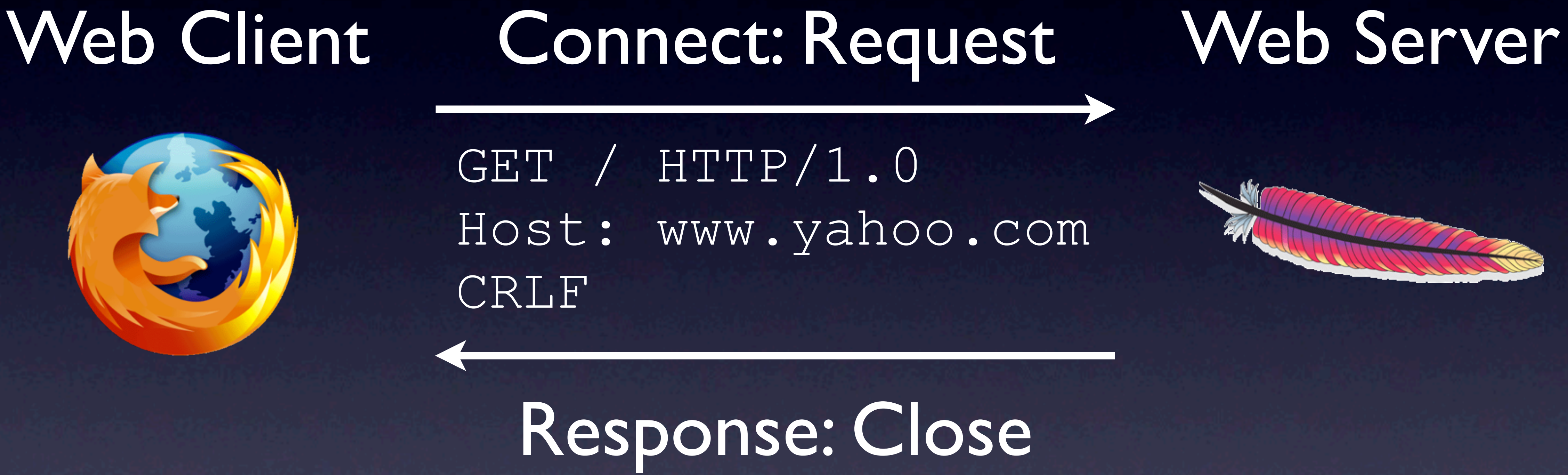
Building a Simple Web Proxy

A Brief History of HTTP

- Mar 1989 - "Information Management: A Proposal"
- Oct 1990 - "WorldWideWeb" coined
- Oct 1994 - W3C founded
- May 1996 - RFC 1945 (HTTP 1.0)
- June 1999 - RFC 2616 (HTTP 1.1)

Anatomy of HTTP 1.0



Web Client

Connect: Request

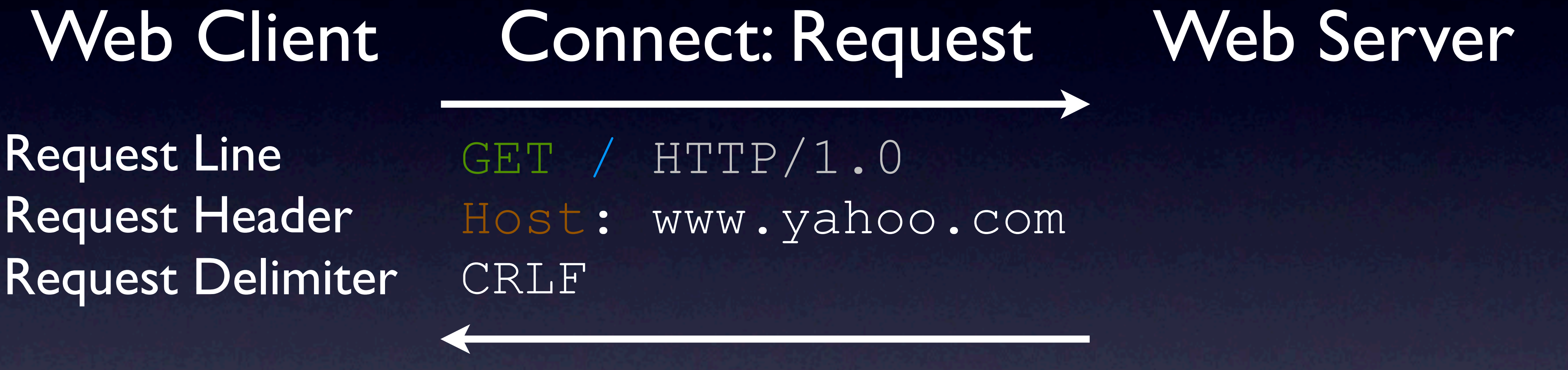
Web Server

```
GET / HTTP/1.0
Host: www.yahoo.com
CRLF
```

Response: Close

```
HTTP/1.0 200 OK
Date: Tue, 16 Feb 2010 19:21:24 GMT
Content-Type: text/html;
CRLF
<html><head>
<title>Yahoo!</title>
```

Anatomy of HTTP 1.0



Web Client — Connect: Request → Web Server

Request Line

```
GET / HTTP/1.0
```

Request Header

```
Host: www.yahoo.com
```

Request Delimiter

```
CRLF
```

← Response: Close

Response Status

```
HTTP/1.0 200 OK
```

Response Header

```
Date: Tue, 16 Feb 2010 19:21:24 GMT
Content-Type: text/html;
```

Response Delimiter

```
CRLF
```

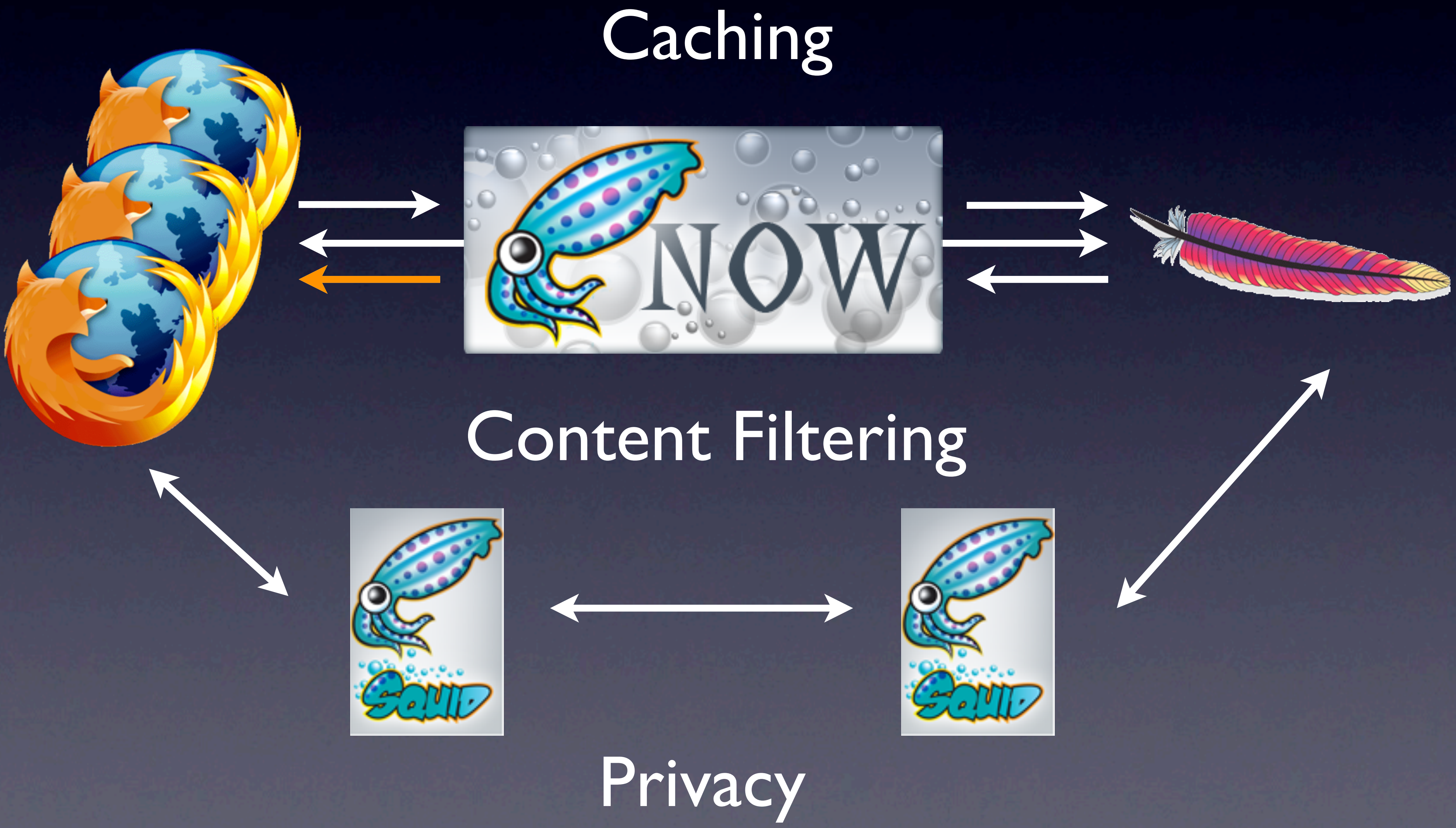
Response Body

```
<html><head>
<title>Yahoo!</title>
```

HTTP 1.1 vs 1.0

- Additional Methods (PUT, DELETE,TRACE, CONNECT + GET, HEAD, POST)
- Additional Headers
- Transfer Coding (chunk encoding)
- Persistent Connections (content-length matters)
- Request Pipelining

Why Use a Proxy?



Caching

Content Filtering

Privacy

Building a Simple Web Proxy

- Forward client requests to the remote server and return response to the client
- Handle HTTP 1.0 (GET)
- Single-threaded, non-caching web proxy
- ./proxy <port>

Handling Requests

- What you need from a client request: host, port, and URI path

  ```
  GET http://www.princeton.edu:80/ HTTP/1.0
  ```

  or

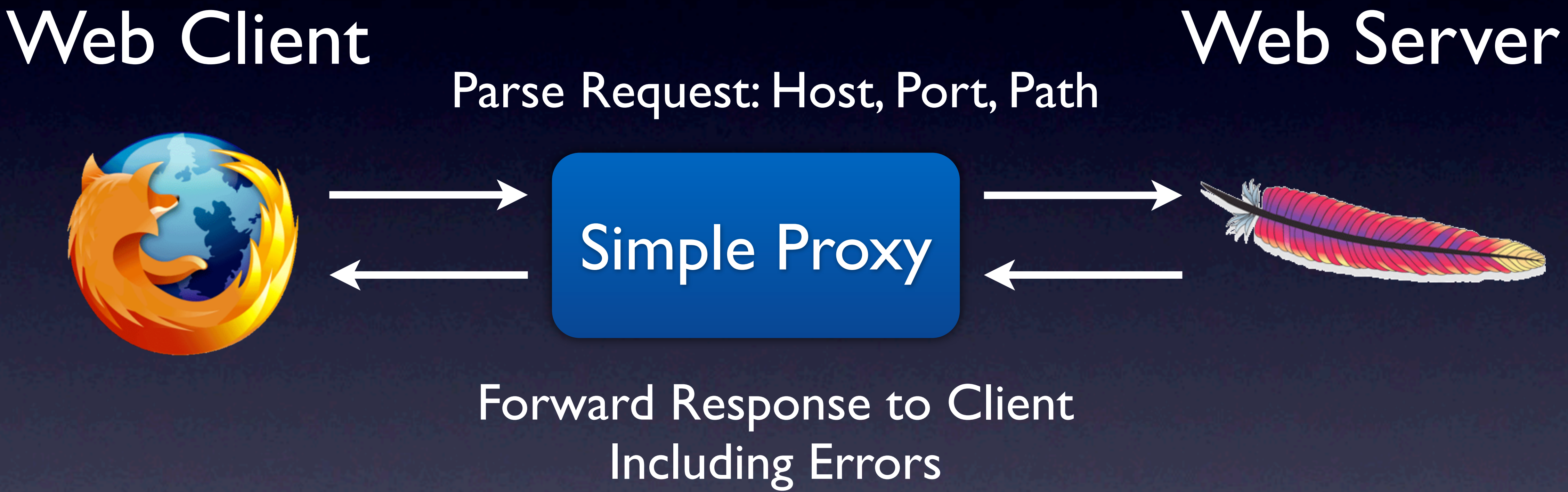
  ```
  GET / HTTP/1.0
  Host: www.princeton.edu:80
  ```

- What you send to a remote server:

  ```
  GET / HTTP/1.0
  Host: www.princeton.edu:80
  (Additional headers, if any...)
  ```

- Check request line and header format

Handling Responses



Web Client

Web Server

Parse Request: Host, Port, Path

Simple Proxy

Forward Response to Client
Including Errors

Handling Errors

- Method != GET: Not Implemented (501)
- Unparseable request: Bad Request (400)
- Keep parsing simple: no need for regex
- Postel’s law: Be liberal in what you accept, and conservative in what you send
  convert HTTP 1.1 request to HTTP 1.0
  convert \r to \r\n
  etc...

Testing Your Proxy

- Telnet to your proxy and issue a request

```
> ./proxy 5000
> telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.localdomain
(127.0.0.1).
Escape character is '^]'.
GET http://www.google.com/ HTTP/1.0

(HTTP response...)
```

- Direct your browser to use your proxy
- Use the supplied proxy_tester.py

Proxy Guidance

- Assignment page
- Assignment FAQ
- RFC 1945 (HTTP 1.0)
- Google, wikipedia, man pages
- Must build on Friend 010 machines
- Submission name = file name