# Rich internet applications: security of HTML5

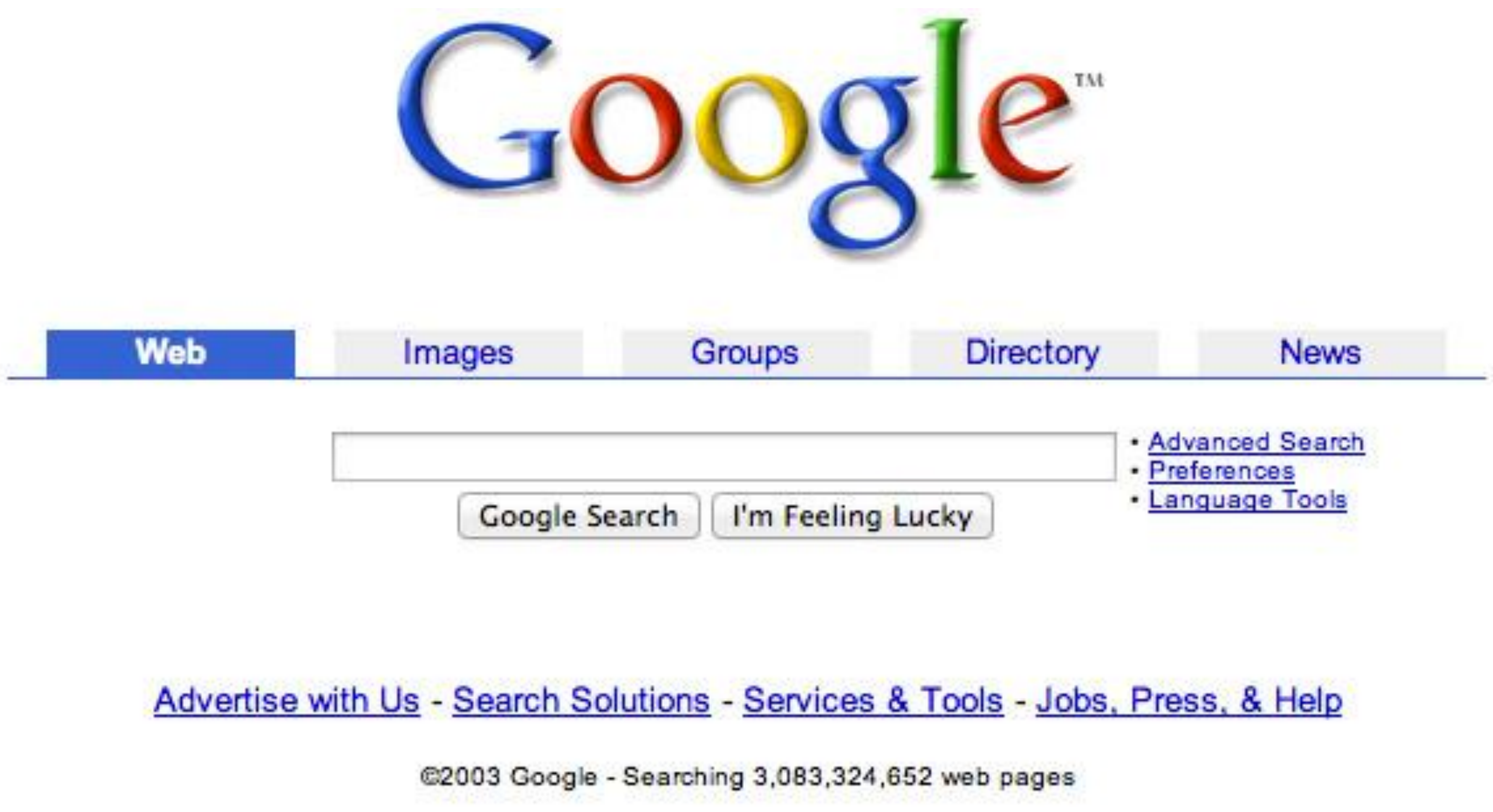
Security of IT infrastructure

Michal Rjaško

## Before HTML5: Flash

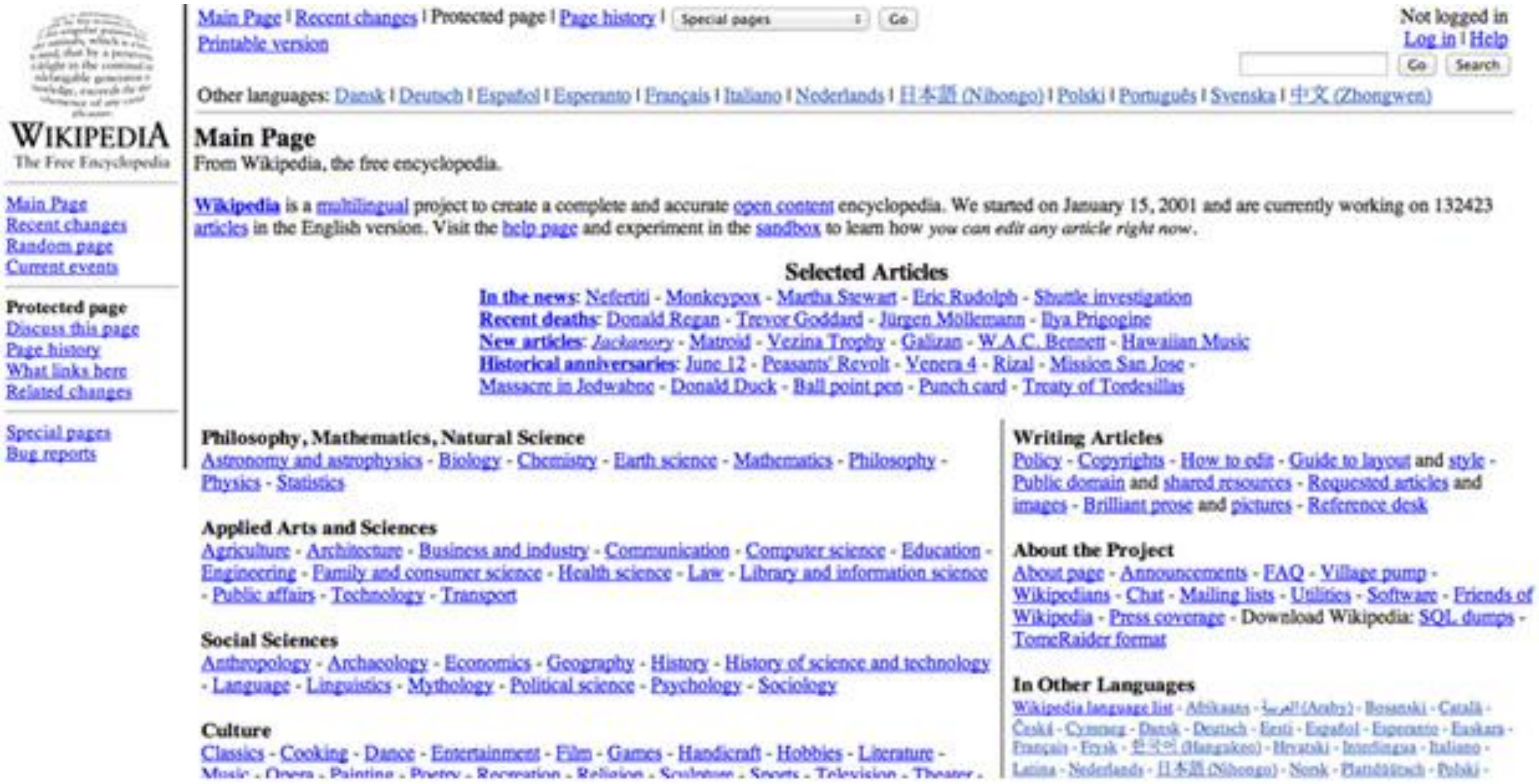
- Poor functionality of HTML environment (HTML3 / HTML4)
- Because of the poor functionality, Flash becomes very popular
  - It allows to add interactive content to websites (video, animations, rich applications)
- Inconsistent HTML among browsers – developers need to test their webs in every browser
- Poor support of w3c standards among browsers (mainly in the most popular IE <= 11)

## Websites back in 2003



©2003 Google - Searching 3,083,324,652 web pages

# Websites back in 2003



WIKIPEDIA
The Free Encyclopedia

Main Page
Recent changes
Random page
Current events

**Protected page**
Discuss this page
Page history
What links here
Related changes

Special pages
Bug reports

Main Page | Recent changes | Protected page | Page history | Special pages Go
Printable version

Not logged in
Log in | Help
Go Search

Other languages: Dansk | Deutsch | Español | Esperanto | Français | Italiano | Nederlands | 日本語 (Nihongo) | Polski | Português | Svenska | 中文 (Zhongwen)

**Main Page**
From Wikipedia, the free encyclopedia.

**Wikipedia** is a multilingual project to create a complete and accurate open content encyclopedia. We started on January 15, 2001 and are currently working on 132423 articles in the English version. Visit the help page and experiment in the sandbox to learn how *you can edit any article right now*.

**Selected Articles**
**In the news**: Nefertiti - Monkeypox - Martha Stewart - Eric Rudolph - Shuttle investigation
**Recent deaths**: Donald Regan - Trevor Goddard - Jürgen Möllemann - Ilya Prigogine
**New articles**: *Jackanory* - Matroid - Vezina Trophy - Galizan - W.A.C. Bennett - Hawaiian Music
**Historical anniversaries**: June 12 - Peasants' Revolt - Venera 4 - Rizal - Mission San Jose - Massacre in Jedwabne - Donald Duck - Ball point pen - Punch card - Treaty of Tordesillas

**Philosophy, Mathematics, Natural Science**
Astronomy and astrophysics - Biology - Chemistry - Earth science - Mathematics - Philosophy - Physics - Statistics

**Applied Arts and Sciences**
Agriculture - Architecture - Business and industry - Communication - Computer science - Education - Engineering - Family and consumer science - Health science - Law - Library and information science - Public affairs - Technology - Transport

**Social Sciences**
Anthropology - Archaeology - Economics - Geography - History - History of science and technology - Language - Linguistics - Mythology - Political science - Psychology - Sociology

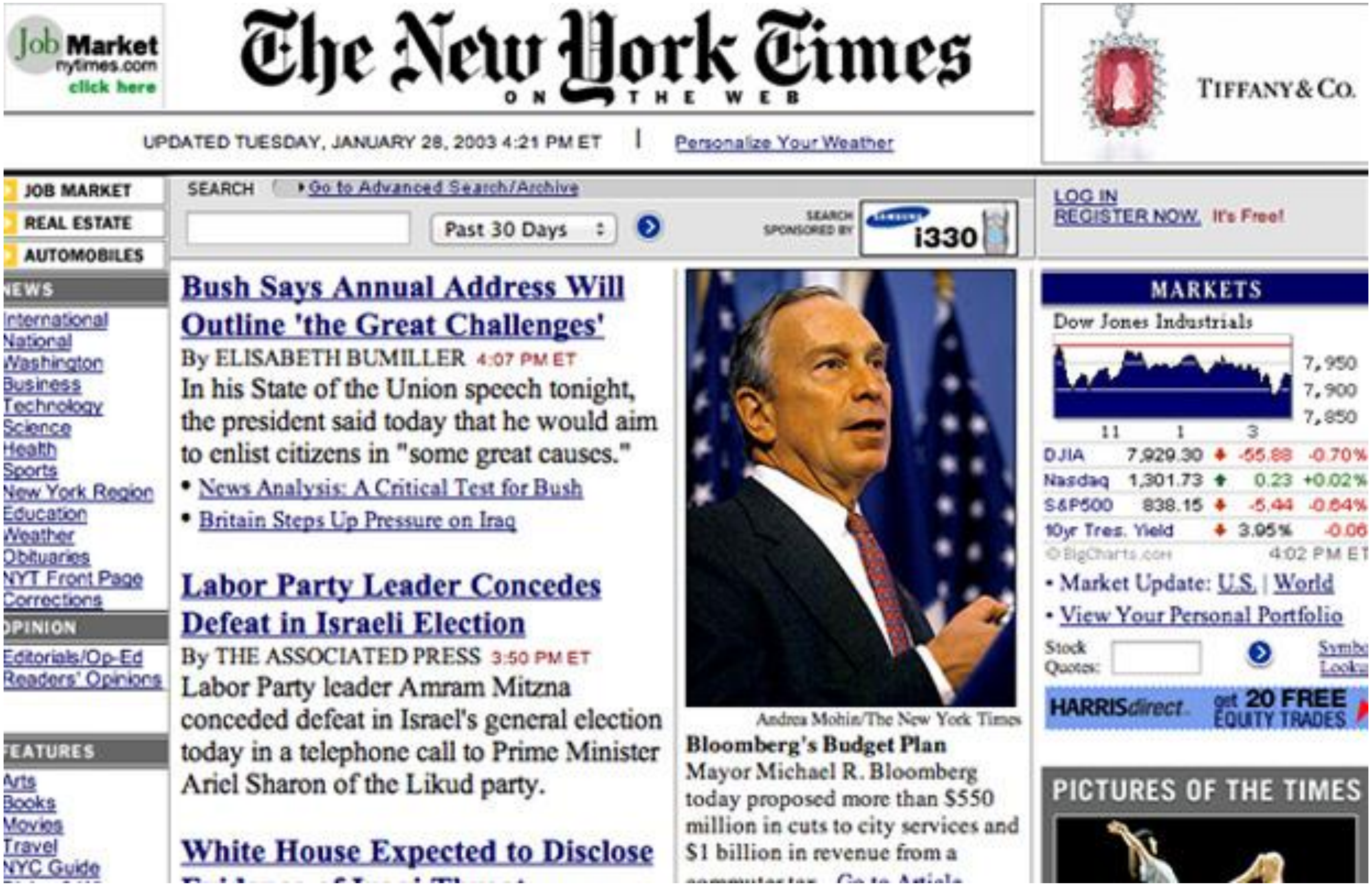
**Culture**
Classics - Cooking - Dance - Entertainment - Film - Games - Handicraft - Hobbies - Literature - Music - Opera - Painting - Poetry - Recreation - Religion - Sculpture - Sports - Television - Theater -

**Writing Articles**
Policy - Copyrights - How to edit - Guide to layout and style - Public domain and shared resources - Requested articles and images - Brilliant prose and pictures - Reference desk

**About the Project**
About page - Announcements - FAQ - Village pump - Wikipedians - Chat - Mailing lists - Utilities - Software - Friends of Wikipedia - Press coverage - Download Wikipedia: SQL dumps - TomeRaider format

**In Other Languages**
Wikipedia language list - Afrikaans - العربية (Araby) - Bosanski - Català - Česká - Cymraeg - Dansk - Deutsch - Eesti - Español - Esperanto - Euskara - Français - Frysk - 한국어 (Hangukeo) - Hrvatski - Interlingua - Italiano - Latina - Nederlands - 日本語 (Nihongo) - Norsk - Plattdüütsch - Polski -

# Websites back in 2003



Job Market nytimes.com click here

The New York Times ON THE WEB

TIFFANY & CO.

UPDATED TUESDAY, JANUARY 28, 2003 4:21 PM ET | Personalize Your Weather

JOB MARKET
REAL ESTATE
AUTOMOBILES

SEARCH ▸Go to Advanced Search/Archive

Past 30 Days

SEARCH SPONSORED BY SAMSUNG i330

LOG IN
REGISTER NOW. It's Free!

NEWS
International
National
Washington
Business
Technology
Science
Health
Sports
New York Region
Education
Weather
Obituaries
NYT Front Page
Corrections

OPINION
Editorials/Op-Ed
Readers' Opinions

FEATURES
Arts
Books
Movies
Travel
NYC Guide

**Bush Says Annual Address Will Outline 'the Great Challenges'**
By ELISABETH BUMILLER 4:07 PM ET
In his State of the Union speech tonight, the president said today that he would aim to enlist citizens in "some great causes."
- News Analysis: A Critical Test for Bush
- Britain Steps Up Pressure on Iraq

**Labor Party Leader Concedes Defeat in Israeli Election**
By THE ASSOCIATED PRESS 3:50 PM ET
Labor Party leader Amram Mitzna conceded defeat in Israel's general election today in a telephone call to Prime Minister Ariel Sharon of the Likud party.

**White House Expected to Disclose**

Andrea Mohin/The New York Times
**Bloomberg's Budget Plan**
Mayor Michael R. Bloomberg today proposed more than $550 million in cuts to city services and $1 billion in revenue from a

MARKETS
Dow Jones Industrials
7,950
7,900
7,850
11 1 3

| | | | | |
|---|---|---|---|---|
| DJIA | 7,929.30 | ↓ | -55.88 | -0.70% |
| Nasdaq | 1,301.73 | ↑ | 0.23 | +0.02% |
| S&P500 | 838.15 | ↓ | -5.44 | -0.64% |
| 10yr Tres. Yield | | ↓ | 3.95% | -0.06 |

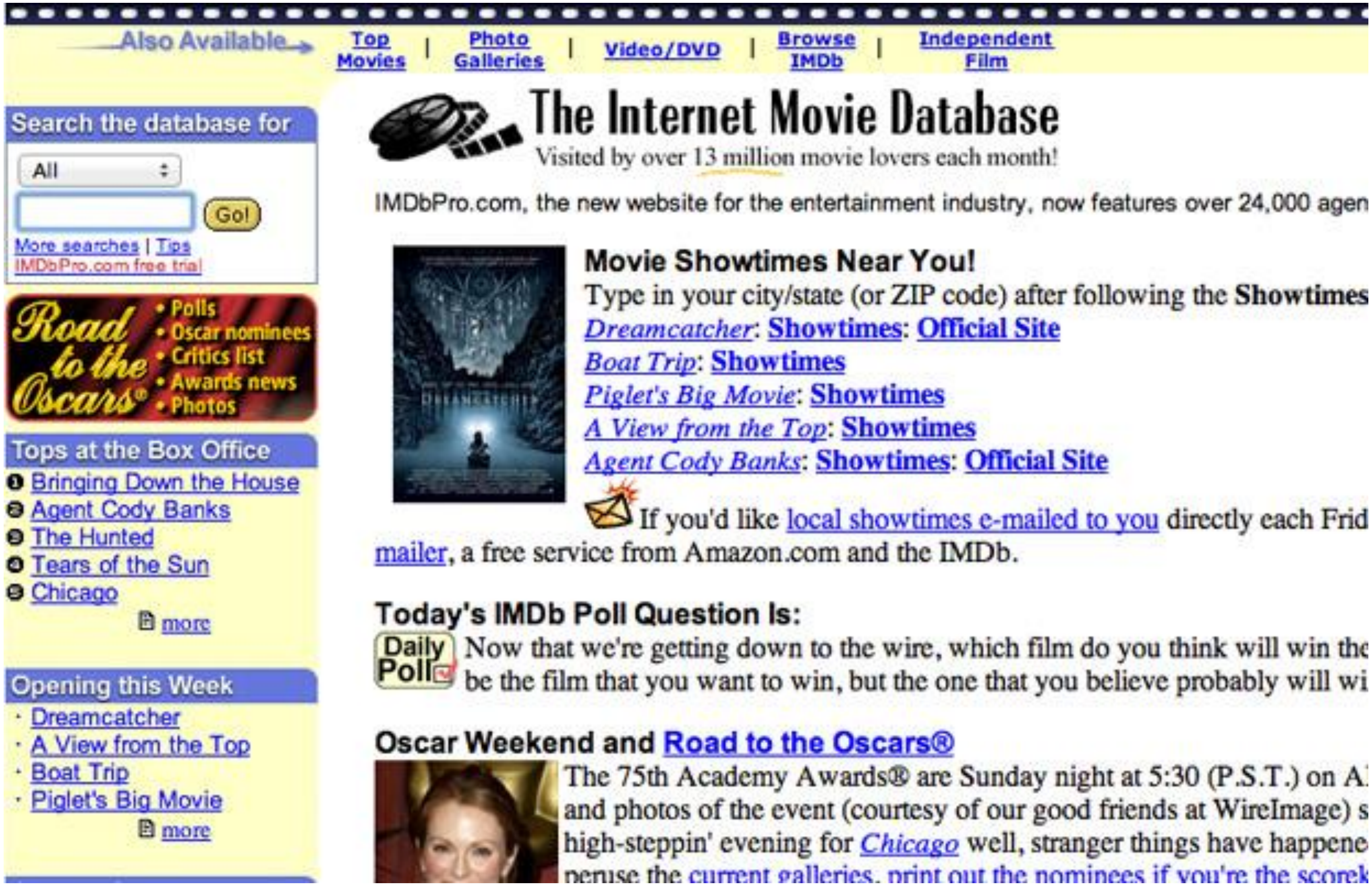
© BigCharts.com 4:02 PM ET
- Market Update: U.S. | World
- View Your Personal Portfolio

Stock Quotes:

HARRISdirect get 20 FREE EQUITY TRADES

PICTURES OF THE TIMES

# Websites back in 2003



Also Available → Top Movies | Photo Galleries | Video/DVD | Browse IMDb | Independent Film

Search the database for
All
Go!
More searches | Tips
IMDbPro.com free trial

Road to the Oscars®
• Polls
• Oscar nominees
• Critics list
• Awards news
• Photos

Tops at the Box Office
1 Bringing Down the House
2 Agent Cody Banks
3 The Hunted
4 Tears of the Sun
5 Chicago
more

Opening this Week
· Dreamcatcher
· A View from the Top
· Boat Trip
· Piglet's Big Movie
more

The Internet Movie Database
Visited by over 13 million movie lovers each month!

IMDbPro.com, the new website for the entertainment industry, now features over 24,000 agen

**Movie Showtimes Near You!**
Type in your city/state (or ZIP code) after following the **Showtimes**
*Dreamcatcher*: **Showtimes**: **Official Site**
*Boat Trip*: **Showtimes**
*Piglet's Big Movie*: **Showtimes**
*A View from the Top*: **Showtimes**
*Agent Cody Banks*: **Showtimes**: **Official Site**

If you'd like local showtimes e-mailed to you directly each Frid
mailer, a free service from Amazon.com and the IMDb.

**Today's IMDb Poll Question Is:**
Daily Poll
Now that we're getting down to the wire, which film do you think will win the
be the film that you want to win, but the one that you believe probably will wi

**Oscar Weekend and Road to the Oscars®**
The 75th Academy Awards® are Sunday night at 5:30 (P.S.T.) on A
and photos of the event (courtesy of our good friends at WireImage) s
high-steppin' evening for *Chicago* well, stranger things have happene
peruse the current galleries, print out the nominees if you're the scorek

## Flash

- Multiplatform multimedia platform ☺
- Very good for animations, graphics, videos and simple games
- Flash file can be embedded into a web page, where it is played by a browser plugin - Flash player
  - Flash Player is a third-party plugin (made by Adobe)
- Logic of flash applications and games is based on language ActionScript (currently in version 3.0)
  - ActionScript is similar to JavaScript, but contains types, classes, …

# Flash was strong platform

**It can:**

- Send HTTP requests to a different (than original) domain
- Create socket connections
- Store data on client computer (SharedObjects)
- Access camera, microphone
- Access DOM of its web page
- Execute JavaScript
- Load other Flash files

## Security of Flash

- In some ways, Flash has good security architecture
  - API controlled for communication with JavaScript
  - Secure API for communication with servers from other domain
  - Secure API for communication between two flash applications
  - ...
- However, Flash was launched using browser plugin
  - Many performance problems
  - Bugs / Problems in AVM (Actionscript Virtual Machine)

## With emerging HTML5
## Flash is dead

- Apple refused to support Flash in mobile devices
- Since version 11.1 Adobe does not develop Flash player for mobile devices
- Browsers ended their support for flash
- Adobe ended development of flash
  - Adobe AIR, mobile applications still around, but dropping

# HTML5 - history

“It must be admitted that many aspects of HTML appear at first glance to be nonsensical and inconsistent.”

“HTML, its supporting DOM APIs, as well as many of its supporting technologies, have been developed over a period of several decades by a wide array of people with different priorities who, in many cases, did not know of each other's existence.”

[w3.org/TR/html5/introduction.html#introduction]

# HTML5 - history

- **1990-1993** :: first versions derived from SGML, utilized by CERN
- **1995** :: W3C released HTML 3.0
- **1997** :: HTML 3.2 – many new features
- **1998** :: HTML 4 – used till now. DOM level 1, W3C decided to develop XHTML
- **2000** :: DOM level 2 - getElementById(), events
- **2000** :: W3C released XHTML 1.0,
  - development of XHTML2 started
- **2004** :: DOM level 3

# HTML5 - history

- **2004** :: Idea of HTML5 was born, WHATWG founded
  - W3C is not participating, but continues to develop XHTML2
- **2005** :: AJAX, XMLHttpRequest
- **2006 – 2007** :: W3C redecided, now participating in HTML5
- **2007 – now** :: WHATWG and W3C cooperate on standardization of HTML5
- **2012** :: HTML5 W3C Candidate Recommendation
- **October 2014** :: HTML5 W3C Recommendation
- **2016 ::** HTML 5.1

## HTML5 - history

- **In July 2012**, WHATWG and W3C decided on a degree of separation
  - W3C focuses on specification of a single definitive standard – “snapshot” of WHATWG
  - WHATWG continues on HTML5 as “living standard” – features can be added but not removed

# HTML5 – Current status

- HTML5 is standardized
  - After more than a decade of development
  - Still evolving standard – some features are dropped, some are added

- W3C != WHATWG
- Incomplete support of the newest features among browsers (but quite good and improving)
  - Old browsers are still in the game (Old android devices, IE, ...)

# HTML5 – Current status

- W3C specification has about 4.4 MB of text
- WHATWG specification – 707 A4 pages
- This is a lot of implementation work
- Don’t forget about
  - CSS3
  - JavaScript
  - SVG, MathML
  - Canvas, etc. etc.

# HTML5 - Security

- Many new usefull security features / APIs
- But ... some say HTML5 itself is a vulnerability
- Secure implementations require:
  - Clear specifications
  - Manageable amount of work
  - Thorough and diverse testing
  - Fast and precise feedback loops
  - Quick and comprehensive patch deployment

## HTML5 – Security

- Inconsistent and still evolving specs
- Browsers rush for implementation
- Web developers still build buggy websites
- Necessary legacy support
  - Old browsers are still around … governments, schools, …
  - Old android phones / tablets

## HTML5

- Main goal:
  - Create a simple platform for creating interactive web applications
  - Less XML strictness (compared to XHTML), more freedom
  - Emphasis on security
- HTML5 = HTML + JS + CSS (+ SVG + SQL + …)

## HTML5 vs HTML4 – New features

- New form elements
  - date, tel, color, number, email, url,...
  - Autofocus
  - Form element outside form
  - Validation on client-side
- New attributes Iframe: sandbox, seemless
- History API – allows developer to modify browser back / forward list
- Local storage
- IndexedDB – database on client

# HTML5 – New features

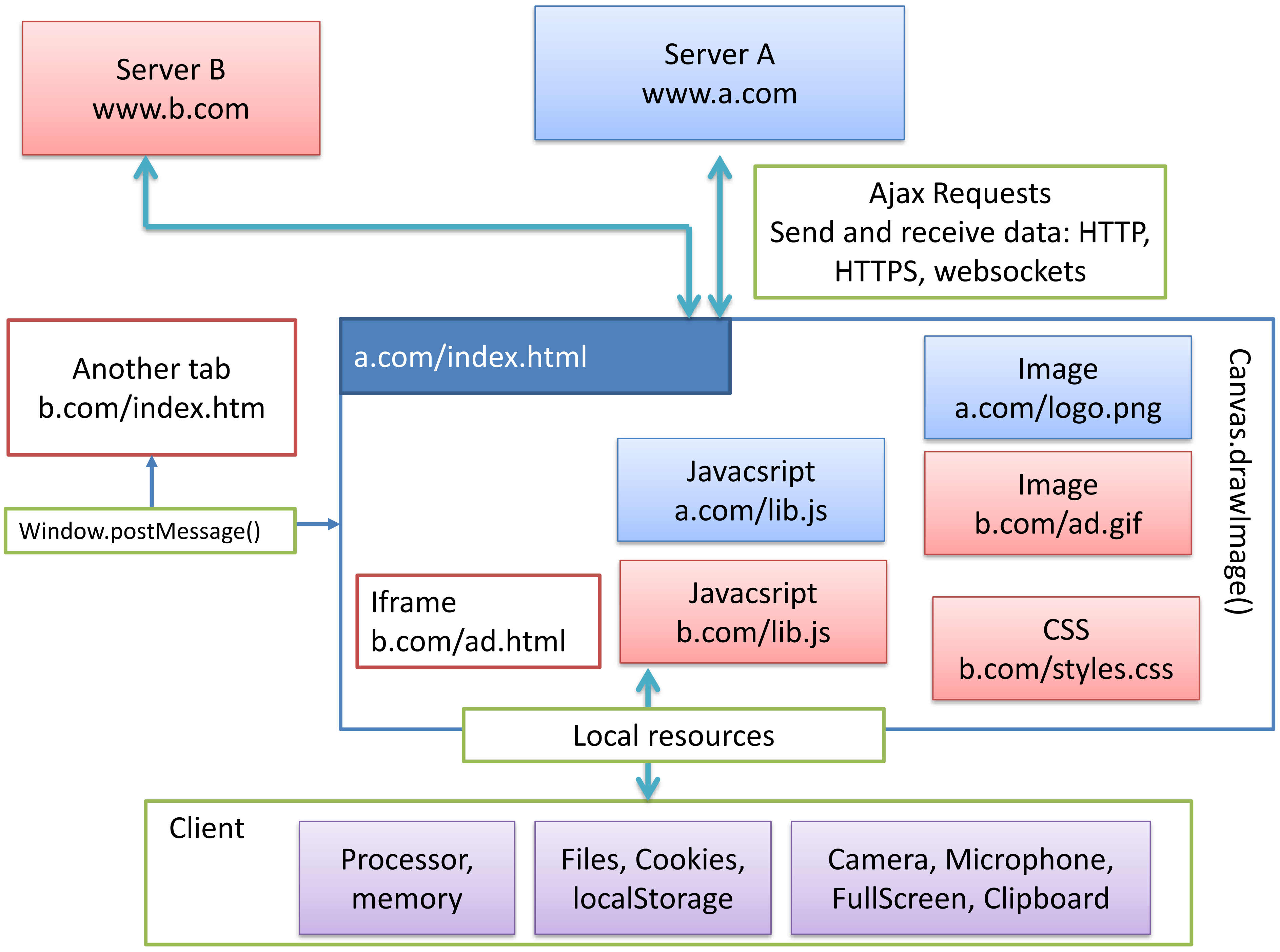
- Geolocation
- Notifications
- SVG, Canvas
- MathML
- Animations and transformations
- WebGL – 3D acceleration in browser
- Audio / Video
- Webfonts
- Offline application cache
- More semantics:
  - nav, figure, section, …
- CORS – Cross Origin Resource Sharing
- WebSockets
- WebWorkers
- HSTS: HTTP Strict-Transport-Security
- CSP: Content Security Policy
- …

## HTML5 – New features

- HTML5 contains several very usefull security improvements, but
- In general, new features of HTML5 make attacks easier
  - Great number of features – new attack scenarios will be developed in future
- It’s definitely easier to
  - Track users (Geolocation, localStorage, history API)
  - Cross-site scripting – XSS (CORS, autofocus, new form elements, …)

## JavaScript and its security

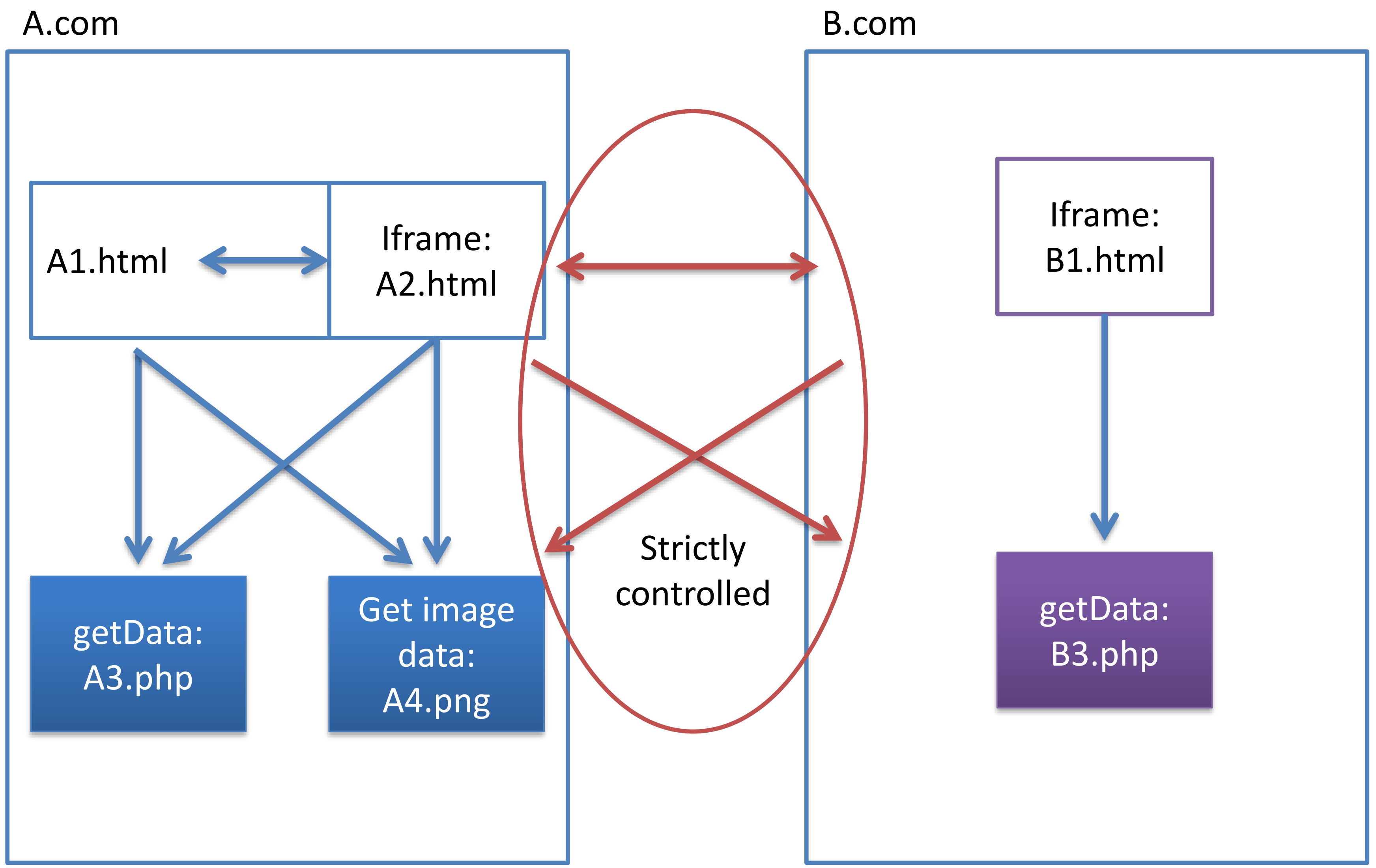
- Security from 1995
- Two main security requirements
  - Restrict malicious websites to access your computer
  - Restrict malicious website to access another website
- However, today we have all our documents in a cloud, who cares about attacker not being able to access your „My documents“ folder?
  - It‘s still important to prevent unwanted access to local resources, but things have changed since 1995


Server B
www.b.com
Server A
www.a.com
Ajax Requests
Send and receive data: HTTP, HTTPS, websockets
Another tab
b.com/index.htm
a.com/index.html
Image
a.com/logo.png
Javacsript
a.com/lib.js
Image
b.com/ad.gif
Canvas.drawImage()
Window.postMessage()
Iframe
b.com/ad.html
Javacsript
b.com/lib.js
CSS
b.com/styles.css
Local resources
Client
Processor, memory
Files, Cookies, localStorage
Camera, Microphone, FullScreen, Clipboard

# JavaScript

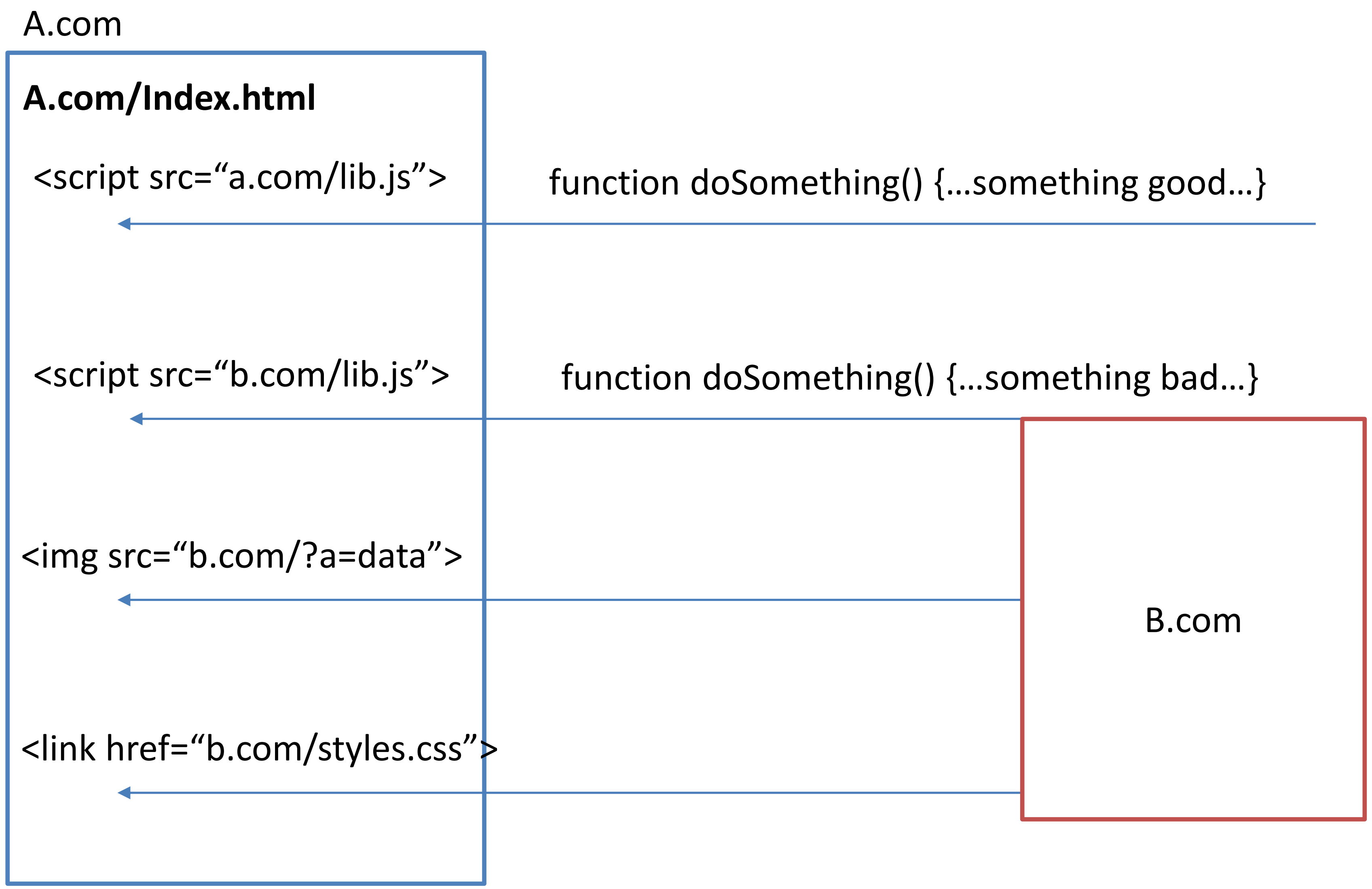
- Same Origin Policy: scripts on a web page cannot communicate with pages in a different domain
  - e.g. script loaded into www.fmph.uniba.sk cannot communicate with www.virus.com
- However, scripts loaded into the same page can interact with each other (even from different domains)
  - JavaScript is inherently global
  - Scripts can modify global variables, functions, objects etc. of other scripts.

# Same-origin policy

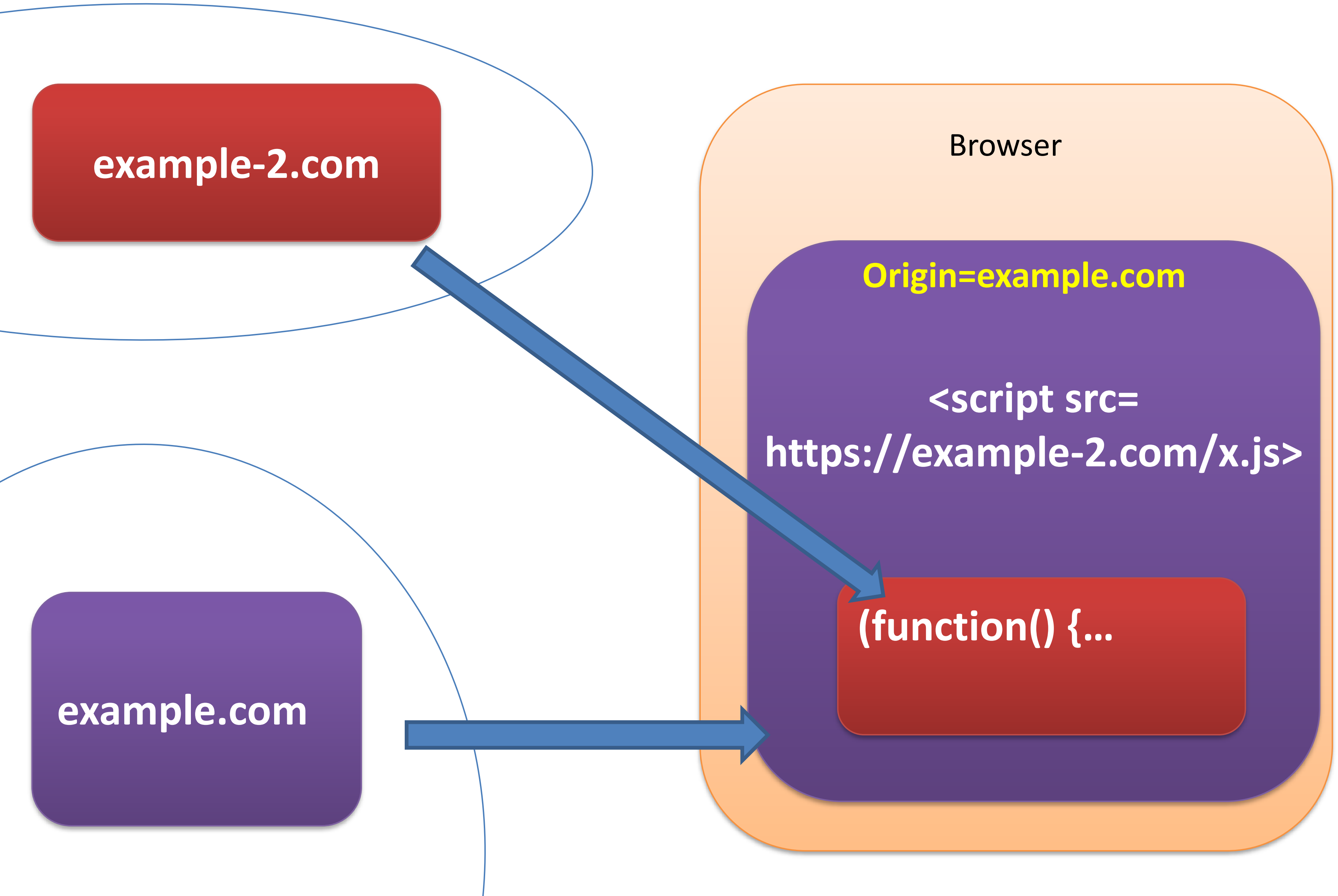


# Same-origin policy

## what is allowed (usually - see COEP)


A.com
A.com/Index.html
<script src=“a.com/lib.js”>
function doSomething() {...something good...}
<script src=“b.com/lib.js”>
function doSomething() {...something bad...}
<img src=“b.com/?a=data”>
<link href=“b.com/styles.css”>
B.com

## Circumventing same-origin policy:
**<script src="foreignOrigin">**



## AKA – “JSONP”

- “JSON with padding”

<script src=“example.com/jsonp?callback=foo”>

- Returns JSON data “padded” with a call to the function you specified.
  - i.e. returned script
    - foo({key: val,…});
- You must trust the origin of the script ... but what with 3rd party APIs

## JSONP and same-origin policy

This pattern injects somebody else’s code into your application.

Remember what the definition of XSS was?

## Cross Origin Resource Sharing (CORS)

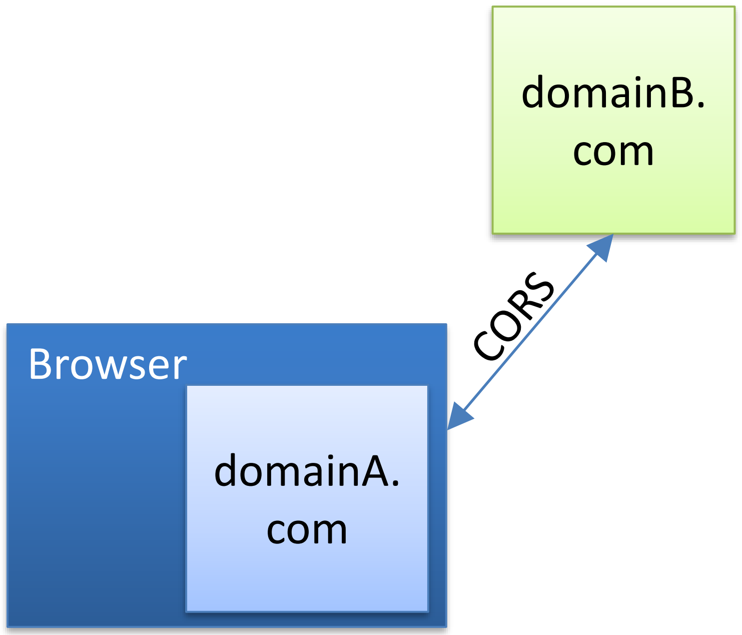
- New feature of HTML5
- Enables JavaScript to access websites from different domain
- Based on the HTTP header, browser decides whether to block the given Ajax request or not
  - Compared to crossdomain.xml in flash - flash player loads cross domain policy before the request and blocks the request without contacting the server

# Cross Origin Resource Sharing GET

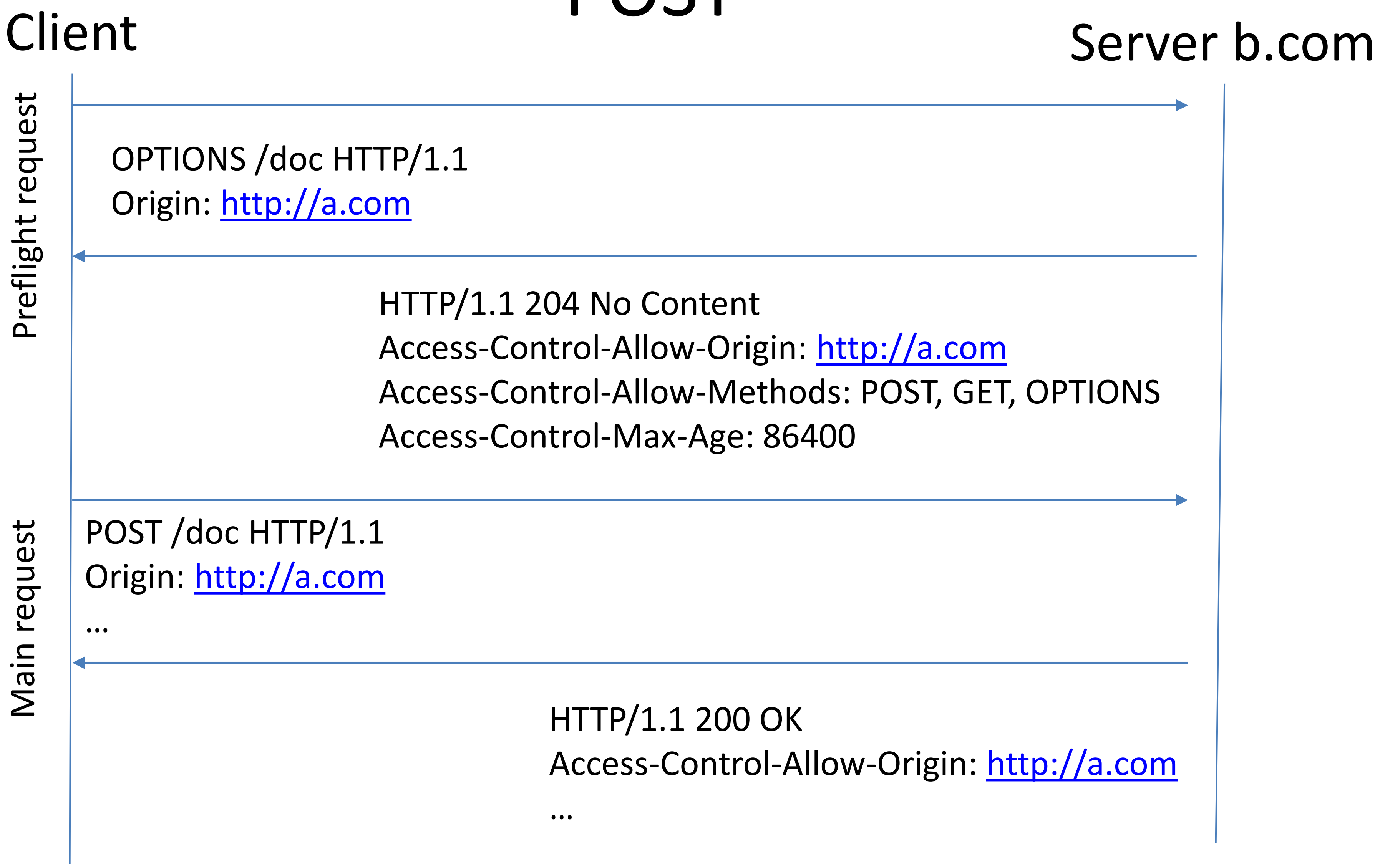
```
GET / HTTP/1.1
Host: domainB.com
Origin: http://domainA.com
…

HTTP/1.1. 200 OK
Content-type: text/html
Access-Control-Allow-Origin: http://domainA.com
…
```

[data]



# Cross Origin Resource Sharing POST



```
OPTIONS /doc HTTP/1.1
Origin: http://a.com
```

```
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: http://a.com
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Max-Age: 86400
```

```
POST /doc HTTP/1.1
Origin: http://a.com
…
```

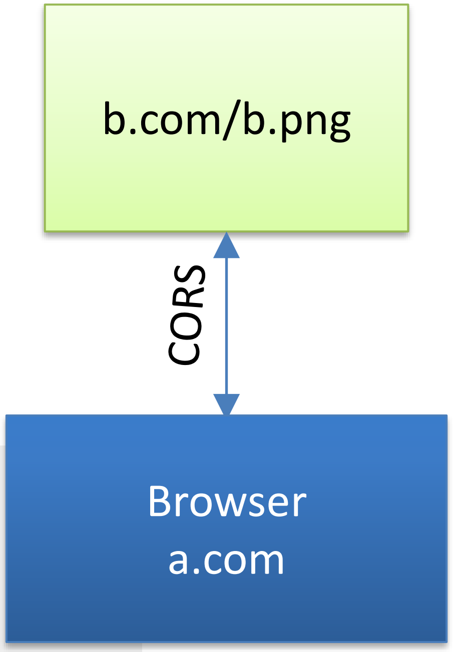
```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://a.com
…
```

# Cross Origin Resource Sharing

## images

```
GET /b.png HTTP/1.1
Host: b.com
Origin: http://a.com
…

HTTP/1.1. 200 OK
Content-type: text/html
Access-Control-Allow-Origin: http://a.com
```



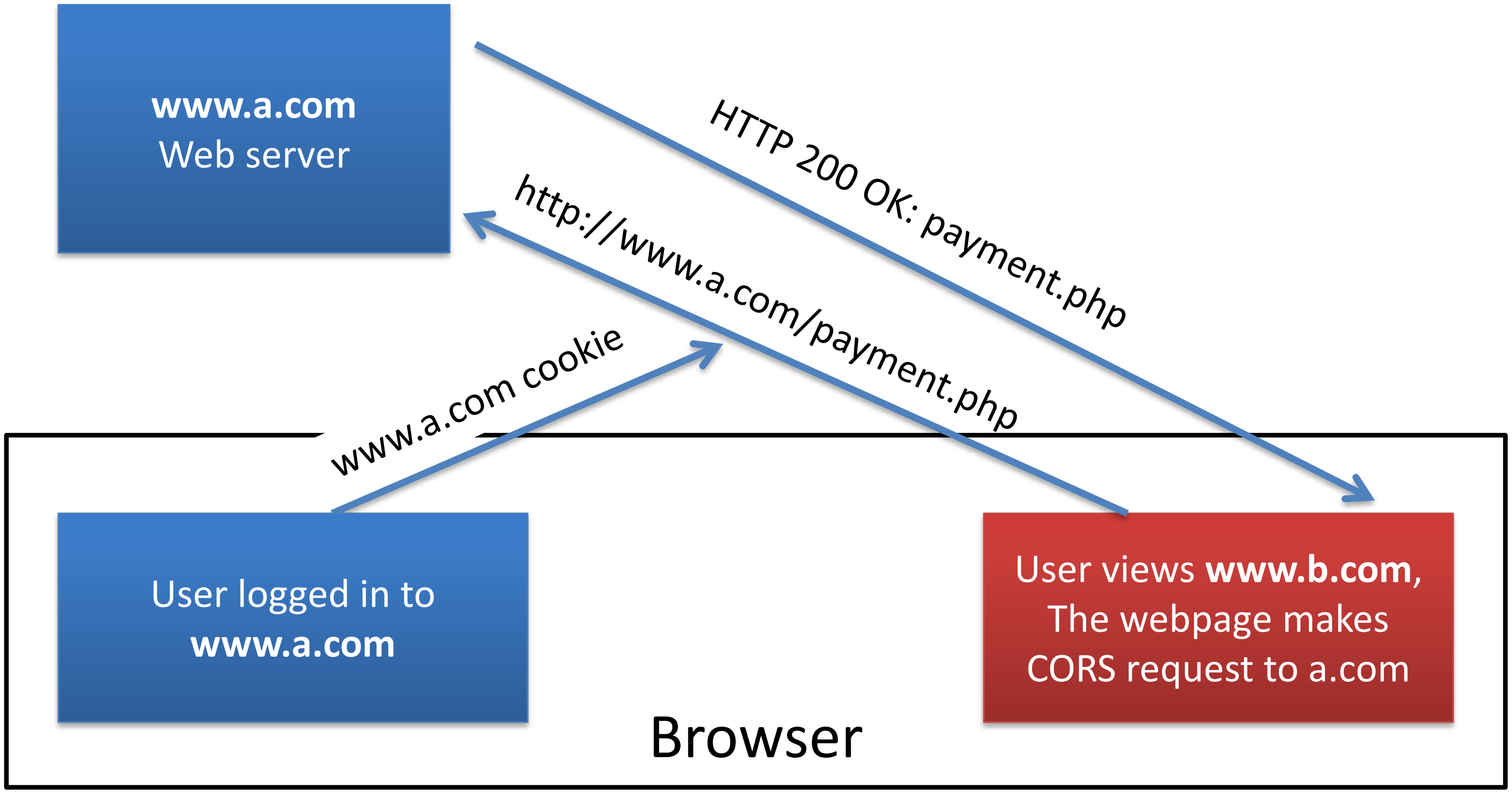
```
<img src="b.com/b.png" id="img" crossOrigin="anoynymous">
…
<script>
        Canvas.drawImage(document.getElementById('img'));
</script>

<script src="b.com/script.js" crossOrigin="anoynymous">
<link rel="stylesheet" href="b.com/script.js"
        crossOrigin="anoynymous">
```

crossOrigin="use-credentials" – cookies for b.com are sent with the request

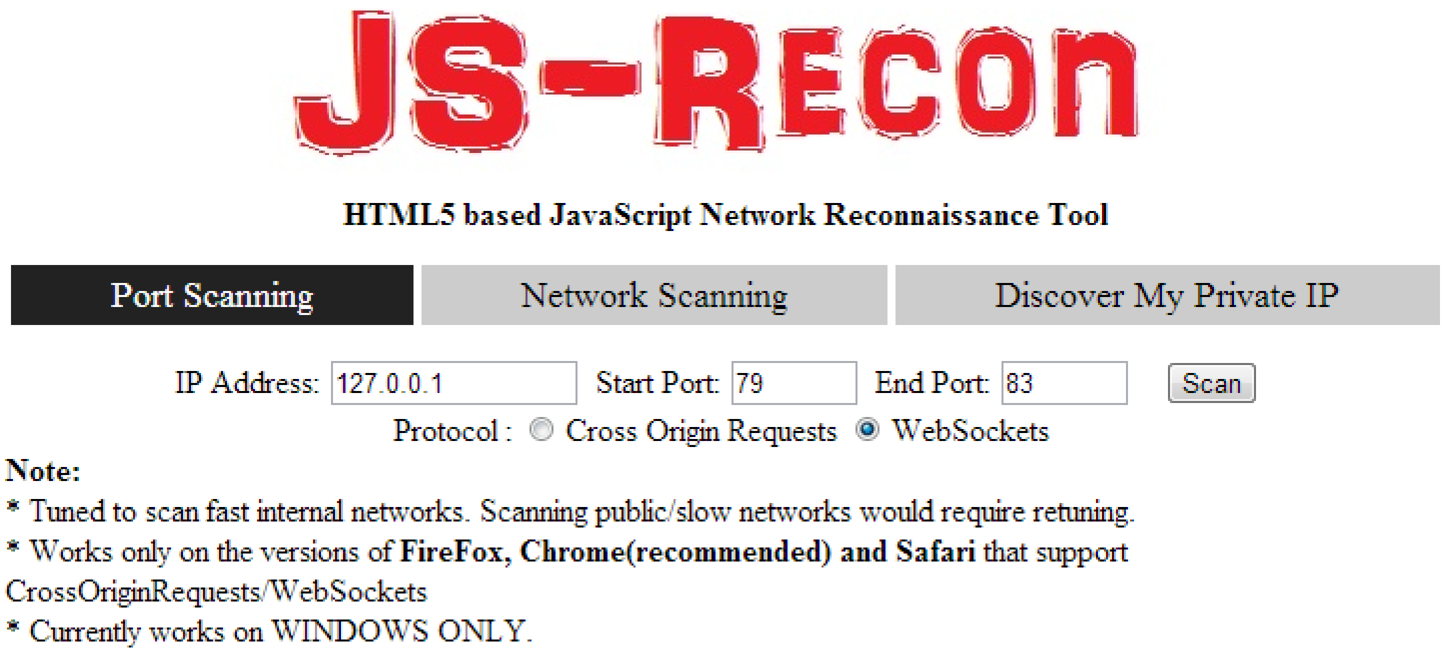
# CORS

`Access-Control-Allow-Origin: *`


**www.a.com**
Web server
HTTP 200 OK: payment.php
http://www.a.com/payment.php
www.a.com cookie
User logged in to **www.a.com**
User views **www.b.com**, The webpage makes CORS request to a.com
Browser

# Cross Origin Resource Sharing

Port / network scanner DEMO:
http://go.ba.net/nmap.html



**HTML5 based JavaScript Network Reconnaissance Tool**

IP Address: 127.0.0.1 Start Port: 79 End Port: 83

Protocol : Cross Origin Requests [✓ WebSockets]

**Note:**
* Tuned to scan fast internal networks. Scanning public/slow networks would require retuning.
* Works only on the versions of **FireFox, Chrome(recommended) and Safari** that support CrossOriginRequests/WebSockets
* Currently works on WINDOWS ONLY.

# Cross-Origin-Resource-Policy (CORP)

- The CORP response header conveys a desire that the browser blocks **no-cors*** cross-origin/cross-site requests to the given resource
  - Cross-Origin-Resource-Policy: same-site | same-origin | cross-origin

* no-cors request: without the crossOrigin attribute

`<img src=“b.com/b.png” id=“img” ~~crossOrigin=“anoynymous”~~>`

## Cross-Origin-Embedder-Policy (COEP)

- The COEP response header configures embedding cross-origin resources into the document
  - Cross-Origin-Embedder-Policy: unsafe-none | require-corp | credentialless
- Cross-Origin-Embedder-Policy: require-corp
  - Blocked: <img src="https://thirdparty.com/img.png" />
  - Allowed:
    <img src="https://thirdparty.com/img.png" crossorigin />

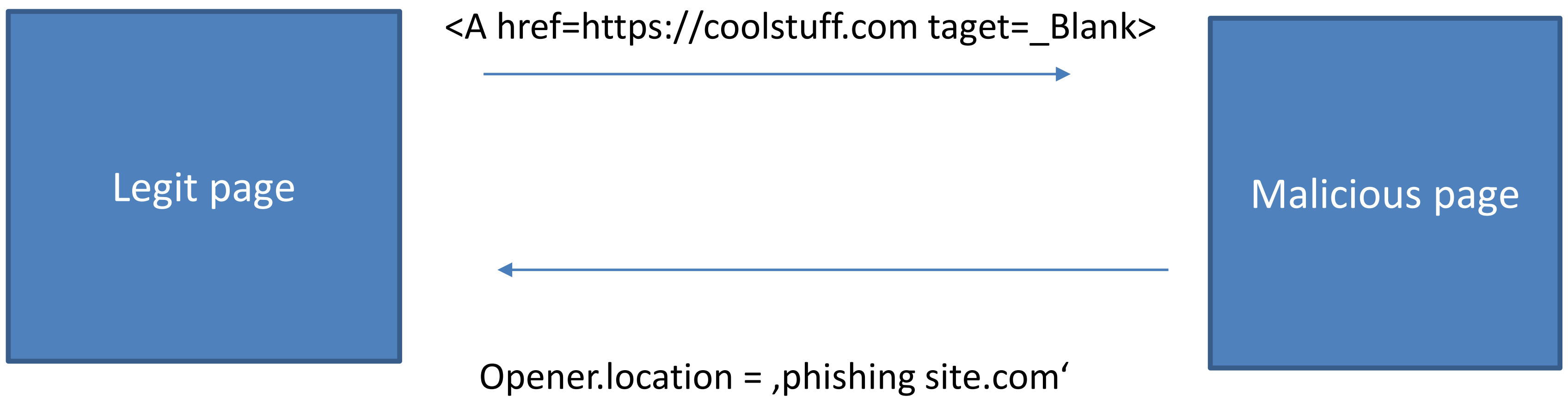
# Cross-Origin-Opener-Policy (COOP)

- The HTTP Cross-Origin-Opener-Policy (COOP) response header allows you to ensure a top-level document does not share a browsing context group with cross-origin documents.
- wnd = window.open(„http://thirdparty.com“)
  - Cross-Origin-Opener-Policy: unsafe-none
  - Cross-Origin-Opener-Policy: same-origin-allow-popups
  - Cross-Origin-Opener-Policy: same-origin
- Prevents TabNabbing
  - See next slides

# COOP + COEP and post-Spectre world

- Meltdown / Spectre vulnerabilities
  - exploit speculative execution in modern CPUs
  - allows an attacker to read sensitive data from memory locations that should be inaccessible
  - can occur through JavaScript code running in a web browser (via some APIs like SharedArrayBuffer and high-resolution timers)
- Browsers unlock SharedArrayBuffer and high-resolution timers only if COOP and COEP headers are set
  - COEP ensures that a document can only load resources which explicitly allow themselves
  - COOP forces the creation of a new browsing context group
- COOP and COEP guarantees that only cooperating resources are present in the same browsing context group (i.e. same process)
  - No risk of leeking sensitive information via such vulnerabilities

# (Reverse) TabNabbing


Legit page
<A href=https://coolstuff.com taget=_Blank>
Malicious page
Opener.location = ‚phishing site.com‘

- User clicks on a  link, new tab opens
- Malicious page redirects the original tab to a phishing site, which looks the same as the original

## (Revese) TabNabbing

```
<html>
 <body>
  <a
   href="bad.example.com„
   target="_blank">
  click me
  </a>
  <button
onclick="window.open('htt
ps://bad.example.com‘)“>c
lick me</button>
 </body>
</html>
```

```
<html>
 <body>
  <script>
   if (window.opener) {

window.opener.location =
"https://phish.example.co
m";
   }
  </script>
 </body>
</html>
```

## (Simple) TabNapping

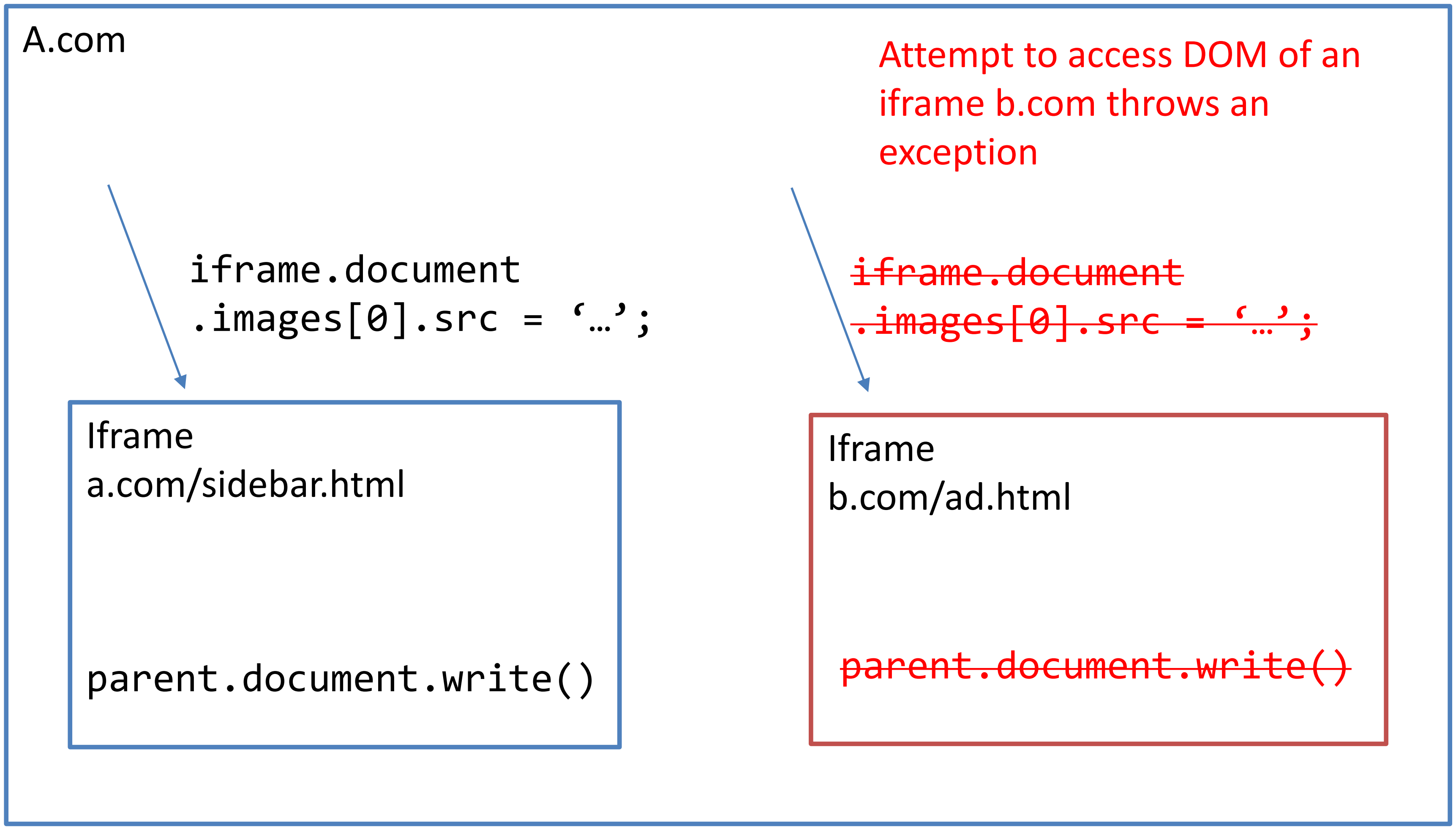
```
var windowHandle = window.open('https://goodsite.example')
// sleep for some time, and suddenly...
windowHandle.location.replace('https://hacked.example')
```

## TabNabbing - prevention

- <a href=bad.example.com target="_blank" rel="noopener noreferrer">click me</a>
- window.open(url, name, 'noopener,noreferrer‘)

- Cross-Origin-Opener-Policy: same-origin-allow-popups | same-origin
  - Response header
- Modern browsers try to implement heuristics and better defaults to combat tabnabbing attacks
  - most browsers these days treat links that have target="_blank" as rel="noopener" by default unless explicitly specified

# Iframes



Attempt to access DOM of an iframe b.com throws an exception

```
iframe.document
.images[0].src = '…';
```

~~`iframe.document .images[0].src = '…';`~~

```
parent.document.write()
```

~~`parent.document.write()`~~

… but what if we do not trust content of iframe from our domain a.com?
(e.g. user provided the content of the iframe)

# Iframe sandbox

- Possibility to put Iframe into „sandbox“
- One can specify what is allowed and what not
- `<iframe src=“infected.html” sandbox=“...”>`

| | |
|---|---|
| No sandbox attribute | Javascript runs normally |
| `sandbox attribute` | JavaScript, Flash disabled |
| `sandbox=“allow-scripts”` | JavaScript enabled<br>~~document.cookie~~<br>~~localStorage()~~<br>~~sessionStorage()~~ |

## Window.postMessage()

- Normally, scripts on different pages are allowed to communicate with each other if and only if they have the same origin
- **targetWindow.postMessage()** provides a controlled mechanism to securely circumvent this restriction (if used properly)

# Window.postMessage()

- Script at A.Com calls:

```
   wnd.postMessage(message,
         targetOrigin, [transfer]);
```

- Script at b.com receives event:

```
window.addEventListener("message",
receiveMessage,      false);

function receiveMessage(event) {
   if (event.origin !== "https://a.com")
         return;
     // ...
}
```

# Window.postMessage()

- Always specify targetOrigin
  - A malicious site can change the location of the window without your knowledge
- Always verify the senders identity
  - And check syntax of the message, since a sender you trust can have a security hole

## Content Security Policy

- Protection against XSS attacks
- In HTTP header, server specifies from which domains client can download data

**Usage**:

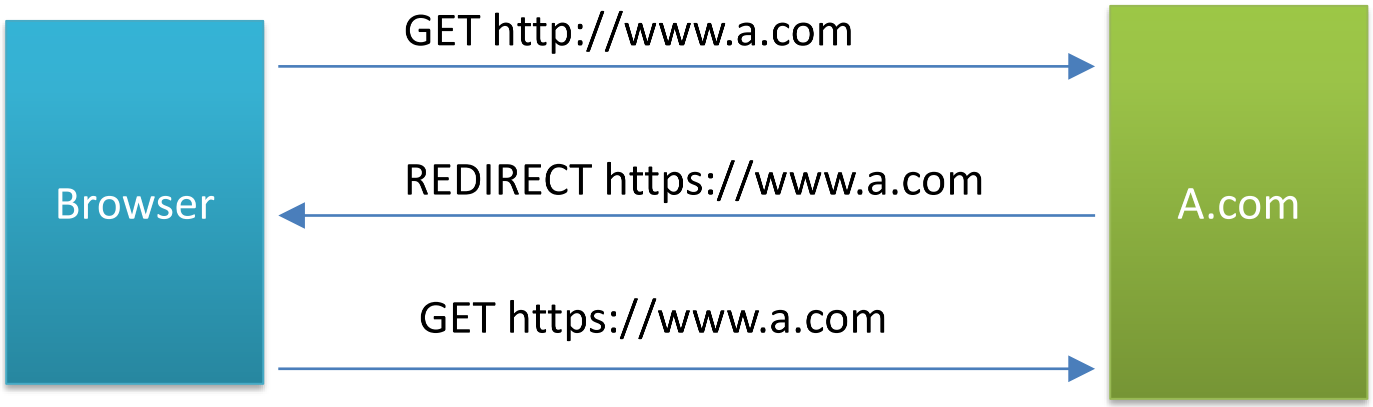
- `Content-Security-Policy: script-src 'self' https://apis.google.com`
- It's possible to specify CSP for different types of data:
  - `connect-src, font-src, frame-src, img-src, media-src, object-src, style-src`

## Content Security Policy

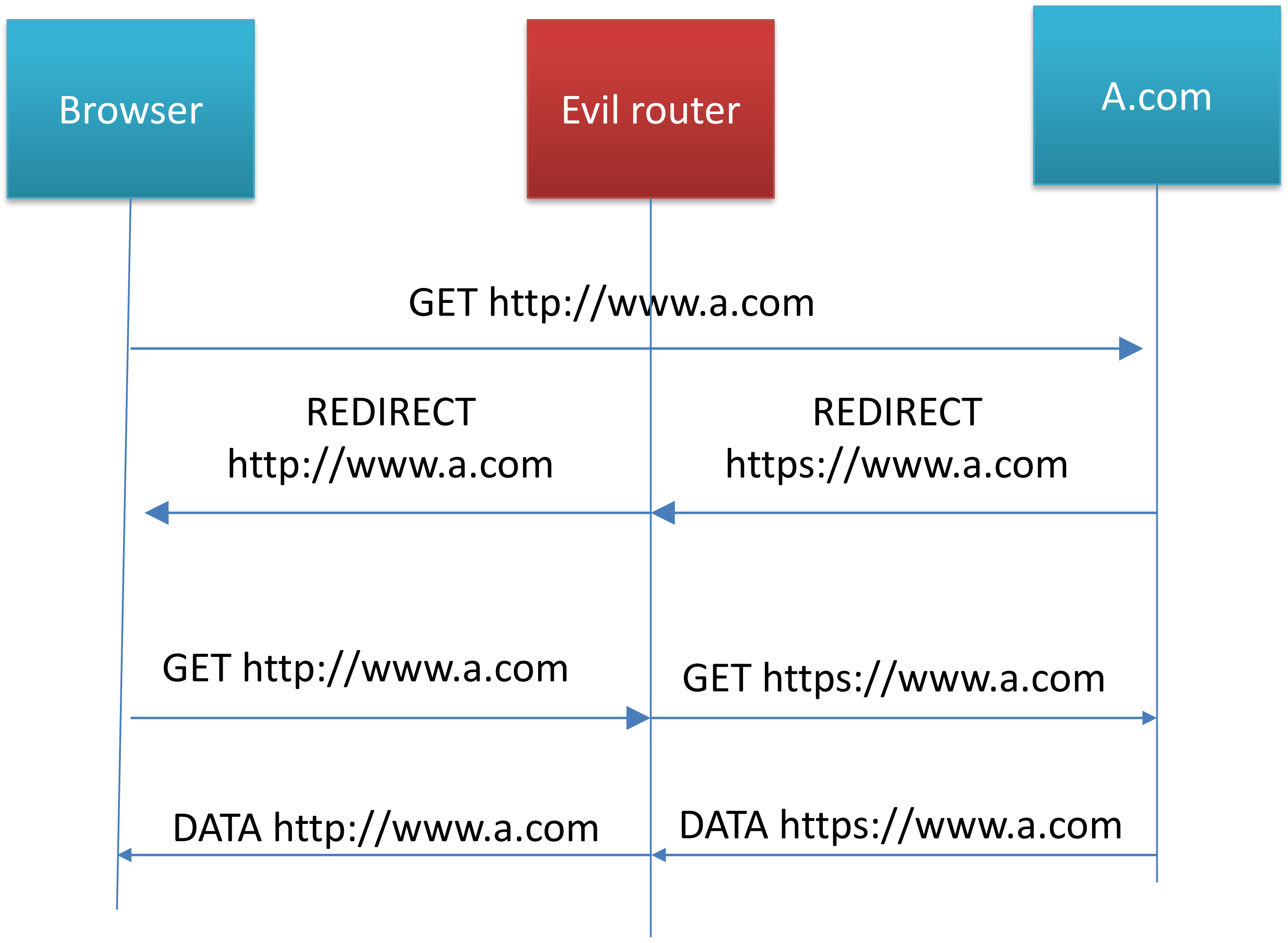
- Need to split resources into separate files
- “inline” <script> is blocked
  - It’s possible to enable inline scripts via “unsafe-inline” parameter – not recommended (does not prevent XSS attacks)
- Supports error reporting
  - report-uri /my_amazing_csp_report_parser;
  - Browser will send list of blocked resources
  - Only „reporting“ mode supported, no resources are actually blocked

# HTTP Strict Transport Security

- Users visits HTTPS sites mostly via redirect from HTTP site
  - Under some circumstances, attacker can block the redirection
    - SSL stripping attack



# HTTP Strict Transport Security
## (SSL stripping attack)



## HTTP Strict Transport Security

- HTTP header, which minimizes possibility of SSL stripping attack
- `Strict-Transport-Security: max-age=2592000;` *`includeSubDomains`*
  - Browser remembers (for specified amount of time) that it must access the website only via HTTPS
  - All requests to the site are transformed to HTTPS requests
- First request to the server (and first request after HSTS header expires) is vulnerable to SSL stripping attack

## X-Frame-Options

- Protection against „Clickjacking“ attacks
  - attacker opens the desired webpage in Iframe on top of which it places its hidden content.
  - Enables attacker to hijack clicks and key presses
- In a HTTP header, it’s possible to restrict viewing page in IFrame:
  - `X-Frame-Options: DENY | SAMEORIGIN | ALLOW-FROM origin`
- Introduced by Microsoft in IE8

# Access to local resources

- Proccessor:
  - Browsers allow to terminate script after several seconds of script unresponsiveness

- Memory:
  - No general restriction, browsers have their (configurable) limits
  - Chrome has limit cca 1gb
    - `window.performance.memory.jsHeapSizeLimit`

## Access to local resources

- Storage:
  - localStorage, sessionStorage
    - Have limits, approx. 5MB
  - IndexedDB
    - Have limits, but bigger than localStorage
  - Using HTML5 you can read and write local files
    - Reading of files
      - Only by using <input type='file' onchange="fileSelected(e)"> DOM element
      - Clicking on the element opens system dialog for file selection and gives javascript access to the selected file.
    - Writing to files:
      - Only via download link:
      - var url = window.URL.createObjectURL(data);

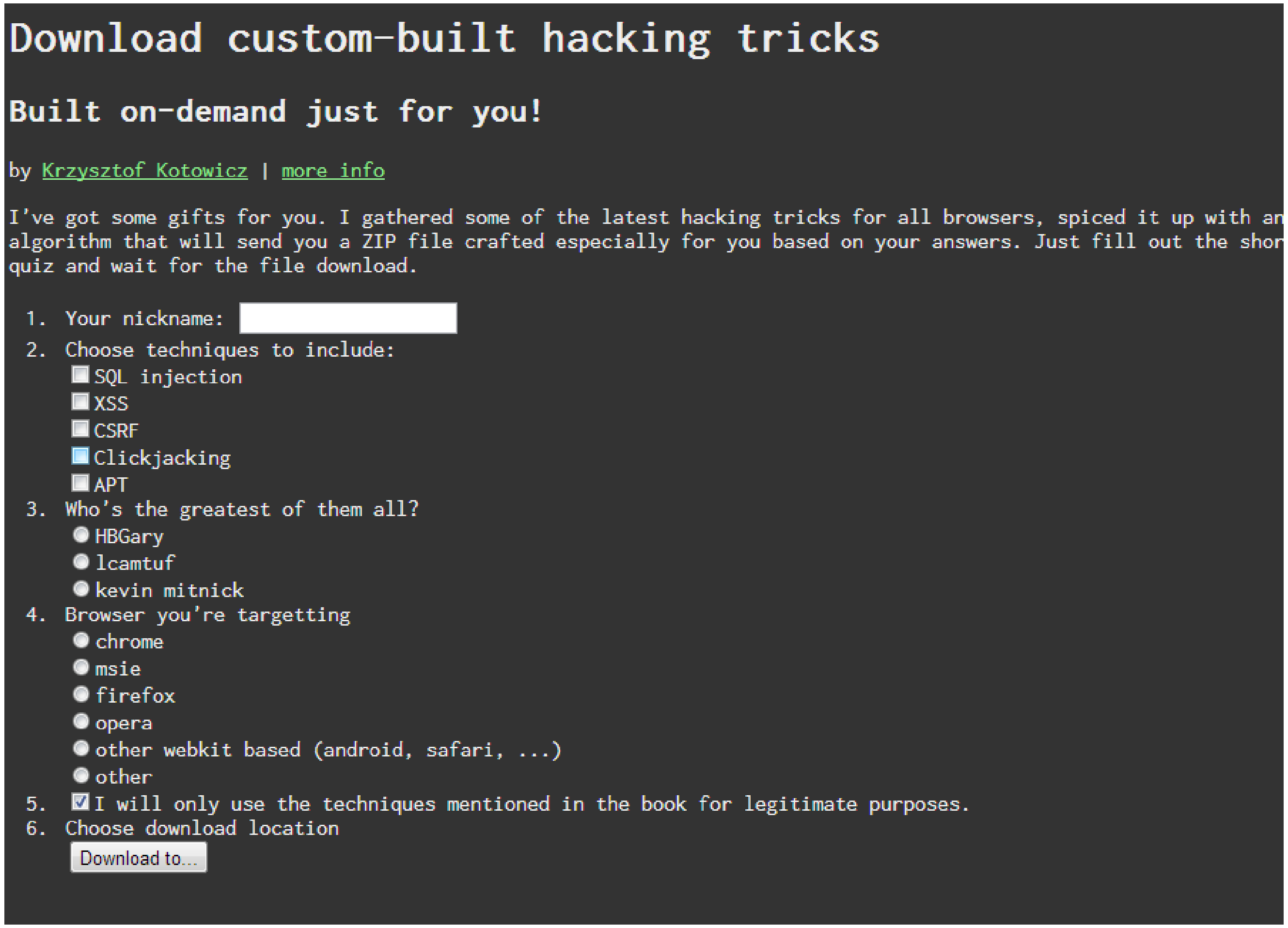
## Attack DEMO: file jacking

- It’s possible to upload an entire directory in HTML5
  - <input type=“file” directory>
- http://kotowicz.net/wu/
- Chrome “select folder” dialog can confuse users (see demo)

## Attack DEMO: file jacking

- Krystof Kotowicz created a page simulating the file jacking attack
  - Mostly visitor interested in web security
  - In 13 months 298 clients (217 IP) uploaded some folder (mostly downloads)
  - Many interesting files:
    - Downloads/#BeNaughtyLive.com/
    - Downloads/#GoLiveTrannies.com/
    - ..
  - Even private data
    - onlinePasswords.txt
    - Faktura_numer_26_2011_<company>.pdf
    - …

# Attack DEMO: file jacking



Download custom-built hacking tricks

Built on-demand just for you!

by Krzysztof Kotowicz | more info

I've got some gifts for you. I gathered some of the latest hacking tricks for all browsers, spiced it up with an algorithm that will send you a ZIP file crafted especially for you based on your answers. Just fill out the shor quiz and wait for the file download.

1. Your nickname:
2. Choose techniques to include:
   - ☐ SQL injection
   - ☐ XSS
   - ☐ CSRF
   - ☐ Clickjacking
   - ☐ APT
3. Who's the greatest of them all?
   - ○ HBGary
   - ○ lcamtuf
   - ○ kevin mitnick
4. Browser you're targetting
   - ○ chrome
   - ○ msie
   - ○ firefox
   - ○ opera
   - ○ other webkit based (android, safari, ...)
   - ○ other
5. ☑ I will only use the techniques mentioned in the book for legitimate purposes.
6. Choose download location

## Access to local resources

- Clipboard
  - Various approaches
    - Document.execCommand() depracted API
    - Clipboard API
      - Sometimes needs permission from user
      - Writing to clipboard in chrome does not need a permission
    - “paste event” – event contains data
      - Reading data possible only in “paste” event handler (i.e. when user presses CTRL+V)
- Camera, Microphone
  - Browser asks for permission

## Access to local resources

- Fullscreen
  – demo

# Covert channels

- „Covert channel“ is a mechanism enabling communication between two applications even if the security model directly restricts the communication
- Web applications contain many „covert channel“ vulnerabilities
- E.g. using CSS
  - An attacker spoofs a webpage with a list of URL addresses
  - Based on the color of the links, the attacker can determine which pages a user has visited
  - Fix: Modify function getComputedStyle() and other JavaScript functions so that they always return color as the link is not visited

# Covert channels

- Example 2: Cache – based attack
  - If an image is in browser‘s cache, it is loaded much faster
  - An attacker can spoof a webpage with several images and check how long it takes to load each image
  - Can be used to find your location (assuming that Google maps has map images near your location cached)
- Example 3: DNS cache
  - If a domain is in browser‘s DNS cache, it is loaded a bit faster
- ...

# HTML5: Conclusion

- HTML5 quite good functionality, it has replaced Flash
- HTML5 has many exciting features and extensions
  - Many features → many vulnerabilities
- Encourage users to use updated modern browsers
  - Legacy support is a pain anyway
- When creating a web application start with an established JavaScript framework (react, angularjs, vue.js)
- Adopt HTML5 security features
  - ...to protect users with HTML5-enabled browsers

## References

- Flashsec Wiki
- OWASP – Finding vulnerabilities in flash applications
- Adobe Flash Player 10 security white paper
  - http://www.adobe.com/devnet/flashplayer/articles/flash_player10_security_wp.html
- HTML 5 rocks
  - http://www.html5rocks.com/
- http://blog.kotowicz.net
- http://www.andlabs.org
- https://onlinecloudsec.com/
- https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#tabnabbing
- https://developer.mozilla.org/