Google
Developers

Instant Mobile Websites

Techniques and Best Practices

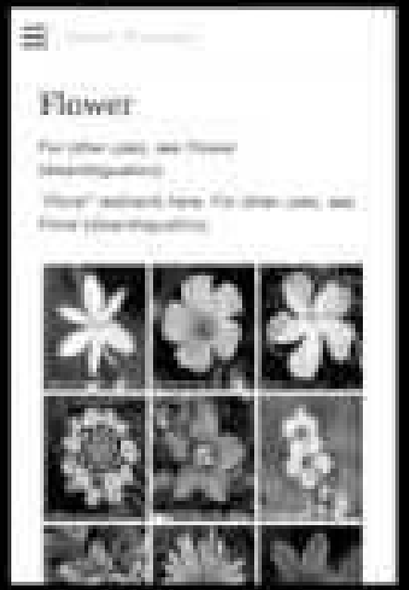
Bryan McQuade, Software Engineer, Google
Doantam Phan, Product Manager, Google
Mona Vajihollahi, Product Manager, Google

Google I/O

Original

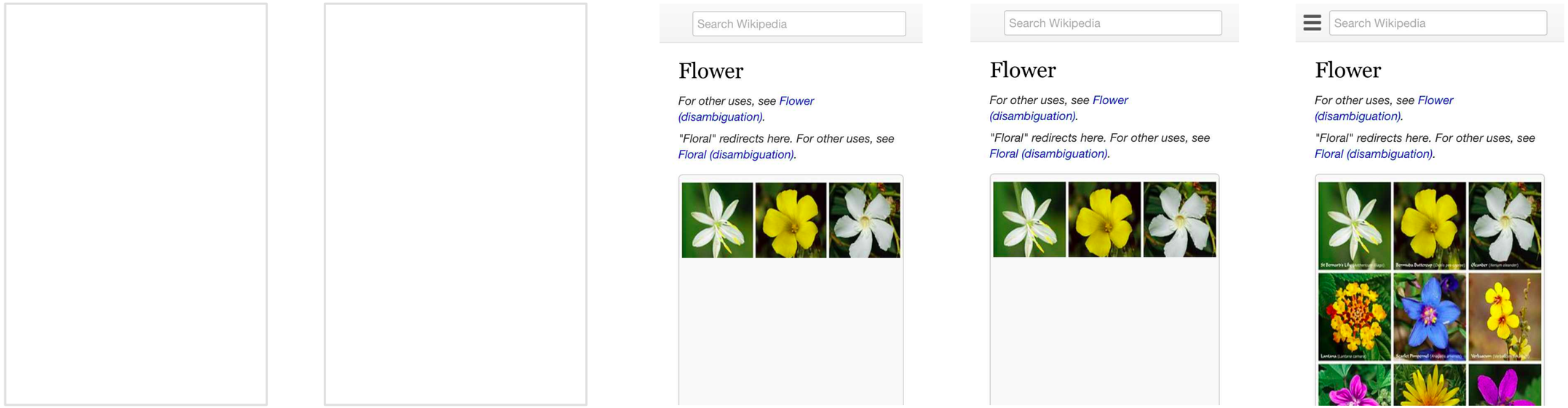
5.0

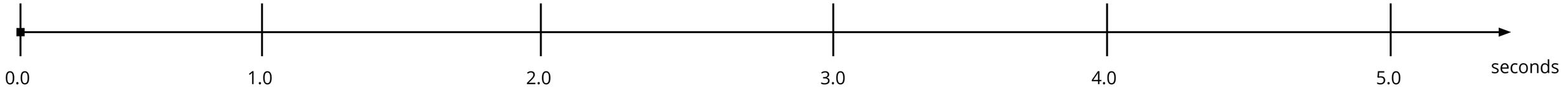
Optimized



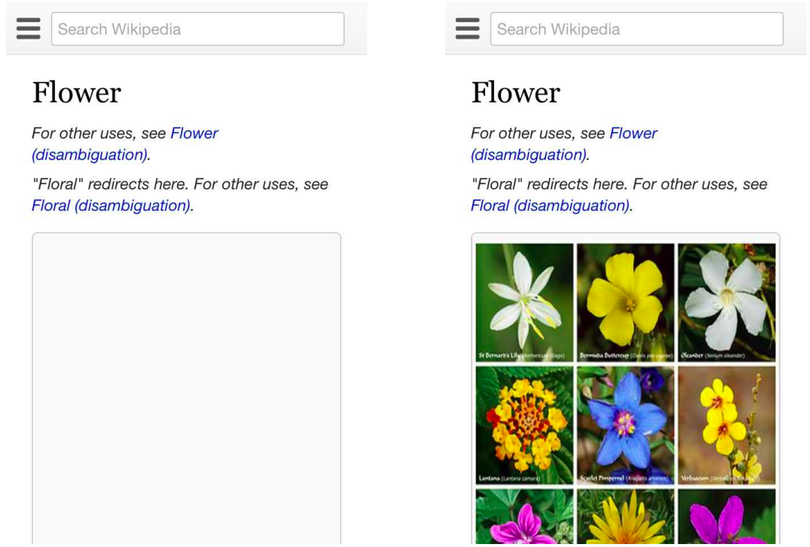
2.0

original





optimized

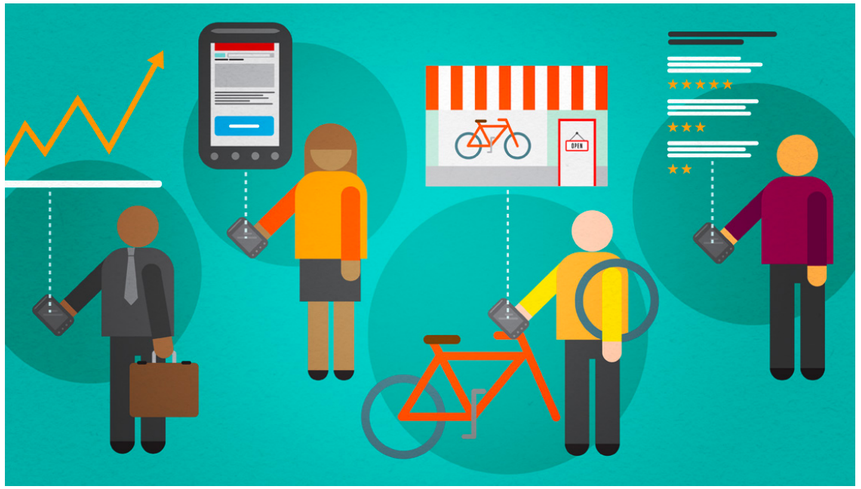


Why speed matters
Instant mobile performance
Deep dive: example optimization

Speed is a critical feature on mobile

Users expect mobile websites to be as fast as desktop

> *"71% of global mobile web users expect websites to load as quickly, almost as quickly or faster on their mobile phone compared to the computer they use at home – up from 58% in 2009."*



What users want from mobile - Gomez

Users will learn to avoid slow sites

First impressions matter!

Experimental impact of 1s additional latency

- 9.4% decrease in page views
- 8.3% increase in bounce rate
- 3.5% drop in conversions
- *"Even after the experiment was over ... shoppers were significantly less likely to return to the site."*

The impact of HTML delay on mobile business metrics - Strangeloop

The mobile experience continues to be slow...

Today's talk: Show the above-the-fold content of your site to a user in under 1 second

1 second limit for a user to not be distracted

But today...

7 seconds average page load time on mobile

Android Device Activations

Is the web getting faster? - Google Analytics Blog

Response Times: The 3 Important Limits - Jakob Nielsen

Our Mobile Planet - Think with Google

Bandwidth is not the bottleneck for page load

- Bandwidth: **amount of data** transferred over the network per unit time, e.g. 5 Mb/s
- Latency: **delay** in the network to transfer a packet, e.g. specified as Round Trip Time
- Latency in loading a web page is dominated by round trips

  More bandwidth doesn't matter (much) - Mike Belshe

- Round trip times are especially high on mobile
  - 3G: 100-450 ms
  - 4G: 60-180 ms

  Internet Services Information - T-Mobile

  Important Coverage Information - Sprint

- We need to design for high latency

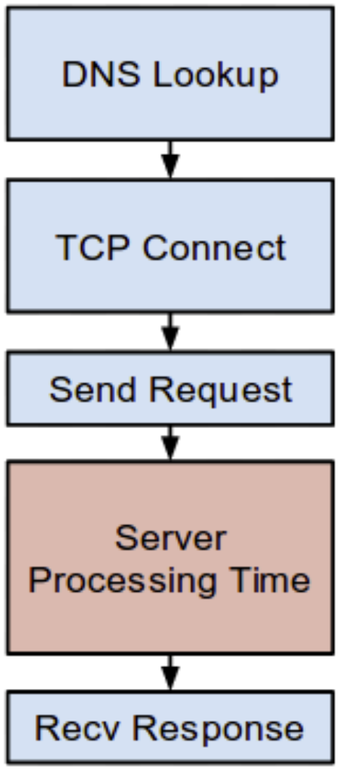
Design for High Latency: The Four Rules

1. Avoid landing page redirects
2. Minimize server processing time
3. Eliminate render blocking resources
4. Prioritize visible content

When a user visits a site...

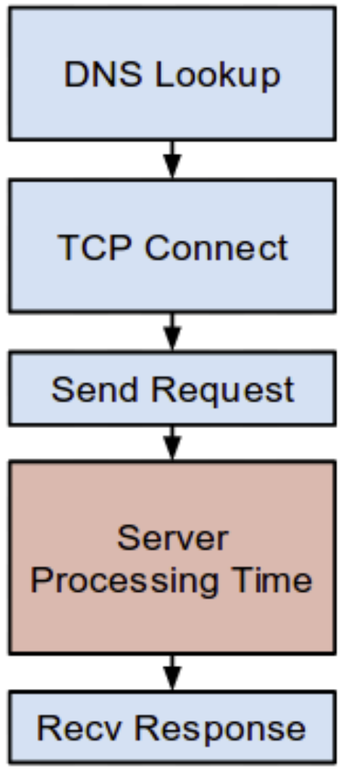
The path to first response byte

- 3 * (Round trip) + Server processing time



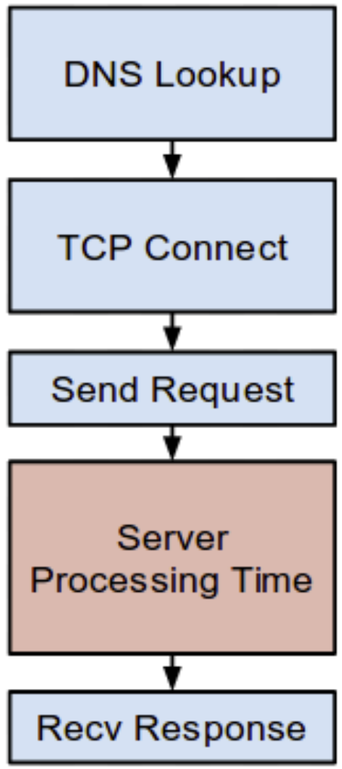
The path to first response byte

- 3 * (Round trip) + Server processing time
- Round trip of 200ms
  600ms + Server processing time



The path to first response byte

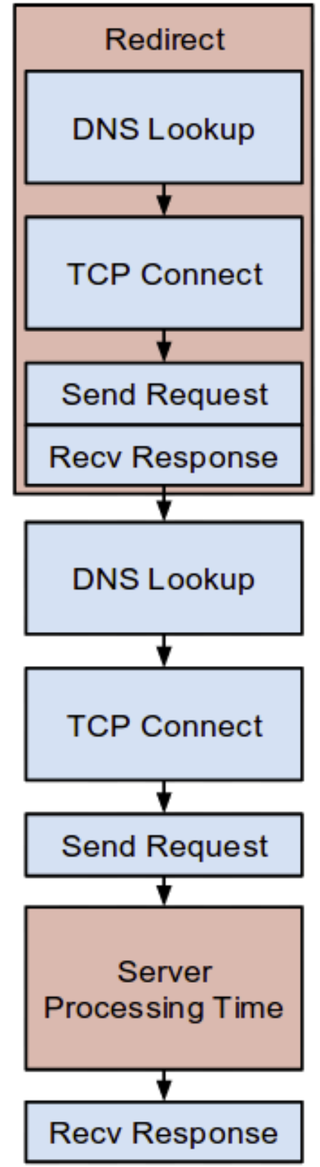
- 3 * (Round trip) + Server processing time
- Round trip of 200ms
  600ms + Server processing time
- What about redirects?
  www.example.com ⇨ m.example.com



The path to first response byte

- 3 * (Round trip) + Server processing time
- Round trip of 200ms
  600ms + Server processing time
- What about redirects?
  www.example.com ⇨ m.example.com

  3 additional round trips (4 over HTTPS)



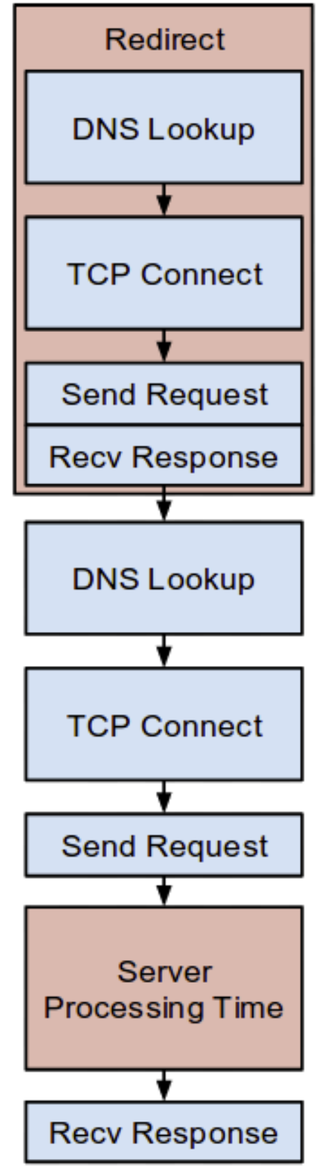
The path to first response byte

- 3 * (Round trip) + Server processing time
- Round trip of 200ms
  600ms + Server processing time
- What about redirects?
  www.example.com ⇨ m.example.com

  3 additional round trips (4 over HTTPS)

1.2 seconds total latency

All before the browser has received any of the HTML content



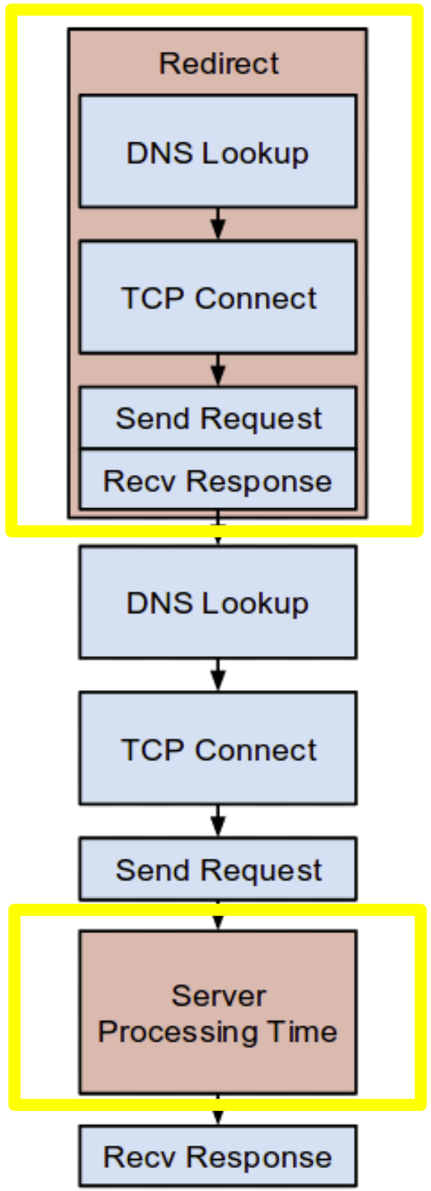
The path to first response byte

- 3 * (Round trip) + Server processing time
- Round trip of 200ms
  600ms + Server processing time
- What about redirects?
  www.example.com ⇨ m.example.com

  3 additional round trips (4 over HTTPS)

1.2 seconds total latency

All before the browser has received any of the HTML content

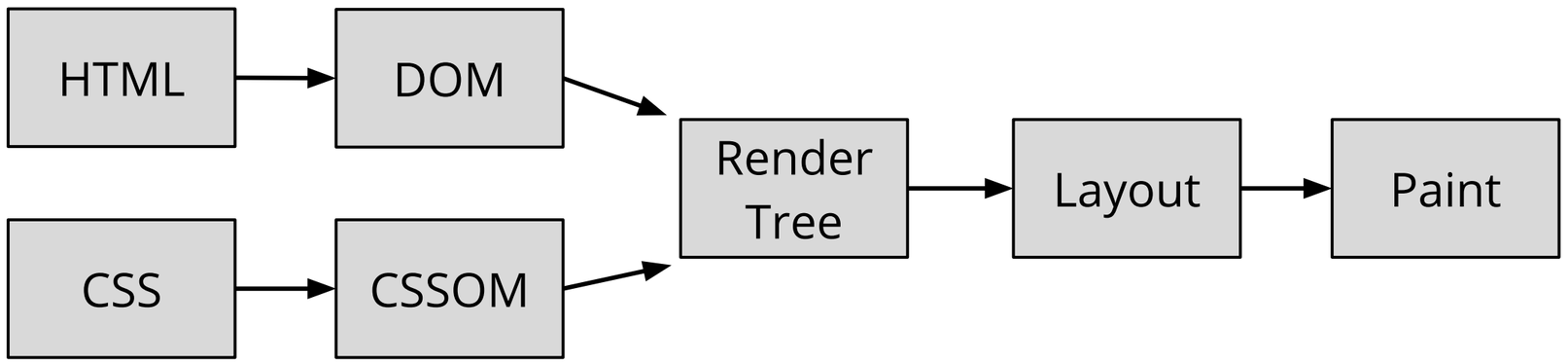


Design for High Latency: The Four Rules

1. Avoid landing page redirects
2. Minimize server processing time
3. Eliminate render blocking resources
4. Prioritize visible content

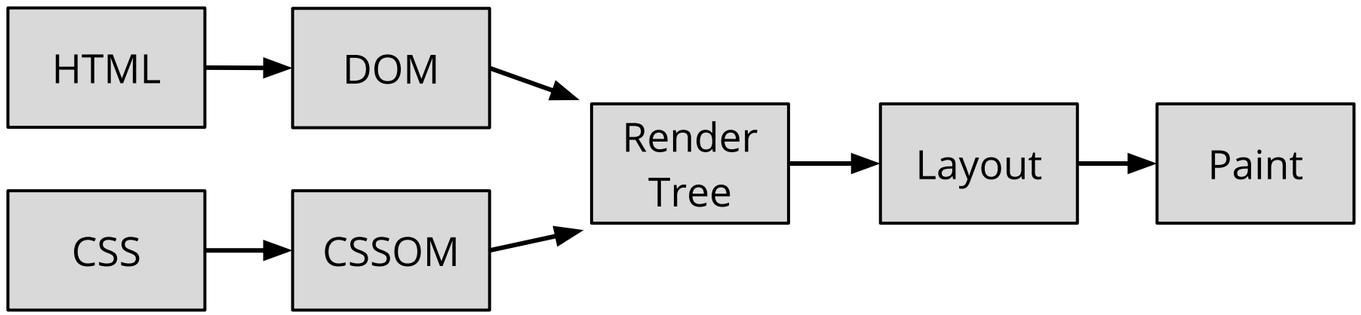
Browser rendering

- DOM tree construction blocks on external scripts
- Render tree construction blocks on external stylesheets



Putting it together: an example web page load

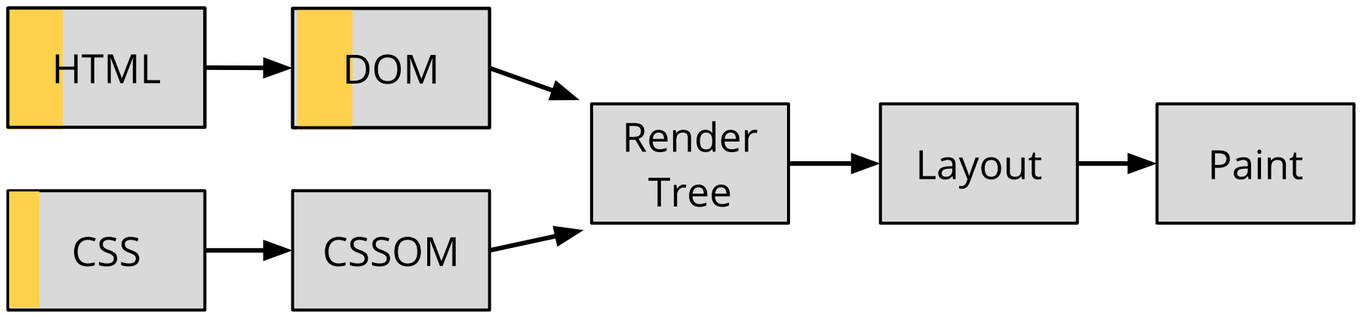
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css		
image.webp		
last.js		

Putting it together: an example web page load

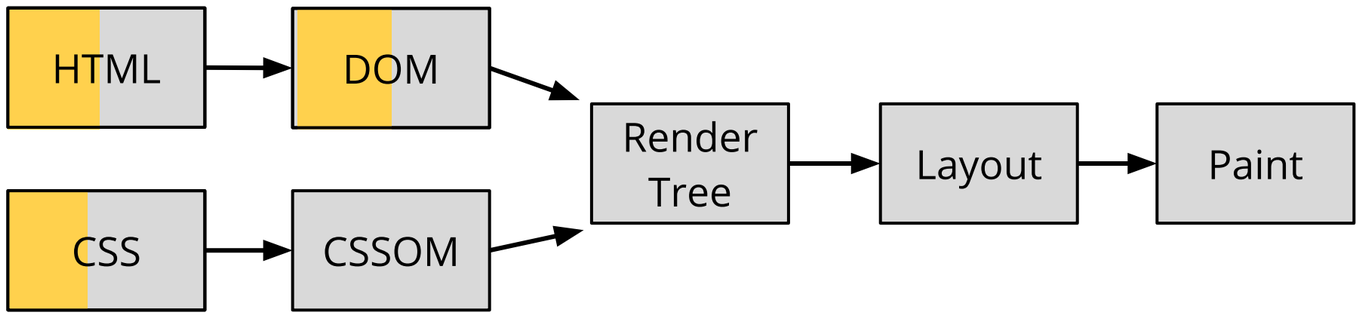
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	
image.webp		
last.js		

Putting it together: an example web page load

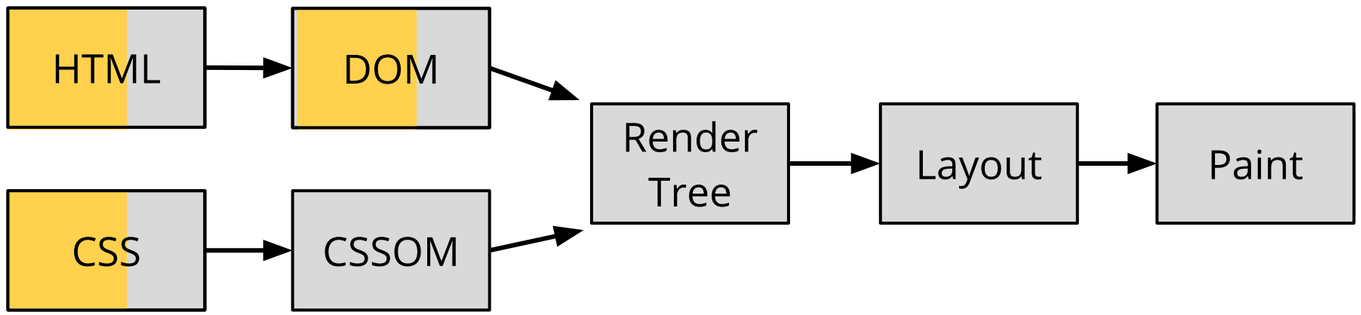
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	
image.webp		
last.js		

Putting it together: an example web page load

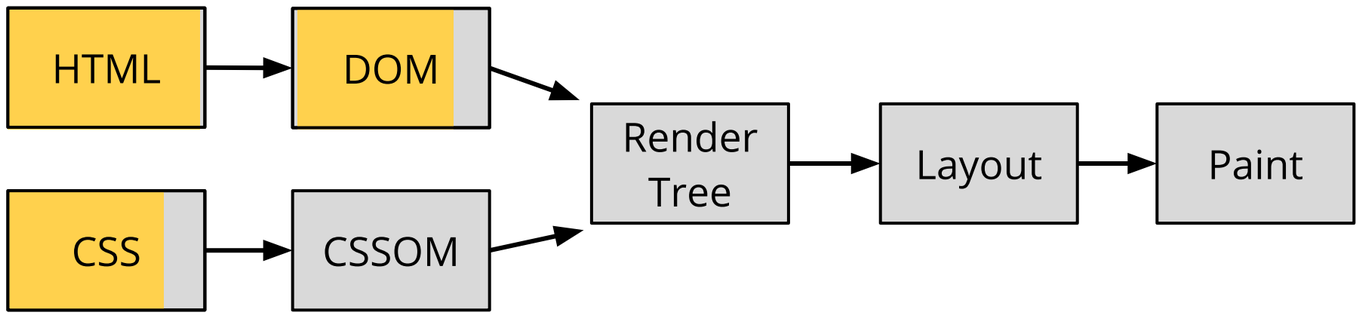
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	
image.webp	✓	
last.js		

Putting it together: an example web page load

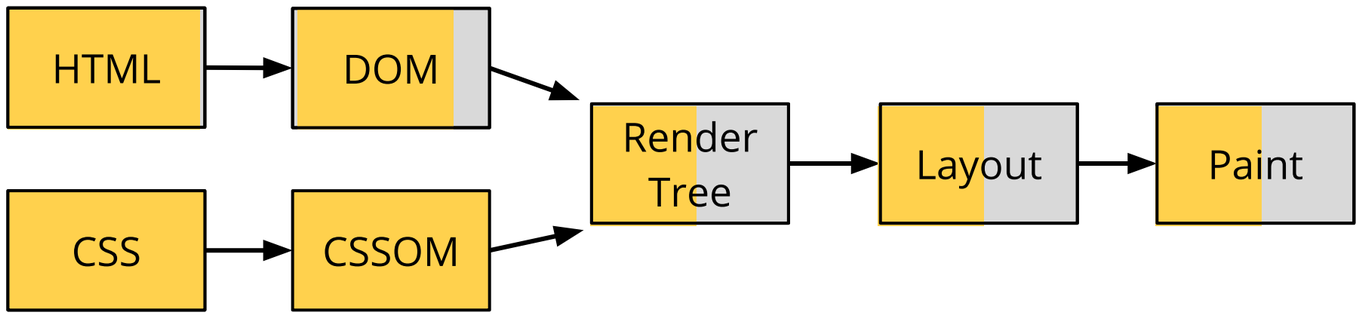
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	
image.webp	✓	
last.js	✓	

Putting it together: an example web page load

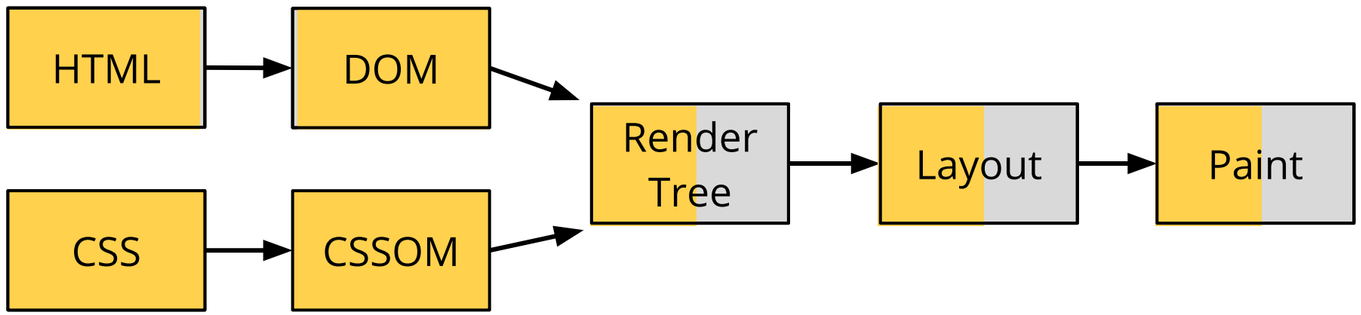
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	✓
image.webp	✓	
last.js	✓	

Putting it together: an example web page load

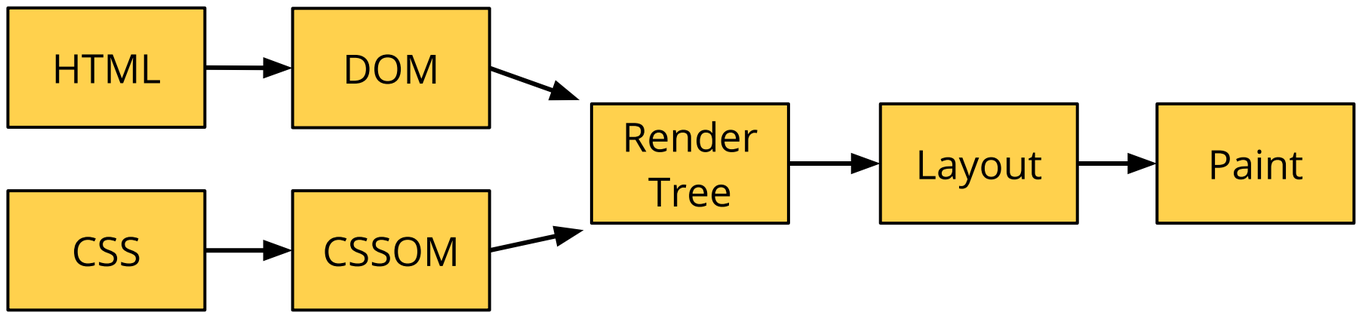
```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	✓
image.webp	✓	
last.js	✓	✓

Putting it together: an example web page load

```
<html><head>
<link rel="stylesheet" href="example.css">
</head><body>
<div>Hi there!</div>
<img src="image.webp">
<script src="last.js">
</script></body></html>
```



resource	discovered?	loaded?
example.css	✓	✓
image.webp	✓	✓
last.js	✓	✓

Mobile rendering performance

- Scripts and stylesheets in the <head> block painting of content in the <body>
- Blocking is costly on mobile
  - 2-4 round-trips = 400-800ms on 3G/4G
- Avoid blocking external scripts and stylesheets to achieve near-instant rendering performance on mobile

Identify and inline critical CSS

HTML

```
<html><head>
<style> ... critical parts of example.css ...  </style>
</head><body>
<div>Hi there!</div>
<img src="example.webp">
<script src="last.js"></script>
```

Design for High Latency: The Four Rules

1. Avoid landing page redirects
2. Minimize server processing time
3. Eliminate render blocking resources
4. Prioritize visible content

Example: additional latency due to TCP slow start

HTML

```
<html><head>
<style>
.square { width:100px; height:100px; background-size:contain; background-repeat:no-repeat; }
.icon-pagespeed { background-image:url('... }
.icon-chrome { background-image:url('... }
.icon-android { background-image:url('... }
...
</style>
</head><body>...
```

Example: additional latency due to TCP slow start

HTML

```
<html><head>
<style>
.square { width:100px; height:100px; background-size:contain; background-repeat:no-repeat; }
.icon-pagespeed { background-image:url('... }
.icon-chrome { background-image:url('data:image/p     ⇦ 14kB cutoff still in <head>
```

Only 14kB of the HTML can be transferred before a new round trip.

Large inline data URIs prevent the visible content from fitting in the first 14kB.

Keep inline scripts and styles in the head to a minimum.

Delay load large data URIs.

Initial Payload Size and Round Trip Time

TCP Slow Start: discover the capacity of the connection

- Start by sending a small amount of data (~14kB)
- If that data is acknowledged, send 2x data during the next round trip

Send the critical data needed for above the fold render in the first 14kB, to avoid additional round trips due to slow start.

With compression, ~45kB of text.

Design for High Latency: The Four Rules

1. Avoid landing page redirects
2. Minimize server processing time
3. Eliminate render blocking resources
4. Prioritize visible content

Deep dive: our example mobile site

- Redirects to m-dot site
- 1 second server processing time
- Small HTML payload
- One stylesheet with images as data URIs

https://demo.modspdy.com/*

* Demo has nothing to do with SPDY. Just a domain we had available. :)

Deep dive: our example mobile site

- Redirects to m-dot site
- 1 second server processing time
- Small HTML payload
- One stylesheet with images as data URIs

https://demo.modspdy.com/

Pretty simple page. It should load quickly, right?...

Instant Mobile Demo

This is the initial, unoptimized version of our simple mobile page. The optimized version is here.

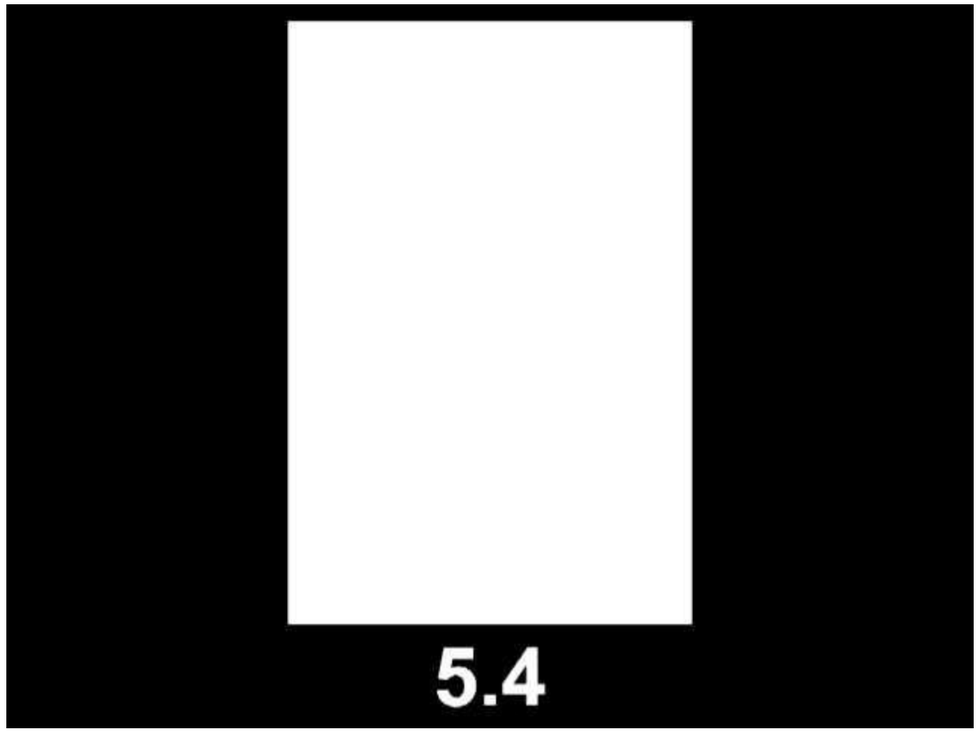
PageSpeed

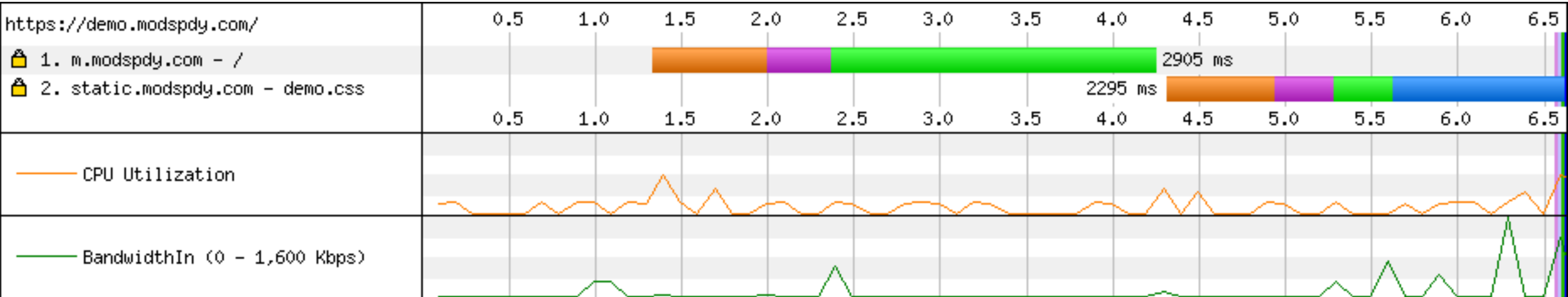
The PageSpeed family of tools is designed to help you optimize the performance of your website. PageSpeed Insights products will help you identify performance best practices that can be applied to your site, and PageSpeed optimization tools can help you automate the process.

Chrome

Google Chrome is a browser that combines a minimal design with sophisticated technology

Unoptimized page: 6.6 seconds





Avoid landing page redirects

Problem

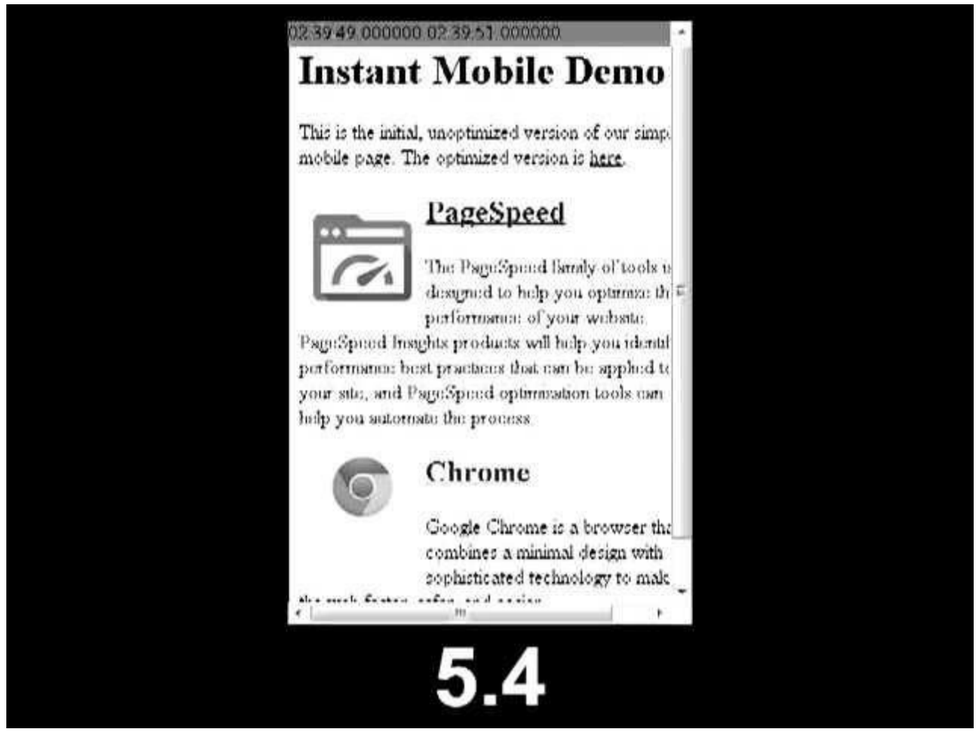
Redirect from demo.modspdy.com ⇨ m.modspdy.com

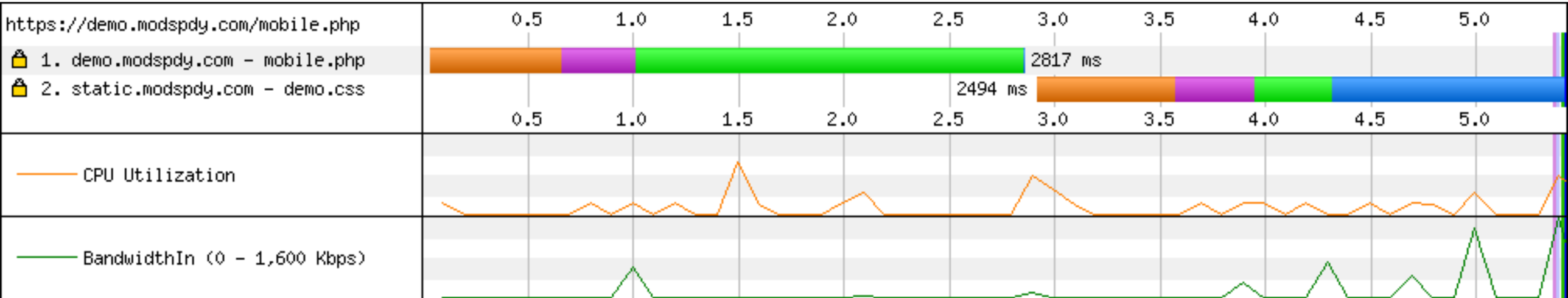
3 additional round trips (4 over HTTPS)

Solution

- Use responsive design, or
- Vary content based on user agent

Removed redirect: 5.4 seconds





Minimize server processing time

Problem

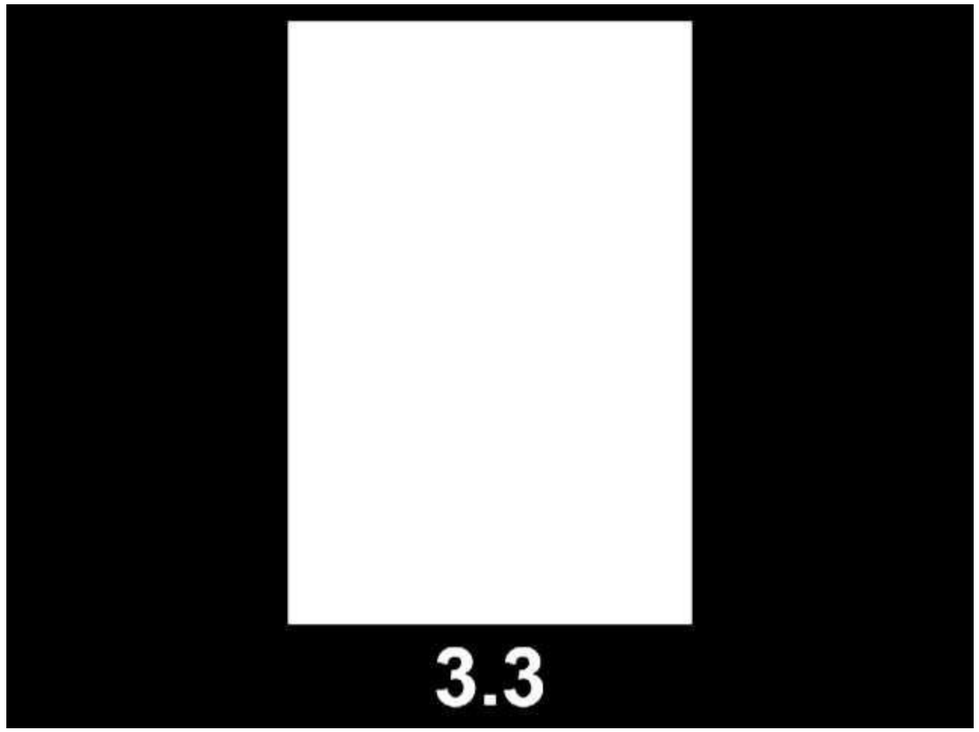
Server processing time blocks sending of the response to the client

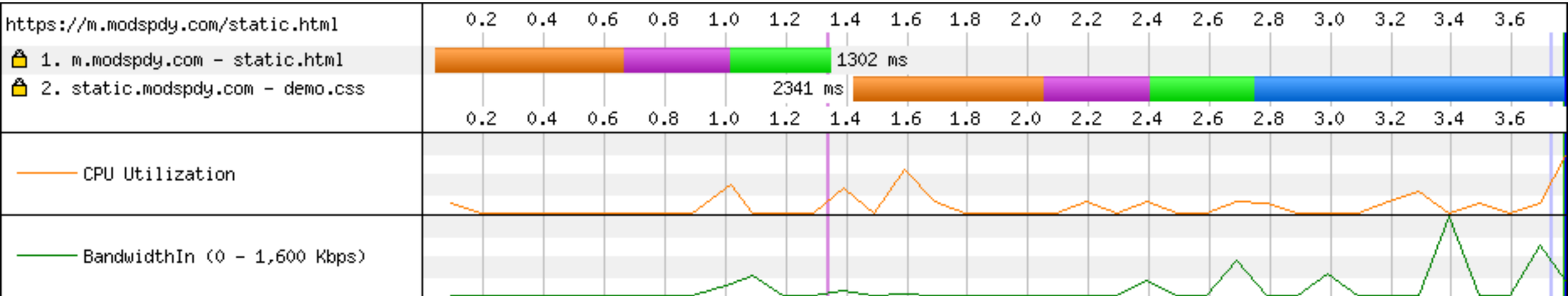
1+ seconds additional render blocking delay

Solution

- Reduce server response time to 200ms
- Monitor server activity using e.g. NewRelic
- Identify expensive operations, e.g. database calls, HTTP fetches, and remove, defer, or optimize

Removed server processing time: 3.8 seconds





Eliminate render blocking resources

Problem

The load of static.modspdy.com/demo.css blocks rendering of all page content.

7 additional render blocking round trips

DNS+TCP+TLS+request/response, 3 more to download file

Solution

- Inline critical JS and CSS in the <head>
- Make sure above the fold content renders with minimal JavaScript. Ideally none!

Initial, unoptimized HTML and CSS

HTML

```
<html><head>
<link rel="stylesheet" href="//static.modspdy.com/demo.css">
</head>
<body>...
```

static.modspdy.com/demo.css

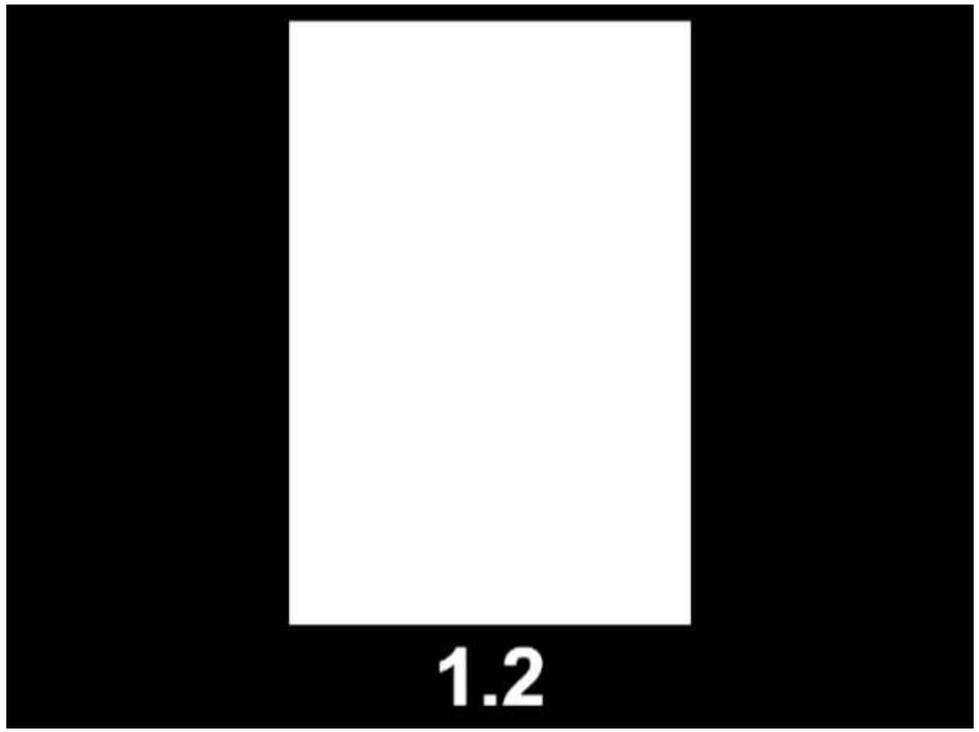
```
.square { width:100px; height:100px; background-size:contain; background-repeat:no-repeat; }
.icon-pagespeed { background-image:url('... }
.icon-chrome { background-image:url('... }
...
```

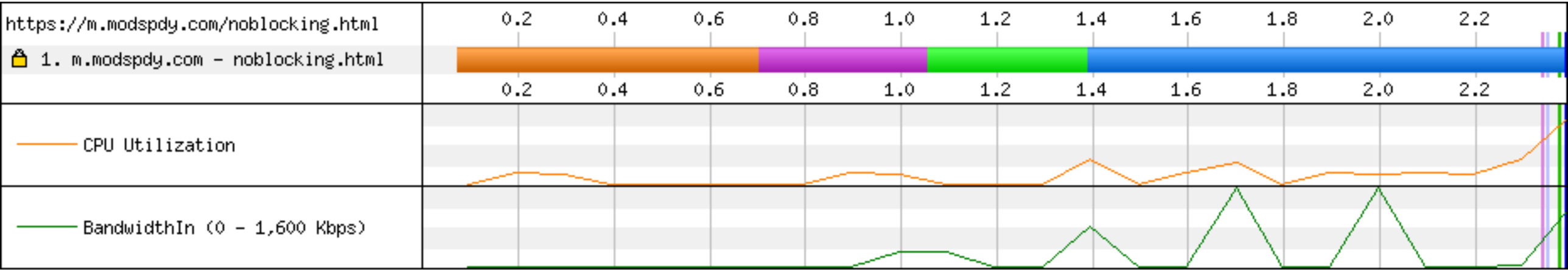
CSS inlined in HTML

HTML

```
<html><head>
<style>
.square { width:100px; height:100px; background-size:contain; background-repeat:no-repeat; }
.icon-pagespeed { background-image:url('... }
.icon-chrome { background-image:url('... }
...
</style>
</head>
<body>...
```

Inlined CSS: 2.4 seconds





Prioritize visible content

Problem

Inlining large CSS incurs round trips due to TCP congestion window growth

Solution

- Inline only critical CSS in HTML
- Delay load remaining CSS, e.g. large data URIs

Additional optimization

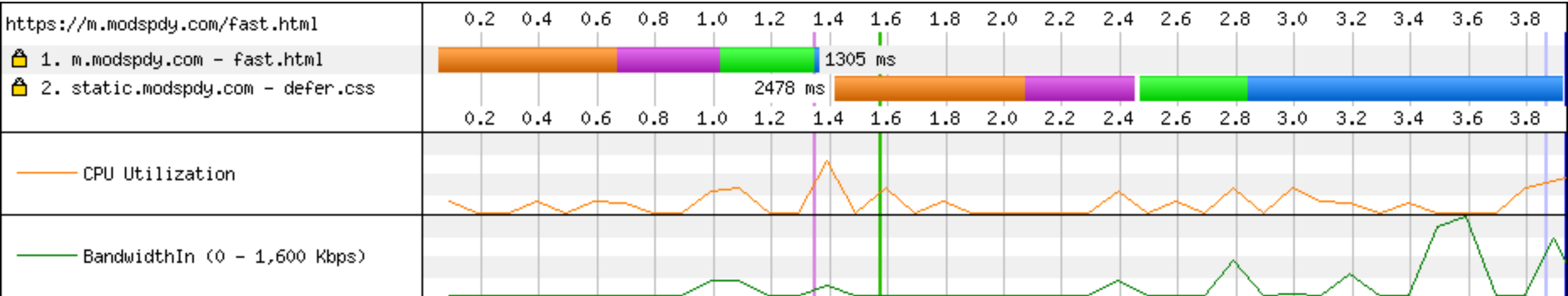
- Inline low resolution image previews after above the fold content, if space available.

Optimized HTML and CSS

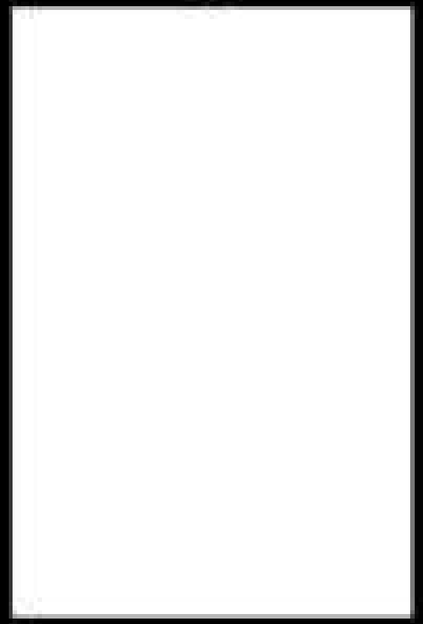
HTML

```
<html><head>
<style>
.square { width:100px; height:100px; background-size:contain; background-repeat:no-repeat; }
</style>
<script>
function delayLoadCss() { var l = document.createElement('link'); l.rel='stylesheet'; l.
href='//static.modspdy.com/defer.css'; document.head.appendChild(l); }
var raf = window.requestAnimationFrame || window.mozRequestAnimationFrame || ...;
if (raf) { raf(delayLoadCss); } else { window.setTimeout(delayLoadCss, 500); }
</script>
</head><body>...
```

Deferred non-critical CSS: 1.5 seconds



Original



4.9

Optimized

3.9

Design for High Latency: The Four Rules

1. Avoid landing page redirects
   - Use responsive design, or vary content based on user agent
2. Minimize server processing time
   - Keep server time under 200ms
   - Use monitoring such as NewRelic to understand high backend times
3. Eliminate render blocking round trips
   - Inline critical CSS in the <head>
   - Make sure above the fold content renders with minimal JavaScript. Ideally none!
   - Delay loading of CSS and JavaScript not needed for initial render
4. Prioritize visible content
   - Make sure mobile above the fold content, including critical CSS, fits in first 14kB of response.

<Thank You!>

(Come see us at office hours!)

Bryan McQuade - Software Engineer, Google
Doantam Phan - Product Manager, Google
Mona Vajihollahi - Product Manager, Google

Google
Developers