KEEPING THIN - Front end performance guidelines

We need to understand each other code, that’s a fact. But we also need to keep our site as efficient and fast as we can.

Don’t block visual content

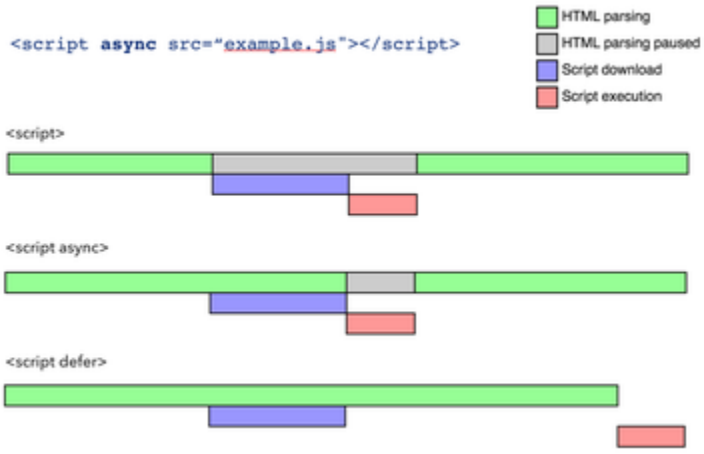
Layout - If we locate scripts before HTML code or CSS style assets, we’ll be blocking the page rendering,
processing JS functions, that should be executed after DOM ready.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Title</title>
    <meta name="description" content="description" />
    <meta name="viewport" content="width=device-width, minimum-scale=1, initial-scale=1" />
    ...
    <link rel="shortcut icon" href="favicon.gif" type="image/gif">
    <link rel="stylesheet" href="css/styles.css" type="text/css" />
    ...
    <!-- additional meta data here -->
  </head>
  <body>
    <header> ... </header>
    <main> ... </main>
    <footer> ... </footer>

    <script src="js/js.js"></script>
  </body>
</html>
```

Third-party content

Always call third-party content using “async” or “defer”.



Avoid document.write()

Third-party scripts sometimes use document.write() to inject and load scripts. This is particularly true of older services that haven't been updated in some time. Thankfully, many third-parties offer an option to asynchronously load themselves, which allows third-party scripts to load without blocking the display of the rest of the content on the page.

https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/loading-third-party-javascript/#avoid_documentwrite

Unify, minify and obfuscate

- **Join all js and css calls in just one per page or less.**
- **Make the code as tiny as is possible removing all spaces, comments and making shorter the name of all variables.**
- **Don't do this by yourself. It's better if you automate this process using less compiler and yui-compressor with gradle.**

```
lessc {
    sourceDir "WebContent/realestate/portal/css/less"
    include "base.less"
    exclude "imports/**/*"
    destinationDir = "WebContent/realestate/portal/css"

    //OPTIONS

    options.compress = true
    options.minify = true
}
```

Images - Light and less

It's very important compress and optimice images. But is also very important to reduce img requests.
If you respect this rules we'll be ok:

- **Make all .jpg images progressive.**
- **If there are few colors, it is better to use PNG.**
- **Use sprites or font-face for icons and logos to reduce requests.**

- **Use data-uri for small images with only one use.**

Reduce domain lookups

The Domain Name System (DNS) maps hostnames to IP addresses. When you type URL into the browser, the browser contacts a DNS resolver that returns the server's IP address. DNS has a cost; typically it takes 20 to 120 milliseconds for it to look up the IP address for a hostname. The browser cannot download anything from the host until the lookup completes.

Specify sizes for images and tables

Especificar el tamaño de los elementos que requieren un espacio reservado como las imágenes o las tablas, evita al browser la necesidad de un recálculo y redibujo cuando se terminan de cargar los contenidos. De esta manera se evita una mala experiencia visual a medida que se van cargando los elementos de la página.

Sin especificar tamaños:

Contamaños especificados:

Cache content

Los recursos estáticos, como imágenes, webfonts, scripts, css, etc. se deben cachear en el browser. De esta forma cada vez que se vuelva a utilizar ese recurso el browser no va a tener que descargarlo.
Por default los browsers ya cachean contenido estático, pero es nuestro trabajo asegurarnos de que el tiempo de expiración del caché de esos recursos no sea demasiado corto y en cambio manejar la actualización de los elemento estáticos modificando los nombres de los archivos para obligar al browser a volver a descargarlos.

Otra forma de aprovechar la cache del browser es recurriendo a CDNs públicos para importar librerías de uso común, como jquery, que muy probablemente haya sido descargada previamente por el usuario en otros sitios.*

Ejemplo:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

> ** Tener en cuenta que en algunos casos el tiempo de respuesta de los CDN de terceros puede ser lento y en esos casos, aún sin aprovechar la cache del browser, termina siendo más eficiente utilizar nuestro propio server para traer ese recurso.*

Optimally order the components of the page

Es una de los puntos más complejos demanejar, en especial en nuestro caso que tenemos implementaciones de productos y mejoras de forma constante. Pero la mejor manera de conservar un buen rendimiento es ordenando los componente de forma inteligente.

Idealmente en cada pagina del portal habría que importar en la mitad superior del documento solamente los recursos que infieren directamente en la interfaz visible, y todos aquellos componentes, elementos, scripts y estilos que no interfieran en la interface básica bisible debería invocarse en la parte inferior del documento para evitar bloqueo en el tiempo de renderizado del DOM.

Use modern CSS and valid markup

Mantener el código actualizado es escencial y relativamente sencillo. Los browsers constantemente están incluyendo mejoras y modificaciones basándose en los estándares más modernos para mejorar la eficiencia. Si utilizamos métodos antiguos para maquetar nuestra interfaz, probablemente estemos desaprovechando esas mejoras de los navegadores.

Choose your user-agent requirements wisely

Se trata de olvidarse del pixel-perfect y elegir sabiamente los dispositivos y navegadores que preferimos priorizar. De esta manera evitamos desperdiciar recursos con condicionales y media querys y destinamos todos los recursos a procesamiento de nuestra interfaz y el contenido útil.

También es importante aprovechar los recursos que nos brindan los distintos dispositivos y evitar sobreescribir el aspecto todos los elementos. Por ejemplo, en lugar de importar webfonts se puede aprovechar la propieded de font-family que permite indicar fuentes alternativas para que en cada dispositivo o SO se vea la opción más cercana y adecuada sin necesidad de descargar nada nuevo.

A big set of tools and recommendations:
https://github.com/zenorocha/browser-diet/wiki/References