

CSC 591

Systems Attacks and Defenses

Web Security

Alexandros Kapravelos
akaprav@ncsu.edu

(Derived from slides by Giovanni Vigna and Adam Doupe)

WorldWideWeb

- Info
- Navigate
- Document
- Edit
- Find
- Links
- Style
- Print...
- Page layout...
- Windows
- Services
- Hide
- Quit

Welcome to the Universe of HyperText

Home

Access to this information is provided as part of the [WorldWideWeb](#) project. The WWW project does not take responsibility 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 ([See information 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

[CERN Technology Interest Group\) news](#).

machine, see also the following topics:

[on this WorldWideWeb application](#)

Info

HyperMedia Browser/Editor

Version 1.0
Alpha only

An exercise 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 with the libWWW library. Bug reports to www-bug@info.cern.ch. Check the list of known bugs in the web too.

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

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

NextStep

PREF

Recycling symbol

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?

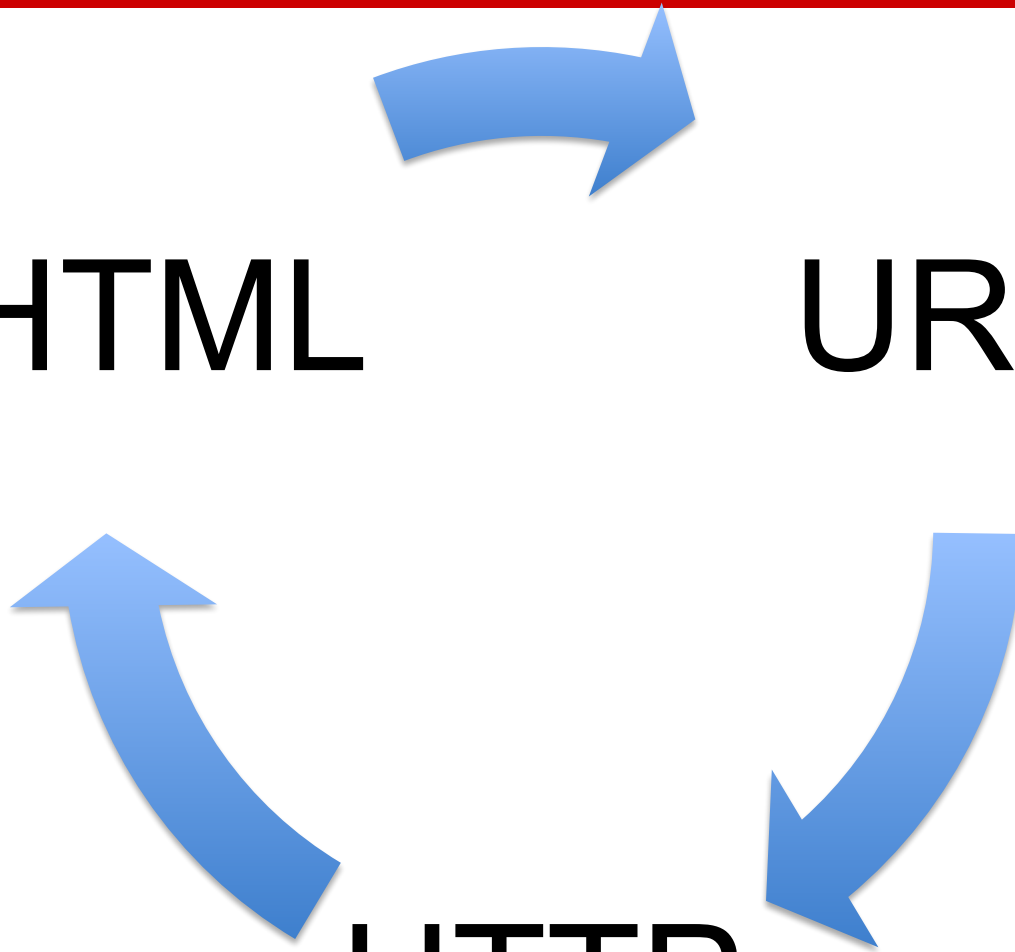
Three Central Technologies

- 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:akapprav@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`

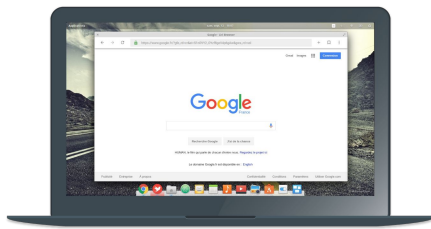
Hypertext Transport Protocol

- 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

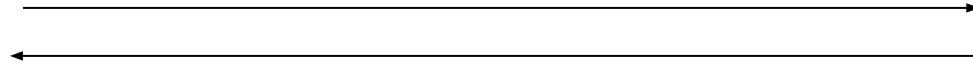
- Client
 - Opens TCP connection to the server
 - Sends request to the server
- Server
 - Listens for incoming TCP connections
 - Reads request
 - Sends response

Architecture



Client

HTTP Request

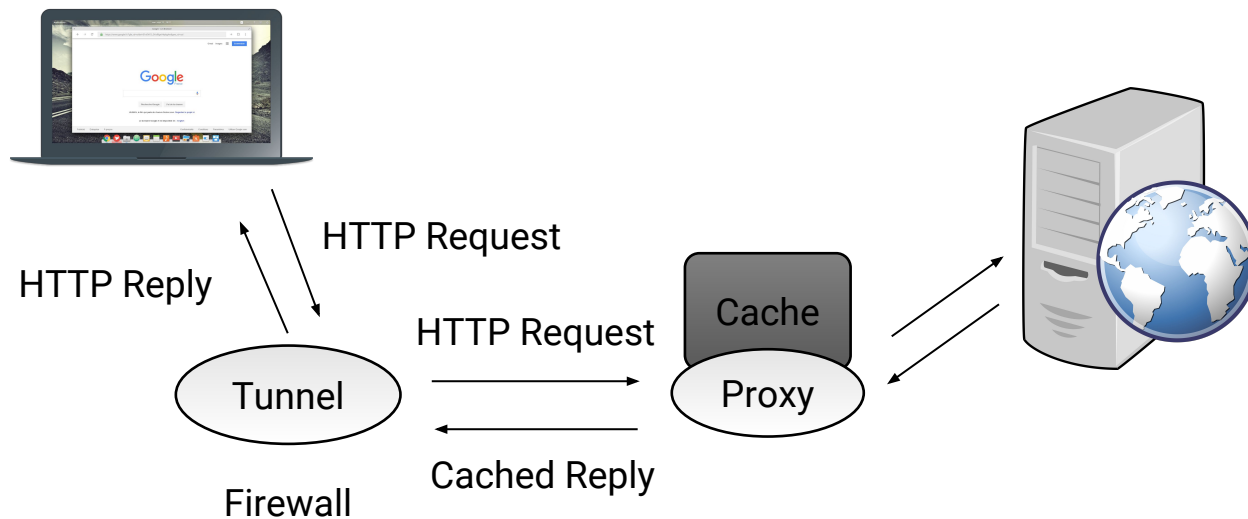


HTTP Reply

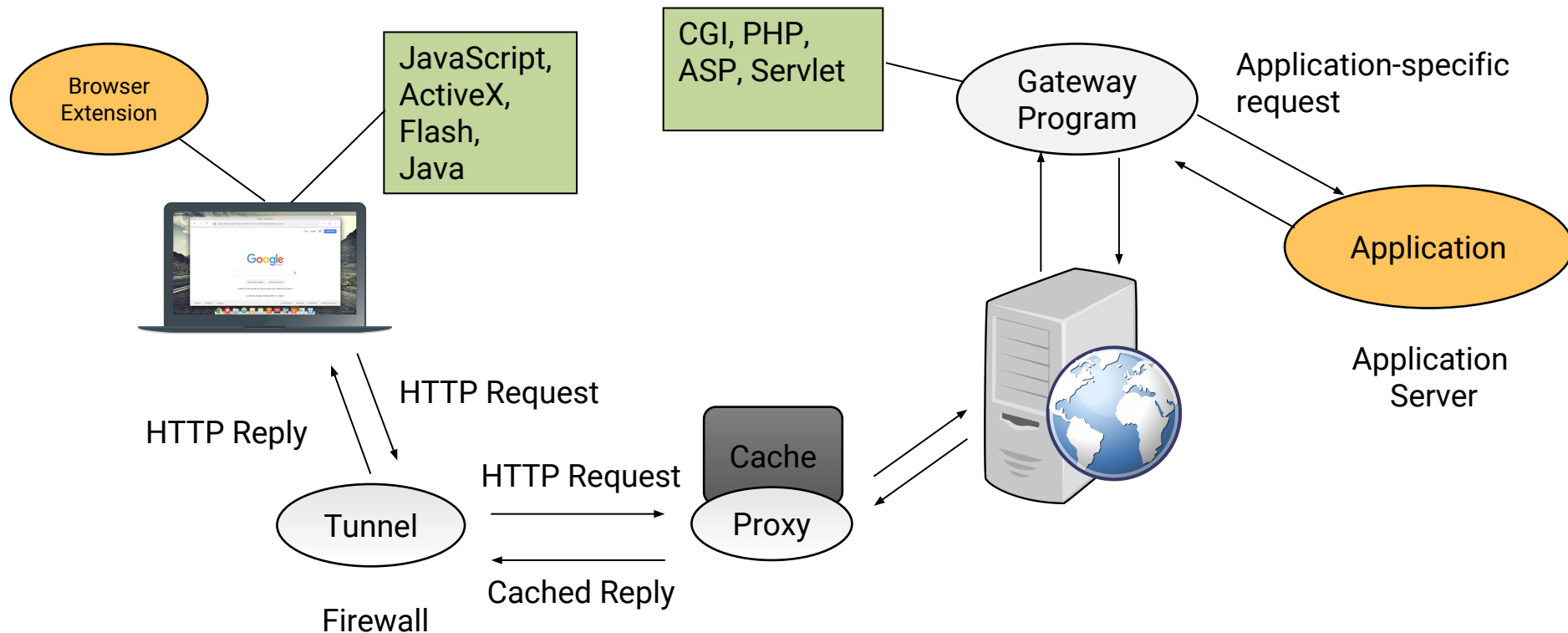


Server

Architecture



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 QWxhZGRpbjpvYVUyIHNLc2FtZQ==`

- 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):
 - ``

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

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

[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
- `&`
- `&`
- `&`
- `&`

HTML – Character References

Example

- é
- é
- é
- é

HTML – Character References

Example

- Why must ‘<’ be encoded as a character reference?
- <
- <
- 0
- 0

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">`

- 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>
```

<input type="text" value="bar"/>	<input type="text"/>	<input type="text"/>	<input type="submit" value="Submit"/>
<input type="text" value="Wolf Pack"/>	<input type="text" value="csc 591"/>	<input type="text" value="A+"/>	<input type="submit" value="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+	Submit
-----------	---------	----	--------

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 send 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=0qGXMZZhmeyihyKi; expires=Sun, 15-Jan-2017 16:11:12 GMT; path=/; domain=.google.com`
- Set-Cookie:
`NID=67=bs1lLyrXtfdUj79I1cuqR7_MWEsyNdLWU_FpGKw1WR9QpEzi3UrVV2UG06LBW3sJNk9m1LcYIJns3PG3NUu-M3pT9qD-V4F8oyyJ_UJnCGKDUDGb11L9Ha8KGufv0MUv; 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=0qGXMZZh  
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=bs1lLyrXtfdUj79I1cuqR7_MWEs  
yNdLWU_FpGKw1WR9QpEzi3UrVV2UG06LBW  
3sJNk9m1LcYIJns3PG3NUu-M3pT9qD-V4F  
8oyyJ_UJnCGKDUDGb11L9Ha8KGufv0MUv;  
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

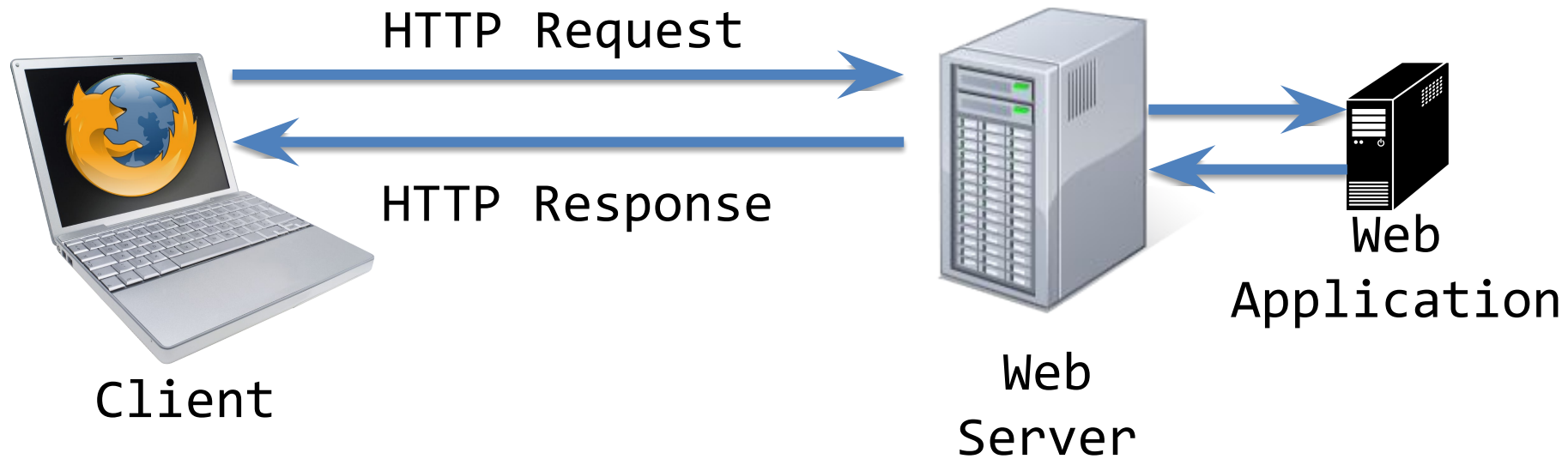
Modern Sessions

- 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



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

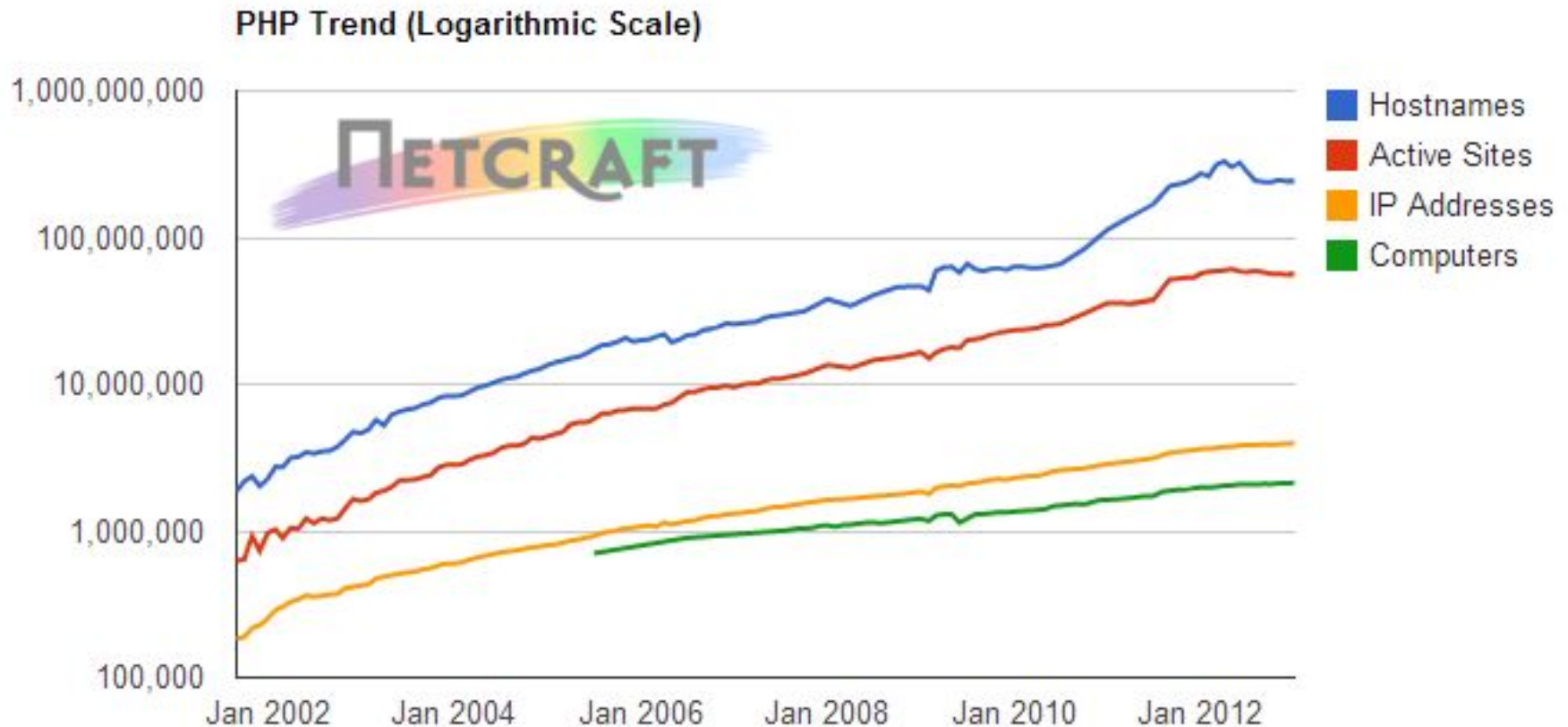
Web Application Frameworks

- 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



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
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']}";
```

PHP – Dynamic include/require

```
<?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
/**
 * 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);
```

PHP – Variable Variables

```
<?php
```

```
$a = 'hello';
```

```
$$a = 'world';
```

```
echo "$a $hello";
```

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

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
// 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";
}
?>
```

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'];
}
```

HTML

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



[Yahoo! Deutschland](#) [CLICK HERE TO VISIT THE STARS](#) [Yahoo! LOS ANGELES](#) [Weekly Picks](#)

[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](#)



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.



[Click here for advertising information - reach millions every month!](#)

Search and Display the Results

Submit

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



[Download free demo versions of AltaVista Technology software](#)



[[Creative](#)][[Search](#)][[Humor](#)][[Email](#)]



Search the web using Google!

Google Search

I'm feeling lucky

Special Searches

[Stanford Search](#)

[Linux Search](#)

[Help!](#)

[About Google!](#)

[Company Info](#)

[Google! Logos](#)

Get Google!

updates monthly:

your e-mail

[Archive](#)

Copyright ©1998 Google Inc.

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">
```



This page says:

Please enter your name below.

Cancel

OK



This page says:

Please enter your name below.

Cancel

OK

test.html

×

Alexandros

← → ↻ ⓘ file:///tmp/test.html ☆

🔍 📄 🛡️ ⋮

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 ` ` 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} ⓘ  
  ► 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
```

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
```

✕

localhost:8000/test.php ✕

Alexandros

☐

⬅ ➡ ↺ ⓘ localhost:8000/test.php ☆ 🔗 ↕ 🛡 ⋮

Student:

Class:

Grade:

Submit

localhost:8000/test.php

localhost:8000/test.php

Student:

Class:

Grade:

Submit

localhost:8000/test.php x

Alexandros

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 x

Alexandros

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

☆

Error, did not fill out all the forms

Student:

Class:

Grade:

localhost:8000/test.php x

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 ✕

Alexandros

⬅ ➡ ↺ ⓘ 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:

✕

localhost:8000/test.php ✕

Alexandros

← → ↻ ⓘ 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>
```

✕

localhost:8000/test.php ✕

Alexandros

☐

⬅ ➡ ↺ ⓘ localhost:8000/test.php ☆ 🔗 ↕ 🛡 ⋮

Student:

Class:

Grade:

Submit

✕

localhost:8000/test.php ✕

Alexandros

📄

⬅ ➡ ↺ ⓘ localhost:8000/test.php ☆ 🔗 📶 🛡 ⋮

Student:

Class:

Grade:

localhost:8000/test.php

localhost:8000/test.php

Student:

Class:

Grade:

Submit

localhost:8000 says:

Error, must fill out all the form

OK

✕

localhost:8000/test.php ✕

Alexandros

📄

⬅ ➡ ↺ ⓘ localhost:8000/test.php ☆ 🔗 ↕ 🛡 ⋮

Student:

Class:

Grade:

localhost:8000/test.php

localhost:8000/test.php

Student:
Class:
Grade:

localhost:8000 says:

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

OK

✕

localhost:8000/test.php ✕

Alexandros

📄

⬅ ➡ ↺ ⓘ localhost:8000/test.php ☆ 🔗 ↕ 🛡 ⋮

Student:

Class:

Grade:

✕

localhost:8000/test.php ✕

Alexandros

← → ↻ ⓘ localhost:8000/test.php?student=admin&class=CSC591&grade=B&submit=Submit ☆

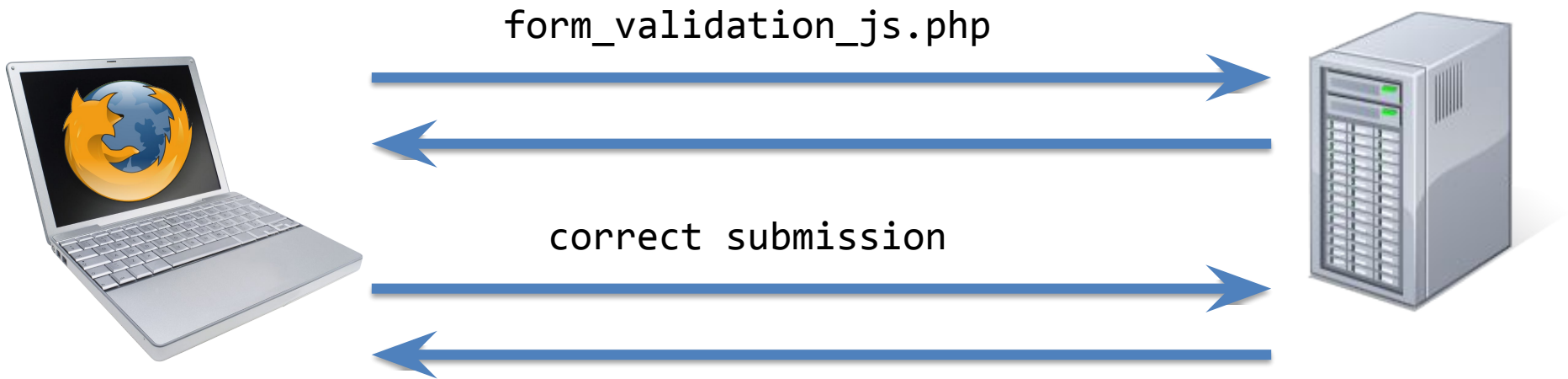
Grade successfully submitted!

Student:

Class:

Grade:

Submit



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**

Paused in debugger

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```
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 ajax.html:30 function)

Paused on a JavaScript breakpoint.

Scope Variables

Global Window

Breakpoints

ajax.html:24 if (http\_request.readyst...

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

Console Search Emulation Rendering

<top frame> Preserve log

AJAX Example

192.168.84.165/code/ajax.html

Paused in debugger

# AJAX Example

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

192.168.84.165

code

ajax.html

```
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 31, Column 1

Watch Expressions +

Call Stack Async

(anonymous ajax.html:31 function)

Paused on a JavaScript breakpoint.

Scope Variables

Global Window

Breakpoints

ajax.html:24 if (http\_request.readyState...

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

Console Search Emulation Rendering

<top frame> Preserve log

Before Request

ajax.html:30

Paused in debugger

# AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets

192.168.84.165

code

ajax.html

```
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

Watch Expressions

Call Stack

ajax.html:24  
http\_request.onreadystatechange  
change

(anonymous function) ajax.html:31

Paused on a JavaScript breakpoint.

Scope Variables

Local

new\_node: undefined  
text: undefined

Console

Search Emulation Rendering

<top frame> Preserve log

Before Request

1

ajax.html:30  
ajax.html:23

The screenshot shows a web browser window titled "AJAX Example" with the address bar displaying "192.168.84.165/code/ajax.html". The main content area shows the heading "AJAX Example" and a "Paused in debugger" notification. The Chrome DevTools interface is open, with the "Sources" panel selected. The file "ajax.html" is loaded, and the code is paused at line 33, column 1. The code in the editor is as follows:

```
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
```

The right-hand sidebar of DevTools shows the "Call Stack" with an entry for an anonymous function at line 33 of ajax.html. Below it, the "Scope Variables" panel is visible, showing a "Global" window. The "Breakpoints" panel shows two active breakpoints: one at line 24 for the condition "if (http\_request.readyState === 4)" and another at line 30 for the console log statement.

The "Console" panel at the bottom shows the output of the script, with the message "Before Request" logged at line 30 of ajax.html.

Paused in debugger

# AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```
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

Watch Expressions +

Call Stack Async

ajax.html:24  
http\_request.onreadystatechange

Paused on a JavaScript breakpoint.

Scope Variables

Local

- new\_node: undefined
- text: undefined
- this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request ajax.html:30

1 ajax.html:23

After Request ajax.html:33

2 ajax.html:23



Paused in debugger

# AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets

192.168.84.165

code

ajax.html

```
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

Watch Expressions

Call Stack

ajax.html:24  
http\_request.onreadystatechange  
change

Paused on a JavaScript  
breakpoint.

Scope Variables

Local

new\_node: undefined  
text: undefined  
this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request  
1  
After Request  
2  
3

ajax.html:30  
ajax.html:23  
ajax.html:33  
ajax.html:23  
ajax.html:23



AJAX Example

Paused in debugger

# AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```
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

Watch Expressions +

Call Stack Async

ajax.html:24  
http\_request.onreadystatechange

Paused on a JavaScript breakpoint.

Scope Variables

Local

- new\_node: undefined
- text: undefined
- this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request  
1  
After Request  
2  
3  
4

ajax.html:30  
ajax.html:23  
ajax.html:33  
ajax.html:23  
ajax.html:23  
ajax.html:23

The screenshot shows a web browser window with the title "AJAX Example" and the address bar displaying "192.168.84.165/code/ajax.html". The page content includes the heading "AJAX Example" and the text "TEST AJAX".

The Chrome DevTools interface is open, showing the "Sources" panel on the left, the "Sources" tab in the center, and the "Console" panel at the bottom.

**Sources Panel:** The file tree on the left shows the directory structure: 192.168.84.165 > code > ajax.html. The "Sources" tab is active, displaying the code for ajax.html. The code is as follows:

```
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
```

The status bar at the bottom of the Sources panel indicates "Line 24, Column 1".

**Console Panel:** The "Console" panel is open, showing the following log messages:

- Before Request
- 1
- After Request
- 2
- 3
- 4

The right side of the DevTools interface shows the "Watch Expressions", "Call Stack", "Scope Variables", and "Breakpoints" panels, all of which are currently empty or show "Not Paused".

The screenshot shows a web browser window with the title "AJAX Example" and the address bar displaying "192.168.84.165/code/ajax.html". The main content area of the browser shows the heading "AJAX Example" and the text "TEST AJAX".

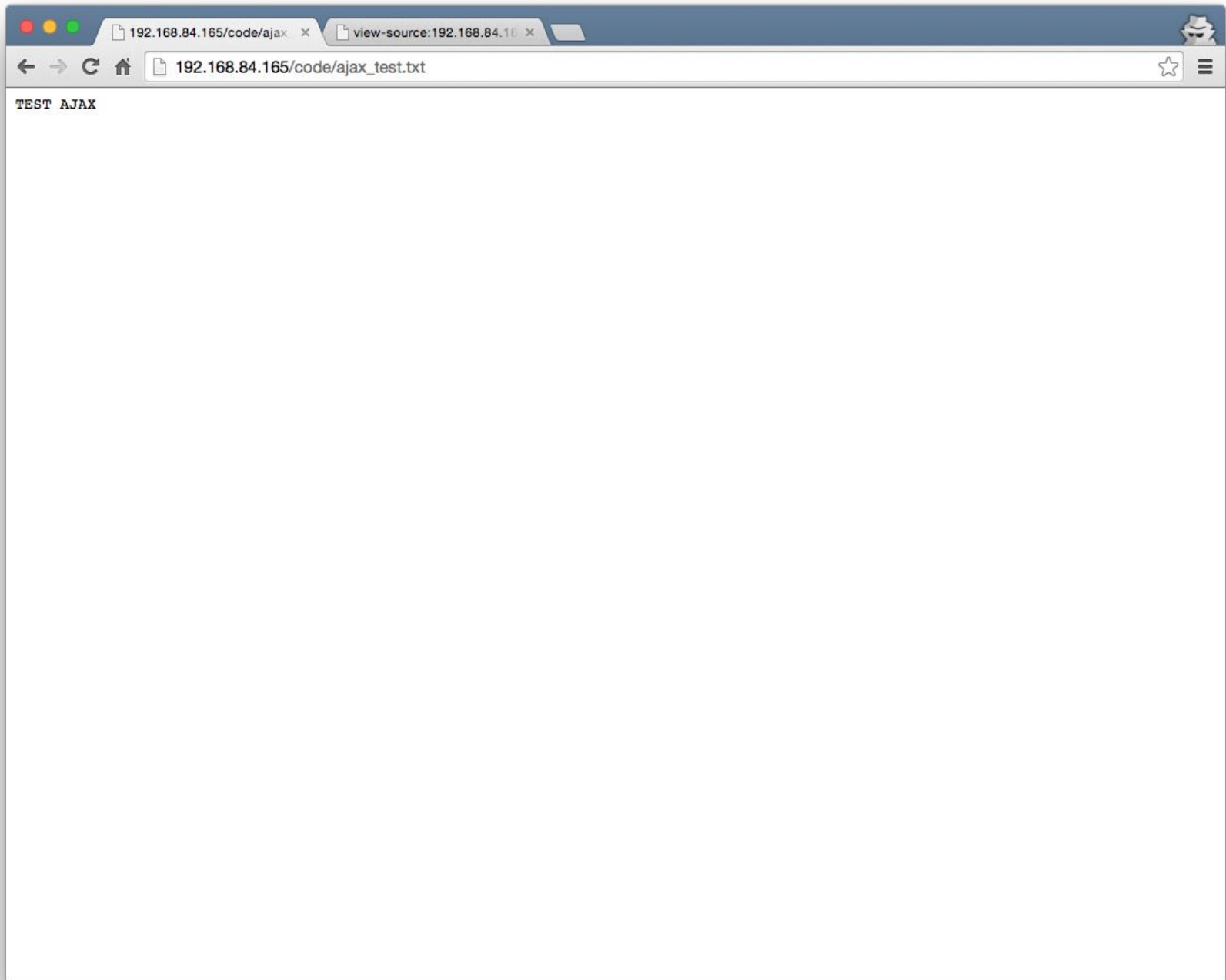
The "Network" panel is open, showing two requests:

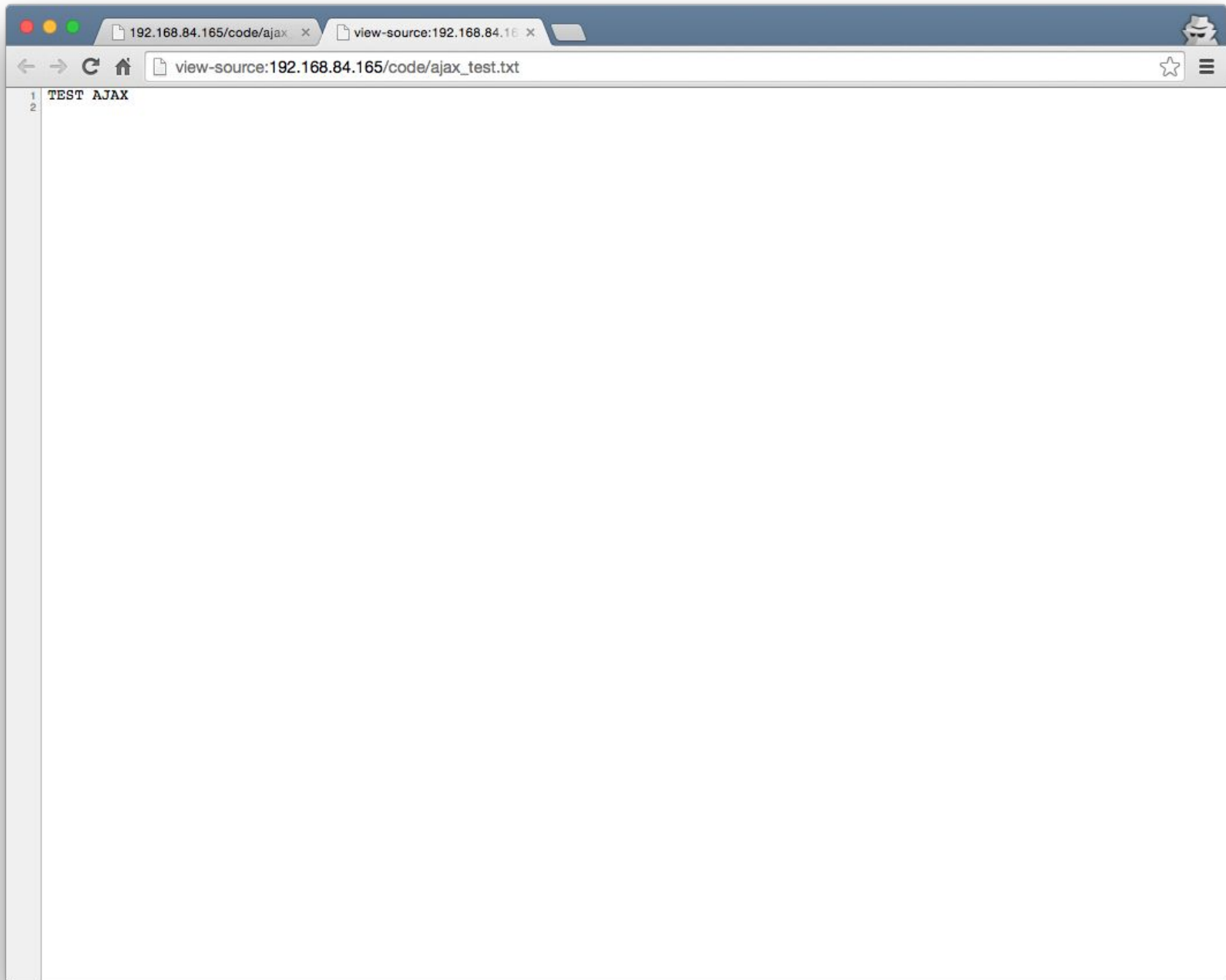
Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
ajax.html /code	GET	200 OK	text/ht...	Other	809 B 983 B	6 ms 4 ms	
ajax_test.txt /code	GET	304 Not M...	text/pl...	ajax.html:32 Script	177 B 10 B	4 ms 3 ms	

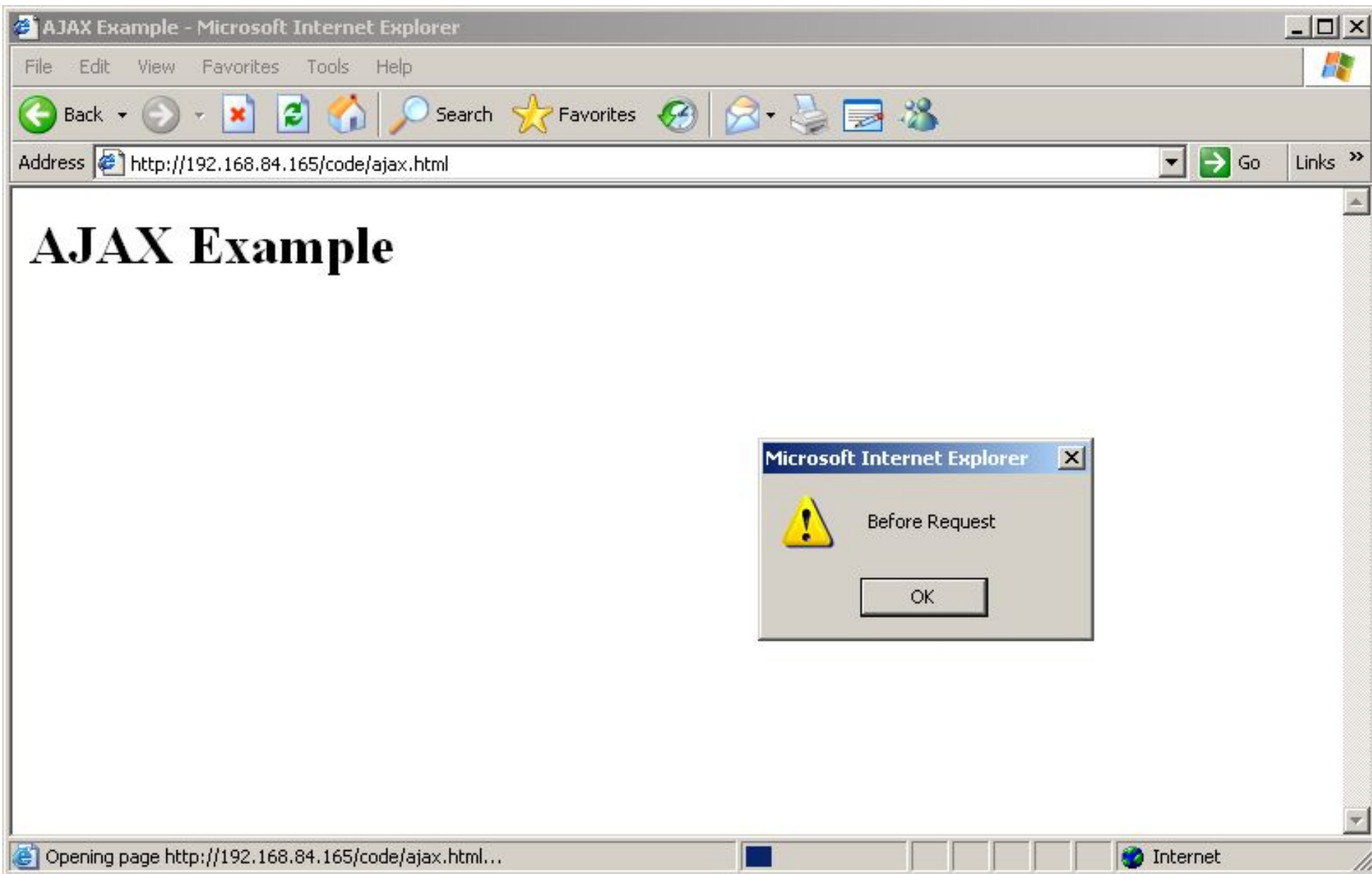
Summary: 2 requests | 986 B transferred | 2.94 s (load: 4.52 s, DOMContentLoaded: 4.52 s)

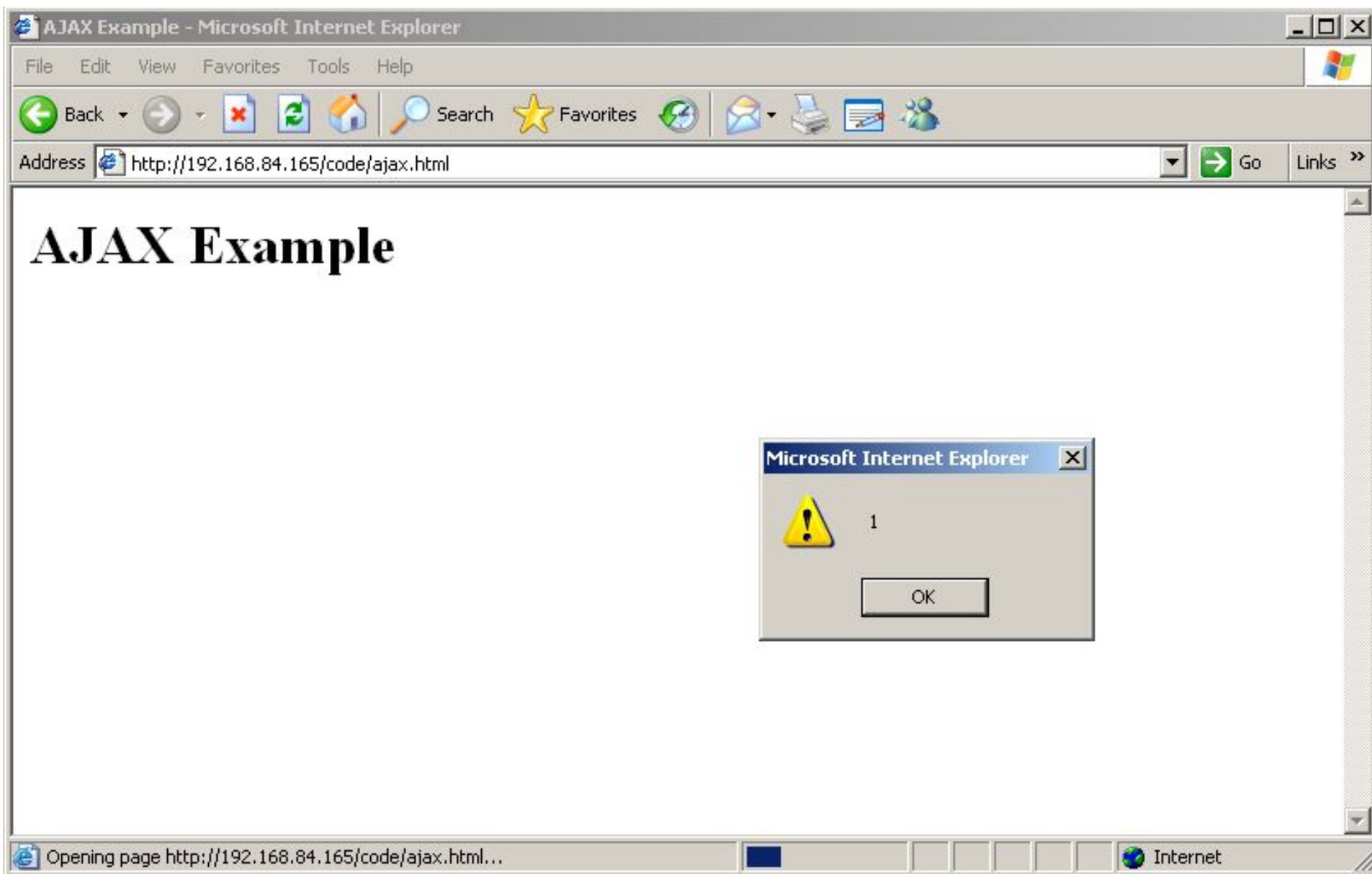
The "Console" panel is also open, showing the following log entries:

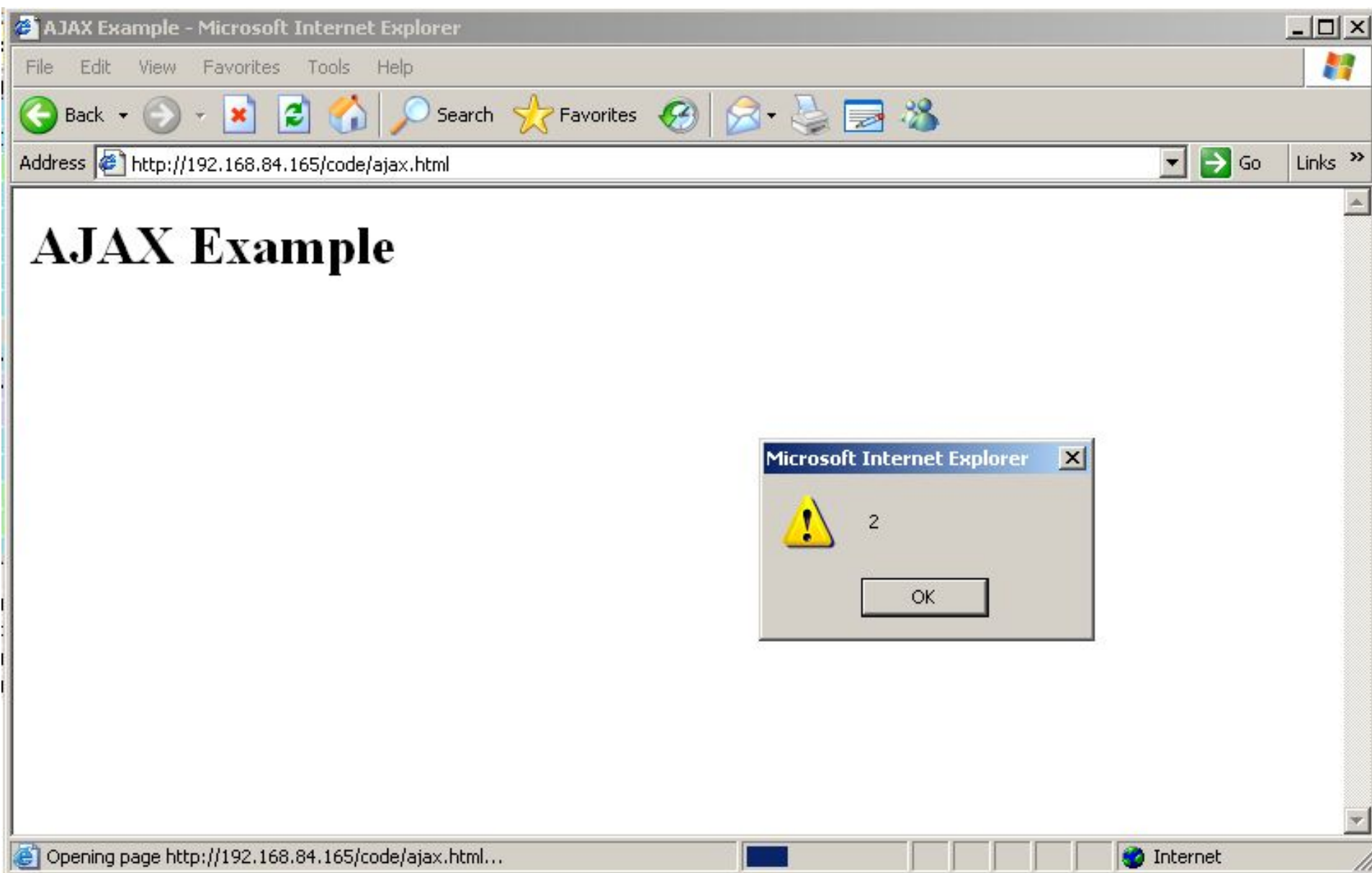
Message	Source
Before Request	ajax.html:30
1	ajax.html:23
After Request	ajax.html:33
2	ajax.html:23
3	ajax.html:23
4	ajax.html:23



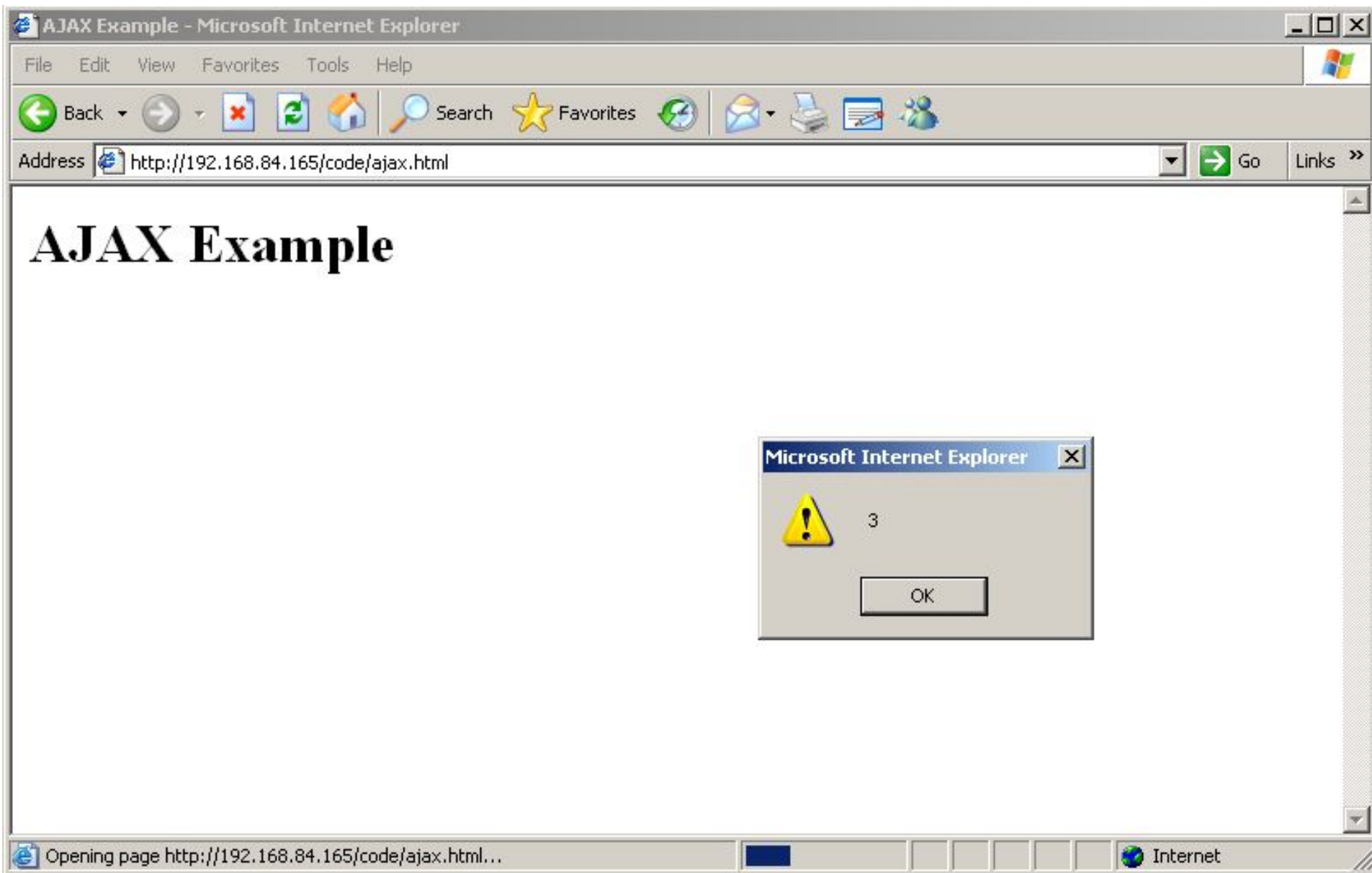


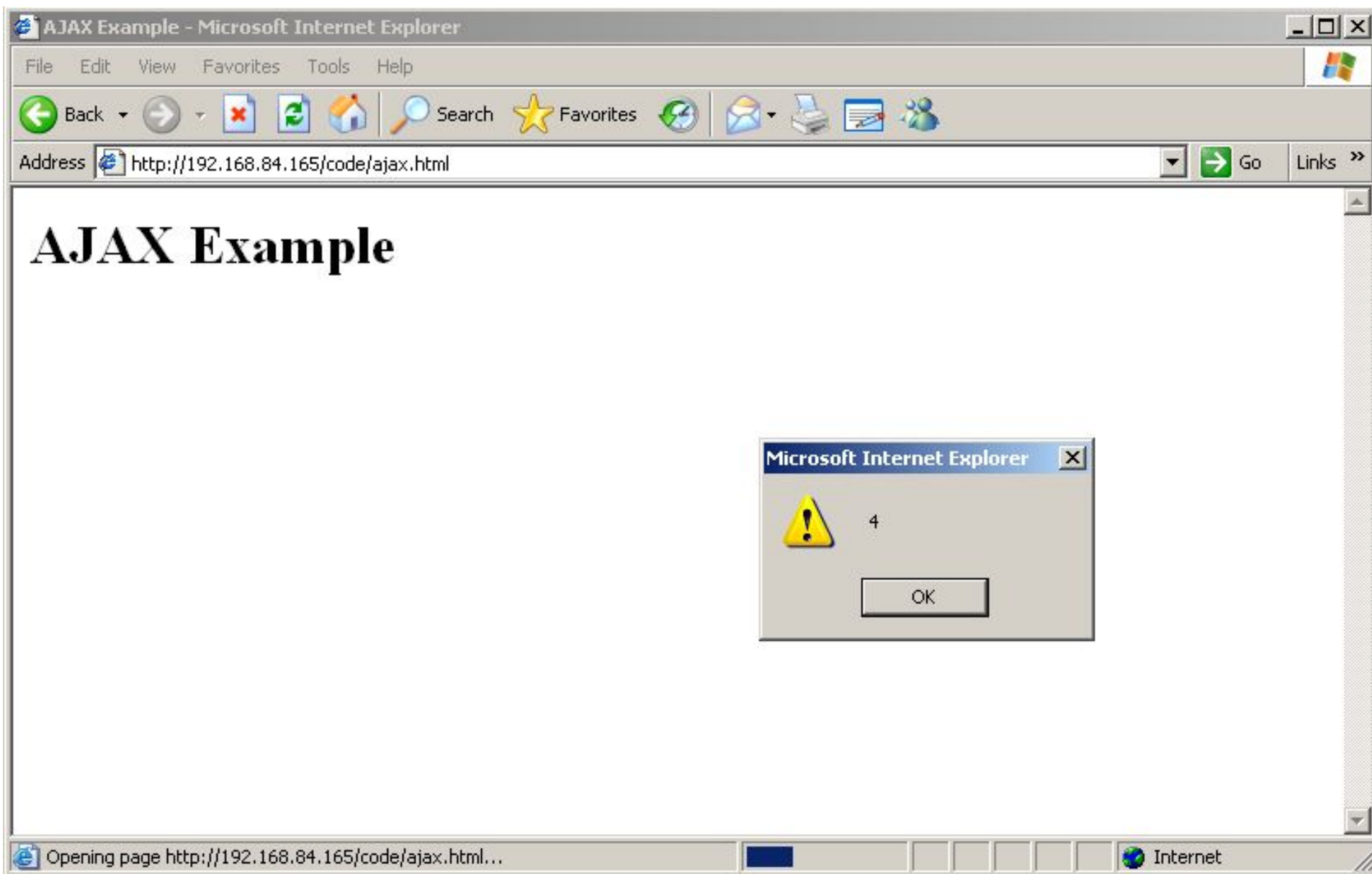


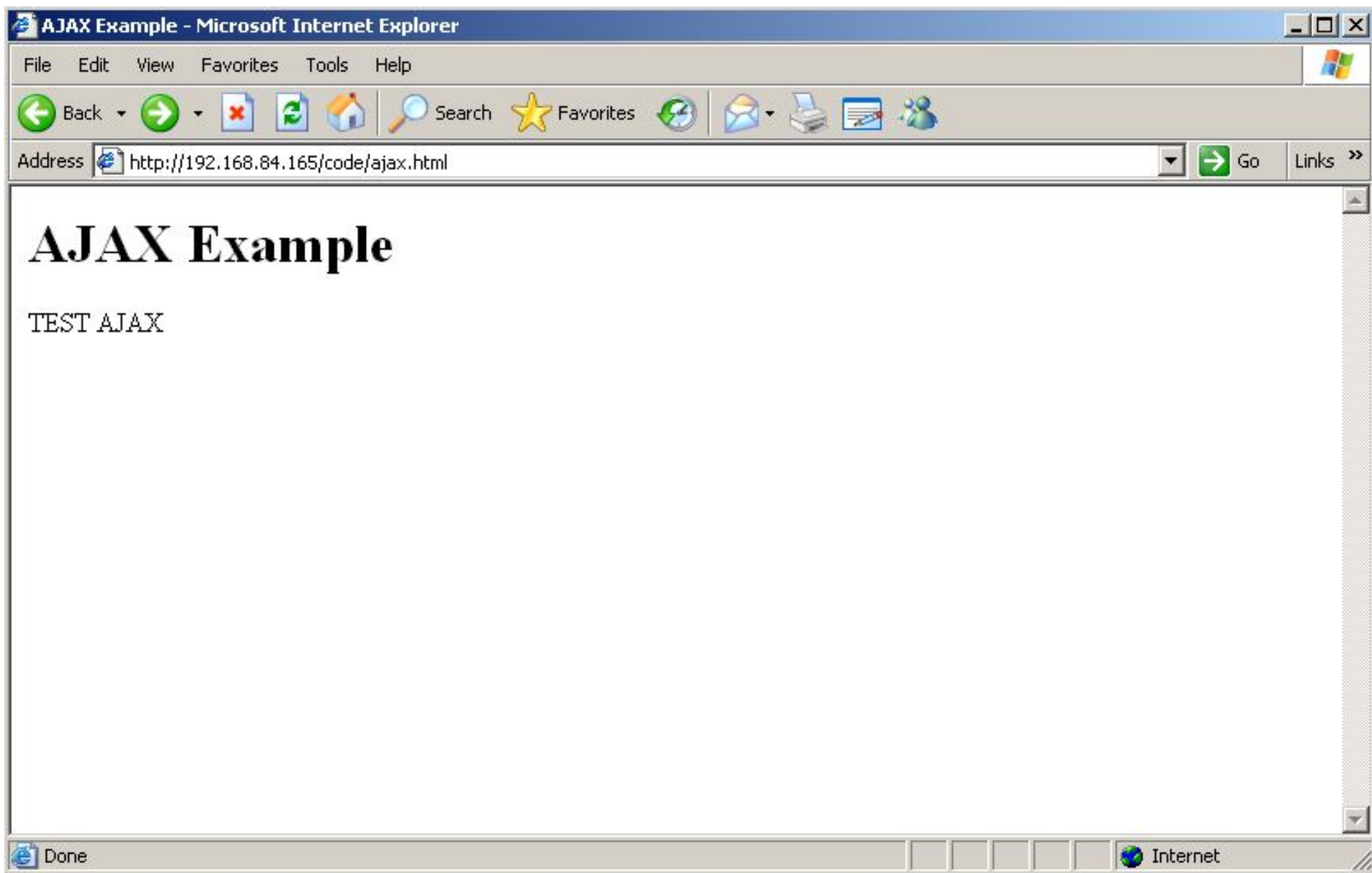


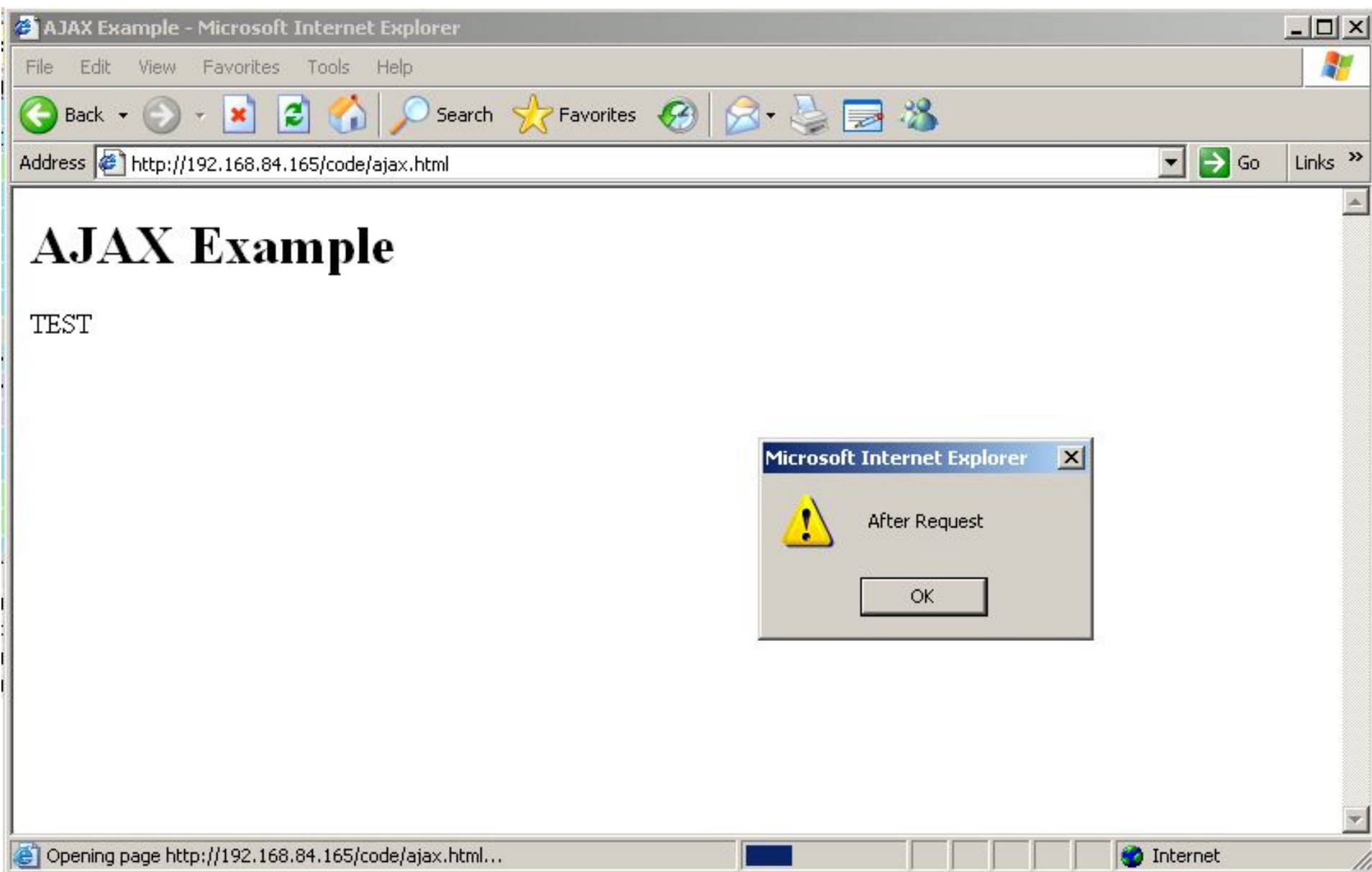


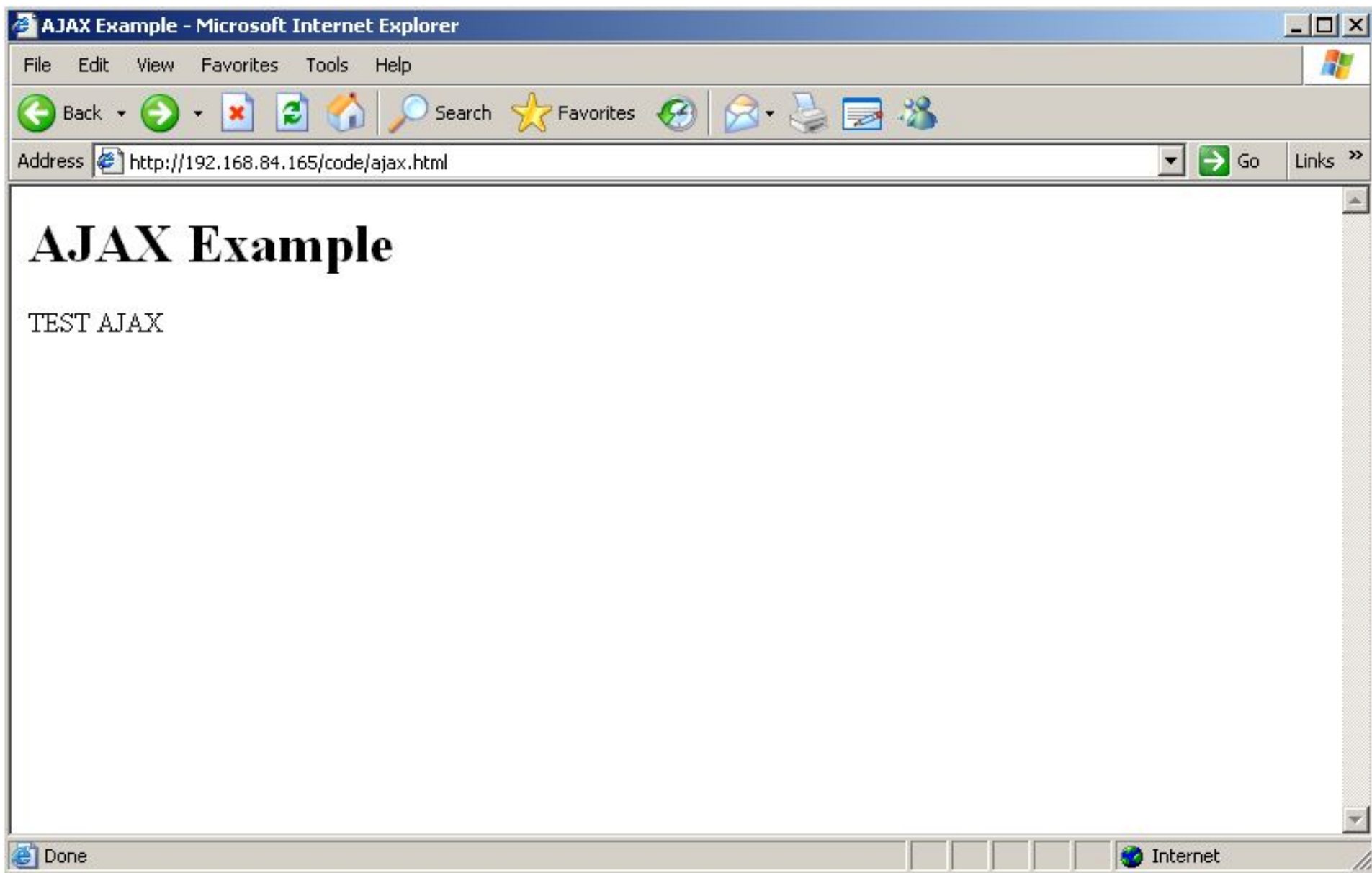


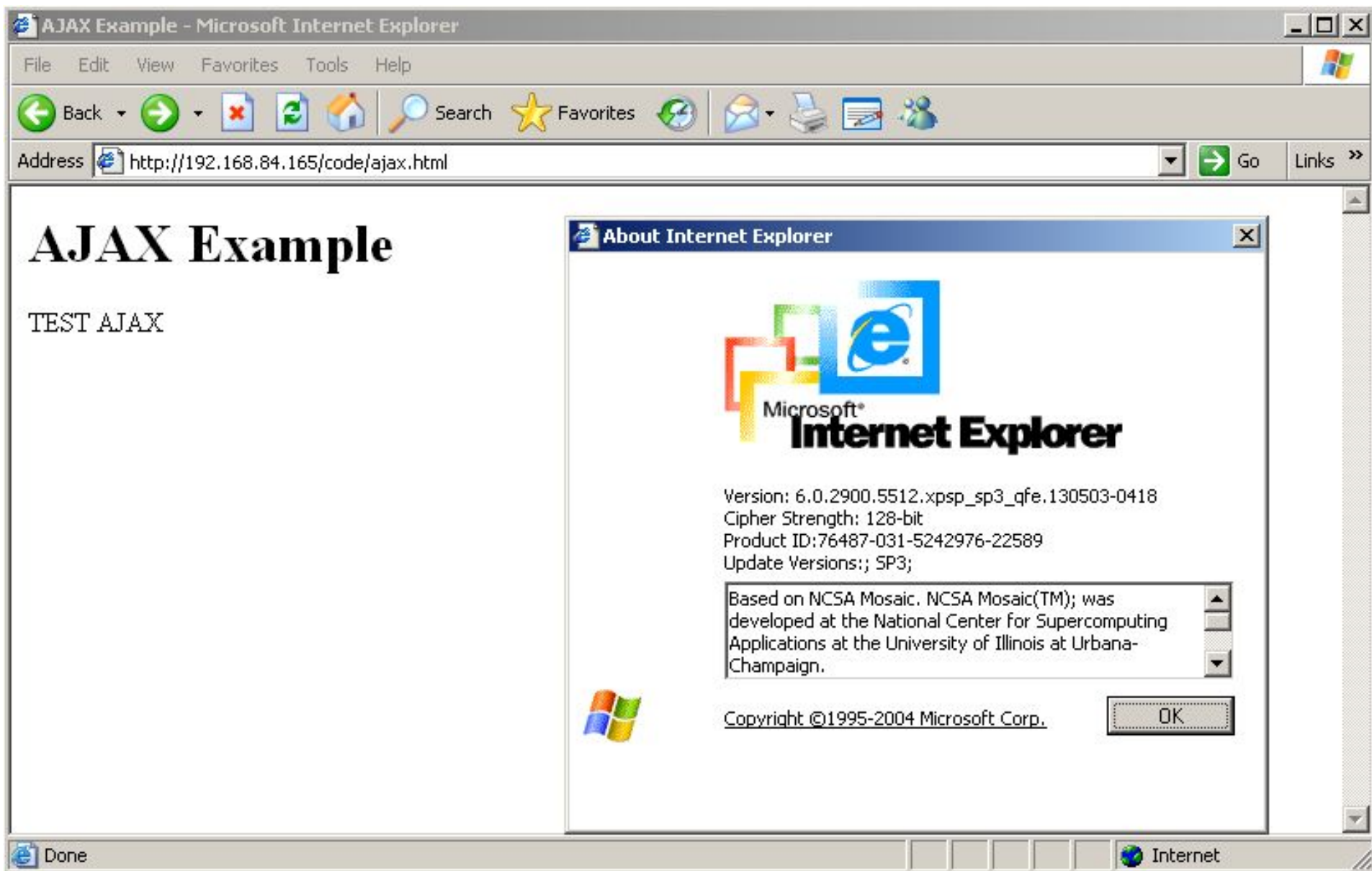












# XMLHttpRequest with jQuery

```
<!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>
```







