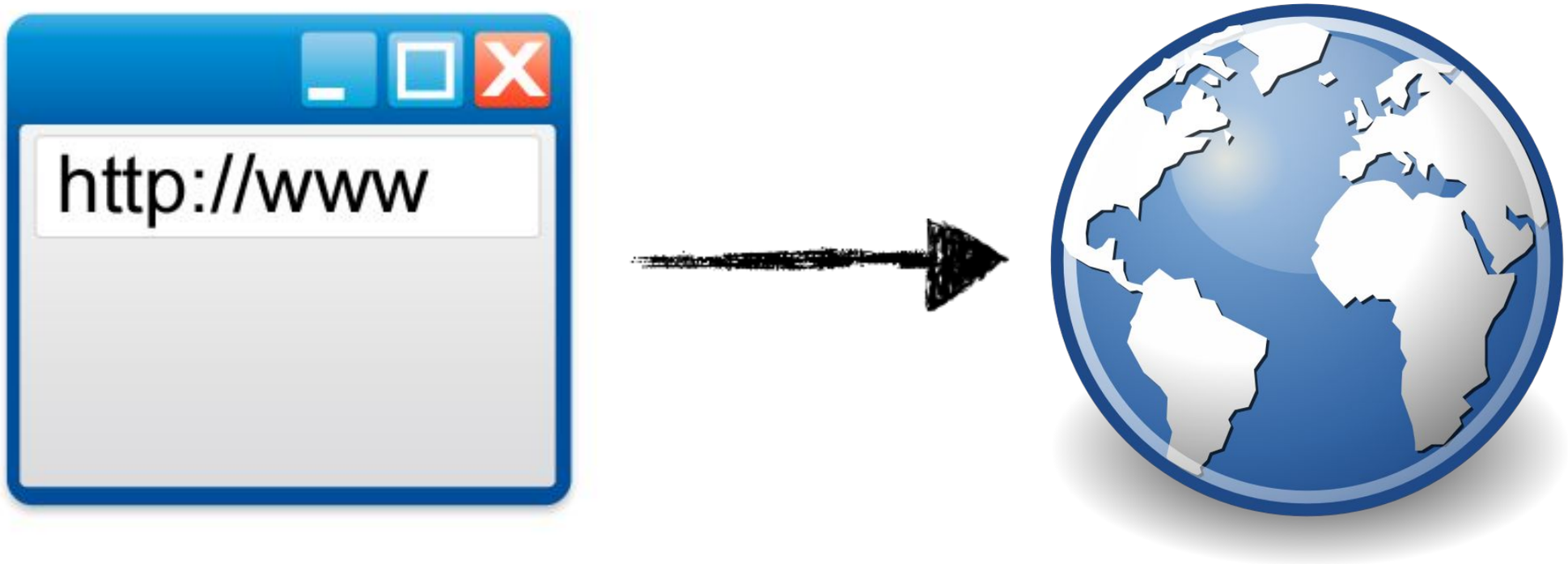
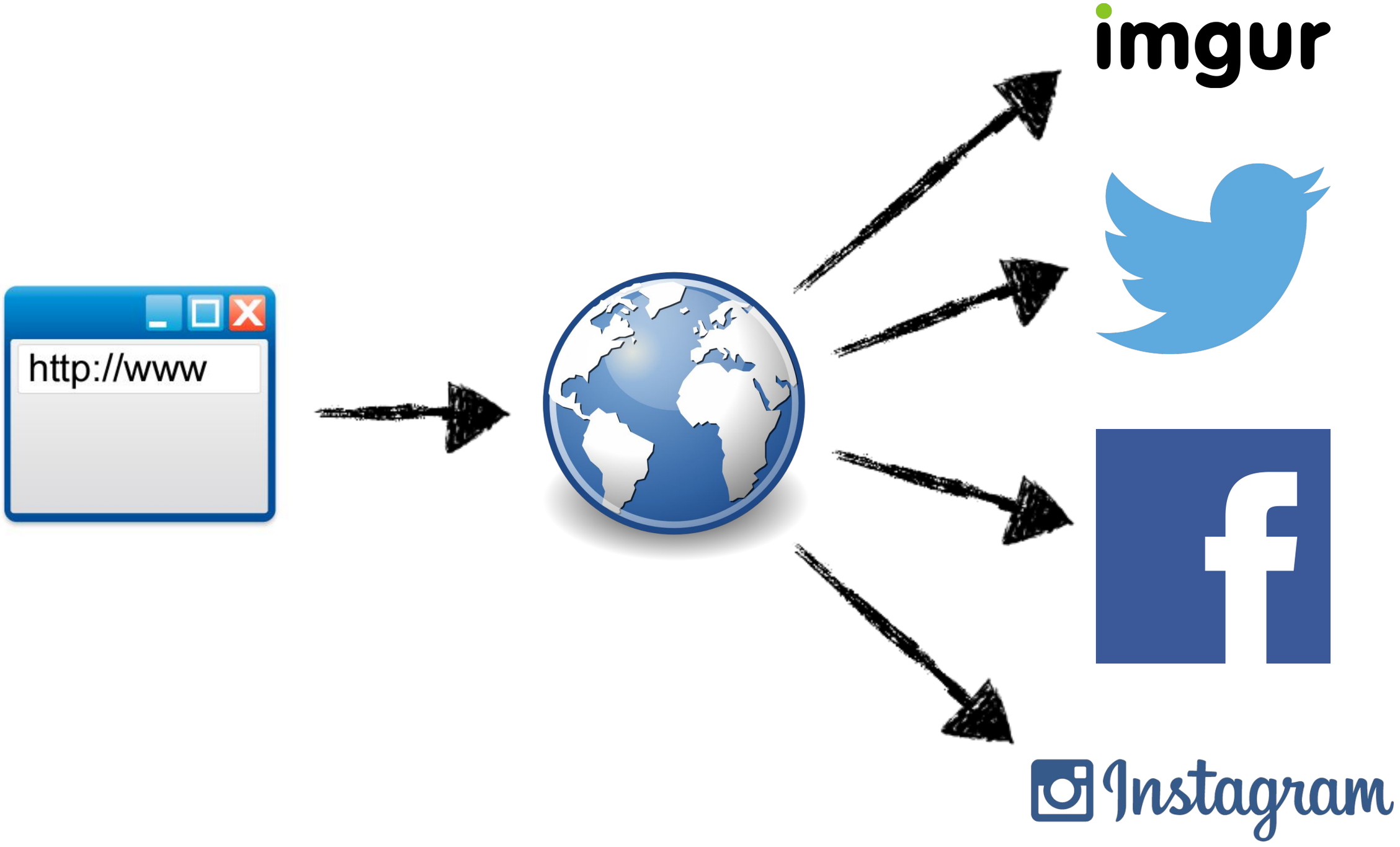
Include Me Out

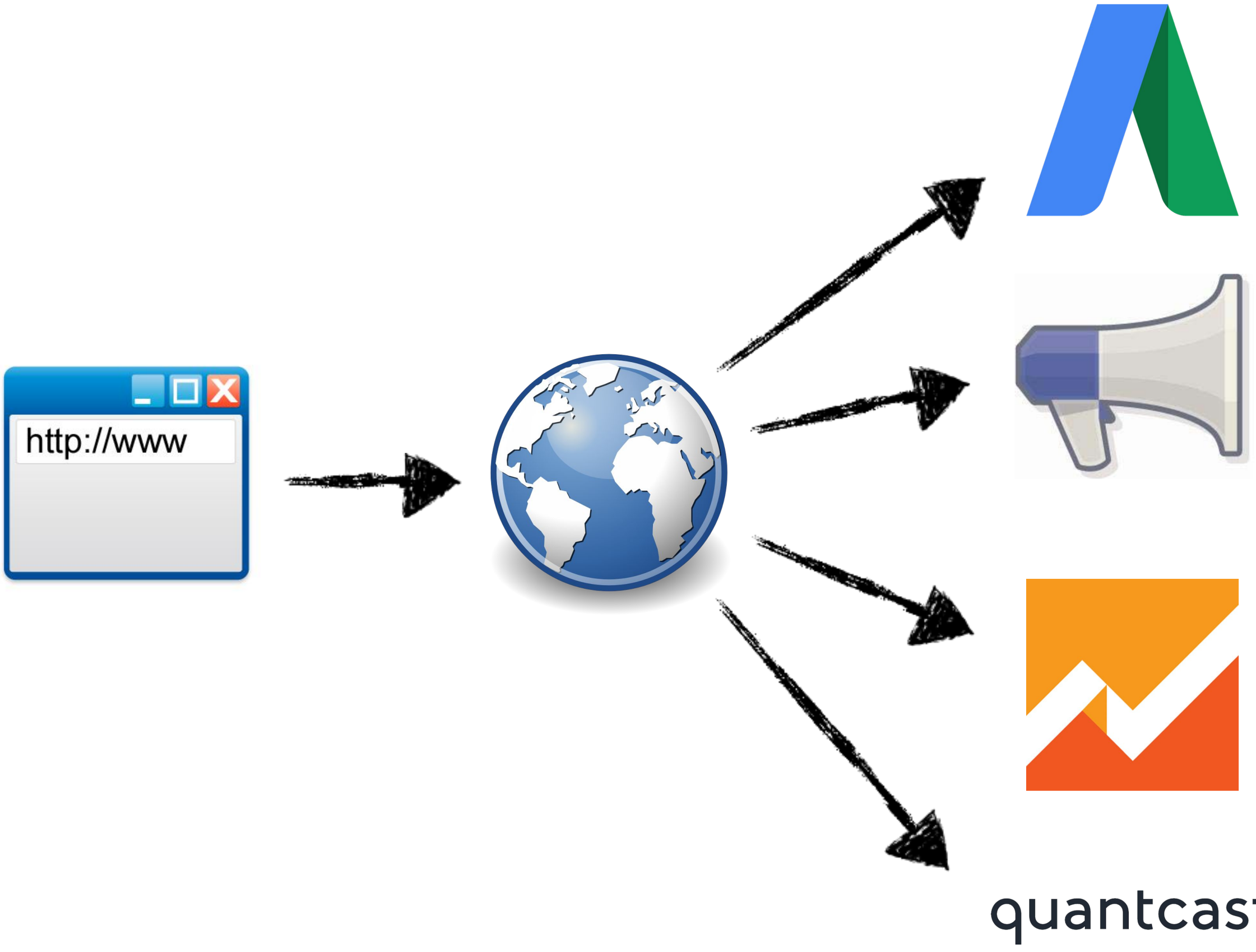
In-Browser Detection of Malicious Third-Party Content Inclusions

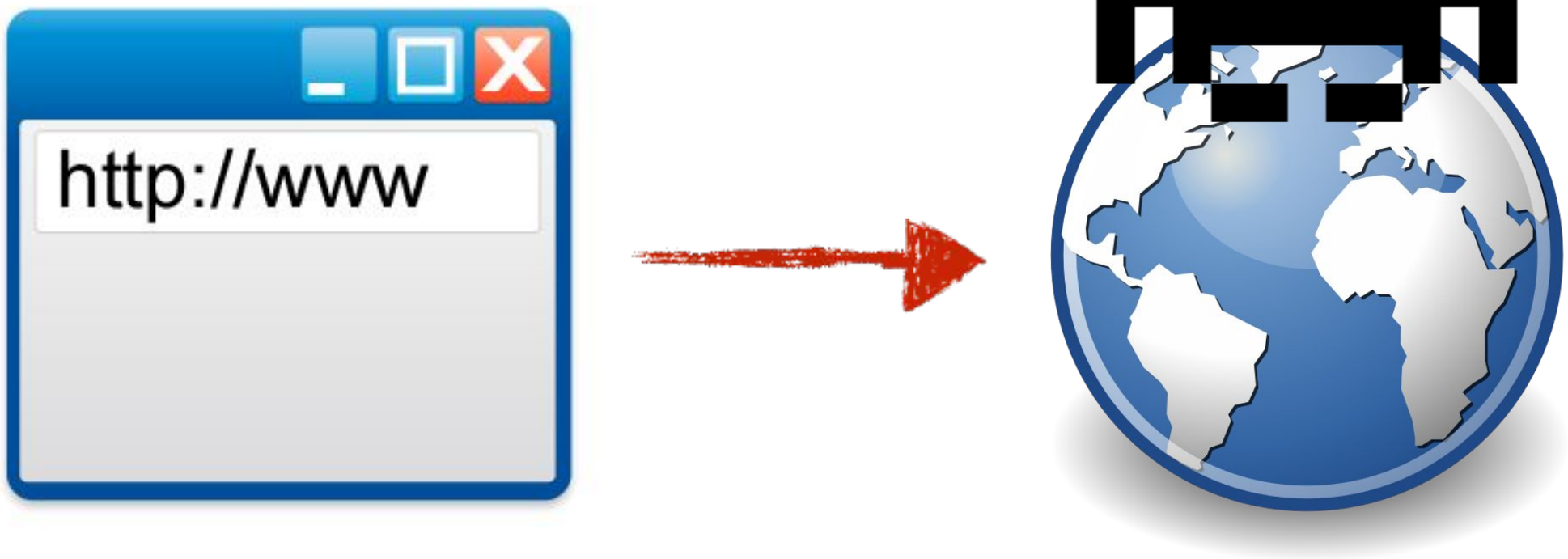
Sajjad Arshad, Amin Kharraz, and William Robertson
Northeastern University

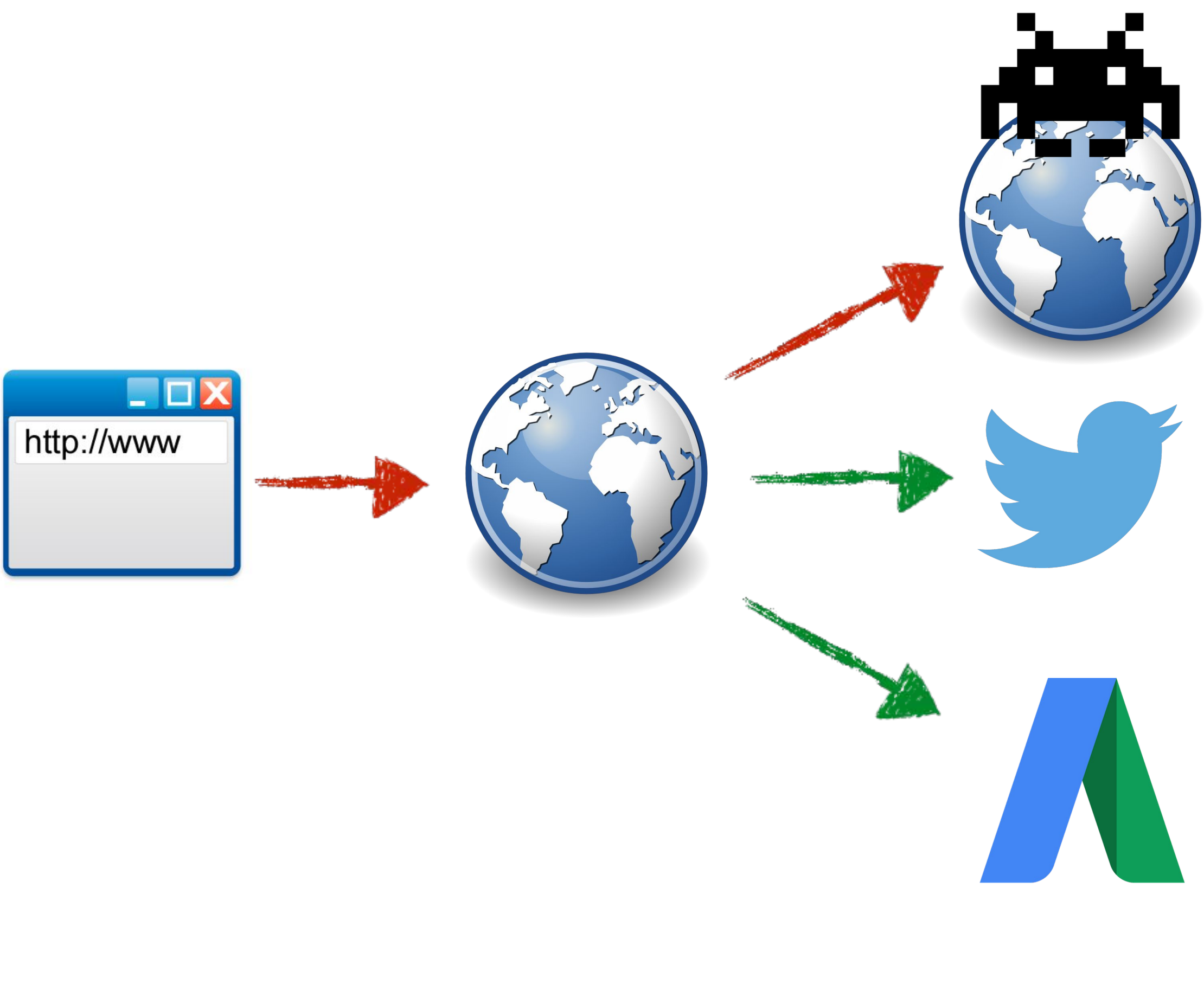
Financial Crypto
February 2016


http://www


http://www
imgur
Instagram


http://www
quantcast


http://www


http://www


http://www

Web Threats

- Drive-by downloads
- Phishing site redirection
- Click fraud
- Ad injection, malvertising

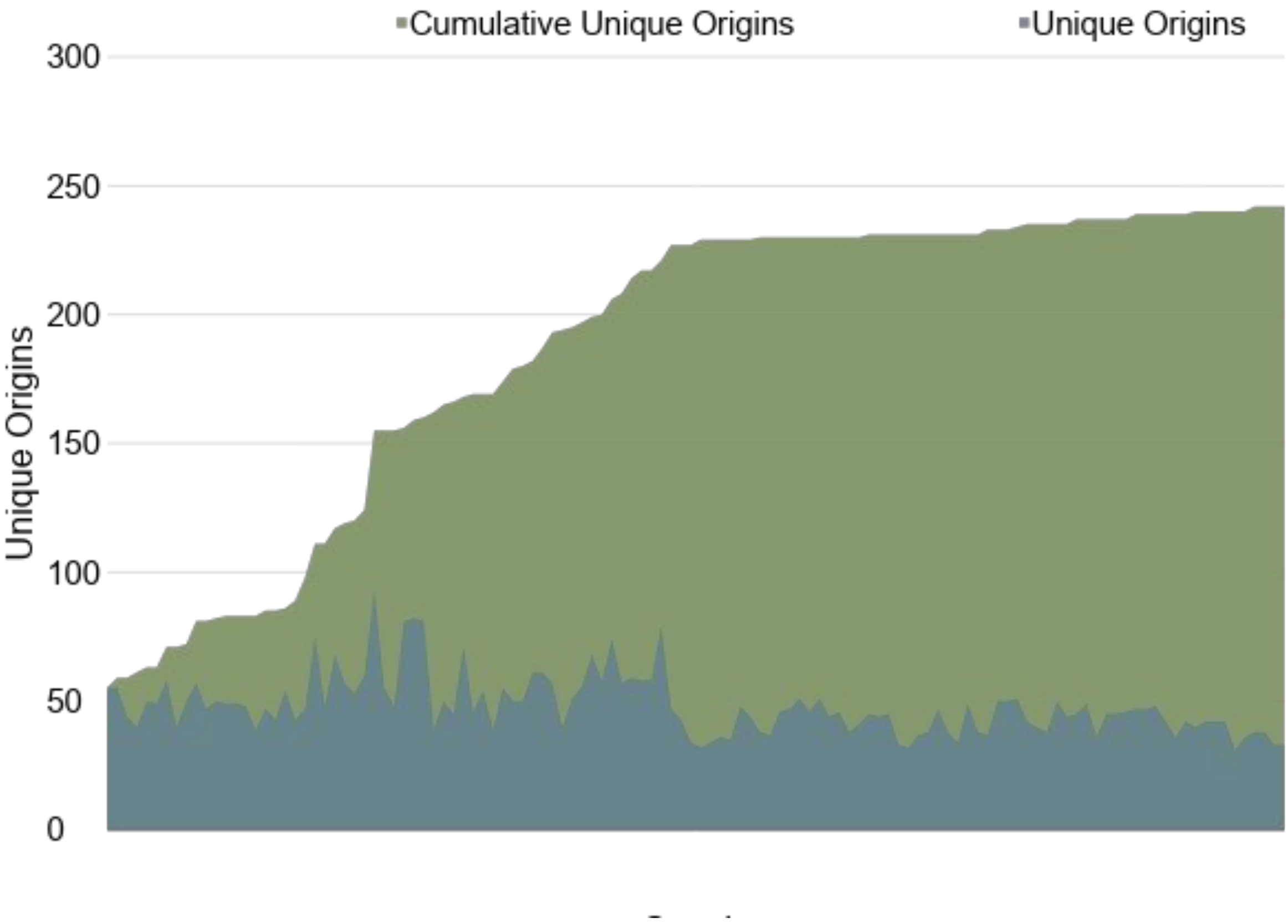
Third-Party Content Defenses

- Same-origin policy (SOP)
- iframe-based isolation
- Language-based isolation
- Policy enforcement
- Content Security Policy (CSP)

Content Security Policy

```
Content-Security-Policy:
    default-src 'self';
    img-src: img.trusted.com;
    script-src: js.trusted.com
```

- Access control policy that refines SOP
  - Sent by web apps, enforced by browsers
- Allows whitelist specification of allowed origins for classes of web resources


Cumulative Unique Origins
Unique Origins
300
250
200
150
100
50
0
Unique Origins

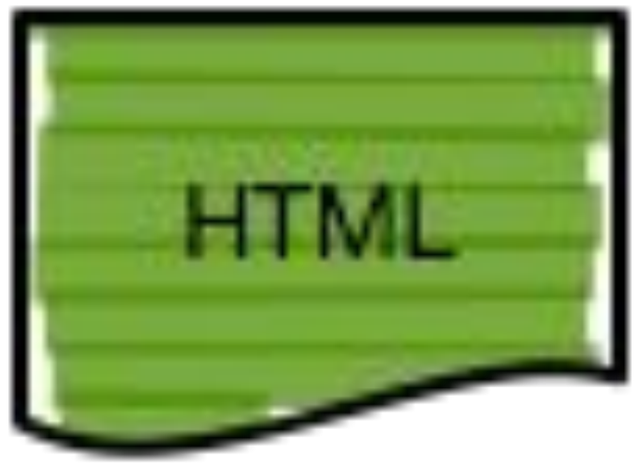
Research Questions

- Could CSP-like trust decisions be made in an automated way without manual policy specification?
- Could these decisions be made and enforced wholly at the browser?

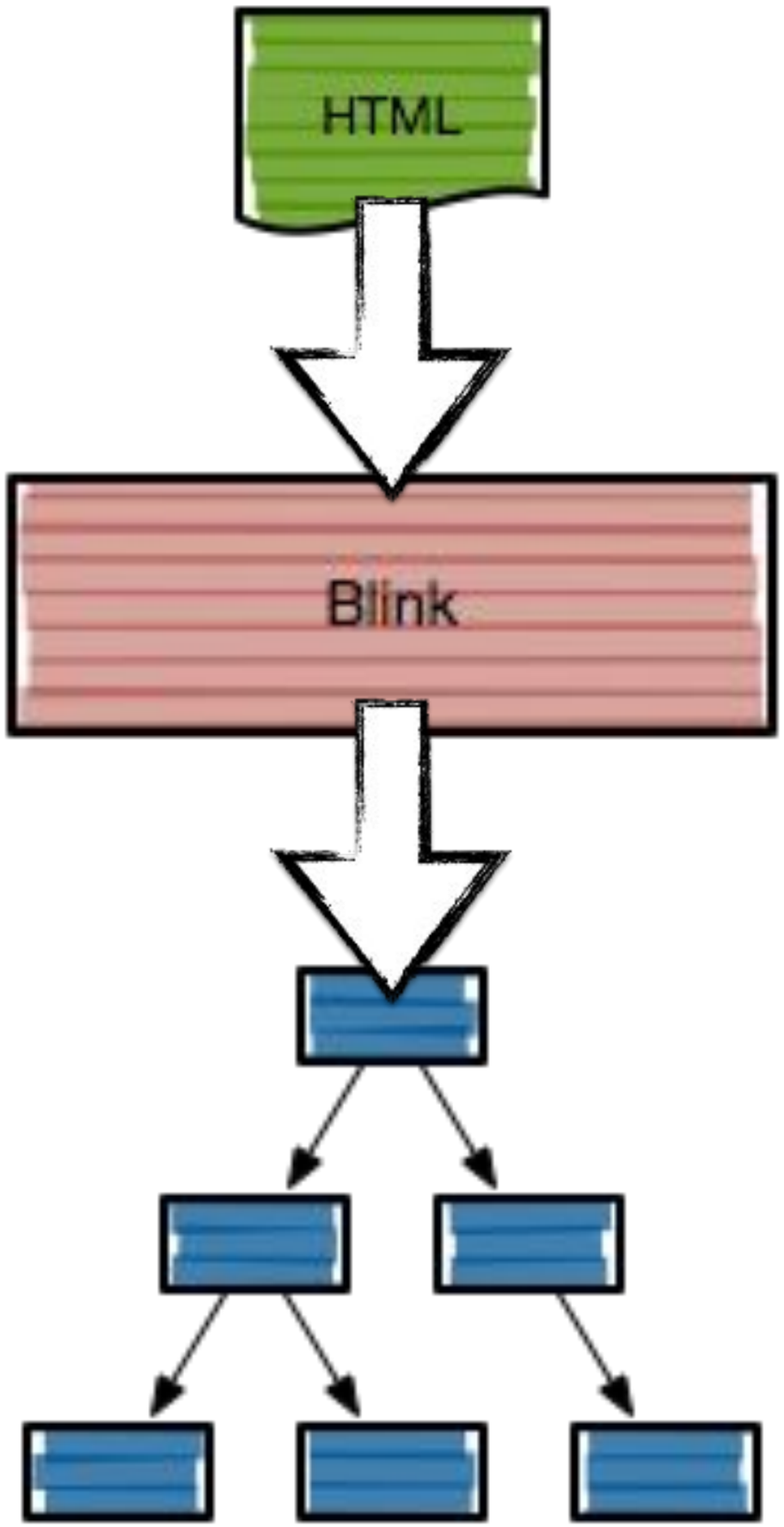
Excision

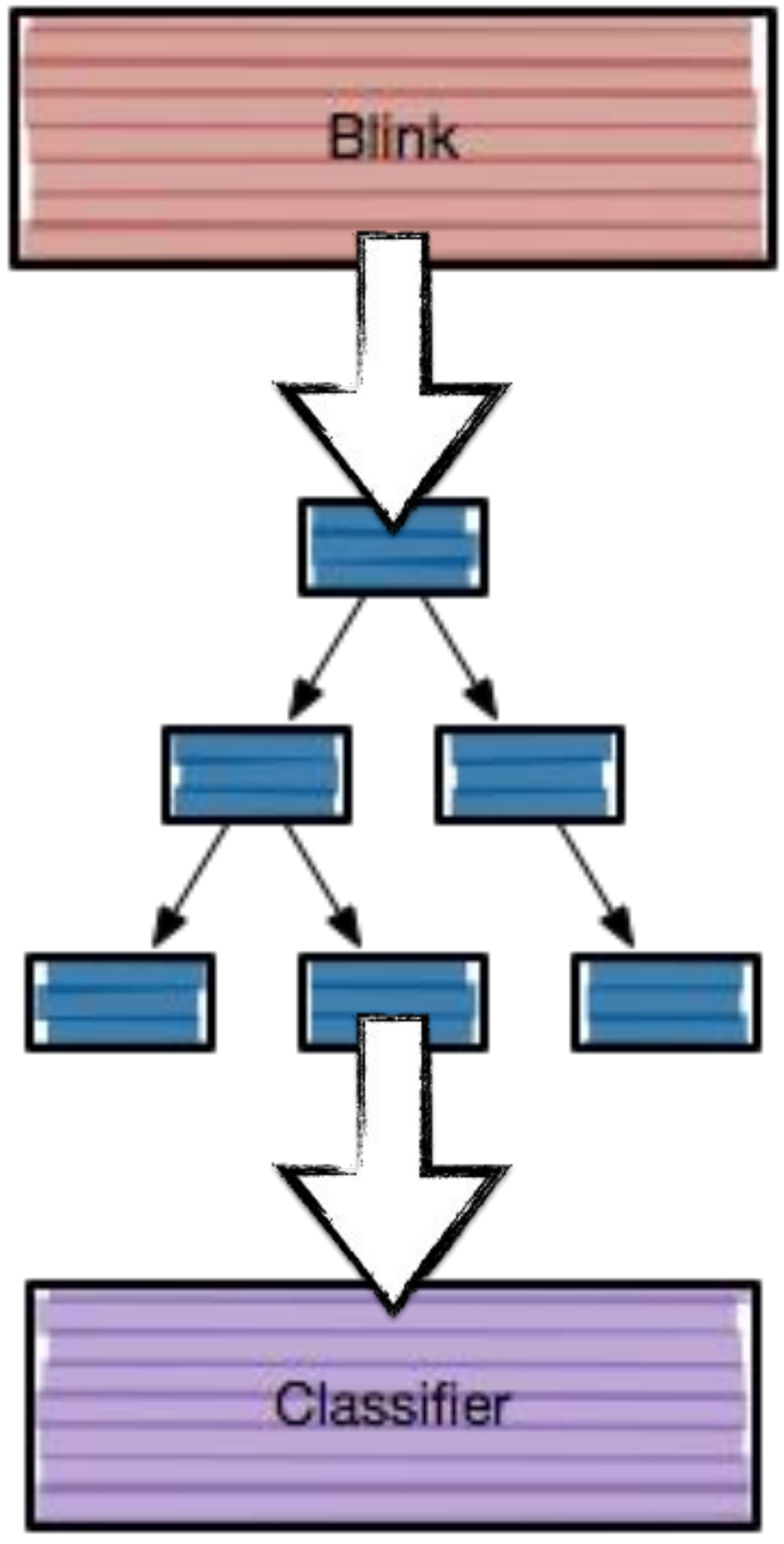
Goal: Detect malicious content before it can attack the browser

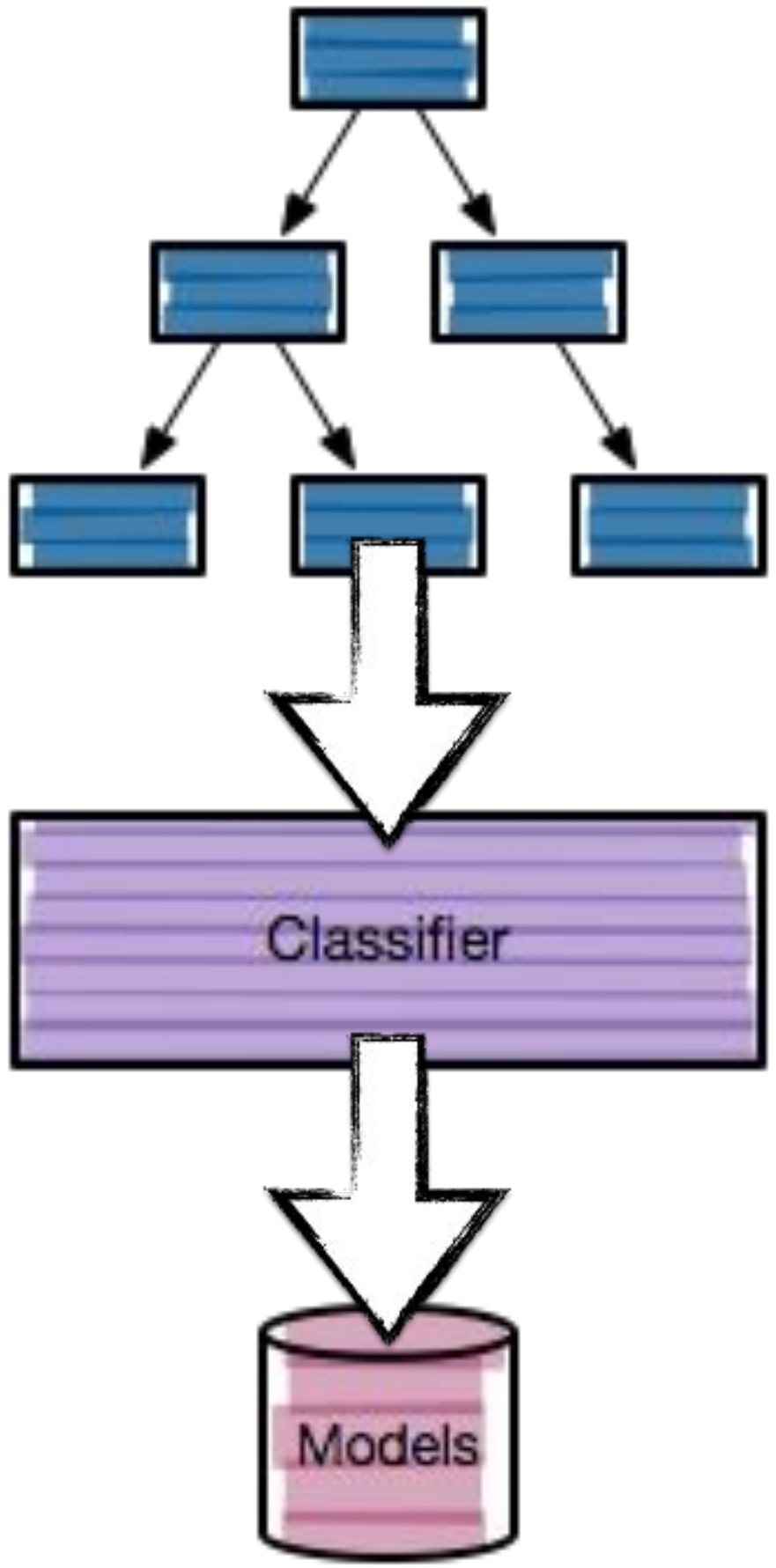
- Builds an abstraction of content inclusion relationships as pages are loaded (*inclusion trees*)
- Pre-learned models classify *inclusion sequences*
- Suspicious sequences blocked (modulo CSP)

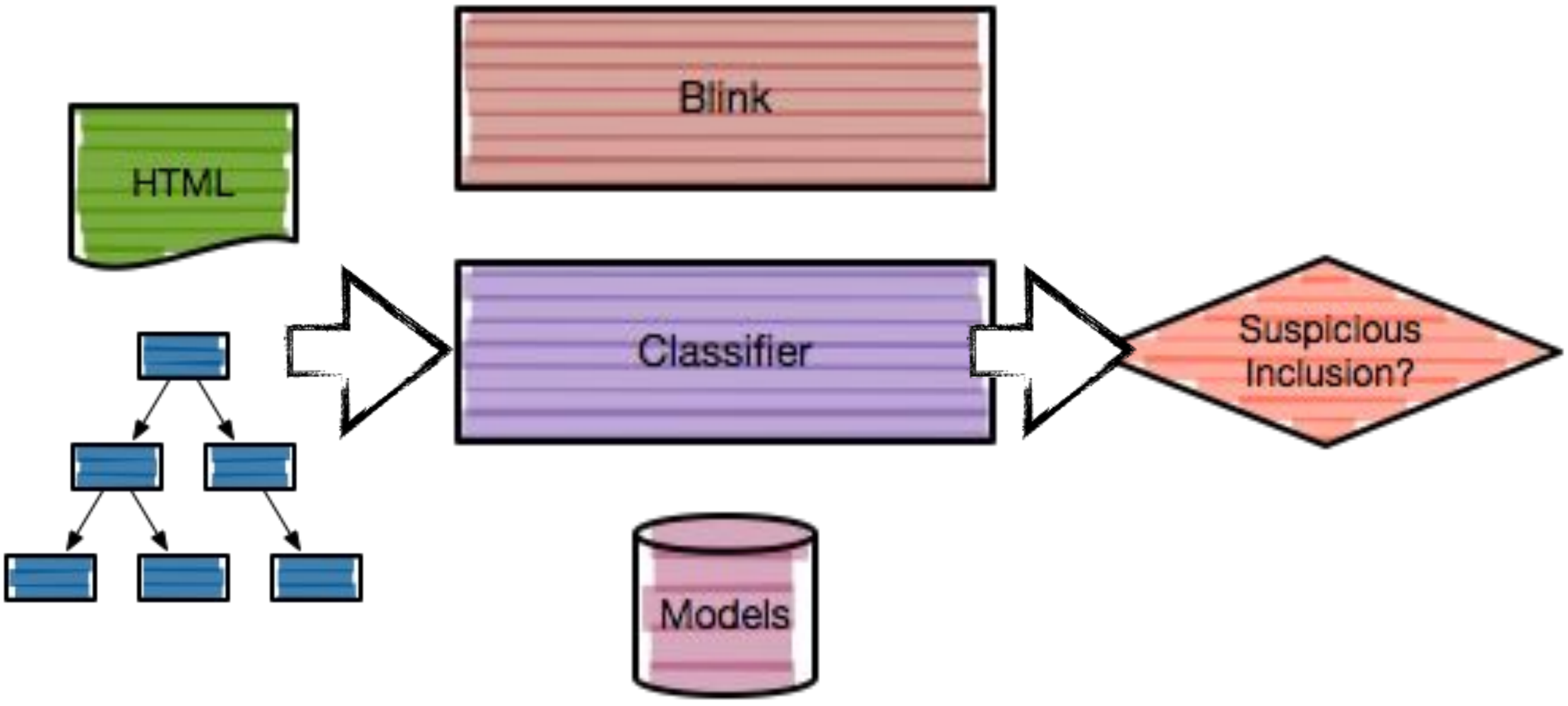


$l \in \{\text{benign}, \text{malicious}\}$


HTML
Blink


Blink
Classifier


Classifier
Models


Blink
HTML
Classifier
Suspicious
Inclusion?
Models

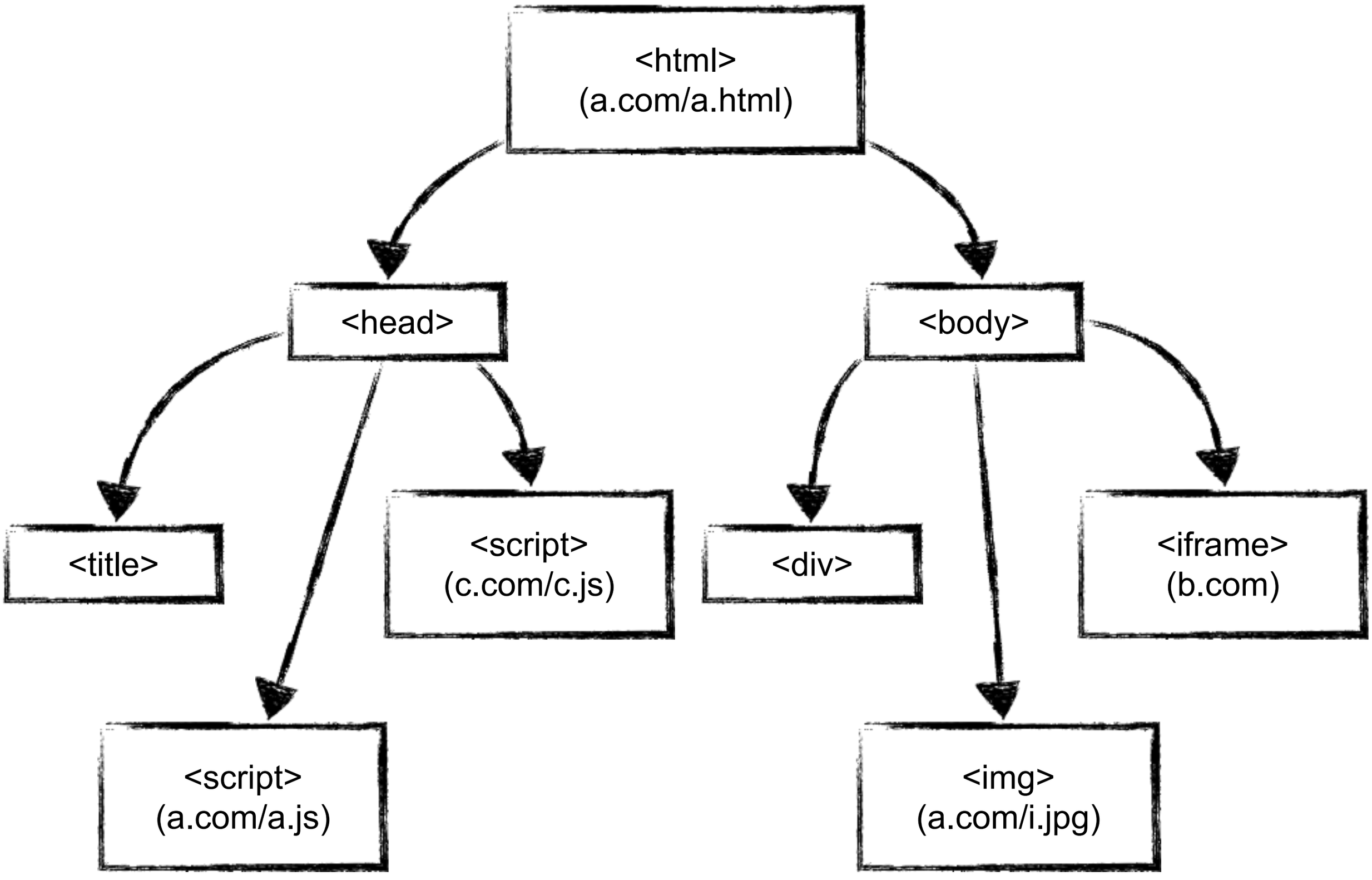
Inclusion Trees

An *inclusion tree* records provenance relationships between remotely-loaded resources in a page

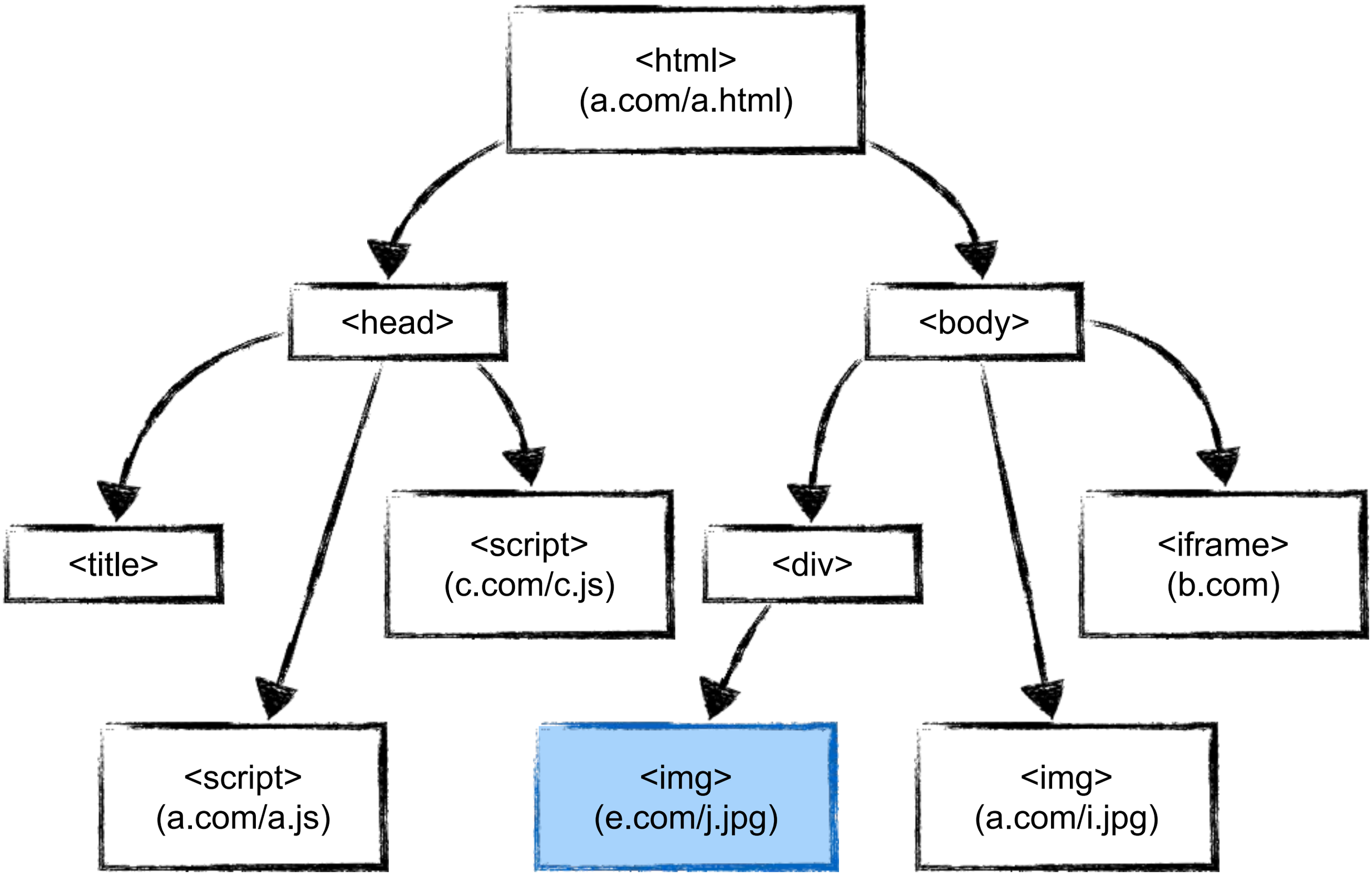
- “What origin loaded this content?”
- Distinct from DOM representation
  - “Where” vs. “who”

```
<!-- http://a.com/a.html -->

<html>
    <head>
        <title>...</title>
        <script src="http://a.com/a.js"></script>
        <script src="http://c.com/c.js"></script>
    </head>
    <body>
        <div id="status"></div>
        <img src="/i.jpg">
        <iframe src="http://b.com/">
    </body>
</html>
```

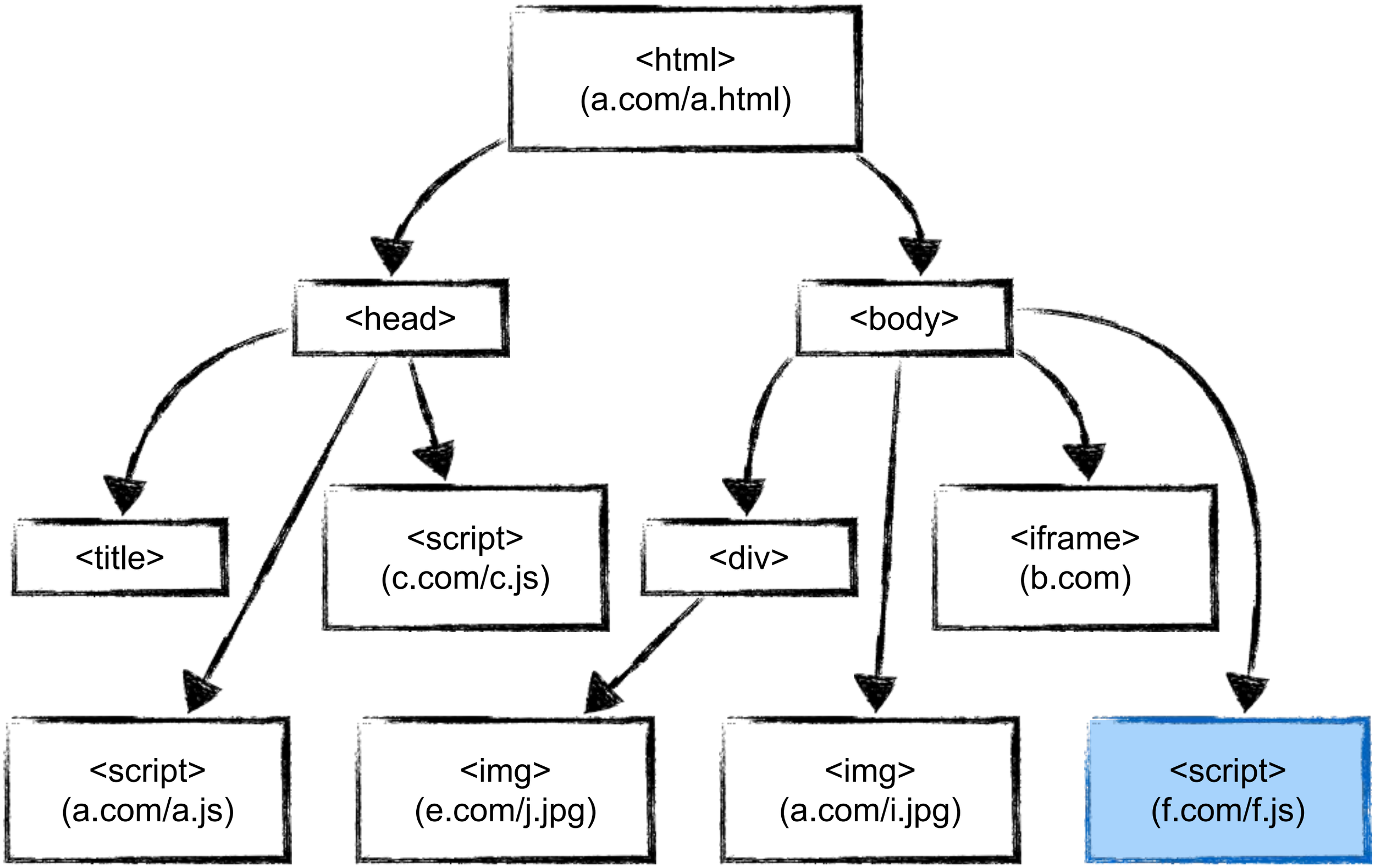
```
<!-- http://a.com/a.html -->

<html>
  <head>
    <title>…</title>
    <script src="http://a.com/a.js"></script>
    <script src="http://c.com/c.js"></script>
  </head>
  <body>
    <div id="status"></div>
    <img src="/i.jpg">
    <iframe src="http://b.com/">
  </body>
</html>
```


<html>
(a.com/a.html)
<head>
<body>
<title>
<script>
(a.com/a.js)
<script>
(c.com/c.js)

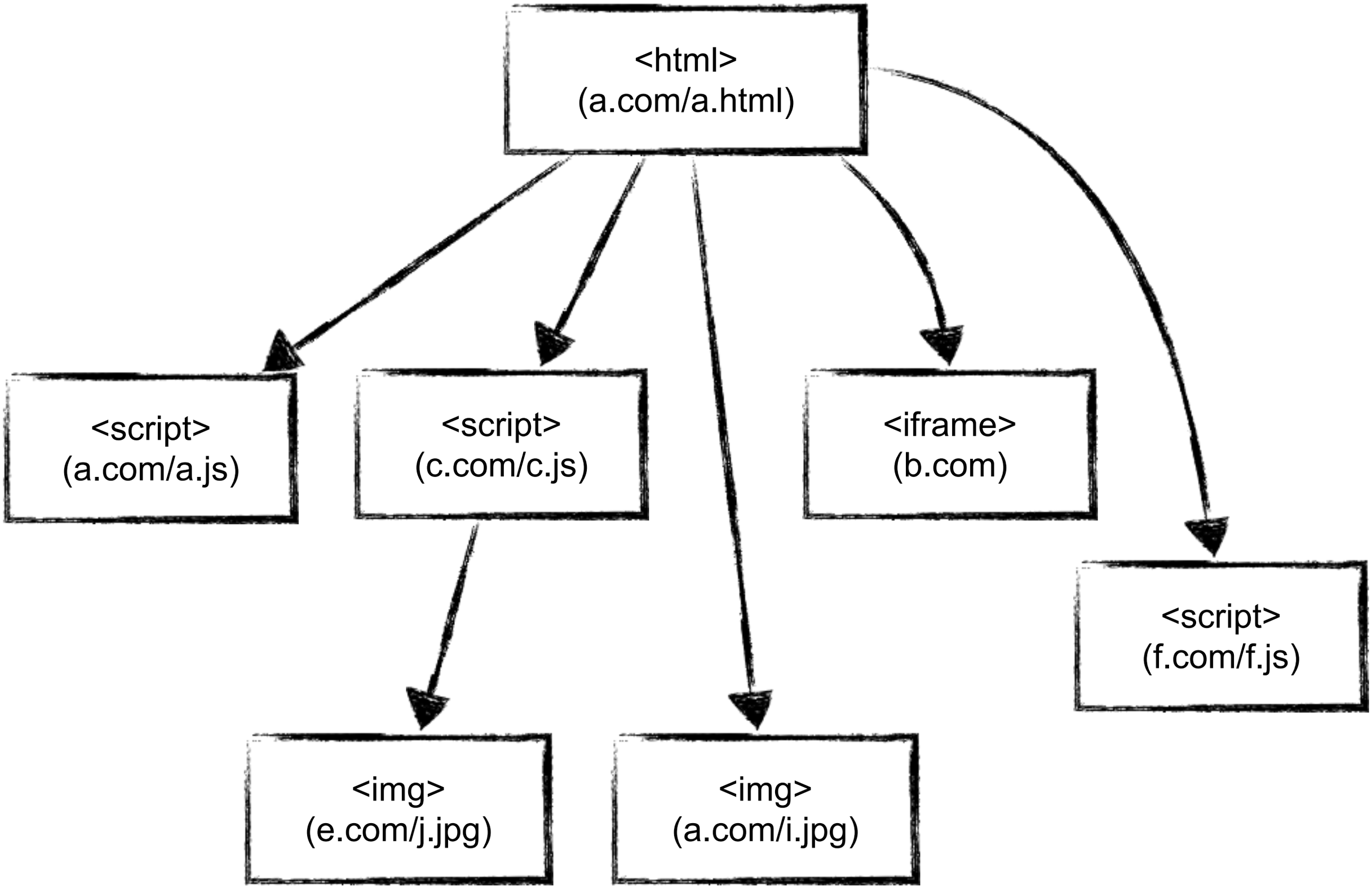
<img>
(a.com/i.jpg)
<iframe>
(b.com)

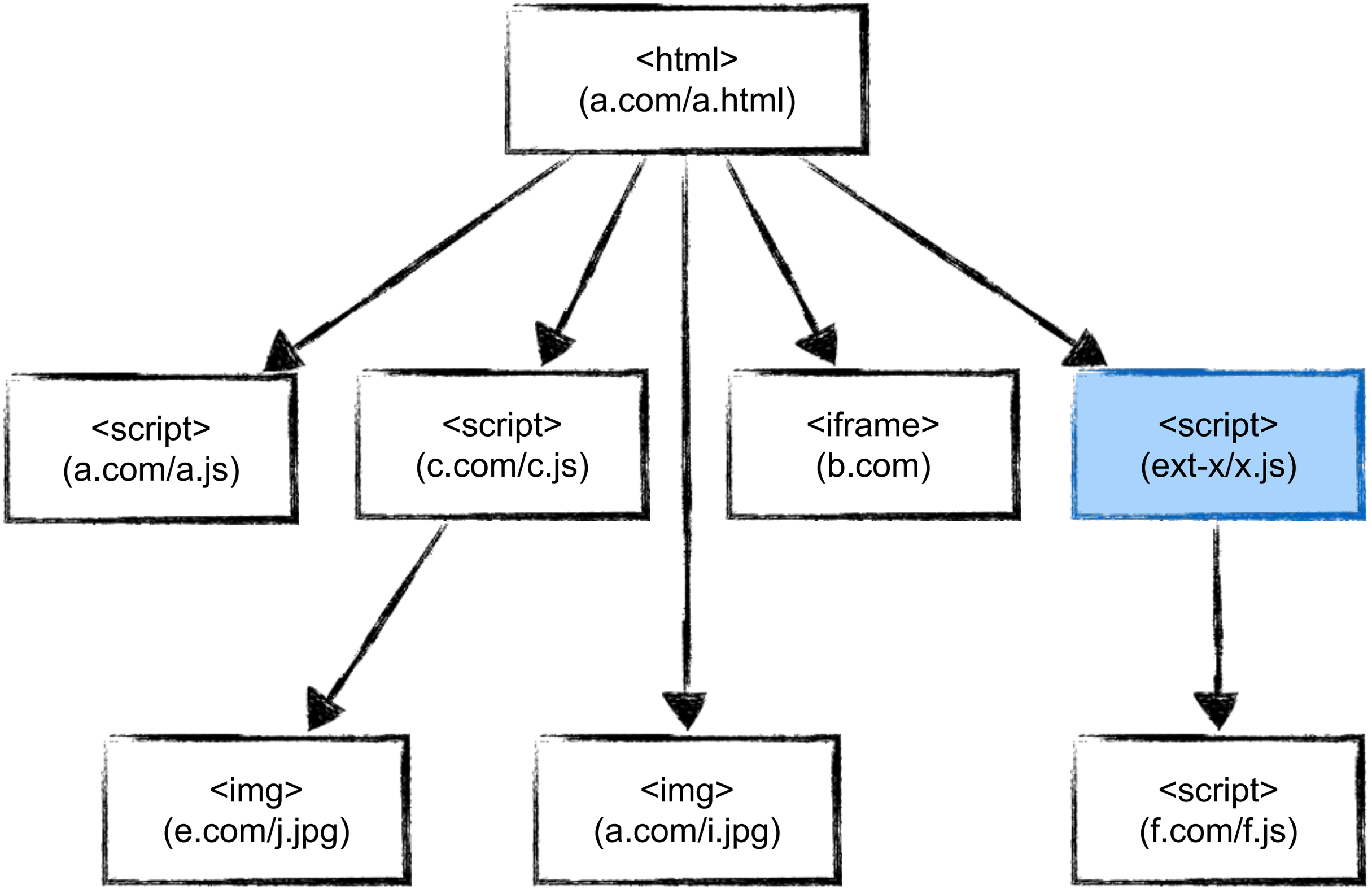

<html>
(a.com/a.html)
<head>
<body>
<title>
<script>
(a.com/a.js)
<script>
(c.com/c.js)

<img>
(e.com/j.jpg)
<iframe>
(b.com)
<img>
(a.com/i.jpg)


<html>
(a.com/a.html)
<head>
<body>
<title>
<script>
(c.com/c.js)

<iframe>
(b.com)
<script>
(a.com/a.js)
<img>
(e.com/j.jpg)
<img>
(a.com/i.jpg)
<script>
(f.com/f.js)


<html>
(a.com/a.html)
<script>
(a.com/a.js)
<script>
(c.com/c.js)
<iframe>
(b.com)
<script>
(f.com/f.js)
<img>
(e.com/j.jpg)
<img>
(a.com/i.jpg)


<html>
(a.com/a.html)
<script>
(a.com/a.js)
<script>
(c.com/c.js)
<iframe>
(b.com)
<script>
(ext-x/x.js)
<img>
(e.com/j.jpg)
<img>
(a.com/i.jpg)
<script>
(f.com/f.js)

Inclusion Sequence Classification

Goal: Given trained models, label inclusion sequences as either benign or malicious

- Feature vectors comprised of *DNS*, *String*, and *Role*-based features
- HMMs trained from labeled data set

DNS Features

- Top-level domains
- Host types
- Domain level
- Alexa ranking

Value	Examples
none	IP addresses, extensions
gen	*.com, *.org
gen-sub	*.example.com
cc	*.us, *.de
cc-sub	*.co.uk, *.com.cn

DNS Features

- Top-level domains
- Host types
- Domain level
- Alexa ranking

Value	Examples
{got,lost}-tld	ext → *.de, *.us → addr
gen-to-cc	*.org → *.de
cc-to-gen	*.uk → *.com
same-gen	*.com → *.com
diff-gen	*.com → *.org

DNS Features

- Top-level domains
- Host types
- Domain level
- Alexa ranking

Value	Examples
ipv4-public	8.8.8.8
ipv4-private	10.0.0.1
dns-sld	google.com
dns-sld-sub	a.example.com
dns-non-sld	b.dyndns.org

DNS Features

- Top-level domains
- Host types
- Domain level
- Alexa ranking

Value	Examples
same-site	a.x.com → b.x.com
same-sld	1.dyndns.org → 2.dyndns.org
same-org	example.com → example.de
same-eff-tld	a.co.uk → b.co.uk
diff	x.com → y.com

String Features

- Non-alphabetic characters
- Unique characters
- Character frequency
- Length
- Entropy

Role Features

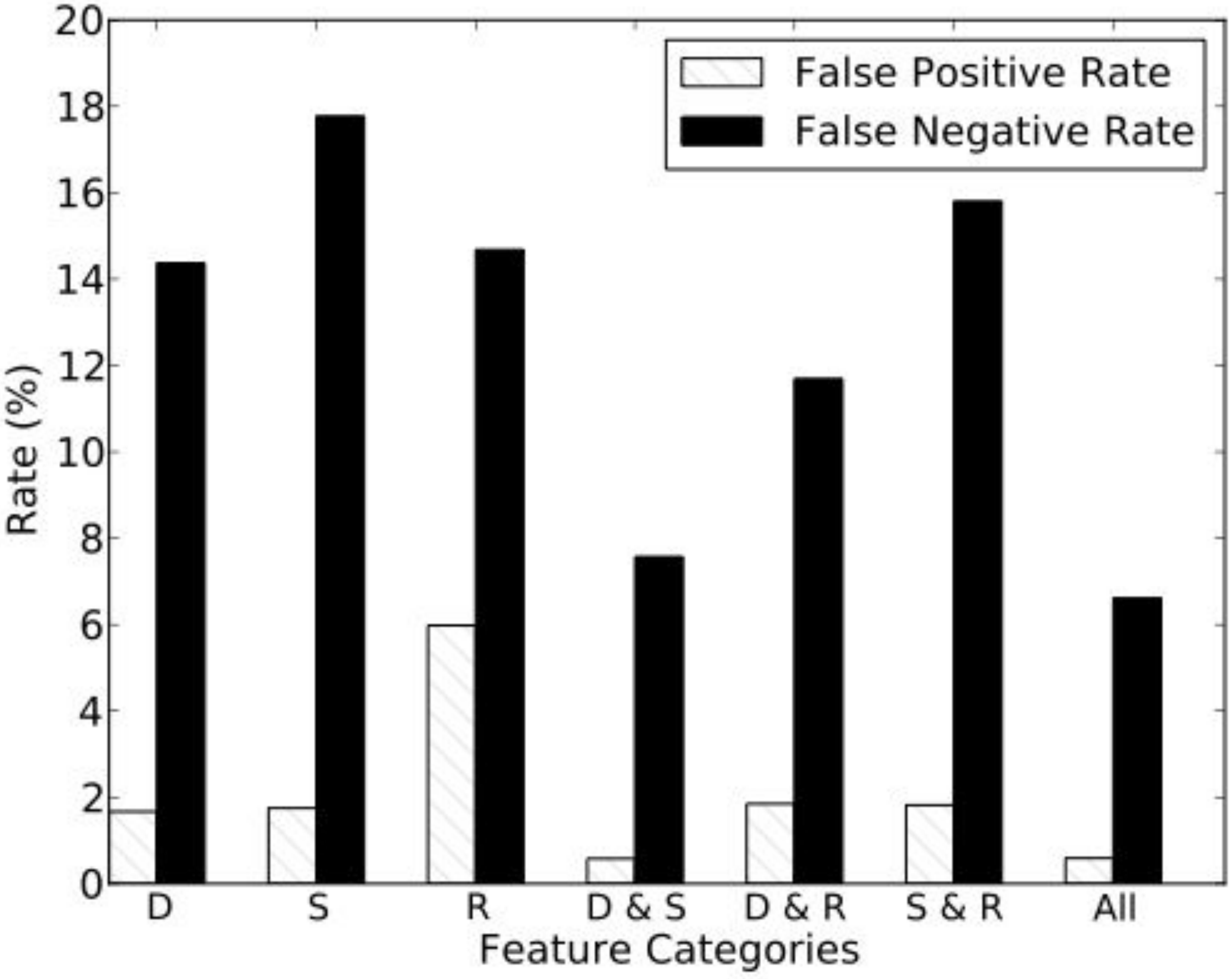
- Three binary features
  - Ad network
  - Content delivery network (CDN)
  - URL shortening service
- Manually compiled list

Evaluation

1. Are inclusion sequences useful for detecting malicious content?
2. How does this method compare with traditional blacklists?
3. What are the method's performance and usability characteristics?

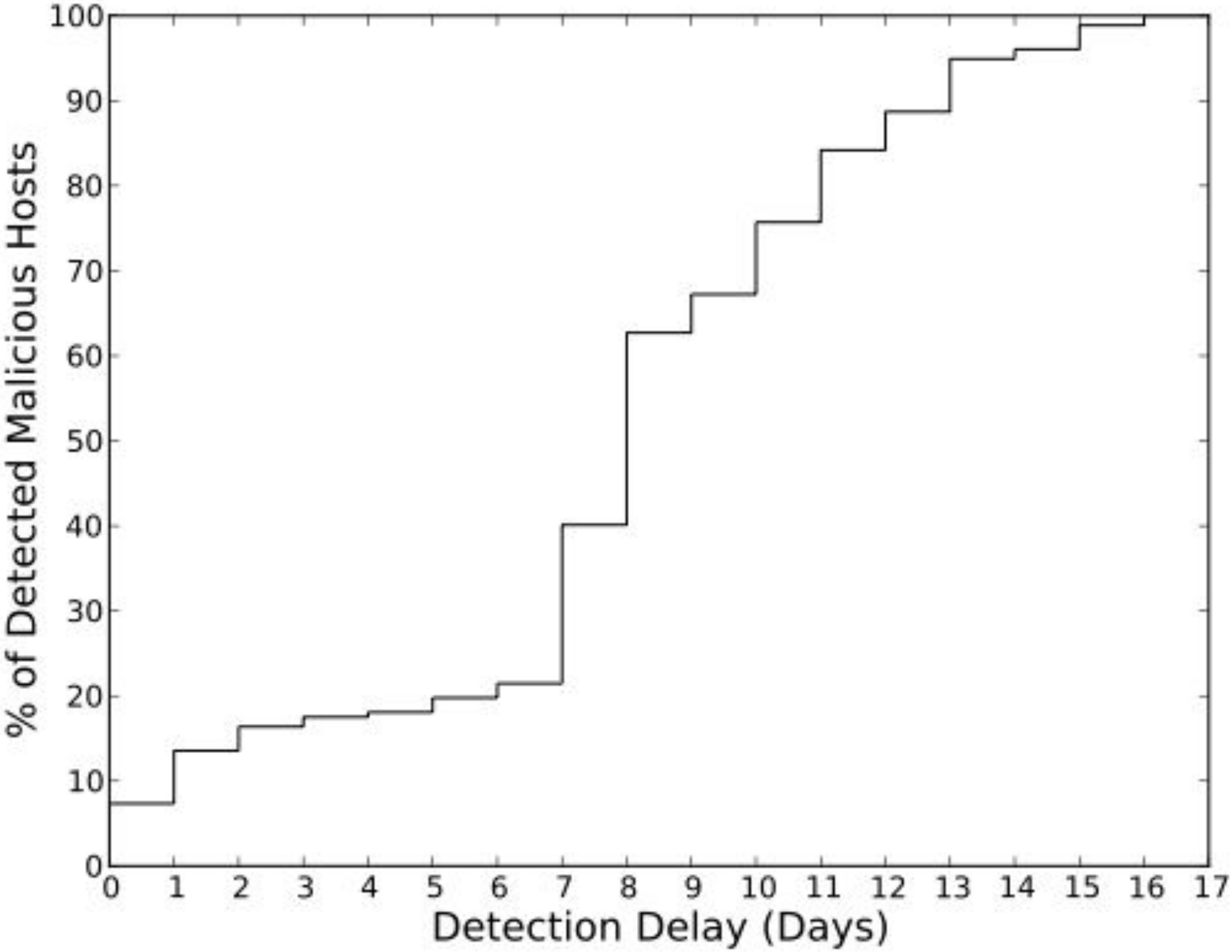
Evaluation

- Data set
  - Repeatedly crawled Alexa Top 200K from June 2014 to May 2015
    - Crawled 20 popular shopping sites in presence of 292 ad-injecting extensions
    - Anti-cloaking, anti-fingerprinting countermeasures
- Trained classifiers using VT as ground truth


False Positive Rate
False Negative Rate
Rate (%)
0
2
4
6
8
10
12
14
16
18
20
D
S
R
D & S
D & R
S & R
All
Feature Categories

Early Detection

- Compared detection results on new data from June 1 to July 14, 2015
- Detected 177 new malicious hosts later reported in VT
- Also found 89 suspicious hosts that were likely dedicated redirectors


% of Detected Malicious Hosts
0
10
20
30
40
50
60
70
80
90
100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Detection Delay (Days)

Performance and Usability

- 10 students that self-reported as expert Internet users
- Each participant explored 50 random websites from Alexa Top 500
  - 31 malicious inclusions
  - 83 errors (mostly high latency resource loads)
- Average 12.2% page latency overhead

Conclusion

- Excision identifies malicious resource inclusion sequences and allows for preemptive blocking
- Prototype implementation successfully detected malicious hosts before appearing in blacklists
- Moderate performance overhead
- Inclusion trees are a generally useful abstraction