Detecting JavaScript Races that Matter

Erdal Mutlu[1], Serdar Taşıran [1], Benjamin Livshits [2]

[1] Koç University, Istanbul

[2] Microsoft Research, Redmond

JavaScript Races: Is It Even Possible?

Asynchrony in JavaScript: A Short Primer

```
setInt
setTim
```

```
ck(doSomething);
```

Asynchronous XmlHttpRequest (XHR) Network Calls

A onous user events

  eact to button clicks

```
me/data.json”);
```

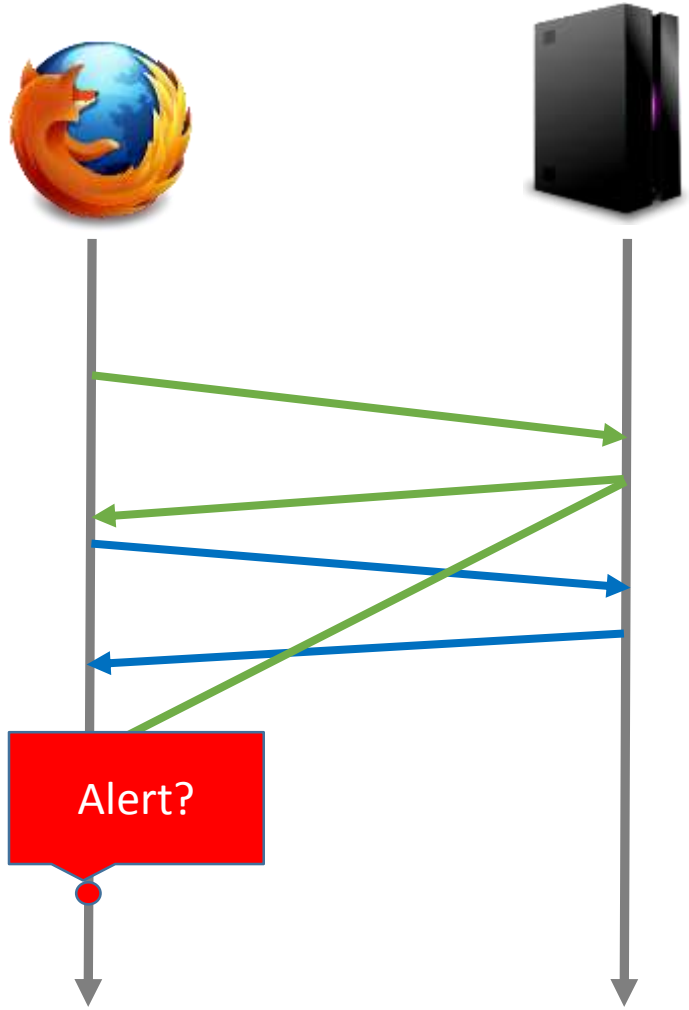
```
x  .onreadystatechange = parseResult(result);
```

```
$.getJSON(“http://server_name/data.json”,parseResult(result));
```

JavaScript Race Example: Memory

```
<html>
<script>
var localVar = “http://server_name/data.json”
var url1 = localVar;
$.getJSON( url1, function(result){
                localVar = getUrl(result);});
</script>
…
<script>
var url2 = localVar;
$.getJSON( url2, function(result){
                localVar = getUrl(result);});
</script>
…
<body>
alert(localVar);
</body>
</html>
```

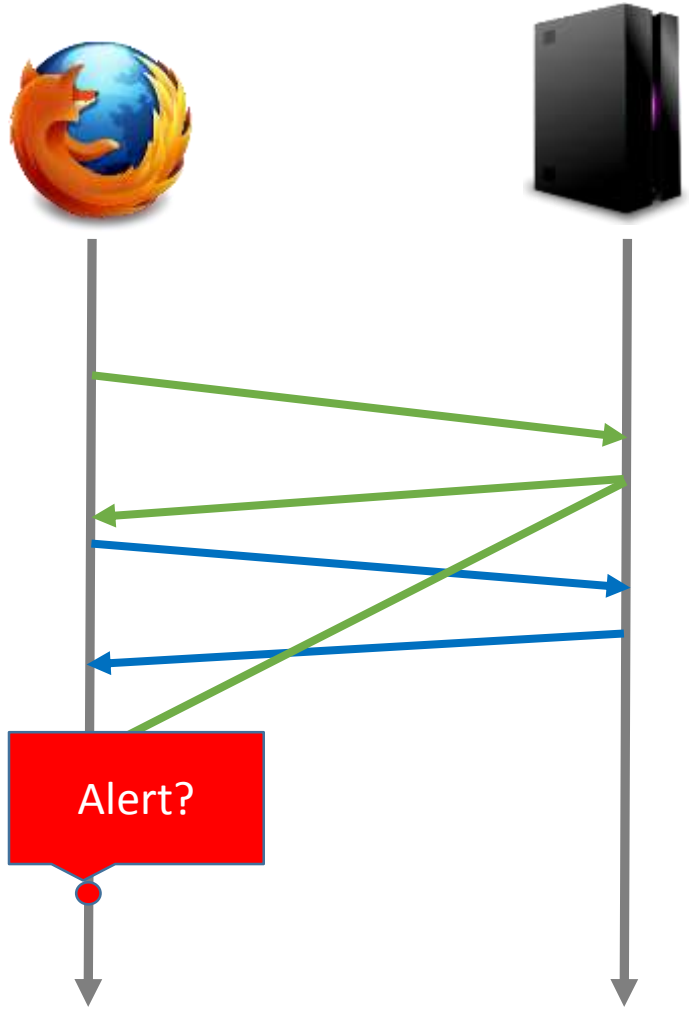
Refresh



Cookies: Example of Persistent Data Storage

JavaScript Race Example: Cookie

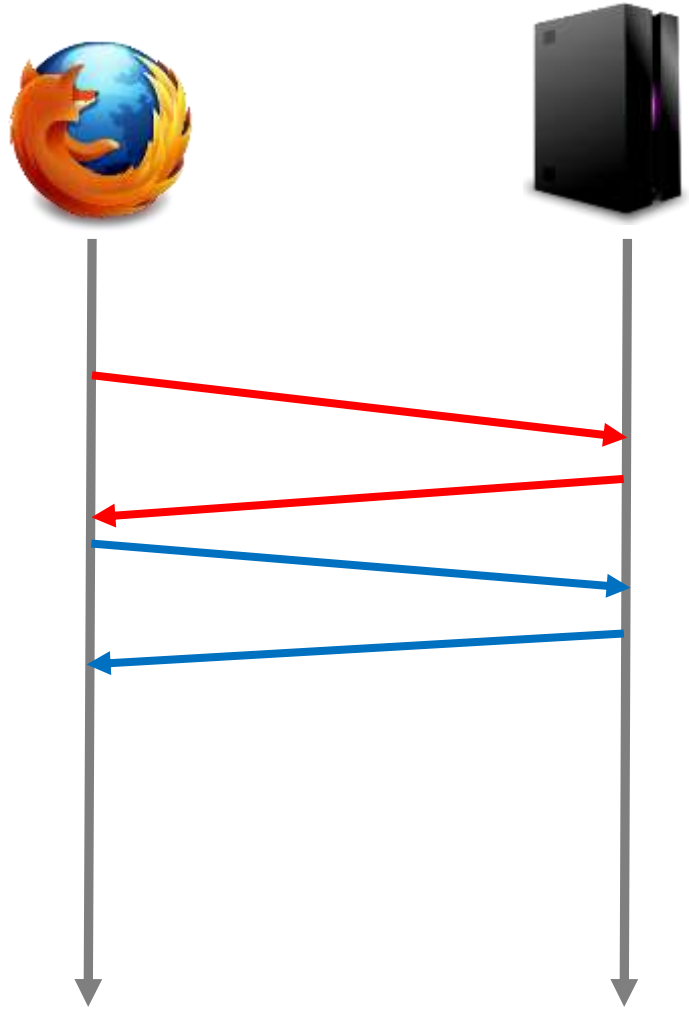
```
<html>
<script>
$.getJSON($.cookie('url'), function(result){
        $.cookie('url', getUrl(result));
});
</script>
…
<script>
$.getJSON($.cookie('url'), function(result){
        $.cookie('url', getUrl(result));
});
</script>
…
<body>
alert($.cookie('url'));
</body>
</html>
```



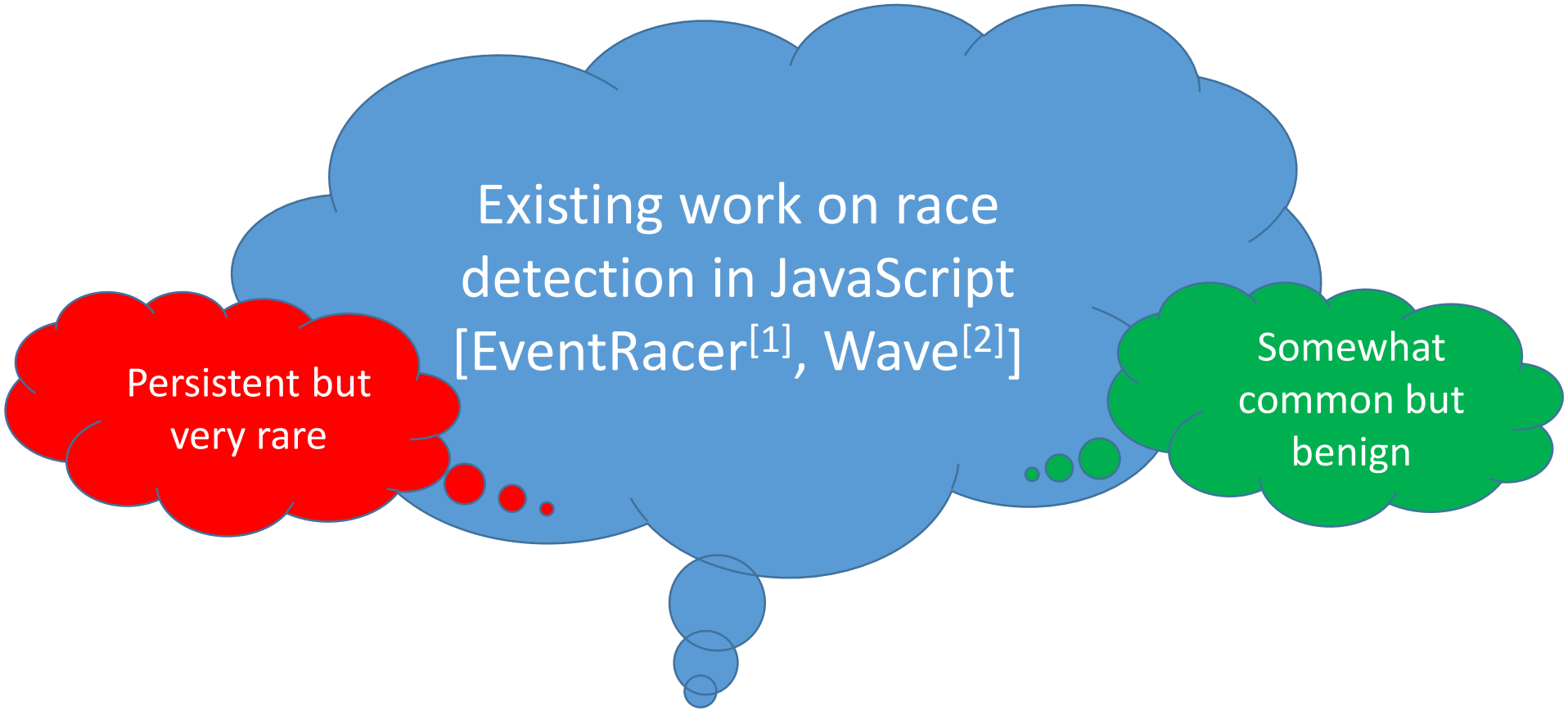
JavaScript Race Example: Cookie

```
<html>
<script>
$.getJSON($.cookie('url'), function(result){
        $.cookie('url', getUrl(result));
});
</script>
…
<script>
$.getJSON($.cookie('url'), function(result){
        $.cookie('url', getUrl(result));
});
</script>
…
<body>
alert($.cookie('url'));
</body>
</html>
```

Refresh



Not All JavaScript Races Are Made Equal



[1] Raychev et. al. “Effective race detection for event-driven programs”, OOPSLA’13
[2] Hong et. al. “Detecting concurrency errors in client-side JavaScript web applications”, ICST’14

Research Questions?

- **RQ1:** How common are races on ***persistent state*** such as `document.cookie`, `localStorage`, and side-effects such as `POST` requests?

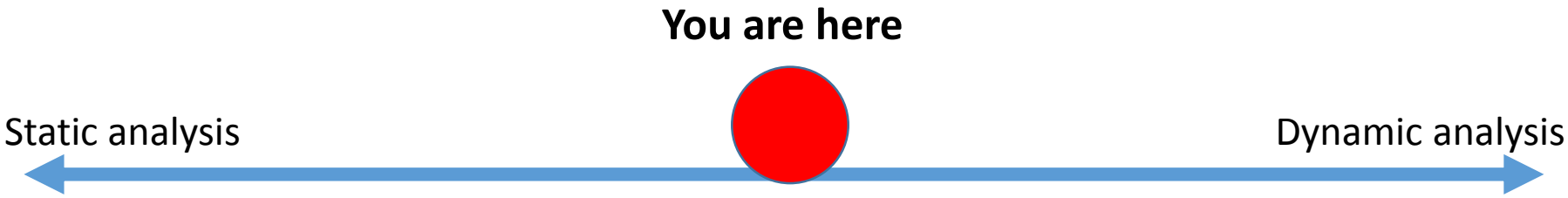
- **RQ2:** How common are races on ***session state***, such as `sessionStorage` which is cleared on browser restarts?

- **RQ3:** How common are races on ***transient state*** such as memory locations and DOM elements?

Analysis Technique

Analysis Tradeoffs

- Collect traces **dynamically** by instrumenting Firefox browser
- Analyze traces **statically** for possible data races



Firefox Instrumentation

- Instrumented Firefox for recording
  - Memory read-writes
  - Persistent state read-writes (`document.cookie`, `localStorage` etc.)
  - Session state read-writes (`sessionStorage`, `DOM state` etc.)
  - Asynchronous callback block begin-end

- Around 430 lines of instrumentation code spanning
  - Cross platform component objects (XPCOM)
  - Gecko (Layout Engine)
  - SpiderMonkey (JS Engine)

Firefox Instrumentation

Static source code

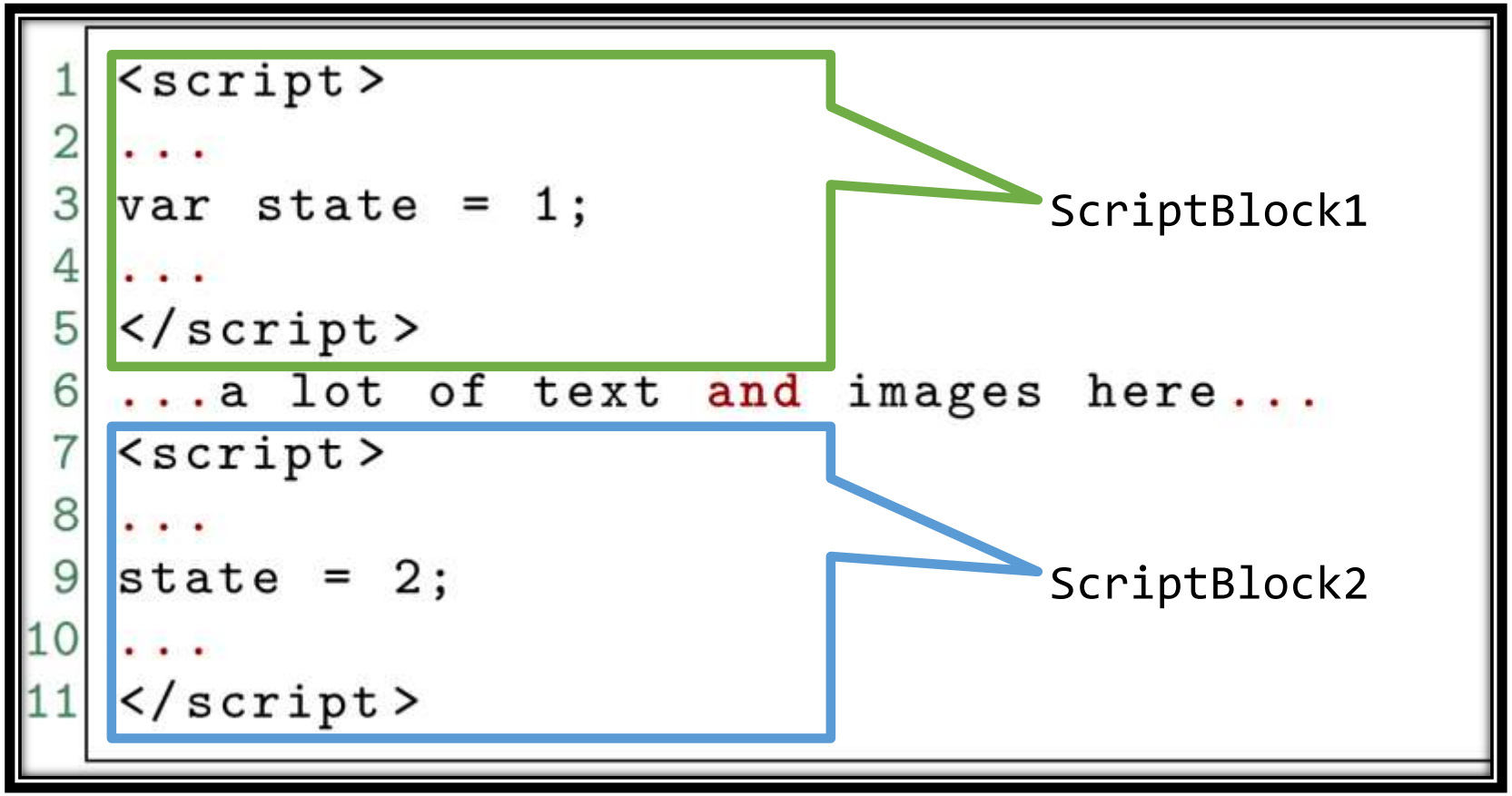
```
1  <html>
2   <script>                                          // 1
3    var xhr = new Xhr();
4    xhr.open('', false);
5    xhr.onreadystatechange = function(){              // 2
6      document.cookie = 'var1=1';
7    };
8    xhr.send();
9    //for(i=0;i<10000000; i++) console.trace(i);
10 </script>
11 ...
12 <!-- <input id='mydiv' /> -->
13 <script>                                          // 3
14   document.cookie = 'var1=2';
15 </script>
16 ...
17 <script>                                          // 4
18   var xhr2 = new Xhr();
19   xhr2.open('', false);
20   xhr2.onreadystatechange = function(){           // 5
21     document.cookie = 'var1=3';
22   };
23   xhr2.send();
24 </script>
25 </html>
26 Put your code here.
```

Runtime execution trace

```
1  READ PROP ID[99044816] = "open" :  JSFunction
2  XHR [0000001] Open GET
3  READ PROP ID[98817648] = "send" :  JSFunction
4  XHR [0000001] Send
5  XHR [0000001] Callback
6  BEGIN XHR_Callback [0000001]
7   READ ID[235205312] = document :  JSObject
8   WRITE PROP ID[242159440] = cookie :  JSInteger (1)
9   Cookie [1020D800] Write "var1=1"
10 END XHR_Callback [0000001]
11
12 READ ID[235205312] = document :  JSObject
13 WRITE PROP ID[242159440] = cookie : JSInteger (2)
14 Cookie [1020D800] Write "var1=2"
15
16 READ PROP ID[99044816] = "open" : JSFunction
17 XHR [0000002] Open GET
18 READ PROP ID[98817648] = "send" : JSFunction
19 XHR [0000002] Send
20 XHR [0000002] Callback
21 BEGIN XHR_Callback [0000002]
22  READ PROP ID[108330176] = "readyState": JSInteger(4)
23  READ ID[235205312] = "document" :  JSObject
24  WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
25  Cookie [1020D800] Write "var1=3"
26 END XHR_Callback [0000002]
```

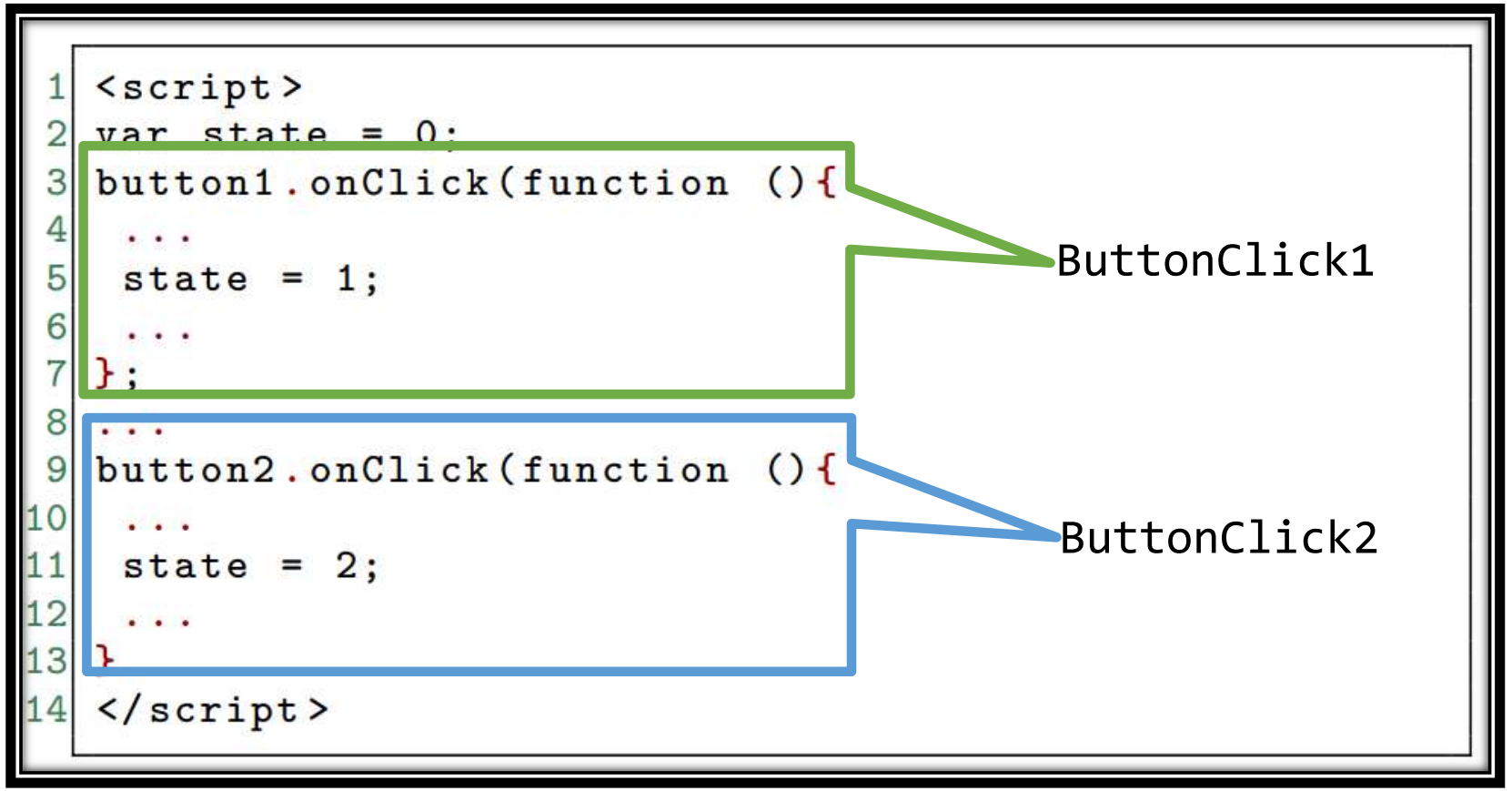
Happens-Before Relation in JavaScript

Happens-Before: Sequential Blocks



ScriptBlock1 ← ScriptBlock2

Happens-Before: User Interactions



ButtonClick1 ← ButtonClick2

Happens-Before: Chained XHR Blocks

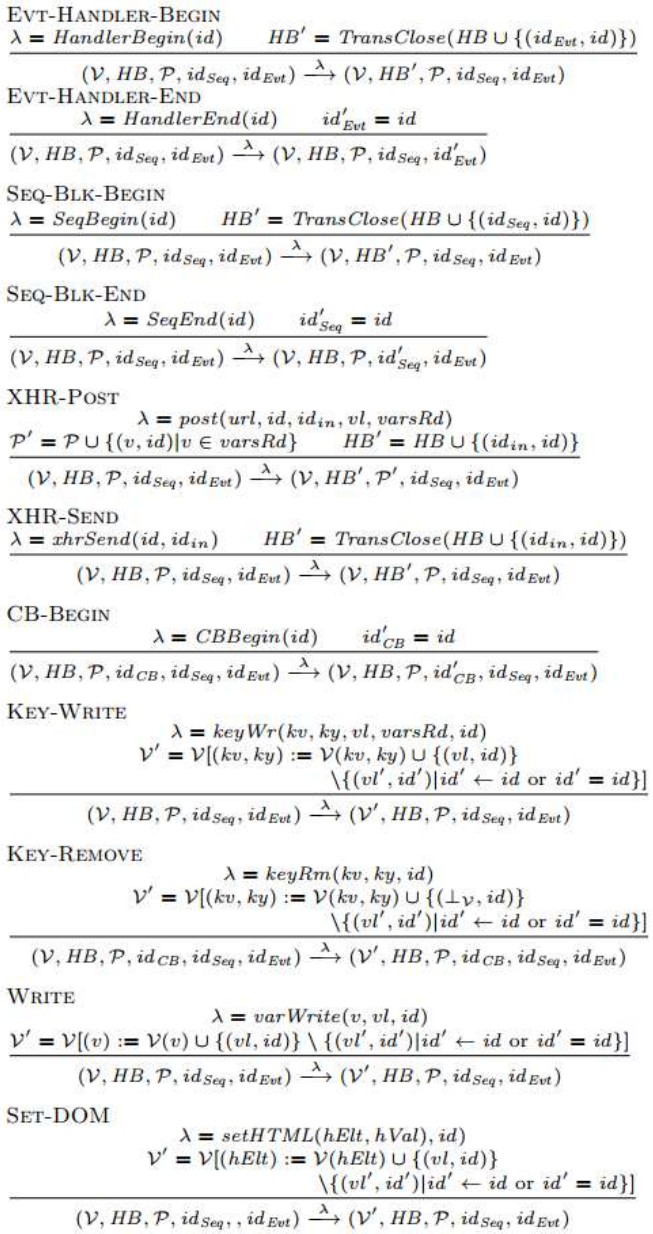
```
1  var xhr1 = new XmlHttpRequest();
2  xhr1.open("GET", "http://www.data.com/mydata.json");
3  xhr1.onreadystatechange = function(e, d){
4   var xhr2 = new XmlHttpRequest();
5   xhr2.open("GET", "http://www.data.com/data.json");
6   xhr2.onreadystatechange = function(e){
7   ...
8   };
9   xhr2.send(null);
10 };
11 xhr1.send(null);
```

xhr1.onreadystatechange ← xhr2.onreadystatechange

Analysis Rules

- Define potential races in terms of flow
  - Track data flow to sensitive locations
  - Determine when values propagated depends on event order

- General outline/intuition:
  - Algorithm parses trace line-by-line
  - Keep happens-before relation between blocks
  - Values written recorded into **memory maps**
  - **Merge values** written by concurrent blocks



Race Detection Mechanism

Cookie Key	Values
…	…

```
READ PROP ID[99044816] = "open" :  JSFunction
XHR [0000001] Open GET
READ PROP ID[98817648] = "send" :  JSFunction
XHR [0000001] Send
XHR [0000001] Callback
BEGIN XHR_Callback [0000001]
 READ ID[235205312] = document :  JSObject
 WRITE PROP ID[242159440] = cookie :  JSInteger (1)
 Cookie [1020D800] Write "var1=1"
 ...
 READ ID[235205312] = document :  JSObject
 WRITE PROP ID[242159440] = cookie :  JSInteger (1)
 Cookie [1020D800] Write "var1=5"
END XHR_Callback [0000001]

READ ID[235205312] = document :  JSObject
WRITE PROP ID[242159440] = cookie : JSInteger (2)
Cookie [1020D800] Write "var1=2"

READ PROP ID[99044816] = "open" : JSFunction
XHR [0000002] Open GET
READ PROP ID[98817648] = "send" : JSFunction
XHR [0000002] Send
XHR [0000002] Callback
BEGIN XHR_Callback [0000002]
 READ PROP ID[108330176] = "readyState": JSInteger(4)
 READ ID[235205312] = "document" :  JSObject
 WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
 Cookie [1020D800] Write "var1=3"
END XHR_Callback [0000002]
```

Race Detection Mechanism

Cookie Key	Values
…	…
var1	{1,[000001]}

```
1  READ PROP ID[99044816] = "open" :  JSFunction
2  XHR [0000001] Open GET
3  READ PROP ID[98817648] = "send" :  JSFunction
4  XHR [0000001] Send
5  XHR [0000001] Callback
6  BEGIN XHR_Callback [0000001]
7   READ ID[235205312] = document :  JSObject
8   WRITE PROP ID[242159440] = cookie :  JSInteger (1)
9   Cookie [1020D800] Write "var1=1"
10  ...
11  READ ID[235205312] = document :  JSObject
12  WRITE PROP ID[242159440] = cookie :  JSInteger (1)
13  Cookie [1020D800] Write "var1=5"
14 END XHR_Callback [0000001]
15
16 READ ID[235205312] = document :  JSObject
17 WRITE PROP ID[242159440] = cookie : JSInteger (2)
18 Cookie [1020D800] Write "var1=2"
19
20 READ PROP ID[99044816] = "open" : JSFunction
21 XHR [0000002] Open GET
22 READ PROP ID[98817648] = "send" : JSFunction
23 XHR [0000002] Send
24 XHR [0000002] Callback
25 BEGIN XHR_Callback [0000002]
26  READ PROP ID[108330176] = "readyState": JSInteger(4)
27  READ ID[235205312] = "document" :  JSObject
28  WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
29  Cookie [1020D800] Write "var1=3"
30 END XHR_Callback [0000002]
```

Race Detection Mechanism

Cookie Key	Values
…	…
var1	{5,[000001]}

Overwrite values with the happened-before block updates

```
1  READ PROP ID[99044816] = "open" :  JSFunction
2  XHR [0000001] Open GET
3  READ PROP ID[98817648] = "send" :  JSFunction
4  XHR [0000001] Send
5  XHR [0000001] Callback
6  BEGIN XHR_Callback [0000001]
7   READ ID[235205312] = document :  JSObject
8   WRITE PROP ID[242159440] = cookie :  JSInteger (1)
9   Cookie [1020D800] Write "var1=1"
10  ...
11  READ ID[235205312] = document :  JSObject
12  WRITE PROP ID[242159440] = cookie :  JSInteger (1)
13  Cookie [1020D800] Write "var1=5"
14 END XHR_Callback [0000001]
15
16 READ ID[235205312] = document :  JSObject
17 WRITE PROP ID[242159440] = cookie : JSInteger (2)
18 Cookie [1020D800] Write "var1=2"
19
20 READ PROP ID[99044816] = "open" : JSFunction
21 XHR [0000002] Open GET
22 READ PROP ID[98817648] = "send" : JSFunction
23 XHR [0000002] Send
24 XHR [0000002] Callback
25 BEGIN XHR_Callback [0000002]
26  READ PROP ID[108330176] = "readyState": JSInteger(4)
27  READ ID[235205312] = "document" :  JSObject
28  WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
29  Cookie [1020D800] Write "var1=3"
30 END XHR_Callback [0000002]
```

Race Detection Mechanism

Cookie Key	Values
…	…
var1	{5,[000001]}

```
1  READ PROP ID[99044816] = "open" :  JSFunction
2  XHR [0000001] Open GET
3  READ PROP ID[98817648] = "send" :  JSFunction
4  XHR [0000001] Send
5  XHR [0000001] Callback
6  BEGIN XHR_Callback [0000001]
7   READ ID[235205312] = document :  JSObject
8   WRITE PROP ID[242159440] = cookie :  JSInteger (1)
9   Cookie [1020D800] Write "var1=1"
10  ...
11  READ ID[235205312] = document :  JSObject
12  WRITE PROP ID[242159440] = cookie :  JSInteger (1)
13  Cookie [1020D800] Write "var1=5"
14 END XHR_Callback [0000001]
15
16 READ ID[235205312] = document :  JSObject
17 WRITE PROP ID[242159440] = cookie : JSInteger (2)
18 Cookie [1020D800] Write "var1=2"
19
20 READ PROP ID[99044816] = "open" : JSFunction
21 XHR [0000002] Open GET
22 READ PROP ID[98817648] = "send" : JSFunction
23 XHR [0000002] Send
24 XHR [0000002] Callback
25 BEGIN XHR_Callback [0000002]
26  READ PROP ID[108330176] = "readyState": JSInteger(4)
27  READ ID[235205312] = "document" :  JSObject
28  WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
29  Cookie [1020D800] Write "var1=3"
30 END XHR_Callback [0000002]
```

Race Detection Mechanism

Cookie Key	Values
…	…
var1	{5,[000001]}, {2,[000000]}

Merge values from concurrent block updates

Multiple values in a state points out to a presence of **scheduling dependencies**

```
1  READ PROP ID[99044816] = "open" :  JSFunction
2  XHR [0000001] Open GET
3  READ PROP ID[98817648] = "send" :  JSFunction
4  XHR [0000001] Send
5  XHR [0000001] Callback
6  BEGIN XHR_Callback [0000001]
7   READ ID[235205312] = document :  JSObject
8   WRITE PROP ID[242159440] = cookie :  JSInteger (1)
9   Cookie [1020D800] Write "var1=1"
10  ...
11  READ ID[235205312] = document :  JSObject
12  WRITE PROP ID[242159440] = cookie :  JSInteger (1)
13  Cookie [1020D800] Write "var1=5"
14 END XHR_Callback [0000001]
15
16 READ ID[235205312] = document :  JSObject
17 WRITE PROP ID[242159440] = cookie : JSInteger (2)
18 Cookie [1020D800] Write "var1=2"
19
20 READ PROP ID[99044816] = "open" : JSFunction
21 XHR [0000002] Open GET
22 READ PROP ID[98817648] = "send" : JSFunction
23 XHR [0000002] Send
24 XHR [0000002] Callback
25 BEGIN XHR_Callback [0000002]
26  READ PROP ID[108330176] = "readyState": JSInteger(4)
27  READ ID[235205312] = "document" :  JSObject
28  WRITE PROP ID[242159440] = "cookie" : JSInteger (3)
29  Cookie [1020D800] Write "var1=3"
30 END XHR_Callback [0000002]
```

Evaluation

Evaluation

- Used 26 web sites from Alexa top 4,000 that use XHRs heavily
- Collected traces by browsing web sites manually using instrumented Firefox
- Analyzed with our race detection mechanism

Benchmarks: Web Sites That Use XHR Heavily

	XHR				Persistent state writes					Session state writes			DOM writes			
Web site	XHR open	XHR send	XHR Callbacks	Nested XHR Count	Cookie	Nested Cookie	Nested XHR POSTs	localStorage	Nested localStorage	sessionStorage	Nested session Storage		Set innerHtml	Nested Set innerHtml	INPUT Element	Nested INPUT
mlb.mlb.com	25	25	92	0	68	2	0	28	2	30	2		93	46	39	0
wireless.att.com	22		72	3	79	4	0	46		11	2		902	847	60	5
aljazeera.net	12		32	2	24	0	0	6		0	0		1,095	29	46	7
bild.de	32	32		1	75	11	0	515			2			17	164	0
eltiempo.com	10	10			81	0	0	521					160	2	72	0
fedex.com.us	10	10	1			1	0	2					251	25	172	0
fujitv.co.jp	41	41	122			0	0	27			0		172	22	34	0
gazetta.it	22	22	67				0	68			0		119	24	52	0
girlsgogames.com	18	1											21	8	43	0
imdb.com	4												66	3	42	0
milliyet.com.tr	7												51	19	210	143
myvideo.de	20												86	0	23	0
nasa.gov	35												703	82	62	8
ntv.com.tr	10												112	3	66	0
nytimes.com	8												35	0	46	0
optimum.net	29												426	258	182	21
politico.com	27	2											110	19	248	10
premierleague.com	21	21	101	0	31	0	0	0	0	0	0		223	11	47	0
radikal.com.tr	13	11	20	0	111	0	0	33	0	2	0		85	5	59	0
sports.ru	28	28	90	1	48	0	0	84	0	0	0		128	58	52	1
sporx.com	5	5	0	0	33	0	0	31	0	0	0		49	0	38	0
tvguide.com	15	15	26	2	90	6	0	15	0	23	0		728	586	57	0
welt.de	23	23	44	1	112	0	0	366	0	4	0		546	19	68	0
zaobao.com	37	37	121	0	83	0	0	0	0	0	0		127	36	125	78

DOM state updated frequently

Trace Statistics

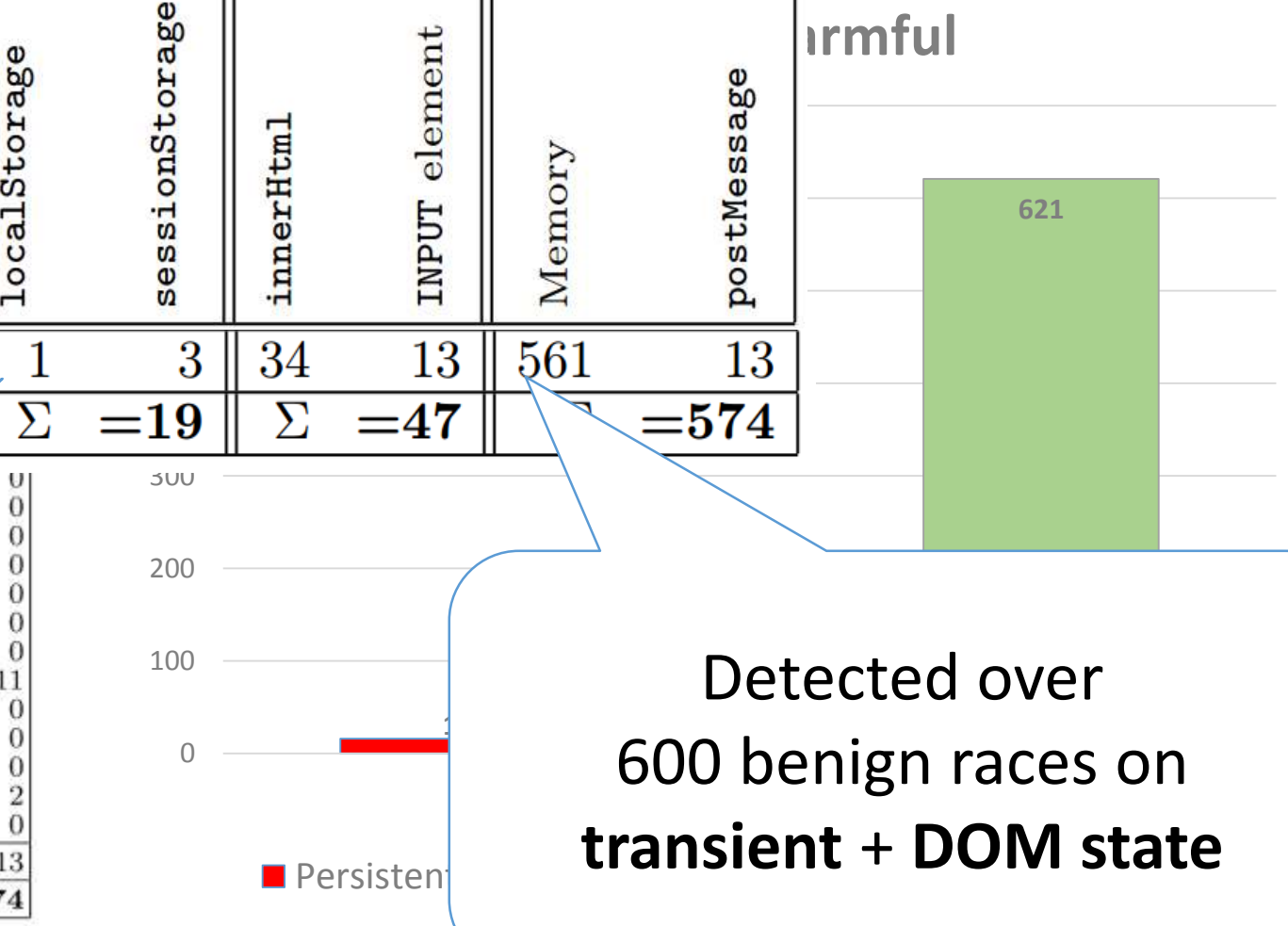
Website	Trace Size (MB)	Compressed Trace Size (MB)	Browsing Time (sec)	Analysis Time (sec)
www.imdb.com	14.28	1.21	58	2
www.zaobao.com	30.75	1.40	38	6
news.qq.com	30.77	1.01	78	4
www.sporx.com	37.49	1.92	49	6
www.nytimes.com	40.58	1.26	47	8
mlb.mlb.com	142.81		4.61	81	21
www.fedex.com.us		3.30	58	19
www.milliyet.com	95.78	3.93	133	16
www.m	104.82	4.38	121	
	125.11	5.39		

Size of the traces are m
small and well sui
compressio

The analysis can be applied online given the analysis time vs browsing time ratio

Detected JS Races

Web site	document.cookie	localStorage	sessionStorage	innerHtml	INPUT element	Memory	postMessage
Totals	15	1	3	34	13	561	13
			Σ =19		Σ =47		=574



Detected only
19 harmful races on
persistent + session state

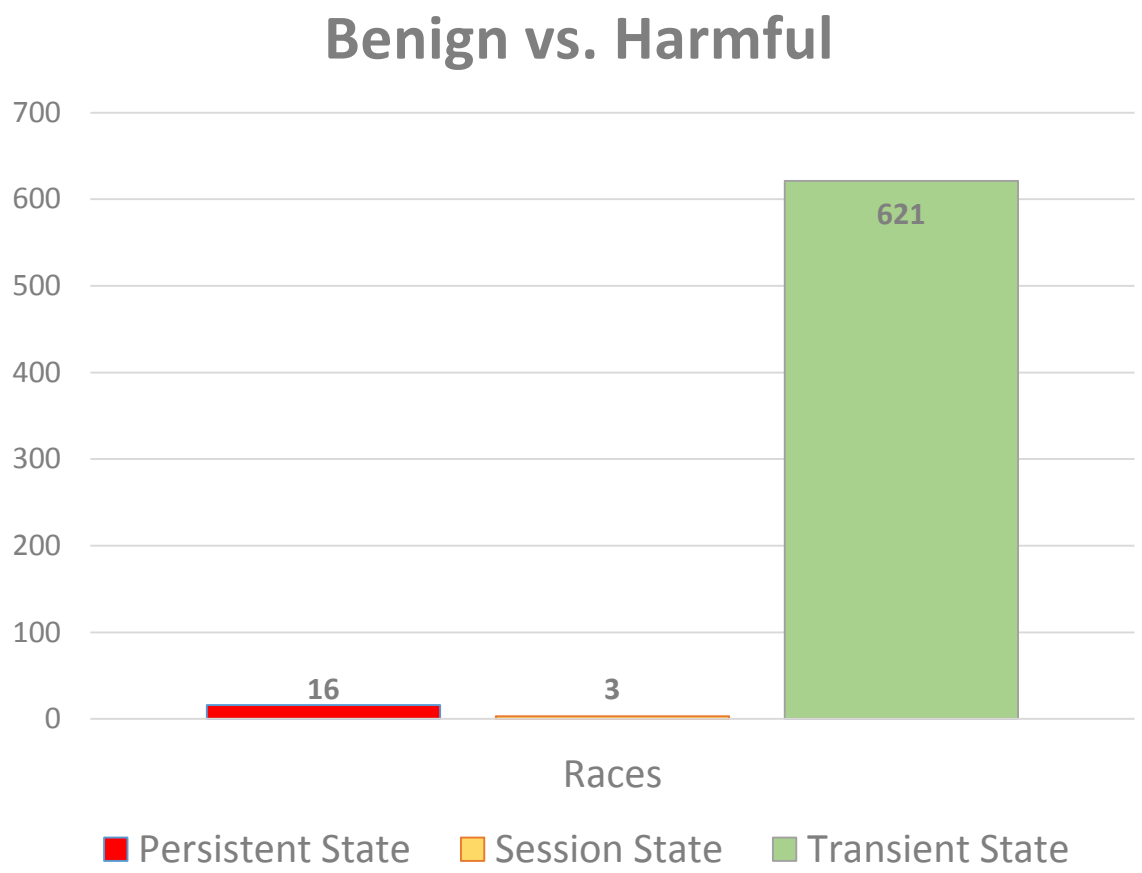
Detected over
600 benign races on
transient + DOM state

Conclusions

- We advocate a differentiation between **benign** and **harmful** data races in JavaScript web applications
- We propose a lightweight hybrid exploration algorithm for finding data races in runtime traces of JavaScript programs
- We find and investigate a total of 19 harmful and 621 benign races in 26 web sites, with only 2 observed false positives

Returning to Research Questions

- **RQ1:** Races on persistent state are quite uncommon, with only 16 for 26 sites
- **RQ2:** Races on session state are also uncommon (only 3 races observed), in part because session state is used not as frequently as cookies
- **RQ3:** Races on transient state are considerably more frequent (527 for 26 sites)



Replication Package

- Replication package available with following contents:
  - Race detector
  - Instrumented Firefox executable
  - Collected traces of our benchmark web sites
  - Evaluation scripts

- msrc.ku.edu.tr/projects/detecting-javascript-races-that-matter

QUESTIONS?

False Positives: optimum.net

```
1  XHR [126F0800] Send
2  ...
3  READ ID[235205312] = document :  JSObject
4  WRITE PROP ID[242159440] = cookie : JSString
5  Cookie [1020D800] Write "fsr.s=%7B%22v2%22%7D"    (1)
6  ...
7  READ ID[235205312] = document :  JSObject
8  WRITE PROP ID[242159440] = cookie : JSString
9  Cookie [1020D800] Write "fsr.s=%7B%22v2%22%2C%7D"(2)
10 ...
11 XHR [126F0800] Callback
12 BEGIN XHR_Callback
13  ...
14  READ ID[235205312] = document :  JSObject
15  READ PROP ID[242159440] = cookie :  JSInteger — (3)
16        "fsr.s=%7B%22v2%22%2C%7D"
17  ...
18  READ ID[235205312] = document :  JSObject
19  WRITE PROP ID[242159440] = cookie : JSString
20  Cookie [1020D800] Write
21     "fsr.s=%7B%22v2%22%2C%22%3A1%7D" (4)
22 END XHR_Callback
```