Stony Brook University

CSE 361: Web Security

Introduction and History of the Web

Nick Nikiforakis

Who am I, and where can you find me?

- Nick Nikiforakis
  - Associate Professor in Department of Computer Science
  - Research interests:
    - Web security and Privacy
    - DNS security
    - Intrusion detection
  - Office: 361

About the course

- Learn the ins and outs of securing web applications
  - Theory: Lectures, mandatory readings, etc.
  - Practice: Gradually (throughout the semester) secure a vulnerable web application

- CSE 331 (Computer Security Fundamentals) used to be a pre-req. for this course
  - We will cover as much "generic" security as necessary for this course

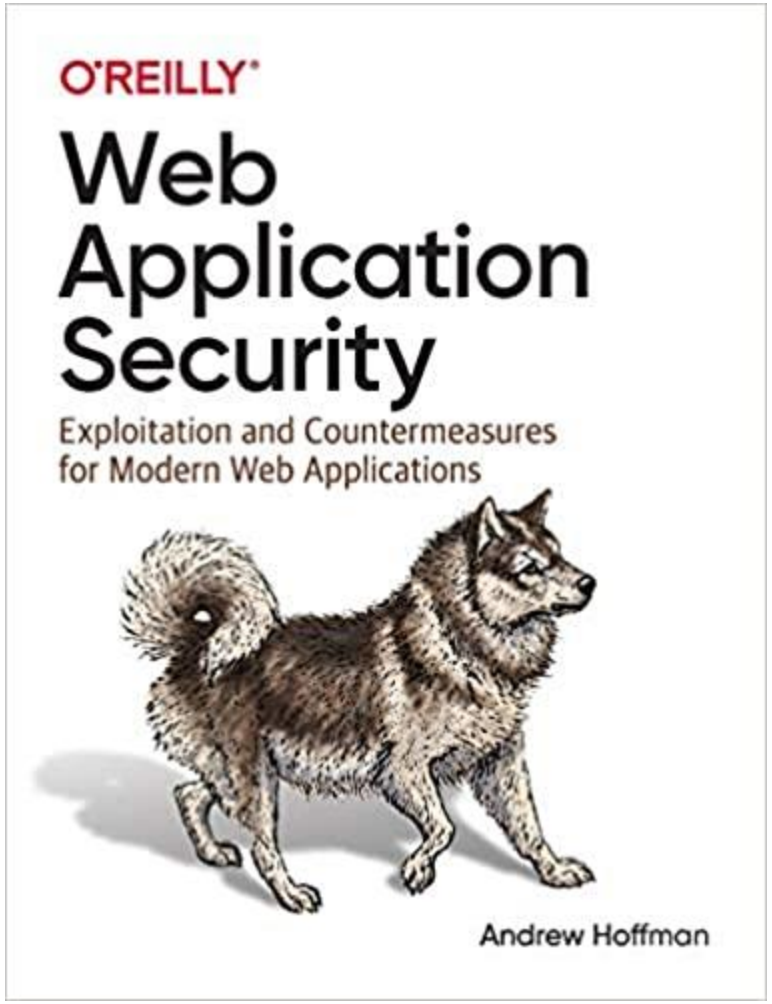
Logistics

- Class will be on Tuesdays and Thursdays, 5 PM – 6:20 PM

- Office Hours: Tuesdays and Thursdays, 3 PM – 4 PM
- Grade breakdown
  - Individual assignments (15%)
    - Mostly reading papers, writing summaries, and answering questions
  - Group assignments (25%)
    - Semester-long project-like assignments on securing a web application
  - Midterm (25%)
  - Final (35%)

Logistics

- No official textbook required
  - Slides and mandatory readings should be sufficient
- You should attend lectures
  - Not mandatory but highly encouraged
    - Live demos will not be recorded
- Optional book
  - “Web Application Security” book by Andrew Hoffman



Late submission policy

- Paper summaries, lab reports, and final project must be delivered by the specified deadlines.
  - Hand them in on time
  - For every day that you are late, there will be a 10% penalty
  - Health-related exemptions will be handled via the appropriate official channels

Code of Conduct

- The work that you present as your own, should be your own
  - Cite the resources that you used (other people's code, documents, etc.)
  - Don't allow your code/paper summaries to be copied
  - Don't copy other people's code or paper summaries

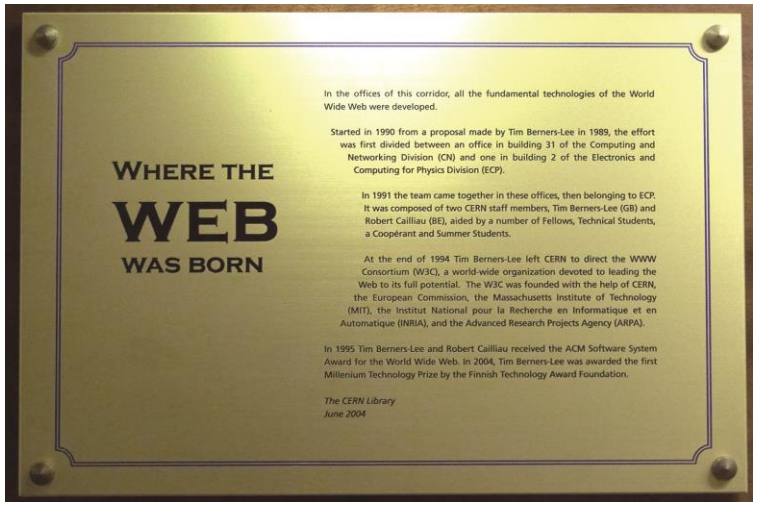
- Anything short of the above, will be grounds for immediate failing of the class and an official report of plagiarism

Generative AI… Just say "No"

- The purpose of this class is for you to learn web security
  - Not for you to ask the LLM to do your homework
  - It's like going to the gym and using a robot to do your strength training
    - It will lift more than you, but you won't get strong
- Short-term reason
  - Moral thing to do
- Long-term reason
  - If you do 1% of the work and the LLM does 99%, who is going to give you a six-figure salary?

The Web has won

- Used by billions of people to retrieve information
  - 2B users monthly on Facebook
  - 2.3M searches per second on Google
  - 464M visitors to ChatGPT each month
- Fully-fledged application platform
  - web-based office applications
    - MS Office vs. MS 365
- Large coverage in mobile applications
  - many mobile apps/SDKs are just Web views
  - Desktop apps can just be Electron web apps



... and the hackers with it

2FA compromise led to Crypto.com hack

By Sead Fadilpašić published January 21, 2022

Thieves managed to withdraw funds without having to input 2FA codes

A cyberattack known as e-skimming is getting more common with the rise of online shopping

PUBLISHED FRI, JAN 31 2020•10:17 AM EST | UPDATED FRI, JAN 31 2020•3:29 PM EST

CISA Adds Five-Year-Old jQuery XSS Flaw to Exploited Vulnerabilities List

Jan 24, 2025 Ravie Lakshmanan

Okta Warns of Credential Stuffing Attacks Targeting Customer Identity Cloud

May 30, 2024 Ravie Lakshmanan Credential Stuffing / Incident Response

— Trending News

Unsecured Tunneling Protoc Expose 4.2 Million Hosts, Including VPNs and Routers

How to Bring Zero Trust to W Security with a Cloud-based Captive Portal?

Trump Terminates DHS Advi Committee Memberships, Disrupting Cybersecurity Rev

Hacker demonstrated 'Remote Code Execution' vulnerability on EBay website

MOVEit, the biggest hack of the year, by the numbers

At least 60 million individuals affected, though the true number is far higher

Carly Page @carlypage_ / 11:45 AM EDT • August 25, 2023 Comment

FTC warns companies to remediate Log4j security vulnerability

By: This blog is a collaboration between CTO and DPIP staff and the AI Strategy team | Jan 4, 2022 9:19AM

SHARE THIS PAGE

Why Web Security?

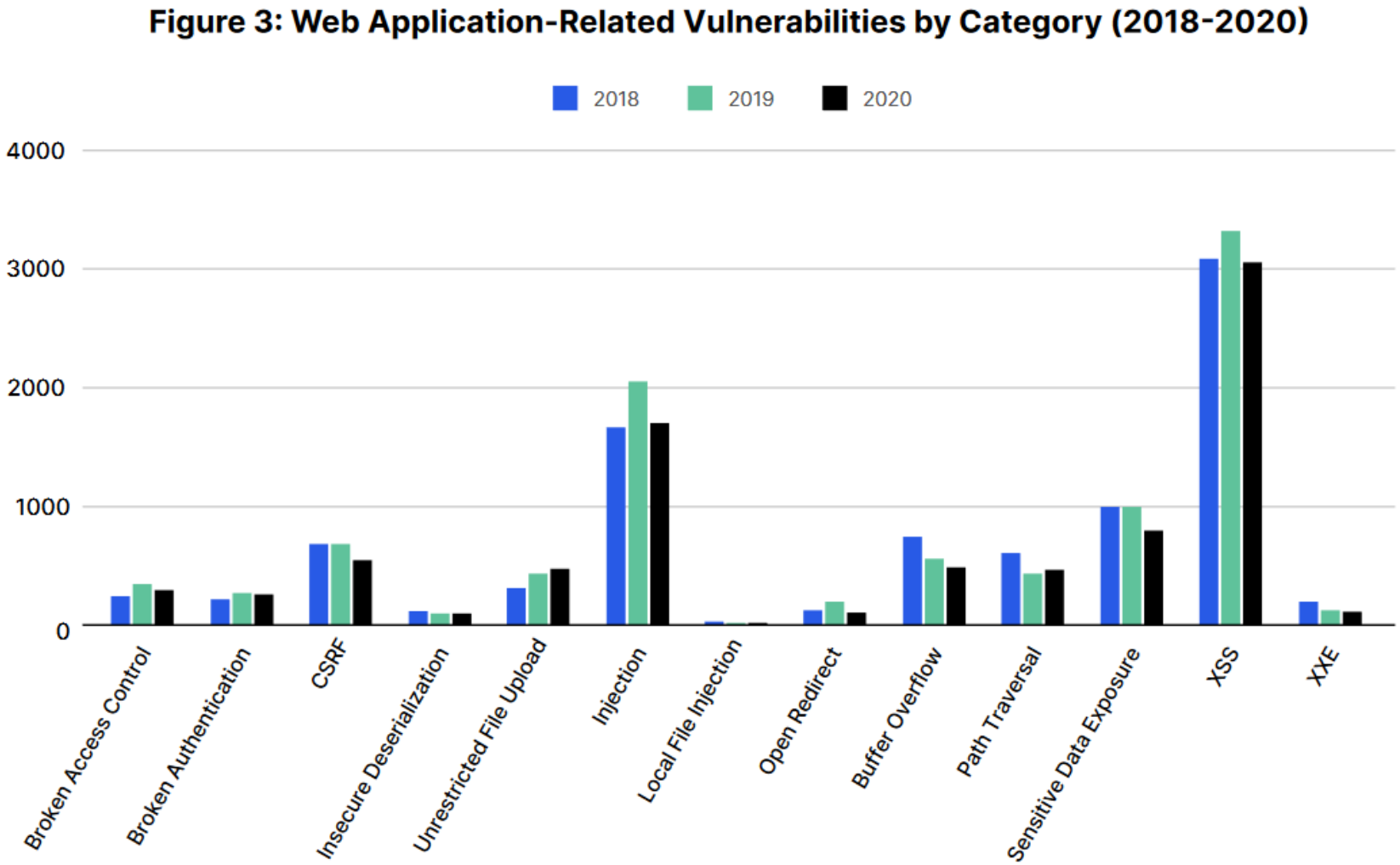


Image source: https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2020/

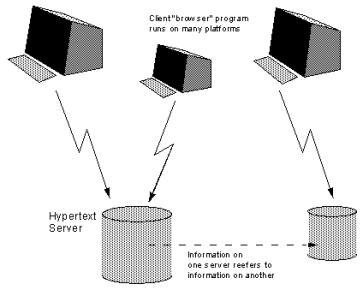

Short History of the Web

From Hypertext to the World Wide Web

- Hypertext concept first mentioned in 1945
  - Theoretical, Memex (Memory Extender) system by Vannevar Bush
  - No "linear text" anymore, links between documents

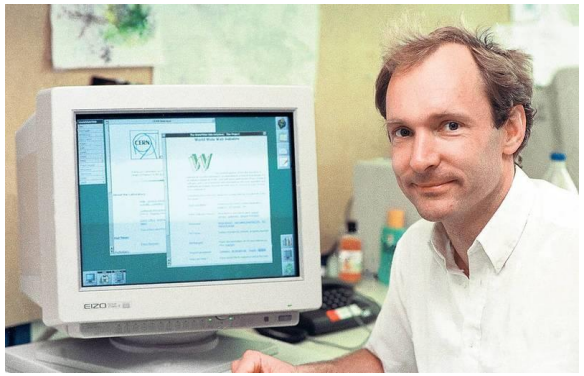
- In 1980, Tim Berners-Lee developed ENQUIRE
  - local links between documents only

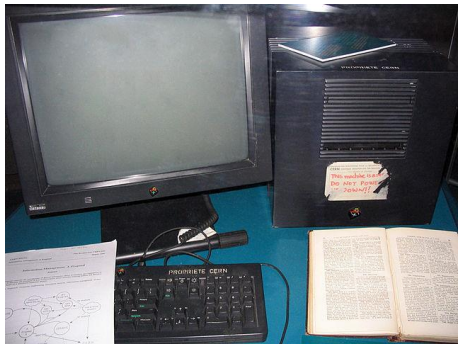
- In 1989, Berners-Lee wrote "Information Management: A Proposal"
  - extends Hypertext to multiple servers and links between them
  - Basis for the modern Web



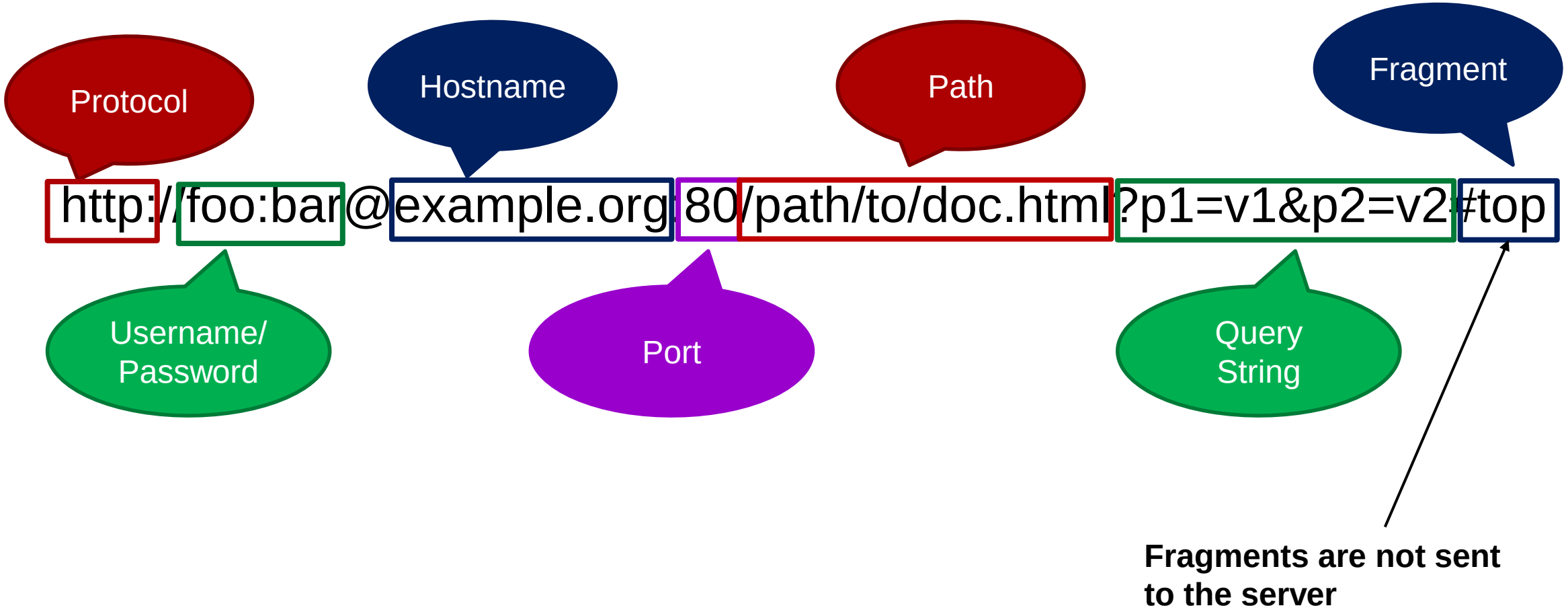
HTTP and HTML

- Web as envisioned by T. Berners-Lee
  - document-centric
  - stateless (just documents linking to one another)
  - structured (based on SGML)
  - tags for semantic interpretation
- HTTP 0.9 introduced in 1991
  - required to answer with an HTML page
  - no headers either way (introduced in 1992 though)
- HTML initially supported 18 tags
  - 11 made it into HTML4 and later versions





Uniform Resource Locator (URL)



HTTP Evolution over Time: HTTP 0.9

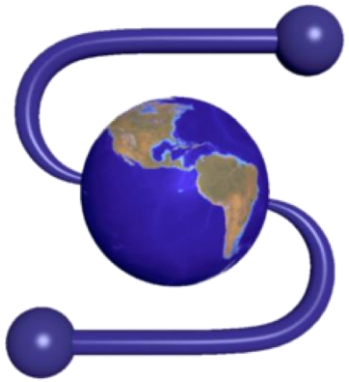
- Requirements
  - as simple as possible
  - serve **single** HTML pages
- Result
  - only GET requests
  - no client or server headers
  - server directly answers with HTML body

```
GET /path/to/doc.html
```

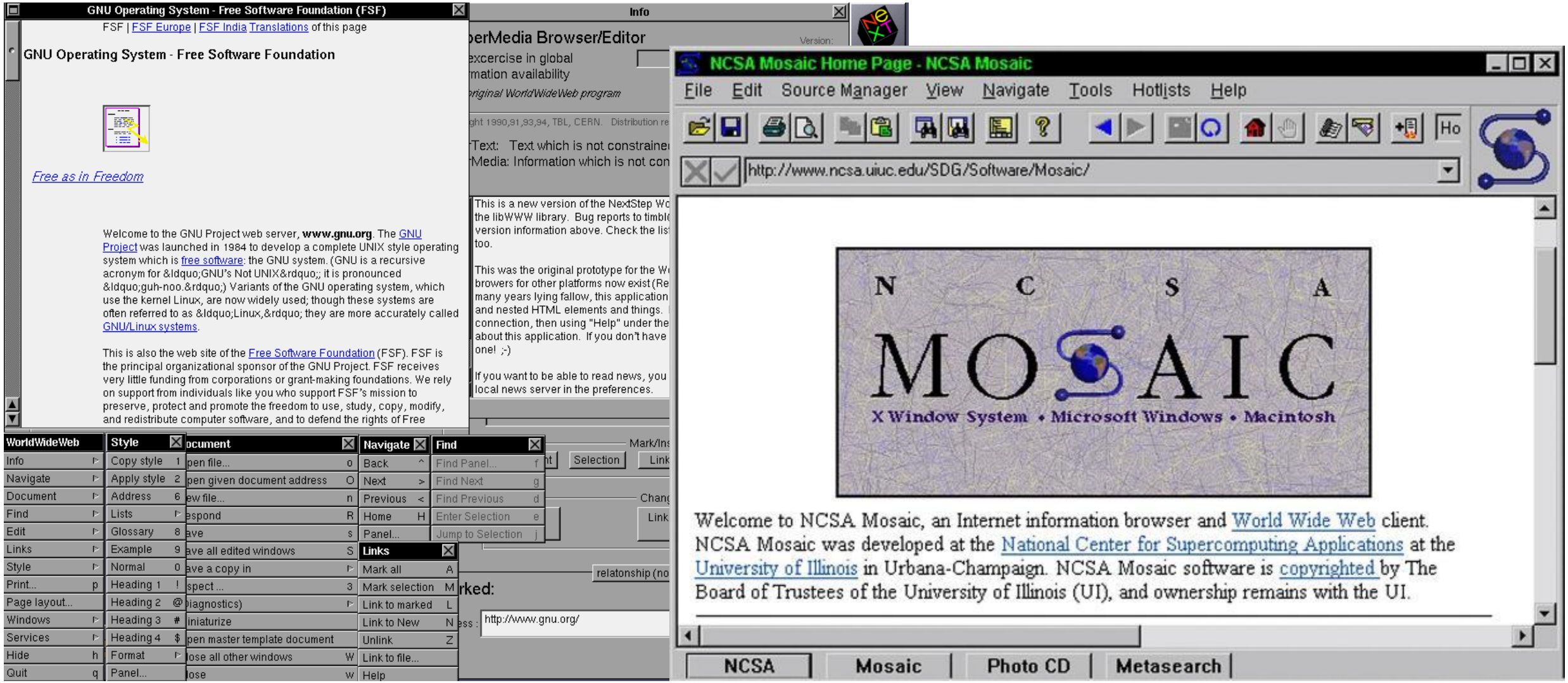
```
<html>…

(connection closed)
```

The first "real" browser: Mosaic (1993)

- Initial version of HTML could only link to images
  - allowed to be on remote server
- Mosaic introduced the `<img>` tag for inline images
  - Implemented by Marc Andreessen
- Images could reside on remote server
  - Birth of the multi-origin Web
  - Followed by many HTML tags later (embed, object, style, script, …)



We've come a long way



HTTP Evolution over Time: HTTP 1.0 (1991-1995)

- Requirements
  - serve content other than plain text documents
  - allow for authentication
  - allow for transmission of meta information, e.g., age of file
  - transmit data to the server (via forms)
- Result
  - Mandatory HTTP version in request
  - Optional headers in request and response
  - Status Line in response
  - New methods: POST and HEAD

```
GET / HTTP/1.0
Host: example.org
```

```
HTTP/1.0 200 OK
Content-Length: 123

<html>…
(connection closed)
```

HTTP Requests (since HTTP/1.0)

- Consists of several, partially optional components
- Request Line with *Verb*, Path, and Protocol
- List of HTTP headers, as header:value
- Empty line to end headers
- Optional body message (used, e.g., with POST requests)

```
GET /index.html HTTP/1.0
Host: stonybrook.edu
Cookie: hello=1
```

HTTP GET request

- Purpose: retrieve resource from server
- Should not cause side effects on Web server's state
  - dubbed "idempotent" in W3C standard
  - although it does often cause side effects in practice, due to developers
- Should not carry a message body
- Parameters passed via URL
  - Special characters percent-encoded (hex value of char, e.g., ? = %3F)
  - **Usually logged on server side together with requested file**

```
GET /index.html?name=value%3F HTTP/1.0
Host: stonybrook.edu
```

HTTP POST request

- Purpose: send data to the server
  - for storage or processing
  - should be used for state-changing operations
- Can be combined with GET parameters
- Message body contains data
  - Depending on content-type, percent-encoded or plain

```
POST /index.html?name=value%3F HTTP/1.0
Host: stonybrook.edu
Content-Length: 10
Content-Type: application/json

{"a": "?"}
```

```
POST /index.html?name=value%3F HTTP/1.0
Host: stonybrook.edu
Content-Length: 5
Content-Type: application/x-www-form-urlencoded

a=%3F
```

HTTP Response (since HTTP/1.0)

- Status Line: Protocol, Status Code, and *Status Text*
- List of HTTP headers, as header:value
- Empty line to end headers
- Response Body

```
HTTP/1.0 200 OK
Server: nginx
Content-Type: text/html
Content-Length: 123

<html>…</html>
```

HTTP Response Codes

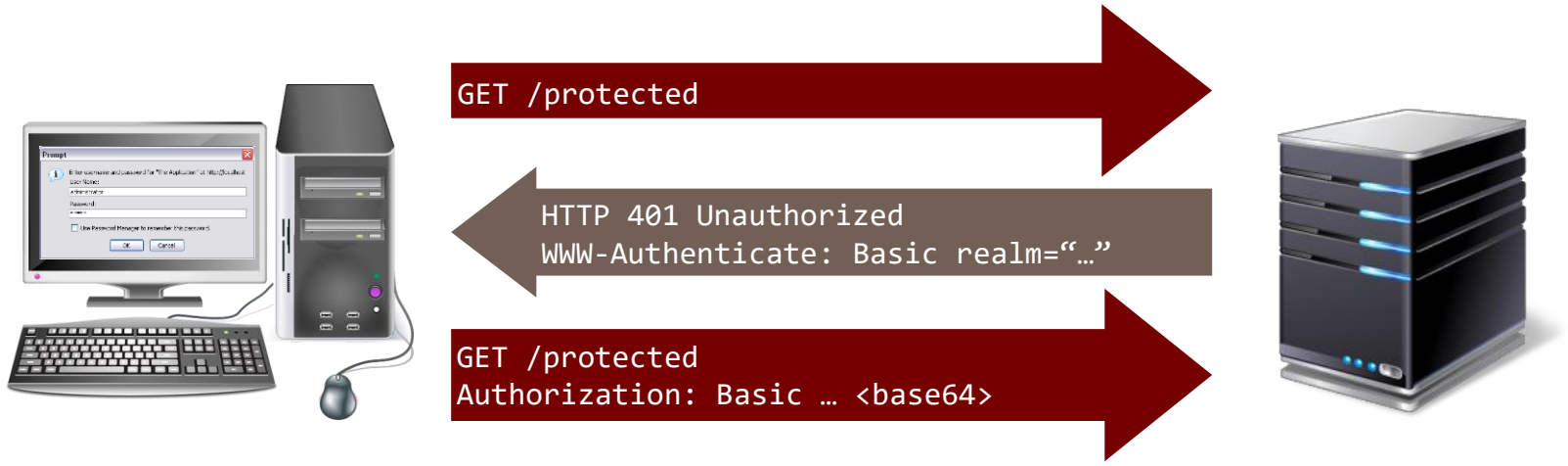
- 2xx Success
  - 200 OK
  - 206 Partial Content (for range requests)
- 3xx Redirection
  - 301 Moved Permanently (always redirect to new URL)
  - 302 Found (redirect once, don’t store redirect)
  - 304 Not Modified (not changed since last client request, not transferred)
  - 307 Moved Temporarily (only redirect to new URL this time)

HTTP Response Codes

- 4xx Client errors
  - 400 Bad Request (e.g., no carriage return in HTTP request)
  - 401 Unauthorized (used for HTTP authentication)
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
  - 418 I’m a teapot (April Fool’s Joke, see RFC 2324)
- 5xx Server errors
  - 500 Internal Server Error
  - 502 Bad Gateway (e.g., timeout in reverse proxies)

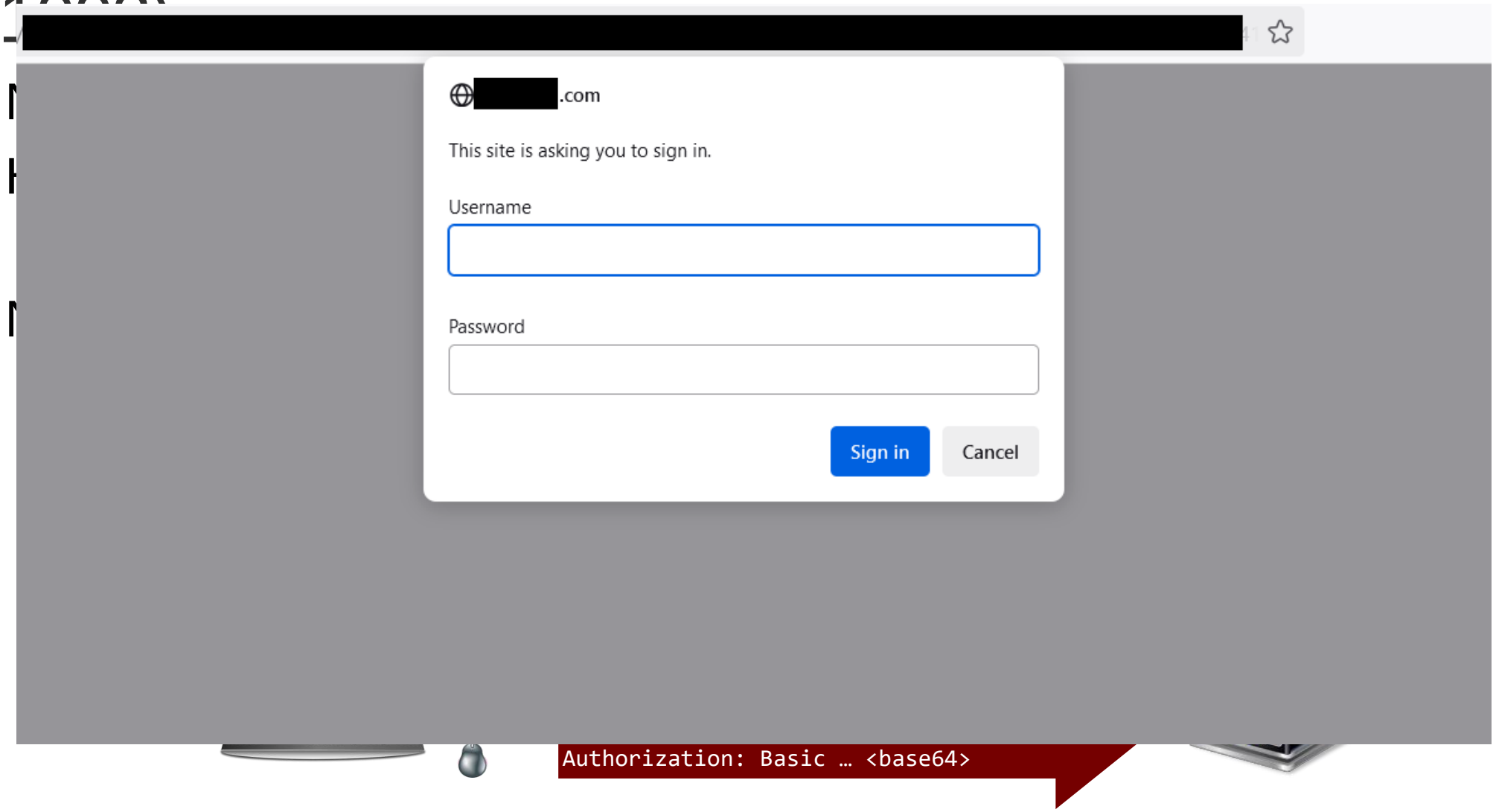
First Security Considerations: HTTP Authentication (1993)

- Need for authentication/authorization was recognized early on
- HTTP remained stateless
  - Authentication via HTTP header
- Not too useful for session management though



First Security Considerations: HTTP Authentication (1

- N
- H
- N



Cookies (1994)

- Adding state to the stateless Web
  - Required to develop applications which should re-identify a user

- Initially added in Netscape Navigator
  - set via HTTP response header
  - sent along with every subsequent request
  - … until lifetime is exceeded, or cookie is deleted

JavaScript (1995)

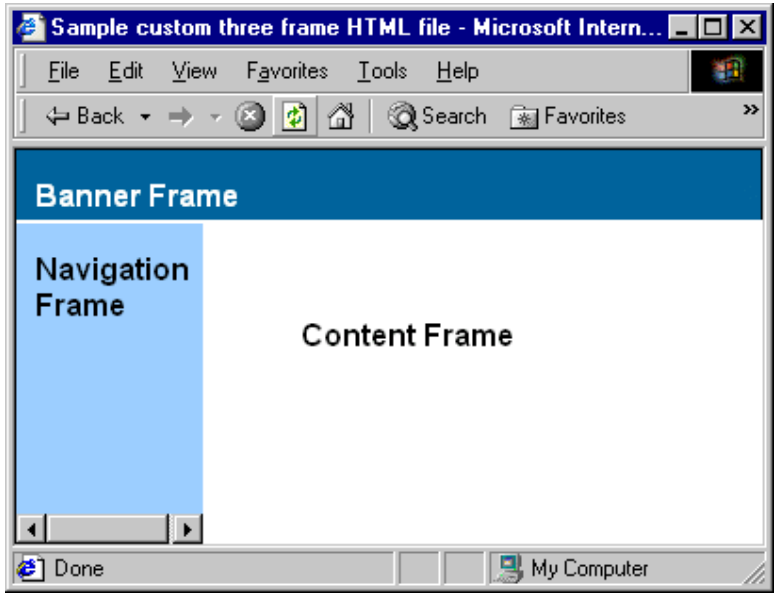
- Netscape wanted a "glue language" added to HTML
- Brendan Eich was hired by Netscape to "implement Scheme in the browser"

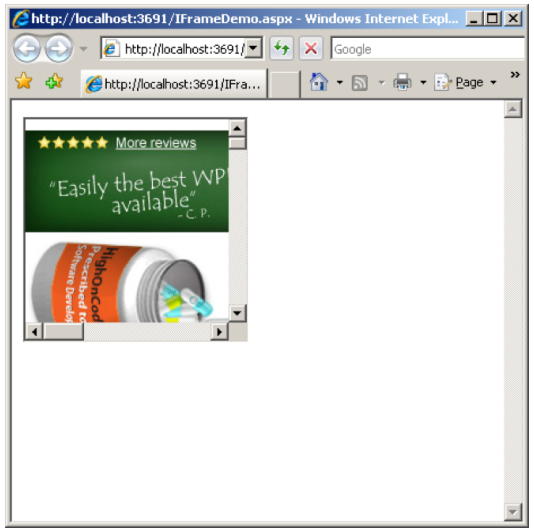
- Instead, he was tasked with developing *Mocha* (initially dubbed *LiveScript)*
  - in the first beta release, already renamed to JavaScript
  - Java was very popular back then
- JavaScript later specified as ECMAScript (ECMA-262)


Netscape - [Welcome to Mosaic Communications Corporation!]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://home.mcom.com/
What's New! What's Cool! Handbook Net Search Net Directory Software
Welcome to
Mosaic
Communications
Corporation
Click on the Image or here to advance

Frames (1995) and Iframes (1997)

- Concept of frames to display more than one HTML page in a window
  - reduce bandwidth by splitting page
  - fixed navigation elements
- Frames are permitted to come from different origins
- Each frame behaves like a browser window
  - Content rendered and interpreted as if page is loaded regularly
- Reason for introducing the Same-Origin Policy
  - separates frames if they don't share an origin
  - only introduced after first cases of abuse…



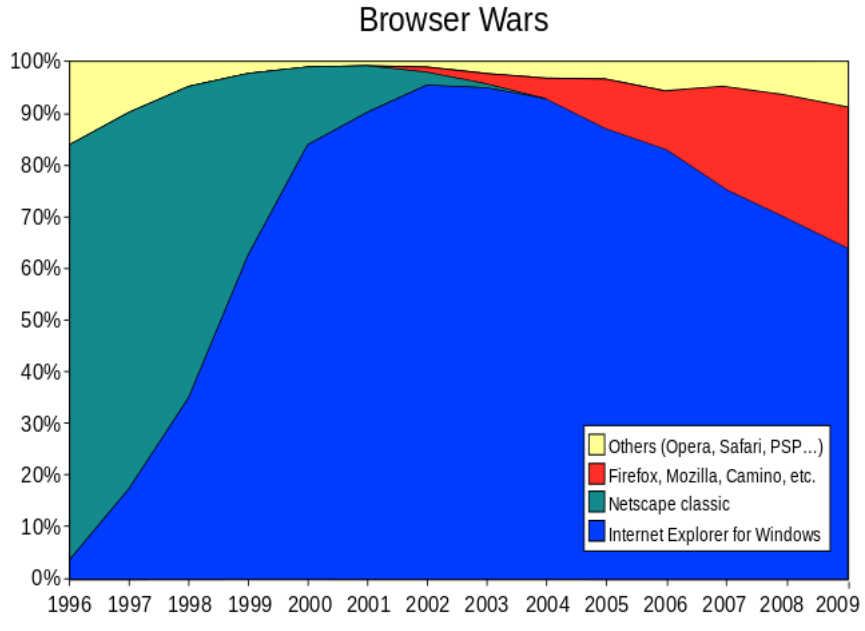


Cascading Style Sheets (1996)

- HTML was initially designed to reflect structure of a document
  - Title, Headings, Sections, Listings, Lists, …
- Web became more popular, should look nicer
  - design tags were add, such as `font`, `b`, `i`
- CSS added to separate *structure* and *presentation*
  - Declarative syntax
  - Could be included remotely or added inline
- Capable of e.g., background images, element placing, opacity

The First Browser War (1996-1999)

- Market share battle between Netscape and Internet Explorer
- Goal: work with as many sites as possible to win the battle (**compatibility**)
  - everybody was "programming" bad HTML
  - resulted in highly relaxed parsing process
  - error-tolerant to a fault…
- Microsoft's Internet Explorer won by a landslide
  - also caused by Microsoft's OS dominance



HTTP Evolution over Time: HTTP 1.1 (finalized 1999)

- Requirements
  - Increased resource size requires other transport and caching strategies
  - Fix some ambiguities in the previous protocol versions
  - Assess server's capabilities to handle requests
- Result
  - New methods: PUT (similar to POST), DELETE, TRACE, CONNECT (proxies), OPTIONS
  - Keep-Alive connections
  - Accept-Encoding info for the server
  - Chunked transfers, range transfers
  - Standardized in RFC 2616

```
GET / HTTP/1.1
Host: example.org
```

```
HTTP/1.0 200 OK
Transfer-Encoding: chunked

7b
<html>…
0
(connection closed)
```

HTTP Evolution over Time: HTTPS (RFC 2818 finalized 1999)

- Initial discussions about S-HTTP (RFC 2660)
  - unencrypted header, only page data and POST bodies encrypted
- Instead: HTTP over TLS/HTTP over SSL/HTTP Secure (**HTTPS**)
  - encapsulates plain HTTP into TLS tunnel
- Server certificate can be verified via chain of trust
  - Trusted root CAs known to browser
- Until 2011, only one hostname per IP
  - Nowadays, Server Name Indication (SNI) allows multiple vhosts via TLS

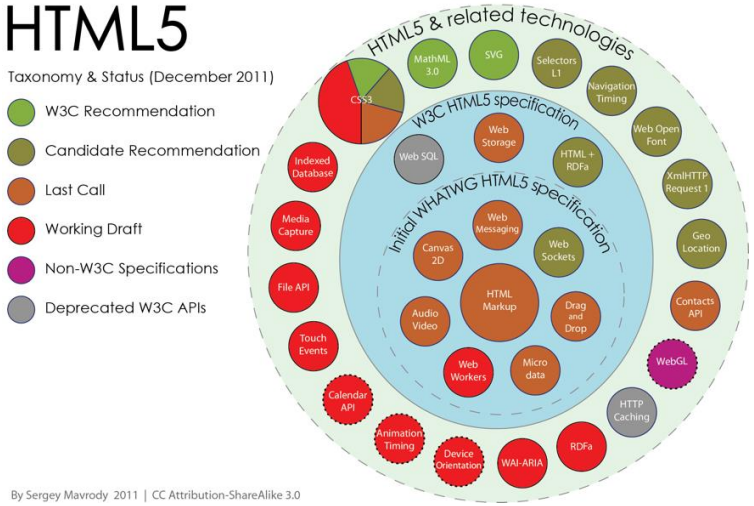
https://www.welivesecurity.com/wp-content/uploads/2016/09/HTTPS-TLS-SSL-Sicherheits.png

Years of Stagnation (2000-2003)

- Netscape gave up fighting Internet Explorer
  - Microsoft reduced investment into new client-side technologies
  - IE4: September 1997
  - IE5: March 1999
  - IE6: August 2001
  - IE7: October **2006**
- New features only added through plugins
  - Flash: audio, video, vector graphics, cross-domain requests
  - Google Gears: client-side persistence, drag&drop, offline support

HTML5 and the WHATWG (2004)

- New browsers introduced to market
  - Apple Safari (2003) and Mozilla Firefox (2004)
- Introduction of the Web Hypertext Application Technology Working Group (WHATWG)
  - Members from Apple, Mozilla and Opera
  - Concerned with "the W3C's direction with XHTML, lack of interest in HTML and apparent disregard for the needs of real-world authors"
- Lead to a number of innovations in the browser
  - Still going on today
  - Final specification of HTML5 by November 2014



HTML5 - Highlights

- Audio and Video tags
  - previously only possible with, e.g., Flash
- Web Storage
  - Easy key/value store on the client
  - Can "only" store strings (objects via serialization)
  - Session and (persistent) Local Storage
- Web Messaging
  - postMessages (we'll cover this soon)
- Web Sockets
  - duplex communication channels with the server

HTML5 - Highlights

- Offline Cache
  - controllable caching behavior enables offline apps
- Web Workers
  - allow developers to have tasks run in background
- Geo Location
  - handy feature when displaying maps or local info
- IndexedDB
  - Mixture of SQL and Web Storage
- New (semantic) HTML tags
  - `nav, menuitem, main, footer, header, ...`

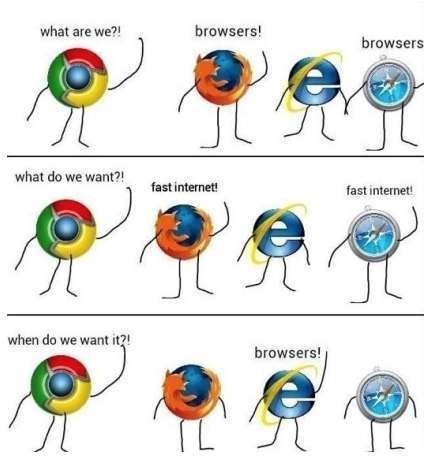
The Web 2.0 (2004 - 2005)

- With new functionality in browser came more powerful Web applications
  - Widespread adoption of Flash
  - XMLHttpRequest (already implemented before under different name in IE) API allowed for "Asynchronous JavaScript and XML" (AJAX)
  - Dynamic mash-up web pages
- Popular application examples include:
  - Google Mail
  - Flickr
  - Facebook

The Second Browser War (since 2005)

- Four major browsers: Internet Explorer, Chrome, Firefox, Safari
- Lively competition (more or less)
- Web standards have been around long enough to focus more on **speed** and not **compatibility**
  - WebKit-based browsers (Chrome, Safari) are fastest nowadays
  - Very active development especially of JavaScript engines
- Both compatibility and speed may be roadblocks for security

- Browser wars 1992-2024:
  - https://www.youtube.com/watch?v=Hdit5-yFHI8

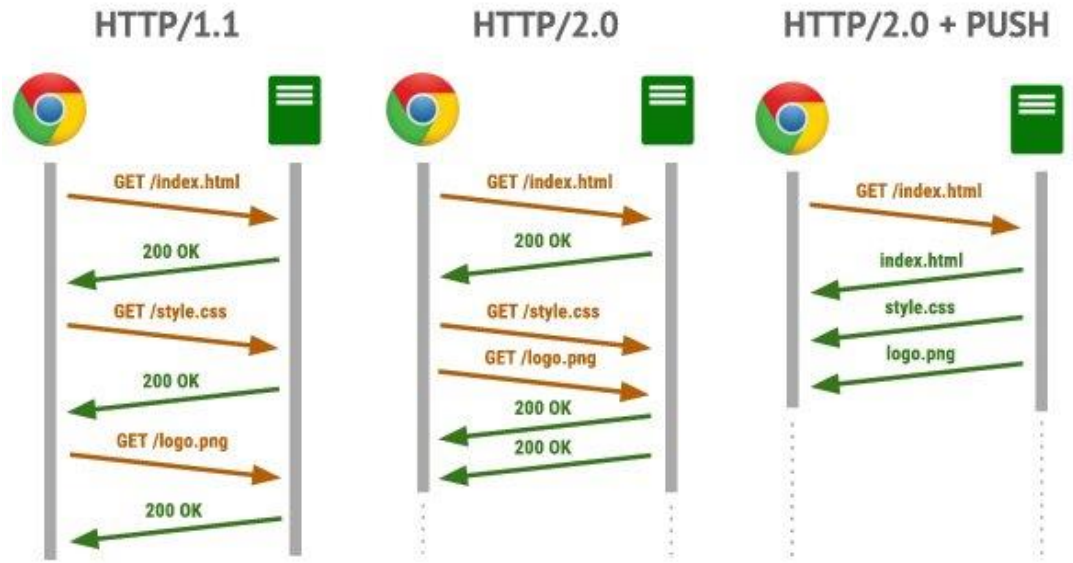


http://www.worthofweb.com/wp-content/uploads/2013/11/internet-browsers.png

http://media02.hongkiat.com/battle-of-browsers-artworks/browsers-battle.jpg

HTTP Evolution over Time: HTTP 2.0 (finalized 2015)

- Requirements
  - Reduce overhead of uncompressed HTTP headers
  - Ensure faster delivery of required resources to client
  - Fix head-of-line blocking from HTTP/1.x
- Result
  - Binary protocol
  - HPACK header compression
  - Server push

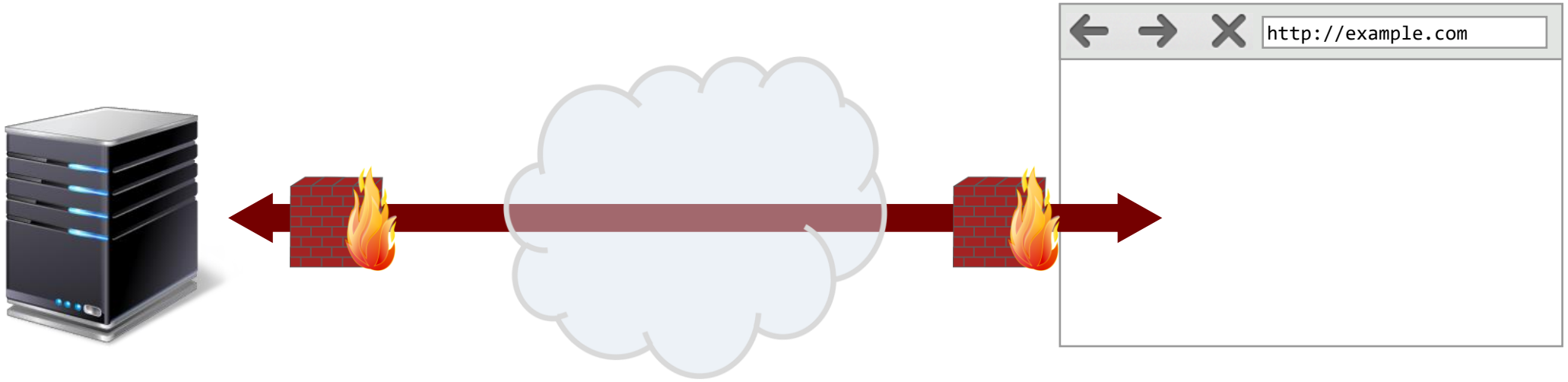


https://giuseppeciotta.net/static/images/h2mosaic_push.jpg

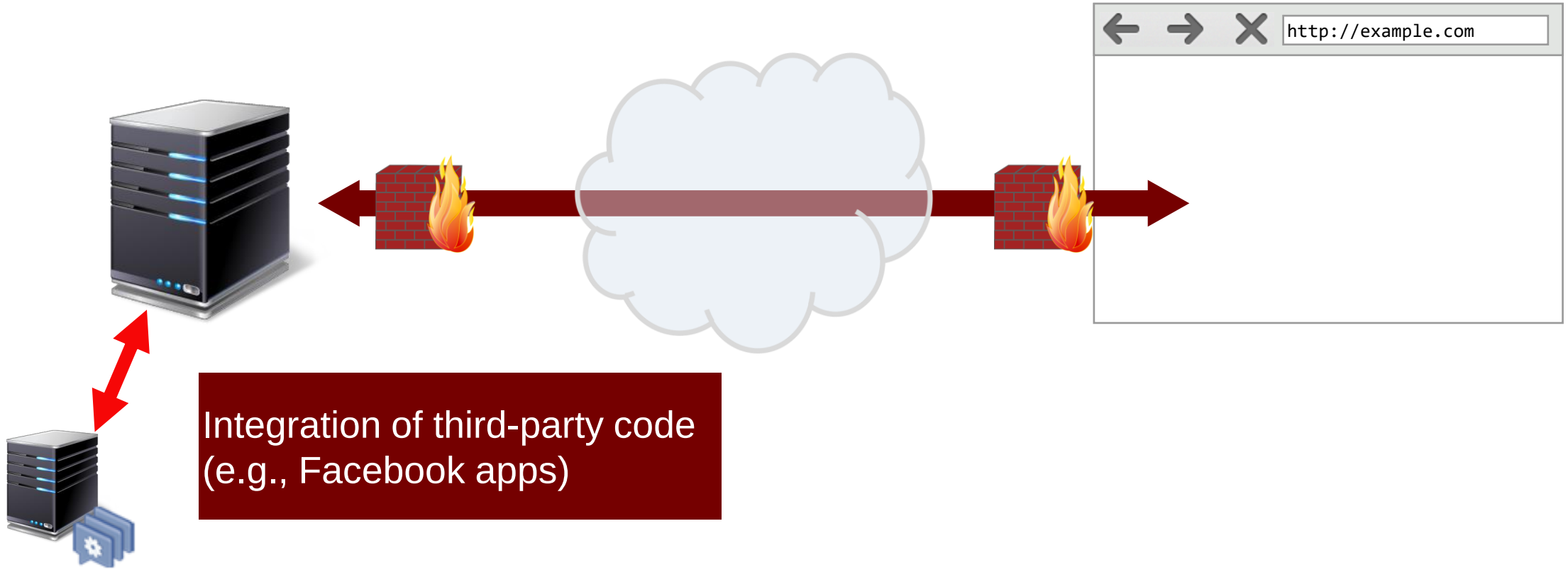
Summary (so far)

- Web was designed to link plain text documents
- Nowadays, we have an application model
  - that is based on multi-origin documents,
  - implements origin-based security models (albeit inconsistently),
  - builds its UI based on at least three languages (HTML, CSS, and JavaScript),
  - and often uses non-security mechanisms (e.g., cookies) for security purposes (e.g., authentication),
  - and supports offline applications with client-side persistence.
- What could possibly go wrong?

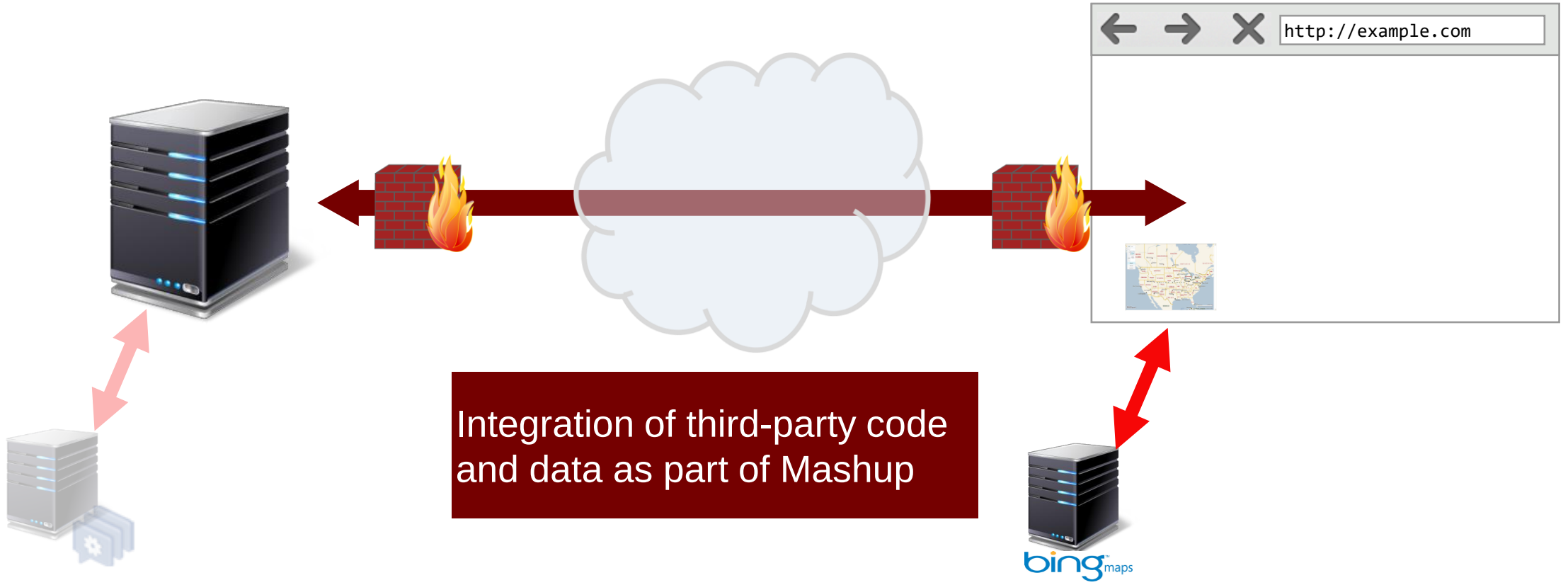
Basic Web Paradigm



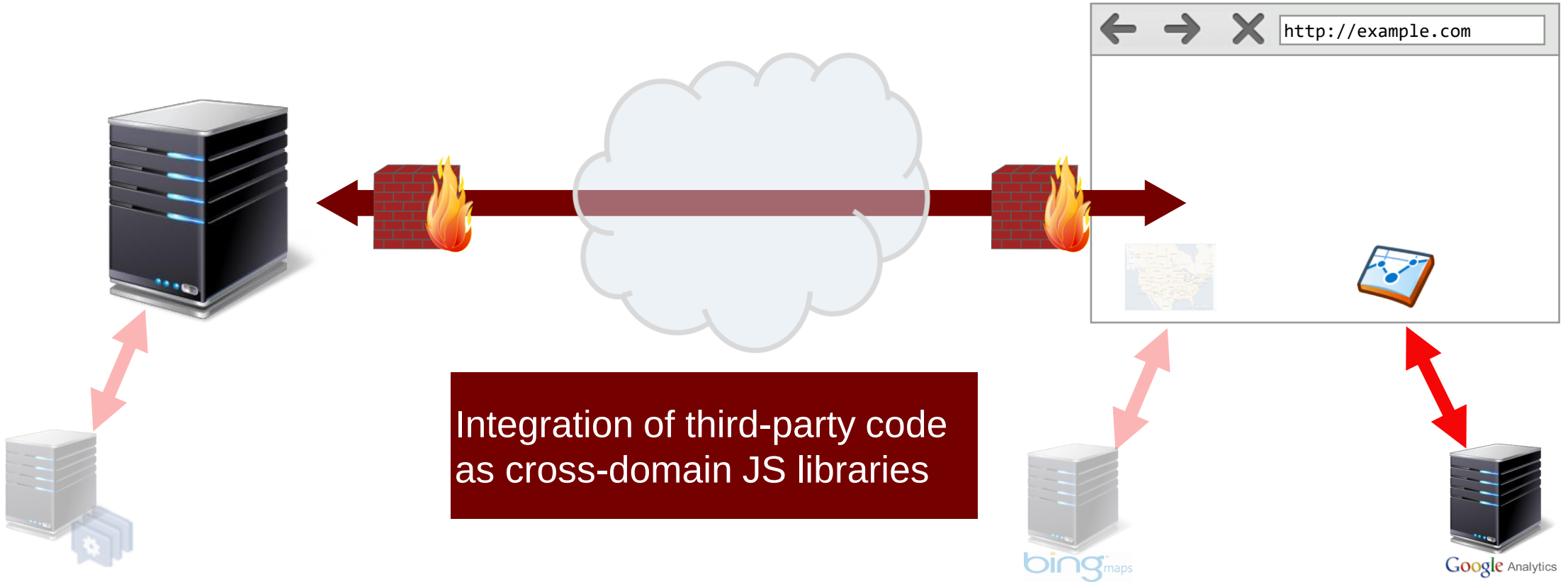
Modern Web Applications



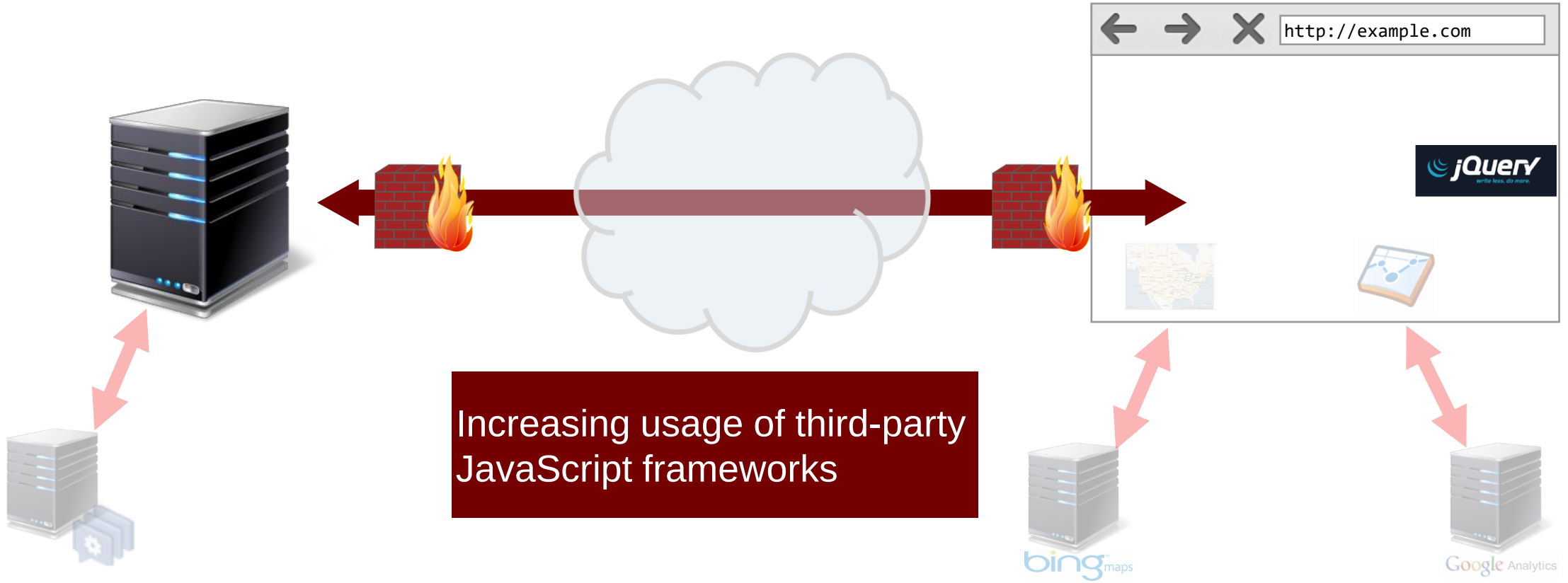
Modern Web Applications



Modern Web Applications

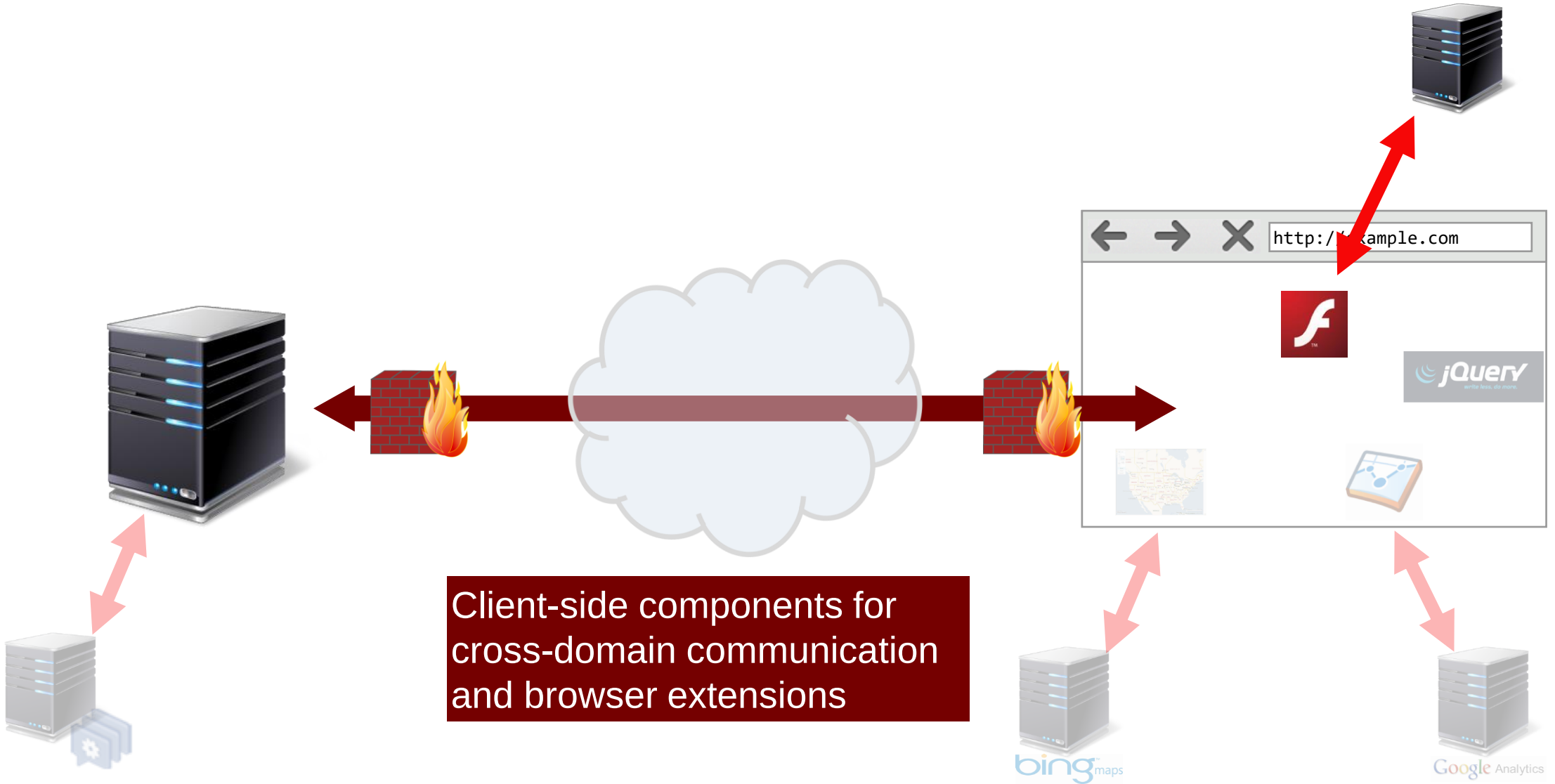


Modern Web Applications



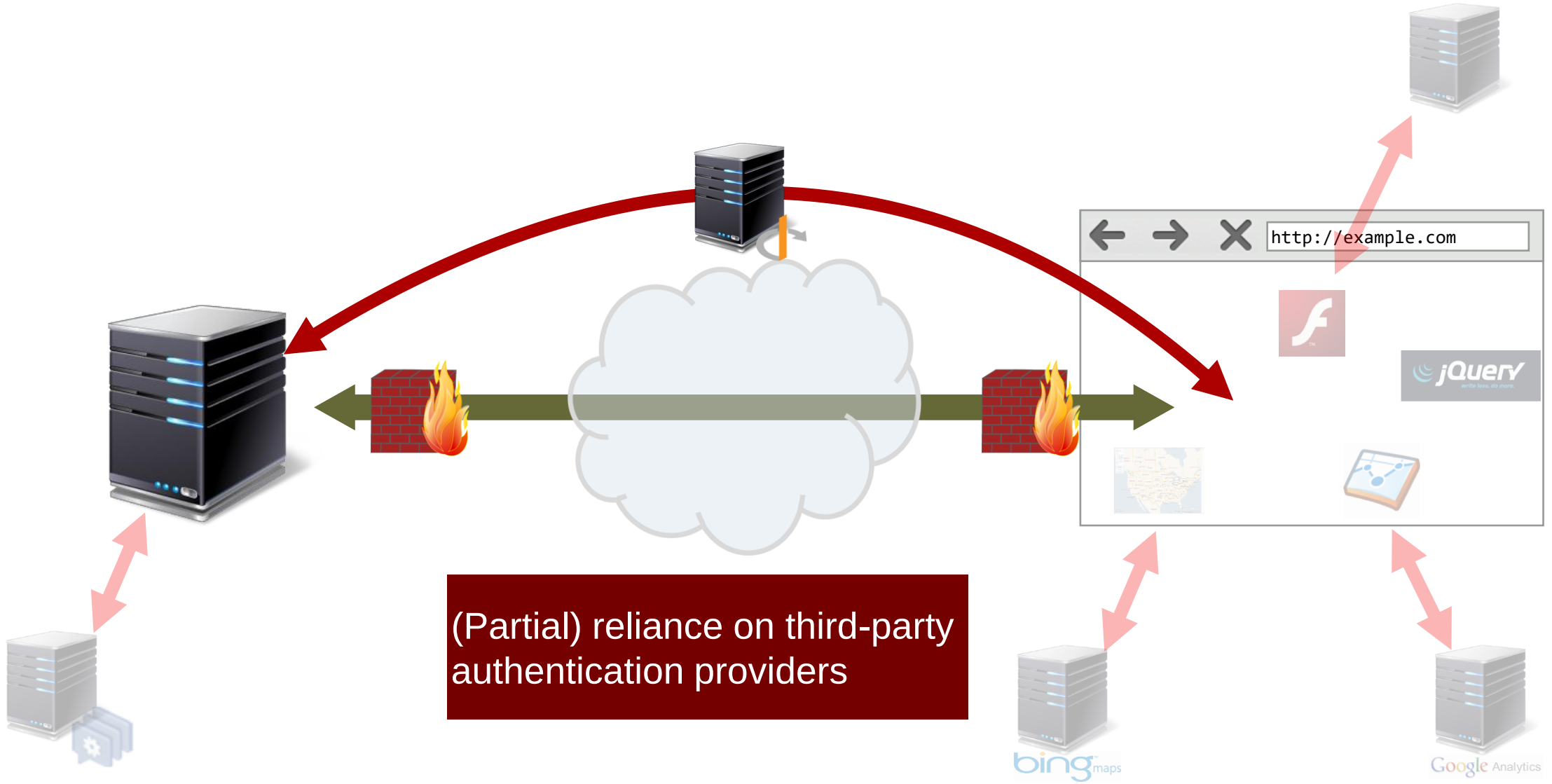
Increasing usage of third-party JavaScript frameworks

Modern Web Applications



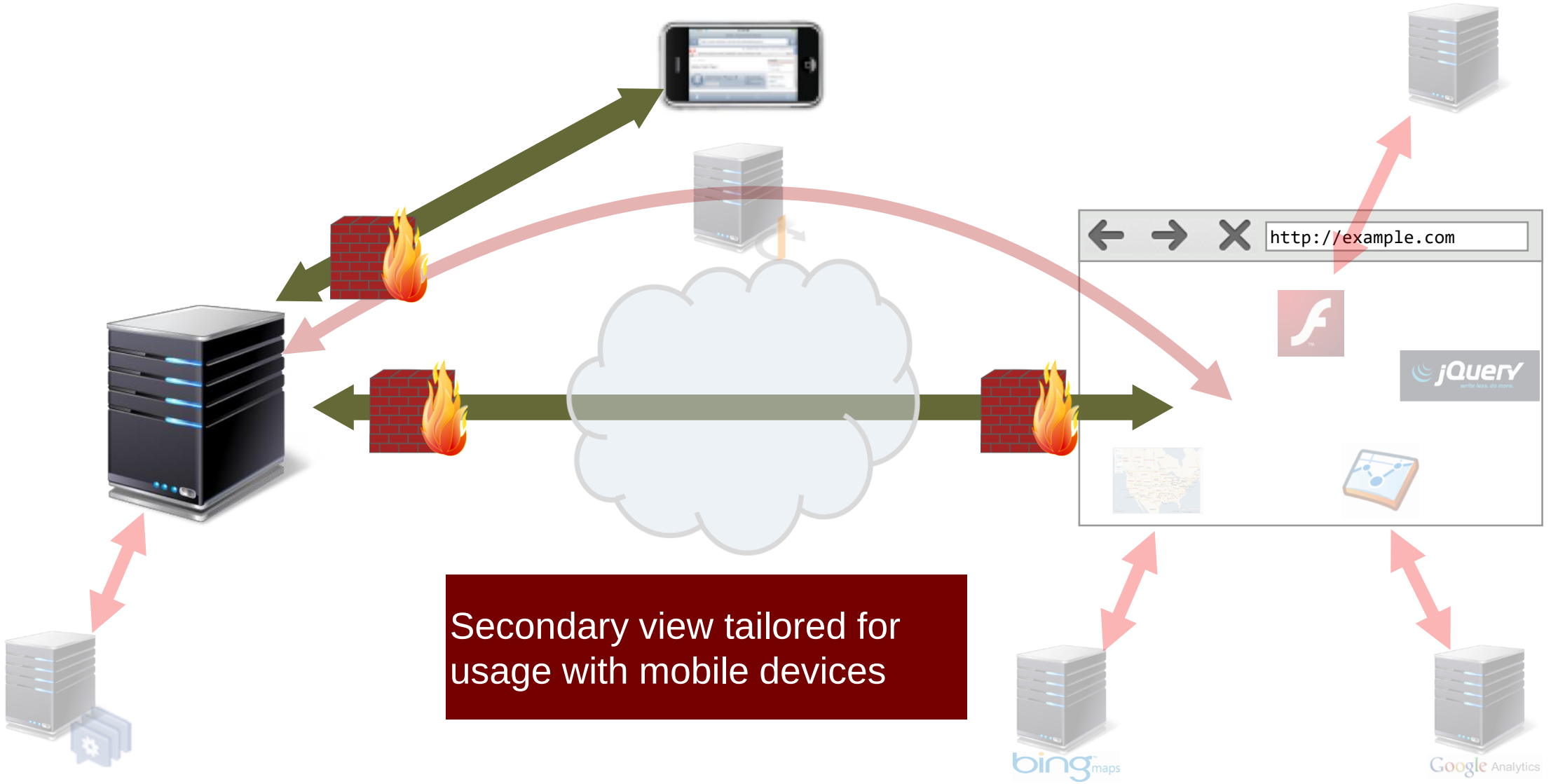
Client-side components for cross-domain communication and browser extensions

Modern Web Applications



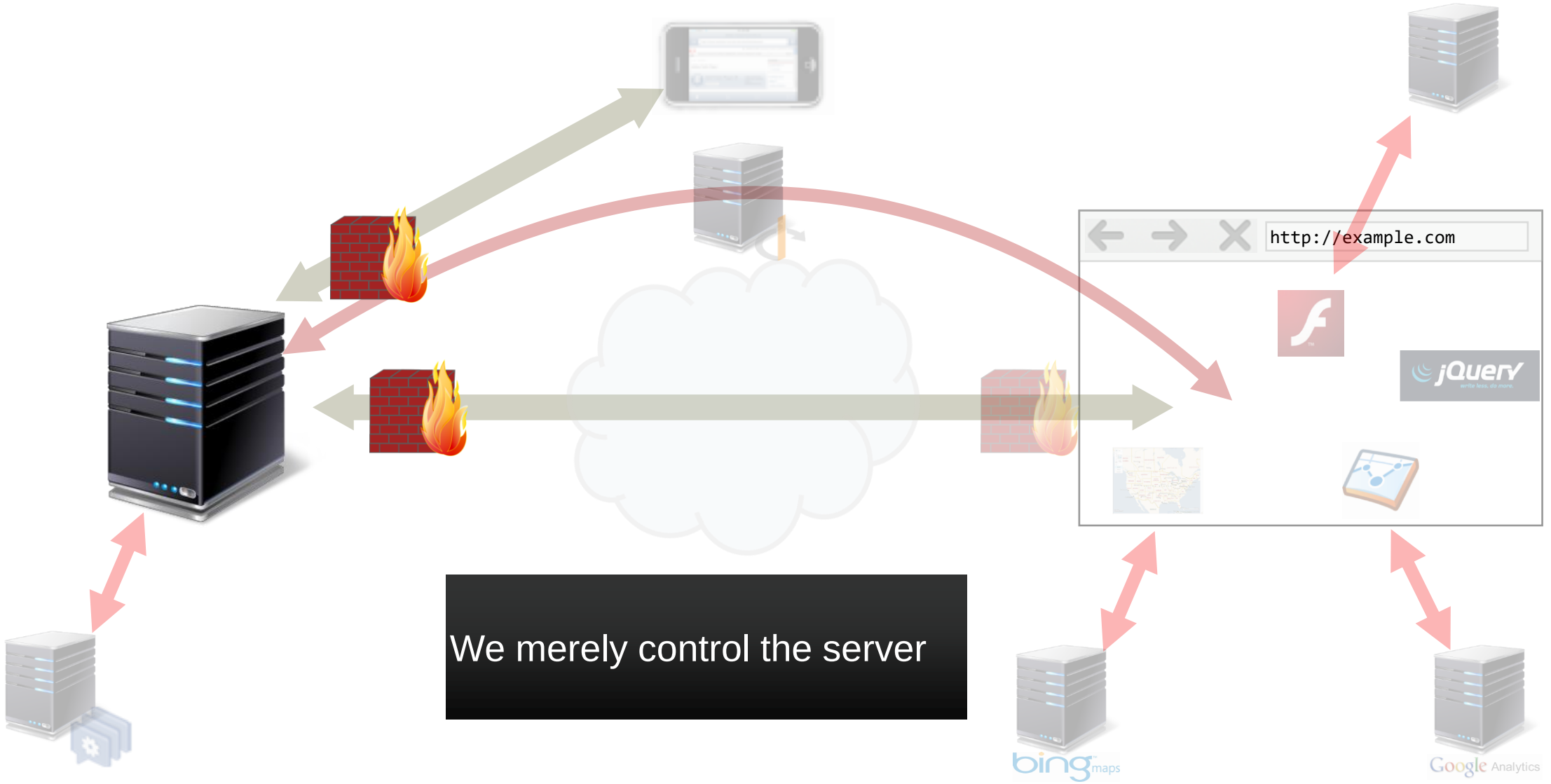
(Partial) reliance on third-party authentication providers

Modern Web Applications

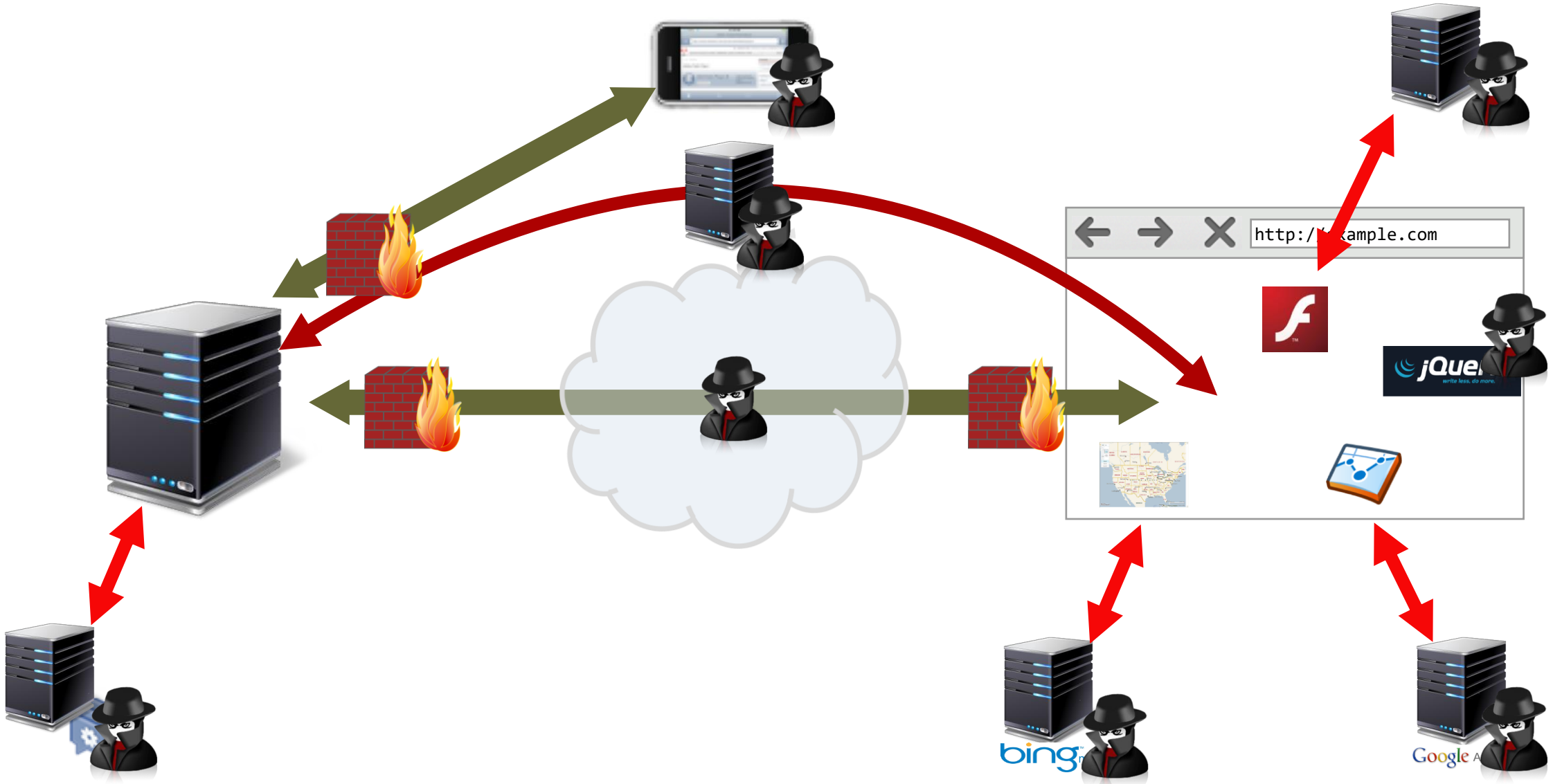


Secondary view tailored for usage with mobile devices

Security Implications



Possible Attackers on the Web



Network Attacker

- Resides somewhere in the communication link between client and server
- Tries to disturb the confidentiality, integrity, and authenticity of the connection
  - Observation of traffic (passive eavesdropper)
  - Fabrication of traffic (e.g., injecting fake packets)
  - Disruption of traffic (e.g., selective dropping of packets)
  - Modification of traffic (e.g., changing unencrypted HTTP traffic)
- "Man in the middle" (MITM)

Remote Attacker

- Can connect to remote system via the network
  - mostly targets the server
- Attempts to compromise the system (server-side attacks)
  - Arbitrary code execution
  - Information exfiltration (e.g., SQL injections)
  - Information modification
  - Denial of Service

Web Attacker

- Attacker specific to Web applications
- "Man in the browser"
  - can create HTTP requests within user's browser
  - can leverage the user's state (e.g., session cookies)
  - Case of "confused deputy"
- Examples
  - Cross-Site Scripting attacker: can execute arbitrary JavaScript in authenticated user's context
  - Cross-Site Request Forgery attacker: can force user's browser to execute certain operations on vulnerable site

Social Engineering Attacker

- No real technical capabilities
  - Abusing users rather than software vulnerabilities
- Can lure victim to perform certain tasks
  - Clickjacking
- May use technical measures to ease his task
  - Unicode URLs to easily fake
  - Use well-known icons to suggest "secure" sites

Summary

7

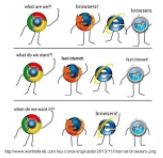
HTTP and HTML

- Web as envisioned by T. Berners-Lee
  - document-centric
  - stateless (just documents linking to one another)
  - structured (based on SGML)
  - tags for semantic interpretation
- HTTP 0.9 introduced in 1991
  - required to answer with an HTML page
  - no headers either way (introduced in 1992 though)
- HTML initially supported 18 tags
  - 11 made it into HTML4 and later versions

33

The Second Browser War (since 2005)

- Four major browser: Internet Explorer, Chrome, Firefox, Safari
- Lively competition (more or less)
- Web standards have been around long enough to focus more on **speed** and not **compatibility**
  - WebKit-based browsers (Chrome, Safari) are fastest nowadays
  - Very active development especially of JavaScript engines
- Both compatibility and speed may be roadblocks for security



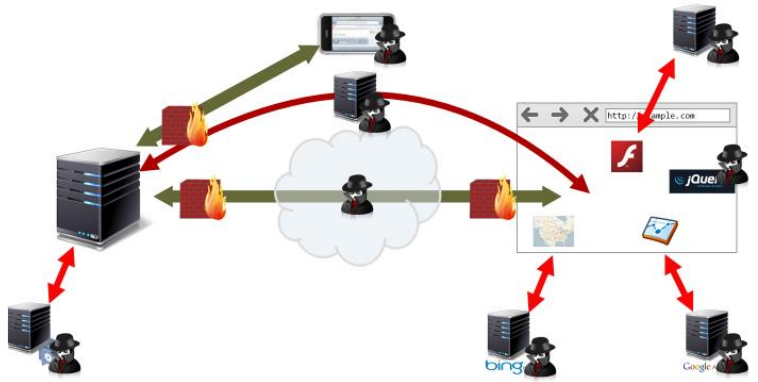
19

First Security Considerations: HTTP Authentication (1993)

- Need for authentication/authorization was recognized early on
- HTTP remained stateless
  - Authentication via HTTP header
- Not too useful for session management though

46

Possible Attackers on the Web



Credits

- Original slide deck by Ben Stock
- Modified by Nick Nikiforakis