How we make JavaScript widgets scream

John Hjelmstad & Malte Ubl
Software Engineers

Google 12


+1


Share

elopers

Search

ucts Events Showcase Live Groups

developers everywhere

+1 17k

Showcase

Be inspired by how others are using Google technologies.

Share your Story

What inspires you as a developer? Share your story—we're listening.

>5 Billion hpr./day

X Billion cm^2

X Billion seconds

How widgets work

<iframe></iframe>

<profit>

Bootstrap /js/plusone.js

Allows rolling out changes quickly. Loads more JS that is infinitely cached.

Ideally tiny | Relatively short cache time

source: Apache 2.0 licensed http://twitter.github.com/bootstrap/index.html

On performance

At Google we are obsessed about speed.

Gaining page weight

source: CC http://photography.mojado.com/archives/2004/04/27/chocolate_cake.php

Social widget value system

Value provided by widget

>

Additional page load time

Topic of this presentation

Measuring performance

Pillars of performance measurement

Objective
Analytics

see http://goo.gl/hZOch

Perceived
QA & User tests
Indirect metrics

Raw metal
Benchmarking
Profiling

Analytics

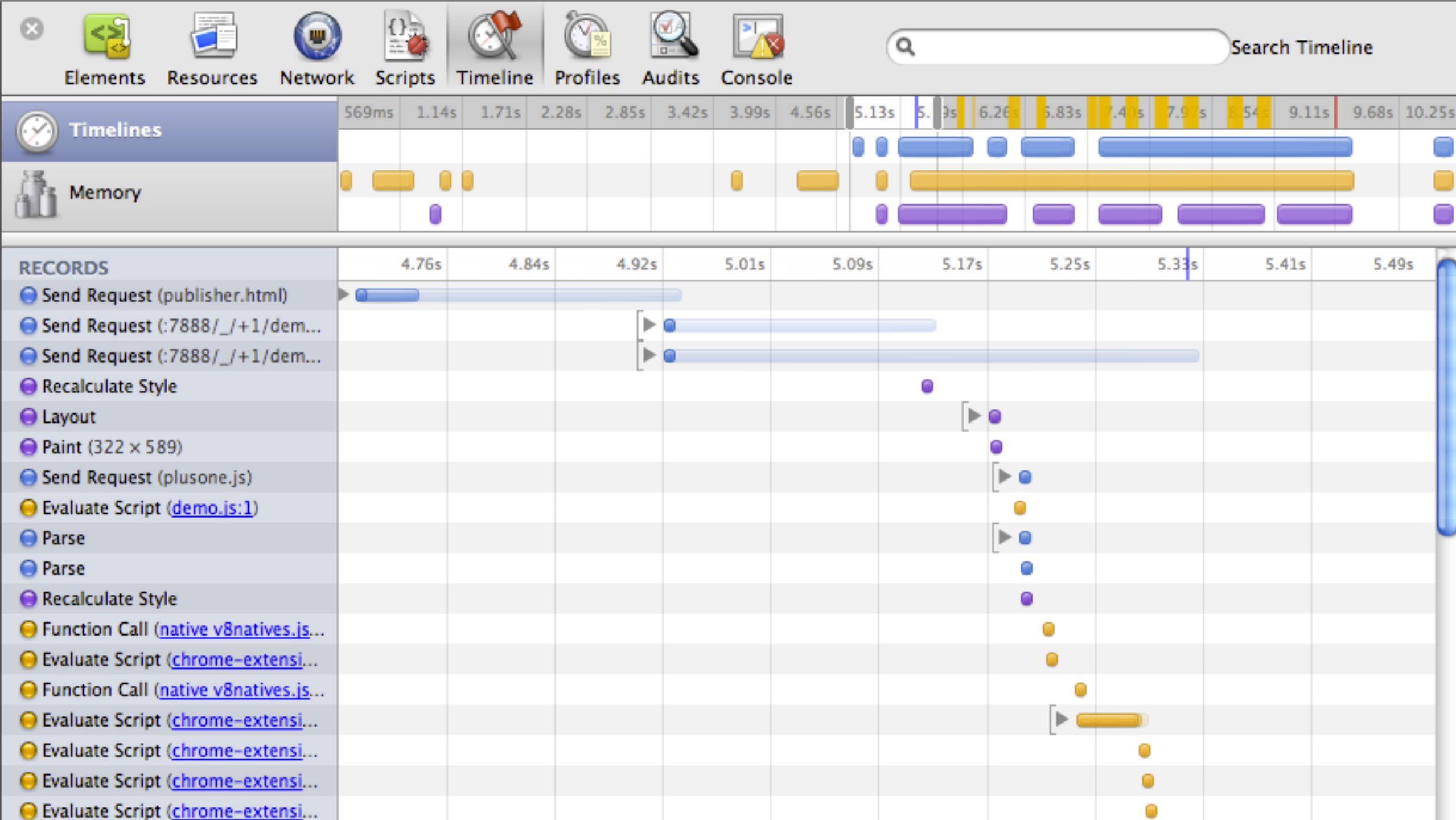
Action	Variable	Median (ms)
+1 Button	Widget Display Time	**1,345**

source: Only exemplary. Made up on the spot like most statistics.

Experiments

Action	Variable	Experiment	Median (ms)
+1 Button	Widget Display Time	A	1,345
+1 Button	Widget Display Time	B	**1,109**

source: Only exemplary. Made up on the spot like most statistics.


Elements
Resources
Network
Scripts
Timeline
Profiles
Audits
Console
Search Timeline
Timelines
Memory
569ms
1.14s
1.71s
2.28s
2.85s
3.42s
3.99s
4.56s
5.13s
6.26s
6.83s
9.11s
9.68s
10.25s
RECORDS
4.76s
4.84s
4.92s
5.01s
5.09s
5.17s
5.25s
5.33s
5.41s
5.49s
Send Request (publisher.html)
Send Request (:7888/_/+1/dem...
Send Request (:7888/_/+1/dem...
Recalculate Style
Layout
Paint (322 × 589)
Send Request (plusone.js)
Evaluate Script (demo.js:1)
Parse
Parse
Recalculate Style
Function Call (native v8natives.js...
Evaluate Script (chrome-extensi...
Function Call (native v8natives.js...
Evaluate Script (chrome-extensi...
Evaluate Script (chrome-extensi...
Evaluate Script (chrome-extensi...
Evaluate Script (chrome-extensi...

Performance best practice themes

source: CC http://www.flickr.com/photos/blank22763/4089930308/in/set-72157622643766597 / person souders@google.com

Reduce HTTP request cost

Combine resources
e.g. CSS Sprites
Resource inlining

Same origin
DNS, SSL

Connection sharing
SPDY, MHTML

Maximize parallelization and load asynchronously

Parallel downloads
Async script loading
Preload resources

Widgets: Play Nice
Async embedding
Never block

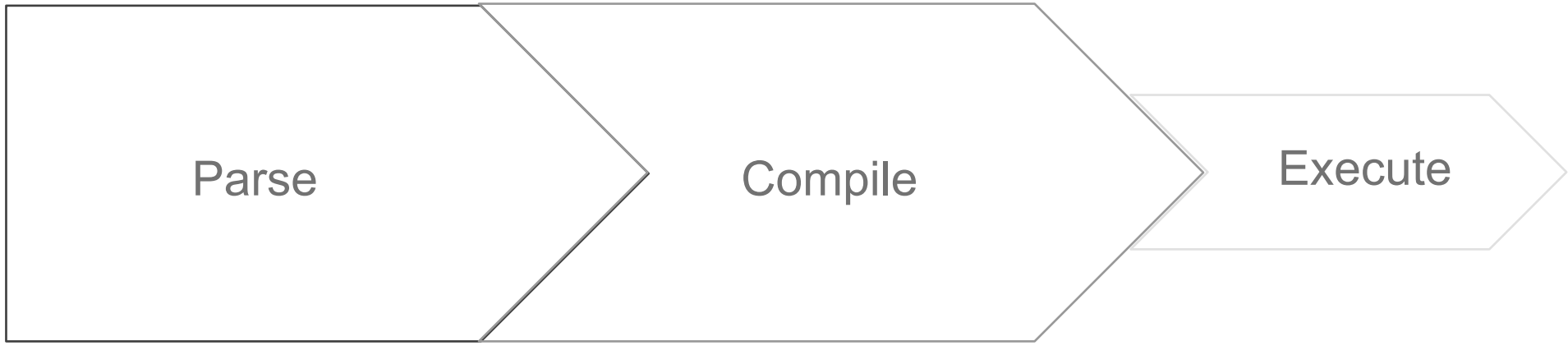
Optimize for Perception

Visible First
Show pixels first, interaction later

Lazyness
Download, compile and execute upon intent

What surprised us

Compiling JavaScript is slow



~ 1 ms per 1 KB of JS

source: CC http://www.fotopedia.com/items/flickr-3460551193

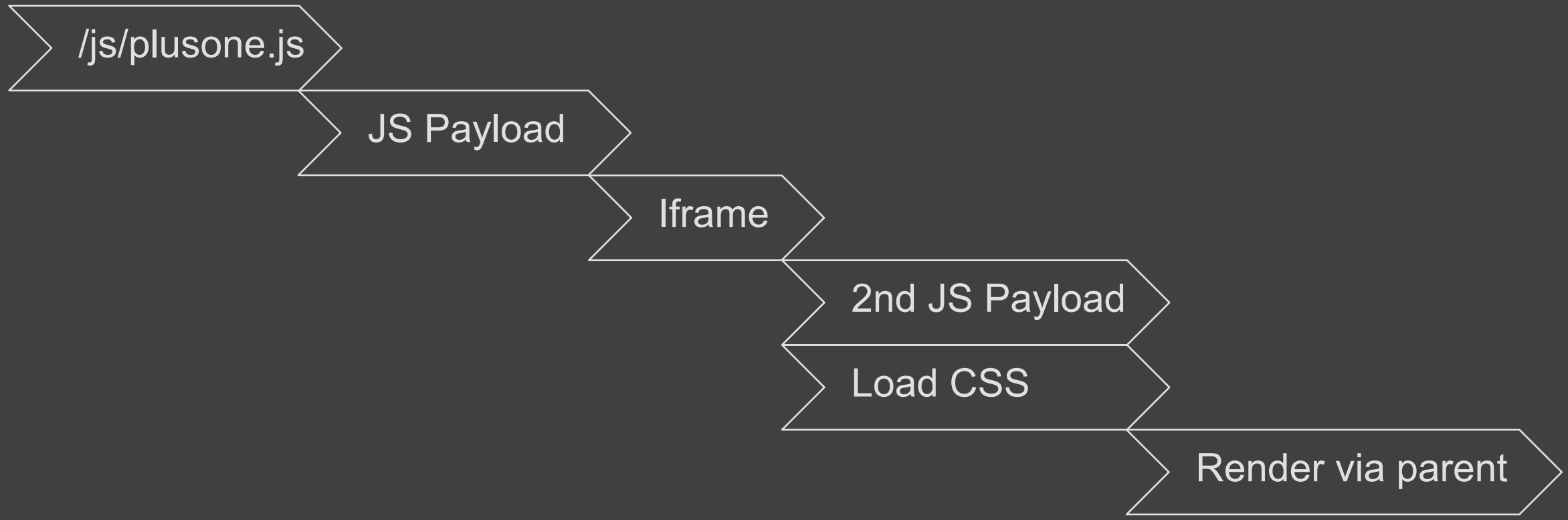
Loading the +1 button

We did not exactly start out very fast.

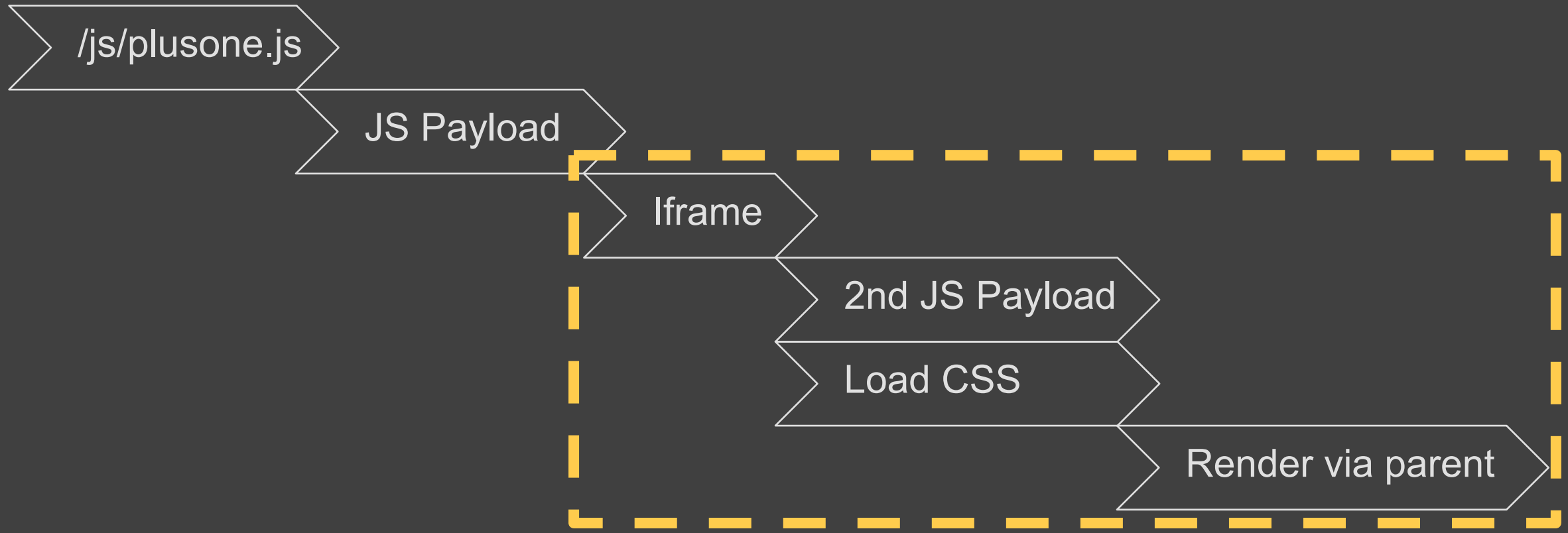
6 step waterfall

source: zivh@ https://pl[illegible]00568016328400036804/al[illegible]ms/5626241160455661297/5626241429560893298?banner=pwa

Waterfall before optimization



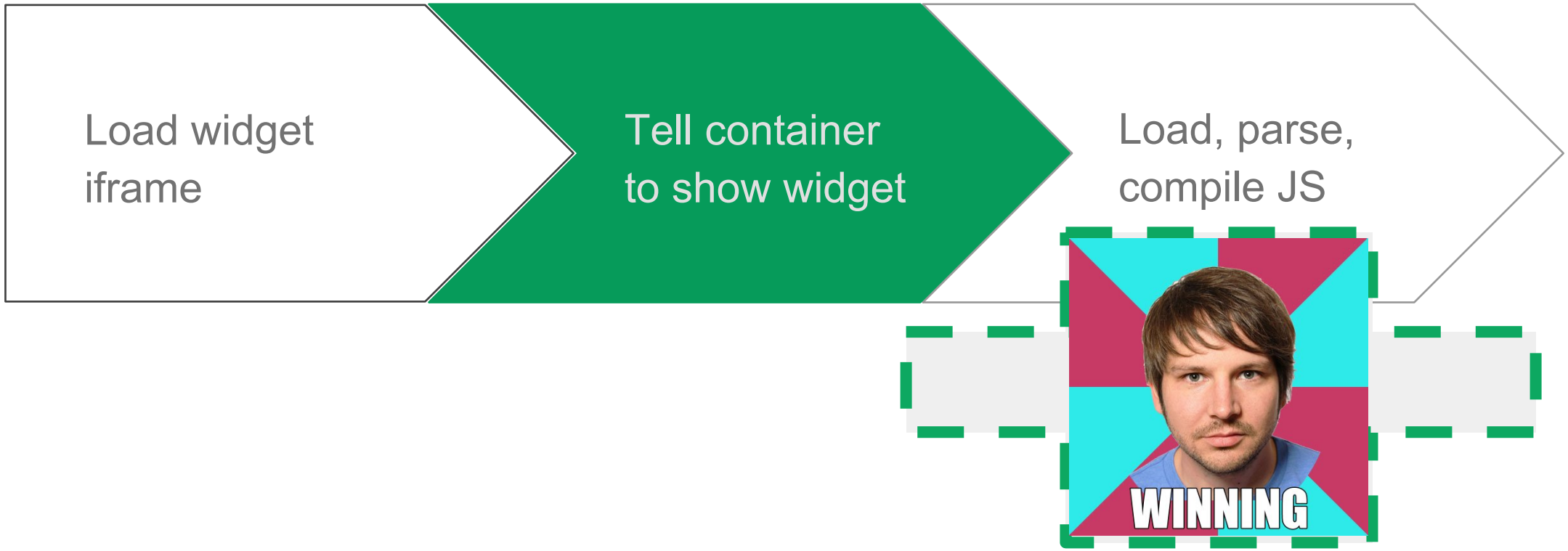
"Ping"


/js/plusone.js
JS Payload
Iframe
2nd JS Payload
Load CSS
Render via parent

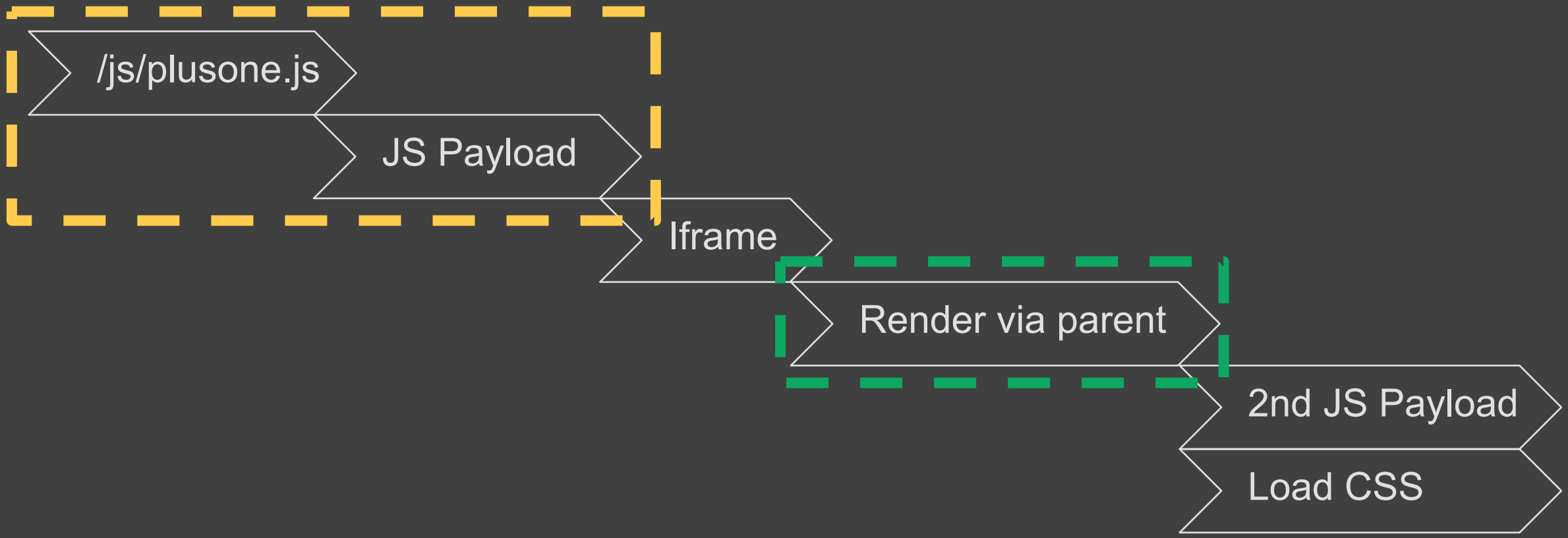
Before


Load widget iframe
Load, parse, compile JS
Tell container to show widget

Ping


Load widget iframe
Tell container to show widget
Load, parse, compile JS
WINNING

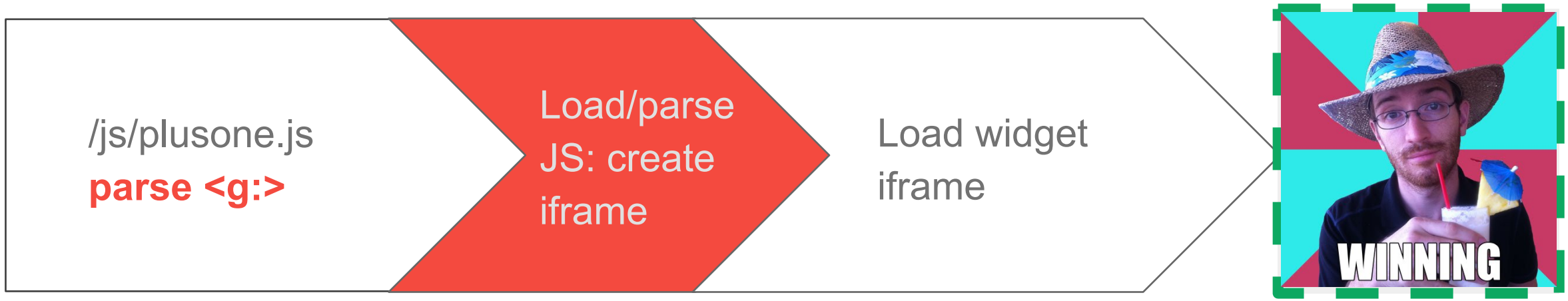
"Widget detection"



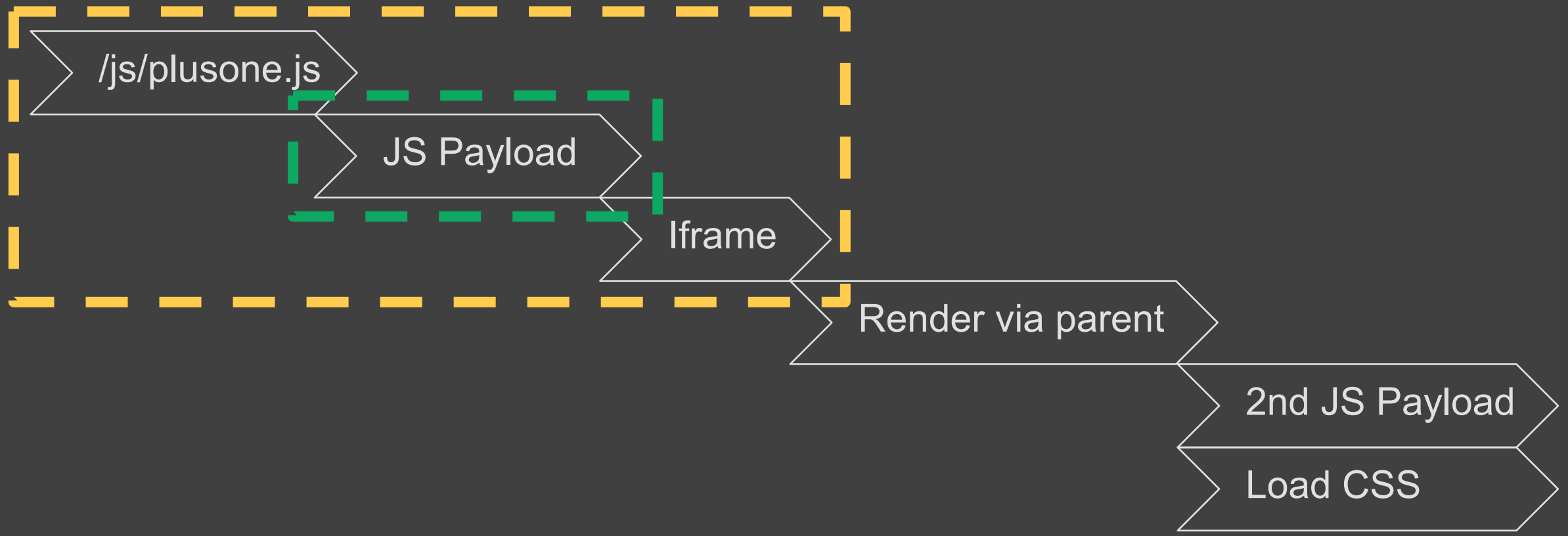
Before


/js/plusone.js
Load/parse JS:
parse <g:>
create iframe
Load widget
iframe

Widget Detection



"Bootstrap rendering"



Before


/js/plusone.js
parse <g:>
Load JS:
create iframe
(inc. style)
Load widget
iframe
Show widget

After bootstrap rendering

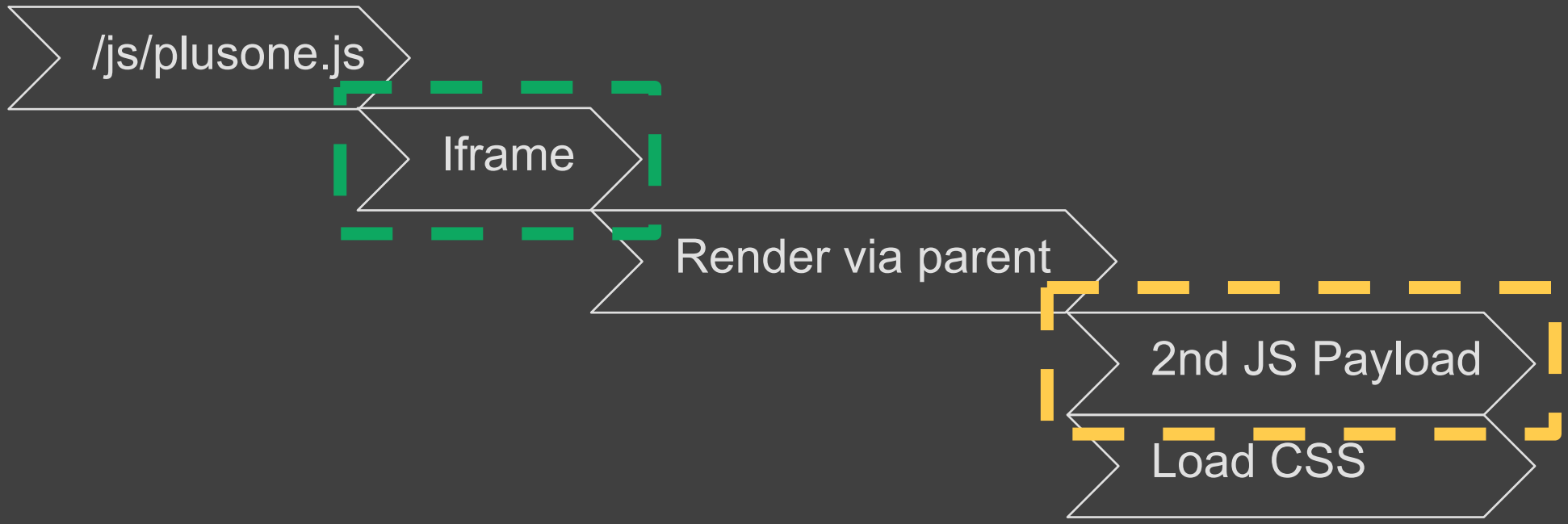
/js/plusone.js
parse <g:>
create iframe

Load JS

Load widget
iframe
(inc. style)

Show widget

"Lazy JS evaluation"



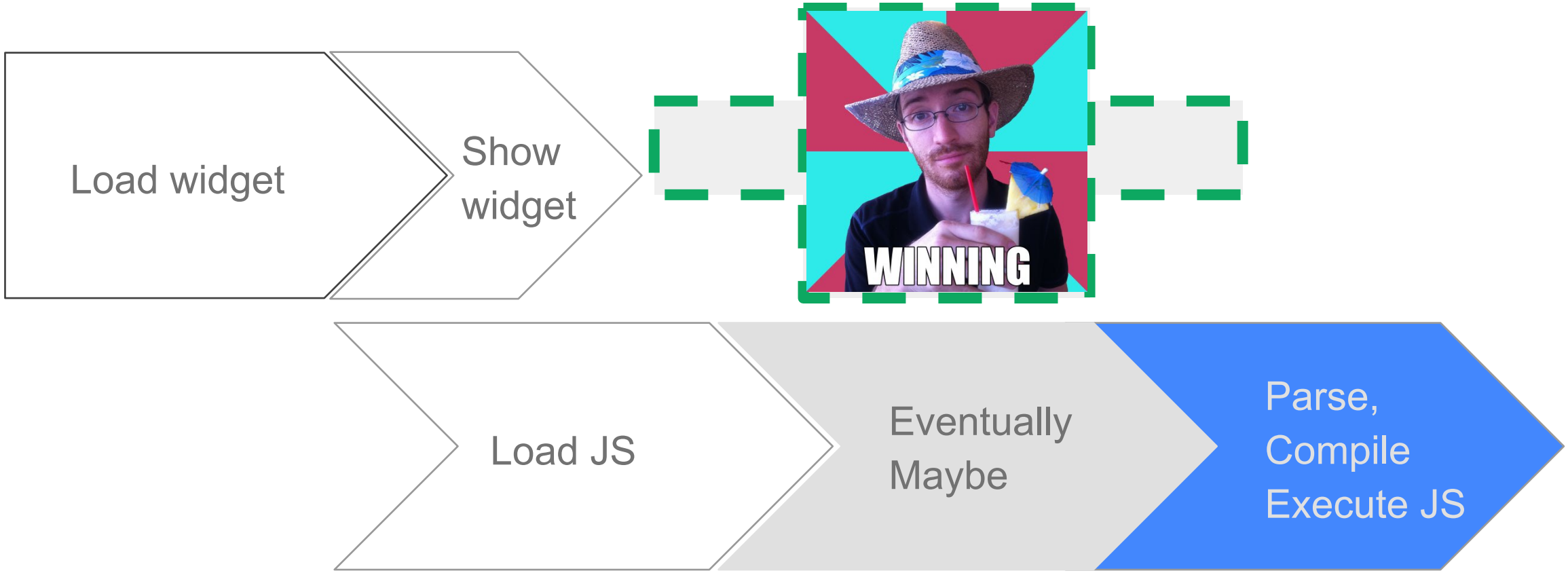
Before

Code sample

JS

```
$.get('/some/file.js', function(js) {
  $(window).bind('mouseover touchstart keydown',
      function() {
    window.eval(js);
  });
});
```

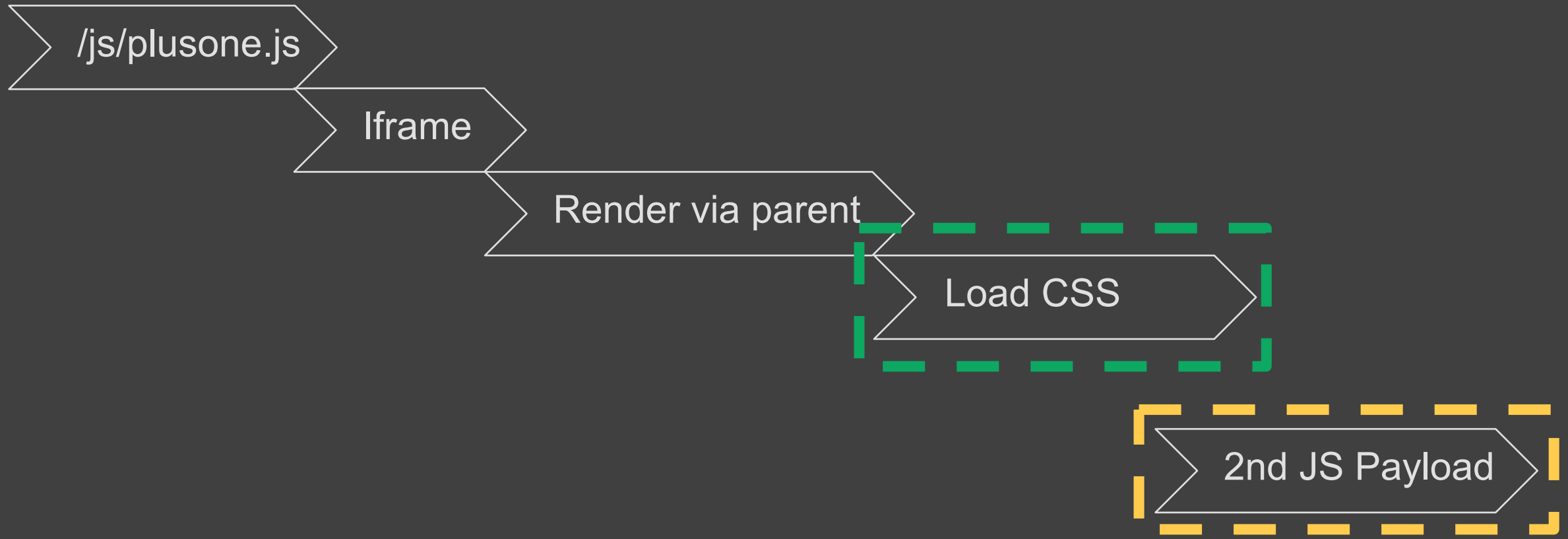
After lazy JS evaluation

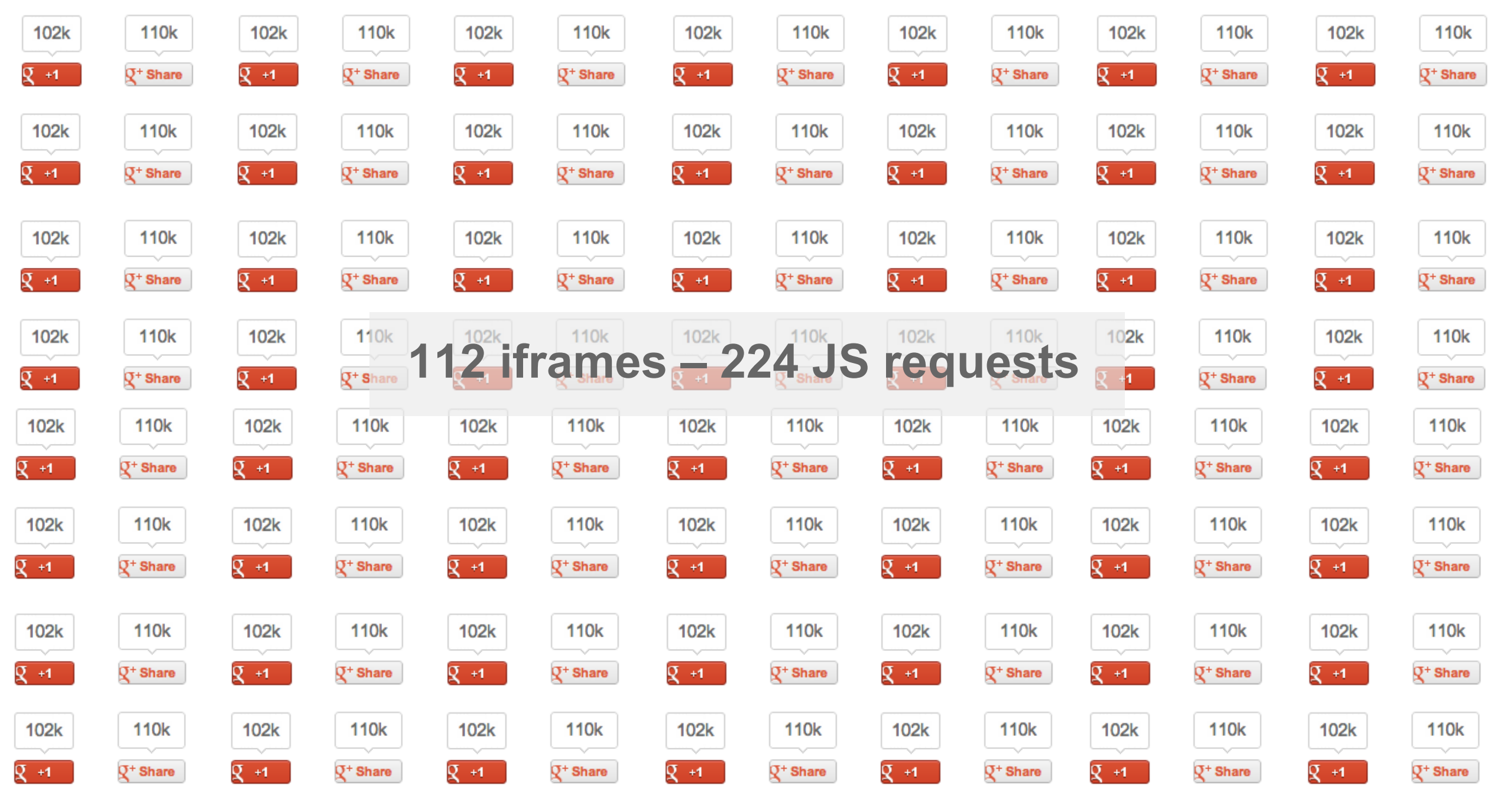


Saves lots of CPU

source: CC http://www.geograph.org.uk/photo/1879702

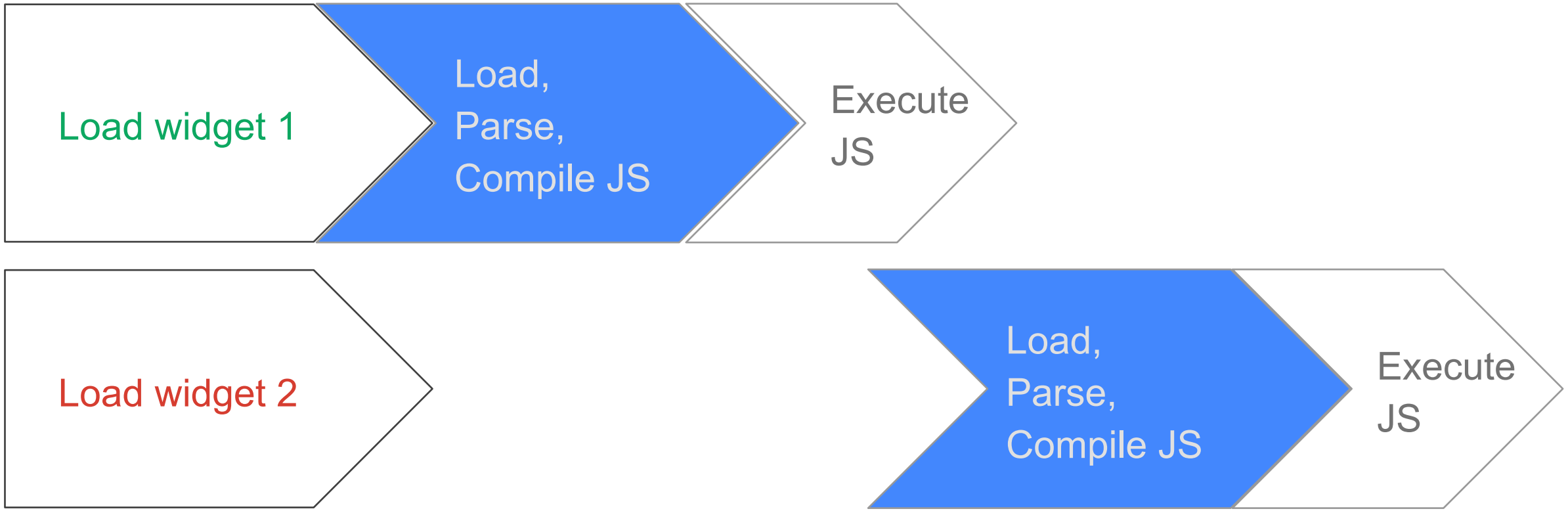
"Code sharing"



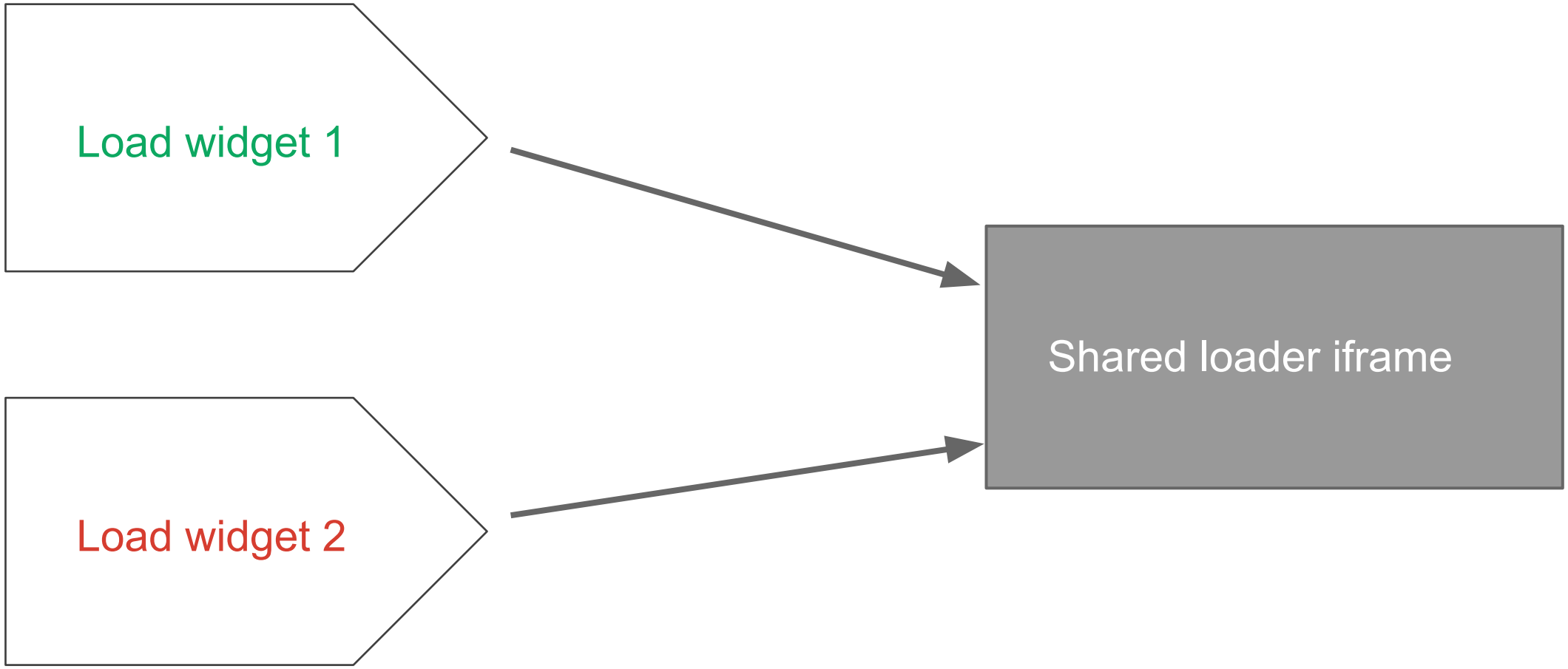


112 iframes – 224 JS requests

Every widget adds cost



Loader iframe



Some random JS

JS

```
var d = 'none';
document.getElementById('foo').style.display = d;
document.querySelector('.bar').style.display = d;
```

Some random JS "windowified"

JS

```
var d = 'none';
window.document.getElementById('foo').style.display = d;
window.document.querySelector('.bar').style.display = d;
```

Some random JS "windowified"

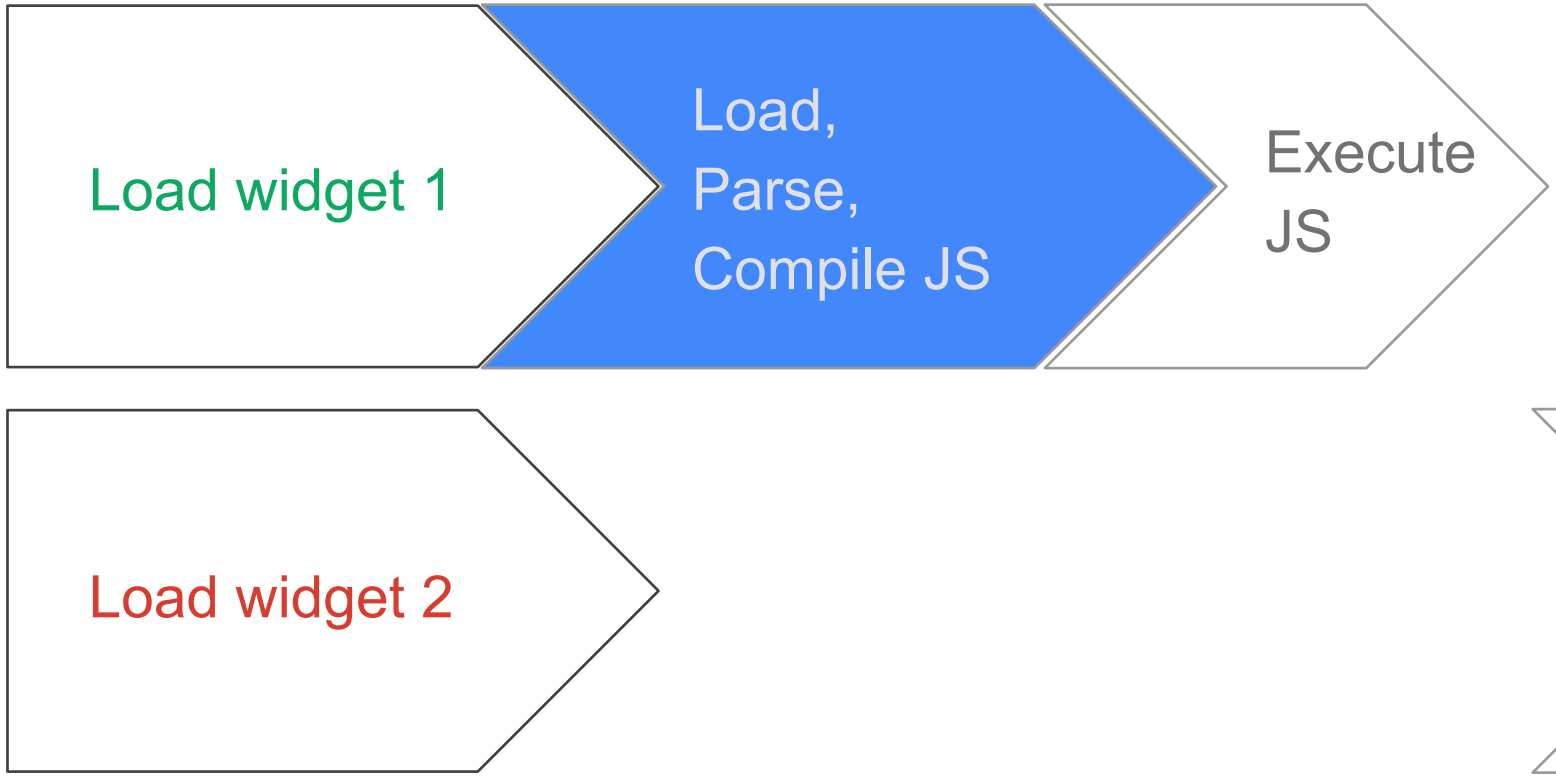
JS

```
function javaScriptCallback(window) {
  var d = 'none';
  window.document.getElementById('foo').style.display = d;
  window.document.querySelector('.bar').style.display = d;
}
```

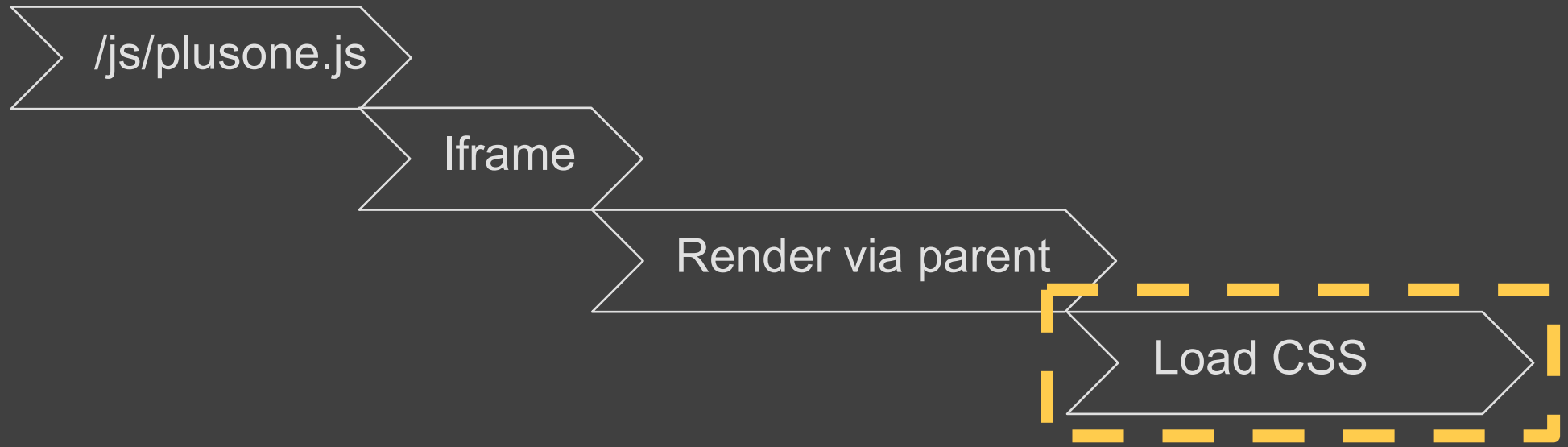
JS

```
javaScriptCallback(iframe1);
javaScriptCallback(iframe2);
```

Loading, parsing and compiling only happens once.



"CSS inlining"



Project: CSS inlining

X Billion Impr. / day

2 KB of CSS

= 2X Terabyte / day

Inlining

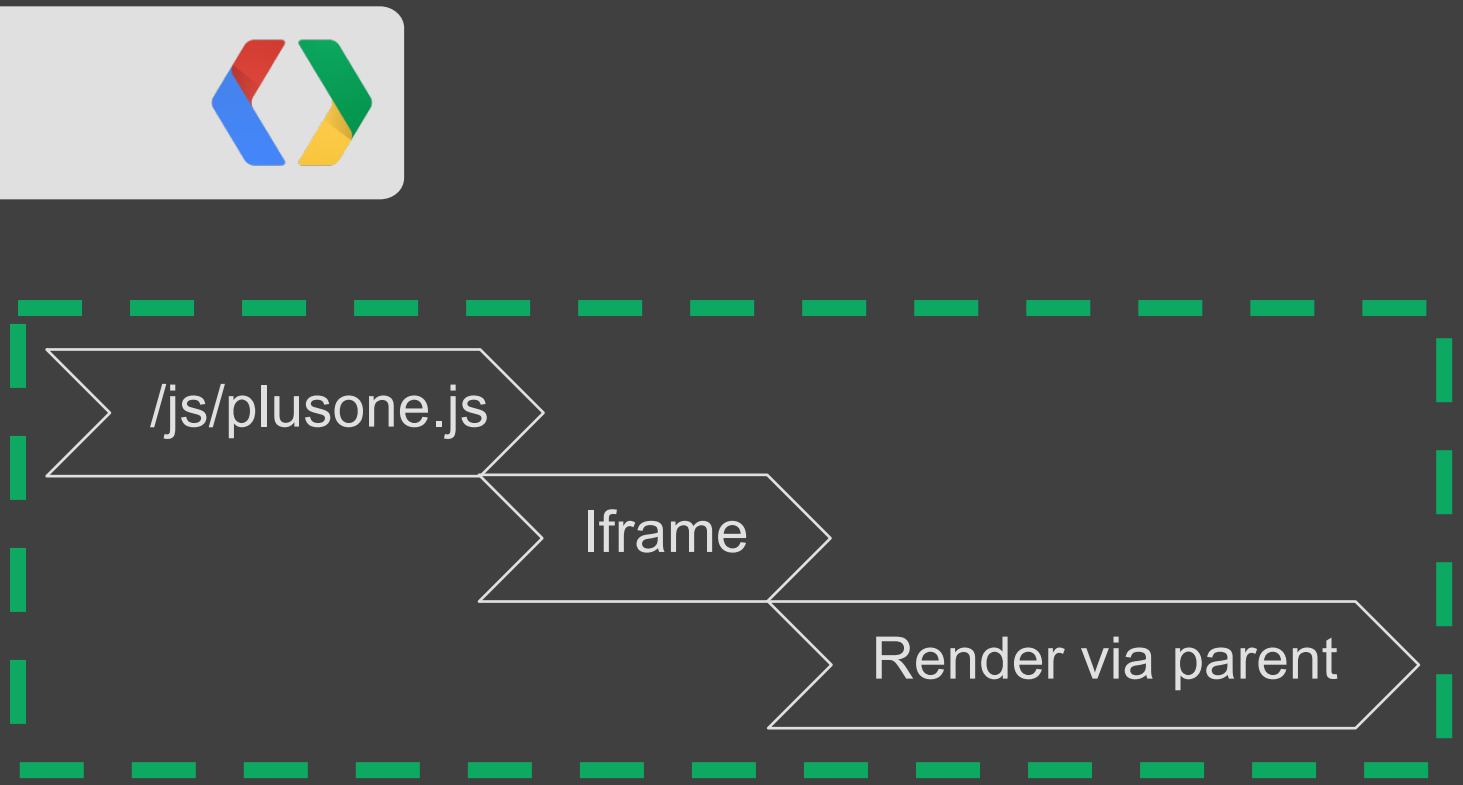
Inlining

Bad for caching
Fewer HTTP requests
Shorter waterfall

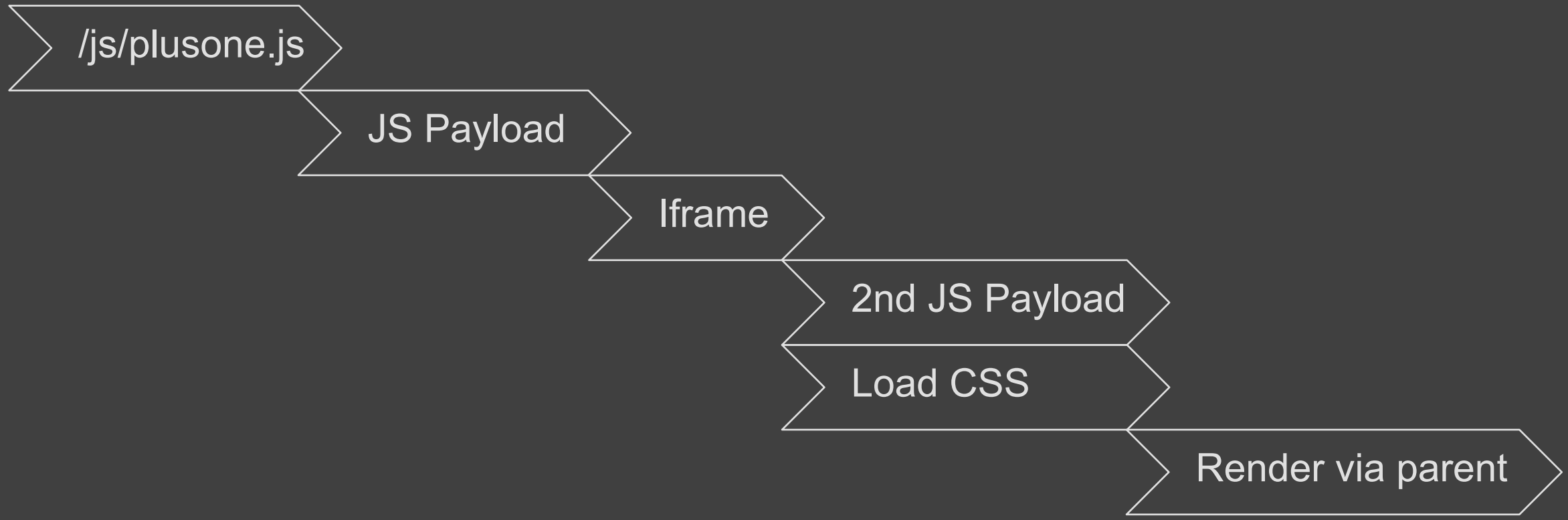
External resources

Awesome caching
Fewer bytes

Project: CSS inlining


/js/plusone.js
Iframe
Render via parent

Waterfall before optimization



Waterfall after optimization

/js/plusone.js

Iframe

Render via parent

Your app / site

Assuming you follow best practices

“Premature optimization is the root of all evil.”

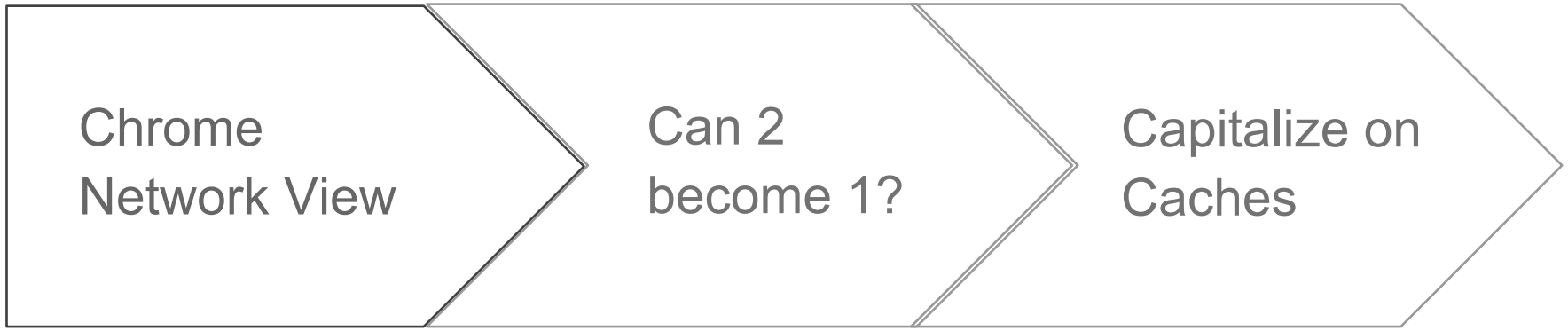
Donald Knuth

Measure! (in Production)

Real World Web Performance Measurement

http://goo.gl/kUYaI
Thursday, 5:15PM - 6:15PM
Room 8

Reduce HTTP requests



Project: CSS Inlining

Project: Widget Detection

Parallelize if possible

Project: Bootstrap rendering

Project: Code sharing

Optimize for perception

Project: Ping

Project: Lazy JS evaluation


WINNING


WINNING

Thank You

+John Hjelmstad
+Malte Ubl