FIT5032 - Internet Applications Development

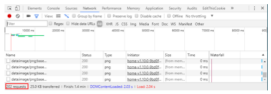
Optimisation

Prepared by Yiwei Zhong
Updated by ABM Russel

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

1. Reduce HTTP request



Optimize HTTP request by:

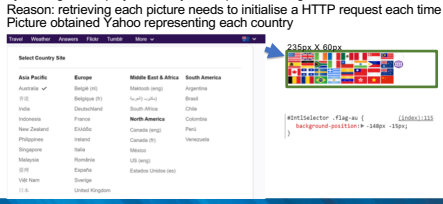
- Combing images
- Reduce the size of files
- Making JavaScript asynchronous

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

2. Image Optimization

Make use of CSS to replace icons instead of small sized images
If small icons is unavoidable, merge the icons in one picture, access each picture by limiting the display boundary of the picture through CSS
Reason: retrieving each picture needs to initialise a HTTP request each time
Picture obtained Yahoo representing each country



2. Image compression – Types

There are various of image types, most common images compression types are:
PNG – **Lossless compression**
(The original data is perfectly reconstructed from the compressed data)
Suitable for icons, pictures with no background

JPEG – **Lossy compression**
(Ideals for balance between the storages size and image quality)

Right picture shows the image after deep compression ↑

2. Image compression – Responsive Images

Srcset & Size attributes

How you load the images normally:

```
<img src="monash.jpg" alt="responsive images car" width="800" height="600">
```

With "Srcset" attribute: (Browser will decide which image it choose, w means the width of the viewport)

```
<img src="monash.jpg" alt="responsive images car" srcset="monash-160.jpg 160w, monash-320.jpg 320w,
monash-640.jpg 640w, monash-1280.jpg 1280w">
```

Usually used with conjunction of the "size" attribute if want to force browser to use the image under certain width

`(max-width: 480px) 100vw` means if the browser is lower than 480px, then show 100% of its size

```
<img src="responsive-images-car.jpg" alt="responsive images car" srcset="responsive-images-car-
160.jpg 160w, responsive-images-car-320.jpg 320w, responsive-images-car-640.jpg 640w,
responsive-images-car-1280.jpg 1280w" sizes="(max-width: 480px) 100vw, (max-width: 900px) 33vw,
254px">
```

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

3. Minify CSS and JavaScript

Removing unnecessary characters from your HTML

- White space characters
- New line characters
- Comments
- Block delimiters

https://javascript.mireller.com/

Input JavaScript

Minified Output

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

4. Render Blocking Resources (CSS + JS + Web font)

Render Blocking Resources
HTML
CSS
JavaScript
Web fonts

Images aren't render blocking

CSS
Ultimate Goal is get it to the client as soon and as quickly as possible to optimize the time to first render.
So it is loaded in <head> section
But,
Some of the css style is not that critical, such as when we only use the css when printing

CSS "media types" and "media queries" allow us to address these use cases:

```
<link href="style.css" rel="stylesheet">
<link href="print.css" rel="stylesheet" media="print">
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
<link href="portrait.css" rel="stylesheet" media="orientation:portrait">
```

4. Render Blocking Resources (CSS + JS + Web font)

Additional way to optimize your CSS
Lessen the amount of CSS files (concatenate your CSS files into one file)
Minify Your CSS (remove extra spaces, characters, comments, etc)
Use less CSS overall

JavaScript

- Move your scripts to the bottom of the page right before your </body> tag.

  Reason: Putting JavaScript in the top/ middle of your body will stop the browser rendering the rest of the content before JavaScript is loaded

- Move your Use the async or defer JavaScript to avoid render blocking.

Async
<script async src="foobar.js"></script>

*Does not mean the webpage is fully loaded before JavaScript is executed

4. Render Blocking Resources (CSS + JS + Web font)

Deferring JavaScript

Ultimate goal is loading or parsing JavaScript only after page content has loaded

```
<script type="text/javascript">
    function downloadJSAtOnload() {
        var element = document.createElement("script");
        element.src = "filesYouWishToLoad.js";
        document.body.appendChild(element);
    }
    if (window.addEventListener)
        window.addEventListener("load", downloadJSAtOnload, false);
    else if (window.attachEvent)
        window.attachEvent("onload", downloadJSAtOnload);
    else window.onload = downloadJSAtOnload;
</script>
```

Best to separate JS into two Groups

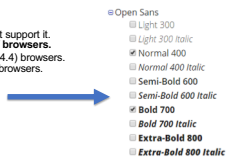
- Essential JavaScript files need to load (jQuery Library, etc)
- Non essential JavaScript files (Click event etc)

4. Render Blocking Resources (CSS + JS + Web font)

Web Font
The disadvantages of web fonts are that they add extra HTTP requests to external resources.

Recommendations:
Prioritize font based on browser support
Serve WOFF 2.0 variant to browsers that support it.
Serve WOFF variant to the majority of browsers.
Serve TTF variant to old Android (below 4.4) browsers.
Serve EOT variant to old IE (below IE9) browsers.

Choose only the styles you need



4. Render Blocking Resources (CSS + JS + Web font)

Recommendations (Cont.):
Keep character sets down to a minimum
Unless you are dealing with multiple languages

Greek (greek) Greek Extended (greek-ext) Latin (latin)
Vietnamese (vietnamese) Cyrillic Extended (cyrillic-ext)
Latin Extended (latin-ext) Cyrillic (cyrillic)

Host fonts locally or prefetch
Reduce HTTP Request

*Check whether it is open source license

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

5. Reduce Latency with a CDN + TTFB

CDN
A **content delivery network** (**CDN**) is a system of distributed servers (**network**) that deliver pages and other Web **content** to a user, based on the geographic locations of the user, the origin of the webpage and the **content delivery** server.

Imagine your hosting server is in Dallas.

POP = points of presence

5. Reduce Latency with a CDN + TTFB

How it will affect the latency:

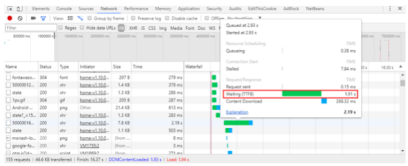
Server (POP) Location	No CDN RTT (ms)	KeyCDN RTT (ms)	Difference %
New York, US	36.908	18.096	- 50.97%
Dallas, US	0.751	1.138	+ 51.53%
San Francisco, US	39.645	18.900	- 52.33%
Frankfurt, DE	123.072	3.734	- 96.97%
London, UK	127.555	4.548	- 96.43%
Paris, FR	112.417	1.689	- 98.5%
Amsterdam, NL	118.418	10.364	- 91.25%
Singapore, SG	202.682	2.002	- 99.01%
Sydney, AU	191.848	0.705	- 99.63%
Tokyo, JP	130.804	3.379	- 97.42%

*RTT = Round-trip time

5. Reduce Latency with a CDN + TTFB

TTFB
Time to first byte (TTFB) is the **measurement of the responsiveness of a web server**

HTTP request time + Process request time + HTTP response time



5. Reduce Latency with a CDN + TTFB

Factors influence TTFB

Database design
Webserver Configuration
Hardware limit of the original server

Mostly done by web server administrator

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

6. Caching

Making use of the browser cache for assets that are rarely changing

Catch control

ETag

- Catch control

Each resource can define its caching policy via the Cache-Control HTTP header.
Cache-Control directives control who can cache the response, under which conditions, and for how long.

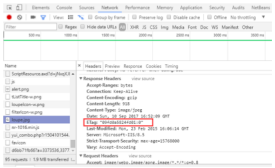
Catch control using header in asp.net

Under web.config

```
<configuration>
    <system.webServer>
        <staticContent>
            <clientCache cacheControlCustom="public"
                         cacheControlMaxAge="12:00:00" />
        </staticContent>
    </system.webServer>
</configuration>
```

6. Caching

ETag
The server uses the ETag HTTP header to communicate a validation token.
The validation token enables efficient resource update checks: no data is transferred if the resource has not changed.



Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

7. Prefetch and Preconnect

DNS Prefetching
DNS prefetching is an attempt to resolve domain names before a user tries to follow a link

```
<link rel="dns-prefetch" href="//www.example.com">
```

Preconnect
Allows the browser to set up early connections before an HTTP request is actually sent to the server.
Connections such as DNS Lookup, TCP Handshake, and TLS negotiation can be initiated beforehand, **eliminating roundtrip latency** for those connections and saving time for users.

```
<link href='https://example.com' rel='preconnect' crossorigin>
```

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

8. Hotlink protection

Restricting HTTP referrers in order to prevent others from embedding your assets on other websites. (For example: cannot load your website image under other website)

Hotlink protection will save bandwidth by **prohibiting other sites from displaying your images/ contents**.

Hotlink Protection

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

9. Fix 404 Errors

Depends on the platforms, some of the 404 errors uses a large amount of memory
For example, Drupal uses 60 – 100 mb memory to deliver a 404

Recommendation:
Make use of external tools to run through the website to see whether it has broken links:
Online Broken Link Checker
http://www.brokenlinkcheck.com/

Or
Screaming Frog
https://www.screamingfrog.co.uk/

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

10. Database Optimization

Cleaning out old unused tables or creating indexes for faster access there are always things that can be optimized.
E.g. Decide which join give best performance Hash join / Merge join / Nested Loops
Whether to create a view
Some of the reporting functions can be performed overnight

If website intention is built on reporting and business intelligence, make use of star schema instead of traditional normalised tables