Best Practices for Speeding Up A Web Site

What is a Web Server?

- Yahoo's The Exceptional Performance team has identified a number of best practices for making web pages fast.
- The list includes 35 best practices divided into 7 categories.

Minimize HTTP Requests

- 80% of the end-user response time is spent on the front-end. Most of this time is tied up in downloading all the components in the page:
  - images
  - stylesheets
  - scripts
  - Flash
- Reducing the number of components in turn reduces the number of HTTP requests required to render the page. This is the key to faster pages.

Minimize HTTP Requests

- **Combined files**
  - Combining all scripts into a single script, and similarly combining all CSS into a single stylesheet.
  - This reduces the number of HTTP requests but makes the response for each request longer. So we have to consider overall download time as well as the number of http requests
  - Combining files is challenging when the scripts and stylesheets vary from page to page, but making this part of your release process improves response times.

Minimize HTTP Requests

- CSS Sprites
  - CSS Sprites are the preferred method for reducing the number of image requests. Combine your background images into a single image and use the CSS background-image and background-position properties to display the desired image segment.

CSS Sprites Example

CSS Sprites Example

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      img.home{
          width:46px;
          height:44px;
          background:url(img_navsprites.gif) 0 0;
      }
      img.next{
          width:43px;
          height:44px;
          background:url(img_navsprites.gif) -91px 0;
      }
    </style>
  </head>
  <body>
    <img class="home" src="img_trans.gif">
    <br><br>
    <img class="next" src="img_trans.gif">
  </body>
</html>
```

CSS Sprites Example

- http://www.w3schools.com/css/css_image_sprites.asp

Optimize CSS Sprites

- Arranging the images in the sprite horizontally as opposed to vertically usually results in a smaller file size.
- Combining similar colors in a sprite helps you keep the color count low, ideally under 256 colors so to fit in a PNG8.
- “Be mobile-friendly” and do not leave big gaps between the images in a sprite. This does not affect the file size as much but requires less memory for the user agent to decompress the image into a pixel map. 100x100 image is 10 thousand pixels, where 1000x1000 is 1 million pixels

Minimize HTTP Requests

- Image maps
  - Image maps combine multiple images into a single image.
  - The overall download 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.
  - Defining the coordinates of image maps can be tedious and error prone.
  - Using image maps for navigation is not accessible too, so it's not recommended.

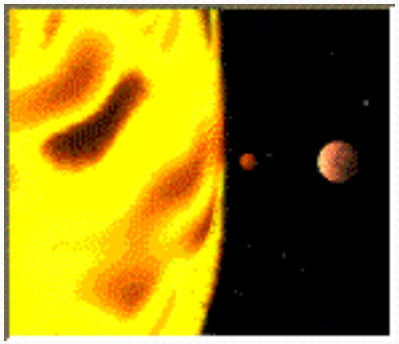
Minimize HTTP Requests

- Image maps - Problems
  - Image maps only work if the images are contiguous in the page, such as a navigation bar.
  - Defining the coordinates of image maps can be tedious and error prone.
  - Using image maps for navigation is not accessible too, so it's not recommended.

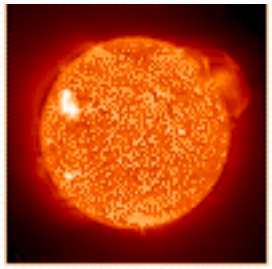
Image maps - Example

```
<!DOCTYPE html>
<html>
  <body>
    <p>Click on the sun or on one of the planets to watch it closer:</p>

    <img src="planets.gif"
         width="145" height="126"
         alt="Planets" usemap="#planetmap">

    <map name="planetmap">
      <area shape="rect"
            coords="0,0,82,126" alt="Sun" href="sun.htm">
      <area shape="circle"
            coords="90,58,3" alt="Mercury" href="mercur.htm">
      <area shape="circle"
            coords="124,58,8" alt="Venus" href="venus.htm">
    </map>

  </body>
</html>
```

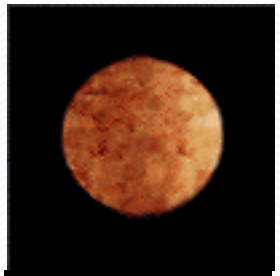
Image maps - Example



planet.gif – Map Image



sun.gif



merglobe.gif

venglobe.gif

Image maps - Example

- http://www.w3schools.com/tags/tag_map.asp

Minimize HTTP Requests

- Inline images
  - Inline images use the data: URL scheme to embed the image data in the actual page.
    - `<img src="smiley.gif" alt="Smiley face" height="42" width="42">`
  - This can increase the size of your HTML document.
  - Combining inline images into your (cached) stylesheets is a way to reduce HTTP requests and avoid increasing the size of your pages. Inline images are not yet supported across all major browsers.

Use a Content Delivery Network (CDN)

- The user's proximity to your web server has an impact on response times.
- Deploying your content across multiple, geographically dispersed servers will make your pages load faster from the user's perspective. But where should you start?

Use a Content Delivery Network (CDN)

- Remember that 80-90% of the end-user response time is spent downloading all the components in the page: images, stylesheets, scripts, Flash, etc.
- So …
  - disperse your static content.
- Content delivery network (CDN) is a collection of web servers distributed across multiple locations to deliver content more efficiently to users.

Use a Content Delivery Network (CDN)

- There are commercial CDNs so you do not have to build your own
  - Akamai
  - EdgeCast
  - Level3
- But they are costly $$$

Add an Expires or a Cache-Control Header

- There are two aspects to this rule:
  - For static components: implement "Never expire" policy by setting far future Expires header
  - For dynamic components: use an appropriate Cache-Control header to help the browser with conditional requests
    - Here you have to determine a Time-To-Live (TTL) value

Add an Expires or a Cache-Control Header

- Web page designs are getting richer and richer, which means more scripts, stylesheets, images, and Flash in the page.
- A first-time visitor to your page may have to make several HTTP requests, but by using the Expires header you make those components cacheable.
- This avoids unnecessary HTTP requests on subsequent page views.
- Expires headers are most often used with images, but they should be used on *all* components including scripts, stylesheets, and Flash components.

Add an Expires or a Cache-Control Header

- Browsers use a cache to reduce the number and size of HTTP requests, making web pages load faster.
- A web server uses the Expires header in the HTTP response to tell the client (the Browser) how long a component can be cached.
- This is a far future Expires header, telling the browser that this response won't be stale until April 15, 2015.

```
Expires: Thu, 15 Apr 2015 20:00:00 GMT
```

Gzip Components

- The time it takes to transfer an HTTP request and response across the network can be significantly reduced by decisions made by front-end engineers.
- It's true that the end-user's bandwidth speed, Internet service provider, proximity to peering exchange points, etc. are beyond the control of the development team.
- But there are other variables that affect response times. Compression reduces response times by reducing the size of the HTTP response.

Gzip Components

- Web clients (Browsers in this case) indicate support for compression with the Accept-Encoding header in the HTTP request.

```
Accept-Encoding: gzip, deflate
```

- If the web server sees this header in the request, it *may* compress the response using one of the methods listed by the client.
- The web server notifies the web client of this via the Content-Encoding header in the response.

```
Content-Encoding: gzip
```

Put Stylesheets at the Top

- Moving stylesheets to the document HEAD makes pages *appear* to be loading faster.
- This is because putting stylesheets in the HEAD allows the page to render progressively.

Put Stylesheets at the Top

- Front-end engineers want a page to load progressively; that is, they want the browser to display whatever content it has as soon as possible.
- This is especially important for pages with a lot of content and for users on slower Internet connections.
- When the browser loads the page progressively the header, the navigation bar, the logo at the top, etc. all serve as visual feedback for the user who is waiting for the page.
- This improves the overall user experience.

Put Stylesheets at the Top

- The problem with putting stylesheets near the bottom of the document is that it prohibits progressive rendering in many browsers
- Many Browsers block rendering to avoid having to redraw elements of the page if their styles change. The user is stuck viewing a blank white page.
- And the kicker
  - the HTML specification says to put your stylesheets in the document HEAD

Put Scripts at the Bottom

- The problem caused by scripts is that they block parallel downloads.
- The HTTP/1.1 specification suggests that browsers download no more than two components in parallel per hostname.
  - If you serve your images from multiple hostnames, you can get more than two downloads to occur in parallel.
- However, while a script is downloading, the browser will not start any other downloads, even on different hostnames.

Put Scripts at the Bottom

- In some situations it's not easy to move scripts to the bottom. If, for example, the script uses document.write to insert part of the page's content, it can't be moved lower in the page. There might also be scoping issues. In many cases, there are ways to workaround these situations.

Avoid CSS Expressions

- CSS expressions are a powerful (and dangerous) way to set CSS properties dynamically.
- As an example, the background color could be set to alternate every hour using CSS expressions

```
background-color: expression(
  (new Date()).getHours()%2 ? "#B8D4FF" : "#F08A00" );
```

Make JavaScript and CSS External

- Using external files generally produces faster pages because the JavaScript and CSS files are cached by the browser.
- JavaScript and CSS that are inlined in HTML documents get downloaded every time the HTML document is requested.
- This reduces the number of HTTP requests that are needed, but increases the size of the HTML document.

Make JavaScript and CSS External

- If the JavaScript or CSS are in external files cached by the browser, the size of the HTML document is reduced without increasing the number of HTTP requests.
  - This is due to the browser determining that it already has the JavaScript or CSS

Make JavaScript and CSS External

- The key factor is the frequency with which external JavaScript and CSS components are cached relative to the number of HTML documents requested.
- This factor, although difficult to quantify, can be gauged using various metrics.
  - If users on your site have multiple page views per session and many of your pages re-use the same scripts and stylesheets, there is a greater potential benefit from cached external files.
- You have to consider how users navigate your site
  - Sitecatalyst

Reduce DNS Lookups

- The Domain Name System (DNS) maps hostnames to IP addresses, just as phonebooks map people's names to their phone numbers.
- When you type `www.asu.edu` into your browser, a DNS resolver contacted by the browser returns that server's IP address.
- DNS has a cost. It typically takes 20-120 milliseconds for DNS to lookup the IP address for a given hostname. The browser cannot download anything from this hostname until the DNS lookup is completed.

Reduce DNS Lookups

- Reducing the number of unique hostnames has the potential to reduce the amount of parallel downloading that takes place in the page.
- Avoiding DNS lookups cuts response times, but reducing parallel downloads may increase response times.
- A guideline is to split these components across at least two but no more than four hostnames.
- This results in a good compromise between reducing DNS lookups and allowing a high degree of parallel downloads.

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, this improves response time performance because the size of the downloaded file is reduced.
  - Two popular tools for minifying JavaScript code are JSMin and YUI Compressor. The YUI compressor can also minify CSS.

Minify JavaScript and CSS

- Obfuscation is an alternative optimization that can be applied to source code.
- It is more complex than minification and therefore more likely to generate bugs as a result of the obfuscation step itself.
- Many companies use Obfuscation when the JavaScript is a core Corporate asset
  - Site Analytics

Avoid Redirects

- Redirects are accomplished using the `301` and `302` status codes. Here's an example of the HTTP headers in a 301 response:

```
HTTP/1.1 301 Moved Permanently
Location: http://example.com/newuri
Content-Type: text/html
```

- The browser automatically takes the user to the URL specified in the Location field.

Avoid Redirects

- The main thing to remember is that redirects slow down the user experience.
- Inserting a redirect between the user and the HTML document delays everything in the page since nothing in the page can be rendered and no components can start being downloaded until the HTML document has arrived.

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.
- An Entity (component) could be: images, scripts, stylesheets, etc.
- An ETag is a string that uniquely identifies a specific version of a component. The only format constraints are that the string be quoted.

Configure ETags

- Example

```
HTTP/1.1 200 OK
Last-Modified: Tue, 12 Dec 2006 03:03:59 GMT
ETag: "10c24bc-4ab-457e1c1f"
Content-Length: 12195
```

- Example

```
GET /i/yahoo.gif HTTP/1.1
Host: us.yimg.com
If-Modified-Since: Tue, 12 Dec 2006 03:03:59 GMT
If-None-Match: "10c24bc-4ab-457e1c1f"
HTTP/1.1 304 Not Modified
```

Configure ETags

- The challenge comes in generating the Etag value
  - It must be quick to generate
  - It should not be dependent on the server generating the Etag
    - Doing this allows entities residing on different servers to have the same ETag