

Google™



# Optimize every bit of your site serving and web pages with Page Speed

Bryan McQuade and Richard Rabbat  
May 20, 2010

**View live notes and ask questions about  
this session on Google Wave**

<http://bit.ly/io-speed>

# What you will get from this talk

- How performance affects your site
- Become familiar with Page Speed
- Learn about 4 new product features
  - Page Speed exports
  - The Page Speed SDK
  - The Apache module
  - Page Speed for ads, analytics

# What you will get from this talk

- How performance affects your site
- Become familiar with Page Speed
- Learn about 4 new product features
  - Page Speed exports
  - The Page Speed SDK
  - The Apache module
  - Page Speed for ads, analytics



# Web Performance 101

# Web Performance 101

## Why speed should matter to you

- Speed = eyeballs
- Google: 400 ms latency increase → 0.6% search decrease
- Yahoo!: 400 ms latency increase → 5-9% traffic decrease
- Shopzilla: 5 sec latency decrease → 12% revenue increase
  - But also decrease in 50% hardware costs

Source: O'Reilly Velocity Conference, May 2009

# Web performance 101

## Building blocks



- Processing time
- Bandwidth
- Round-trip time
- Parse
- Resource fetches
- Layout and Render
- JavaScript

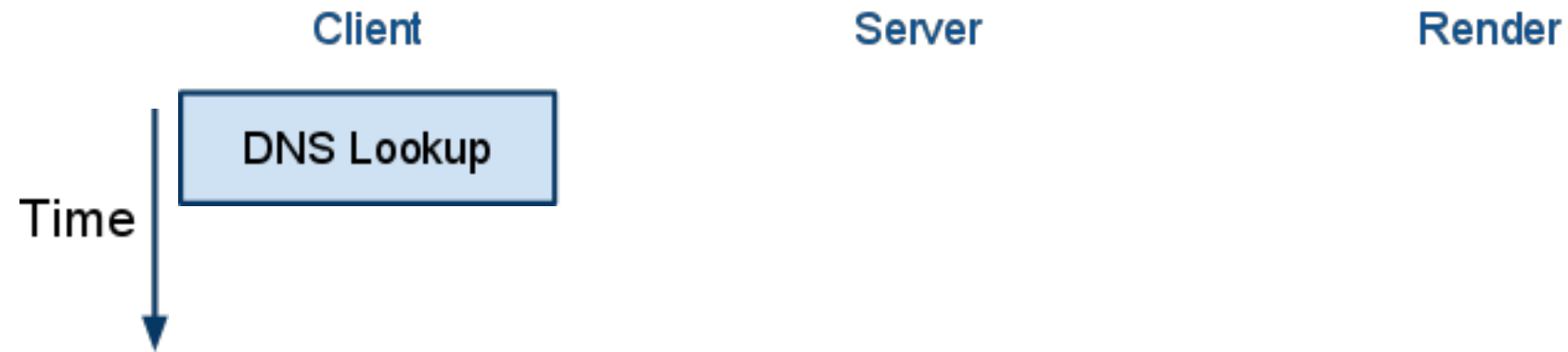


# Web Performance 101

## Example page load

# Web performance 101

## Example page load



# Web performance 101

## Example page load

Client

Server

Render

DNS Lookup

TCP Connect

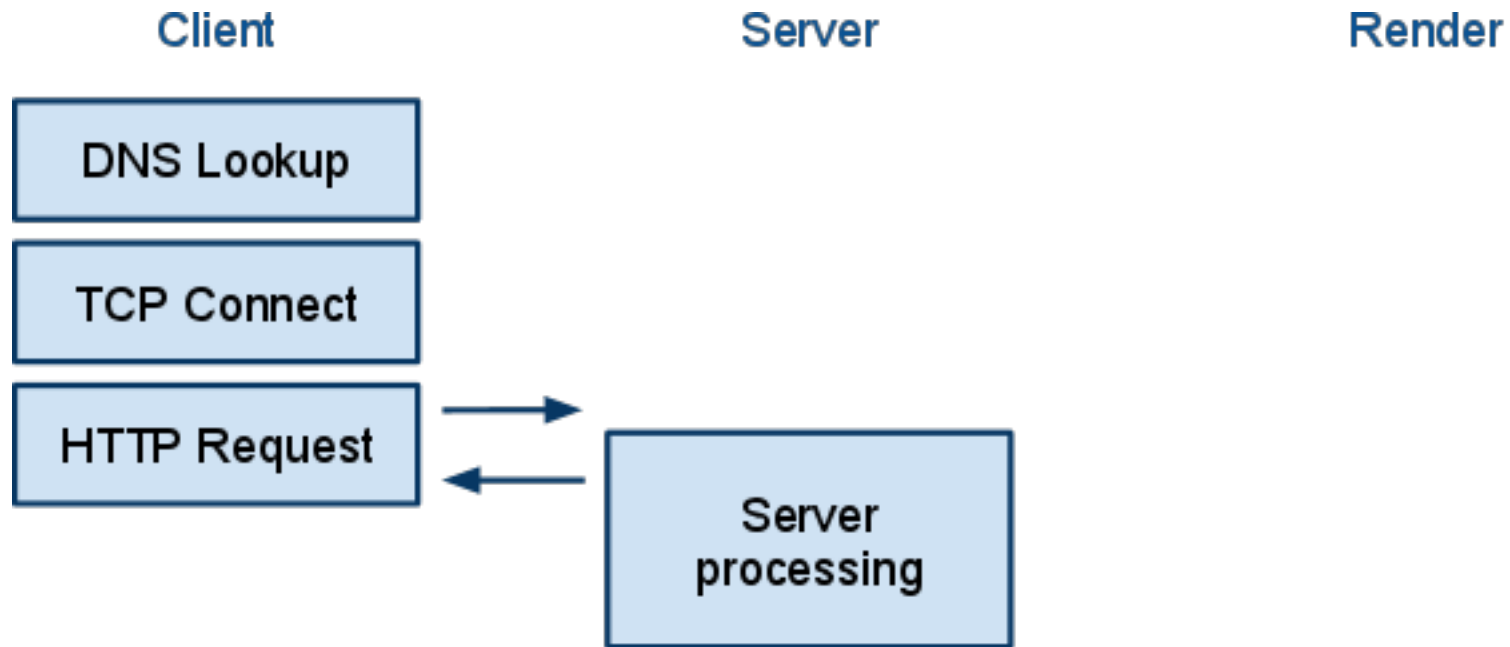
# Web performance 101

## Example page load



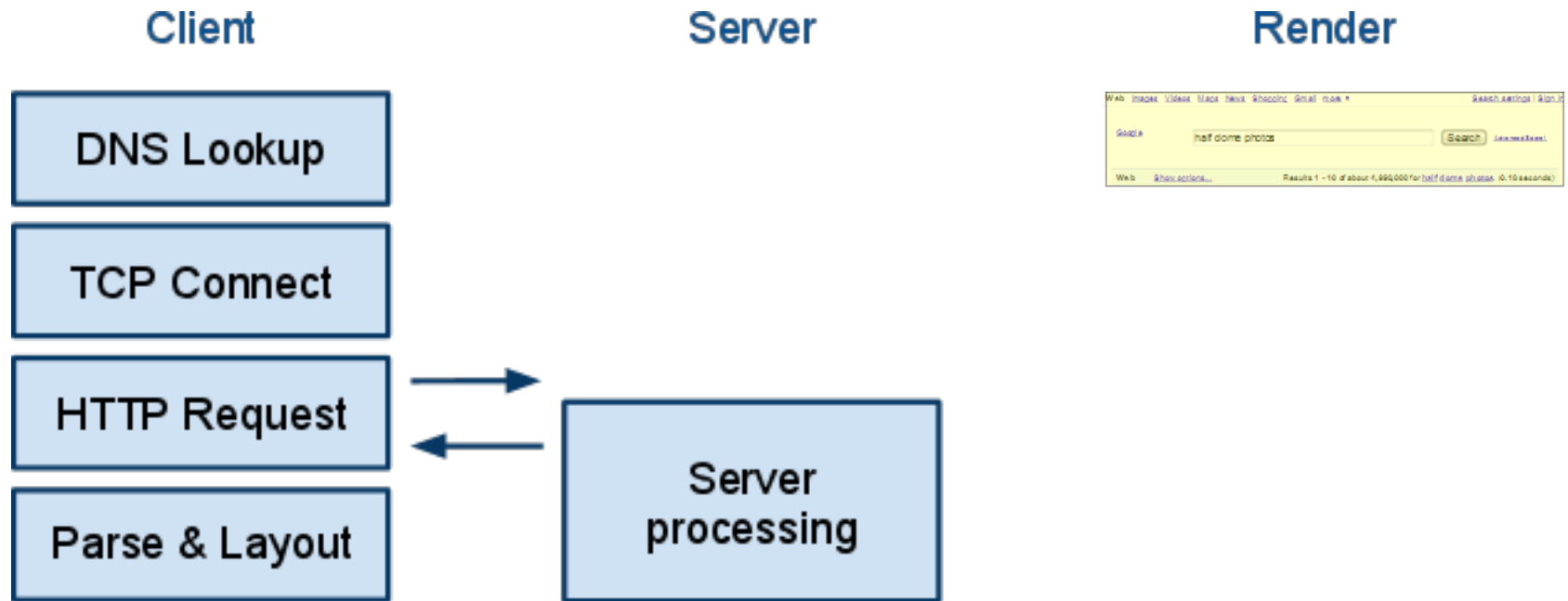
# Web performance 101

## Example page load



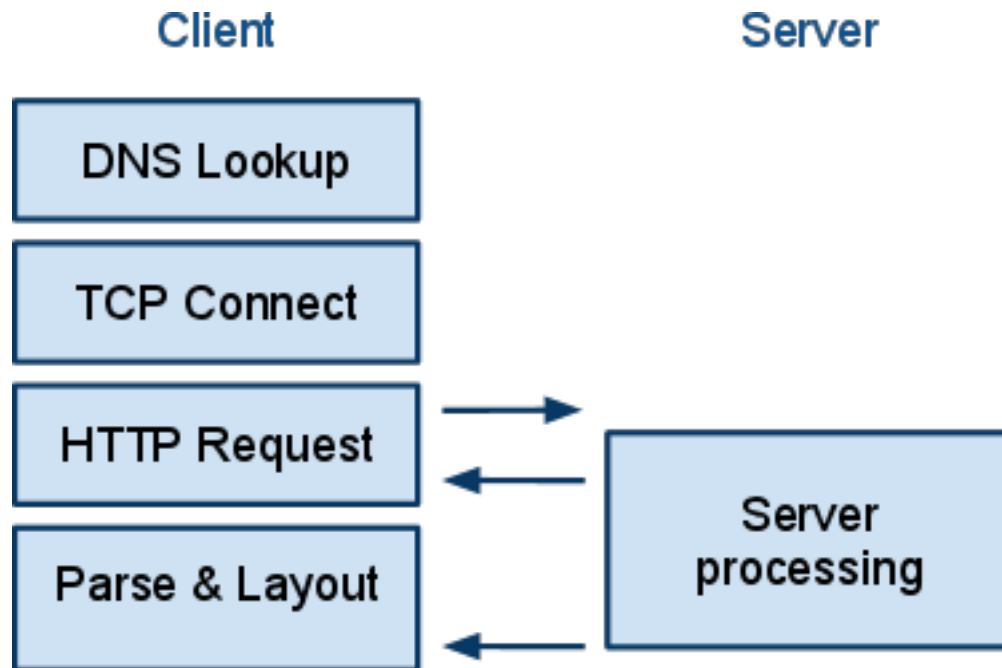
# Web performance 101

## Example page load

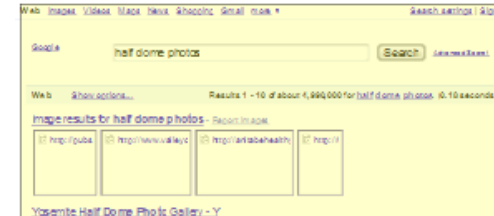


# Web performance 101

## Example page load

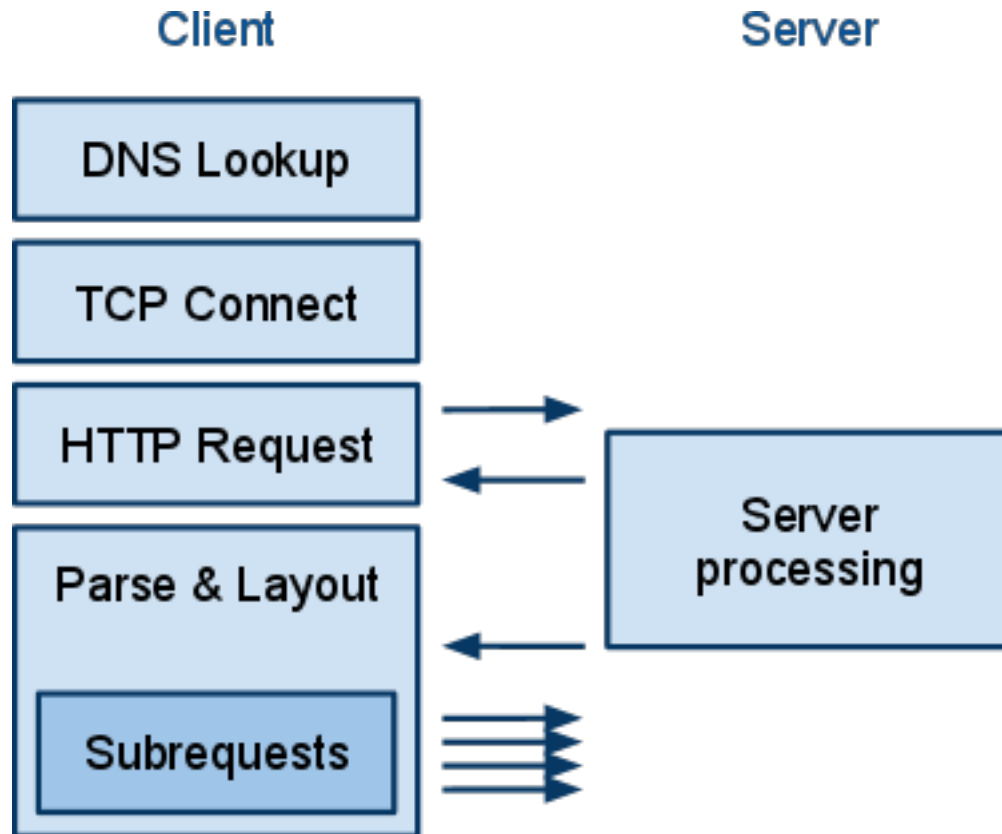


## Render



# Web performance 101

## Example page load



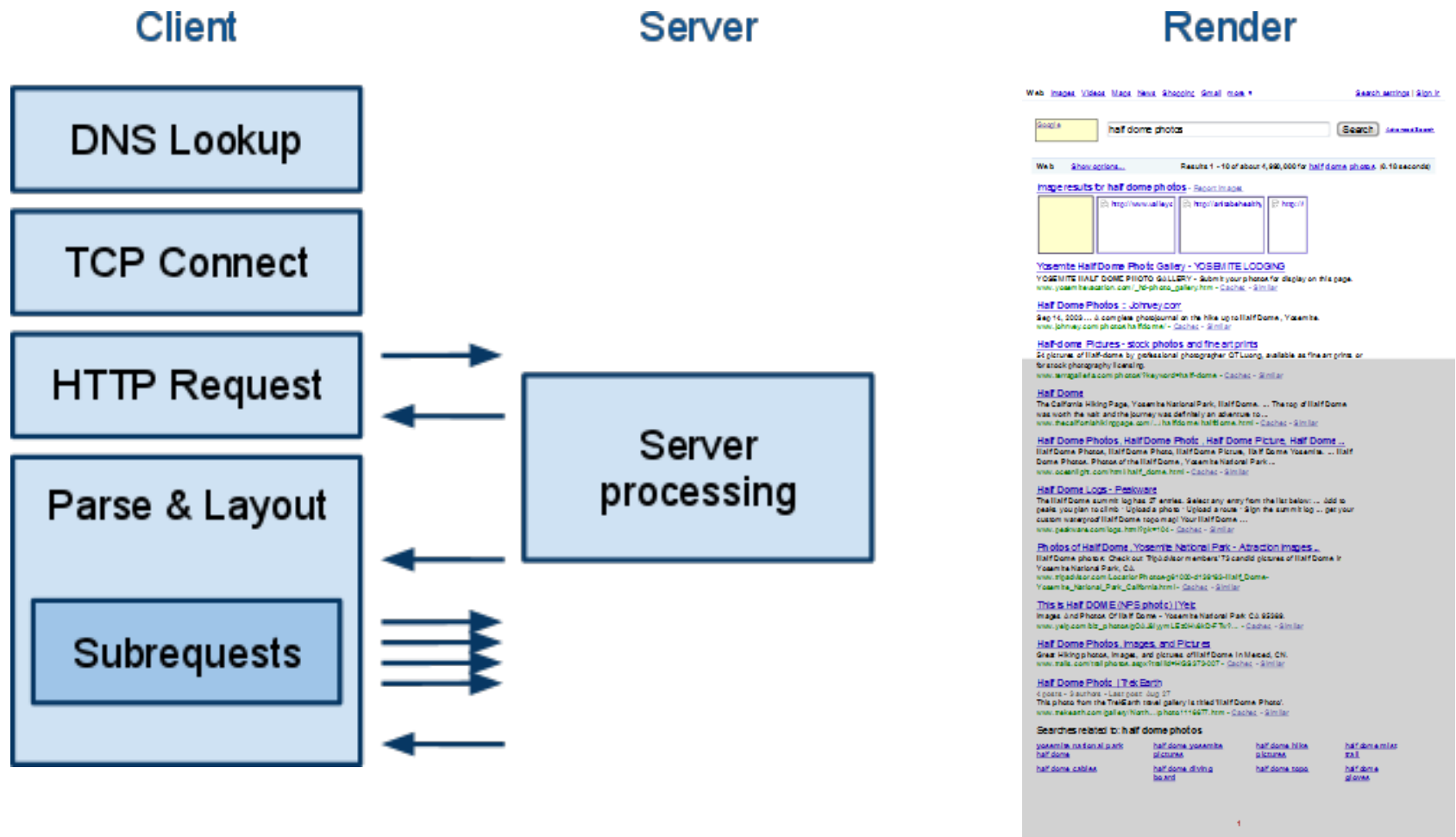
## Render





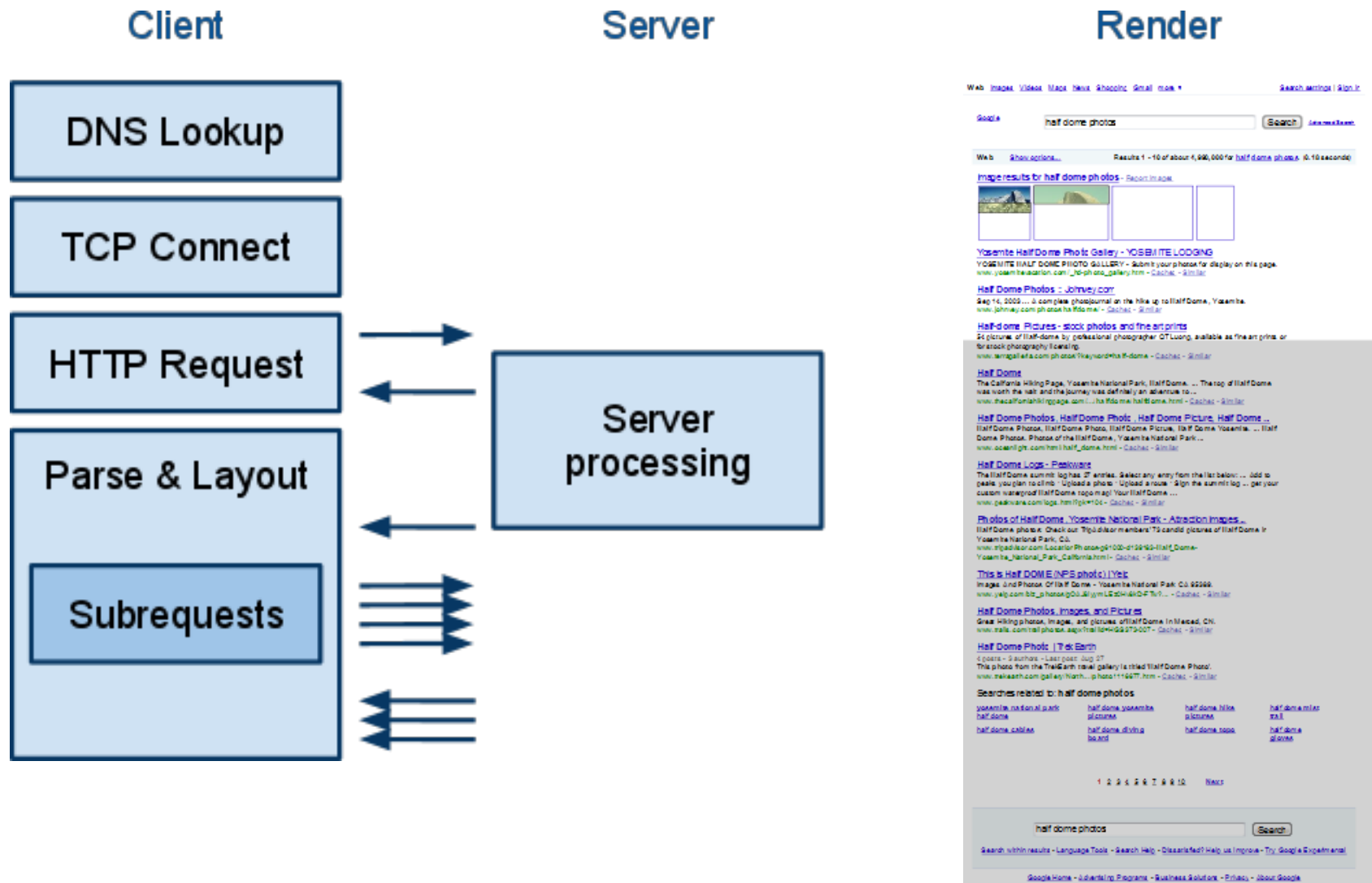
# Web performance 101

## Example page load



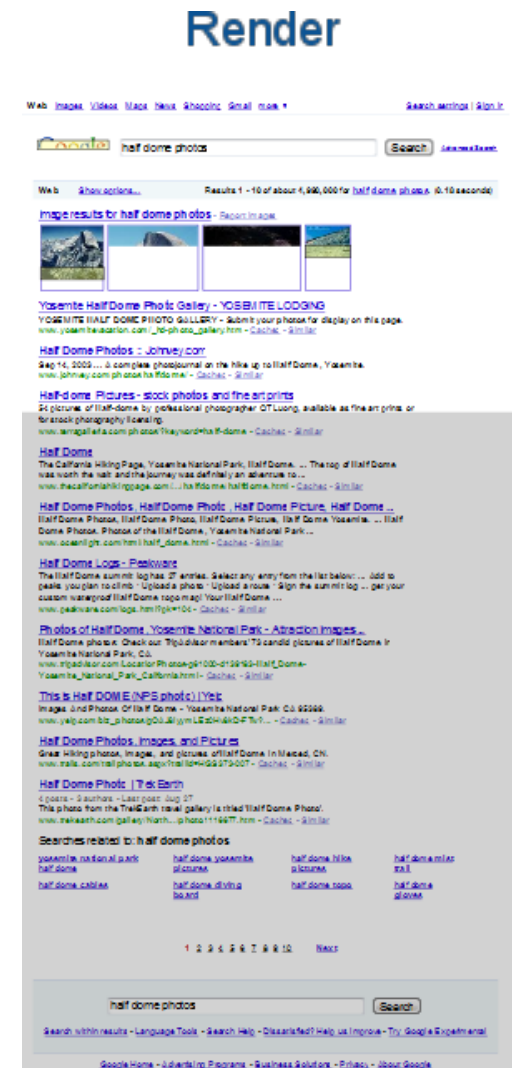
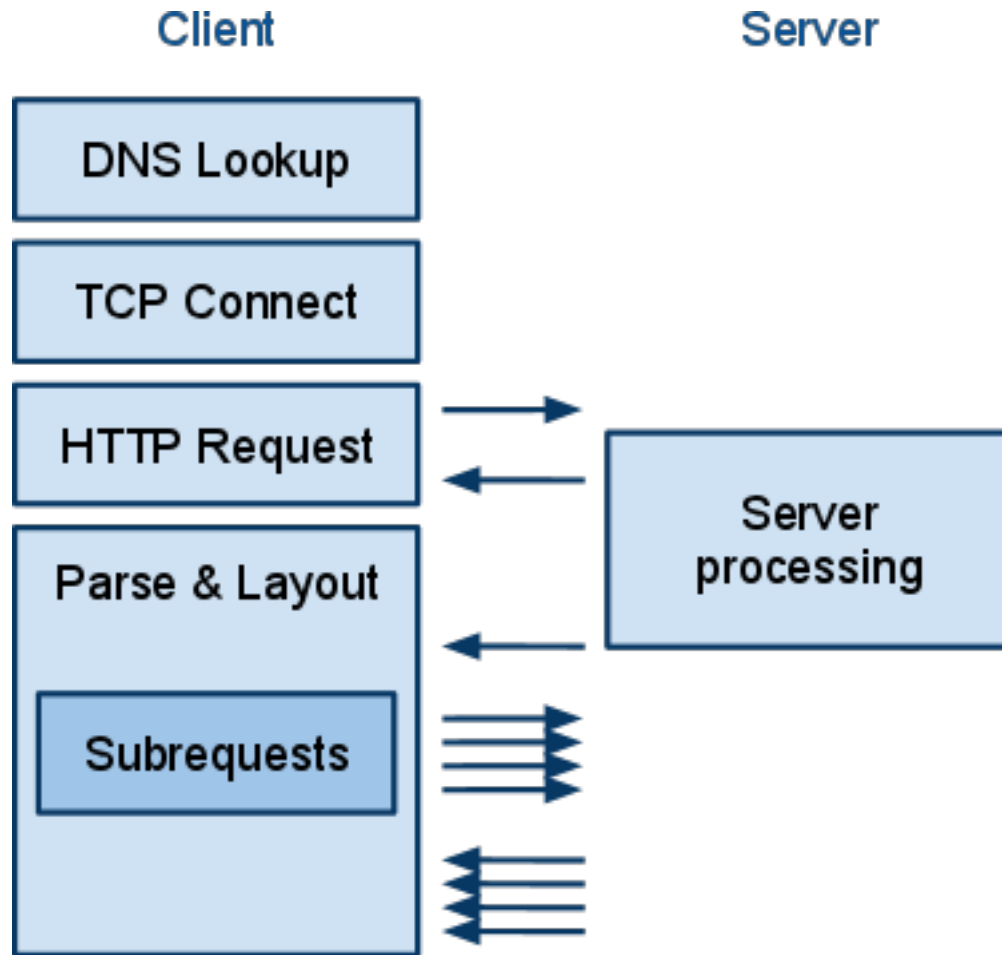
# Web performance 101

## Example page load



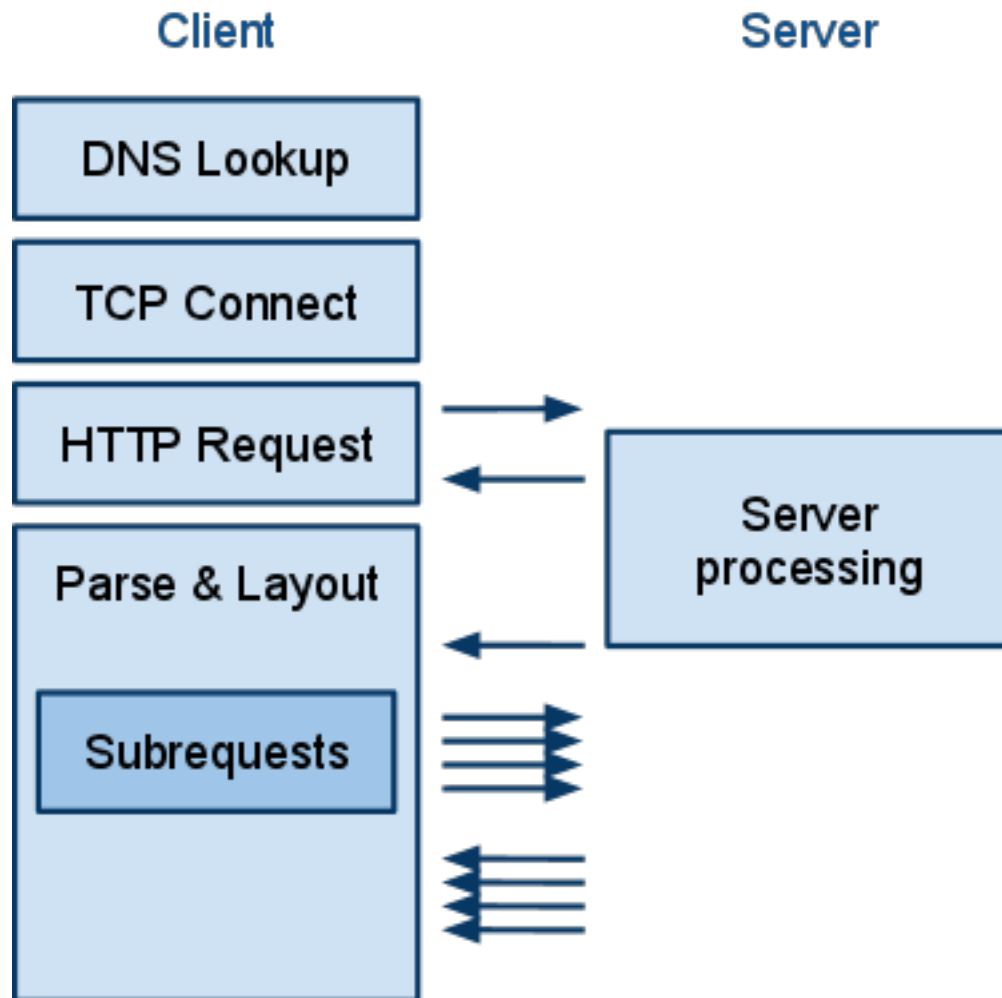
# Web performance 101

## Example page load



# Web performance 101

## Example page load



# Web performance 101

## Follow these three speed guidelines

- Serve fewer bytes
  - Compress at serving
  - Optimize images
  - Minify HTML, JS, CSS
  - Cache aggressively: fastest serving is when you don't have to
- Parallelize resource downloads
  - Optimize the order of styles and scripts (more later)
- Promote modern browsers

# The Page Speed tool (*demo time*)

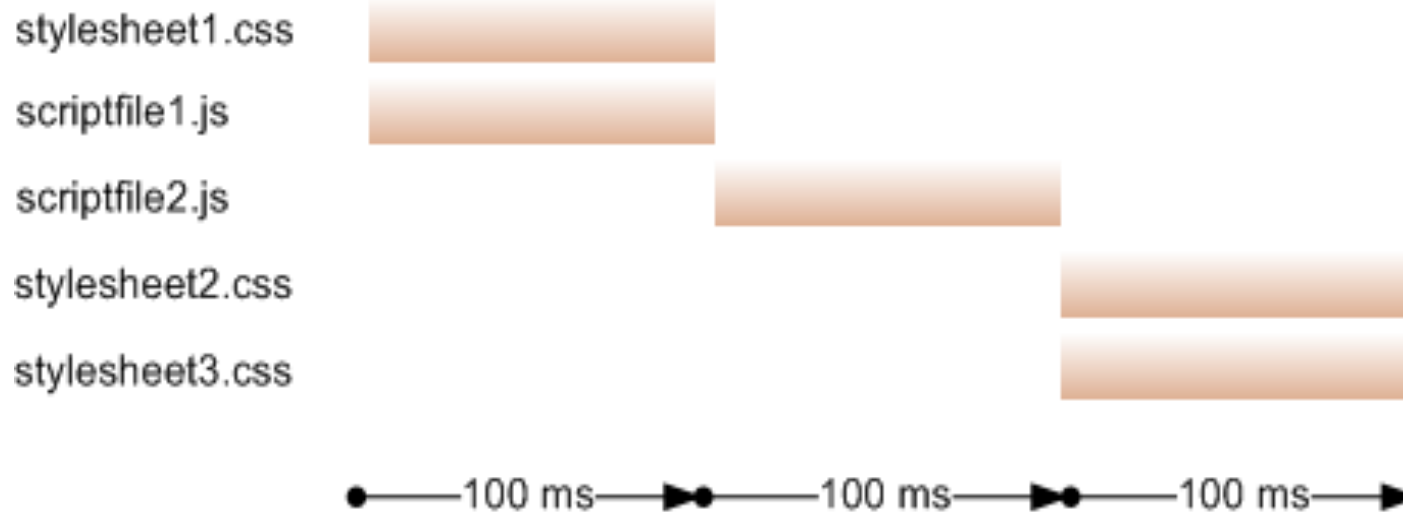
1 Million active users, join the fun!

# Why speed-minded development matters

# Why speed-minded development matters

## Random order of styles and scripts

```
<head>  
<link rel="stylesheet" type="text/css" href="stylesheet1.css" />  
<script type="text/javascript" src="scriptfile1.js" />  
<script type="text/javascript" src="scriptfile2.js" />  
<link rel="stylesheet" type="text/css" href="stylesheet2.css" />  
<link rel="stylesheet" type="text/css" href="stylesheet3.css" />  
</head>
```

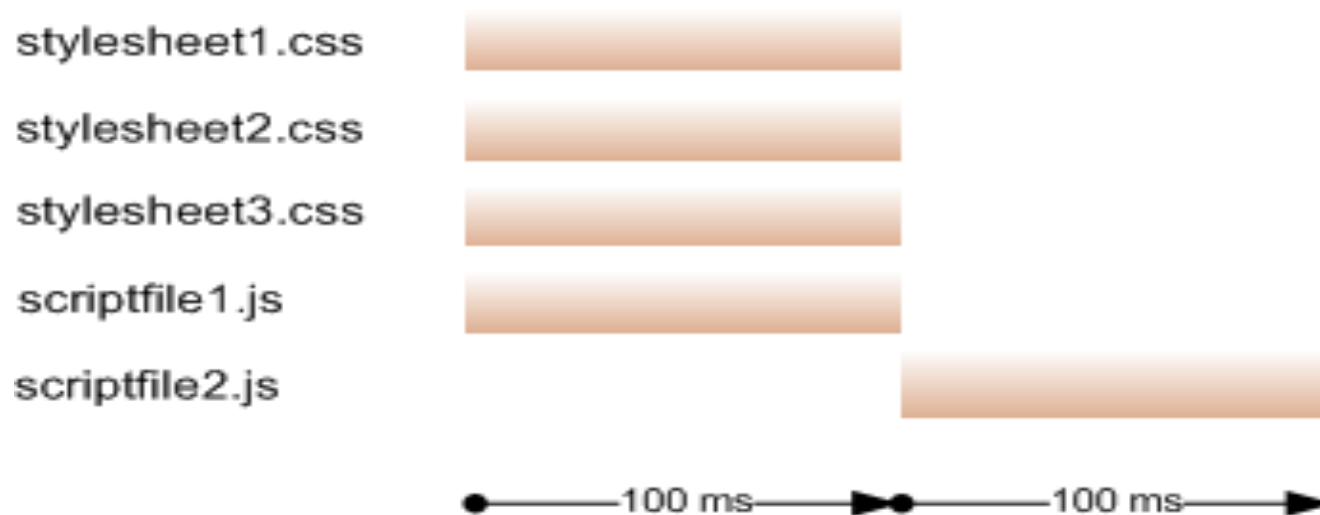




# Why speed-minded development matters

## Put styles before scripts

```
<head>  
<link rel="stylesheet" type="text/css" href="stylesheet1.css" />  
<link rel="stylesheet" type="text/css" href="stylesheet2.css" />  
<link rel="stylesheet" type="text/css" href="stylesheet3.css" />  
<script type="text/javascript" src="scriptfile1.js" />  
<script type="text/javascript" src="scriptfile2.js" />  
</head>
```



# New and improved optimization rules

Be mindful of increasingly important deployments

	Mobile	Infrastructure	Rich web apps	Generally applicable
Minimize request size	✓			
<b>Specify cache validator</b>				✓
Specify character set early			✓	
<b>Minimize DNS lookups</b>		✓		

# New features: the beacon, export to ShowSlow

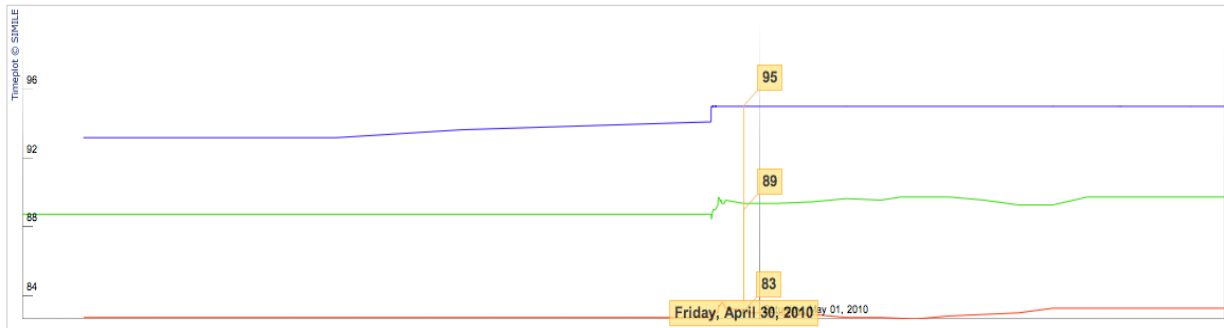
The screenshot shows the Performance tool interface. At the top, there are tabs for 'Performance', 'Resources', 'Export', 'Filter', and 'Help'. Below the tabs, the 'Page Speed Score' is displayed as 87/100 with a yellow warning triangle. A dropdown menu is open under the 'Export' tab, showing two options: 'Write Results in JSON Format' and 'Send Scores to www.showslow.com'. Below the menu, a list of optimization suggestions is shown, each with an icon and a right-pointing arrow:

- ! Minimize redirects
- ! Specify a Vary: Accept-Encoding header
- ▲ Minify HTML
- ▲ Minimize request size
- ▲ Serve resources from a consistent URL
- ✓ Enable compression
- ✓ Specify a character set early

The screenshot shows the header of the Show Slow website. It features the 'Show Slow' logo in a dark green box. To the right, there are links for 'Sign Up Now!', 'log in', and 'powered by showslow'. A yellow diamond-shaped 'SHOW SLOW' logo is also present. Below the header, a navigation bar contains the following links: 'Last 100 measurements', 'URLs measured', 'Compare rankings', 'Configuring YSlow / Page Speed', and 'Download ShowSlow'.

## URLs measured

YSlow grade	Page Speed score	URL
C (78)	B (86)	<a href="http://www.amazon.com/">http://www.amazon.com/</a>
C (79)	B (81)	<a href="http://www.youtube.com/">http://www.youtube.com/</a>



# Page Speed SDK

# Page Speed SDK

- Initially, Page Speed was implemented in JavaScript, tightly coupled to Firefox
- Ported Page Speed rules to browser independent C++ library
- Recently released as a C++ SDK
- Download: <http://bit.ly/cwx7JX>

Google webmaster tools

code.google.com « Back to Home | 11

[Dashboard](#)

- [+ Site configuration](#)
- [+ Your site on the web](#)
- [+ Diagnostics](#)
- [- Labs](#)

[Fetch as Googlebot](#)

[Sidewiki](#)

**Site performance**

## Site performance

This page shows you performance statistics of your site. You can use this information to improve the speed of your site and create a faster experience for your users. [Learn more](#)

### Page Speed suggestions

These are some example pages from your site and some suggestions on how to optimize them, based on the [Page Speed](#) tool.

**URL**

[/apis/ajaxlanguage/documentation/closureeditortransliteration.html](#)

- Details:** Save up to **1.31 MB**, **148** requests
  - Enable gzip compression
  - Combine external JavaScript
  - Combine external CSS

There are **6** CSS files served from closure-library.googlecode.com. few files as possible:

- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)
- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)
- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)
- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)
- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)
- [http://closure-library.googlecode.com/svn/trunk/closure/goog/](#)

**gomez RECORDER**

Playback Agent: Internet Explorer

REC OPEN SAVE PLAY

Script Navigate Wait

Screen Shot Results **PageSpeed**

Metric	Score
AvoidBadRequests	100
CombineExternalCSS	100
CombineExternalJavaScript	100
EnableGzipCompression	100
LeverageBrowserCaching	100
MinifyCSS	100
MinifyHTML	100
MinifyJavaScript	100
MinimizeDnsLookups	0

**Install the Page Speed browser add-on**

Use Page Speed to evaluate the performance of your pages and improve them. [Learn more](#)

## HAR to Page Speed

Generate Page Speed scores from an HTTP Archive file

HAR file:  No file chosen

### 92 Page Speed

- Minimize DNS lookups
- 92** Specify a cache validator
- 93** Leverage browser caching
- 99** Minify HTML
- 99** Minify JavaScript
- 100** Avoid bad requests
- 100** Combine external CSS
- 100** Combine external JavaScript
- 100** Enable compression
- 100** Minify CSS
- 100** Minimize redirects
- 100** Minimize request size
- 100** Optimize the order of styles and scripts
- 100** Serve resources from a consistent URL
- 100** Specify a character set early

**Page Speed Score: 83/100** ▲

- Leverage browser caching**
- Minimize DNS lookups**
- Specify a Vary: Accept-Encoding header**
- Remove query strings from static resources**
- Serve static content from a cookieless domain**
- Specify a cache validator**
- Remove unused CSS**
- Use efficient CSS selectors**
- Combine external JavaScript**
- Minify HTML**
- Minify JavaScript**
- Optimize images**
- Put CSS in the document head**
- Avoid bad requests**
- Combine external CSS**
- Enable compression**
- Minify CSS**
- Minimize redirects**
- Minimize request size**
- Optimize the order of styles and scripts**

# Page Speed SDK

## Code example

- Choose:
  - an output formatter
  - Page Speed rules
  - source of input data
- Invoke the Engine

```
using namespace pagespeed;
```

```
// Text formatter that writes to stdout  
formatters::TextFormatter f(&std::cout);
```

```
// Choose the core ruleset  
std::vector<Rule*> rules;  
rule_provider::AppendCoreRules(&rules);
```

```
// Use HAR data as input  
const char* har = "{ \"entries\": [...] }";  
PagespeedInput* i = ParseHttpArchive(har);
```

```
// Invoke the engine.  
Engine engine(rules);  
engine.Init();  
engine.ComputeAndFormatResults(*i, &f);
```

```
// Clean up  
delete i;
```

# Page Speed SDK

## Code example

- `har_to_pagespeed` command line tool
- Included in the Page Speed SDK
- Reads a HAR file, emits Page Speed results

```
$ ./har_to_pagespeed code.google.com.har
```

```
_Combine external CSS_ (score=100)
```

```
_Combine external JavaScript_ (score=100)
```

```
_Minimize request size_ (score=100)
```

```
_Leverage browser caching_ (score=13)
```

The following cacheable resources have a short freshness lifetime. Specify an expiration at least one week in the future for the following resources:

```
* http://code.google.com/css/codesite.pack.css (1 hour)
```

```
* http://code.google.com/js/codesite_head.pack.js (1 hour)
```

```
* http://code.google.com/js/codesite_tail.pack.js (1 hour)
```



mod\_pagespeed

# Automatic optimization

## mod\_pagespeed Apache module

- Automatically apply Page Speed optimizations
- Minify HTML, JavaScript, CSS, images
- Content rewriting to reduce serialization
- Portable rewriting engine that can be reused in other environments
- Open source, under development:  
<http://code.google.com/p/page-speed/>

# Automatic optimization examples

```
<html>
<head>
  <title>Example page</title>
  <link rel="stylesheet" href="a.css" type="text/css">
  <link rel="stylesheet" href="b.css" type="text/css">
  <script src="c.js">
  <script src="d.js">
</head>
<body>
<h1>Hello, world!</h1>
</body>
</html>
```

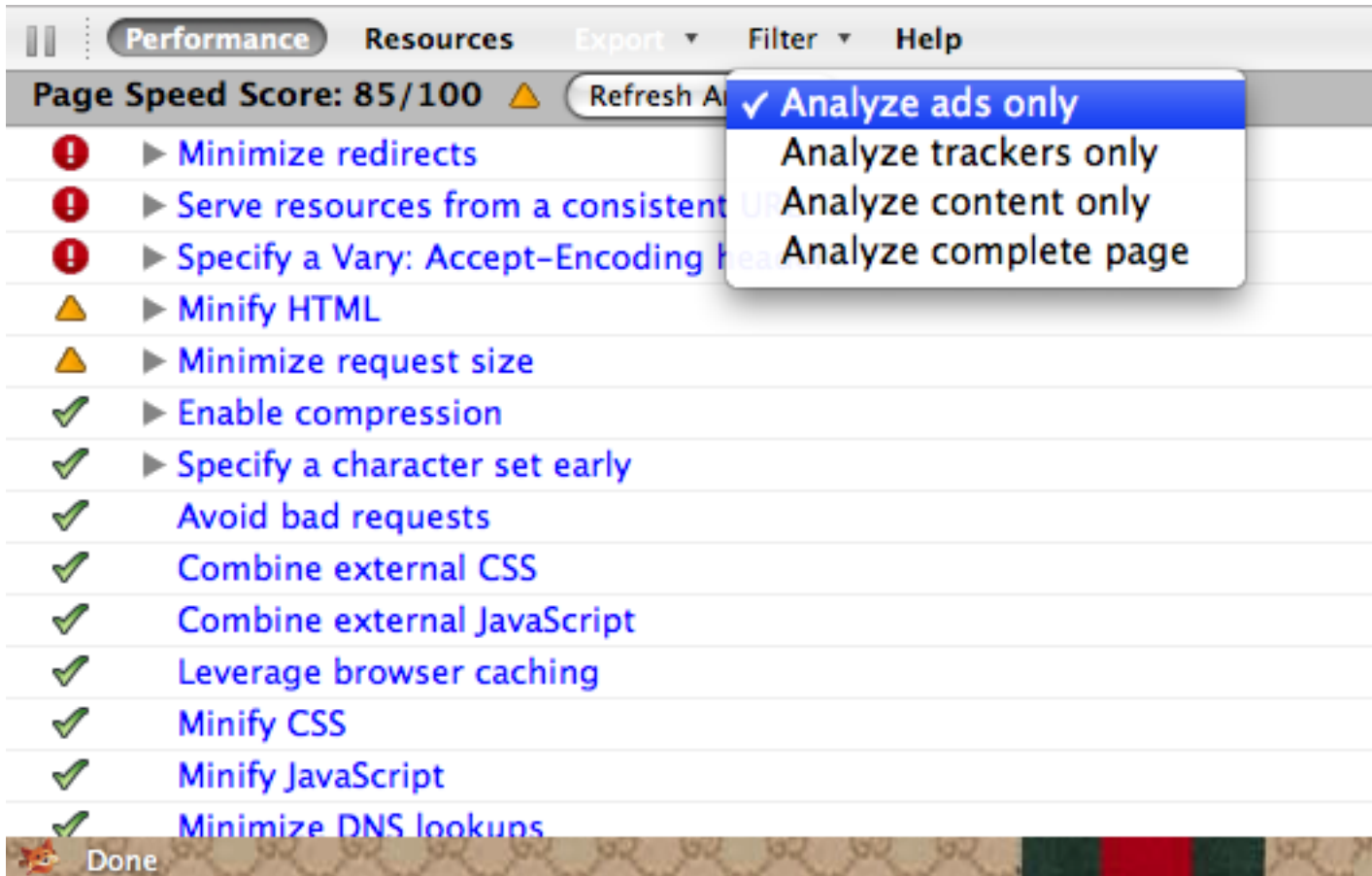
# Automatic optimization examples

```
<html>
<head>
  <title>Example page</title>
  <link rel="stylesheet" href="a.css" type="text/css">
  <link rel="stylesheet" href="b.css" type="text/css">
  <script src="c.js">
  <script src="d.js">
</head>
<body>
<h1>Hello, world!</h1>
</body>
</html>
```

```
<html><head><title>Example page</title>
<link rel="stylesheet" href="/cache/aHR0cDovL2xvY2Fs.css" type="text/css">
<script src="/cache/aG9zdDo5OTk5L3Rlc3QvYS5jc3M.js">
</head><body><h1>Hello, world!</h1></body></html>
```

# Page Speed: looking at ads and trackers

# Page Speed for Ads and Trackers



- Separate optimization recommendations
- Get visibility into performance of different ad and analytics services

Future work

# Possible future rules under evaluation

## Use chunked encoding

- Dynamic responses can be expensive to compute
- But boilerplate header is usually static
- Send header chunk first, then dynamic body later
- Allows browser to begin fetching external JS and CSS sooner
- Big wins for Google properties such as Search and





# Possible future rules under evaluation

## Minimize the size of early loaded resources

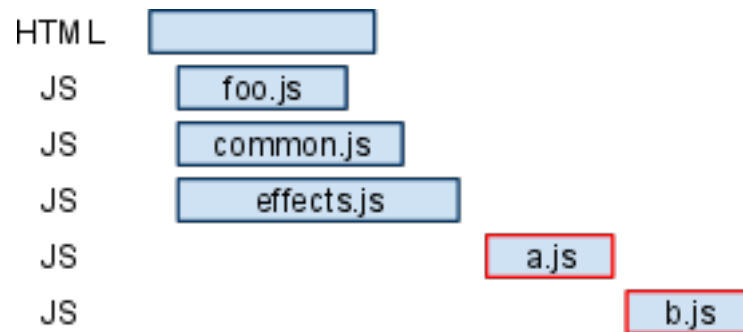
- Modern browsers are much more efficient when fetching resources declared in HTML
- But JavaScript and CSS in the <head> still blocks the renderer!
- Users stare at a blank screen until these resources are finished being fetched, parsed, and executed.
- Load only the minimal JavaScript and CSS needed to display something useful, then load the rest.
- Use Page Speed "Remove unused CSS" and "Defer loading of JavaScript" rules to identify code that can be deferred.

# Possible future rules under evaluation

## Minimize fetches from JavaScript

- Modern browsers use speculative fetching to fetch JavaScript resources in parallel
- But speculative fetchers only parse HTML
- So resources fetched using JavaScript are still serialized.
- Especially important for early loaded JavaScript!

```
<script src="foo.js">  
<script src="common.js">  
<script src="effects.js">  
<script>  
  loader.load("a.js");  
  loader.load("b.js");  
</script>
```



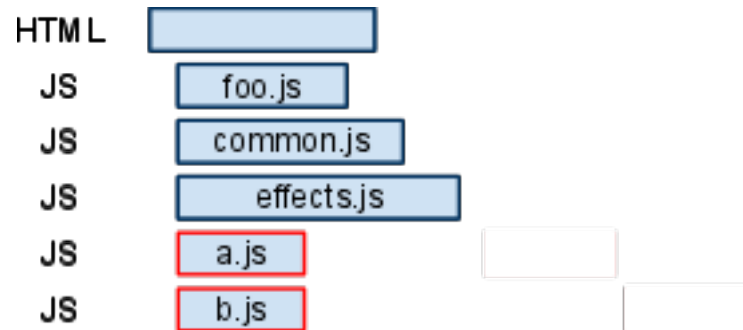
# Possible future rules under evaluation

## Minimize fetches from JavaScript

- Modern browsers use speculative fetching to fetch JavaScript resources in parallel
- But speculative fetchers only parse HTML
- So resources fetched using JavaScript are still serialized.
- Especially important for early loaded JavaScript!

```
<script src="foo.js">  
<script src="common.js">  
<script src="effects.js">
```

```
<script src="a.js">  
<script src="b.js">
```



# Page Speed for Chrome



- Chrome extension
- Integrated with Chrome Developer Tools
- Coming later this year

# Thanks

- Learn and use
  - <http://code.google.com/speed/page-speed/>
- Contribute
  - <http://code.google.com/p/page-speed/>
- Discuss
  - <http://groups.google.com/group/page-speed-discuss>

**View live notes and ask questions about  
this session on Google Wave**

<http://bit.ly/io-speed>

Google™

