Web Privacy Beyond Extensions: New Browsers Are Pursuing Deep Privacy Protections

Peter Snyder <pes@brave.com>
Privacy Researcher at Brave Software

In a slide…

- Web privacy is a mess.
- Privacy activists and researchers are limited by the complexity of modern browsers.
- New browser vendors are eager to work with activists to deploy their work.

Outline

1. **Background**
   Extension focus in practical privacy tools

2. **Present**
   Privacy improvements require deep browser modifications

3. **Next Steps**
   Call to action, how to keep improving

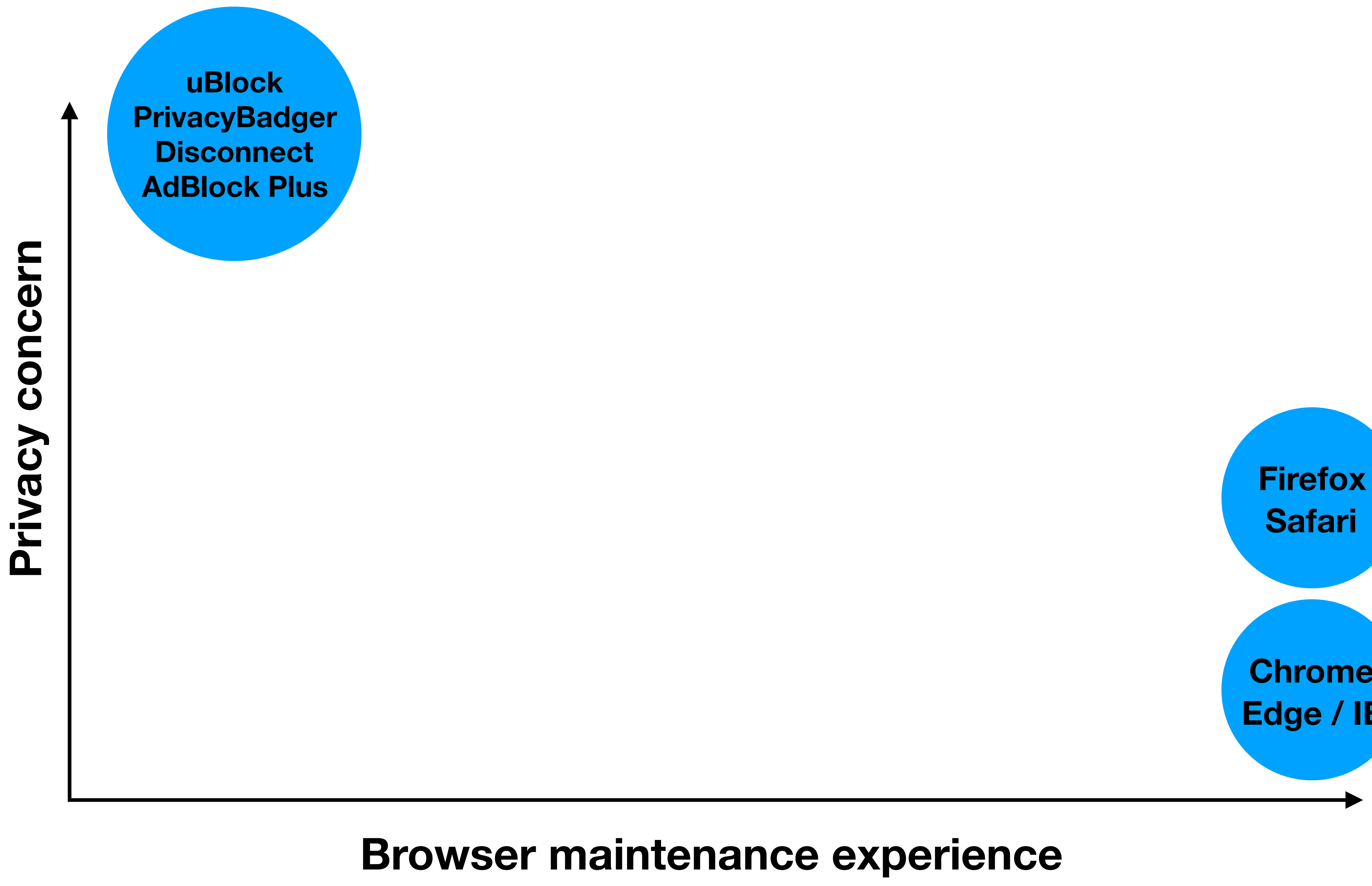
Outline

1. <u>**Background**</u>
   Extension focus in practical privacy tools

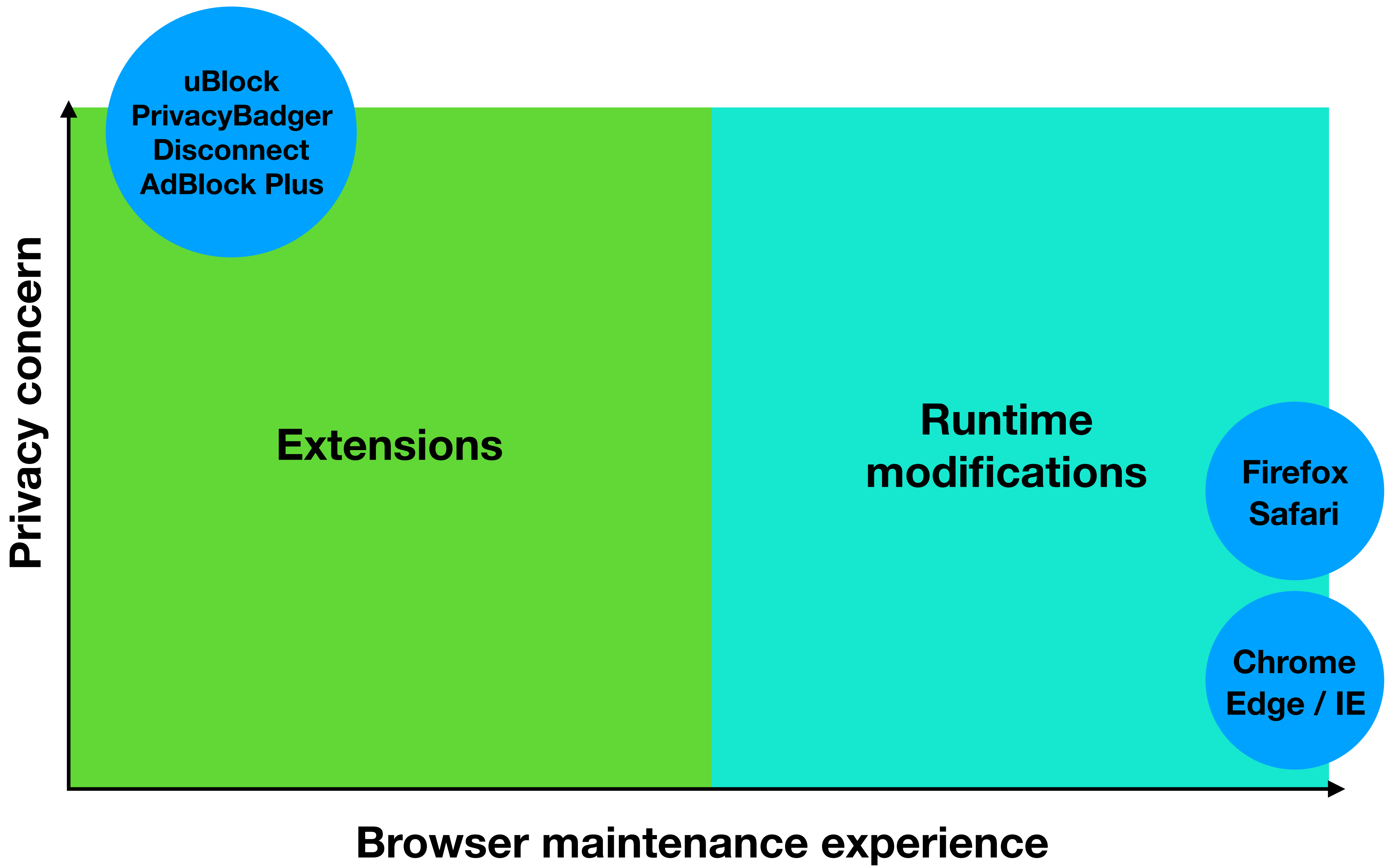
2. **Present**
   Privacy improvements require deep browser modifications

3. **Next Steps**
   Call to action, how to keep improving

Browsers are Complicated



Extensions as a Compromise



Privacy and Browser Extensions 👍

- **Successes!**
  uBlock Origin, HTTPS Everywhere, Ghostery, Disconnect, Privacy Badger, EasyList / EasyPrivacy, etc…

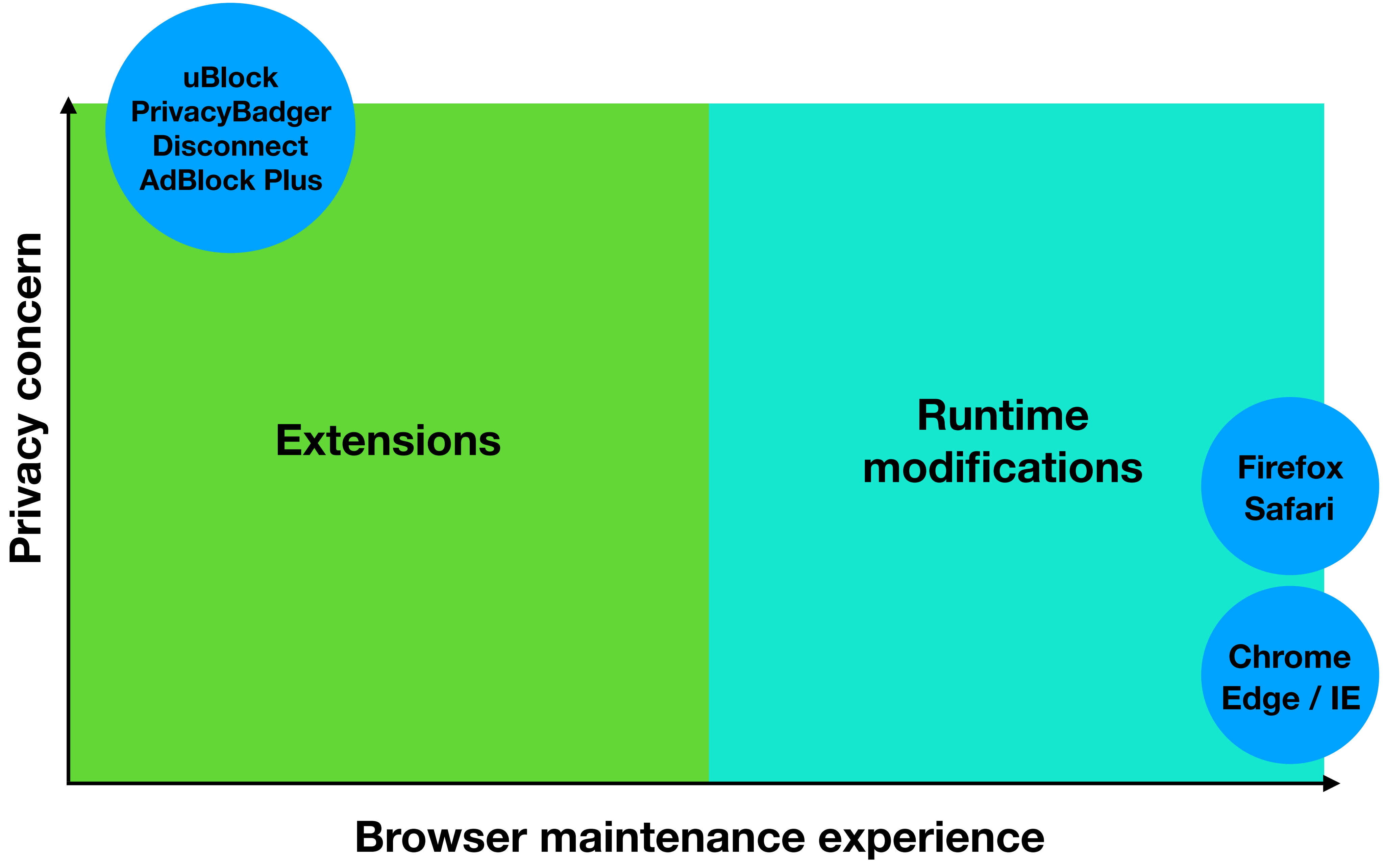
- **Appealing**
  Easy(er) to build, easy to share

- **Popular**
  Hundreds of thousands of extensions, Millions of users

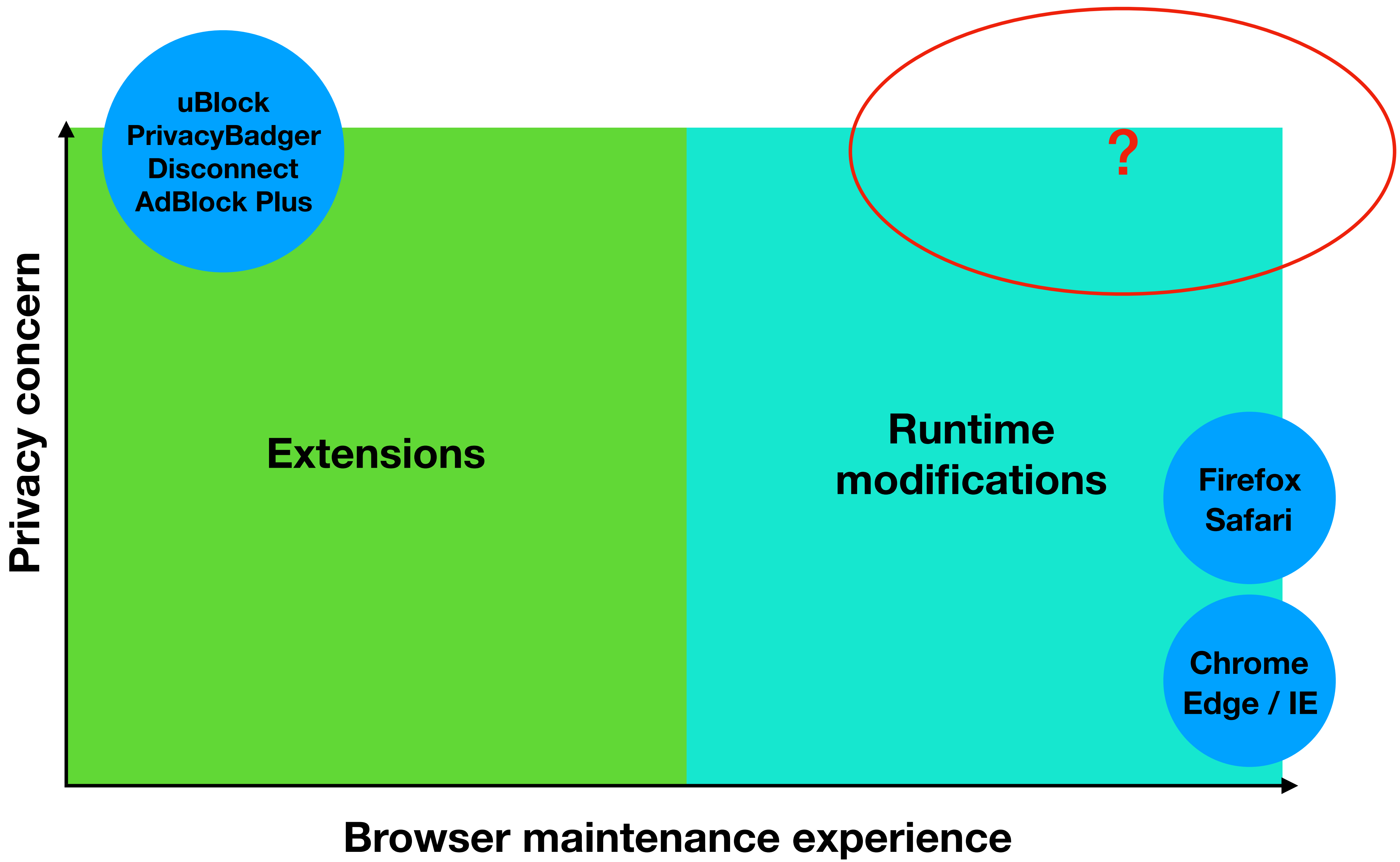
Browser Extension Limitations 👎

- **Limited Capabilities**
  Networking, request modification, rendering, layout, image processing, JS engine, etc…
- **Security and Privacy**
  Possibly giving capabilities to malicious parties
- **Performance**
  Limited to JS, secondary access

Extensions vs Runtime



Under Explored Space



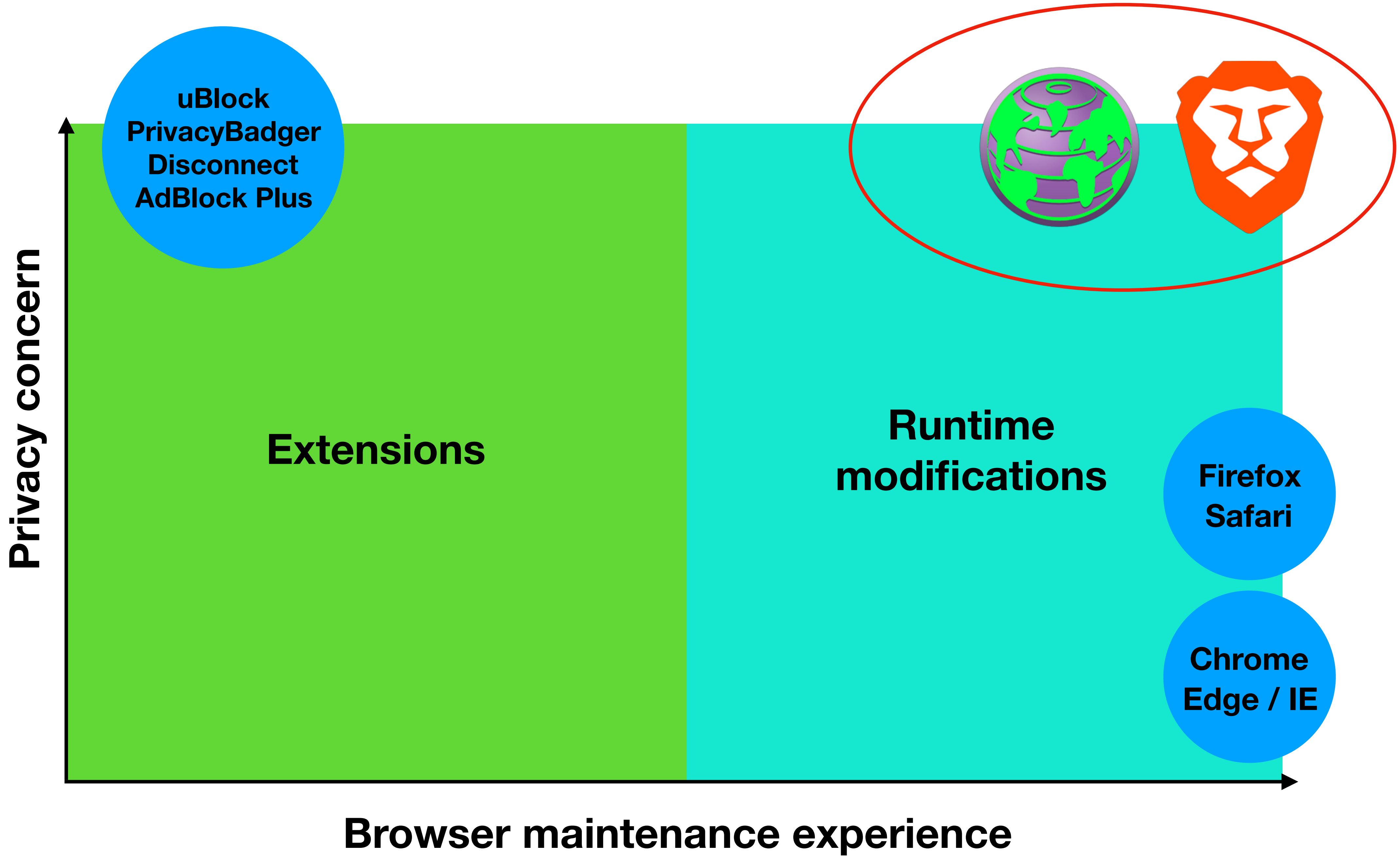
Outline

1. **Background**
   Extension focus in practical privacy tools

2. **<u>Present</u>**
   Privacy improvements require deep browser modifications

3. **Next Steps**
   Call to action, how to keep improving

Under Explored Space



Runtime Privacy Improvements

- **AdGraph**
  Client-side, ML, graph-based tracking detection
- **SpeedReader**
  Privacy enhancing content extraction
  (i.e. “aggressive reader mode”)

Runtime Privacy Improvements

- **AdGraph**
  Client-side, ML, graph-based tracking detection

- **SpeedReader**
  Privacy enhancing content extraction
  (i.e. “aggressive reader mode”)

AdGraph: A Machine Learning Approach to Automatic and Effective Adblocking

Umar Iqbal
The University of Iowa

Zubair Shafiq
The University of Iowa

Peter Snyder
Brave Software

Shitong Zhu
University of California, Riverside

Zhiyun Qian
University of California, Riverside

Benjamin Livshits
Brave Software
Imperial College London

ABSTRACT

Filter lists are widely deployed by adblockers to block ads and other forms of undesirable content in web browsers. However, these filter lists are manually curated based on informal crowdsourced feedback, which brings with it a significant number of maintenance challenges. To address these challenges, we propose a machine learning approach for automatic and effective adblocking called AdGraph. Our approach relies on information obtained from multiple layers of the web stack (HTML, HTTP, and JavaScript) to train a machine learning classifier to block ads and trackers. Our evaluation on Alexa top-10K websites shows that AdGraph automatically and effectively blocks ads and trackers with 97.7% accuracy. Our manual analysis shows that AdGraph has better recall than filter lists, it blocks 16% more ads and trackers with 65% accuracy. We also show that AdGraph is fairly robust against adversarial obfuscation by publishers and advertisers that bypass filter lists.

1 INTRODUCTION

Background. Adblocking deployment has been steadily increas-

scripts [40, 49, 60], which can then be blocked by adblockers. Second, some advertisers have started to manipulate the delivery of their ads to bypass filer lists used by adblockers. For example, Facebook recently obfuscated signatures of ad elements that were used by filter lists to block ads. Adblockers, in response, quickly identified new signatures to block Facebook ads. This prompted a few back and forth actions, with Facebook changing their website to remove ad signatures, and adblockers responding with new signatures [52].

Limitations of Filter Lists. While adblockers are able to block Facebook ads (for now), Facebook's whack-a-mole strategy points to two fundamental limitations of adblockers. First, adblockers use manually curated filter lists to block ads and trackers based on informally crowdsourced feedback from the adblocking community. This manual process of filter list maintenance is inherently slow and error-prone. When new websites are created, or existing websites make changes, it takes adblocking community some time to catch up by updating the filter lists [1]. This is similar to other areas of system security, such as updating anti-virus signatures [31, 42]. Second, rules defined in these filter lists are fairly simple HTTP and HTML signatures that are easy to defeat for financially motivated publishers and advertisers. Researchers have shown that random-

Current Tracking Blocking

- Extremely useful!
- Uses well known, targeted approaches
- Vulnerable to practical countermeasures
- We see increasing evasion
- Two typical approaches…

URL Based Blocking

- **Representative Extension**
  AdBlock Plus + EasyPrivacy

- **Approach**
  1. Identify URLs that trackers come from
  2. Build rules to instruct the browser to ignore these URLs

- **Example**
  1. Notice: https://example.org/tracking.js
  2. Block: */tracking.js

URL Based Evasions

- **Rotate Domains**
  - Domain generation algorithms (DGA)
  - Host on CDNs

- **Move to First Party**
  Sites host local copies of tracking code

- **Compose with “benign” code**
  - Concatenate into one single file
  - Magnification / packing / browserify / require.js / etc.

Behavior Based Blocking

- **Representative Extension**
  PrivacyBadger

- **Approach**
  1. Look for code that does suspicious things
  2. Block or restrict similar code

- **Example**
  1. Notice script from **tracker.com** uses Canvas and WebGL oddly
  2. Prevent all code from **tracker.com** from accessing any privacy sensitive functionality

Behavior Based Evasions

- **Rotate Domains**
  - Domain generation algorithms (DGA)
  - Host on CDNs

- **Split Suspicious Activity Across Parties**
  Avoid detection thresholds by distributing activity

- **Evade Attribution**
  - eval
  - new Function()
  - Promise.then()
  - etc…

AdGraph Alternative

- **Blocking tracking resources**
  JS, tracking pixels, iFrames…

- **Deep browser instrumentation**
  - Network: requests made during page execution
  - Layout: page structure and modifications
  - JavaScript: attribute above to responsible code

- **Block based on context**
  ML classification based on above described context

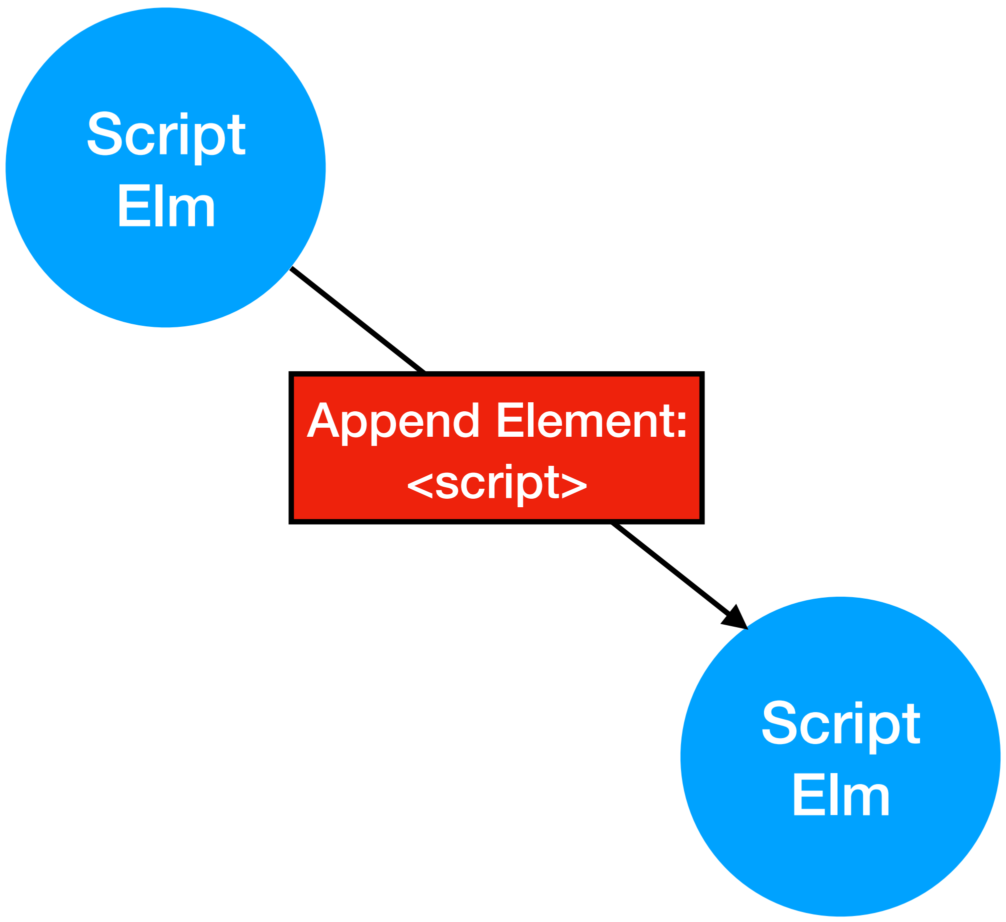
Common JS Example

1. Script element with inline code, that…
2. Appends a script element after itself, with remote script, that…
3. Reads document cookies (and other FP elements), creates an adjacent image, and then…
4. Fetches images from unknown URLs

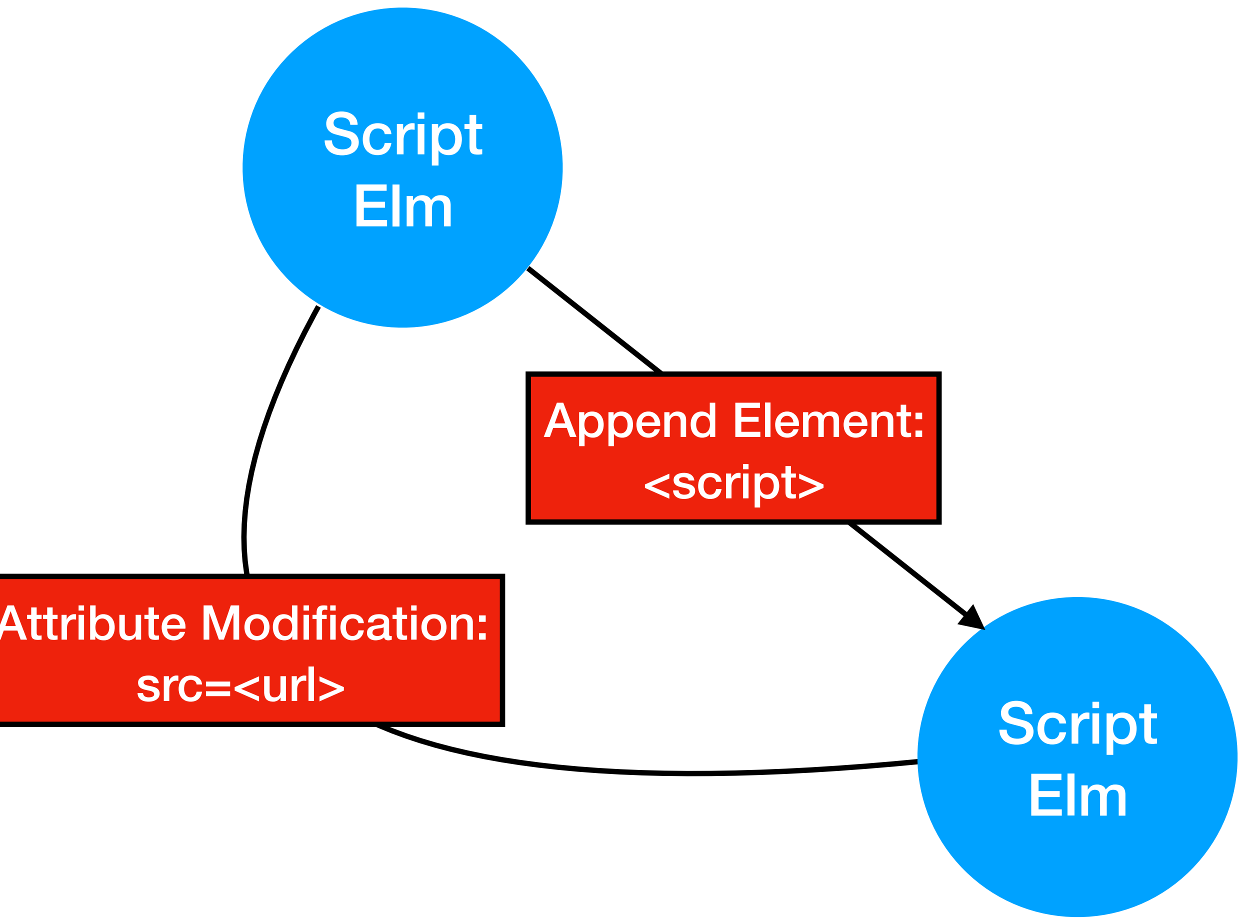
AdGraph Example

AdGraph Example

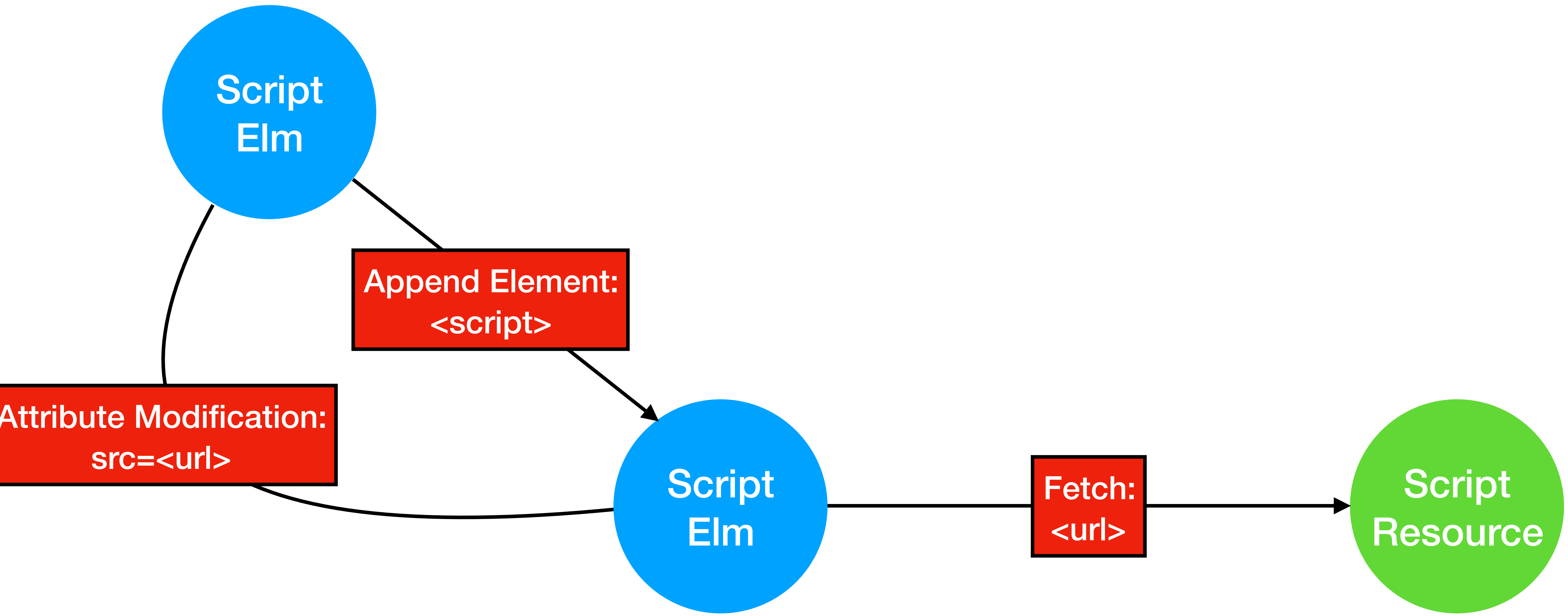
AdGraph Example



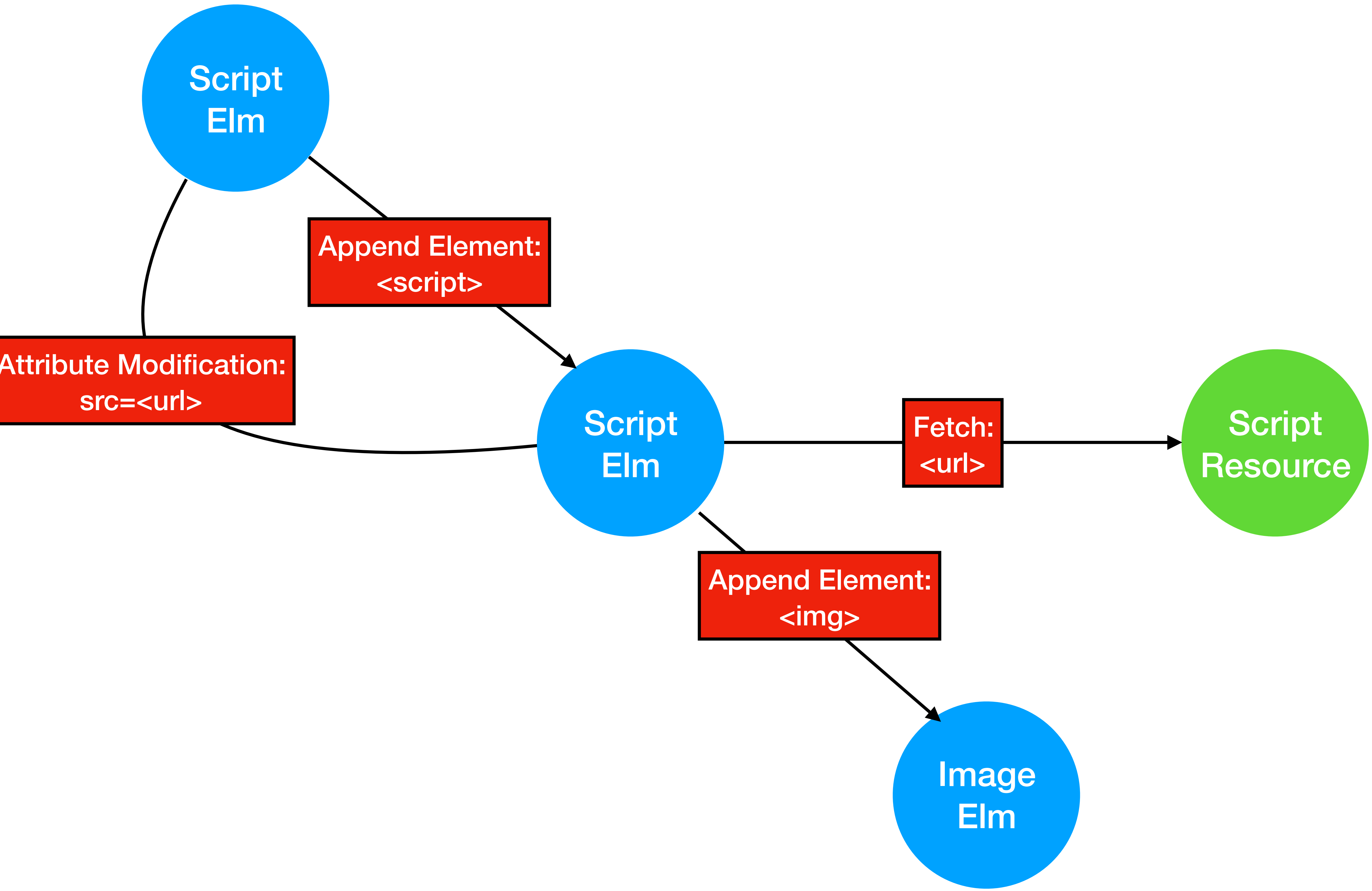
AdGraph Example



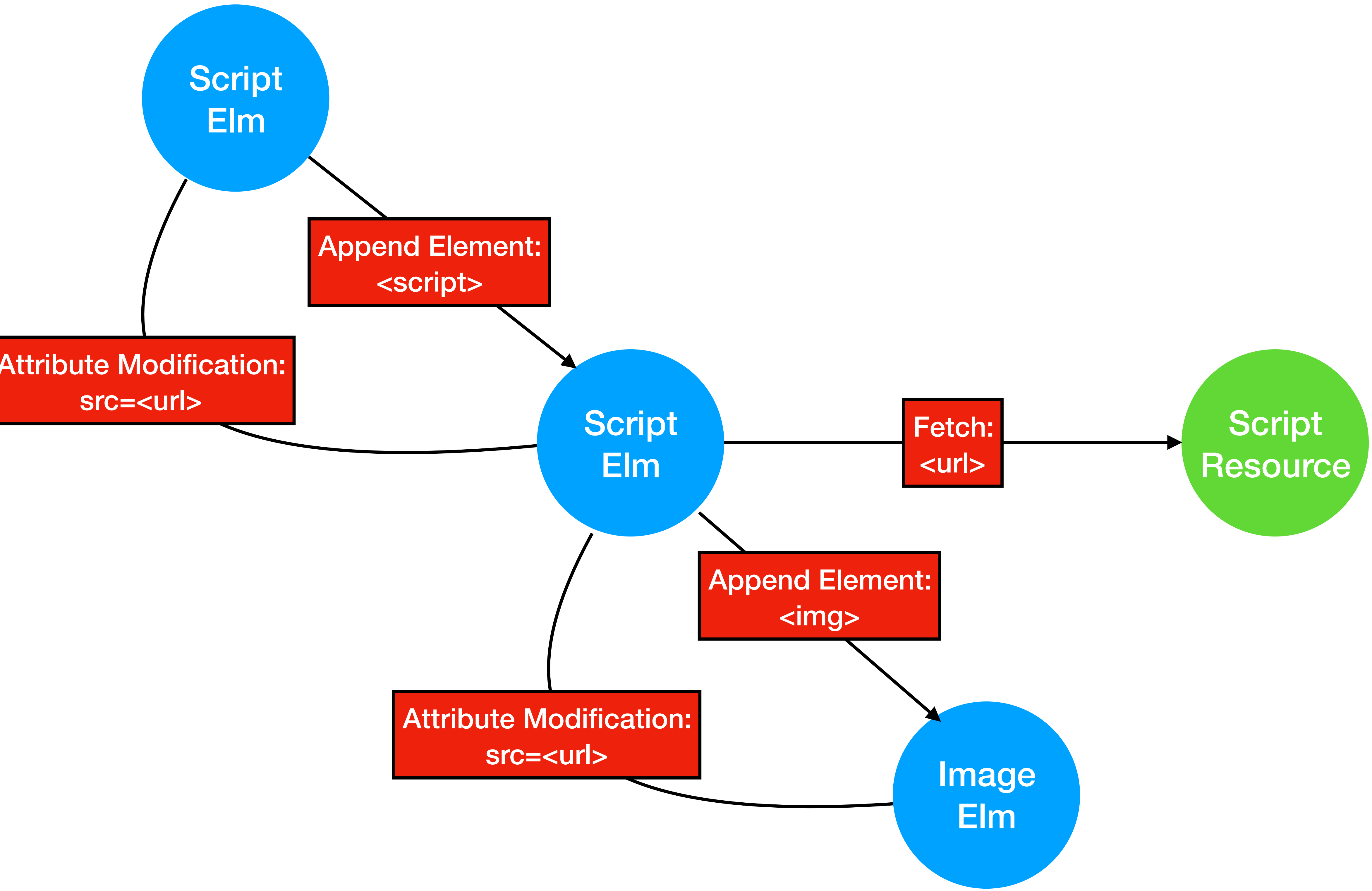
AdGraph Example



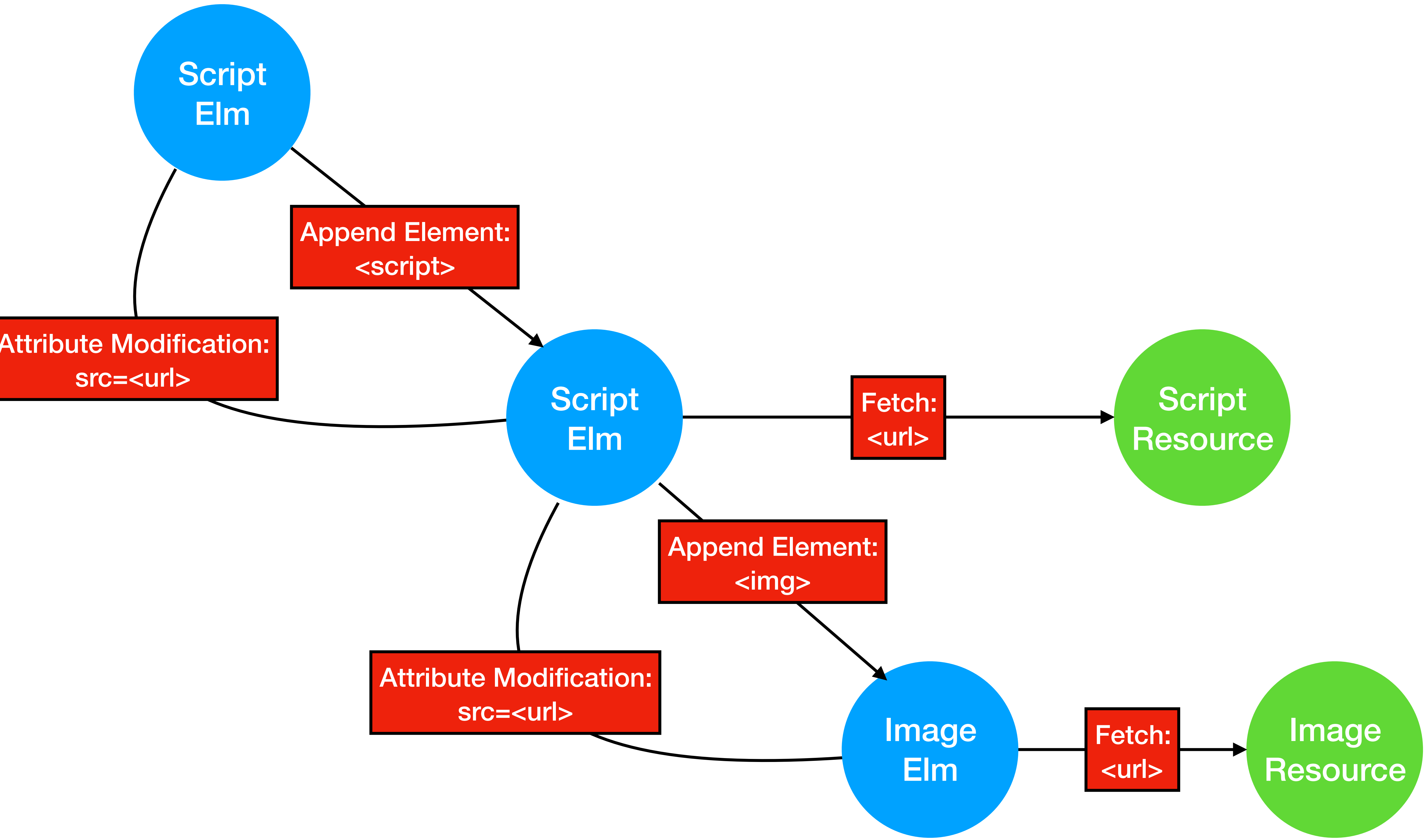
AdGraph Example



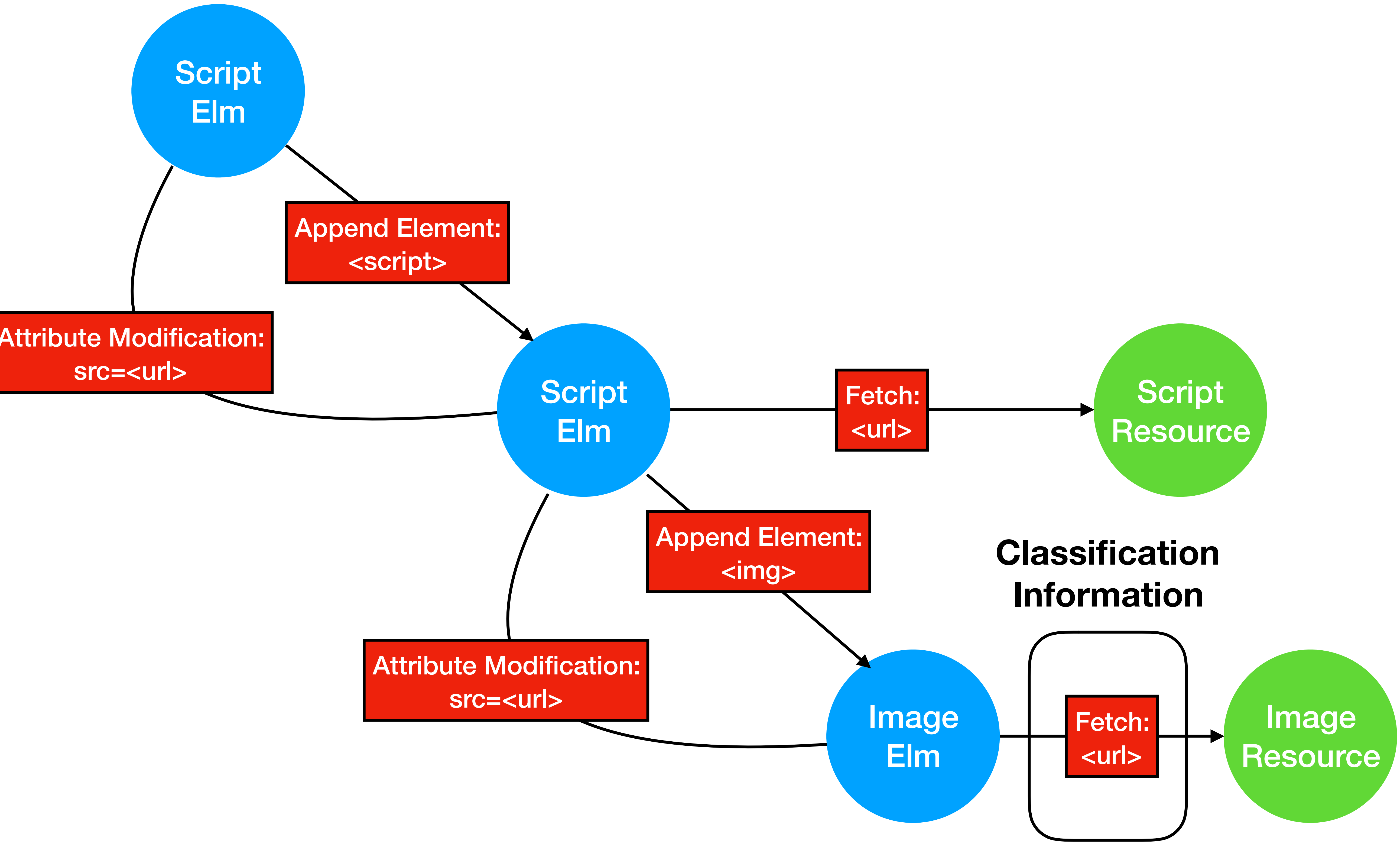
AdGraph Example



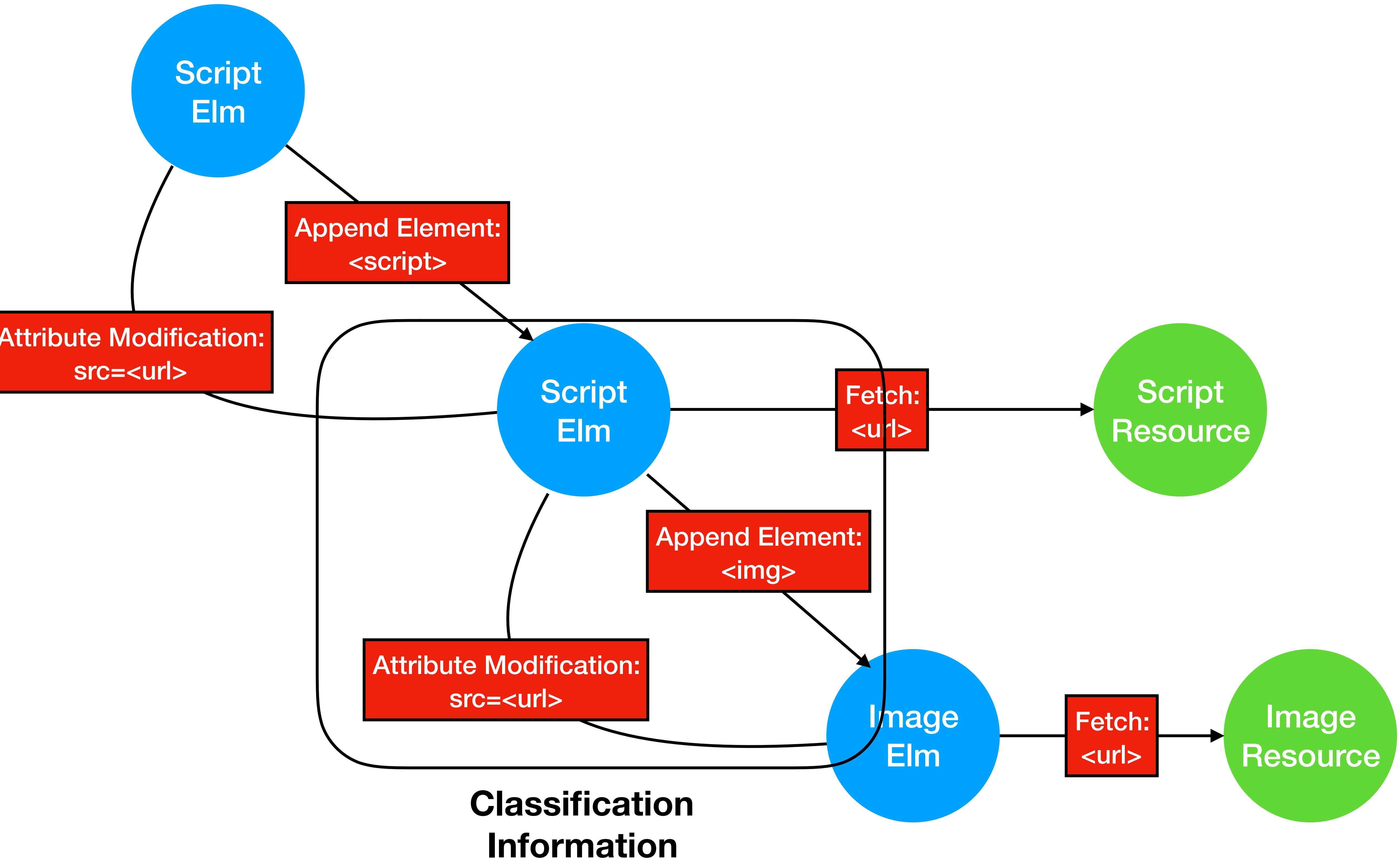
AdGraph Example



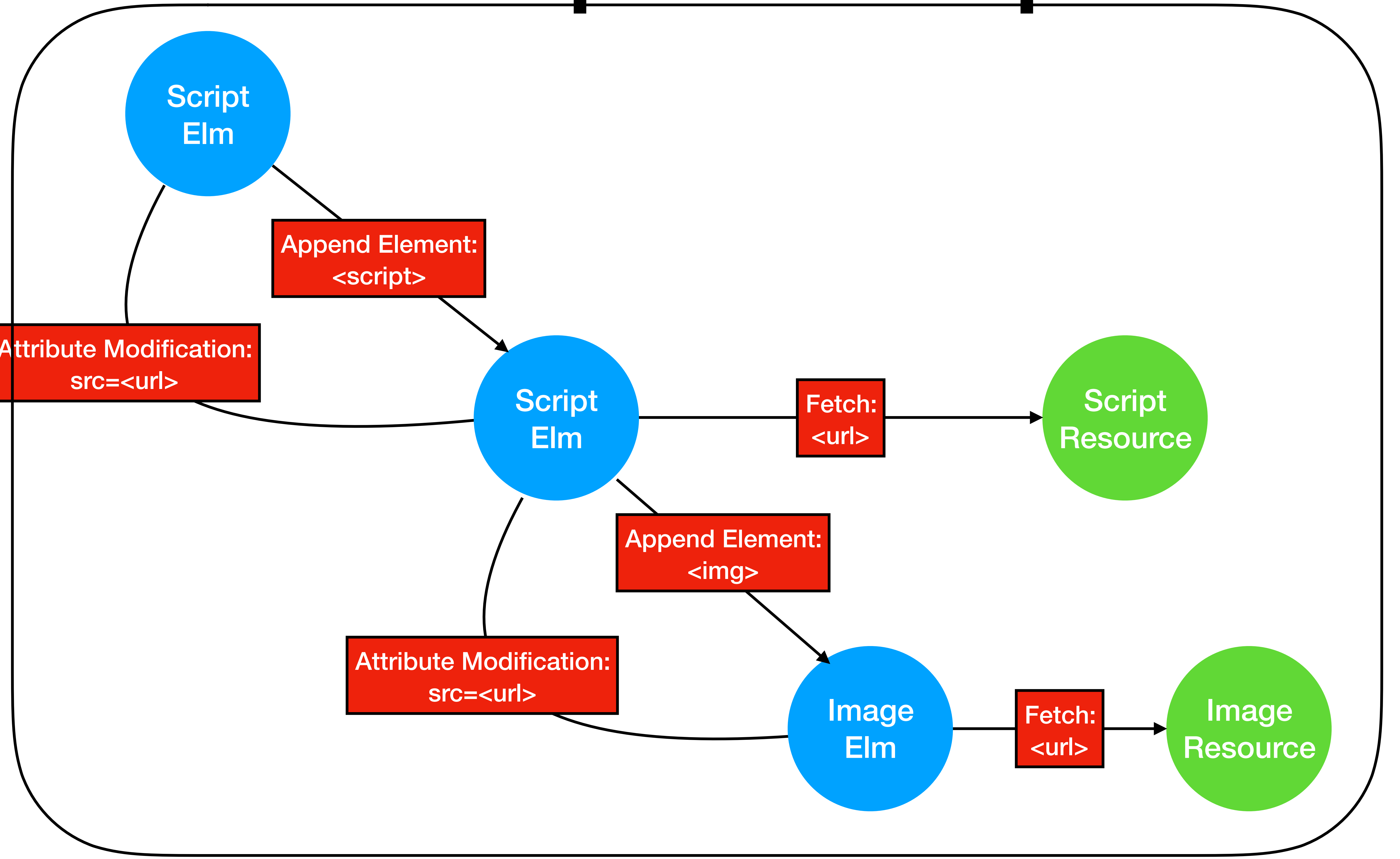
AdGraph Example



AdGraph Example



AdGraph Example



AdGraph Results

- **High accuracy**
  > 95% compared to current, human approaches

- **Strong privacy protections**
  Identifies tracking resources missed by current tools

- **High performance**
  As fast or faster than current approaches (and default Chromium!)

- **Not limited to lists**
  Can adapt as trackers adapt

Not Possible with Extensions

- **Information Breath**
  Needed information not available to browser extensions

- **Information Depth**
  JS information not available to other browsers!

- **Performance**
  Blocking ML classifier benefits from C++ implementation

Runtime Privacy Improvements

- **AdGraph**
  Client-side, ML, graph-based tracking detection

- **SpeedReader**
  Privacy enhancing content extraction
  (i.e. “aggressive reader mode”)

SpeedReader: Reader Mode Made Fast and Private

Mohammad Ghasemisharif
University of Illinois at Chicago
mghas2@uic.edu

Peter Snyder
Brave Software
pes@brave.com

Andrius Aucinas
Brave Software
aaucinas@brave.com

Benjamin Livshits
Brave Software / Imperial College London
ben@brave.com

ABSTRACT

Most popular web browsers include "reader modes" that improve the user experience by removing un-useful page elements. Reader modes reformat the page to hide elements that are not related to the page's main content. Such page elements include site navigation, advertising related videos and images, and most JavaScript. The intended end result is that users can enjoy the content they are interested in, without distraction.

In this work, we consider whether the "reader mode" can be widened to also provide performance and privacy improvements. Instead of its use as a post-render feature to clean up the clutter on a page we propose `SpeedReader` as an alternative multistep pipeline that is part of the rendering pipeline. Once the tool decides during the initial phase of a page load that a page is suitable for reader mode use, it directly applies document tree translation *before the page is rendered.*

Based on our measurements, we believe that `SpeedReader` can be continuously enabled in order to drastically improve end-user experience, especially on slower mobile connections. Combined with our approach to predicting which pages should be rendered in reader mode with 91% accuracy, it achieves drastic speedups and bandwidth reductions of up to 27× and 84× respectively on average. We further find that our novel "reader mode" approach brings with it significant privacy improvements to users. Our approach effectively removes all commonly recognized trackers, issuing 115

are large [41], they are small as a proportion of all URLs on the web. Similarly, while these lists are updated often, they are updated slowly compared to URL updates on the web.

Similarly, "reader mode" tools, provided in many popular browsers and browser extensions, are an effort to reduce the growing visual complexity of web sites. Such tools attempt to extract the subset of page content useful to users, and remove advertising, animations, boiler plate code, and other non-core content. Current "reader modes" do not provide the user with resource savings since the referenced resources have already been fetched and rendered. The growth and popularity of such tools suggest they are useful to browser users, looking to address the problem of page clutter and visual "bloat".

In this work, we propose a novel strategy called **`SpeedReader`** for dealing with resource and bloat on websites. Our technique provides a user experience similar to existing "reader mode" tools, but with network, performance, and privacy improvements that exceed existing ad and tracking blocking tools, on a significant portion of websites. Significantly, `SpeedReader` differs from existing deployed reader mode tools by operating *before page rendering*, which allows it to determine which resources are needed for the page's core content before fetching.

How we achieve speedups. `SpeedReader` achieves its performance improvements through a two-step pipeline:

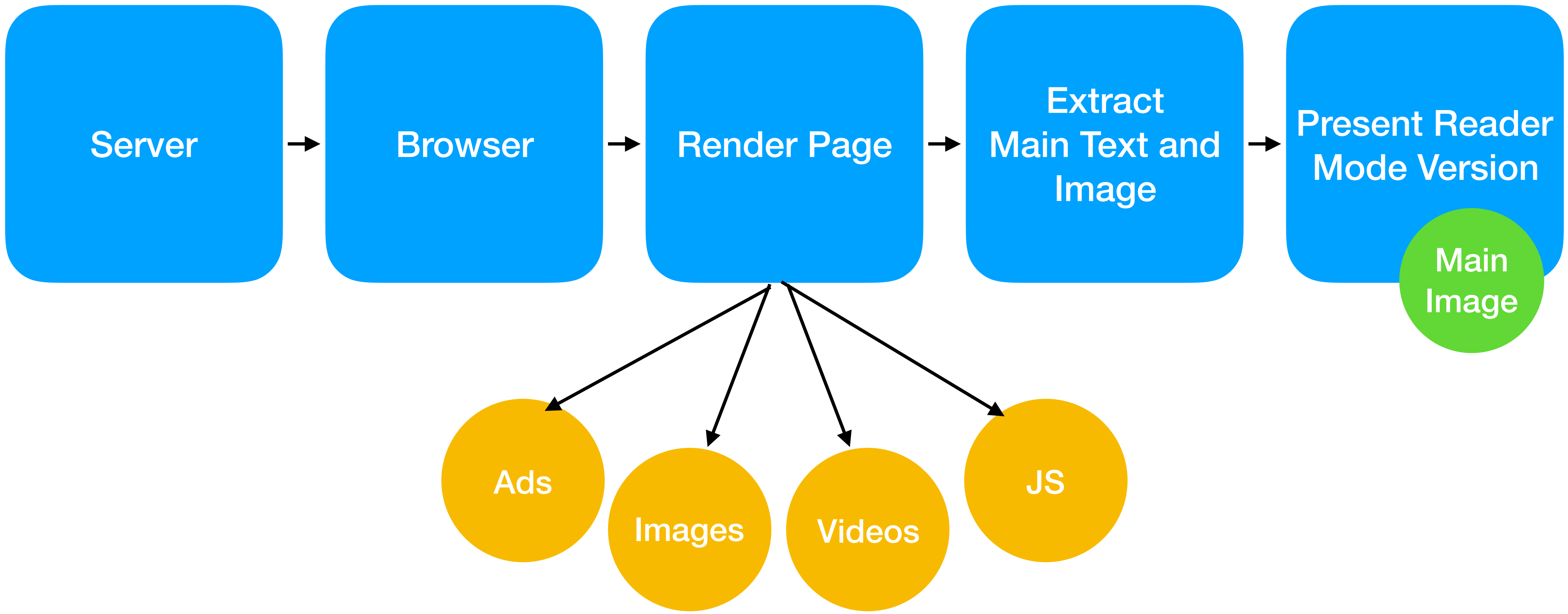
SpeedReader

- **Prevent tracking resources**
  JS, tracking pixels, iFrames…

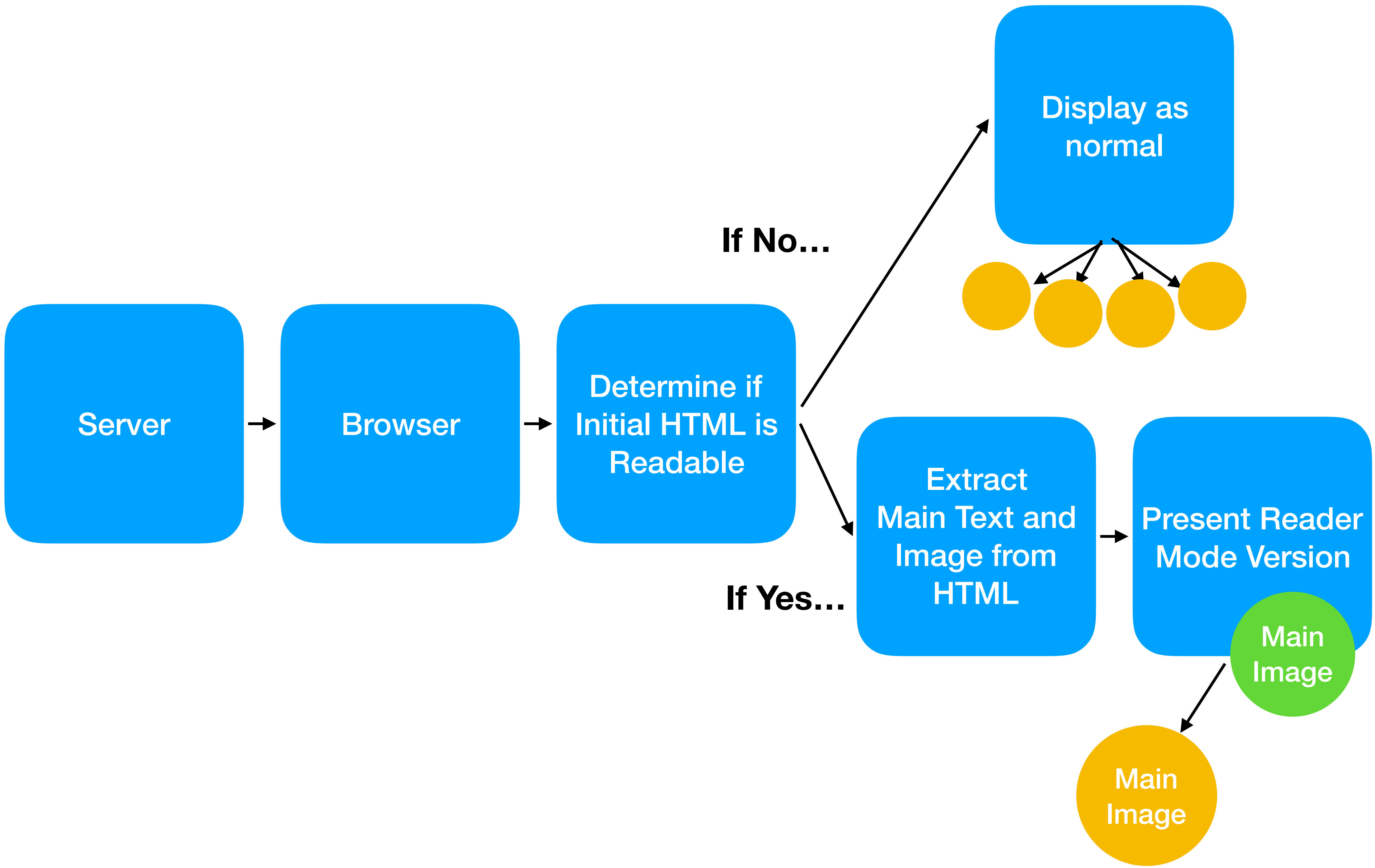
- **Most of a page isn't immediately useful**
  - Boilerplate: navigation, you might like…
  - Third party ads: often undesirable, offensive, or both
  - JavaScript: animations and distractions

- **Extract good content, don't block bad content**
  Focus on identifying the valuable parts of the page, not the harmful ones

Existing Reader Modes



SpeedReader



SpeedReader Results

Comparison of SpeedReader to standard browsing on a large sample of websites.

	3rd Party (Avg)	3rd Party (Median)	Scripts (Avg)	Scripts (Median)	Ads and Trackers (Avg)	Ads and Trackers (Median)
Default	117	63	83	51	63	24
SpeedReader	1	1	0	0	0	0

Not Possible with Extensions

- **Access Restrictions**
  Most browser’s don’t allow extensions to modify pages

- **Performance**
  ML classifier benefits from C++ implementation

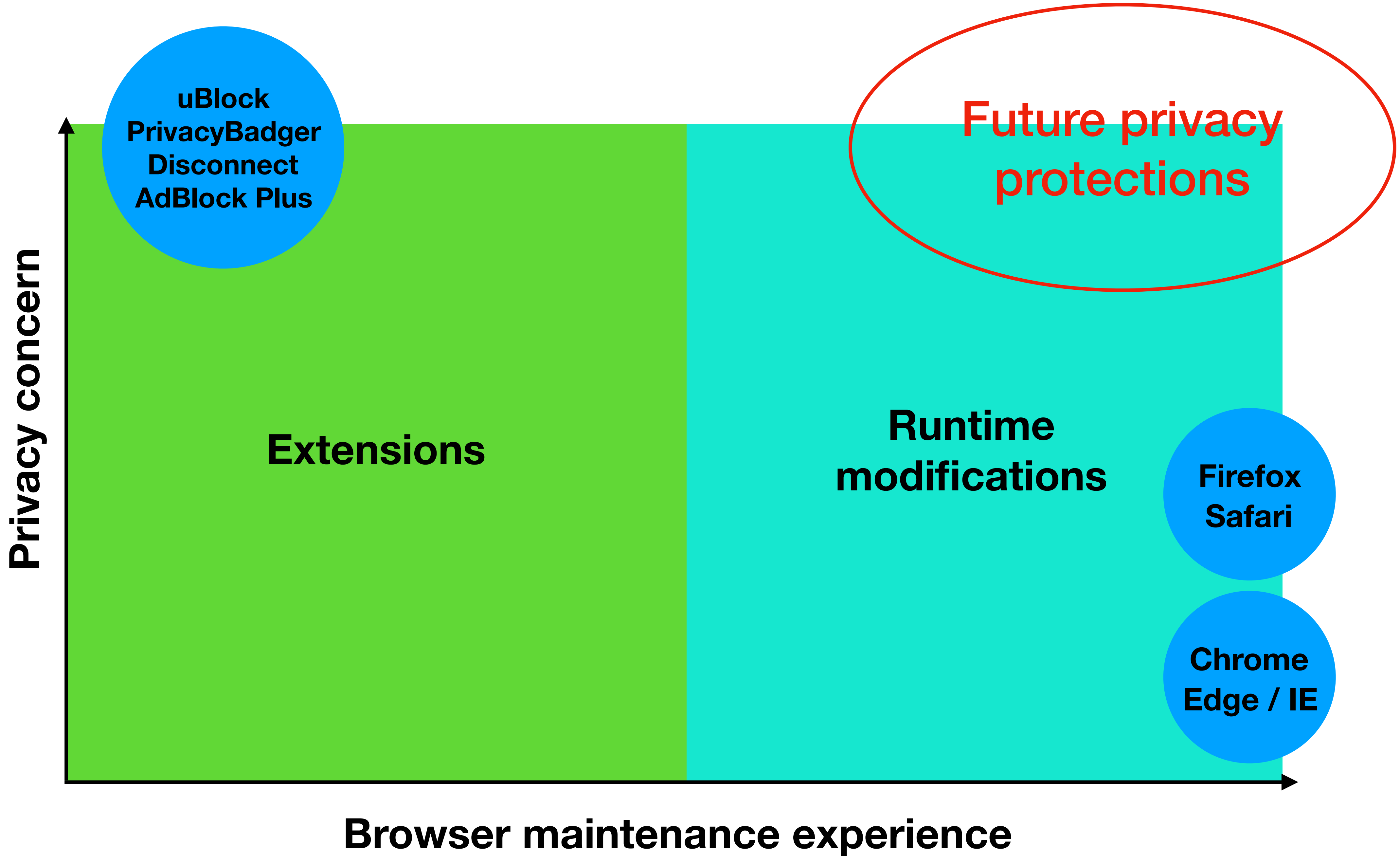
Outline

1. **Background**
   Extension focus in practical privacy tools

2. **Present**
   Privacy improvements require deep browser modifications

3. <u>**Next Steps**</u>
   Call to action, how to keep improving

Unclaimed Space



Better Privacy is Possible

- **New Browser Vendors**
  The “big four” aren’t the only game in town anymore

- **Many New Browsers are Privacy Focused**
  Privacy as top-level goal, willing to be aggressive

- **Eager to Collaborate**
  The new browsers are willing and interested to develop and maintain ambitious privacy protecting browser changes.

- **Reach Out!**
  Peter Snyder, Privacy Researcher
  pes@brave.com – @pes10k