**Estándar y Buenas Practicas Frontend**

|  |  |
| --- | --- |
| **Fecha** | 23/04/2018 |
| **Responsables** | Javier Padrón |
| **Objetivo** | Dar una visión de que estándares y buenas prácticas deben ser implementadas en los desarrollos frontend para incrementar su calidad y reusabilidad. |
| **Versión** | 1.0 |

Contenido

[1. Introducción 2](#_Toc512352824)

[2. Comentarios sobre el Código 2](#_Toc512352825)

[2.1 Comentarios en HTML 2](#_Toc512352826)

[2.2 Comentarios en JavaScript 2](#_Toc512352827)

[2.3 Comentarios en CSS 2](#_Toc512352828)

[3. Estándares HTML y CSS 3](#_Toc512352829)

[3.1 Declaración de DOCTYPE 3](#_Toc512352830)

[3.2 Definición de metadatos 3](#_Toc512352831)

[3.2 Verificación de Etiquetas HTML cerradas: 4](#_Toc512352832)

[3.3 Uso de minúsculas en etiquetas: 4](#_Toc512352833)

[3.4 Usar el Atributo alt en las imágenes: 4](#_Toc512352834)

[3.5 Usar el Atributo TITLE en las etiquetas <a> 5](#_Toc512352835)

[3.6 Usar una indentación consistente 5](#_Toc512352836)

[3.7 No usar estilos CSS incrustados en el HTML 5](#_Toc512352837)

[3.8 Usa etiquetas semánticas y evita colocar muchos Divs 6](#_Toc512352838)

[3.9 Codificación de la página HTML 7](#_Toc512352839)

[3.10 Reutilización de Código 7](#_Toc512352840)

[3.11 Organización de Archivos 7](#_Toc512352841)

[3.11 Tamaños estándar para Media Querys en CSS 8](#_Toc512352842)

[4. Mejoras para el rendimiento de aplicaciones Web 9](#_Toc512352843)

[4.1 Análisis de rendimiento general 9](#_Toc512352844)

[4.2 Compresión de Imágenes PNG / JPG 9](#_Toc512352845)

[4.3 Minificación de archivos JS 10](#_Toc512352846)

[5. Pruebas según resolución, navegadores, dispositivos 10](#_Toc512352847)

[5.1 Conocer información sobre comportamiento del usuario 10](#_Toc512352848)

[5.2 Herramientas para probar nuestra aplicación: 11](#_Toc512352849)

[6. Conclusiones 11](#_Toc512352850)

# 1. Introducción

El siguiente documento ha sido desarrollado con el objetivo de detallar estándares y buenas prácticas de desarrollo para aplicaciones web. El uso de la implementación de estas prácticas permitirá tener un código de mayor calidad, mejor organizado y con mayor mantenibilidad en el tiempo.

# 2. Comentarios sobre el Código

Los comentarios en el código fuente de los desarrollos web es una práctica que permite al desarrollador entender mejor las partes que conforman el proyecto.

### 2.1 Comentarios en HTML

Un desarrollador puede colocar comentarios donde él lo desee, una idea general sobre donde colocarlos es:

* En el inicio y final de cada sección del HTML, esta práctica nos ayudara a entender dónde comienza y termina el código escrito.
* Antes de nuestros script JavaScript para tener una descripción sobre ellos.
* Antes de nuestros archivos CSS para tener una descripción sobre ellos.

<!-- Comentario en HTML, puede contener varias líneas -->

### 2.2 Comentarios en JavaScript

Un desarrollador puede colocar comentarios donde él lo desee, una idea general sobre donde colocarlos es:

* Al inicio del JS para colocar información sobre el autor, versión, utilidad del JavaScript creado, fecha de creación.
* Al inicio de cada función para describir el comportamiento de la misma

En JavaScript dos tipos de comentarios permitidos, los comentarios en línea que comienzan con una doble barra: //, y los comentarios multilínea, que comienzan con /\* y terminan con \*/.

// Comentario de una sola linea

/\* Comentario multilinea\*/

### Comentarios en CSS

Un desarrollador puede colocar comentarios donde él lo desee, una idea general sobre donde colocarlos es:

* Al inicio del CSS para colocar información sobre el autor, versión, utilidad del CSS creado, fecha de creación.
* Al inicio de cada grupo de CSS para describir de forma general a que elemento HTML modifica.

En CSS existen dos tipos de comentarios, los comentarios en línea que comienzan con una doble barra: //, y los comentarios multilínea, que comienzan con /\* y terminan con \*/.

// Comentario de una sola linea

/\* Comentario multilinea\*/

# 3. Estándares HTML y CSS

### 3.1 Declaración de DOCTYPE

La declaración del DOCTYPE es un estándar usado al construir un documento HTML. Al ser usado los navegadores que lean esta declaración interpretarán el documento con el lenguaje y versión definidos.

La etiqueta !DOCTYPE es colocada de la siguiente forma:

**<! DOCTYPE html >**

### 3.2 Definición de metadatos

Los metadatos son usados para dar información a los buscadores sobre el contenido que posee una página en particular, estas serán de gran importancia para describir el sitio web y posicionarlo en los buscadores.

Las etiquetas recomendadas a usar dentro de nuestros sitios web son:

<head>  
<meta charset="UTF-8">  
<meta name="description" content="Descripcion">  
<meta name="keywords" content="Palabras, Clave ">  
<meta name="author" content="Nombre de la Empresa">  
<meta name="viewport" content="width=device-width, initial-scale=1.0">  
</head>

**Keywords:** Las palabras clave deben estar separadas por “,” y no deben superar un total de 155 palabras. El recomendado es de 25-30.

**Description:** La descripción no debe ser mayor a 320 palabras.

**Viewport:** permite que la página se ajuste al ancho del dispositivo desde el cual el usuario está viendo la página web.

### 3.2 Verificación de Etiquetas HTML cerradas:

Es importante validar que el código posee un estándar y si todas las etiquetas se encuentran cerradas. Si las etiquetas no se cierran correctamente se corre el riesgo de que los scripts o la presentación en diferentes dispositivos puedan sufrir discrepancias en sus funcionabilidades.

Para la validación podemos usar herramientas online como:

[**https://validator.w3.org/**](https://validator.w3.org/)

### 3.3 Uso de minúsculas en etiquetas:

Se deben usar minúsculas en los nombres de etiquetas, atributos y valores de forma de mejorar la visibilidad del código, esto no afecta en absoluto su comportamiento en el navegador.

**Mal:**

<SECTION>

<h1>Web App Design</H1>

<P>Contenido</P>

<section>

**Bien:**

<section>

<h1>Web App Design</h1>

<p>Contenido</p>

<section>

### 3.4 Usar el Atributo alt en las imágenes:

Las imágenes deben incluir siempre el atributo **alt**. Este es colocado en las imágenes para proporcionar un texto alternativo visible de no ser cargada la imagen. Al ser descriptivo da información adicional al usuario y a los navegadores, para entender que describe la imagen.

Dicho valor debe ser lo suficientemente descriptivo al contenido. Si la imagen no tiene ninguna relevancia (como un icono, por ejemplo), es recomendable seguir incluyendo el atributo y asociarlo al contenido al cual corresponde.

**Mal:**

<img src="img/logo.png">

**Bien:**

<img src="img/logo.png" alt="Logo - Web App Design">

### 3.5 Usar el Atributo TITLE en las etiquetas <a>

El uso del atributo **title** en los links o cualquier etiqueta HTML debe ser usado cuando se desee dar información adicional al usuario sobre el elemento. En general cuando un usuario se posiciona con el mouse sobre el elemento HTML, el mostrara la descripción colocada en el **title**.

**Uso:**

<a title="Biografía de Ana Martín" href="ana.html">Ana Martín</a>

### 3.6 Usar una indentación consistente

Indentar significa utilizar espacios a la derecha de cada línea con el objetivo de mejorar la legibilidad del código en los desarrollos realizados.

Un código desarrollado con mala indentación o sin ella carece en gran medida de legibilidad, por lo cual puede volverse más complicado de mantener y entender.

Es posible realizarla mediante tabulaciones (Tab) o espacios (1, 2, o los que se prefieran) existe una referencia sobre cómo realizar identación desarrollada por Google [guía de estilos de Google](https://google.github.io/styleguide/htmlcssguide.html#Indentation) que recomienda dos espacios.

**Mal:**

<div>

<h1>Web App Design</h1>

<ul>

<li>Elemento 1</li>

<li>Elemento 2</li>

<li>Elemento 3</li>

</ul>

</div>

**Bien:**

<div>

<h1>Web App Design</h1>

<ul>

<li>Elemento 1</li>

<li>Elemento 2</li>

<li>Elemento 3</li>

</ul>

</div>

### 3.7 No usar estilos CSS incrustados en el HTML

Incrustar elementos CSS dentro de nuestro HTML vía la etiqueta **style**es una forma sencilla de dar estilos a los elementos HTML, que inicialmente puede parecer la indicada pero que a largo plazo puede convert irse en un gasto de tiempo innecesario.

Cuando se usan CSS incrustados en el HTML o inline, para realizar mantenimientos sobre la aplicación, se tendrá que identificar cada archivo donde se encuentra estos estilos CSS y modificar individualmente cada estilo CSS.

Igualmente el usar esta práctica no permite reutilizar los estilos CSS, sino que individualiza los estilos al área que se está modificando.

Si se usa hojas de estilo externas, donde se definen todas las clases a usar en nuestro proyecto, le permitirá al desarrollador ubicar de forma sencilla donde esta cada CSS debido a que están en un lugar común preestablecido establecido.

**Mal:**

<p style="color: red;">Información</p>

**Bien:**

<p class="alerta">Información</p>

**En el archivo css**

.alerta{

color: red;

}

### http://www.desarrolloweb.com/articulos/images/html5/semantica-html.png3.8 Usa etiquetas semánticas y evita colocar muchos Divs

Las etiquetas semánticas estructurales  sirven para que los motores de búsqueda establezcan con exactitud qué partes del contenido HTML corresponden a cada una de las partes típicas de un sitio. Generalmente en cualquier documento se tiene una cabecera, un cuerpo y un pie de página, elementos que definen la estructura representados por diversas etiquetas:

Actualmente a nivel de HTML existen etiquetas que funcionan igual que los divs, pero que a su vez permiten dar un mejor entendimiento semántico para el navegador para entender el HTML. Para ello, podemos utilizar los elementos estructurales de HTML5 de forma adecuada, como:

**Mal:**

<div id="encabezado"></div>

<div id="navegacion"></div>

<div id="contenido">

<div class="articulo"></div>

<div class="articulo"></div>

</div>

<div id="pie"></div>

**Bien:**

<header></header>

<nav></nav>

<main>

<article></article>

<article></article>

</main>

<footer></footer>

### 3.9 Codificación de la página HTML

Se debe especificar correctamente la codificación de caracteres, la cual permitirá la correcta visualización del contenido del proyecto web, incluyendo tildes y caracteres especiales como ñ, indicando al navegador cómo debe interpretarlos.

Para especificarla, se debe añadir la siguiente línea al header de la página:

**Uso:**

<meta charset="UTF-8"/>

El sistema de codificación UTF-8 al ser el más extendido y usado.

### 3.10 Reutilización de Código

La reutilización de código en las aplicaciones permite usar múltiples veces el mismo código en diferentes partes del proyecto, haciendo más sencillo de mantener y más organizado.

Es importante la utilización de comentarios al desarrollarlas para entender cuáles son las características que se pueden reutilizar y cuál es su función dentro de la aplicación.

Las consecuencias de la poca reutilización del código traen como consecuencia duplicación de código y un mayor costo de tiempo para entender y mantener el proyecto posteriormente, además de un costo de refactorización importante cuando se desea mejorar la aplicación.

### 3.11 Organización de Archivos

Al momento de organizar los archivos de las aplicaciones a desarrollar pueden usarse diferentes estructuras.

#### 3.11.1 Según su tipo y finalidad

Los archivos pueden ser almacenados dentro de carpetas en los cuales primero se tome como prioridad su tipo, y posteriormente su finalidad:

* dao
* controllers
* models
* images
* js
  + controllers
    - index.js
  + jquery.min.js
  + bootstrap.min.js
* css
  + style.css
  + bootstrap.min.css
* index.aspx

#### 3.11.2 Según Componentes

Los archivos se almacenar en carpetas dependiendo de a que componente pertenecen, permitiendo separar más fácilmente unos de otros.

* dao
* register
  + register.html
  + register.js
  + register.css
* users
  + user.html
  + user.js
  + user.css
* assets
  + js
    - jquery.min.js
    - bootstrap.min.js
  + css
    - jquery.min.css
    - bootstrap.min.css
* public

### 3.11 Tamaños estándar para Media Querys en CSS

Los **Media Querys** en CSS permiten adaptar la presentación del contenido a características del dispositivo. Debido a la gran cantidad de dispositivos en el mercado, no existen unas medidas estándar, mas sin embargo existen medidas usadas por herramientas de desarrollo como **Bootstrap o Foundation** que sirven para establecer la grilla de trabajo de la página web, como referencia puede ser usado los siguientes **Media Querys**

/\* Large Devices, Wide Screens \*/

@media only screen and (max-width : 1200px) {}

/\* Medium Devices, Desktops \*/

@media only screen and (max-width : 992px) { }

/\* Small Devices, Tablets \*/

@media only screen and (max-width : 768px) {}

/\* Extra Small Devices, Phones \*/

@media only screen and (max-width : 480px) { }

/\* Custom, iPhone Retina \*/

@media only screen and (max-width : 320px) {}

Para más documentación: **https://developer.mozilla.org/es/docs/CSS/Media\_queries**

# 4. Mejoras para el rendimiento de aplicaciones Web

### 4.1 Análisis de rendimiento general

Existen diversas herramientas que permiten medir el nivel de optimización de la aplicación web en diferentes renglones. Para realizar este análisis se puede usar la siguiente herramienta online:

[**https://developers.google.com/speed/pagespeed/insights/**](https://developers.google.com/speed/pagespeed/insights/)

A través de esta herramienta se obtendrá un reporte de estadísticas sobre sugerencias de optimización basadas en: compresión de imágenes, archivos archivos JS, CSS, HTML a minificar, almacenamiento en cache, entre otras.

### 4.2 Compresión de Imágenes PNG / JPG

La compresión del peso de las imágenes en las aplicaciones web es un punto importante a la hora de la medición del rendimiento de los tiempos de carga.

Siempre se recomienda usar extensiones **.JPG** en el caso de que estas no necesiten ser transparentes, esto debido a que este formato de imagen posee un peso inferior a las demás extensiones.

En caso de requerir que las mismas sean **.PNG** por temas de transparencia es necesario realizar un tratamiento de las mismas a través de alguna herramienta que reduzca su tamaño. Por ejemplo:

[**https://tinypng.com/**](https://tinypng.com/)

**TinyPNG** es una herramienta gratuita con la que se puede reducir el peso de archivos de tipo **PNG** y **JPEG**. **TinyPNG** toma las imágenes y revisa que colores son similares y los combina, esta técnica es llamada quantización y al usarla se reduce el número de colores.

Al finalizar los archivos PNG de 24 bits pueden ser convertidos en archivos mucho más pequeños de 8 bits, la diferencia a nivel visual es realmente mínima, pero el tiempo de carga de tu sitio web disminuirá debido a la reducción realizada.

### 4.3 Minificación de archivos JS

La reducción de archivos **.JS** en nuestra aplicación es un punto importante a la hora de la medición del rendimiento de los tiempos de carga de la aplicación web. Entre mayor minificación del código posean los archivos, mejor rendimiento tendrá la aplicación.

Una herramienta sencilla a usar para la reducción de dichos archivos es:

[**https://jscompress.com/**](https://jscompress.com/)

# 5. Pruebas según resolución, navegadores, dispositivos

### 5.1 Conocer información sobre comportamiento del usuario

Es importante cuando se desarrolla una aplicación web asegurarnos que este se visualice de forma correcta en diferentes navegadores, dispositivos, resoluciones de pantalla y sistemas operativos. Esta tarea es especialmente complicada y desafiante, dadas las combinaciones de dichas variables.

Sin embargo es importante conocer estadísticamente que está usando exactamente el usuario para tener una brecha de uso más corta.

Para realizar esta tarea podemos verificar en la siguiente web las estadísticas generales para conocer el comportamiento general y así poder establecer pruebas más específicas y asegurarnos que estén cubiertos los targets más importantes.

**http://gs.statcounter.com/**

### 5.2 Herramientas para probar nuestra aplicación:

#### 5.2.1 Pruebas Generales en navegadores en el computador:

Para asegurarnos que la aplicación web desarrollada funciona de forma adecuada podemos probarla en los diferentes navegadores instalados en el computador, laptop o dispositivo mobile.

Esta alternativa es válida y nos sirve para validar al momento del desarrollo que todo esté funcionando de manera adecuada, más sin embargo se puede profundizar con otras herramientas.

Esto presenta como desventaja las versiones que poseen los navegadores y el sistema operativo usado por el Tester.

Dado que sistema operativo tiene sus propias versiones de los navegadores, las aplicaciones podrían verse de forma diferentes en cada uno de ellos.

#### 5.2.2 Pruebas tipo Screenshots:

Este tipo de pruebas permite tomar **screenshots** de una aplicación web, pudiendo ser visualizada desde una resolución, varios sistemas operativos y navegadores en el mercado, de forma de poder visualizar el aplicativo web a través de una imagen capturada.

Esta prueba es complementaria dado que no permite visualizar el aplicativo, pero no probar que esté funcionando correctamente con sus componentes JS.

Existen plataformas pagas y otras gratuitas con limitaciones, una aplicación que realiza esta tarea es:

**http://browsershots.org**

#### 5.2.3 Pruebas tipo Sistema Operativo + Browser:

Este tipo de aplicaciones, permiten visualizar desde acceso remoto o virtualización de sistemas operativos como se presenta al usuario la aplicación web en diferentes sistemas operativos y diferentes navegadores.

Existen plataformas pagas y otras gratuitas con limitaciones, una aplicación que realiza esta tarea es:

[**https://www.browserling.com/**](https://www.browserling.com/)

# 6. Conclusiones

Haciendo uso de las buenas prácticas, estándares mostradas en este documento y las pruebas generales se puede garantizar que la aplicación va a tener una mayor organización e independencia entre sus componentes, mejor rendimiento en navegadores, buenos tiempos de carga, mayor facilidad a la hora de mantenerlo y que cualquier desarrollador sea capaz de entenderlo e implementar nuevas características.

Estas características son vitales para tener un proyecto más prolijo con un código de mayor calidad.