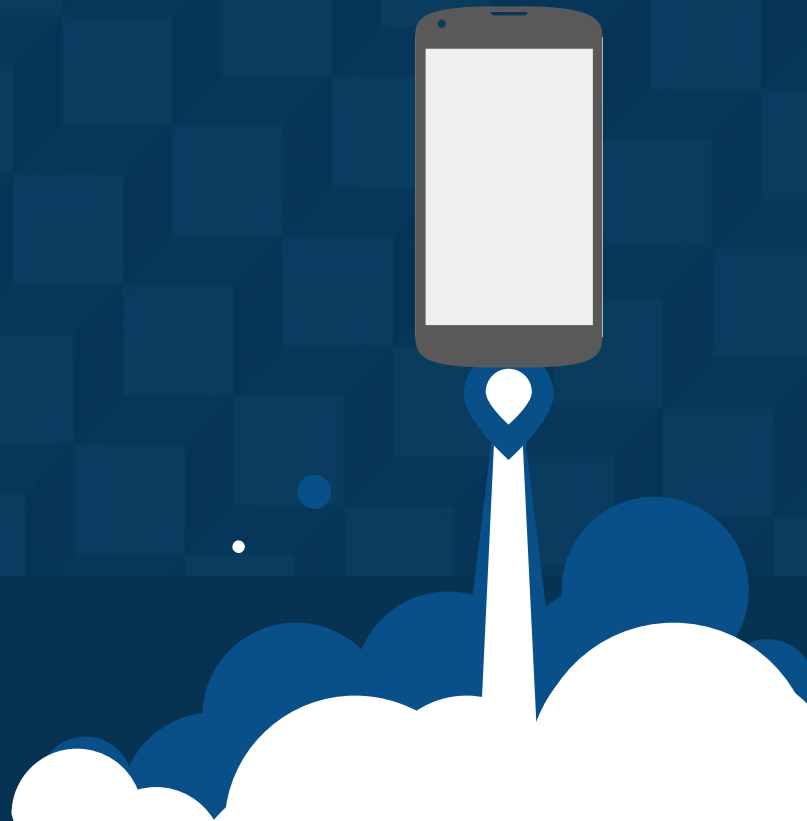
Mobile Site Speed Hackathon

NEED FOR SPEED

1. Connect to Wi-Fi: Mobile Academy
   Password: speedup17
2. Join Googlespeedhackathon.slack.com or sign up on goo.gl/v7Fx9F

Nov 7, 2017



Speed Hackathon : say what?

win prizes!
And eternal glory...

1

Introduction

Why speed matters

2

Presentation with interaction

We go through usual problem areas, you check, ask questions

3

Hack away!

Pick one page to focus on – solve the usual problem areas!

4

Who decreased load time the most?

Winners and Best Buddy is chosen!

Measurement
Attribution
Search
Video
Conversion Rate Optimization
Analytics

The specialist team

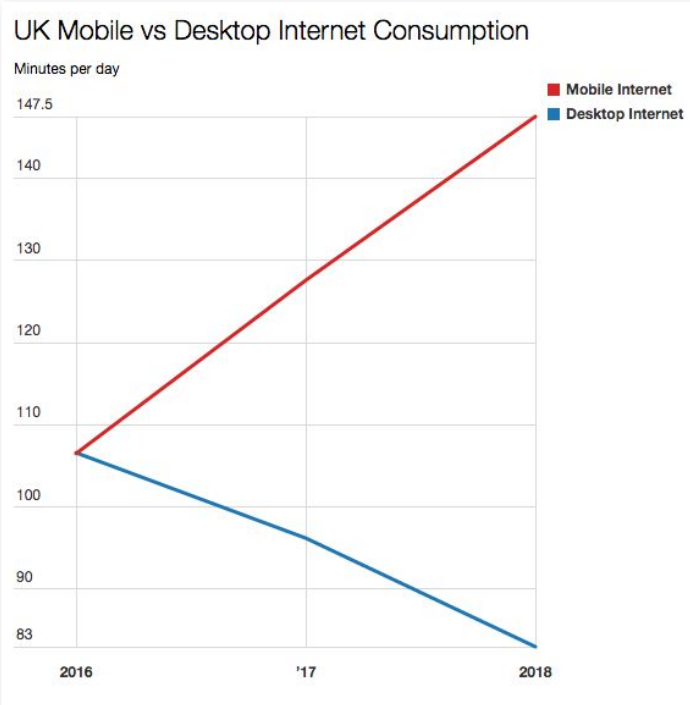
Discover **problems** (slash opportunities!)

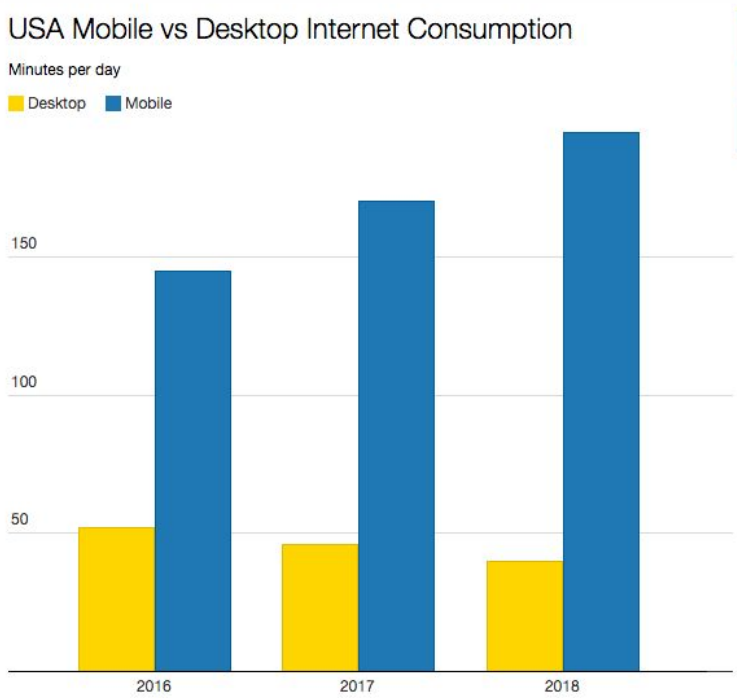
Help priority clients **fix** the problems

Fixed problems give higher **results**!


Problem / opportunity
FRT division
Share of market activity
Distribution of the securities market key players

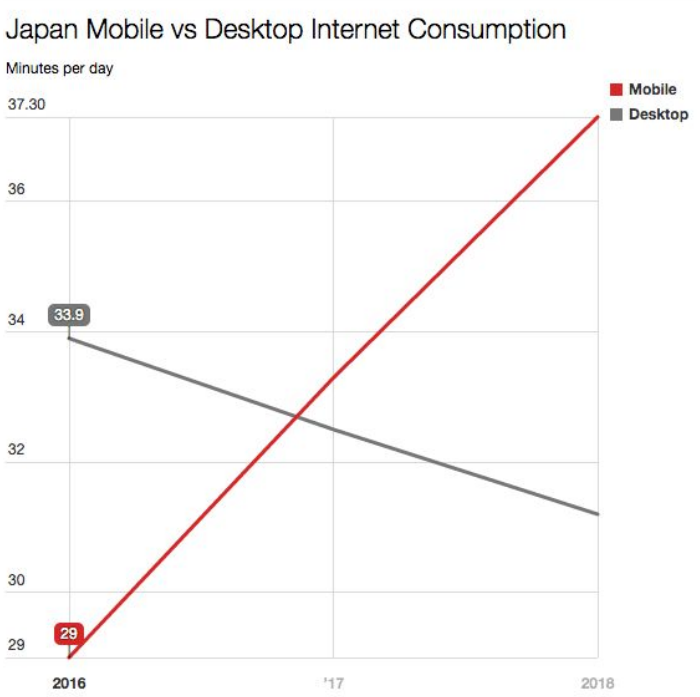
Time spent on mobile vs desktop – Digiday

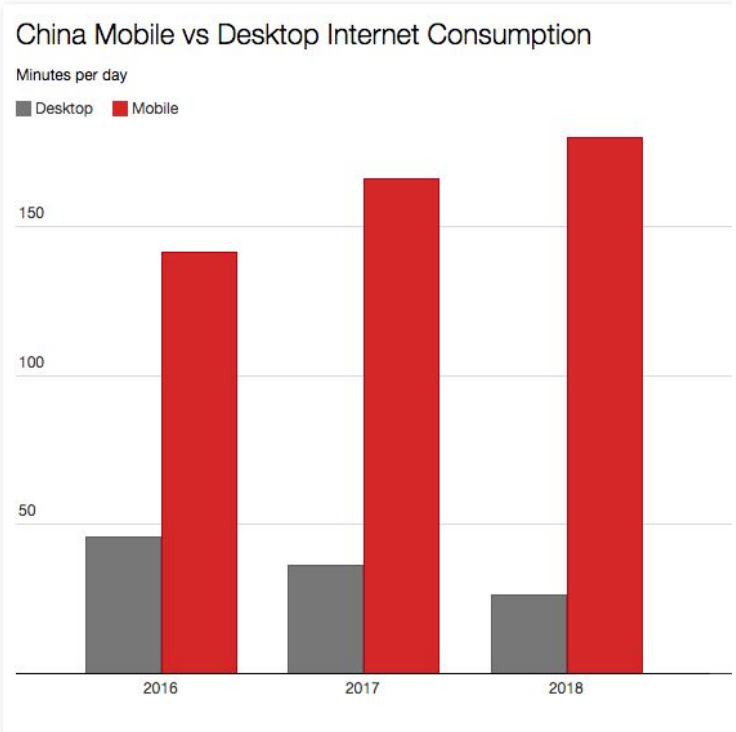




Source: How mobile is overtaking desktop for global media consumption, Digiday

Time spent on mobile vs desktop – Digiday



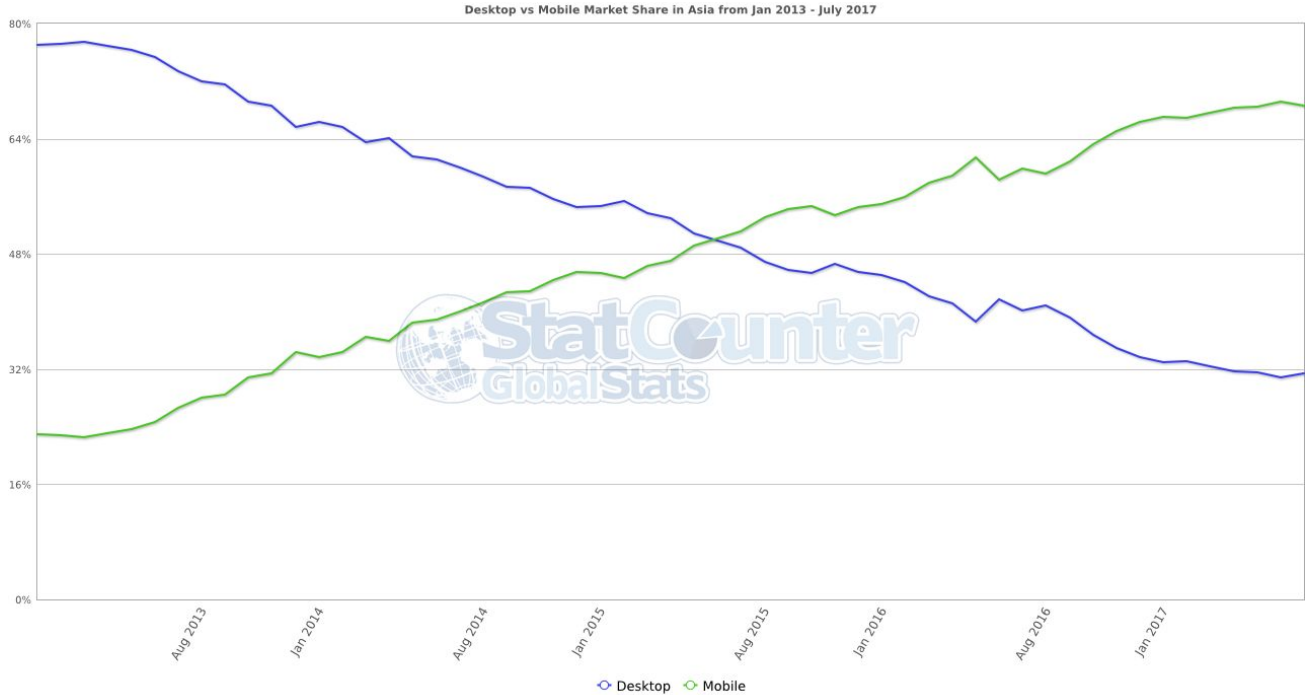


Source: How mobile is overtaking desktop for global media consumption, Digiday

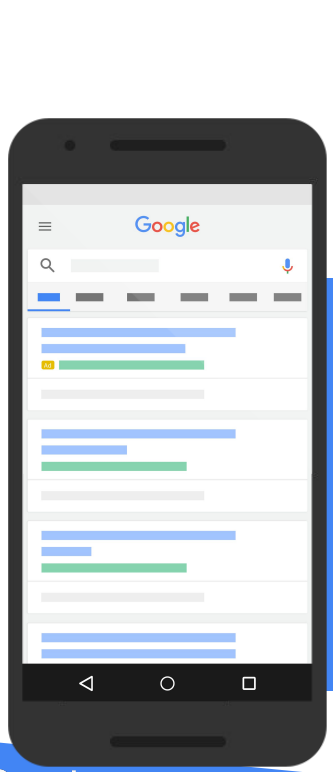
Market share mobile vs desktop in Asia - StatCounter

Desktop

Mobil

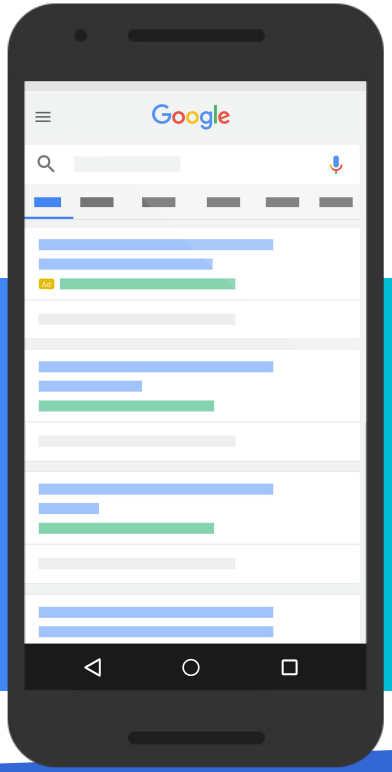


Source: StatCounter



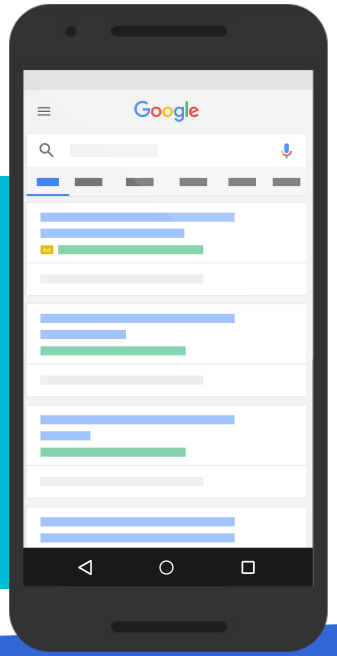
2015

Google started getting more search queries from mobile than desktop

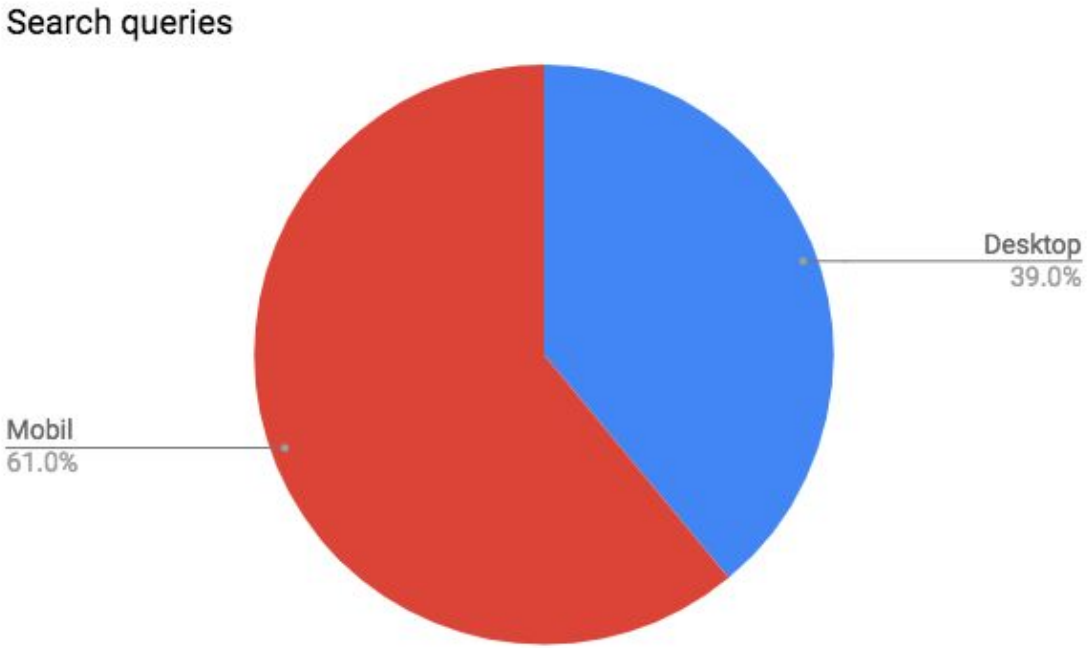


2017

By Black Friday, revenues from mobile will pass desktop in the US (Mobify)



Search queries on mobile vs desktop – Denmark



Bad at mobile
= big problems

Good at mobile
= great possibilities

Master mobile

with...

1

Speed

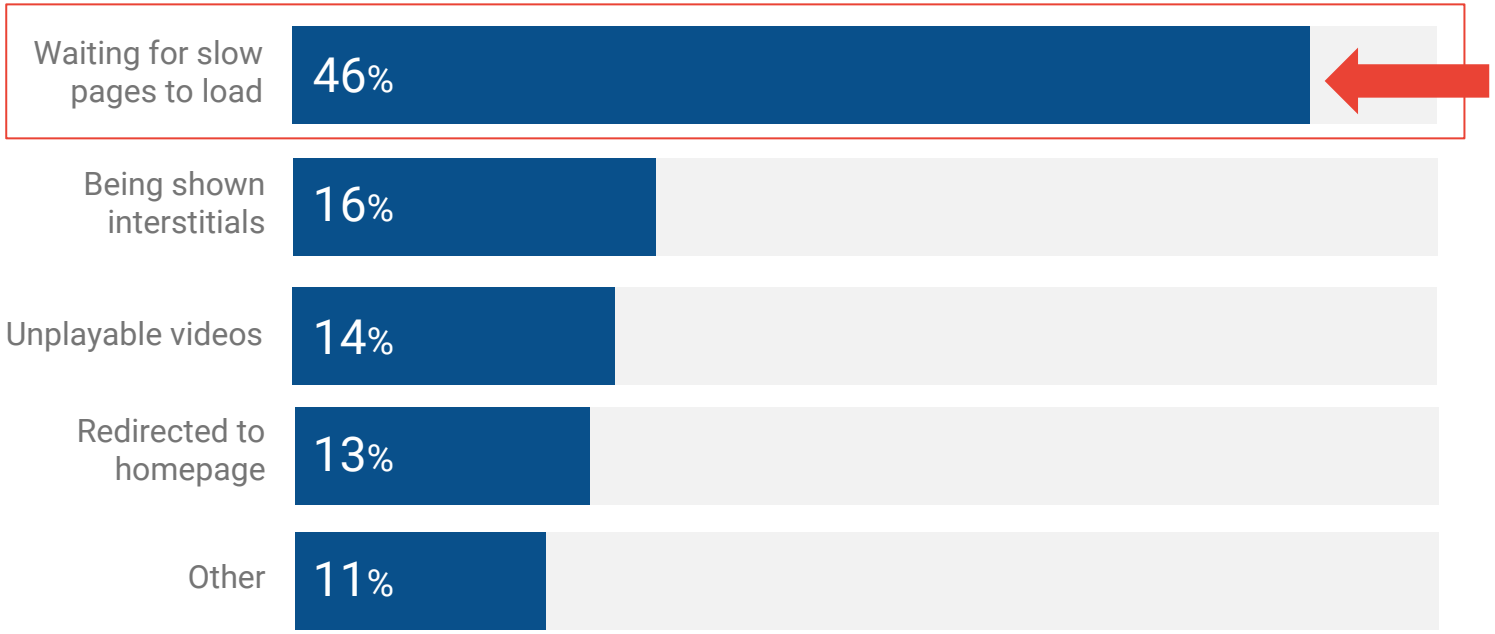
we need to help
our developers

2

A different design

CRO will help us

“What do you dislike the most when browsing the web on your mobile device? ”



=

+38%

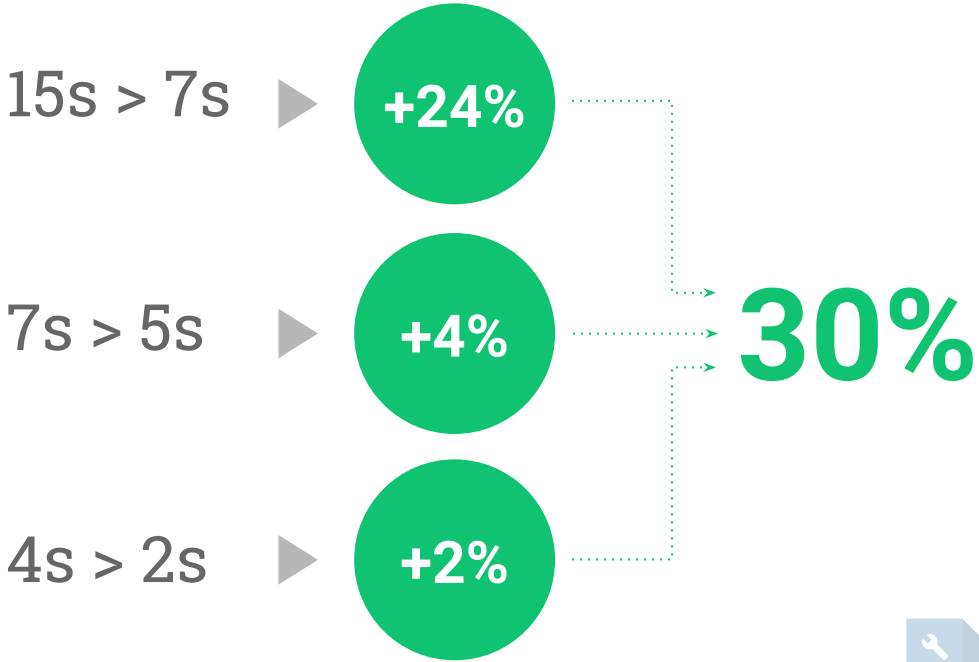
53%

Abandon a site that loads in more than

3 seconds

Get fast – get conversions

(The money issue)



3 seconds!

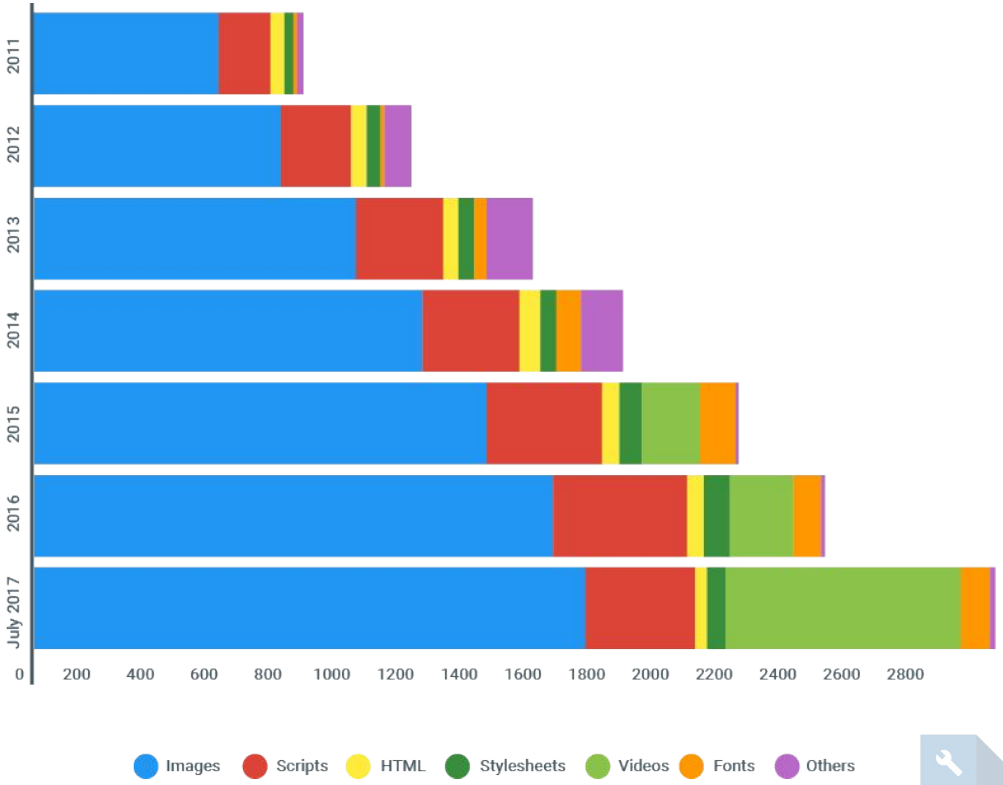
Ha! My site isn’t that slow!

The average web page weight has now reached

3 MB

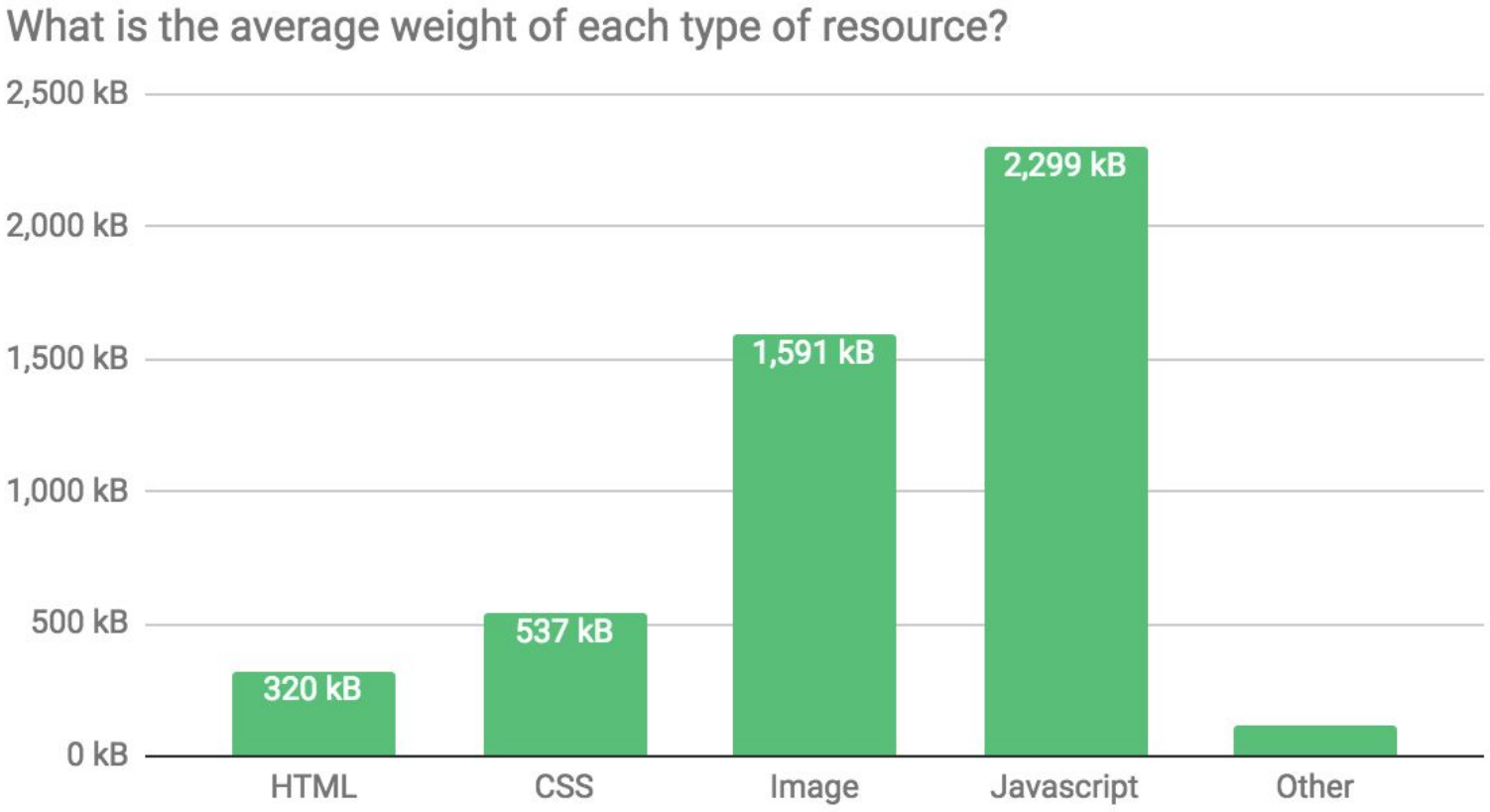
Best practice:

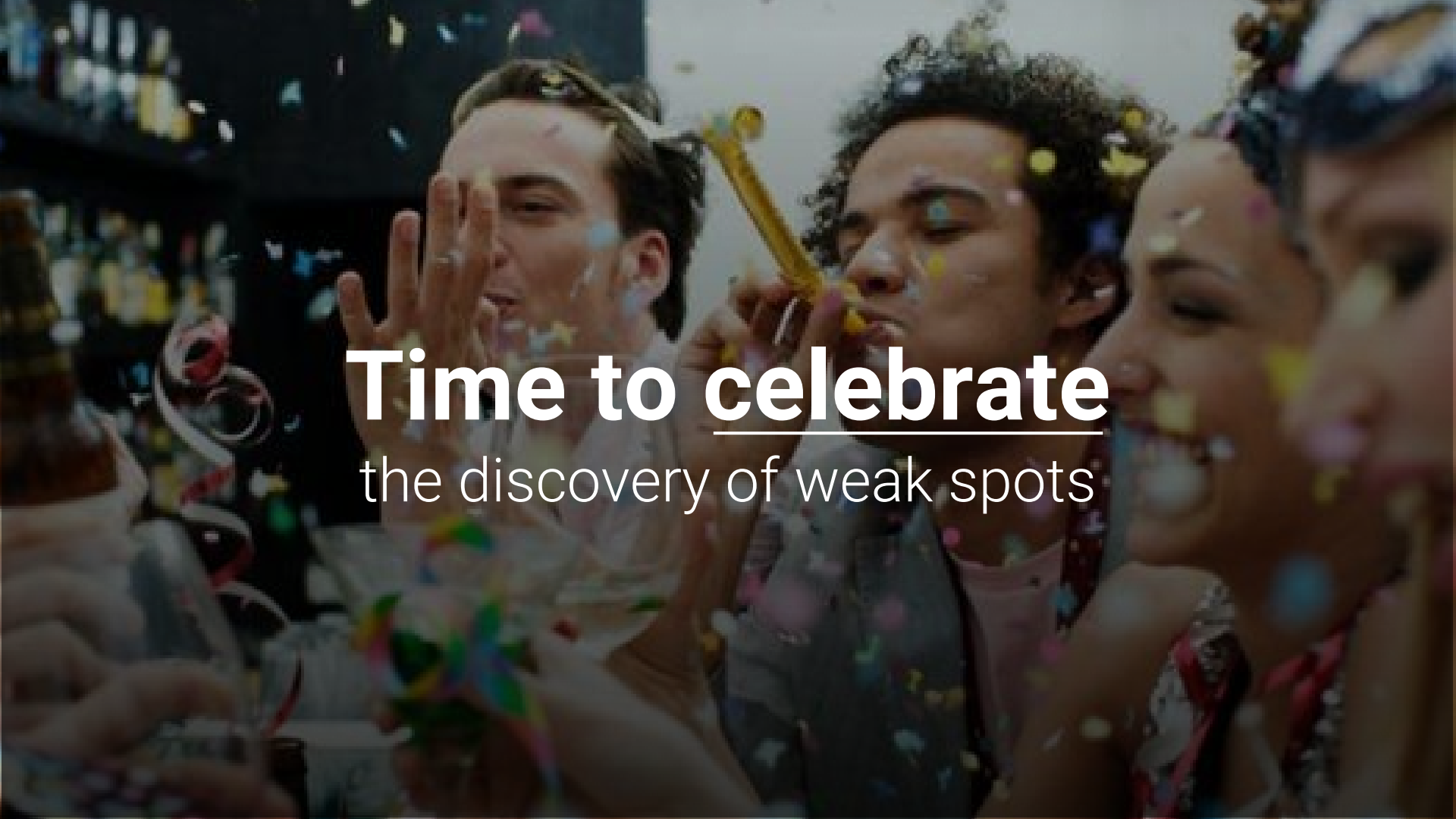
1 MB



Source: The average web page is 3MB. How much should we care? By Tammy Everts, Aug 9 2017

In this room: **Average site weight ~ 4Mb**




Time to celebrate
the discovery of weak spots

Today we'll fix problems!

Why

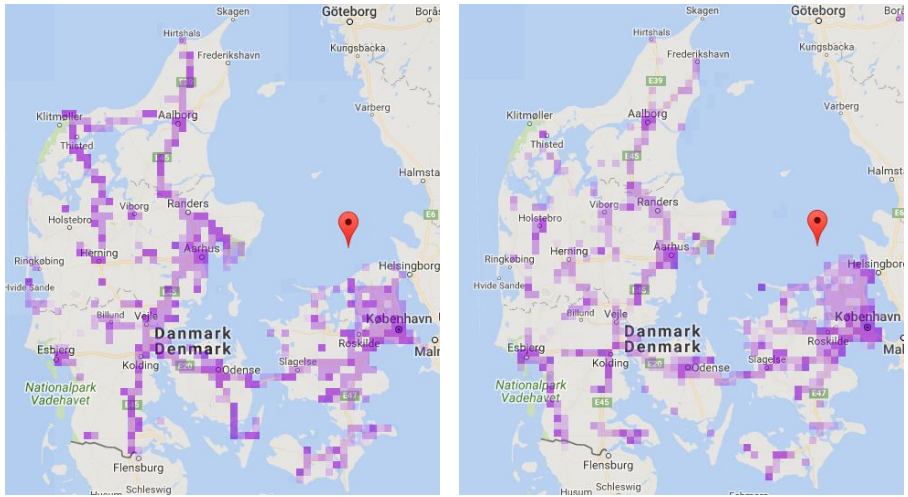
do we measure on 3G Fast?

Google

70% of cellular network connections globally will occur at 3G or slower speeds through 2020

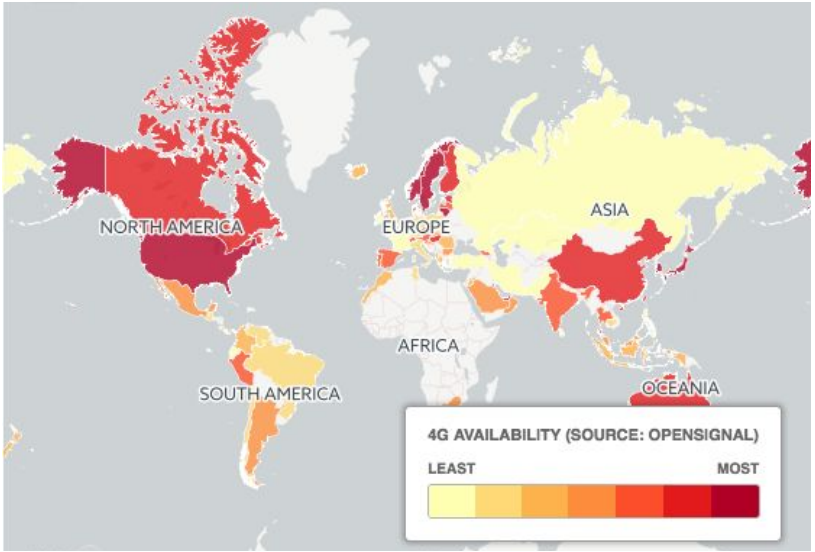
Source

Central Copenhagen is **not** the world – build for slower networks



4G coverage in Denmark
Source: Sensorly

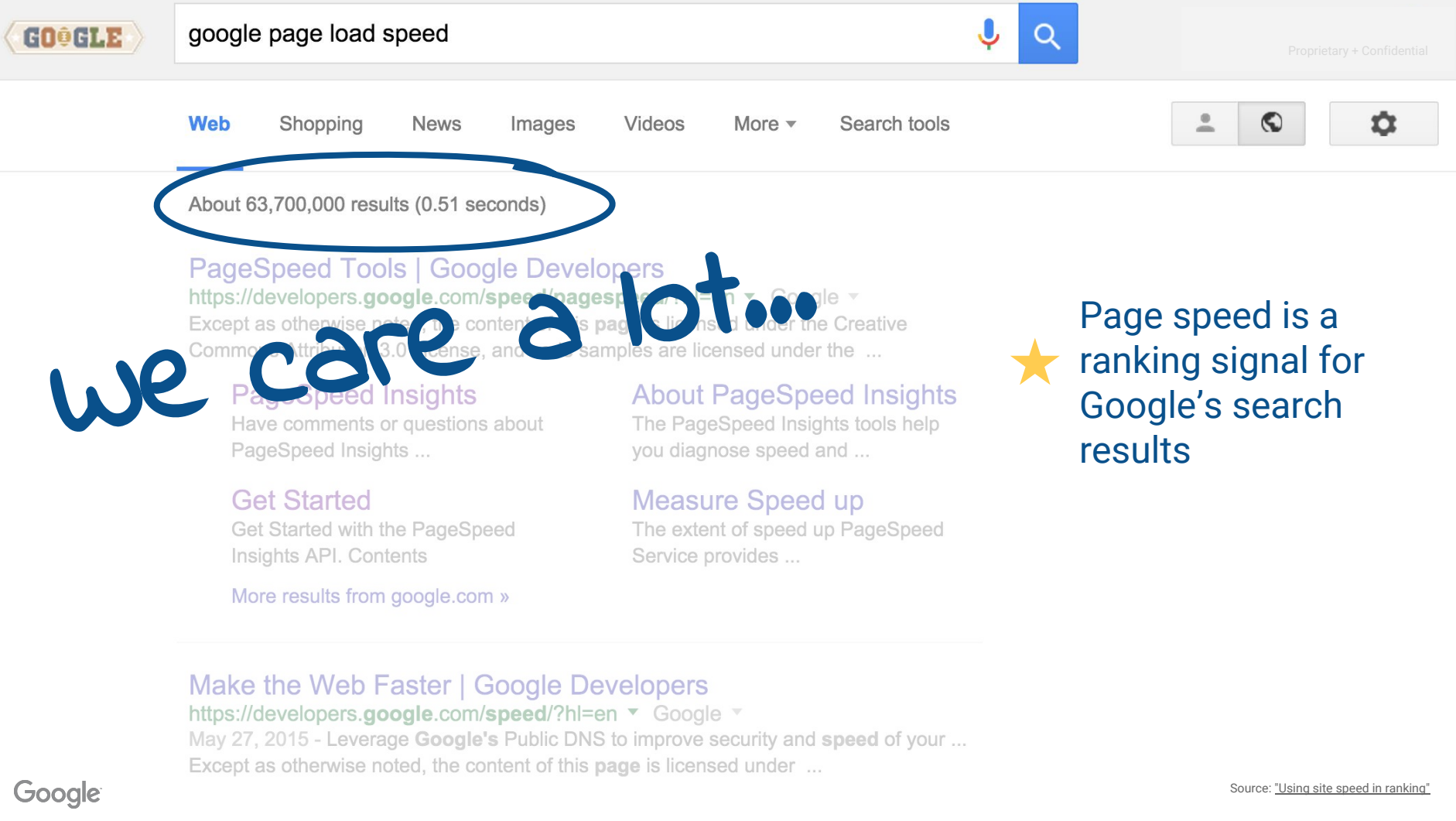
Central Copenhagen is **not** the world
– build for slower networks



4G coverage in the world
(Dec-2016)
(source: OpenSignal)

Want to be great?
(not only in Copenhagen)

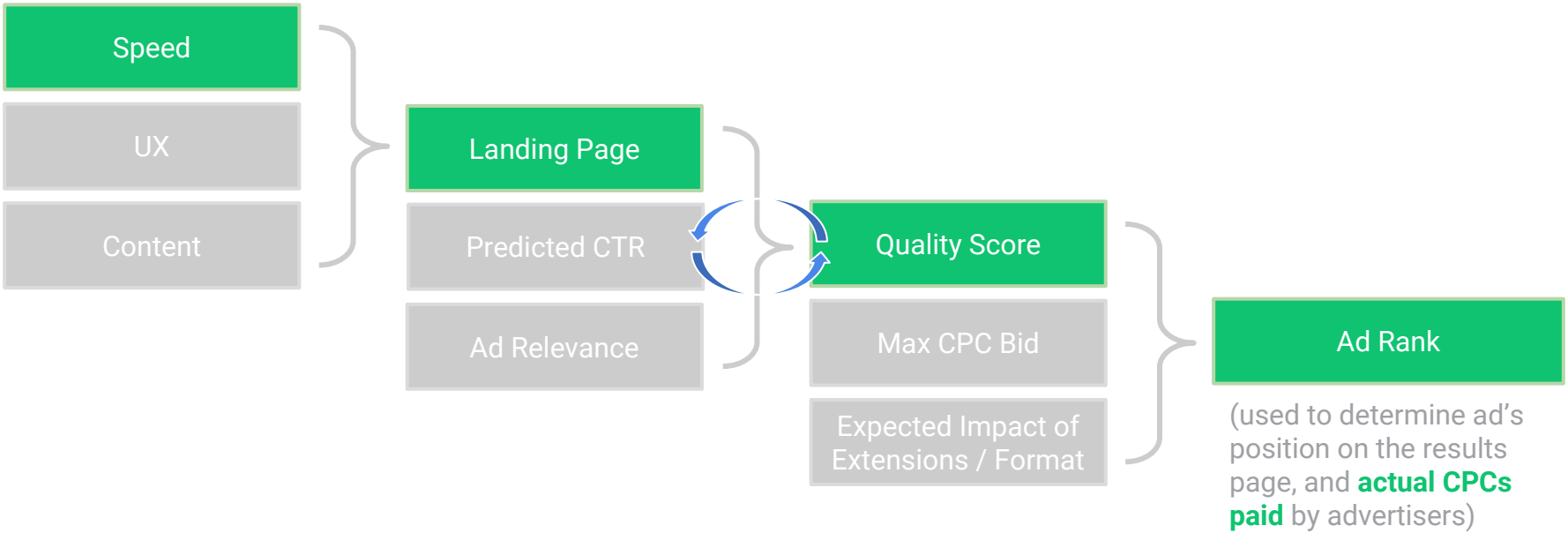
Be great at speed!



★ Page speed is a ranking signal for Google's search results

Source: "Using site speed in ranking"

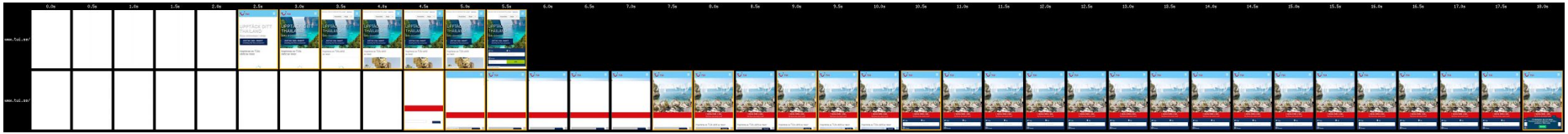
The speed of your mobile site directly influences ad rank!



No worries...

It can be fixed!

The winner in Sweden – TUI



Load time: -55%
Speed Index: -60%
Start render: -50%

More wins! **Speed = Business Impact**

Google

2% *slower*

↓

2%

fewer searches/user

YAHOO!

400ms *faster*

↓

9%

more traffic

Aol.

Faster pages

↓

more page views

amazon.com

100ms *faster*

↓

1%

more revenue

shopzilla

5s *faster*

↓

25%

more pageviews, 7-12% more revenue

Fathead

37% *faster*

↓

70%

increase in mobile revenue per user

GQ

80% *faster*

↓

108%

increase in ads interaction rate

Now it’s your turn

Let’s start fixing problems!

How to get a great Hackathon:

Ask / Share

googlespeedhackathon.slack.com
Channel: #speedupdk

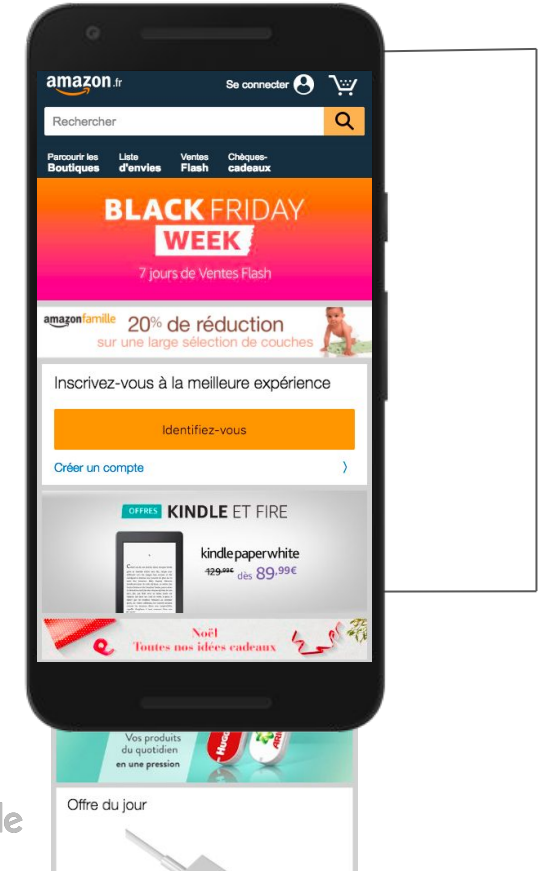
Measuring performance

What metrics do we use for mobile page speed?

Which metrics are most important ?

- Load Time: **4.033s**
- First Byte: **0.805s**
- Start Render: **2.282s**
- Fully Loaded: **10.041s**
- Speed Index: **2309**

Which metrics are most important ?



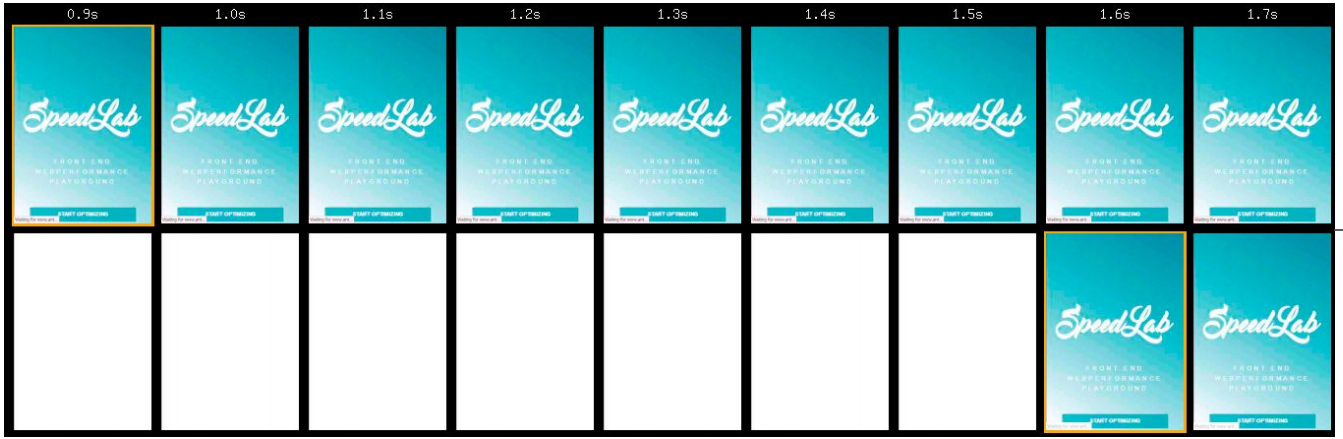
Speed Index:
2309

The **Speed Index** gives us an indication of the perception of speed!

Google

Speed Index : the perception of speed

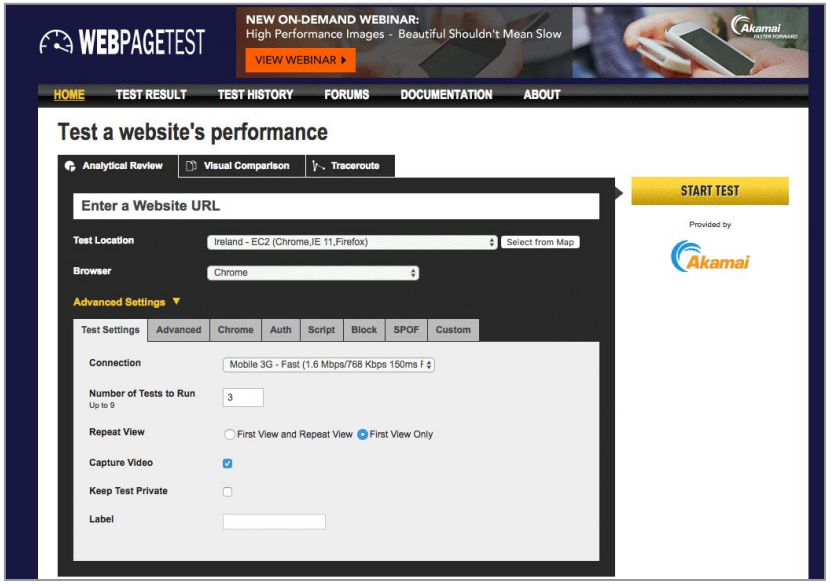
0.9s 1.6s



Speed Index : **750**

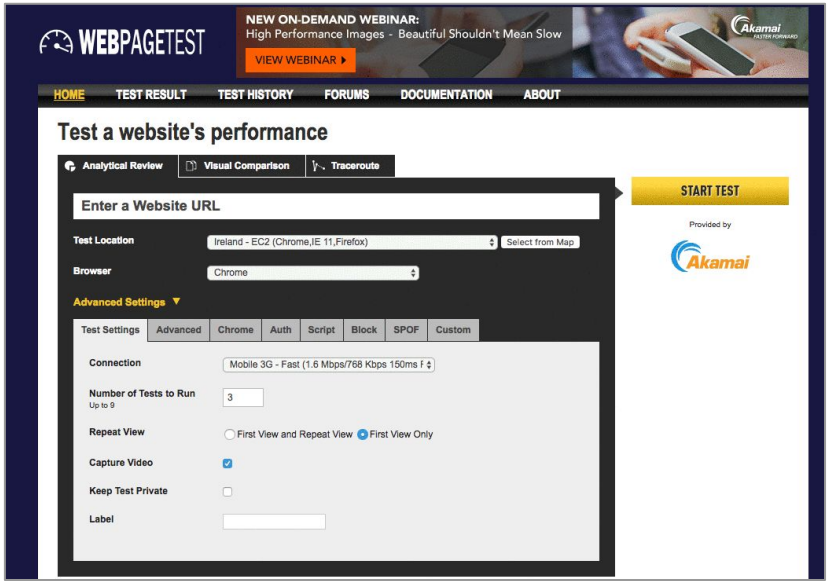
Speed Index : **1500**

Measure your speed index at webpagetest.org



- Connection: **3GFast**
- Number of tests: **3**
- First View Only
- Chrome (tab) :
  Emulate Mobile Browser

Measure your speed index at webpagetest.org



- Connection: **3GFast**
- Number of tests: **3**
- First View Only
- Chrome (tab) :
  Emulate Mobile Browser

Which metrics are most important ?

Performance Results (Median Run)

						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	4.174s	0.805s	2.386s	2413	335	4.174s	30	394 KB	10.003s	96	817 KB	$$---

Plot Full Results

Tracking automatically / tests

WebPageTest API Wrapper for NodeJS

build passing | npm v0.3.4 | downloads 10k/month | dependencies out of date

WebPageTest API Wrapper is a NPM package that wraps WebPage
line tool.

Getting started

```
$ npm install webpagetest -g
```

https://github.com/marcelduran/webpagetest-api

Optimizing

Mobile v.s. Desktop

Bandwidth Limitations

600ms to 1s

is consumed by mandatory 3G network overhead which can't be avoided (average 750 Kbps to 1 Mbps)

Source: "The Search Agency, Optimization Strategies for the Mobile Web"

Why does bandwidth matter?

Desktop = # of Bytes

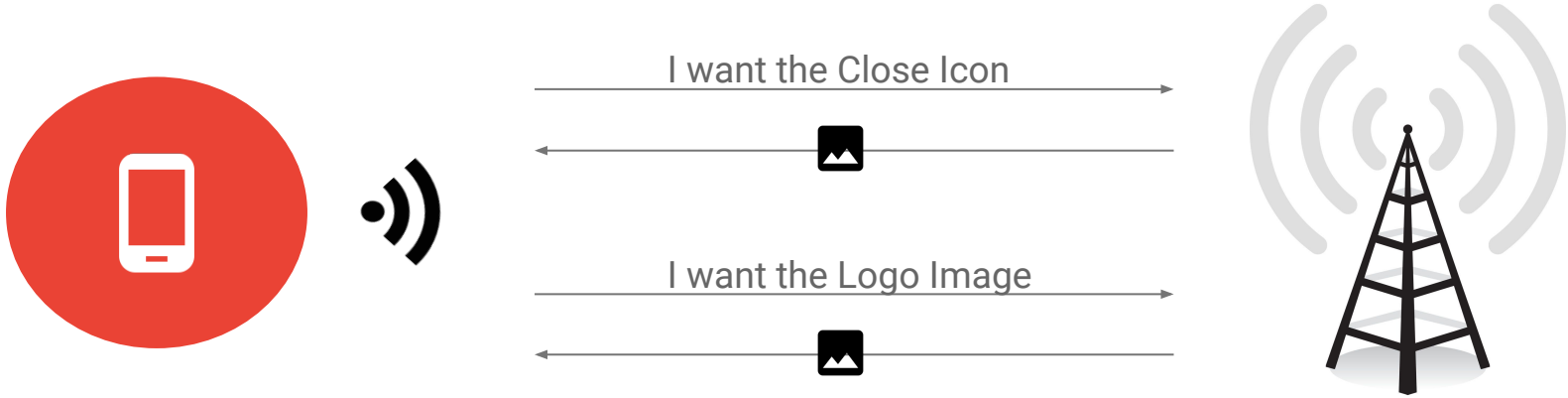
The more you load, the longer it takes

Mobile = # of Requests

Each request involves increased overhead, slowing down your site!

Source: "Mobile Landing Page Optimization"

Mobile round trip = **overhead**



1 second

~~2 second~~

sweet spot

Images

Loading optimization

On average **the majority**
of the loading size of web pages
is **coming from images**

Most performance issues are related to images on mobile:

```
img {
 display: none;
}
```

```
img {
 max-width: 100%;
}
```

Download & shrink

1800 x 1200px

```
img {
  max-width:100%;
}
```

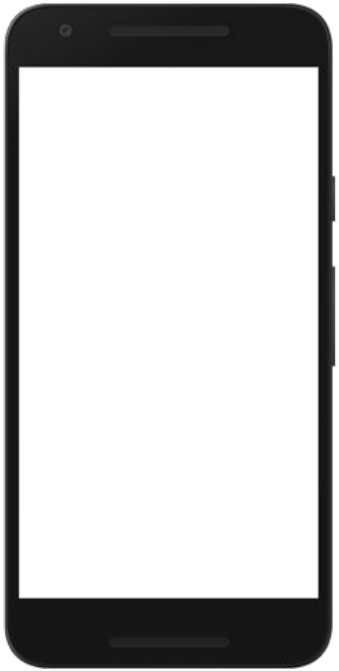
Download & shrink: **srcset**


```
<img
src="css/img/1200x800.jpeg"
media="(min-width: 300px) 300w,
(min-width: 400px) 400w,
(min-width: 640px) 640w,
(min-width: 1000px) 1000w"
srcset="
css/img/450x400.jpeg  300w,
css/img/450x400.jpeg  400w,
css/img/768x400.jpeg  640w,
css/img/1200x800.jpeg 1000w"
alt="" class="img-responsive img-center" />
```

Download & shrink: **background-images**


```
@media (min-width: 768px) {
  .myImage { background-image: url(http://placehold.it/992x150); }
}

@media (min-width: 992px) {
  .myImage { background-image: url(http://placehold.it/1200x150); }
}

@media (min-width: 1200px) {
  .myImage { background-image: url(http://placehold.it/2000x150); }
}
```

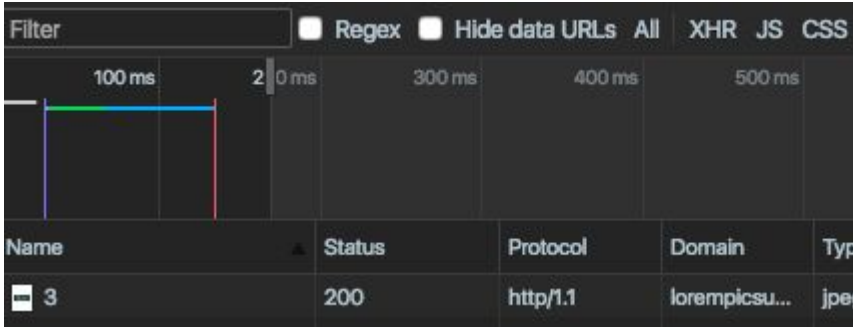
Download & hide



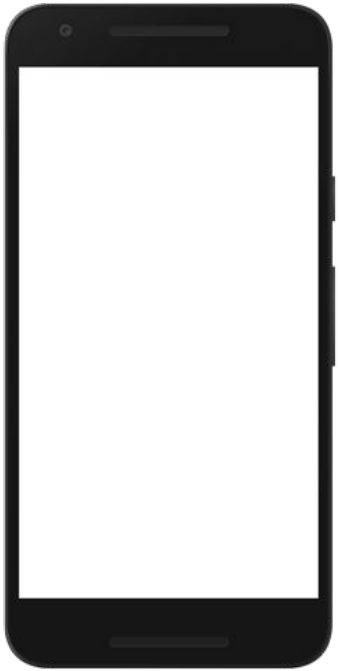
style.css

```
@media only screen and (max-width: 768px){
  img{
    display: none;
  }
}
```

network tab



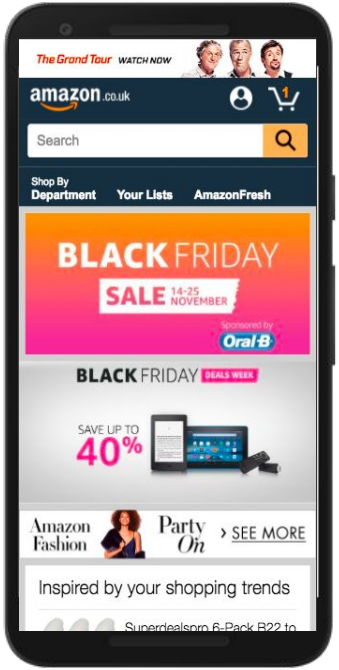
Download & hide: **background-images**



```
@media (max-width: 768px) {
  .myImage { background-image: none; }
}

@media (min-width: 768px) {
  .myImage { background-image: url(http://placehold.it/992x150); }
}
```

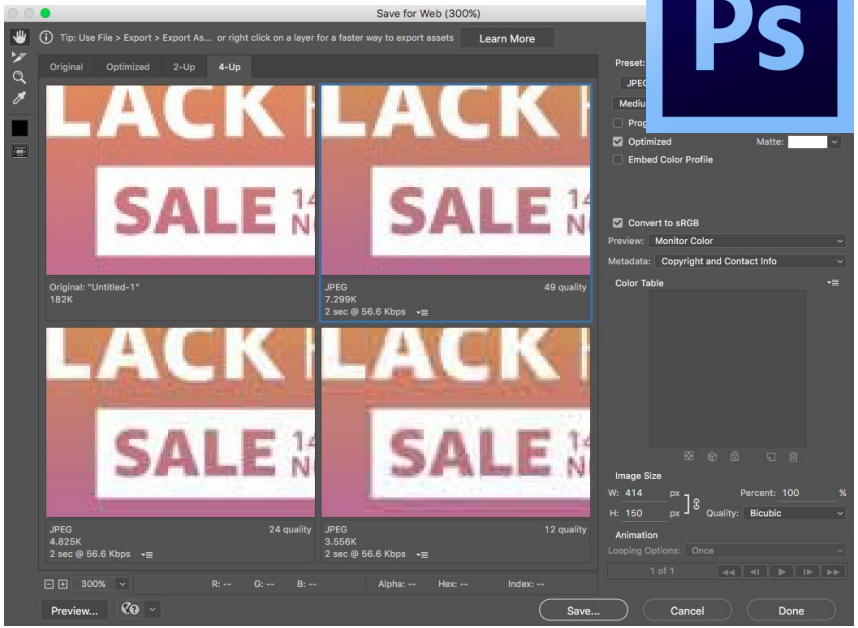
Give up some of the details
Of the images for
Increase in performance

Compress your images



8.4 KB
414 x 150

8.8 KB
414 x 150



Compress your images

150 KB

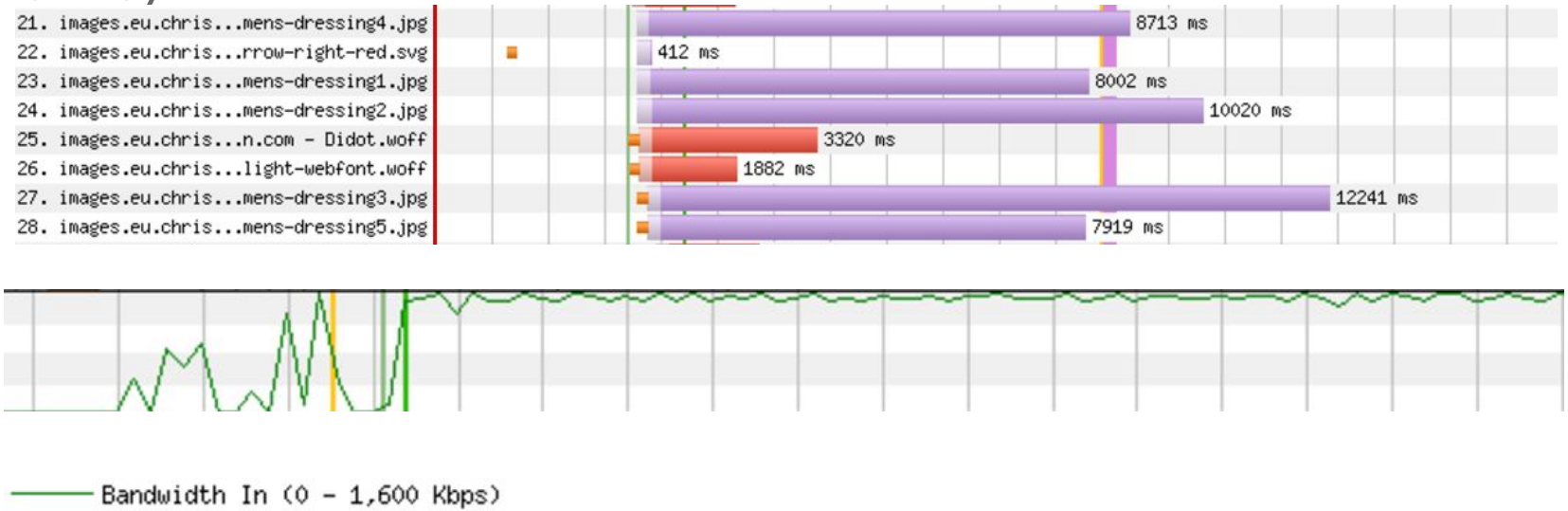
86 KB

Load on demand

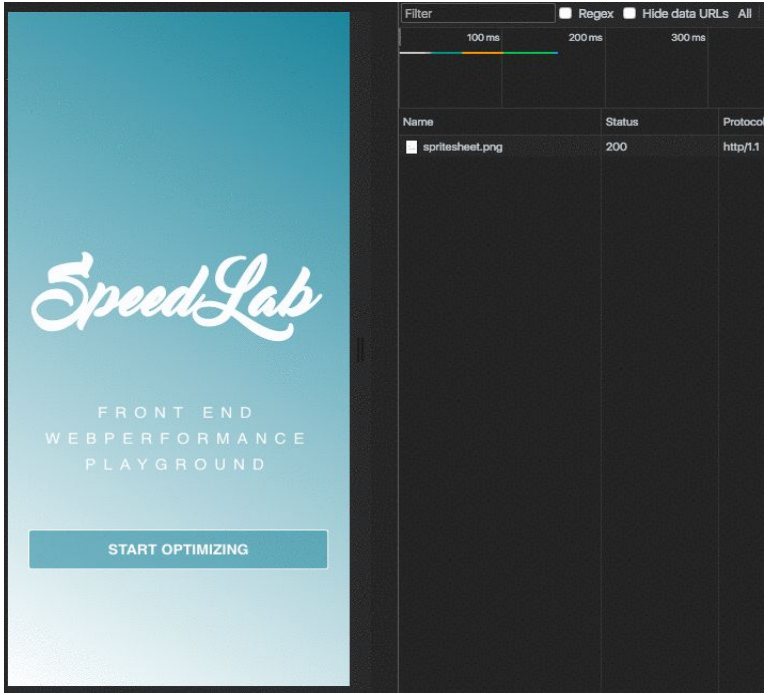
a.k.a:

Lazy Loading

Prevent using max no. of connections due to the loading of many images (at the same time)



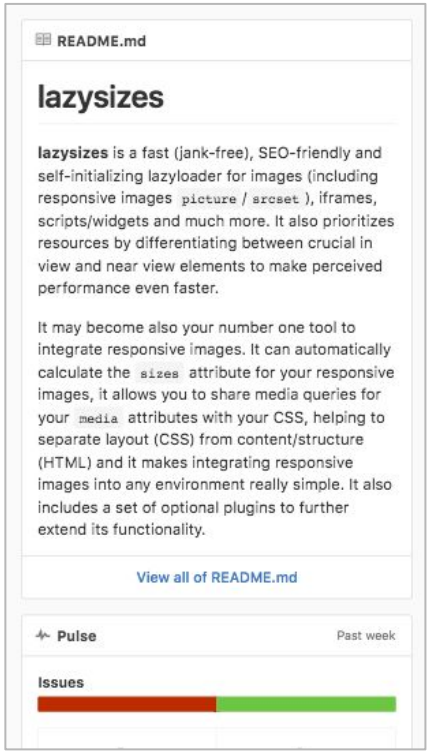
Lazy loading



Lazy loading allows the loading of the image **before** the user needs it

https://github.com/aFarkas/lazysizes

Lazy loading


README.md

lazysizes

lazysizes is a fast (jank-free), SEO-friendly and self-initializing lazyloader for images (including responsive images `picture` / `srcset`), iframes, scripts/widgets and much more. It also prioritizes resources by differentiating between crucial in view and near view elements to make perceived performance even faster.

It may become also your number one tool to integrate responsive images. It can automatically calculate the `sizes` attribute for your responsive images, it allows you to share media queries for your `media` attributes with your CSS, helping to separate layout (CSS) from content/structure (HTML) and it makes integrating responsive images into any environment really simple. It also includes a set of optional plugins to further extend its functionality.

View all of README.md

Pulse — Past week

Issues

```
<script src="lazysizes.min.js" async=""></script>


<img data-src="image.jpg" class="lazyload" />
```

```
<img
alt=""
data-sizes="auto"
data-srcset="css/img/450x400.jpeg 300w,
css/img/450x400.jpeg 400w,
css/img/768x400.jpeg 640w,
css/img/1200x800.jpeg 1000w"
data-src="css/img/1200x800.jpeg"
class="lazyload" />
```

https://github.com/aFarkas/lazysizes

In this room: **Average of 65 resources loaded**

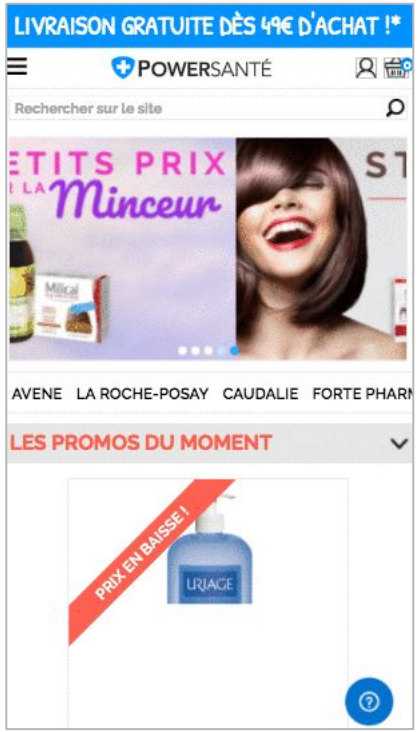
Let’s fix problems!

- Image display: none
- Download & shrink: srcset
- Compress your images
- Lazy loading

An image slider on mobile
will guarantee
a bad user experience!

Who has a slider
on their mobile website ?

Why ban sliders on mSites ?



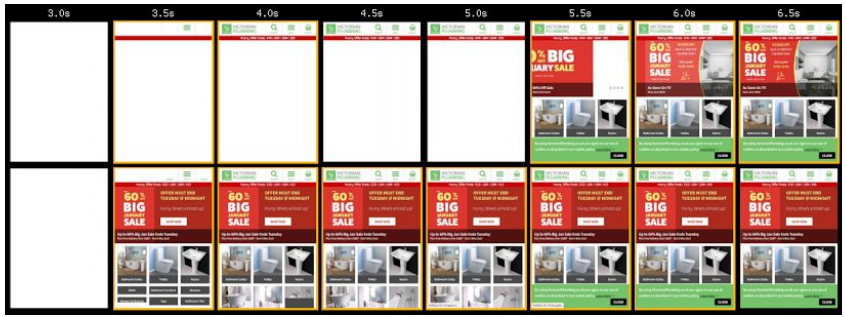
- Poor performance
- The first screen captures the majority of clicks
- Distracts the user by adding movement
- Moves when you read
- Looks like a public transport ad
- Needs multiple images and js to run

Why ban sliders on mSites ?

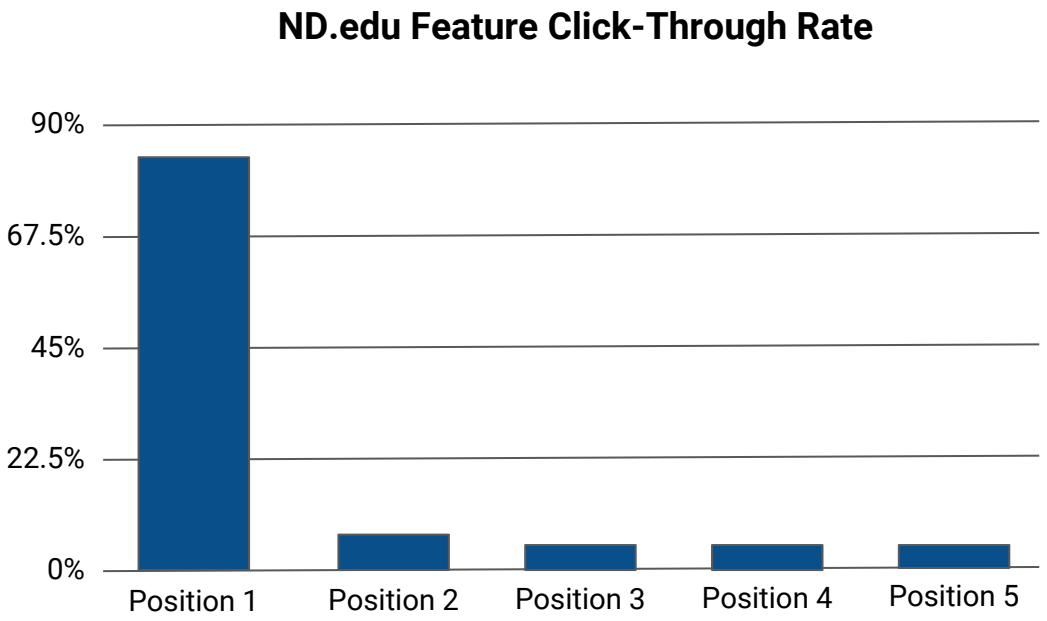
4s

3.5s

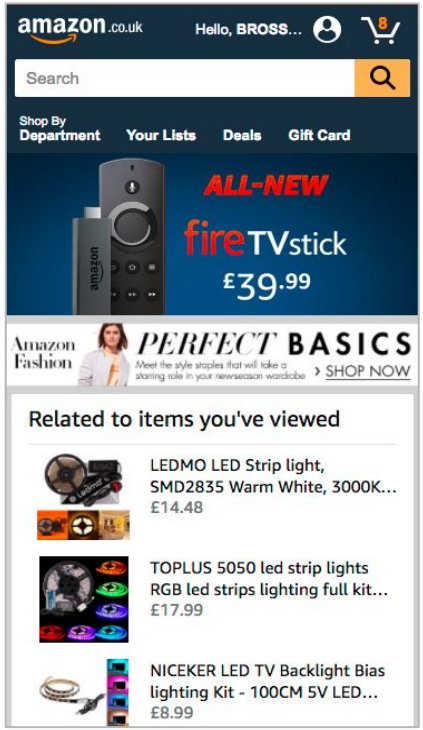
6.5s

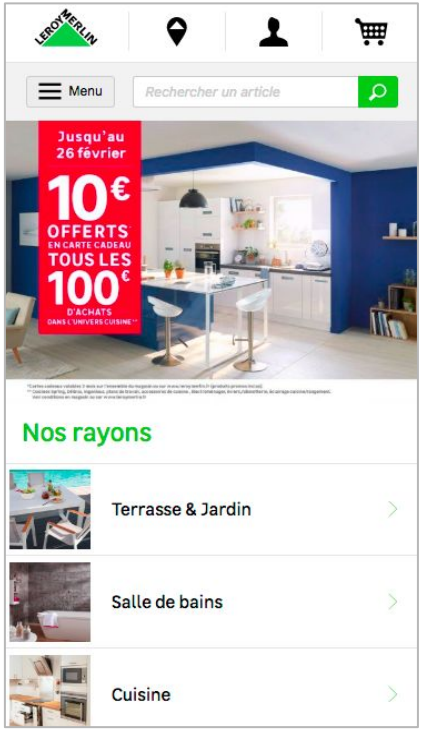


Why ban sliders on mSites ?

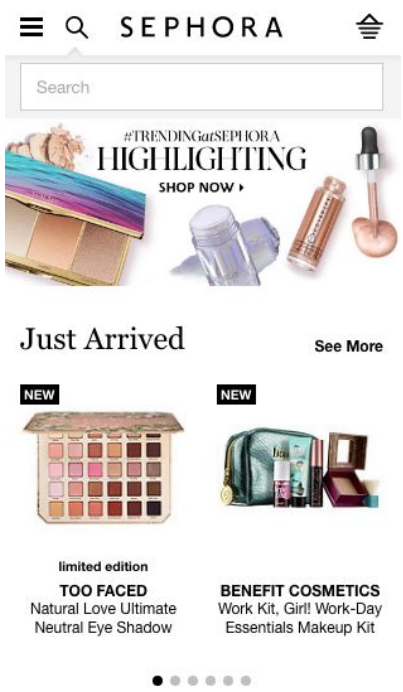


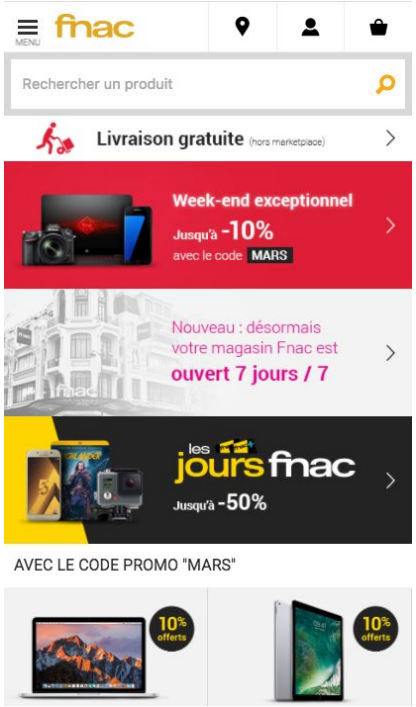
Why ban sliders on mSites ?





Why ban sliders on mSites ?





Live Demo

Original Site | Forked Copy | Optimized Site

Let’s fix problems!

Do you have an image slider?

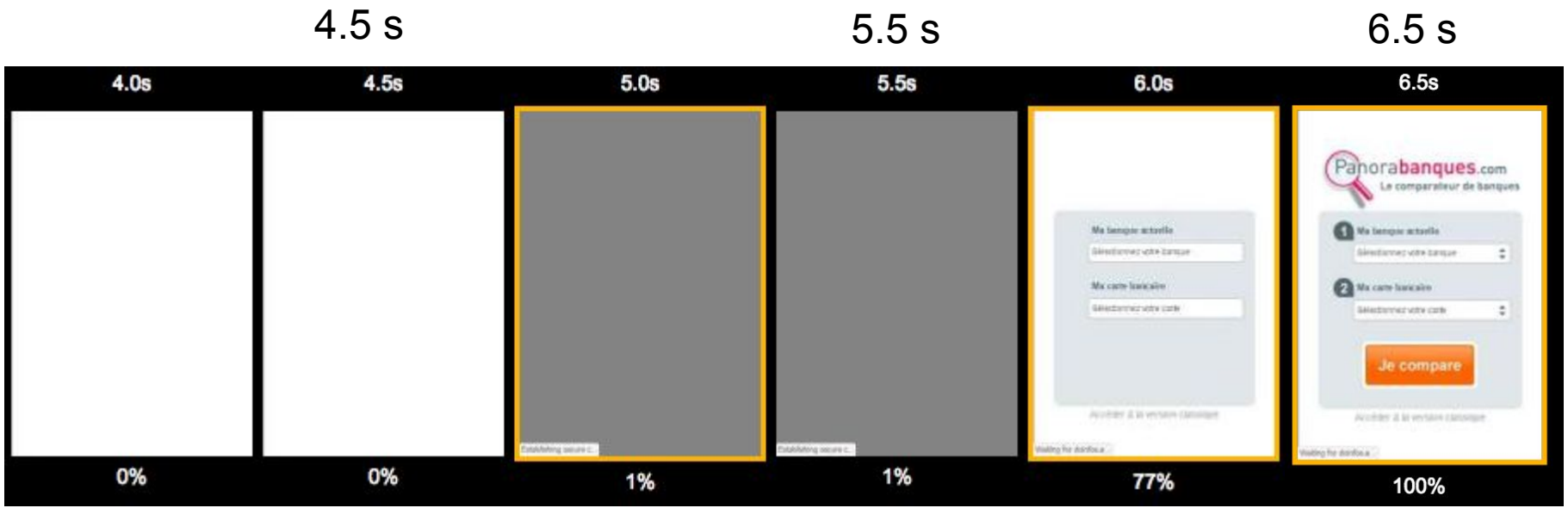
How would your page speed change if you removed it?

Optimizing the critical render path

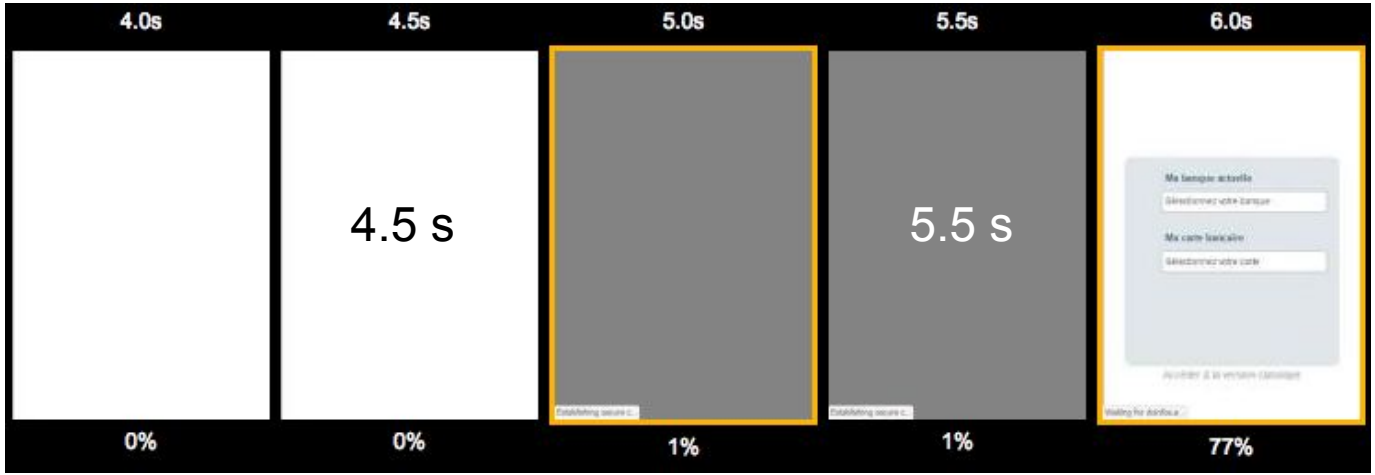
When optimizing the critical path the main idea is to prioritize the loading of the contents above the fold

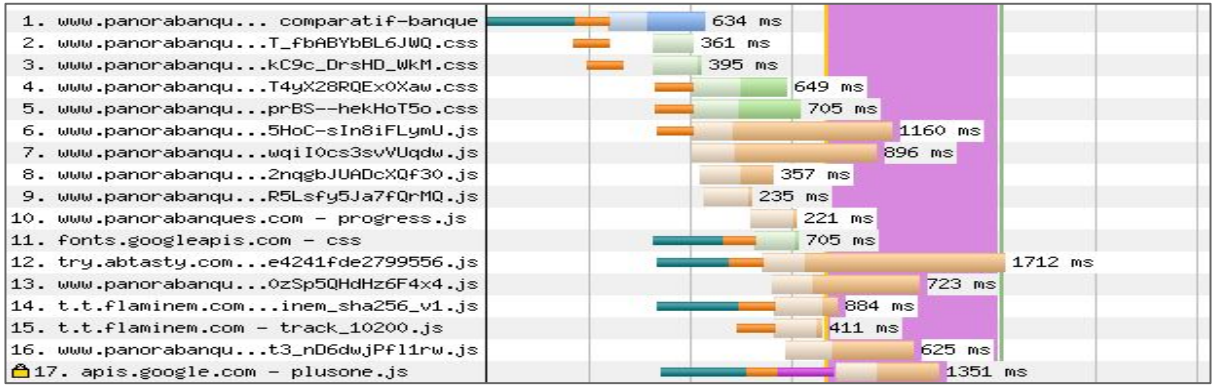
Optimizing the critical render path

Optimizing the critical render path



Optimizing the critical render path





The rendering process

index.html

```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="app.css" />
  <script src="app.js"></script>
</head>

<body>
...
</body>
```

Construction DOM
(parsing HTML)

CSS

JavaScript

CSSOM + DOM
= render Tree

Layout & Paint

CSS is render blocking

```
p{color: red;}
p{color: green;}
```

Hello Amsterdam Hackathon 2017

CSS is render blocking

app.css

```
<?php
header("Content-type: text/css; charset: UTF-8");
sleep(8);
?>
```

Construction DOM
(parsing HTML)

CSS

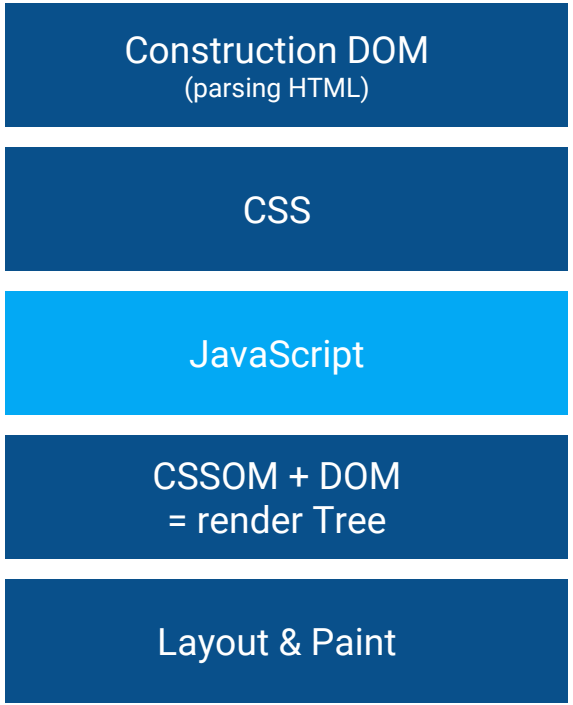
JavaScript

CSSOM + DOM
= render Tree

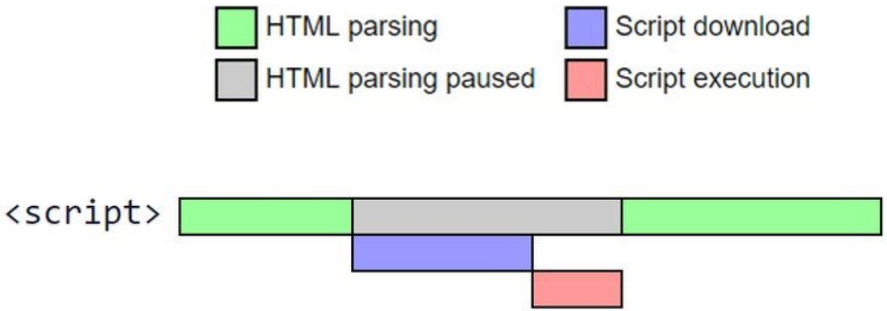
Layout & Paint

JS blocks the parser

```
<link rel="stylesheet" href="app.css" />

<p>Hello Paris

  <script>
    var e = document.getElementsByTagName("p")[0];
    e.style.color = "red";
  </script>

Hackathon 2016</p>
```



JS blocks the parser



Construction DOM
(parsing HTML)

Requête CSS / JS

Parsing CSS : CSSOM

Execution JS

CSSOM + DOM
= render Tree

Layout & Paint

JS blocks the parser: Header Demo

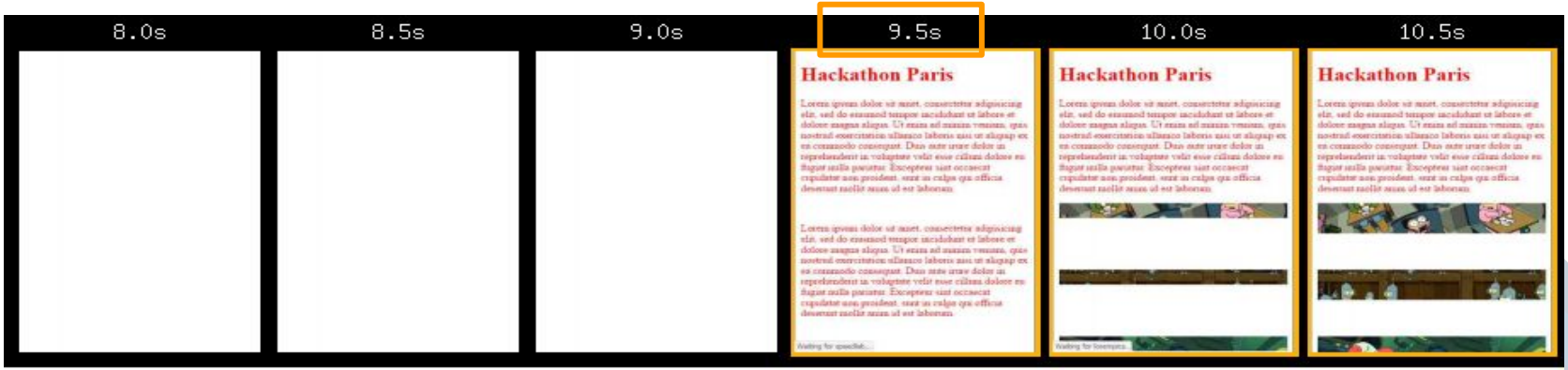
Header

index.html

```
<html>
<head>
  <script src="app.js.php"></script>
</head>
<body>
  ...
</body>
```

app.js

```
<?php header('Content-type: text/javascript'); ?>
<?php sleep(8);?>

function hello(){
  console.log('Hello World !');
}
```



JS blocks the parser: Footer Demo

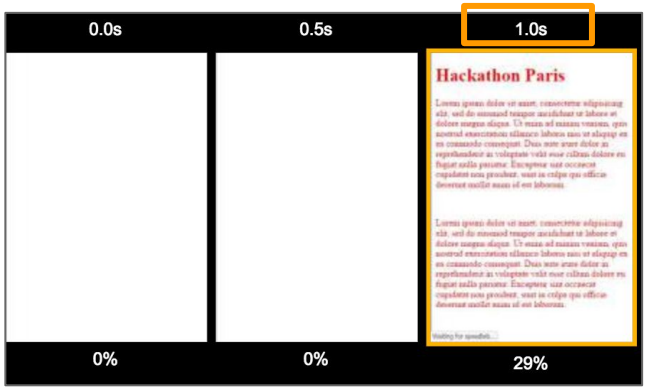
Footer

index.html

```
</body>
  <script src="app.js.php"></script>
</html>
```

app.js

```
<?php header('Content-type: text/javascript'); ?>
<?php sleep(8);?>

function hello(){
  console.log('Hello World !');
}
```



How to **optimise** the loading of my **JavaScript ?!**

Consider Fixing:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

Your page has 1 blocking script resources. This causes a delay in rendering your page.

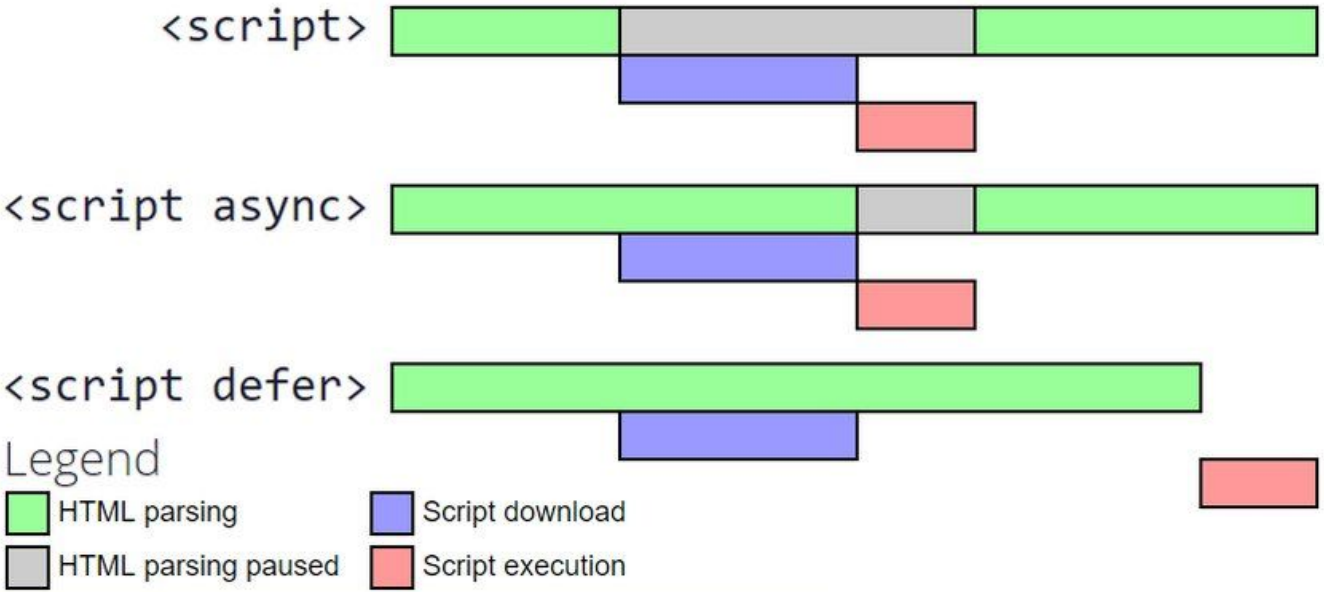
None of the above-the-fold content on your page could be rendered without waiting for the following resources to load. Try to defer or asynchronously load blocking resources, or inline the critical portions of those resources directly in the HTML.

Remove render-blocking JavaScript:

http://brianjackson.io/wp-content/keycdn-cs-01/jquery.min.js

▲Hide details

Optimize the loading of JS



Optimize the loading of JS

Put your **JavaScript in the footer !**

Utilize **defer and async**

Defer : if the execution order of the script is important

Async : if the script is independent

Why **optimise** the loading of my **CSS** ?!

! Should Fix:

Eliminate render-blocking JavaScript and CSS in above-the-fold content

Your page has 2 blocking CSS resources. This causes a delay in rendering your page.

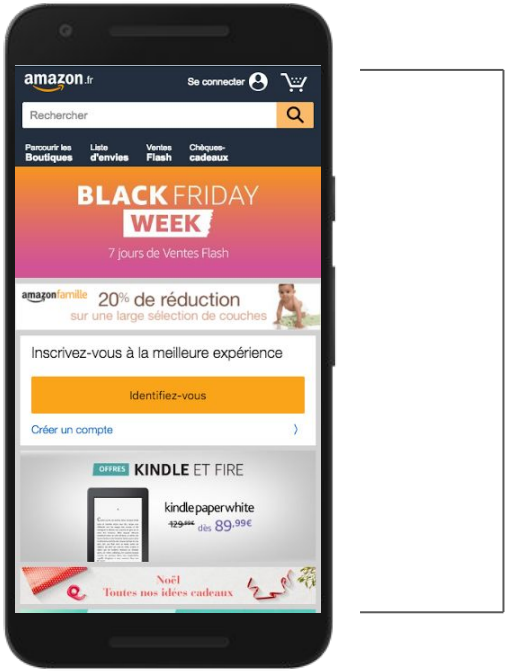
None of the above-the-fold content on your page could be rendered without waiting for the following resources to load. Try to defer or asynchronously load blocking resources, or inline the critical portions of those resources directly in the HTML.

Optimize CSS Delivery of the following:

- http://speedlab.antoinebrossault.com/css/app.css
- http://speedlab.antoinebrossault.com/css/sprite.css

▲Hide details

Optimize the loading of CSS



<style> Inline critical CSS </style>

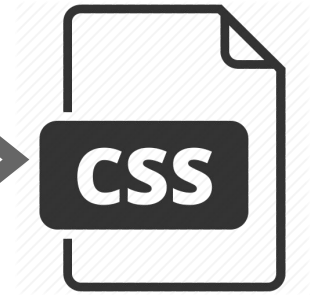
<script> Load the rest of the css </script>

Extracting critical CSS

CSS critique

```
var critical = require('critical');

critical.generate({
  src: "http://amazon.fr",
  dest: 'critical.css',
  width: 412,
  height: 732
});
```

extraction (node.js)



critical.css

https://github.com/addyosmani/critical

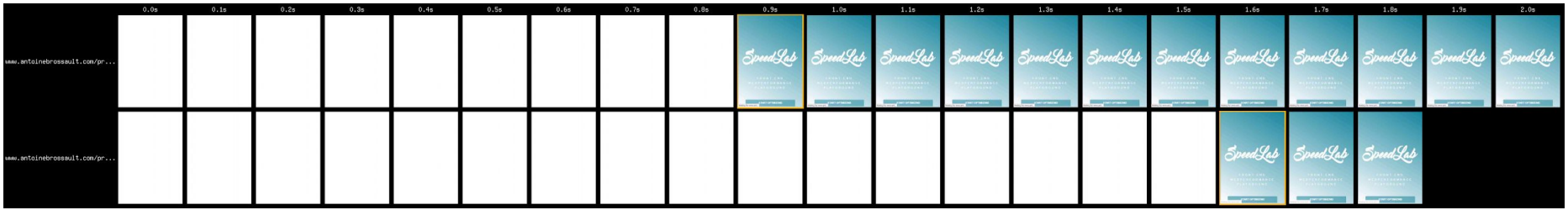
Loading non-critical CSS

CSS NON critique

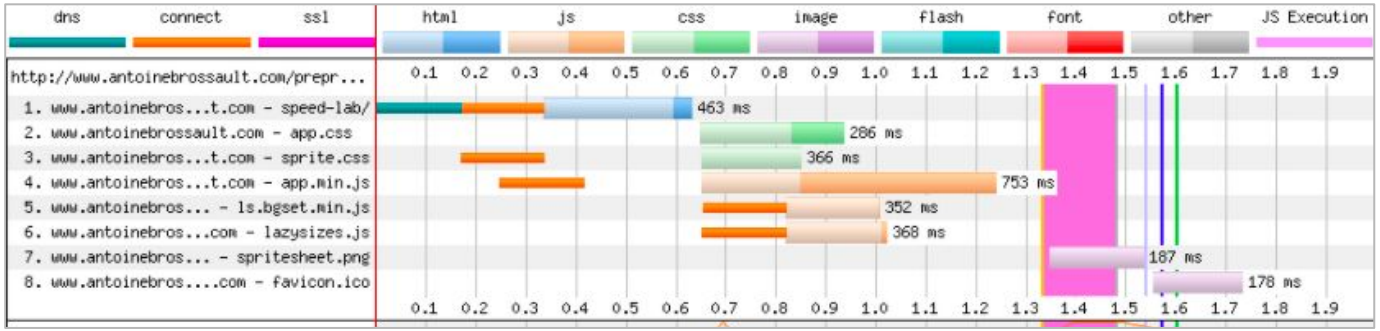
```
<script>loadCSS( "css/app.css" );</script>

<script>loadCSS( "css/sprite.css" );</script>
```

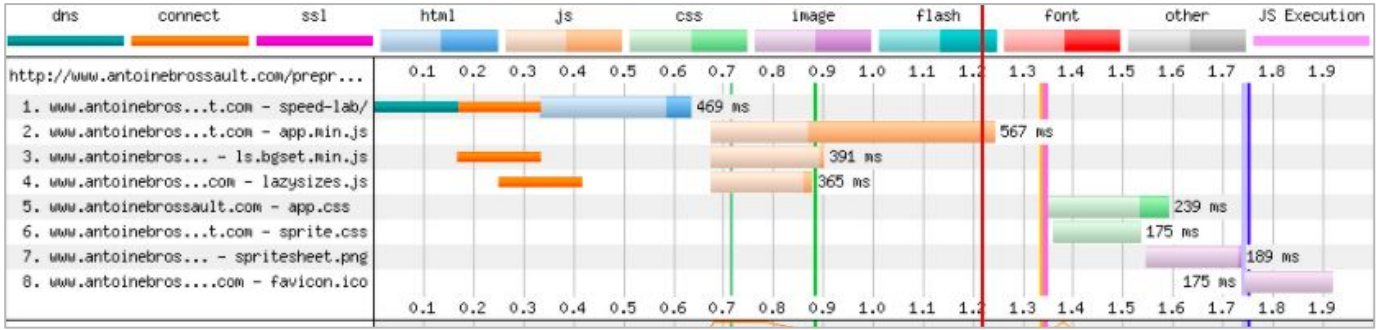
\

Loading non-critical CSS & async



Loading non-critical CSS & async





Let’s fix problems!

- Prioritize ATF content
- Optimize JavaScript
- Optimize CSS

```
Hack away!

isAudienceReady
   ? hack on your chosen page()
   : questions();
```

Hacking

- 10:30 Enter your url, measure your page in goo.gl/RFE2XG
- 10:45 Start hacking
- 12:00 Lunch time!
- 15:00 Deploy your page, measure new metrics
- 15:30 Ending keynote - highlights - awards

Prizes

Summary

- Start measuring mSpeed, consider using Speed Index as a KPI
- Start optimizing images
    1. Compress
    2. Lazy load
    3. Don’t use sliders. EVER.
- Optimizing the critical render path
    1. Prioritize ATF content
    2. Optimize JavaScript
    3. Optimize CSS

mSite Speed Resources for Developers

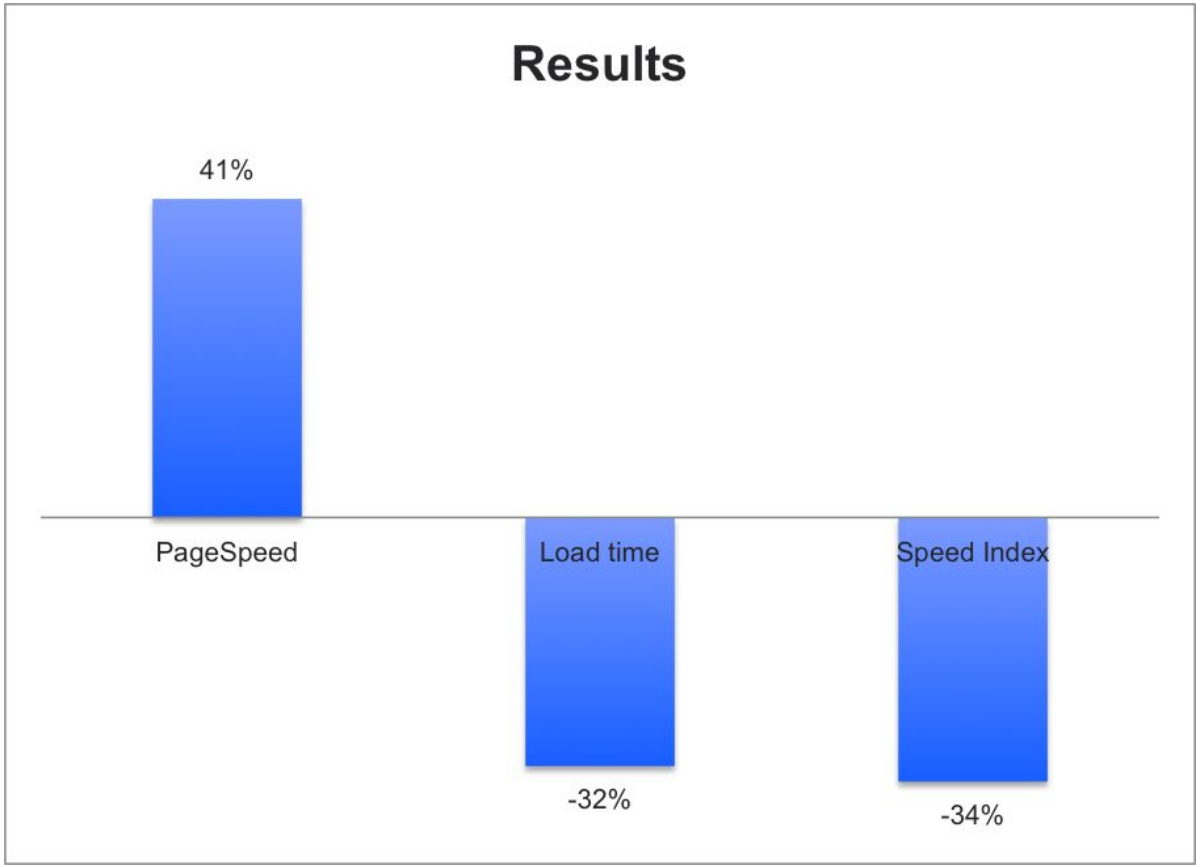
Access a PDF version by typing the following short link into your browser:

goo.gl/JAUPK1

Link is case sensitive.

```
console.log( “Thanks” );
```

RESULTS


Results
41%
PageSpeed
Load time
-32%
Speed Index
-34%

How to get a great Hackathon:

Ask / Share

googlespeedhackathon.slack.com
Channel: #speedupdk

Moving forward...

Rock Stars

Within the art of web performance

Ilya Grigorik
@igrigorik
Web performance at Google

Rick Viscomi
@rick_viscomi
DevRel at Google

Thomas Steiner
@tomayac
Mobile web at Google

Tammy Everts
@tameverts
CXO Speed Curve

Andy Davies
@AndyDavies
Web performance at NCC Group

Lara Hogan
@lara_hogan
VP Engineer at Kickstarter

Patrick Meenan
@patmeenan
Created WebPageTest

Cheney Tsai
@cheneytsai
Mobile web at Google

Tim Kadlec
@tkadlec
Developer Advocate at snyk.io

How do we fix the whole site?

How can non-developers work with speed?

Let’s create a process!

How to get great at speed

1. To your analytics

Start making speed a part of your KPIs

2. To your developers

Get the knowledge you need, and give them resources to finish the project

3. To your creative team

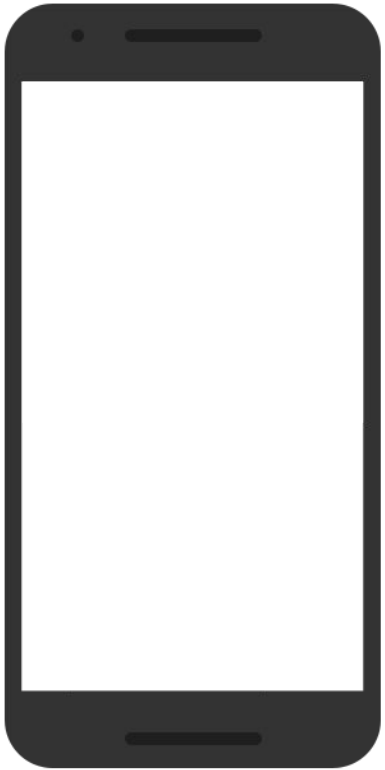
Introduce a performance budget. This will connect developers and creatives

1. Show this to your analytics: Start making speed a part of your KPIs

- Go to WebPageTest.org
- Enter your url with the settings to the right
- Set up a dashboard where you see mobile conversion rate and the correlation with load time

- Connection: **3GFast**
- Number of tests: **3**
- First View Only
- Chrome (tab) : **Emulate Mobile Browser**

1. Set targets for speed



Load Time: **X.xx s** ← Psst! Set 5 s as an initial target. Then go after 3 s!

Speed Index: **x** ← Set below 3000 as a target.

How to get great on speed

1. To your analytics

Start making speed a part of your KPIs

2. To your developers

Get the knowledge you need, and give them resources to finish the project

3. To your creative team

Introduce a performance budget. This will connect developers and creatives

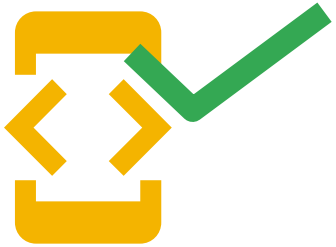
2. To your developers:

- Now and again, Google host Speed Hackathons. Make sure your developers are there!
- Make sure your developers have the resources to fix the whole site.
- Want to fix the problems now with external help? Get experts to analyze and/or implement the advice. We recommend our hero partner Netlight, that you can read more about here. Contact jimmy.peterson@netlight.com to find out more.

How to get great on speed

1. To your analytics

Start making speed a part of your KPIs



2. To your developers

Get the knowledge you need, and give them resources to finish the project

3. To your creative team

Introduce a performance budget. This will connect developers and creatives

3. To your creative team: Introduce a performance budget

Concrete and easy to start with

1. Page weight < 1 Mb
2. Start render < 0,5 s
3. Load time < 5 s
4. Speed Index < 3000

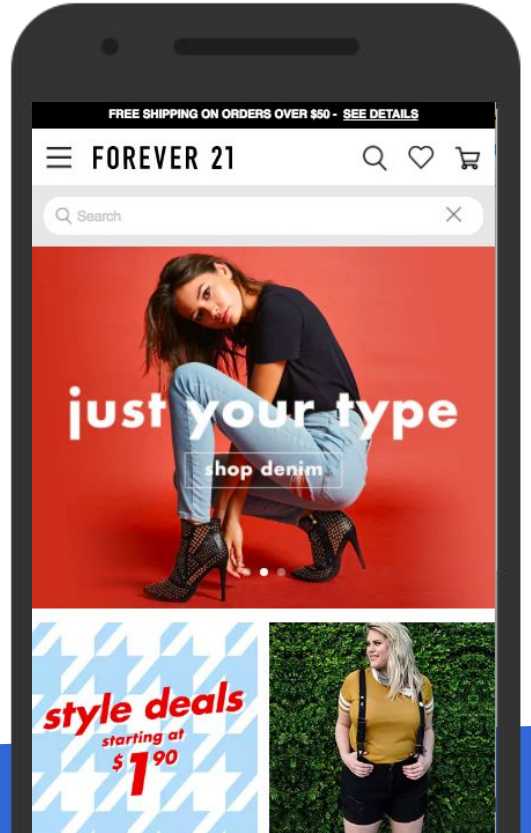
Advanced teams - soon you'll be here!

Learn more about Performance Budgets: The Path to Performance by Katie Kovalcin - Slide 56 and onwards

3. If the target is <1Mb in page weight, you can should not create heavier pages

Page weight budget: < 1 Mb

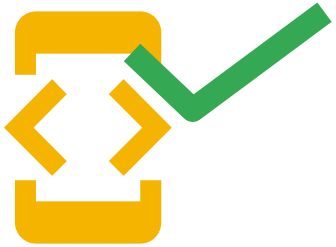
Images: 1.035 Mb
JS: 0.337 Mb
Font: 0.068 Mb
HTML: 0.020 Mb
CSS: 0.014 Mb
= 1.496 Mb



How to get great on speed

1. To your analytics

Start making speed a part of your KPIs



2. To your developers

Get the knowledge you need, and give them resources to finish the project

3. To your creative team

Introduce a performance budget. This will connect developers and creatives

Encourage this way of working

https://youtu.be/tXn0_tZAI1o

Get mWeb certified

Introducing the Mobile Sites certification, for web developers

Monday, April 3, 2017

Posted by Chris Hohorst, Head of Mobile Sites Transformation

Mobile now accounts for over half of all web traffic[1], making performance on small screens more important than ever.

Despite this increase, a recent study by Google found that the average time it takes to load a mobile landing page is 22 seconds. When you consider that 53% of mobile site visitors will leave a site if it takes more than three seconds to load, it's clear why conversion rates are consistently lower on mobile than desktop.

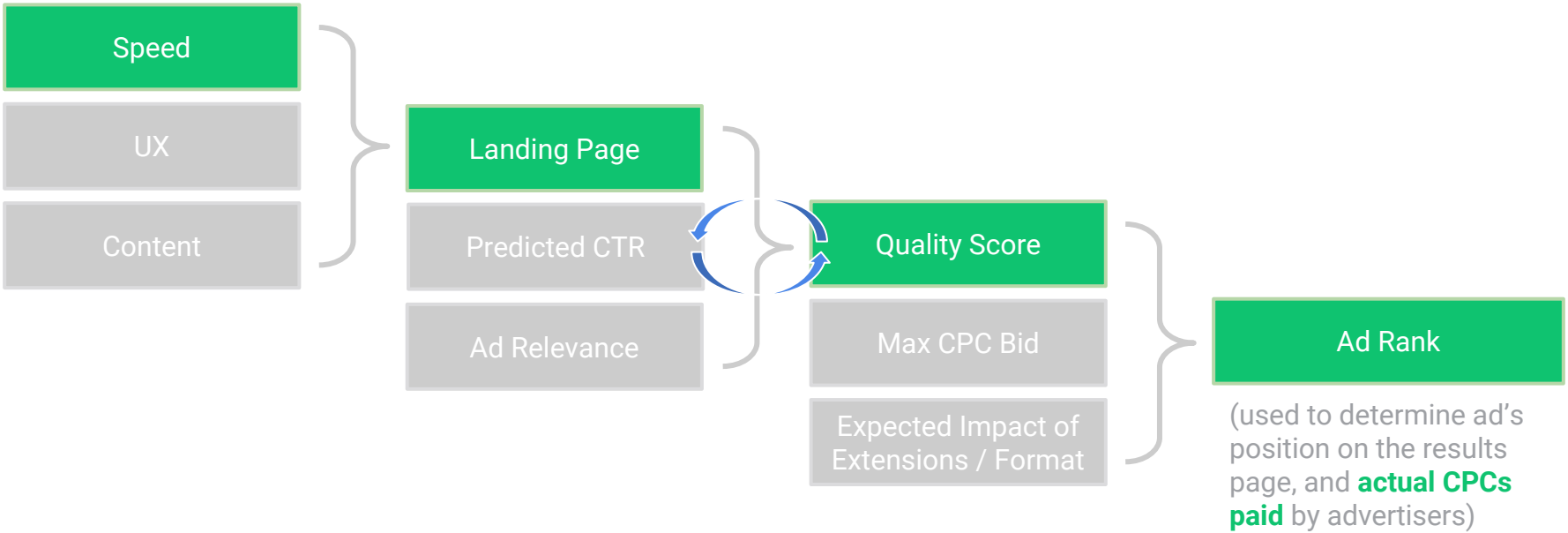
https://developers.googleblog.com/2017/04/introducing-mobile-sites-certification.html

You are heroes!

THANK YOU

INSIGHTS ON THE WALL

The speed of your mobile site directly influences ad rank!



More wins! **Speed = Business Impact**

Google

2% *slower*

↓

2%

fewer searches/user

YAHOO!

400ms *faster*

↓

9%

more traffic

Aol.

Faster pages

↓

more page views

amazon.com

100ms *faster*

↓

1%

more revenue

shopzilla

5s *faster*

↓

25%

more pageviews, 7-12% more revenue

Fathead

37% *faster*

↓

70%

increase in mobile revenue per user

GQ

80% *faster*

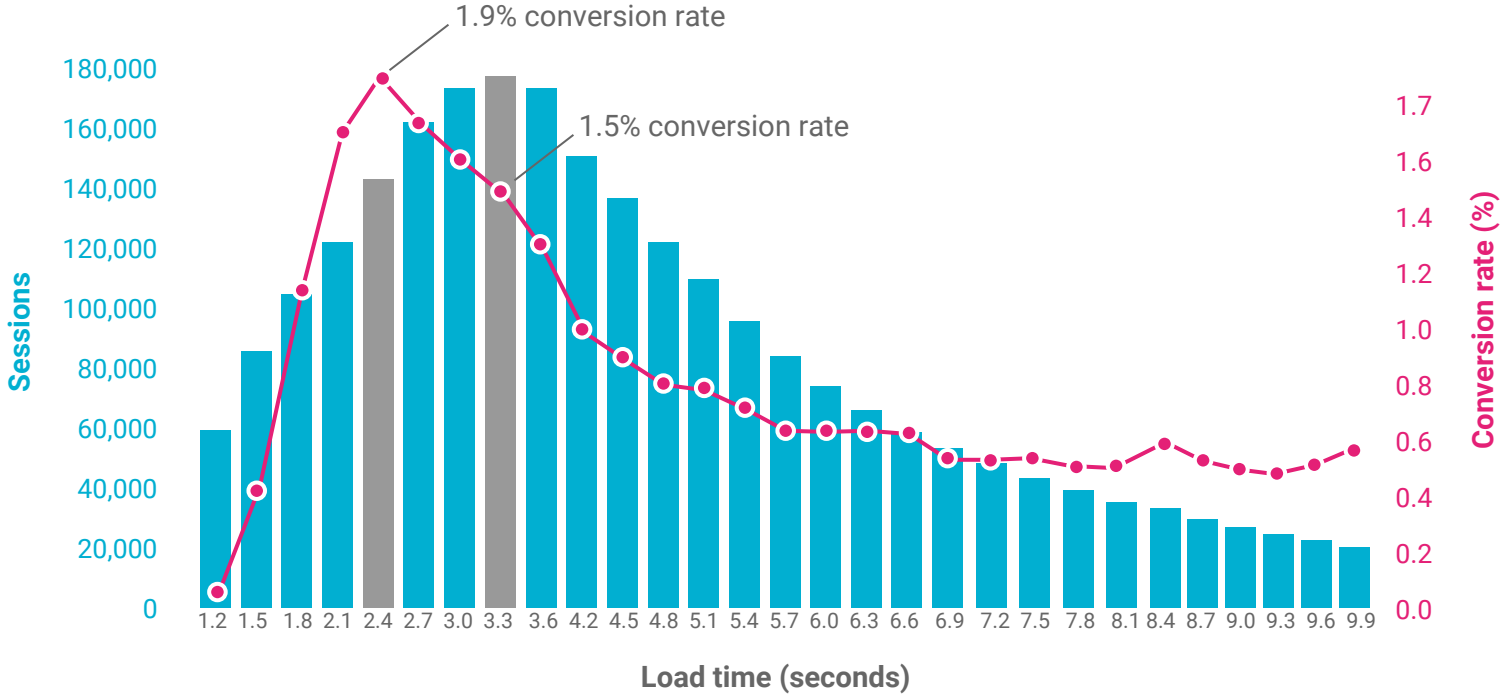
↓

108%

increase in ads interaction rate

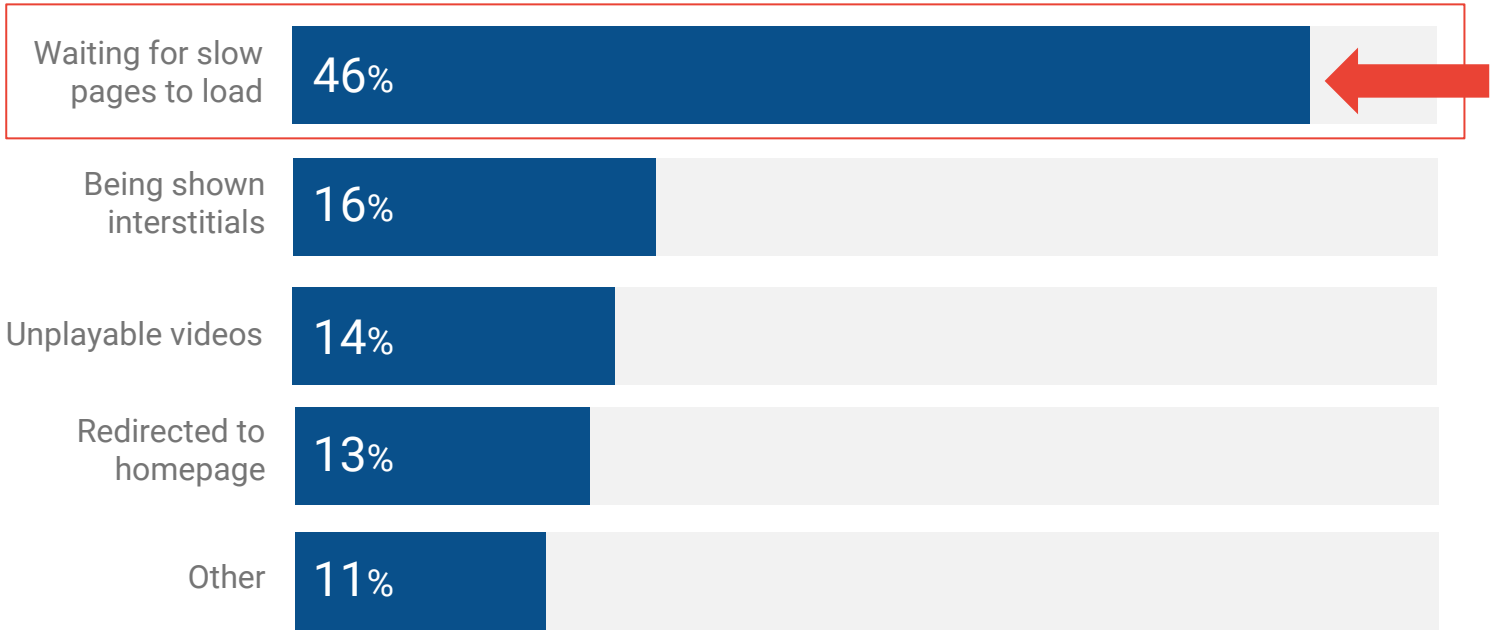
Case Study:

Mobile pages that load 1s faster see up to 27% increase in CvR



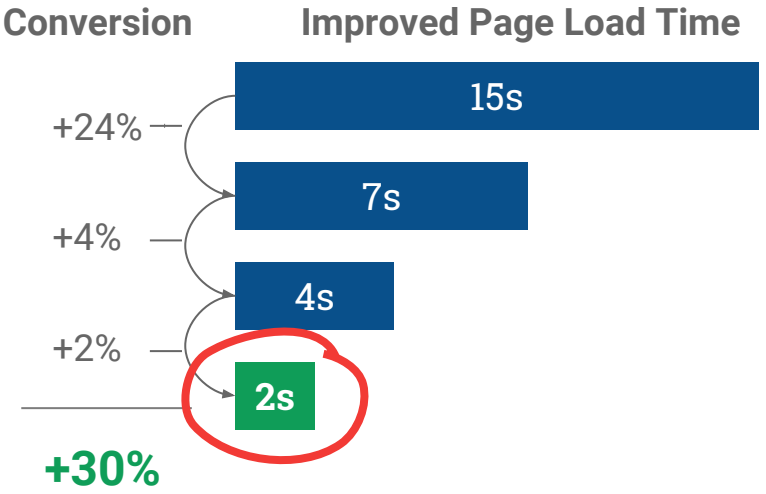
Source: SOASTA Case study (September 1st, 2015)

“What do you dislike the most when browsing the web on your mobile device? ”



How customers feel and react to speed

Seconds	General sentiment
1 to **2s**	Feeling **good**, let's continue
3 to 6s	Ok, this is slow, but **tolerable**
7 to 10s	**Ugh**, this better be worth it
10s+	Wow, **finally**...



Source: "Why Page Load Time Matters for Conversion Optimization", Source: "A Beginner's Guide to Website Speed Optimization"

On average **the majority**
of the loading size of web pages
is **coming from images**

In **2020**, 70% of cellular network connections globally will occur at **3G and slower speeds**

Pages from the **automotive, retail, and technology** sectors, on average, take the **longest to load**. And they also have some of the most bloated pages on the web.

If page load time goes from **one second to seven seconds**, the probability of a mobile site visitor **bouncing increases 113%**.

Fifty-three percent of people
will leave a page if it takes longer than 3 seconds to load.

Simply **compressing images and text** can be a game changer—30% of pages could save more than 250KB that way.

from “Why Mobile Page Speed Is a Visual Designer’s Problem”

Fifty-three percent of people will leave a page if it takes longer than 3 seconds to load.

SOURCE: Google Data, Global, n=3,700 aggregated, anonymized Google Analytics data from a sample of mWeb sites opted into sharing benchmark data, Mar. 2016.

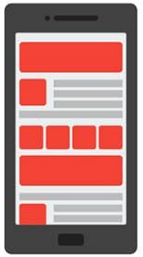
Fewer images per page creates more conversions

Converted

19

images per page
on average

Unconverted



31

images per page
on average

A faster full site load time leads to a lower bounce rate

Average load time for all elements on the page

Top mobile site factors that increased bounce rate

DOM ready time

Full page load time

79% of shoppers who are **dissatisfied** with site performance say they're less likely to purchase from the same site again

Reduce number of requests

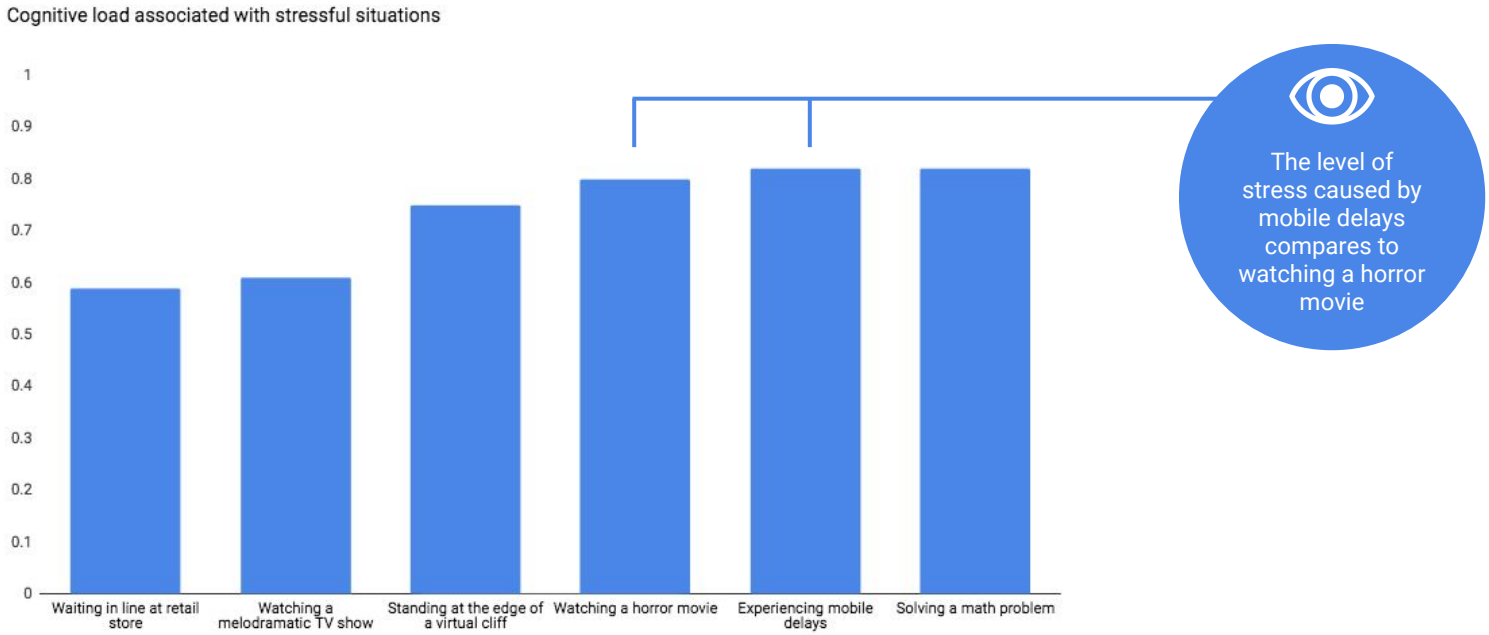
Requests = Number of resources the browser needs to request from the server.

Fewer requests = faster load time. Goal: <80 requests.ce
Number of Requests

Compress resources with GZIP

GZIP can reduce the size of a text-based resource (like CSS or JS) by as much as 70-80%. The server sends a compressed file, and the browser uncompresses it. All modern browsers support GZIP compression; it simply needs to be enabled on the server.

Users associate mobile load time with stressful situations

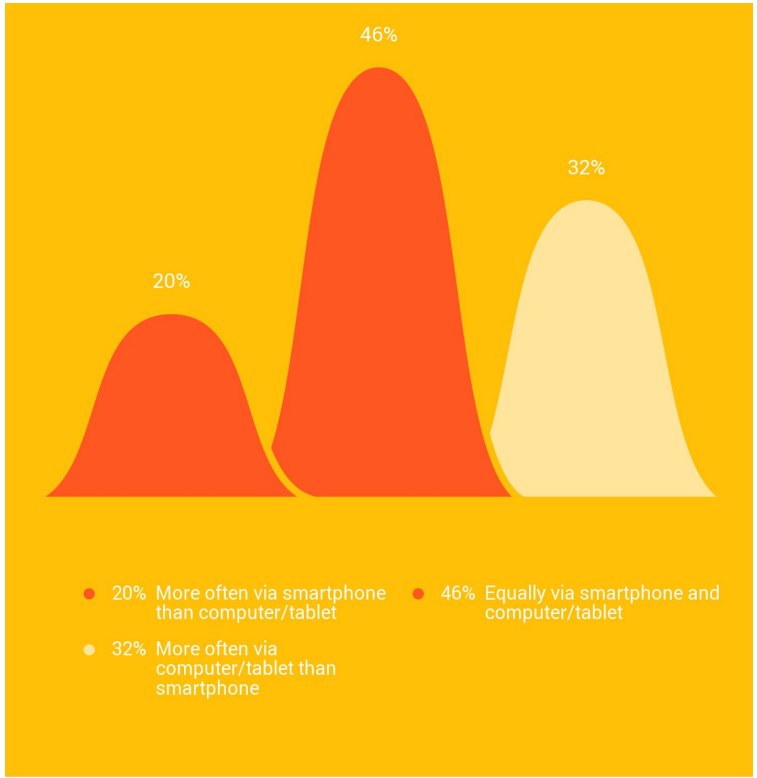


Source: The stress of streaming delays, Ericsson Consumer Lab, Neurons Inc., February 2016

Online Resources

- Chrome DevTools
- Chrome: Optimizing Performance
- W3C Mobile Checker
- PageSpeed Insights
- CSS-Tricks
- YSlow
- WebPagetest
- Smashing Magazine
- Pingdom
- ImageOptim
- http archive (Performance Stats)
- SFGate.com Sample Audit w/Paul Irish
- The Performance Beacon (Blog)
- SSL Server Test
- Paul Lewis mWeb Performance Auditing

Mobile is a crucial access point for people coming online

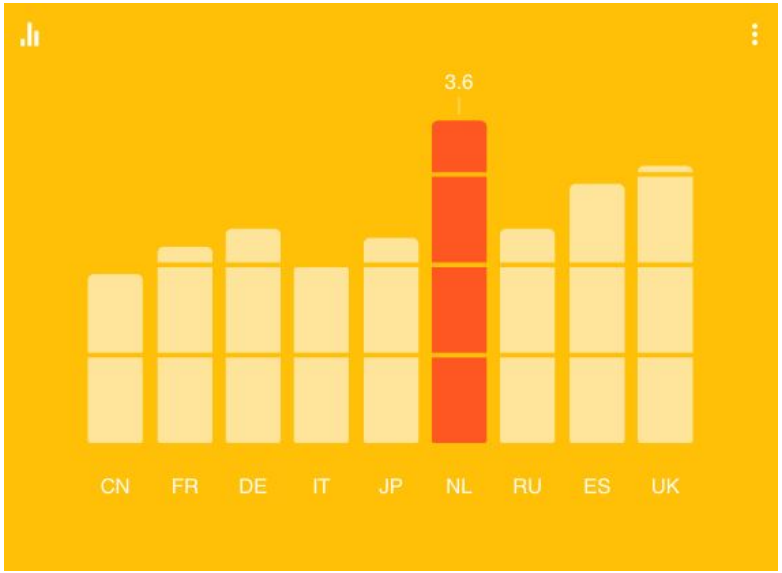


Devices Most Often Used to Come Online

Consumer Barometer Survey, 2014/2015 - Netherlands.

Netherlands has more screens than ever

Number of connected devices per person



Consumer Barometer Survey, 2014/2015 - Netherlands.

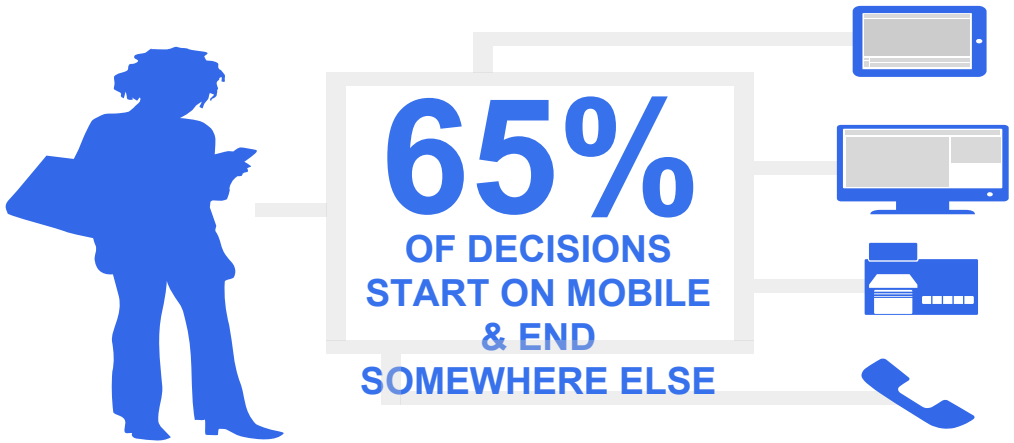
Search behaviour has evolved

16% of monthly search queries are new

10 more Google searches take place on mobile devices than on computers in 10 countries

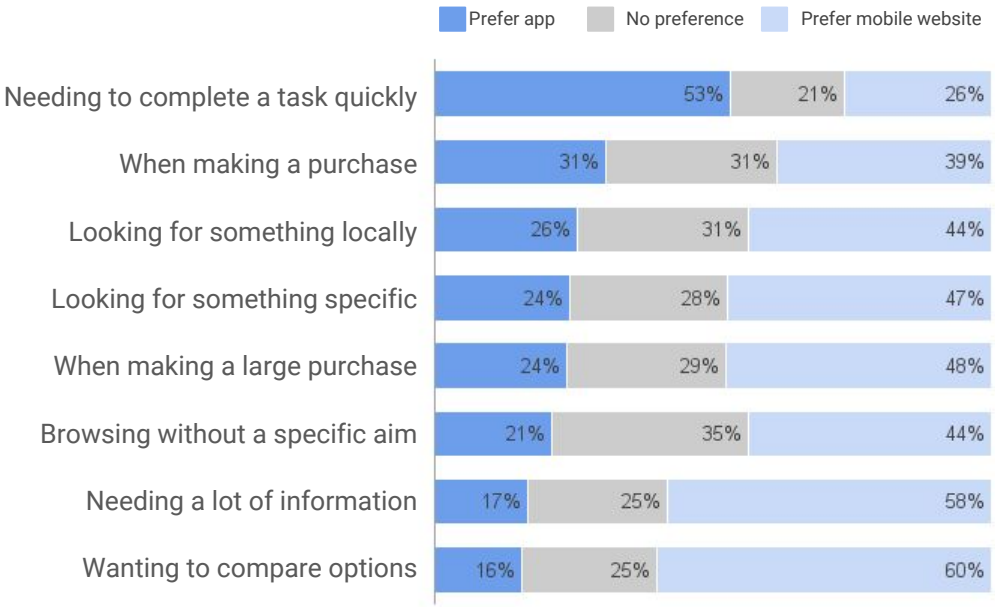
90% of users search across multiple screens

With device proliferation, the path to purchase has changed



Source: The new Multiscreen World Study

Platform preferences for shopping activities



- Mobile websites are preferred for most shopping activities except when speed is needed.
- There is an especially strong lean for mobile websites when it comes to broad information gathering situations such as browsing without a specific aim or wanting to compare options.
- Containing more information makes mobile websites the preferred platform for large purchases (see slide 13).

Base: 501
Q12. Imagine you were in each of the following situations. Would you prefer to use a Shopping app, a mobile website or would you have no preference?

Why aren’t travelers booking on mobile?

54% and **69%**

of leisure travelers **of business travelers**

say that mobile limitations or mobile usability are their main reasons for booking on another device.

88%

of mobile travelers would **switch to another site** or app if yours doesn’t satisfy their needs.

Source: Google/Ipsos MediaCT, “The 2015 Traveler's Road to Decision,” base: U.S. leisure/business travelers, n=599, Aug. 2015. Google/Ipsos Connect, travel playbook omnibus, base: U.S. travelers 18+ who use a smartphone, n=1,304, Apr. 2016.

150x
per day

At work, book Coachella tickets
11:36am

At lunch, play Scrabble while waiting in line
1:33pm

Buy new tote to take to Coachella
11:15am

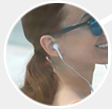
Use maps to get directions to Creole food truck
1:13pm

Wake up and read news online
6:50am

On the bus, read articles about Coachella
8:42am

Browse festival styles on YouTube
7:15pm

Use flashlight app to find dropped earring
11:09pm



At bus stop, listen to new music playlist
8:30am

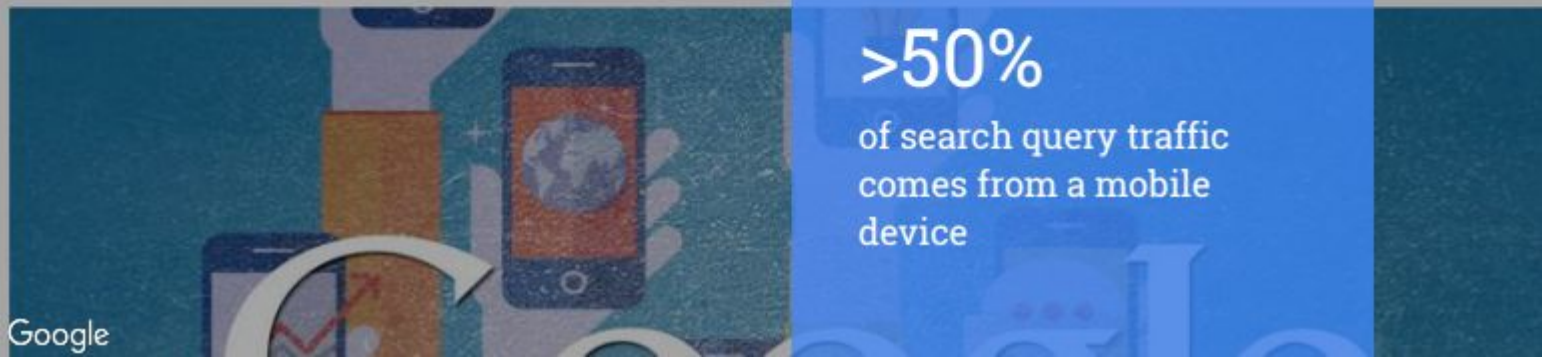
On bus, check email for sales this weekend
5:29pm

Mobile

It's Official: Google Says More Searches Now On Mobile Than On Desktop

Company officially confirms what many have been anticipating for years.

Greg Sterling on May 5, 2015 at 12:34 pm



Smartphone usage

43%
in 2012

80%
in 2016