Web Performance

Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
DEI, FEUP, U.Porto

Web Performance

- ➔ Web optimization techniques are designed to improve the overall response time of a web application to the end-user.
- ➔ Usability studies show that page speed has a direct impact on conversion rates. Ideally a web page should load in less than 0.1 seconds, giving the user the feeling of an instantaneous response.
- ➔ A response time of less than 1 second keeps the user's flow seamless. Up to 10 seconds the user attention is kept. Over 10 seconds, the user is more likely to leave the page.
- ➔ Optimization opportunities both at the back-end or the front-end level.
  - ➔ Front-end: reduce images, reduce HTTP calls, etc.
  - ➔ Back-end: improve hardware, tune database, etc.

The Golden Rule

➔ In most web pages, less than 10-20% of the end user response time is spent getting the HTML document. To achieve significant improvements in response times, it is important to focus on front-end optimizations.

➔ **80% of the end-user response time is spent on the front-end.**

➔ Where the time is spent:

- ➔ Parsing HTML, Scripts, CSS, and images.
- ➔ Retrieving other page components (scripts, CSS, and images).

➔ Start with front-end optimizations:

- ➔ Greater potential for improvements.
- ➔ Simpler and proven to work.

Rules for High Performance Web Sites

From: High Performance Web Sites by Steve Souders (2007) &

Best Practices for Speeding Up Your Web Site (Yahoo)

Make Fewer HTTP Requests

- Given that 80-90% of the time is spent making HTTP requests for all the components (images, scripts, stylesheets, etc), a simple way to reduce response time is to reduce the number of HTTP requests.
- These techniques can reduce response times by as much as 50%.
- Main techniques:
  - Image Maps
  - CSS Sprites
  - Combine Scripts and Stylesheets

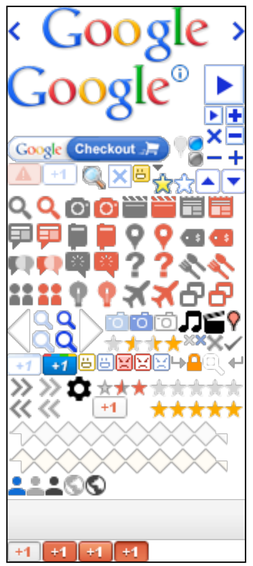
Image Maps

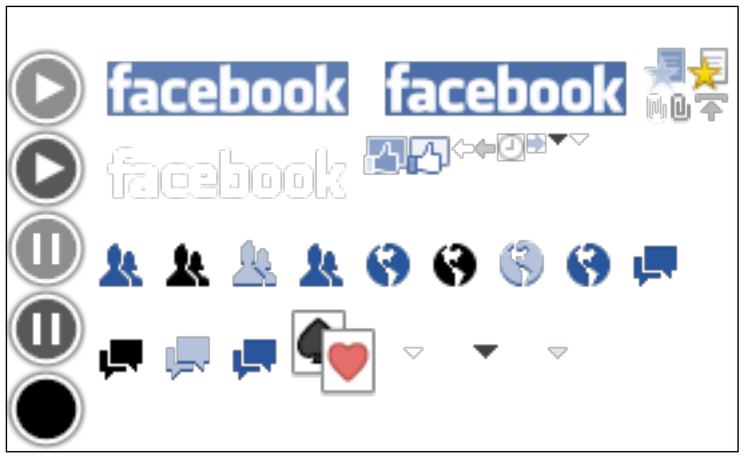
- An image map combines multiple images into a single image.
- The overall size is about the same, but reducing the number of HTTP requests speeds up the page. Image maps only work if the images are contiguous in the page, such as a navigation bar.
- Drawbacks:
  - Defining the coordinates of image maps is tedious and error prone.
  - Has accessibility limitations, thus should be avoided for important tasks.

CSS Sprites

➔ Using CSS sprites, multiple images are combined into a single file and displayed using CSS rules. This is the preferred method for reducing the number of image requests.

➔ Drawbacks: sprites are hard to maintain.





CSS Sprites Basic Example

sprite.png

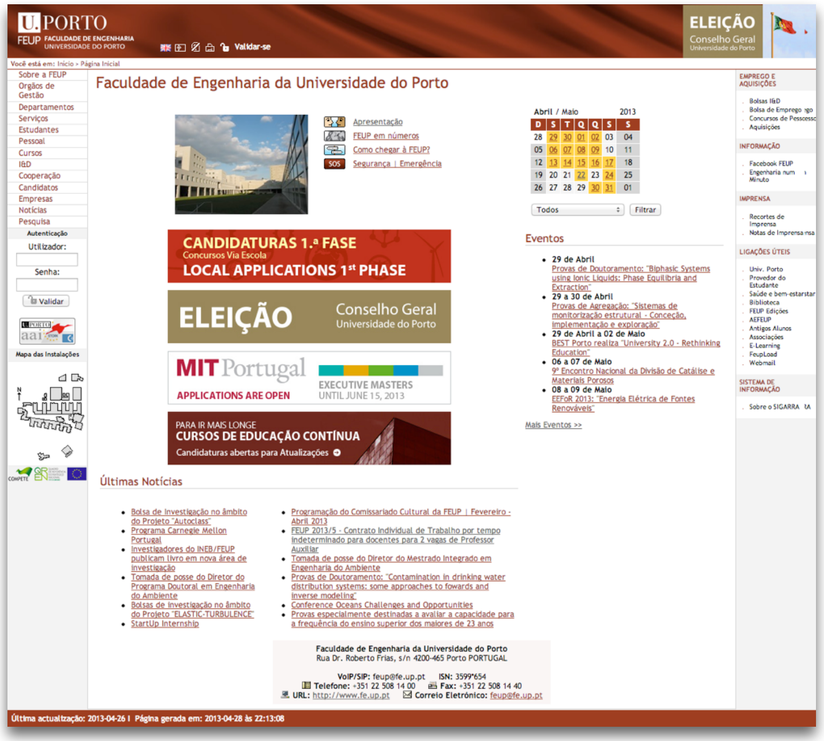
```
a {
  background: url("sprite.png")
  0px 0px no-repeat;
}
```

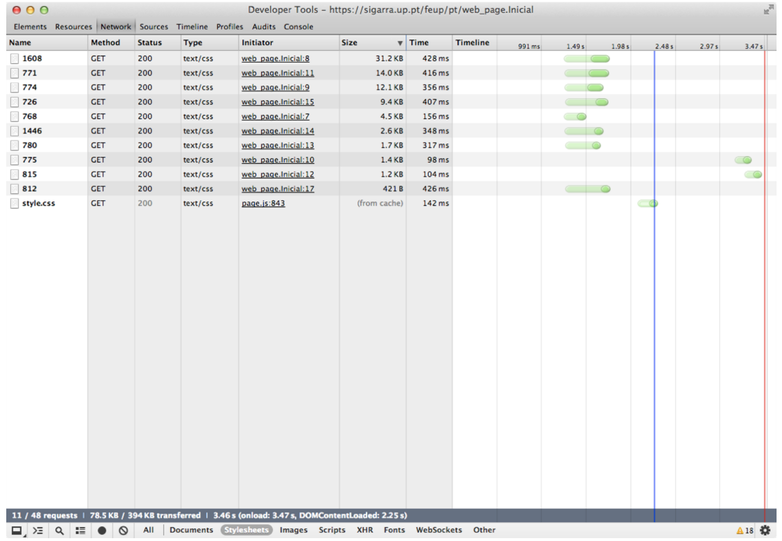
```
a:hover {
  background-position: 0px -100px;
}
```

Combine Scripts and Stylesheets

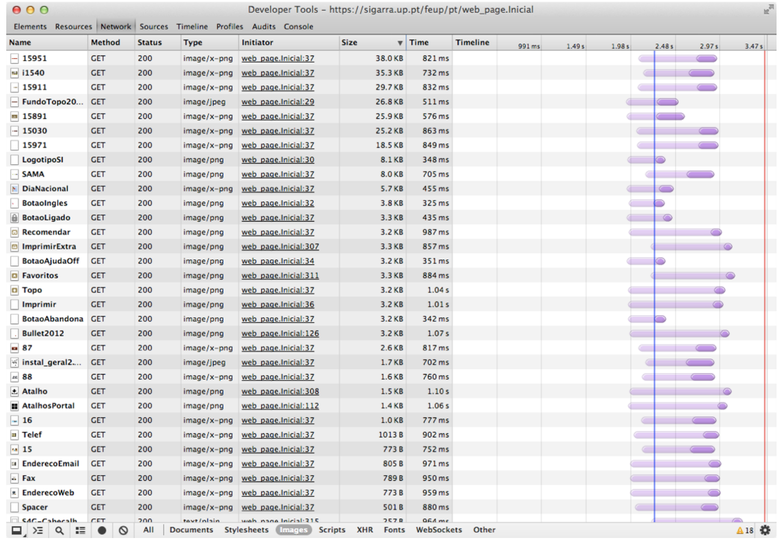
➔ One way to reduce the number of HTTP requests, is by combining all scripts into a single script, and similarly combining all CSS into a single stylesheet.

➔ Might be challenging when scripts and stylesheets vary from page to page.





11 CSS resources



34 image resources

Optimize Images

➔ Use the right formats: JPEG for photos (lossy), PNG for graphics (lossless).

➔ Don't resize using with HTML/CSS.

➔ Optimize for the web: optimize for the web features.

➔ Yahoo! Smush.it — Image optimization service (lossless tool).
ysmush.it (*discontinued, March 2015*)

➔ Alternative: http://resmush.it/

Use a Content Delivery Network

➔ The user's proximity to the web server has impact on a page's response time.

➔ A content delivery network (CDN) is a collection of web servers distributed across multiple locations to deliver content to users more efficiently.

➔ CDNs are used to deliver static content, such as images, scripts, stylesheets, binaries, and Flash. Serving dynamic HTML pages involves specialized hosting requirements.

➔ Top CDN providers: Akamai, CloudFlare, Mirror Image, Limelight, SAVVIS.

Add an Expires Header

➔ A first-time visitor to a web page needs to make several requests to obtain all elements. By using a future Expires header, these components can be made cacheable, and thus re-used in following requests.

➔ Most commonly used with images, but should be used on all components, including scripts, stylesheets, etc.

➔ The Expires header is sent in the HTTP response.

➔ If a far future date is used (e.g. years), the filename must be changed if the component changes.

Gzip Components

➔ Response times can be reduced either by reducing the number of requests, or by reducing the size of the response in each request.

➔ Gzip encoding can be used to compress HTTP response, and thus reduce network response times.

➔ Using gzip generally reduces the response size by about 70%. Approximately 90% of today's Internet traffic travels through browsers that claim to support gzip.

➔ Configured at the web server.

Make JavaScript and CSS External

➔ Using inline CSS or JavaScript makes HTML documents bigger.

➔ Using external files results in more HTTP requests, but cacheable.

➔ The key factor in deciding which option is better is the frequency with which external JavaScript and CSS components are cached relative to the number of HTML documents requested.

Reduce DNS Lookups

➔ The Domain Name System (DNS) maps hostnames to IP addresses.

➔ A DNS lookup for a given hostname typically costs 20-120 milliseconds.

➔ DNS lookups can be reduced by using fewer hostnames (ideal: 2-4).

Minify JavaScript and CSS

- Minification is the practice of removing unnecessary characters from code to reduce its size thereby improving load times.
- When code is minified all comments are removed, as well as unneeded white space characters (space, newline, and tab). In the case of JavaScript and CSS, this improves response time performance because the size of the downloaded file is reduced.
- Popular tools:
  - JSMin — www.crockford.com/jsmin.html
  - YUI Compressor — yui.github.io/yuicompressor/
- The YUI compressor can also minify CSS.

Avoid Redirects

➔ Redirects are achieved using 3xx status codes, mostly 301 and 302.

➔ Redirects slow down the user experience since nothing in the page can be rendered and no components can start being downloaded.

➔ One of the most wasteful redirects happens when a trailing slash (/) is missing from a URL that should otherwise have one. For example, going to http://example.com/tag results in a 301 response containing a redirect to http://example.com/tag/.

➔ Although redirects degrades the user experience, it can reduce the complexity for developers in several situations.

Remove Duplicate Scripts

➔ It hurts performance to include the same JavaScript file twice in one page.

➔ Two main factors increase the odds of a script being duplicated in a single web page: team size and number of scripts.

➔ Hurts performance because the scripts are downloaded (in some browsers) and executed multiple times.

Configure ETags

➔ Entity tags (ETags) are a mechanism that web servers and browsers use to determine whether the component in the browser's cache matches the one on the origin server.

➔ The problem with ETags is that for a single entity there are always differences across servers (eg. file timestamps). Using multiple servers is a common situation in large web sites.

➔ ETags should not be used if the number of servers is larger than 1.

Make AJAX Cacheable

- ➔ Some of the previous rules also apply to AJAX components (e.g. JSON, scripts), namely:
  - ➔ Gzip Components
  - ➔ Reduce DNS lookups
  - ➔ Minify JavaScript
  - ➔ Avoid Redirects
  - ➔ Configure ETags
- ➔ A personalized response should still be cacheable by that person.

Performance Evaluation Tools

Google Performance Evaluation Tools

➔ developers.google.com/speed/

➔ **PageSpeed Insights**
developers.google.com/speed/pagespeed/insights/

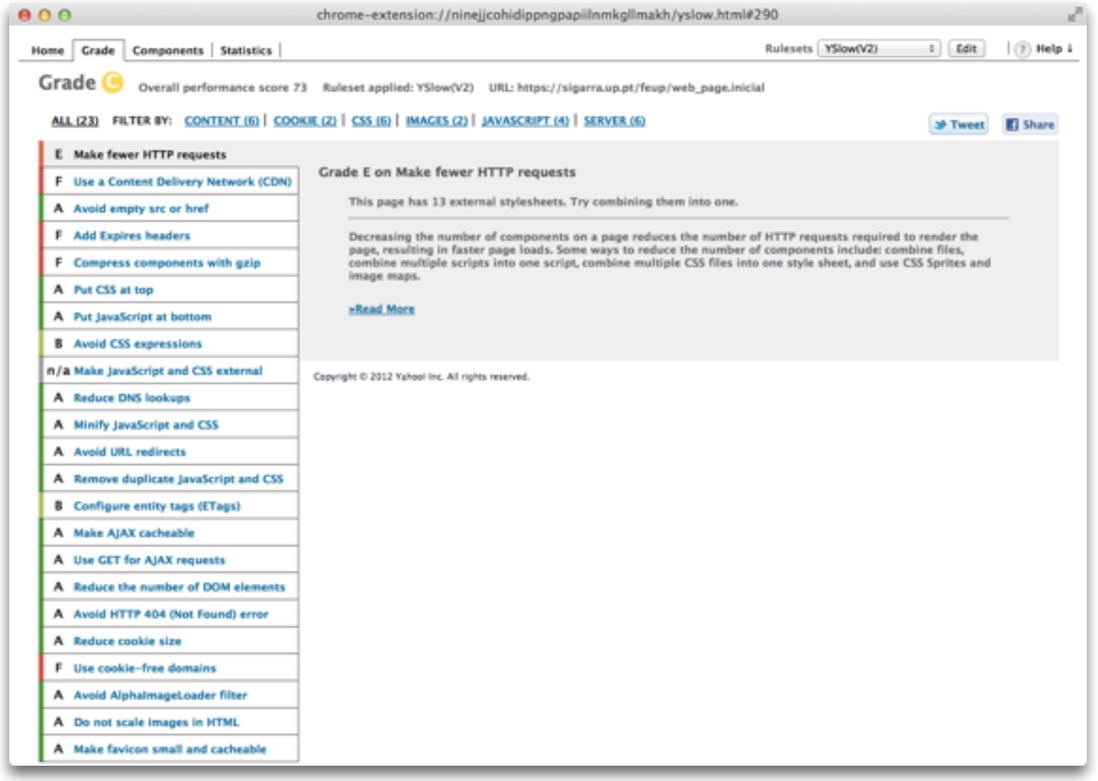
➔ PageSpeed Insights for www.fe.up.pt

➔ **Google Lighthouse** - Open-source tool (Chrome, command line, online)
developers.google.com/web/tools/lighthouse/

➔ **WebP** - Image format
developers.google.com/speed/webp

YSlow

➔ YSlow analyzes web pages and why they're slow based on Yahoo!'s rules for high performance web sites. — http://yslow.org



References

➔ **Yahoo's Exceptional Performance Team**
developer.yahoo.com/performance/ [archived]

➔ **Best Practices for Speeding Up Your Web Site**
developer.yahoo.com/performance/rules.html

➔ **Make the Web Faster | Google Developers**
developers.google.com/speed/

➔ **High Performance Web Sites**
by Steve Souders. O'Reilly 2007.

➔ **Even Faster Web Sites**
by Steve Souders. O'Reilly 2009.