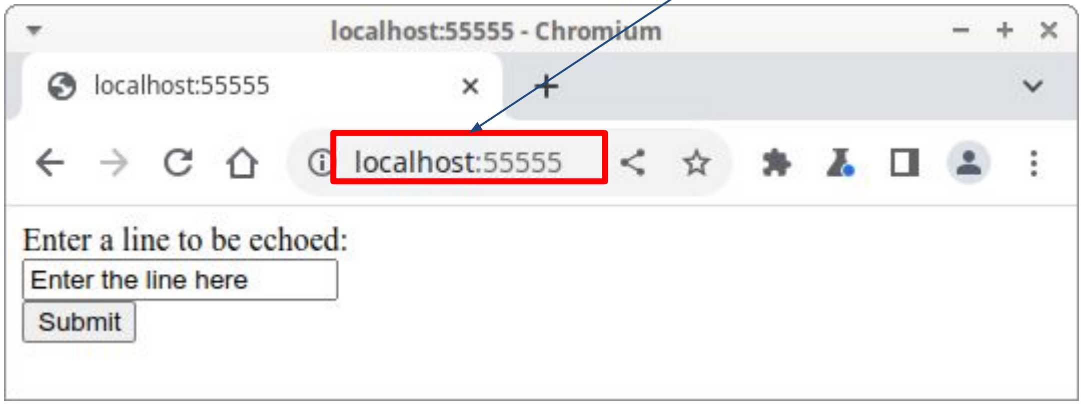
CGI: POST Method

- See **EchoPost** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

CGI: POST Method

- See **EchoPost** app (cont.)



Enter a line to be echoed:

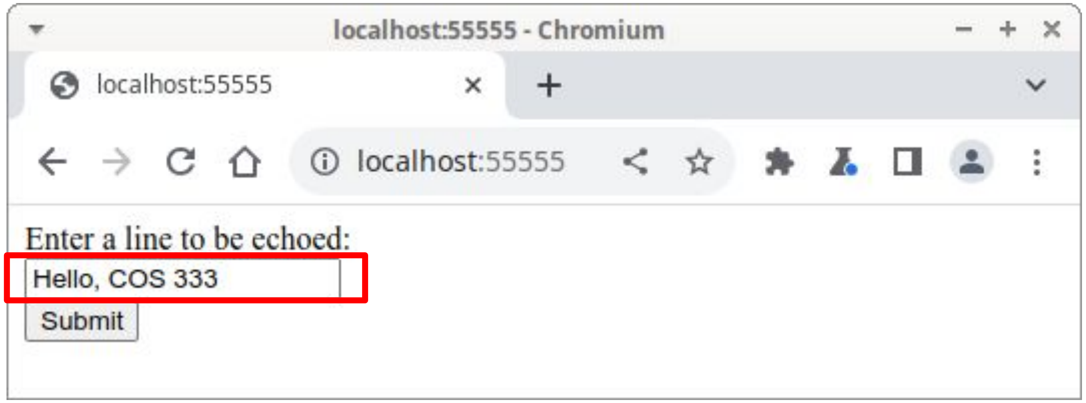
Enter the line here

Submit

Default protocol is HTTP
Default page is index.html

CGI: POST Method

- See **EchoPost** app (cont.)

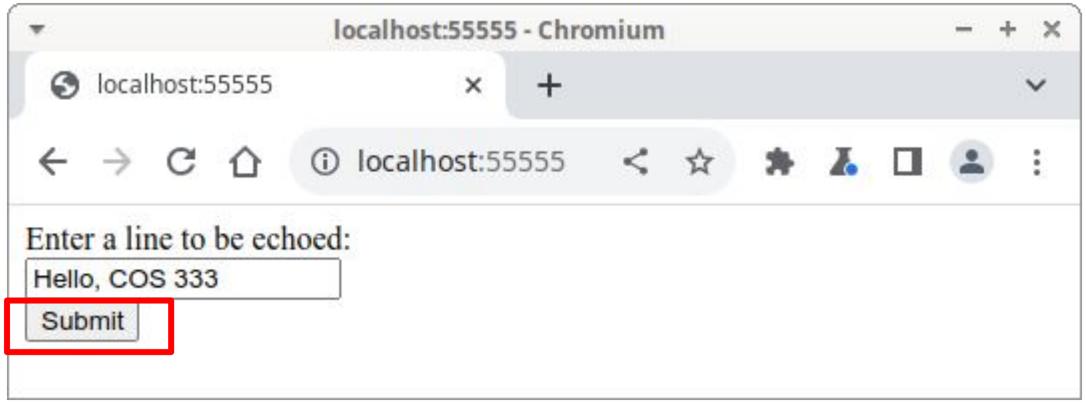


Enter a line to be echoed:

Hello, COS 333

CGI: POST Method

- See **EchoPost** app (cont.)



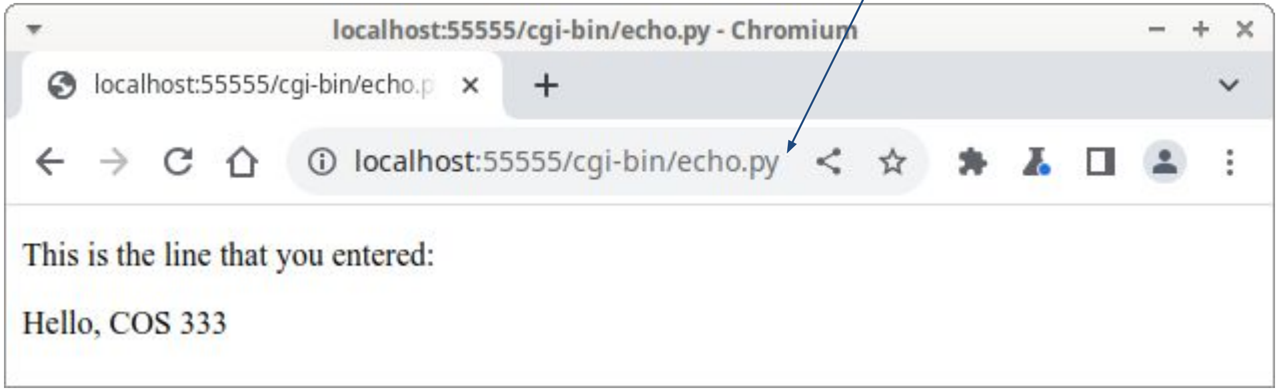
Enter a line to be echoed:

Hello, COS 333

CGI: POST Method

- See **EchoPost** app (cont.)

No name=value pairs in URL

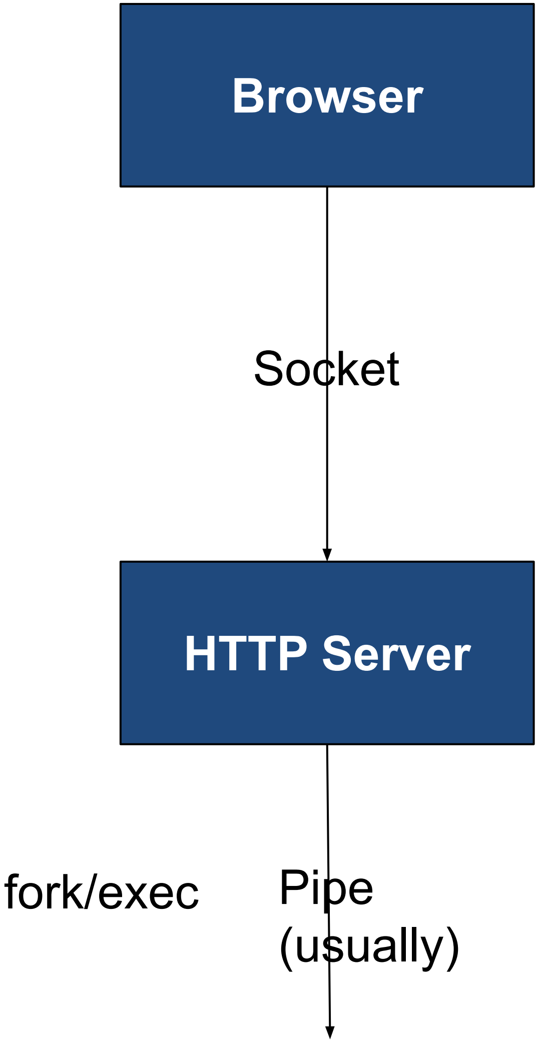


localhost:55555/cgi-bin/echo.py - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py

This is the line that you entered:

Hello, COS 333

CGI: POST Method



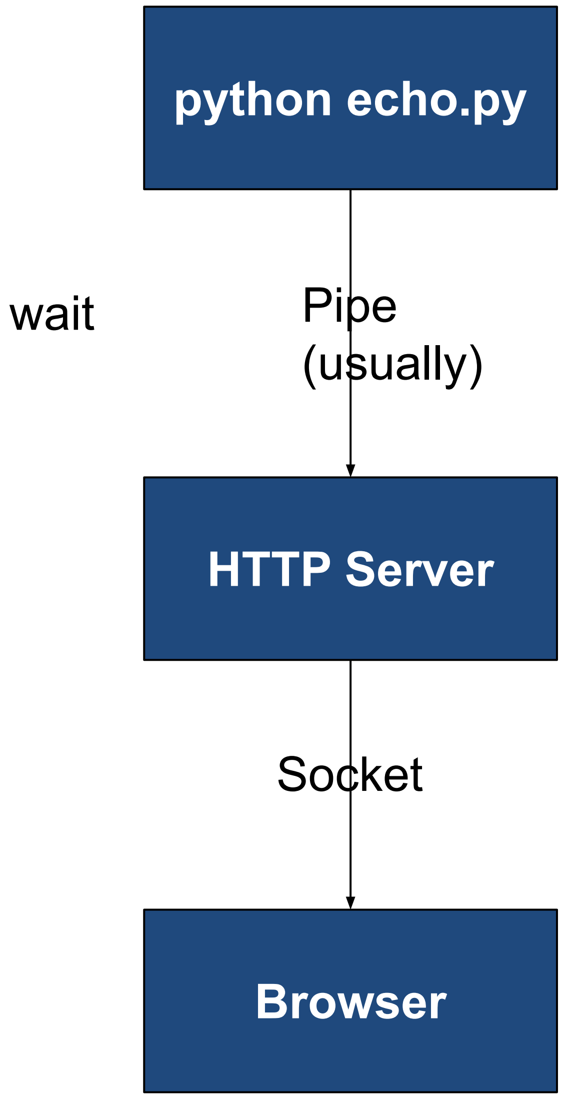
User enters “Hello, COS 333”
and clicks form “submit” button

```
POST /cgi-bin/echo.py HTTP/1.1
Host: localhost
Content-type: application/x-www-form-urlencoded
Content-length: 19

line=Hello%2C+COS+333
```

```
stdin: line=Hello%2C+COS+333
```

CGI: POST Method



python echo.py

wait

Pipe
(usually)

HTTP Server

Socket

Browser

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

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

Reads from stdin
Writes to stdout

```
HTTP/1.1 200 OK
…
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

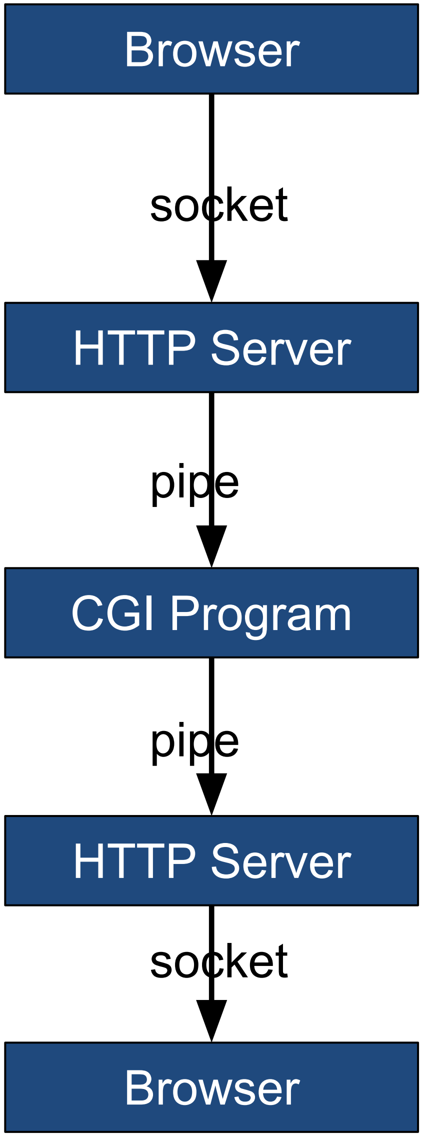
There are
some additional
headers

CGI: POST Method

- See **EchoPost** app (cont.)
  - runserver.py
  - **index.html**
  - cgi-bin/parseargs.py
  - **cgi-bin/echo.py**

CGI: POST Method

Review:



HTTP request that specifies a program and name=value pairs
GET: `name=value` pairs in request header
POST: `name=value` pairs in request body

Name=value pairs
GET: `name=value` pairs in QUERY_STRING env var
POST: `name=value` pairs in stdin

HTTP response (partial headers & complete body)
Writes to stdout

HTTP response (complete headers & complete body)

Agenda

- CGI: GET Method (review)
- CGI: POST Method
- **GET vs. POST**

GET vs. POST

- When to use **GET**?
- When to use **POST**?

GET vs. POST

- Debugging considerations
  - Easy to send GET requests from browser with arbitrary `name=value` pairs
  - Use **GET** when debugging

GET vs. POST

- Technical considerations
  - Some HTTP servers reject long headers
  - Use **POST** when:
    - There are very many `name=value` pairs
    - Some `name=value` pairs are very long

GET vs. POST

- Security considerations
  - Some HTTP servers store request URLs in log files
  - Use **POST** when:
    - Some `name=value` pairs must be kept secure
    - E.g., passwords

GET vs. POST

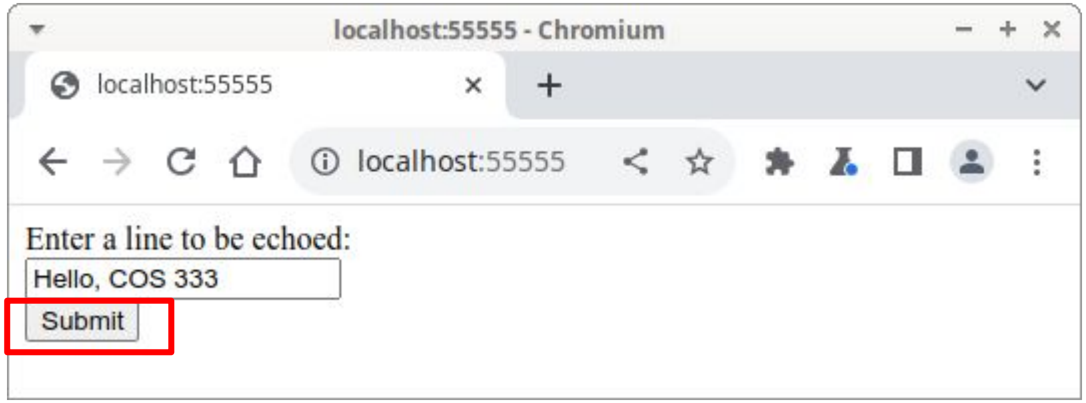
- Convention (informally):
  - Use **GET** if the app's processing **will not** change persistent data
  - Use **POST** if the app's processing **will** change persistent data

GET vs. POST

- Browser has sent **GET** request =>
  - Browser assumes request **will not** change persistent data
  - Browser **does not warn** about page reload
- Browser has sent **POST** request =>
  - Browser assumes request **will** change persistent data
  - Browser **warns** about page reload

GET vs. POST

- See **EchoGet** app

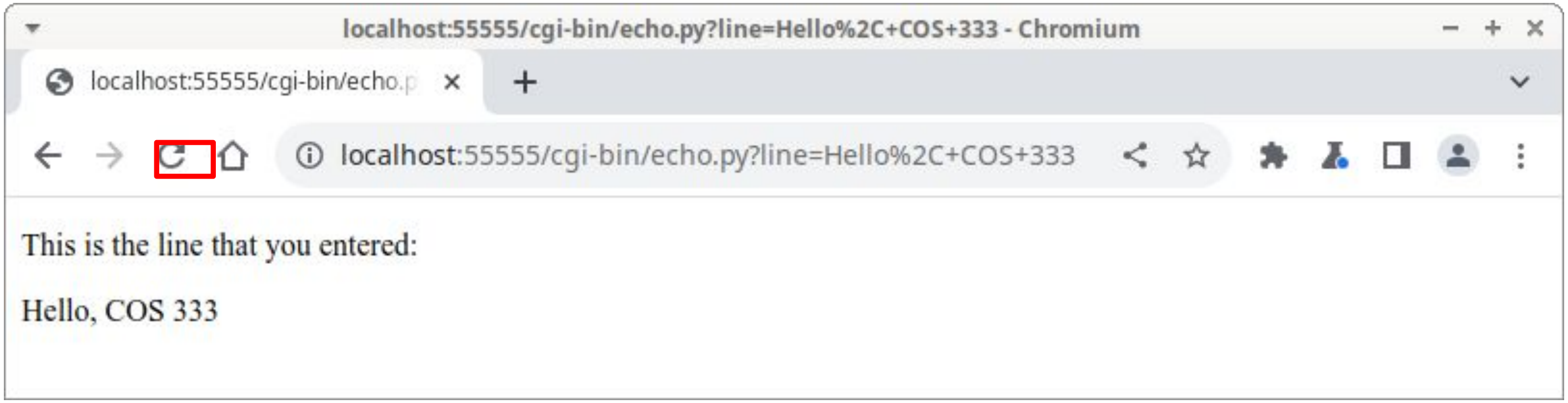


Enter a line to be echoed:

Hello, COS 333

GET vs. POST

- See **EchoGet** app (cont.)



localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333 - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333

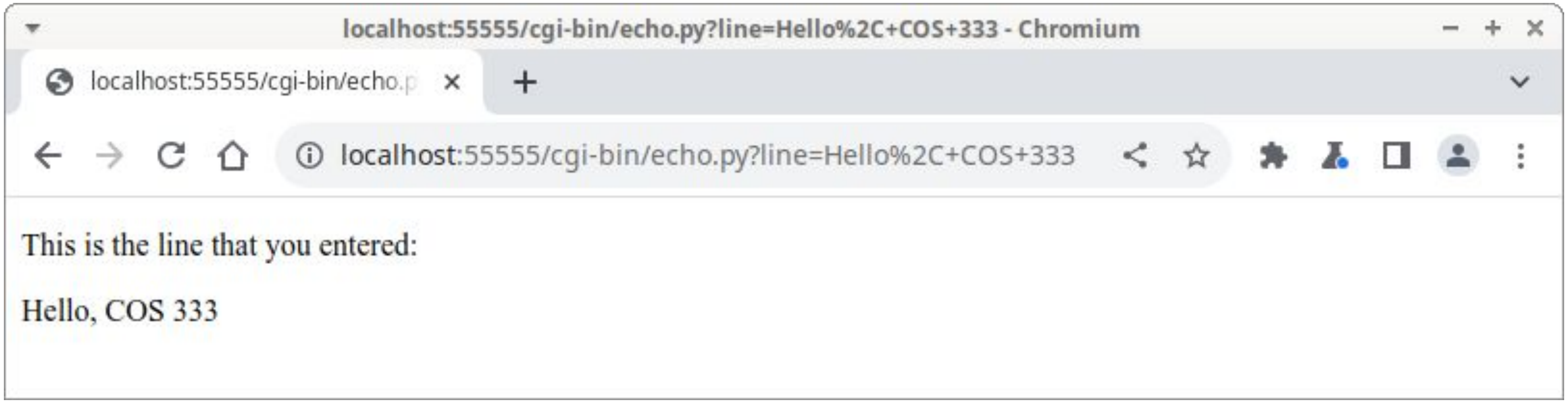
This is the line that you entered:

Hello, COS 333

Reload this page

GET vs. POST

- See **EchoGet** app (cont.)



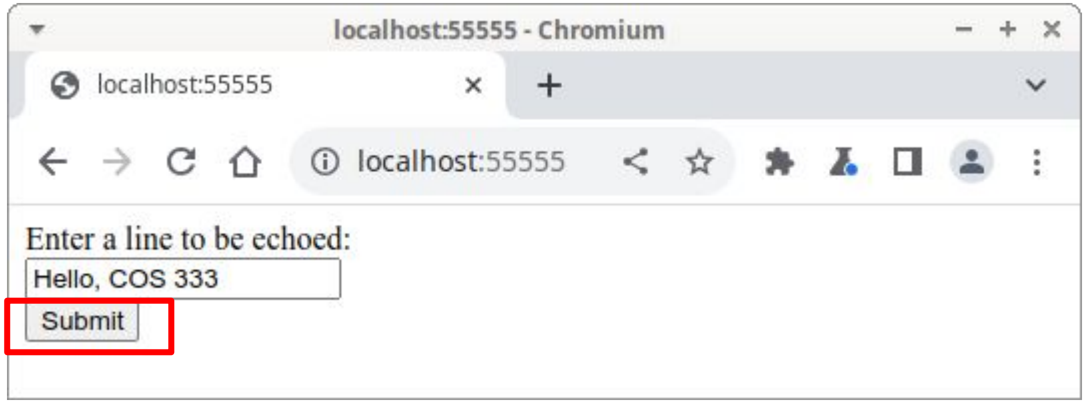
localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333 - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333

This is the line that you entered:

Hello, COS 333

GET vs. POST

- See **EchoPost** app

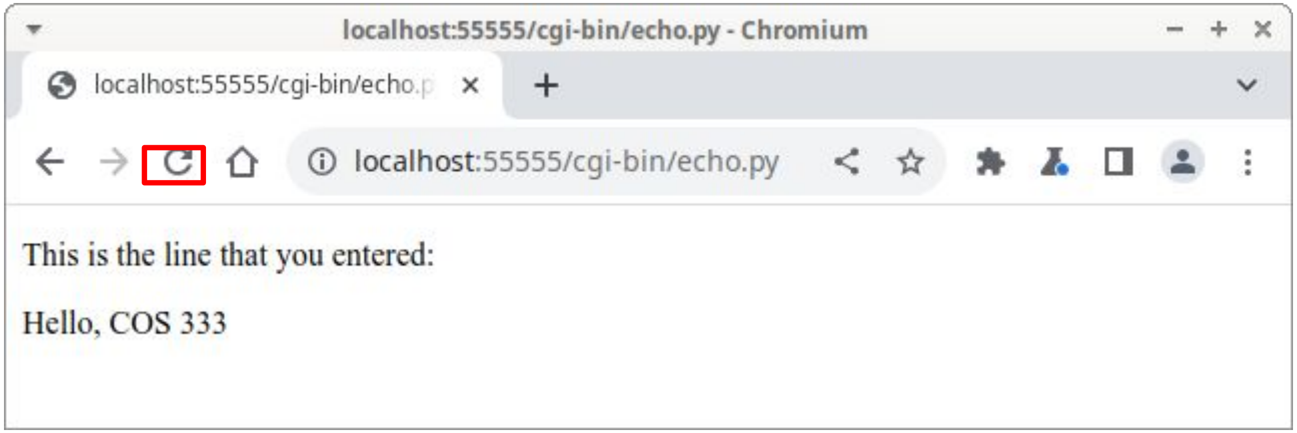


Enter a line to be echoed:

Hello, COS 333

GET vs. POST

- See **EchoPost** app (cont.)



localhost:55555/cgi-bin/echo.py - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py

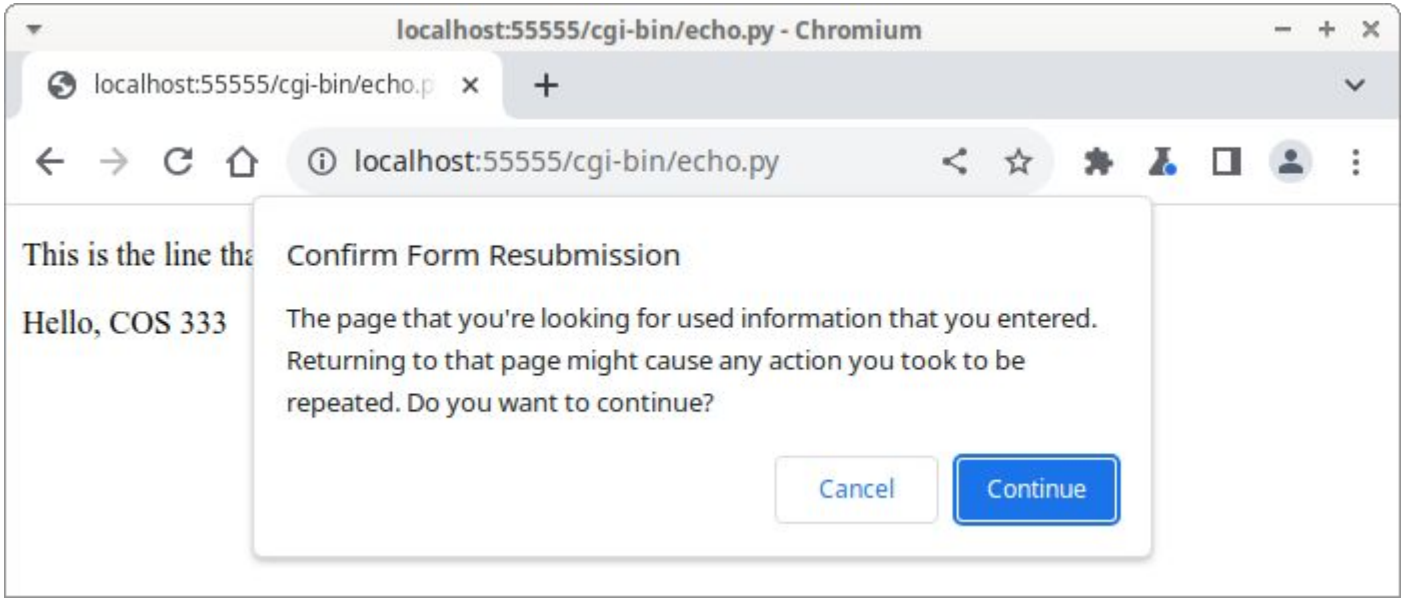
This is the line that you entered:

Hello, COS 333

Reload this page

GET vs. POST

- See **EchoPost** app (cont.)



localhost:55555/cgi-bin/echo.py - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py

This is the line tha

Hello, COS 333

Confirm Form Resubmission

The page that you're looking for used information that you entered. Returning to that page might cause any action you took to be repeated. Do you want to continue?

Aside: HTTP Methods

- **HTML** uses:
  - GET, POST
- **HTTP** defines:
  - GET, POST
  - PUT, PATCH, DELETE, CONNECT, HEAD, OPTIONS, TRACE
    - Sometimes used with AJAX requests (described later)

Aside: HTTP Methods

Method	Conventional Purpose	Idempotent?
GET	Retrieve resource	Yes
POST	Create resource	No
PUT	Replace resource	Yes
PATCH	Update resource	No
DELETE	Delete resource	Yes

Others: **CONNECT**, **HEAD**, **OPTIONS**, **TRACE**

See Chapter 14 of *Flask Web Development*
by Miguel Grinberg

Lecture Summary

- In this lecture we covered:
  - CGI using the HTTP GET method (review)
  - CGI using the HTTP POST method
  - The GET method vs. the POST method