Performance Measurements and Optimization

Why You Should Care

Page Load Optimizations

JavaScript Optimizations

CSS Optimizations

User Expectations

47% of consumers expect a web page to load in 2 seconds or less.

User Expectations

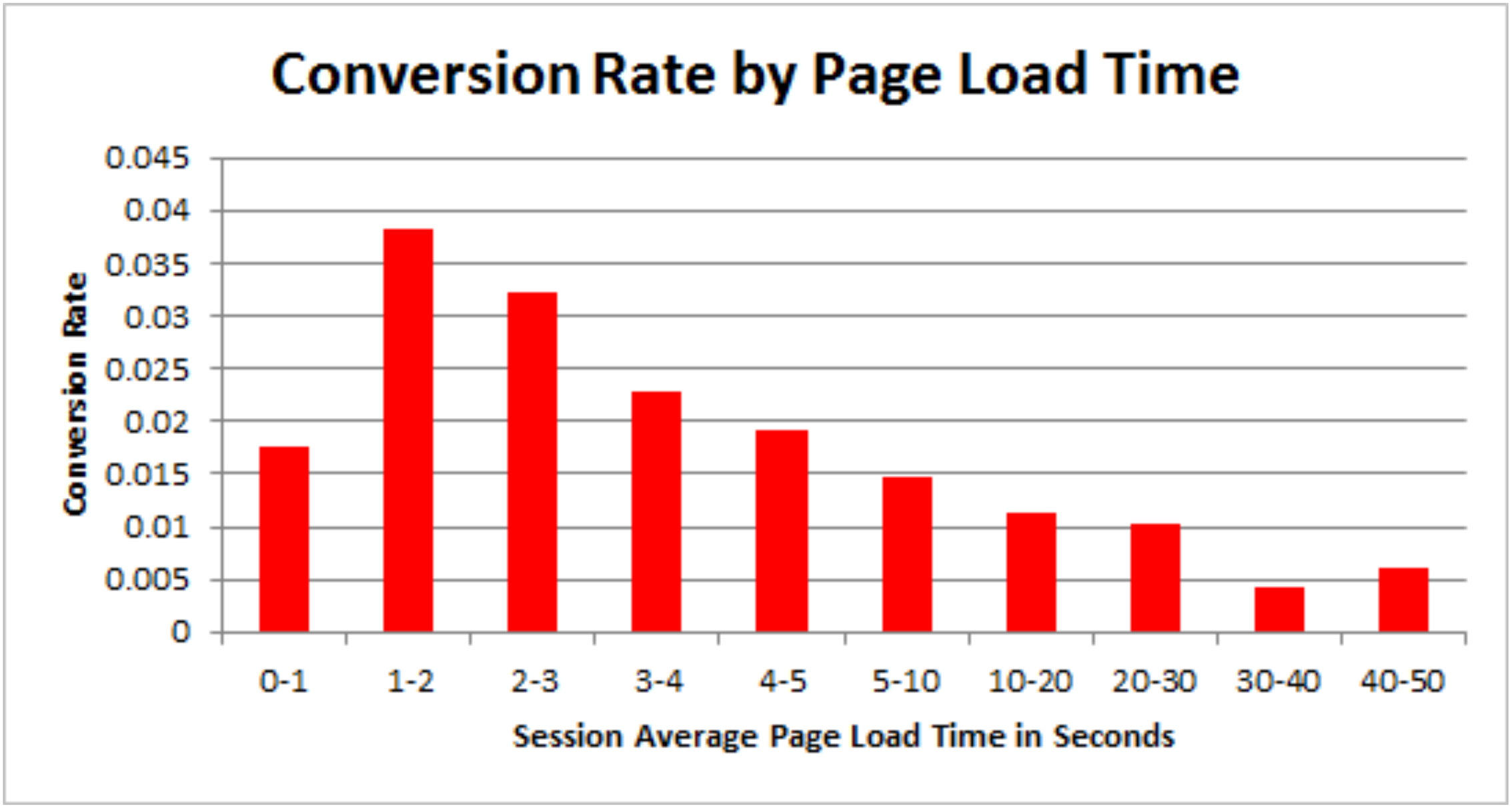
73% of mobile internet users say that they've encountered a website that was too slow to load.

User Expectations

Just One Second Delay In Page-Load Can Cause 7% Loss In Customer Conversions

How Load Time Affects Conversion Rate

- Glasses Direct measured



How Load Time Affects Conversion Rate

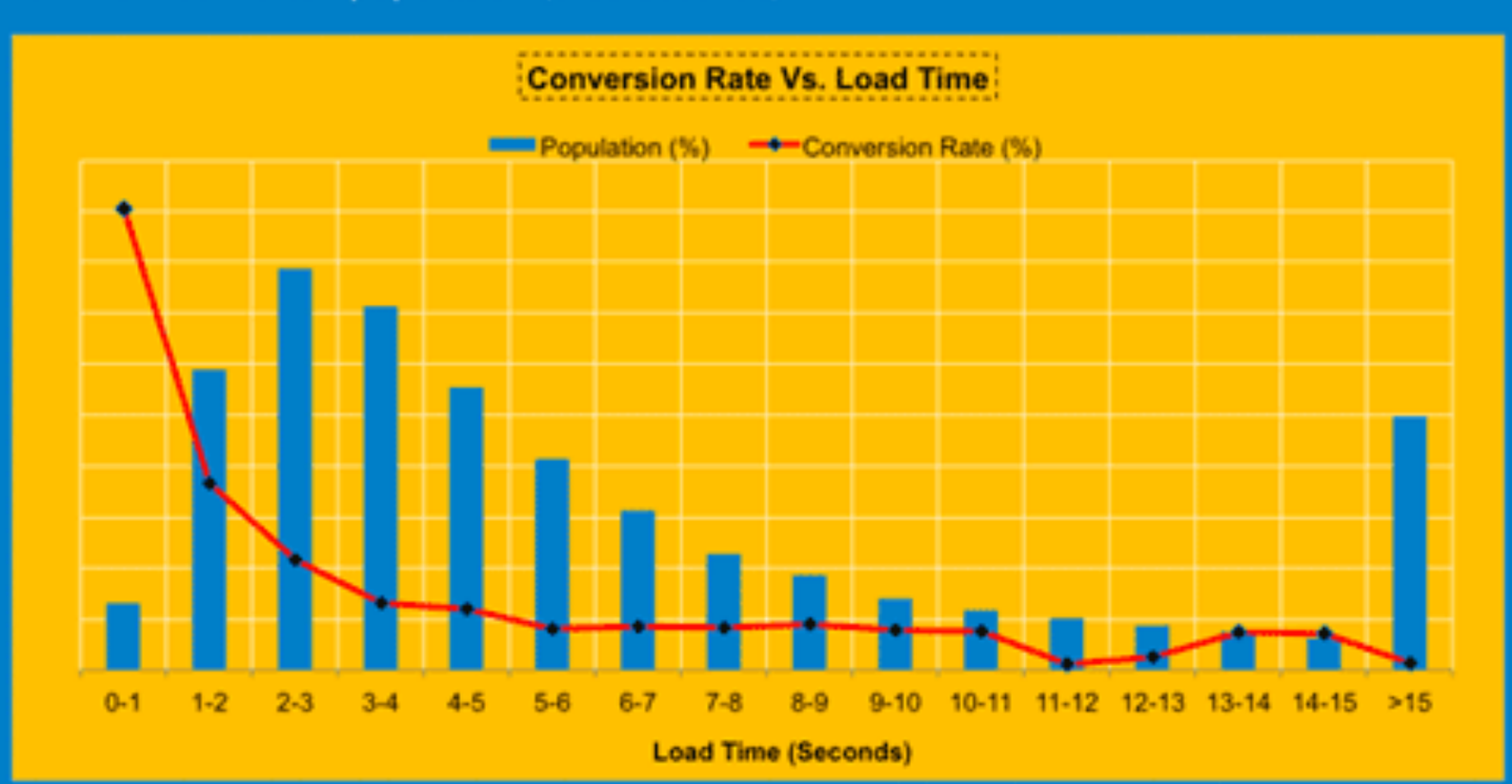
- Wallmart measured

Impact of site performance on overall site conversion rate….

Baseline – 1 in 2 site visits had response time > 4 seconds

* Sharp decline in conversion rate as average site load time increases from 1 to 4 seconds

* Overall average site load time is lower for the converted population (3.22 Seconds) than the non-converted population (6.03 Seconds)



Note: Load Time here is the time taken from head of the page to page ready (T_Page)

How Load Time Affects Conversion Rate

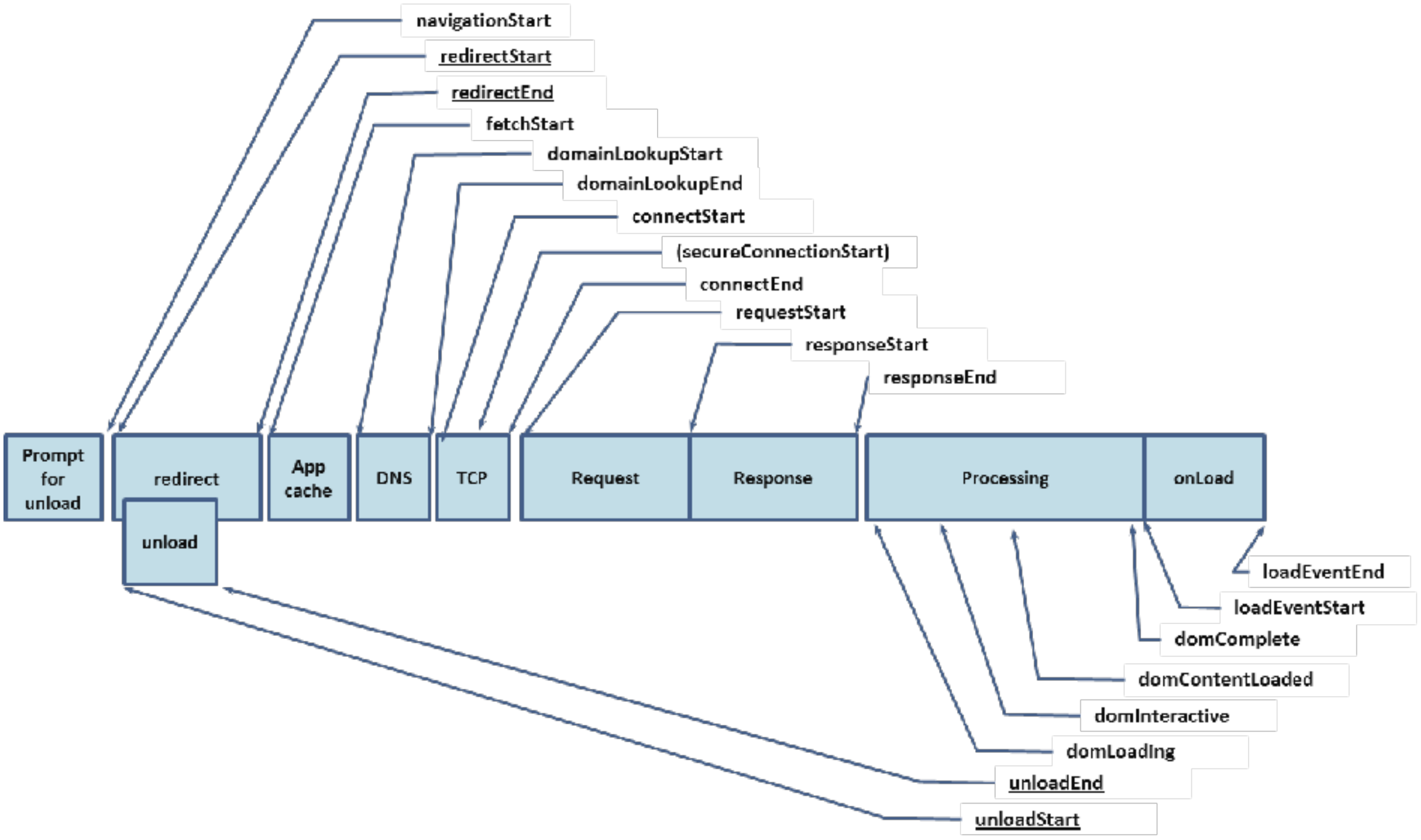
- Google measured

Type Of Delay	Delay (ms)	Duration (weeks)	Impact
Pre Header	50	4	-
Pre Header	100	4	-0.2%
Post Header	200	6	-0.59%
Post Header	400	6	-0.59%

Are you affected?

- Some stats I'd love to see:
    - PLT / Visited Pages
    - PLT / Conversion
    - User timings
- Can get data via timings API or google analytics

Measuring Page Load Times



Web Performance Stats (Retailers)

Website	Load Time (In Seconds)
Barnes & Noble	4.88
Grainger	4.65
Buy.com	5.03
Amazon.com	6.31

Data from: http://www.keynote.com/keynote_competitive_research/performance_indices/mobile/retail/index.html

What Is Page Load Time

Page Load Starts Here

GET data

Here It Is

Demo: Page Load walla.co.il

- View using HAR Viewer
- View using Google Critical Path Explorer

walla.co.il Takeaways

- Look for Critical Rendering Path
- Optimise for faster first paint
- Use less data & smaller request count

Lab: Page Load Time Analysis

- Given the website: https://www.bookdepository.com/

- What's the critical rendering path of the site?

- How would you improve page load time?

Q & A

Page Load Time is affected by:

Page Load Time is affected by:

- Network
- Server Code
- Number Of HTTP Requests
- Size Of HTTP Responses
- How Fast Is The Browser
- Browser Decisions

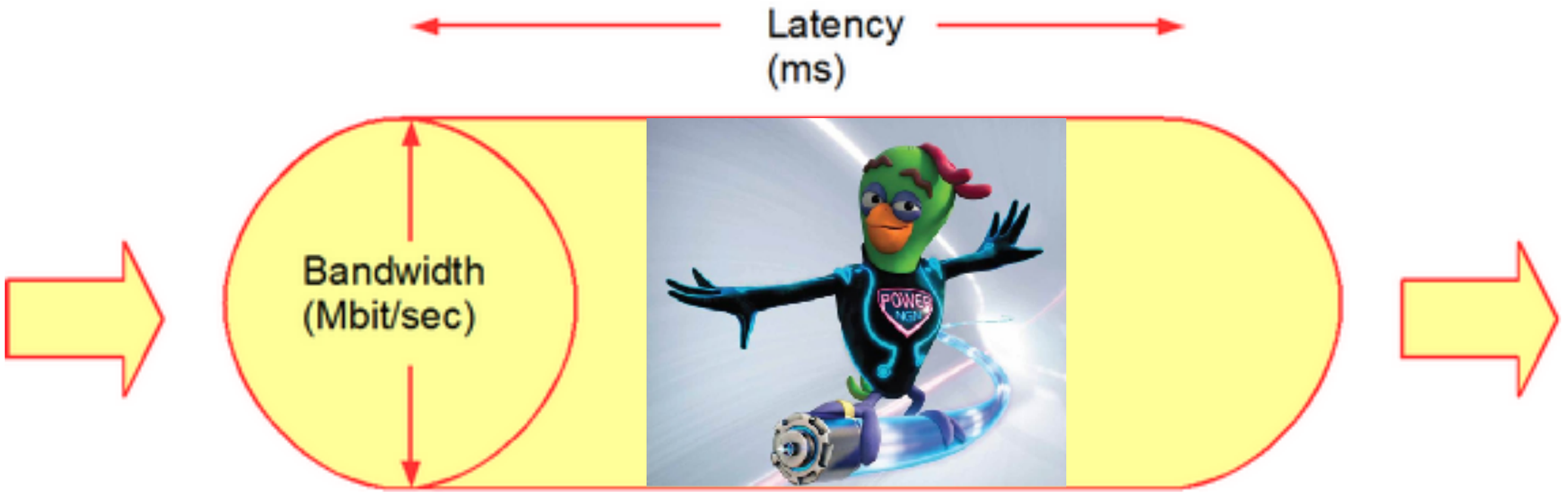
Step-By-Step Page Load

- User types the address
- Browser performs DNS lookup
- Browser opens socket to server
- Browser sends an HTTP request
- Server produces HTTP response (HTML page)
- Server sends back the page
- Browser reads and detects more resources it needs
- (After getting all data) Browser paints the page

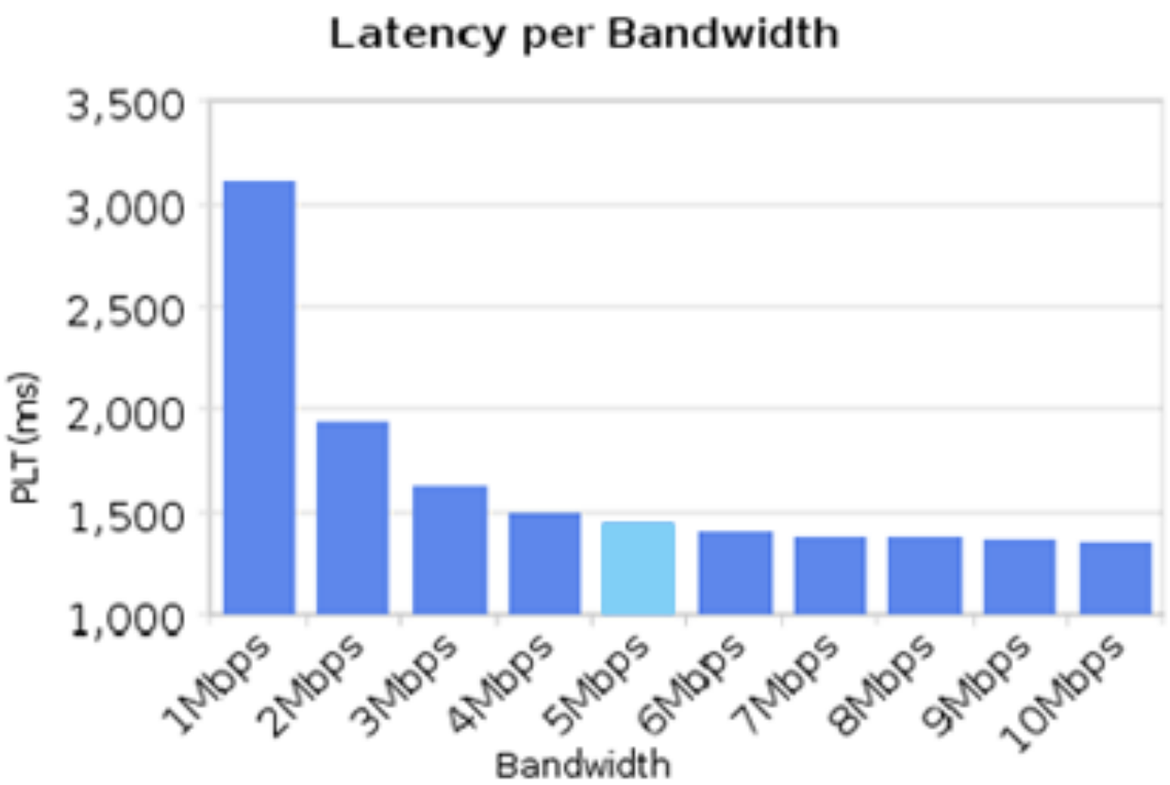
Optimizing Page Load

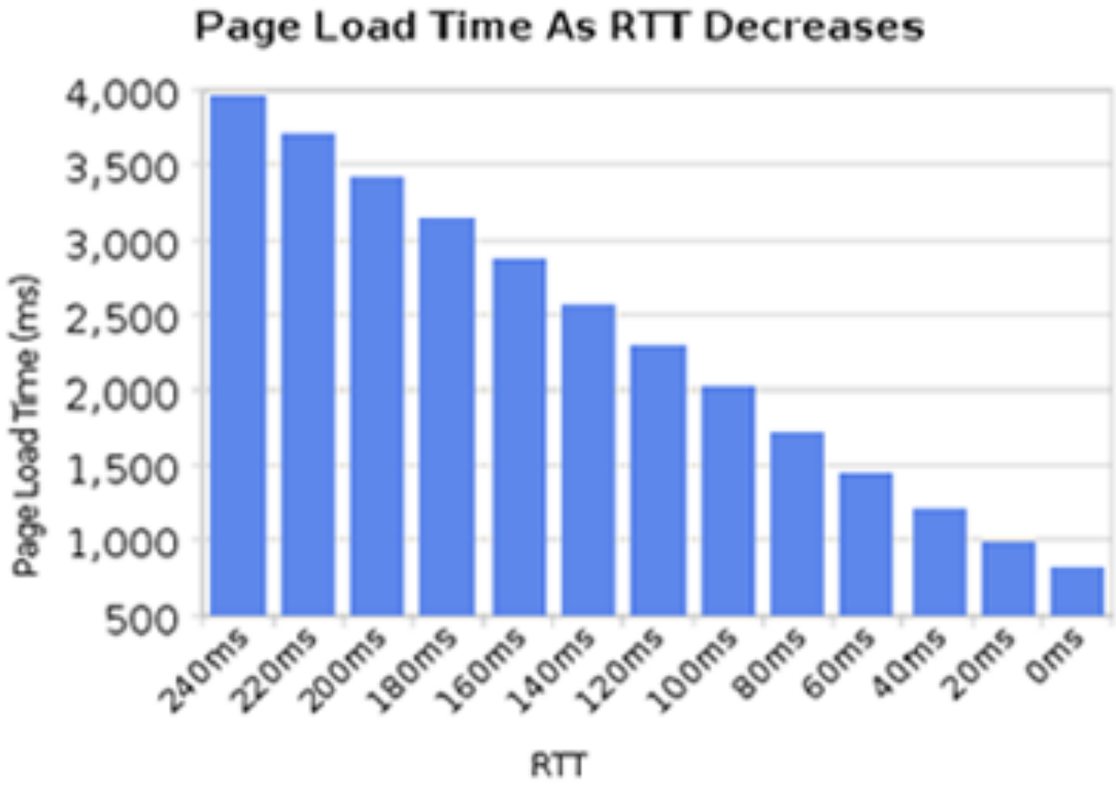
Bandwidth vs. Latency
HTTP Basics
Reduce # Of Requests
Reduce Distance
Reduce Size

Bandwidth Vs. Latency



Bandwidth Vs. Latency





Demo: Testing Latency With httping

Latency And You

- Use a server that’s close to your users
- Make less requests
- Consider using a CDN
- Consider latency when testing

Simple Network Optimizations

HTTP Basics

- Verify using HTTP 1.1 and Keepalive
- Verify using GZip compression
- Can use chrome or:
- `curl -I --compressed http://yoursite.com`

Using CSS Sprites To Reduce Images

- Group all small images into one large image called sprite sheet
- Use CSS code to “cut” the relevant pieces of the sprite sheet

Using CSS Sprites To Reduce Images

- Demo: Use http://www.spritecow.com/ to generate a new sprite sheet
- Full text: http://coding.smashingmagazine.com/2009/04/27/the-mystery-of-css-sprites-techniques-tools-and-tutorials/

Inline Images Using Data URLs

AP+gvaeTAAADLklEQVR4nO2dwZLjMAgFN1vz/7+cuemiCgZEx5lU9zGWLecVIghw/Hg+n/+E4f/dN/
DNKC6I4oIoLojigiguiOKC/Lw68Hg8Tq67h8/rgkFkHYw5vJ/MjU1dcKHIgiguyEu3sCjtj/
cIFiy6zHrcx2S8SmbwzuE33dFyQRQX5NotLIKFECyodWidnIm8e9gQOJw1JhgcnBWM2Sm5Di0XRHFB
Cm6hR+AEdkeRifAz63qf4paagJYLorgguFsoRQI9/5DhFv+g5YIoLkjBLfQWVGk9BruA3sa/
d89TrkPLBVFckGu3MFUC6BFsB6Y+WYx/
Uy0XRHFBHjc24o2HBJk8hpuIL0FxQWb6FjIIgMPTS1Nk6hfB4F6wsaPIgiguyPuihU+L8HsrvRThaLkgigty
7RZKbQb7mKnTS66jt8CDO+yh5YIoLkghWpgKv3dKRcxe71Mv/
DhMTWi5IIoL8tItINZ1wBs2CIdTTFVFd7RcEMUF6eQWev3Dh7uAqX3K4SFzC5+C4oIU+hZ6oX4vnu/
VQXbG26ozky60XBDFBbneRAQcJvNvaUMqxSHBPWfQckEUF6RQieg9crgYTxWWnNJhwjNzaEfLBVFc
kEI702FFr/fkY+n0Ke80VQzVckEUF6TwNyxTVb/
SIQ8zGwGHpRajhZtRXJDOJqL0w801HvTueadXzcyg5YIoLsj1JiKzsjKHenF4L6LYL9j7FuYWPhfFBRkuUA
anZyg5it5ch5WI/RNzC/
eguCCFLsfMmN7CjO7vrE5a2p70vnKAIguiuCDDf9o2If0rTZEpmgT+oddamUHLBVFckM4zEeMNRcF1p
mqFU49dICbVckEUF+RItBBk7TKDF+8sWQaTHp5uI+PHobgg73urVIapZEUpnuH2O1ouiOKC4G+VKqX
vuL6FqSuXog4tF0RxQfC3SgXLp5TiG28qGC+a7Gi5IIoLgr8+pkRp7zDIKGx+/
pMoLshnuYXeHy9MeYPxLmgtF0RxQfC3Si2mfrinkpnBpL2ay46WC6K4IDe8Var37BXXwDD+PPVCywVR
XJA73yr19Wi5IIoLorggiguiuCCKC6K4IIoL8gvjNAP/KJZ4+gAAAABJRU5ErkJggg==

Inline Images Using Data URLs

- A data url is a base64 encoded version of the image bytes

- Data URL basic form:

```
data:[<mediatype>][;base64],<data>
```

- Use server side code or http://websemantics.co.uk/online_tools/image_to_data_uri_convertor/ to generate

- From Unix:

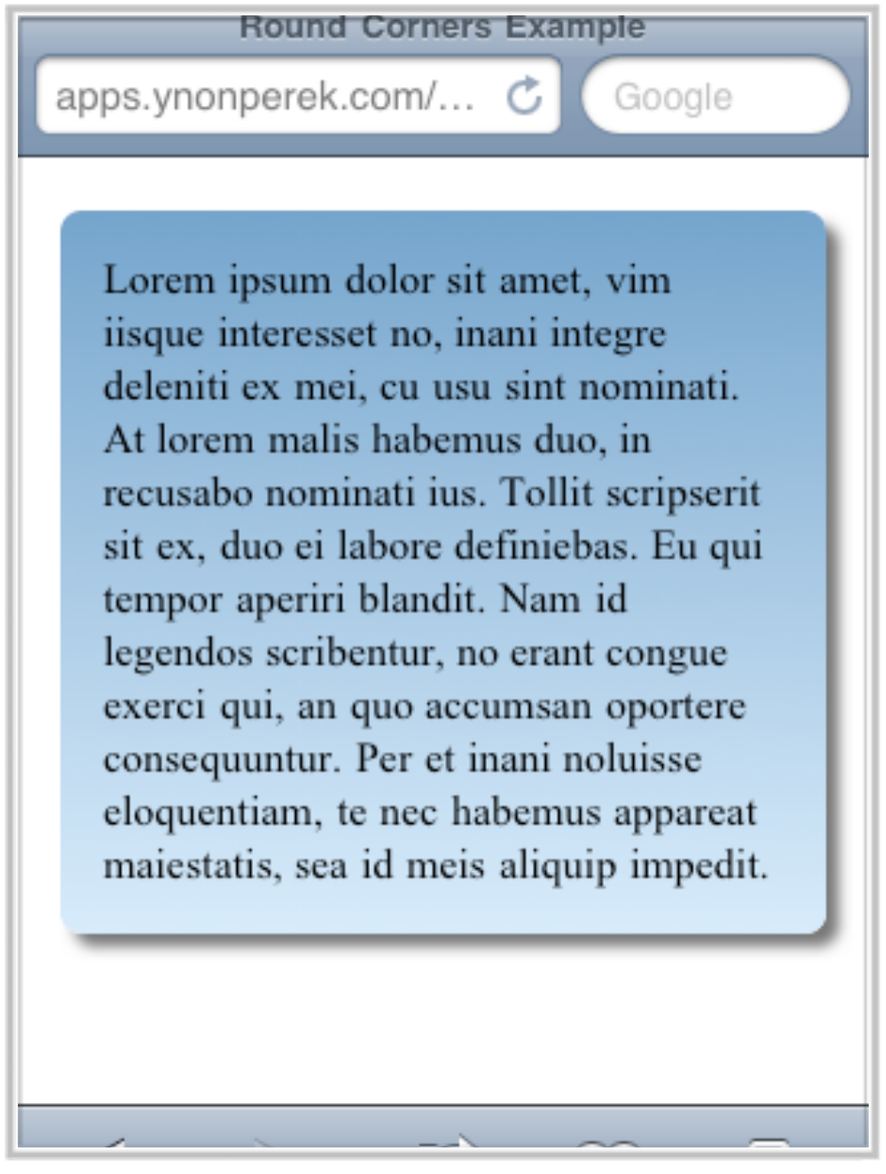
```
echo -n '';base64 image.png
```

Why DataURLs are not awesome

- They make your HTML ugly
- They’re not saved in browser cache
- Image blocks page load
- Larger in size (about 1.333%)

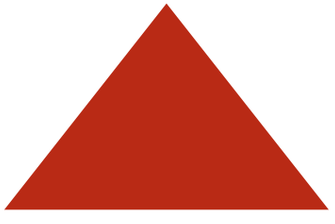
Use CSS3 Instead of Images

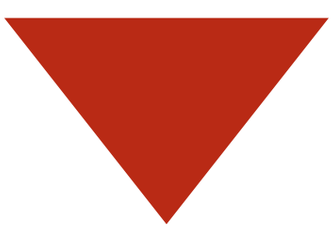
- Replace onions with border-radius
- Use css gradients instead of images
- Use box-shadow instead of images
- Tools:
  - http://css3generator.com/
  - http://www.colorzilla.com/gradient-editor/



Use CSS Shapes Instead Of Images

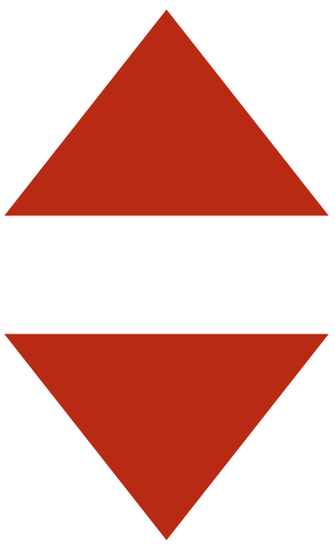
```
<body>
    <div class="shape triangle-up red"></div>
    <p></p>
    <div class="shape triangle-down red"></div>
</body>
```





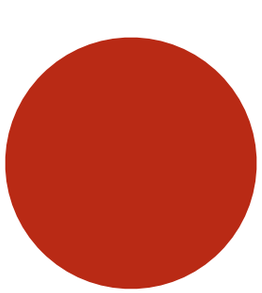
Use CSS Shapes Instead Of Images

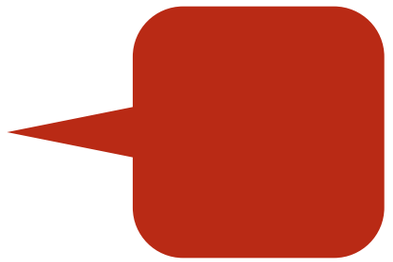
```css
.shape {
  border: 30px solid;
  width:0; height: 0;
}

.red { border-color: red; }

.triangle-up {
  border-left-color: transparent;
  border-top-color: transparent;
  border-right-color: transparent;
}

.triangle-down {
  border-left-color: transparent;
  border-bottom-color: transparent;
  border-right-color: transparent;
}
```



More CSS Shapes

http://css-tricks.com/examples/ShapesOfCSS/



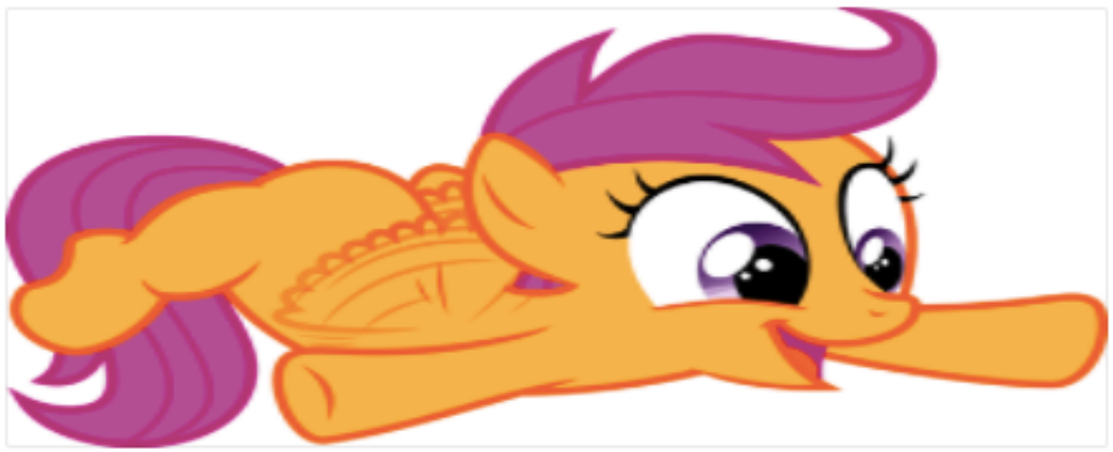


Old Browsers Fallback

- Old browsers may not support CSS shapes
- Use Modernizr to detect and fallback

```
.no-js .glossy,
.no-cssgradients .glossy {
    background: url("images/glossybutton.png");
}

.cssgradients .glossy {
    background-image: linear-gradient(top, #555, #333);
}
```

If possible, use SVG



Sending the right image size

Sending the right image size

- Use Media Queries to set correct background image
- Demo: http://pastie.org/7909992
- Use WUFRL to send only relevant content for mobile

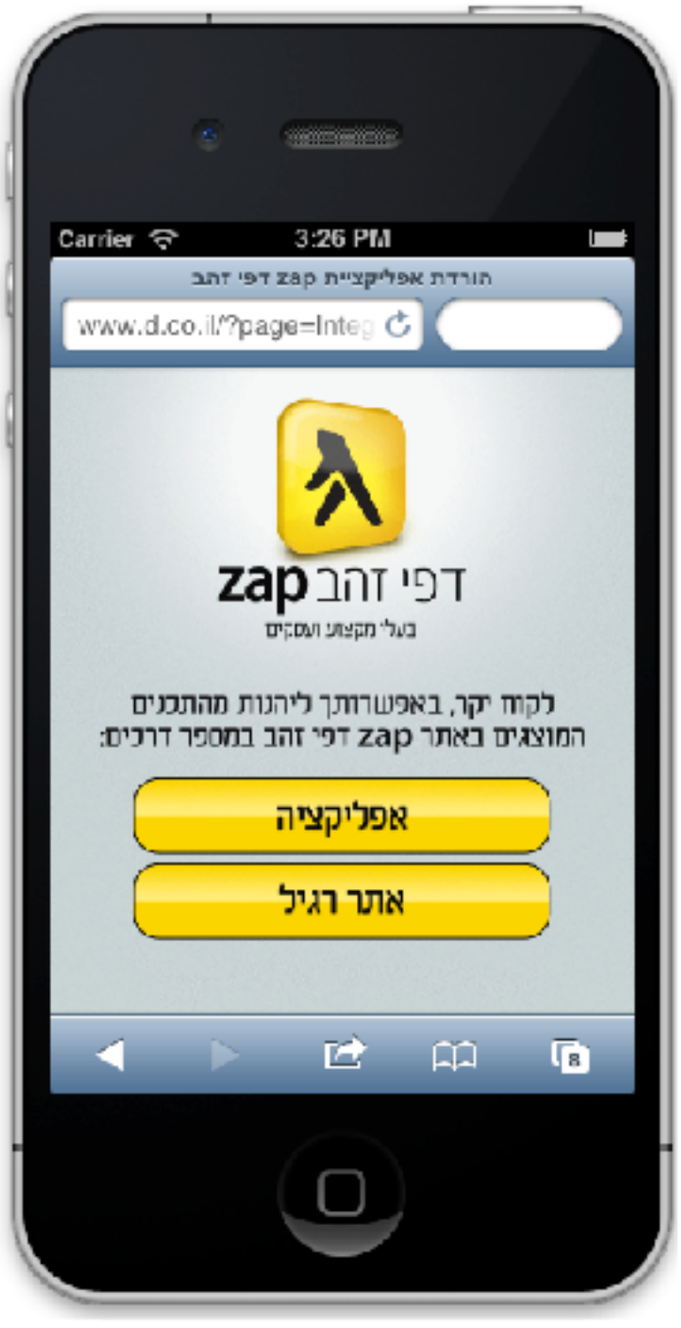
Combine Scripts And Stylesheets

- Combine multiple JS into a single script
- Combine multiple CSS files into a single stylesheet
- Ideal: Use one stylesheet and one script per page

- Tools:
  - http://yui.github.com/yuicompressor/
  - http://yeoman.io/

Avoid Redirects At All Cost

- Redirects add another request but with no value
- Worst kind: Mobile Redirects
- Demo: d.co.il



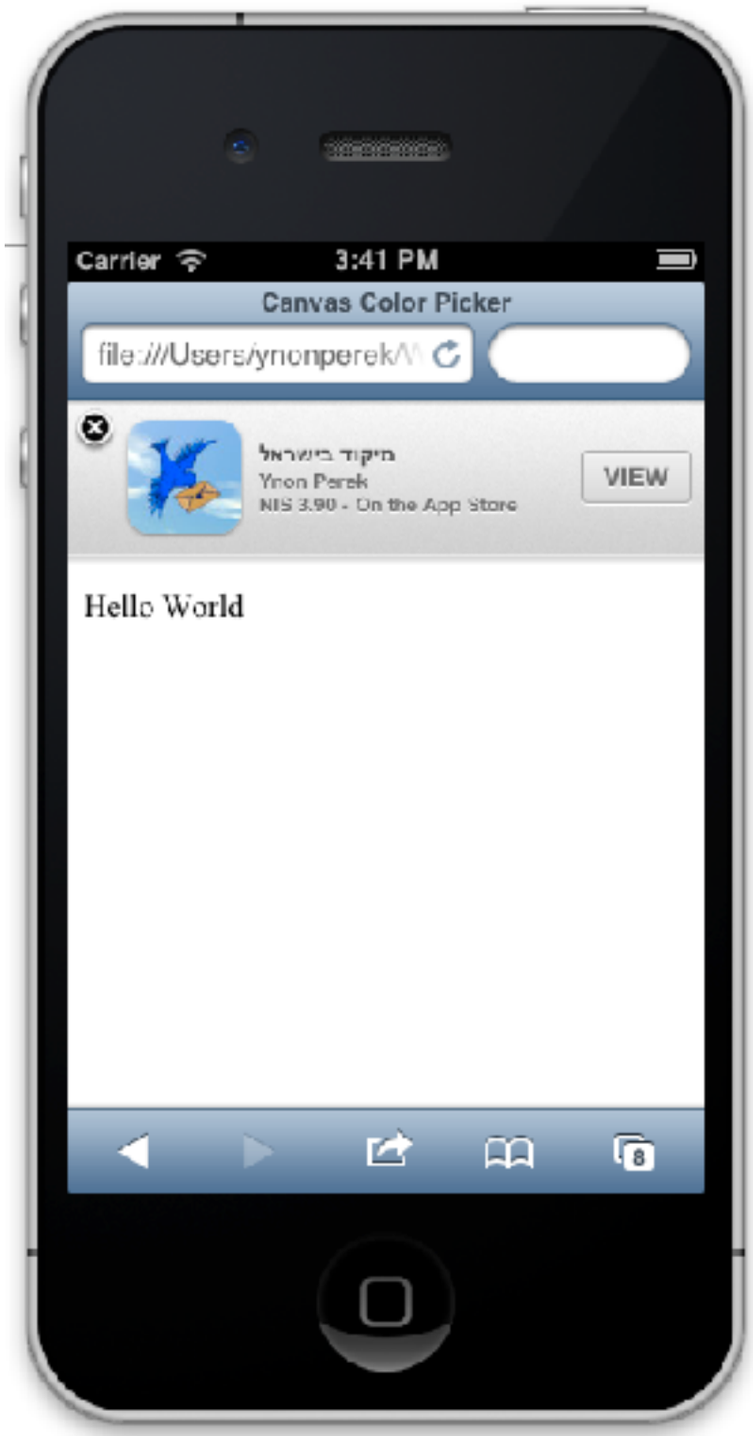
Avoid Redirects At All Cost

- Better Way: Offer users to download the app from your web page

- iOS 6 has the option built-in (smart app banners):

```
<meta name="apple-itunes-app" content="app-id=366977202">
```

- Or use the jQuery Plugin: http://jasny.github.com/jquery.smartbanner/#android



Use Cache To Reduce # Of Requests

- No Request is better than fast request
- Browser Cache Is Your Friend

File size doesn't matter

ynonperek.com	GET	200 OK	text/html	Other	5.06KB 17.11KB	117ms 76ms	
css_obm0lsQQ7A7WCCIMgxLho6mI_kBNgznNUV /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:19 Parser	(from cache)	7ms 7ms	
css_ytP0uhATbbzOz8lVlYkYIlyt_Fwn5atRE7uzgZj /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:20 Parser	(from cache)	10ms 10ms	
css_38xr3EmoN3XLqIfnTe6TAIyJuVmrsPZnwmw1 /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:21 Parser	(from cache)	11ms 11ms	
css_8dPLVLq_LOYZ1GUI5CVoQMvd8VlnMeOpsbh /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:22 Parser	(from cache)	12ms 12ms	
css_6j_kM_lFBpjKIMCcWlOeu7BL5nlp_0sIDeauzJy /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:23 Parser	(from cache)	12ms 12ms	
css_c0Ol8l1L8cGJ7KxcUbImLkO_IvlPDoZXuBZMW /sites/default/files/css	GET	200 OK	text/css	ynonperek.com:24 Parser	(from cache)	374ms 374ms	
908f05d5-fad9-4f62-bfc5-8dac406d611d.js fast.fonts.com/jsapi	GET	200 OK	text/javascript	ynonperek.com:35 Parser	(from cache)	17ms 17ms	
modernizr.min.js cdnjs.cloudflare.com/ajax/libs/modernizr/2.0.6	GET	200 OK	application/x...	ynonperek.com:35 Parser	(from cache)	47ms 47ms	
font-awesome.css /sites/all/themes/yp2/css	GET	200 OK	text/css	ynonperek.com:35 Parser	(from cache)	84ms 84ms	
jquery.min.js ajax.googleapis.com/ajax/libs/jquery/1.7.1	GET	200 OK	text/javascript	ynonperek.com:35 Parser	(from cache)	103ms 103ms	
prettify.js cdnjs.cloudflare.com/ajax/libs/prettify/188.0.0	GET	200 OK	application/x...	ynonperek.com:35 Parser	(from cache)	103ms 103ms	
js_xAPl0qIk9cowy_I59tNkCWXLUVoat94SQT48UB /sites/default/files/js	GET	200 OK	application/ja...	ynonperek.com:35 Parser	(from cache)	103ms 103ms	

Cache Headers

- HTTP 1.0 Had an Expires header:

  Expires: Thu, 15 Apr 2010 20:00:00 GMT

- HTTP 1.1 Added Cache-Control header:

```
Cache-Control: max-age=315360000
```

Caching vs. Inlining

- Use external CSS and JavaScript for most web sites to allow caching
- Use inline CSS and JavaScript for pages that are only visited once (for example: landing pages)

```
<style>
  h1 {
    color: red
  }
</style>
```

```
<head>
    <title>Canvas Color Picker</title>

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

Reduce Distance

Closer => Lower RTT

Reduce Distance

- Consider using a local server
- Use a CDN
- Demo: Loading jQuery from CDN vs. Non-local server

Reduce Size

- Minify JS and CSS
- Use GZip Compression
- Split Payload
- Optimize Images

Help Thy Browser

Flush Early
Place Scripts At The Bottom
Place Stylesheets At The Top
Shard Dominant Domains

Flushing Early

- Web browsers try to do their best to load resources in parallel.
- Send some data first, and browser will start rendering

```
<html>
  <head>
    <title>HeadFirst</title>
    <link rel="stylesheet" href="style.css" />
  </head>

  <body>
    <p>Intro Text</p>
    <!-- long running server code -->
  </body>
</html>
```

Flush Early Caveats

- Verify presence of “Transfer-Encoding=chunked” header
- Verify header is long enough
    - Chrome has a minimum threshold of ~2KB
    - Safari has a minimum threshold of ~1KB
- Verify output buffering is off in php
  (or use ob_ functions)

Place Scripts At The Bottom

- Scripts block rendering when they execute
- By placing them at the bottom, browser can display prior data
- Demo Blocking Script Tag
- Note: Some browsers won’t start downloading ANYTHING while script tags are in queue

Defer vs. Async Scripts

- Defer downloads in parallel, executes by order
- Async downloads in parallel, executes when ready
- Use async if possible
- Best: Use webpack

Place Stylesheets At The Top

- styles block rendering
- Placing them at the top hints the browser to download first

```
<html>
  <head>
    <title>HeadFirst</title>
    <link rel="stylesheet"
href="style.css" />
  </head>

  <body>
    <p>Intro Text</p>
  </body>
</html>
```

Shard Dominant Domains

- Don’t automatically shard
- Find critical path using https://developers.google.com/speed/pagespeed/
- Sharding adds DNS work
- Usually 2 domains are enough

Measuring Times: Browser timing events

```
timingInfo = performance.timing;
```

Measuring Times: Browser timing events

- timingInfo.fetchStart - the time just before browser starts to fetch your page

- timingInfo.domainLookupStart,
  timingInfo.domainLookupEnd

- timingInfo.connectStart,
  timingInfo.connectEnd

- timingInfo.requestStart,
  timingInfo.responseStart

- And more:
  https://developer.mozilla.org/en-US/docs/Web/API/PerformanceTiming

HTTP/2.0

- Will fix many of the problems inherent to HTTP/1.1
- Expect:
    - Binary protocol
    - Cheaper requests
    - Server push
    - Connection multiplexing

Q & A

Optimize Site Responsiveness

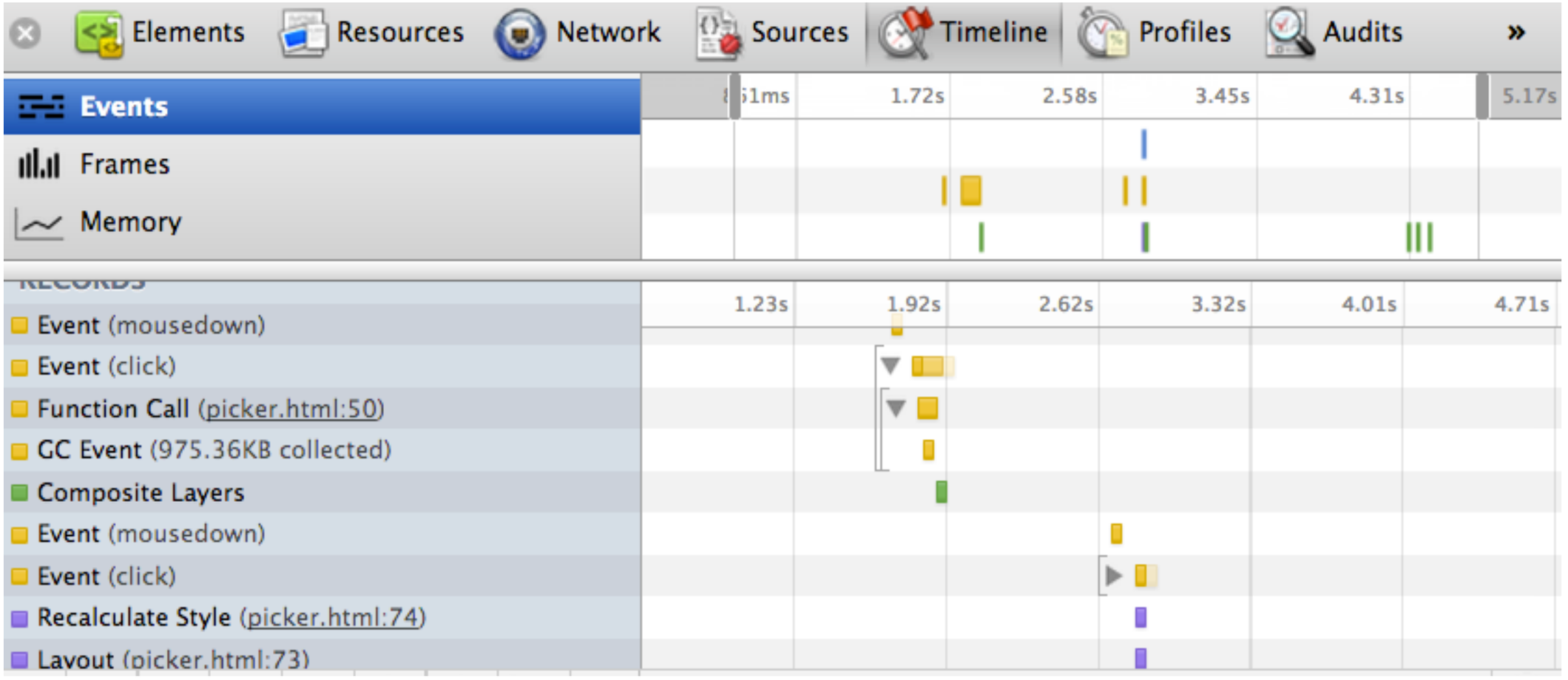
Responsiveness 101
How To Measure Responsiveness
Use Short Event Handlers
Prefer CSS3 Transitions over JS
Avoid Expensive JS
Simplify CSS Selectors

Responsiveness 101

Delay	**User Reaction**
0 - 100 ms	instant
100 - 300 ms	*Feels sluggish*
300 - 1000 ms	Machine is working
1s+	Context switching
10s+	Comes back later

Measuring Responsiveness

- Use Chrome Developer Tools, Timeline tab
- Keep event handlers < 100ms



Measuring Responsiveness

- Demo1: Measuring event handling
- Demo2: Measuring CSS Animation
- Demo3: Measuring Doodle Jump game
- Demo4: Long event handler causing application freeze
- Demo5: Measuring Mobile Performance using Chrome Remote Debugging

Responsiveness Takeaways

- On modern desktop browsers everything usually works fine
- Mobile is a different story
- Old browsers are also a big issue

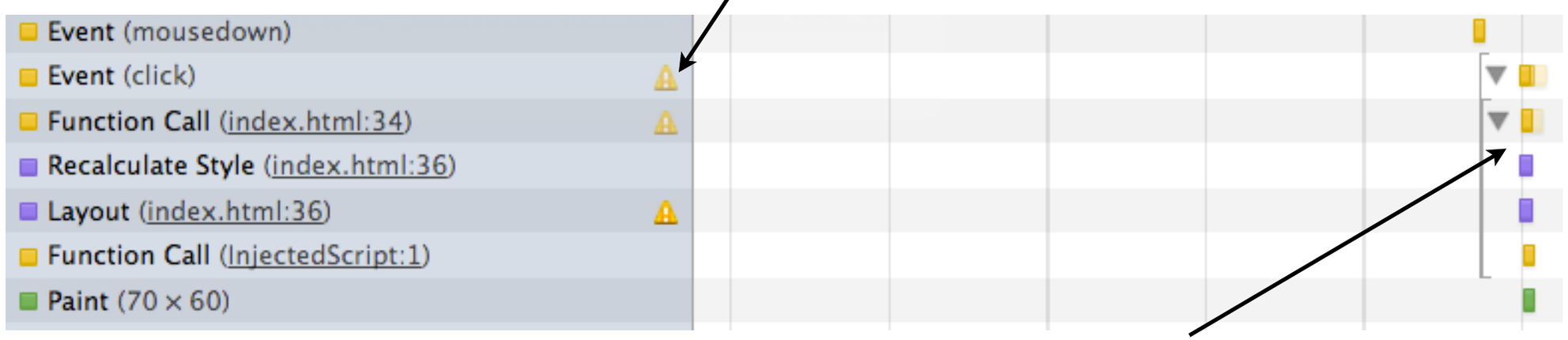
Avoid Expensive JavaScript

Chrome Developer Tools Will Help You Find And Mitigate Expensive JS Calls

Render Tree Is Expensive

- A browser saves its data in multiple data models
- Some JavaScript calls require syncing these models
- Demo

Chrome performance warning



Recalculate style inside event handler

DOM Is Expensive

- HTMLNodeList acts as a live query to the DOM
- The following hides a performance hit:

```
var images = document.querySelectorAll('img');

for ( var i=0; i < images.length; i++ ) {
  console.log( images[i].src );
}
```

DOM Is Expensive

- HTMLNodeList acts as a live query to the DOM
- Fix with:

```
var images = document.querySelectorAll('img');

for ( var i=0, len = images.length; i < len; i++ ) {
  console.log( images[i].src );
}
```

DOM Is Expensive

- Querying is expensive

- This has a performance hit:

```
document.querySelector('img').src = 'http://www.wallpaper-
valley.com/animal/animal_101.jpg';
document.querySelector('img').alt = 'Cute puppy';
document.querySelector('img').width = 100;
```

DOM Is Expensive

- Querying is expensive

- Fix by caching to a local variable

```
var img = document.querySelector('img');

img.src = 'http://www.wallpaper-valley.com/animal/animal_101.jpg';
img.alt = 'Cute puppy';
img.width = 100;
```

Some loops are expensive or long

- Looping on big data can be expensive

```
var data = [ 'foo', 'bar', 'buz', 'his' ];

for ( var i=0; i < data.length; i++ ) {
  console.log( data[i] );
}
```

Some loops are expensive or long

- Looping on big data can be expensive
- Faced with the problem - use timers to yield

```
function chunk( array, process, context ) {
  setTimeout(function() {
    var item = array.shift();
    process.call( context, item );

    if ( array.length > 0 ) {
      setTimeout( arguments.callee, 100 );
    }

  }, 100);
}
```

Some loops are expensive or long

- Looping on big data can be expensive
- Faced with the problem - use timers to yield
- Here's how you might use chunk

```
var copy = data.concat();
chunk( copy, function(item) {
  console.log( item );
});
```

Regexps Can Be Expensive

- Can you improve the following ?

```
function trim(str) {
  return str.replace(/^ +| +$/g, "");
}

console.log(trim("    hello    "));
```

Regexps Can Be Expensive

- Splitting the regexp is more efficient

```
function trim(str) {
  return str.replace(/^ +/, "").replace(/ +$/, "");
}

console.log(trim("   hello   "));
```

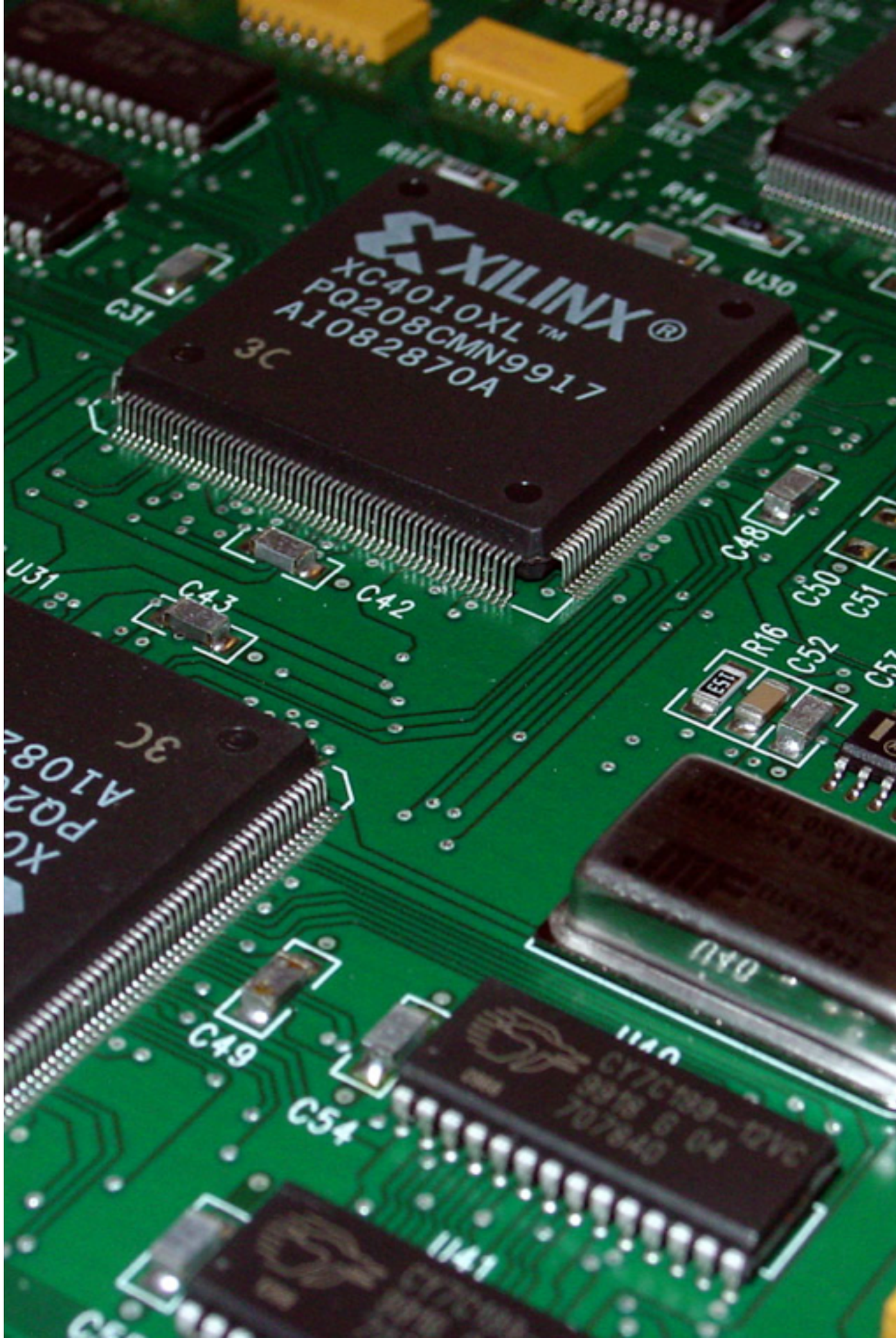
Smooth JavaScript Animations

- Replace setTimeout with requestAnimationFrame for animations
- No need to specify timeout value yourself
- Demo:
  http://jsbin.com/epatut/2/

Q & A

JavaScript Memory Management

Memory Lifecycle
Garbage Collector
Common Leaks



Memory Lifecycle

- All languages are the same:
- (1) Allocate some memory
- (2) Use that memory
- (3) Free that memory
- In JS, #3 is implicit

Memory Is Allocated When You Define Literals

```
var n = 123;
var s = "azerty";

var o = {
  a: 1,
  b: null
};

var a = [1, null, "abra"];

function f(a){
  return a + 2;
}

someElement.addEventListener('click', function(){
  someElement.style.backgroundColor = 'blue';
}, false);
```

Hidden Memory Allocations

```
var d = new Date();
var e = document.createElement('div'); // allocates an DOM
element
```

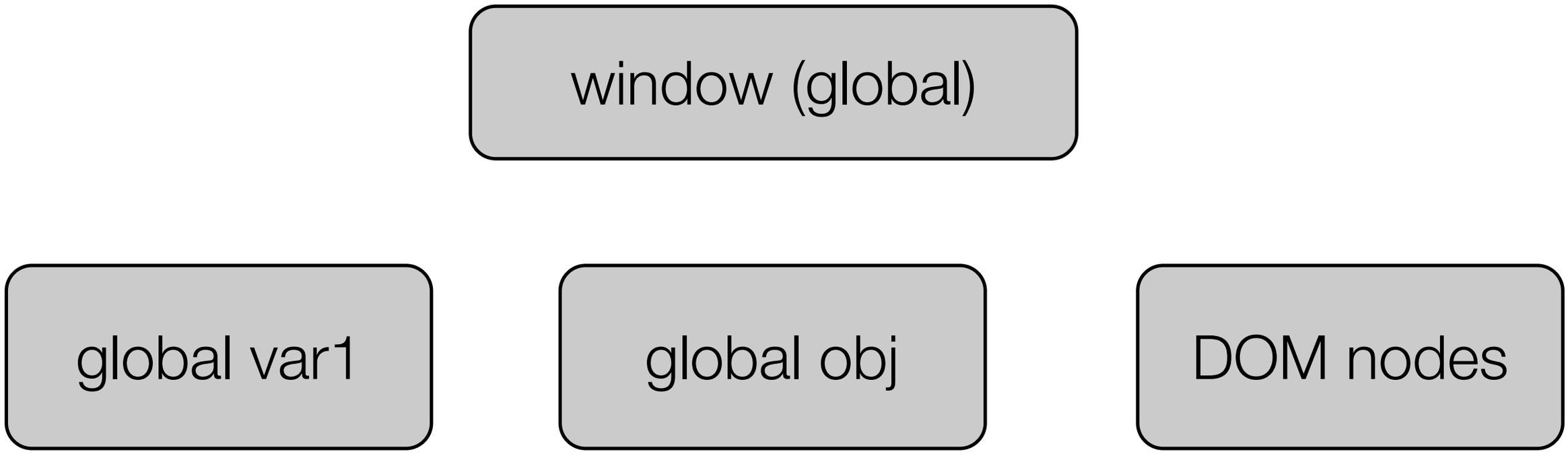
```
var s = "foo";
var s2 = s.substr(0, 3); // s2 is a new string

var a = ["ouais ouais", "nan nan"];
var a2 = ["generation", "nan nan"];
var a3 = a.concat(a2);
```

Releasing Memory

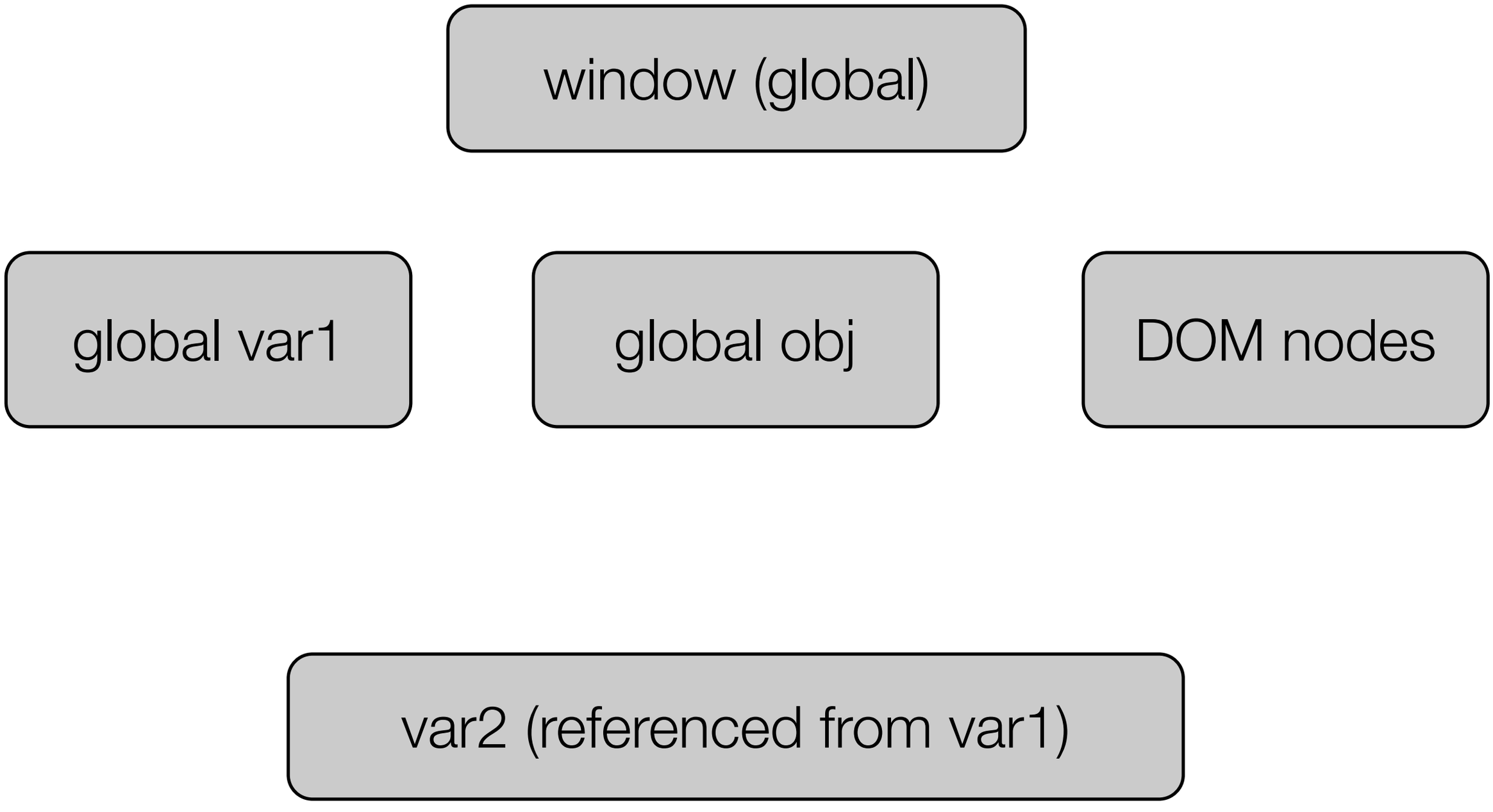
- JavaScript uses Mark-And-Sweep garbage collection algorithm
- It starts from known memory (global object)
- Follows all references
- In the end, clear the unreachable

Objects Graph

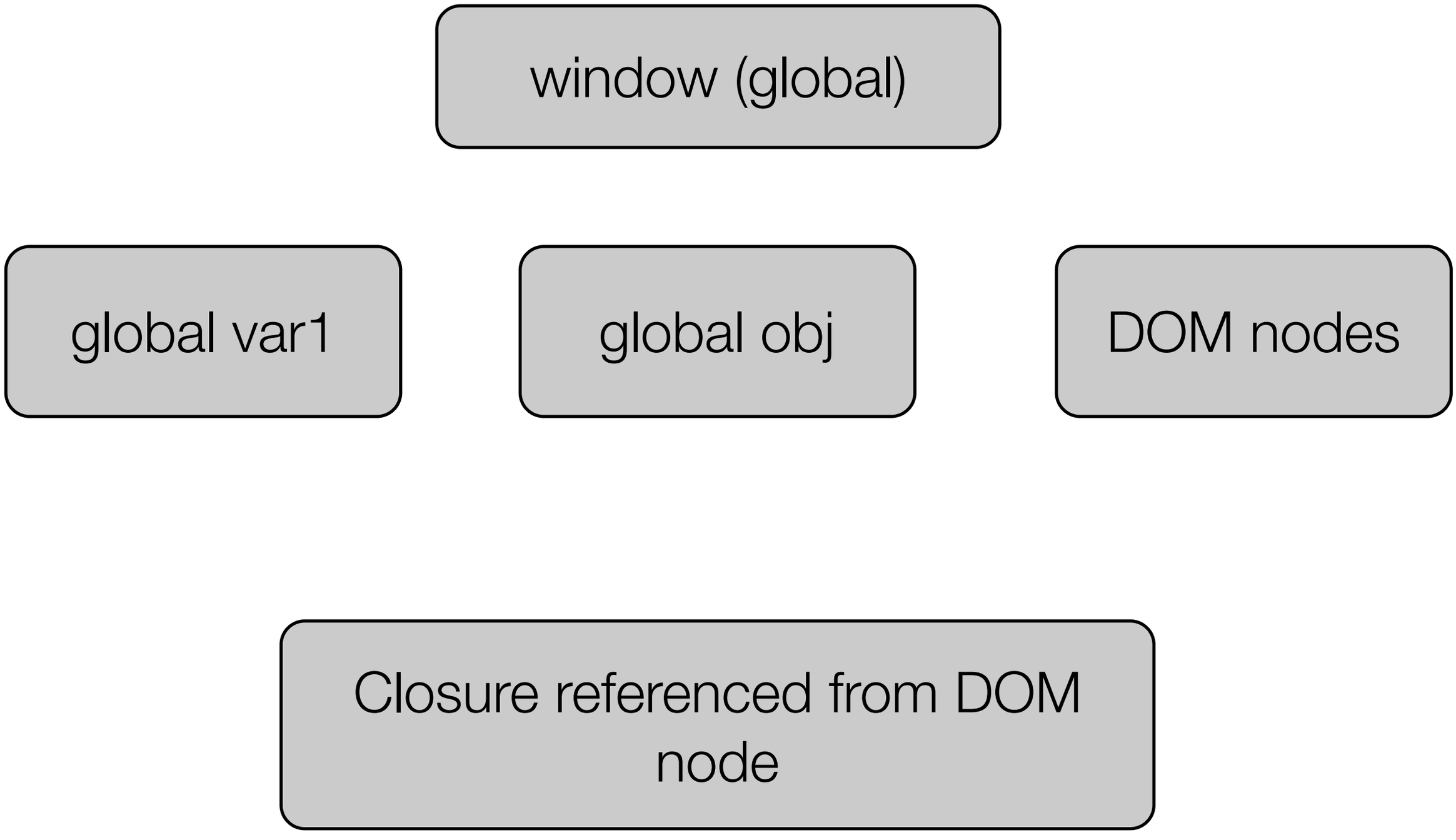
Objects Graph



Objects Graph



Objects Graph



Cleaning Up Memory

An object is released when:

garbage collector runs

AND

that object is unreachable

Good Release Candidates

- Function scope variables (lexicals) defined with var
- Detached DOM elements that nobody needs anymore
- Deleted object properties

Simplify CSS Selectors

- Browsers need to match CSS rules to elements

- The less matching attempts => The more efficient is the rule

```
div {
   overflow: hidden;
}

<body>
   <div></div>
   <div class="one"></div>
   <div id="two"></div>
</body>
```

Selectors Ordered By Complexity

- Most Simple: ID Selectors (#wrapper { ... })
- Class Selectors (.item { ... })
- Type Selectors (a { ... })
- Adjacent Sibling (h1 + #wrapper { ... })
- Child Selectors (#toc > li {...})
- Descendants Selector (ul li { ... })
- Universal Selector (* { ... })
- Attribute Selector ([href=”#home”] { ... })
- Pseudo-Classes / Elements (a:hover { ... })

How Browsers Read Selectors

- Selectors are parsed right-to-left

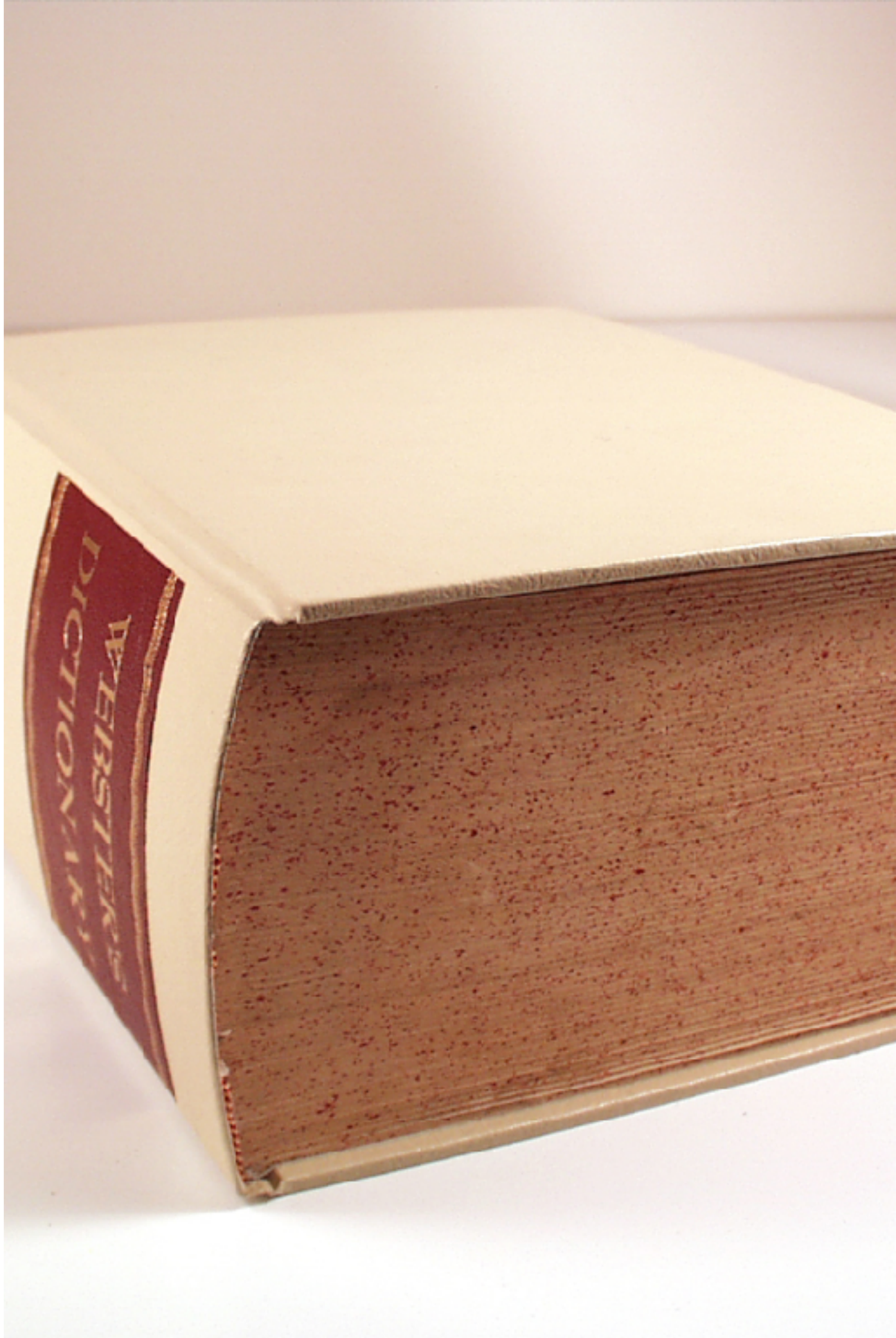
- `#top-menu a {...}` finds all a elements on the page, and checks if they are under a #top-menu element

- Try to use the most specific right part of a rule

- `a.menu-item` is better

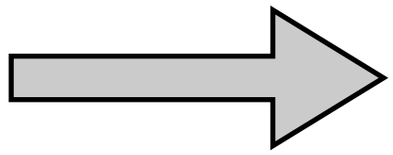
CSS Selectors Guidelines

Avoid Universal Rules
Don’t Qualify ID Selectors
Don’t Qualify Class Selectors
Use Specific Rules
Avoid Descendants Selectors
Avoid Tag-Child Selectors
Use Less CSS



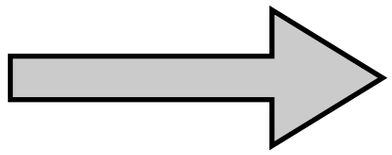
Avoid Universal Rules

- Try to minimize using selectors other than:
  ID, class and tag

a[href $= “pdf”] { ... }  a.pdf { ... }

Don’t Qualify ID Selectors

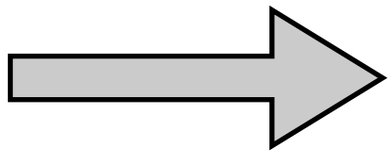
- It’s simply unnecessary
- There’s only one element with that ID

div#wrapper  #wrapper

Don’t Qualify Class Selectors

- Extend the class name to be more specific instead
- Classes are indexed, so class-only search is faster

ul.tasks

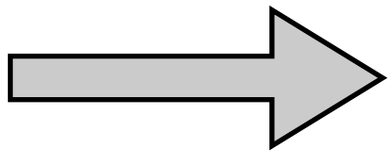


.task-list

Use Specific Rules

- Long lists give the browser a hard time

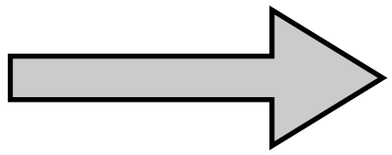
ol li a



.list-anchor

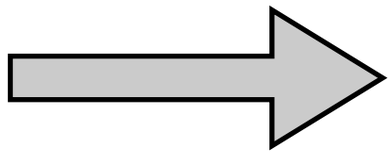
Avoid Descendant Selectors

- They’re expensive, because search is done right-to-left

ol li a  .list-anchor

Avoid Tag Child Selectors

#menu > li > a

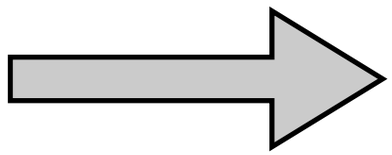


.menu-item

Minimize CSS Rule Content

- Can set a property on the parent and have it inherited to contained elements

```
ul li {
  list-style-image: ...
}
```



```
ul {
  list-style-image: ...
}
```

Q & A

Performance Bottom Line

- “97% of the time: premature optimization is the root of all evil”
  --- Donald Knuth

- First measure, then optimize
- Verify speed actually improves for real users

Awesome Resources

- Ilya Grigorik's free performance crash course http://www.igvita.com/2013/01/15/faster-websites-crash-course-on-web-performance/

- Web Performance Today Blog: http://www.webperformancetoday.com/

- Steve Souders Blog and Books: http://stevesouders.com/

Thanks For Listening

- Ynon Perek
- ynon@ynonperek.com
- www.tocode.co.il/blog