MetricSpot
SEO & Web Performance

Nº1

MetricSpot Executive Guides

WEB PERFORMANCE OPTIMIZATION

GUÍA PRÁCTICA PARA OPTIMIZAR EL RENDIMIENTO DE TU WEB

Acerca de este manual

Este manual forma parte de la serie **MetricSpot Executive Guides**.

En él encontrarás directrices que te ayudarán a **incrementar la velocidad de tu Web**, organizadas según el ámbito de aplicación:

- Servidor y Conectividad
- Hojas de estilos CSS
- Imágenes
- Tablets y SmartPhones
- Código HTML
- JavaScript
- Cookies

En ocasiones se incluyen **fragmentos de código**, a modo de ejemplo o para que puedas copiar y pegar en el código de tu Web. Aparecen dentro de cuadrados grises.

El último capítulo incluye enlaces a **herramientas gratuitas** de análisis de rendimiento Web, así como a **otros libros y guías** acerca del Rendimiento Web.

Acerca del autor

Ángel Díaz es consultor SEO/WPO y responsable técnico de MetricSpot. Puedes conectar con él vía Twitter.

Acerca de MetricSpot

En MetricSpot **creamos herramientas y manuales** que te ayudarán a optimizar tu Web para que posicione mejor en los buscadores, atraiga tráfico relevante y genere más conversiones.

También ofrecemos **servicios de consultoría especializada** a empresas de desarrollo Web y Marketing Online en los ámbitos del SEO, rendimiento Web y usabilidad.

Enlaces:

1. bit.ly/cc-by-sa-es

Utiliza GZIP y DEFLATE

GZIP comprime los datos en el servidor antes de enviarlos al navegador del usuario. Es muy fácil de configurar, y todos los navegadores y servidores modernos lo soportan.

Hay **determinados archivos** que es mejor no comprimir (como las imágenes) pero gran parte del contenido de un sitio es simple texto (HTML, CSS, JavaScript, JSON, XML, etc.) y conviene comprimirlo.

En **servidores Apache** se activa editando el archivo **.htaccess** añadiendo estas líneas:

```
<IfModule mod_deflate.c>
# Filtramos los tipos de contenido
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/x-httpd-php
AddOutputFilterByType DEFLATE application/x-httpd-fastphp
AddOutputFilterByType DEFLATE image/svg+xml

# Quitamos los navegadores que dan problemas con GZIP
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

# Nos aseguramos de que los Proxy no muestren contenido equivocado
Header append Vary User-Agent env=!dont-vary
</IfModule>
```

Activa el Caché/Expiración

Hay algunos tipos de archivos, los llamados **contenidos estáticos** (CSS, JavaScript, imágenes, etc.) que no suelen cambiar en mucho tiempo.

Cada vez que se carga una página el navegador tiene que descargar todos los archivos. Podemos ahorrarle tiempo al usuario y **evitar solicitudes innecesarias** al servidor permitiendo al navegador cachear archivos.

Para ello, primero tenemos que **separar todos los contenidos estáticos** en ficheros aparte y después activar el cacheado editando el archivo **.htaccess** añadiendo estas líneas:

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresDefault "access plus 600 seconds"
  ExpiresByType image/x-icon "access plus 604800 seconds"
  ExpiresByType image/jpg "access plus 604800 seconds"
  ExpiresByType image/jpeg "access plus 604800 seconds"
  ExpiresByType image/png "access plus 604800 seconds"
  ExpiresByType image/gif "access plus 604800 seconds"
  ExpiresByType application/x-shockwave-flash "access plus 604800 seconds"
  ExpiresByType text/css "access plus 604800 seconds"
  ExpiresByType text/javascript "access plus 604800 seconds"
  ExpiresByType application/x-javascript "access plus 604800 seconds"
  ExpiresByType text/html "access plus 600 seconds"
  ExpiresByType application/xhtml+xml "access plus 600 seconds"
</IfModule>

<IfModule mod_headers.c>
  <FilesMatch "\\.(ico|jpeg|jpg|png|gif|swf|css|js)$">
    Header set Cache-Control "max-age=604800, public"
  </FilesMatch>
  <FilesMatch "\\.(x?html?|php)$">
    Header set Cache-Control "max-age=600, private, must-revalidate"
  </FilesMatch>
</IfModule>
```

Con las instrucciones de arriba le estamos diciendo al navegador que cachee los **contenidos estáticos durante una semana** (604800 segundos) y los **contenidos dinámicos durante cinco minutos** (600 segundos).

Si estás en versión de desarrollo o **aplicando cambios a la Web**, deberías desactivar el caché.

Coloca los CSS arriba y los Scripts abajo

Debemos colocar los **CSS dentro del <head>** porque si los colocamos abajo la página se empezará a renderizar sin estilos hasta que se descargue el archivo.

Por el contrario, los archivos **JavaScript deberían ser cargados al final de la página** porque si no bloquean el renderizado mientras se descargan y se ejecutan. El lugar donde deben colocarse es justo **antes de la etiqueta de cierre </body>**.

Un ejemplo de una página con los CSS y JS bien emplazados:

```
<!DOCTYPE html>
<html lang="es">

  <head>
    <meta charset="UTF-8">

    <title>Mi Página Web</title>
    <meta name="description" content="La META DESCRIPTION de mi página." />

    <!-- Enlace a la hoja de estilos CSS -->
    <link rel="stylesheet" href="style.css" media="all" >

  </head>

  <body>

    <h1>El Título de mi Página</h1>
    El contenido de mi página

    <!-- Enlace al JavaScript -->
    <script src="script.js"></script>

  </body>

</html>
```

Comprime los CSS

Al igual que ocurre con el código HTML, es buena práctica usar **indentación**, **tabulaciones** y **comentarios** en los CSS para mantener el código legible:

```
/* Inicio de la cabecera */

.cabecera {
  width: 960px;
  margin: 0 auto;
}

/* Inicio del cuerpo */

.cuerpo {
  width: 960px;
  margin: 0 auto;
  padding: 10px;
  color: #555555;
}
```

El navegador puede **prescindir de ellos**, por lo que comprimir las hojas de estilos **utilizando herramientas automatizadas** hará que la descarga y análisis del CSS sea mucho más rápido.

Además, algunas herramientas **combinan elementos con propiedades iguales**, comprimen los códigos de color, etc. reduciendo aún más el tamaño de los archivos:

```
.cabecera,.cuerpo{width:960px;margin:0 auto;} .cuerpo{padding:10px;color:#555;}
```

HERRAMIENTAS:
CSS Compressor & Minifier [1]
CSS Compressor [2]
CSS Minifier [3]
YUI Compressor [4]
Clean CSS [5]
Code Beautifier [6]
CSS Beautify [7] - (Proceso inverso)
CSS Lint [8] - (Validador de CSS)

4. JavaScript

Externaliza todos los Scripts

Poner todos los **estilos CSS en un archivo externo** y enlazarlo desde el **<head>** es una buena práctica porque permite el cacheado y ahorra solicitudes y ancho de banda durante la navegación por nuestra Web.

Además el código quedará **más limpio y ocupará menos**, haciendo que las descarga del HTML sea más rápida.

Aplaza la ejecución de Scripts

Los archivos JavaScript deben colocarse **abajo del todo**, justo antes de la etiqueta de cierre **</body>** para evitar que bloquee la descarga de otros elementos.

Otra solución es utilizar el **atributo defer** que indica al navegador que **posponga la descarga de un <script>**. Sólo debe utilizarse para Scripts que no tengan **document.write**.

```
<script defer type="text/javascript" src="script.js" ></script>
```

Carga los Scripts asincrónicamente

El **atributo async** permite que el Script pueda ser **descargado en paralelo** de forma asincrónica, sin que bloquee el análisis del resto de la página.

```
<script async type="text/javascript" src="script.js" ></script>
```

¡Cuidado! El código se ejecutará en cuanto se termine de descargar.

Esto hace que si estamos descargando un archivo de **funciones que utilizan jQuery** y todavía no se ha descargado la librería de jQuery en sí, **el Script dará error** porque no se habrán declarado todavía las funciones que utiliza.

Utiliza Sprites

Una Web con **muchas imágenes** puede tardar mucho tiempo en cargar y genera **multitud de solicitudes HTTP**.

Los Sprites son **conjuntos de imágenes agrupadas** en un solo archivo con el fin de reducir el número de peticiones al servidor y **disminuir tiempo de carga**.

MetricSpot utiliza varios Sprites:

Las imágenes se posicionan **mediante CSS**:

```
.toplist {
  width: 20px;
  height: 20px;
  background: url("http://www.metricspot.com/img/topsprite.png") 0 0;
}

.toplist:hover {
  background: url("http://www.metricspot.com/img/topsprite.png") -20px 0;
}
```

HERRAMIENTAS:
Sprite Pad [7]
Sprite Me [8]
Sprite Cow [9]
Texture Packer [10]
Stitches [11]