Buenas prácticas en la programación

En general

- El desarrollo en HTML **es lento**. No esperen hacer un sitio con foro, portada, noticias, auto-administrable, segura y que regule toda la parte contable de una empresa en un mes. ¡Un desarrollo de estas características toma fácilmente 6 meses!
- Eviten nombres de archivos con espacios o caracteres que no sean parte del alfabeto inglés. Asimismo, en la gran mayoría de los casos el servidor estará montado en Linux, el cual diferencia entre mayúsculas y minúsculas. HoLa.txt no será lo mismo que hola.txt.
- Google lo sabe todo: en el 99% de los casos, lo que estemos haciendo también lo ha hecho otra persona. Ocupen términos relevantes al problema que quieran solucionar. Ejemplo:
  - *Saber cómo aplicar la propiedad position de CSS:*
    - Escibir en Google: “CSS position” (sin comillas)
  - *El Javascript no funciona y el mensaje de error no es claro:*
    - En Firefox, es muy probable que la “Consola de Errores” nos tire un error. Copiamos y pegamos ese error en Google.
  - *No funciona una consulta en PHP:*
    - Revisar el uso de la función en PHP.net
    - Si PHP tira un error, copiar y pegar el error en Google.
  - *La base de datos tira número de error pero no especifica de qué se trata:*
    - MySQL errno XXXXX, donde XXXXX es el número del error.
- Todas las modificaciones a un sistema existente, tendrán que ser probadas primero en un ambiente protegido, nunca en caliente.
- Indenten su código: es de más fácil lectura.
- No olvidar este esquema:

Ítem	Participación Servidor	Participación Cliente
HTML	La genera	**Lo interpreta**
Javascript	La genera	**Lo ejecuta**
CSS	La genera	**Lo interpreta**
PHP	**Lo ejecuta**	*Ninguna*
Base de Datos	**Lo ejecuta**	*Ninguna*

HTML

- Construyan su código primero para todos los navegadores y finalicen su desarrollo con Internet Explorer: es más fácil hacer la página compatible primero con todos los navegadores y posteriormente con Internet Explorer. Además, según recientes estadísticas globales, estos últimos dos años IE ha bajado su participación global de un 75% a un 50%.
- En lo posible, apéguense al estándar más actual. (HTML 4.01).
- No comenten mucho el HTML: ocupa espacio en la transferencia y al usuario final le podrá parecer lenta la carga.
- Todo input debería tener el atributo maxlength. Esto evita en cierta manera la inclusión de código malicioso.

Javascript

- No confíen en que el usuario tenga habilitado Javascript. Si es necesario validar un formulario en Javascript, háganlo también en PHP. Cualquier navegador permite desactivar Javascript.
- Para el ingreso de caracteres, no ocupen el evento onKeyPress: si no se suelta la tecla y se presiona enter o se pulsa en Aceptar, la validación nunca ocurre. En cambio, se recomienda hacer la validación una vez que el usuario ha llenado todo el formulario y pulsa en Aceptar.
- Javascript no muestra errores: al más mínimo error simplemente no funciona. Revisen si está bien escrito, ya que Javascript **diferencia** entre mayúsculas y minúsculas.

- Ocupen la “Consola de Errores” de Mozilla Firefox o bien instalen la extensión “Firebug” para Firefox. Con el uso de estas herramientas, el 99% de los problemas relacionados con Javascript los podrán encontrar fácilmente.
- Siempre terminen las sentencias con punto y coma (“;”). Es de más fácil lectura posterior. Además, resulta más fácil la minificación.
- Cuando el Javascript es único para una página y no resulta muy grande, inclúyanla en el HTML. De lo contrario, si es muy usado o pesa mucho, déjenlo en un archivo separado.
- Los tags de `<script>` pueden ir en cualquier parte del documento: en lo posible, cuando se incluyan scripts grandes, déjenla al final de la página para que de esta forma, primero cargue la página y al final el Javascript.

CSS

- Es mucho más óptimo dejar todo el CSS en un archivo separado que incluido en el HTML.
- No comenten mucho su CSS: ocupa espacio en la transferencia y al usuario final le parecerá lenta la carga. Asimismo, traten de evitar espacios y retornos de carros innecesarios.
- En ciertos casos, se puede acortar el código hexadecimal siempre y cuando cumpla con la regla de que los pares de colores RGB sean iguales. Así, el color #1111FF se puede acortar como #11F. Lo que no se puede hacer es acortar este código: #1244BB, ya que el primer par no es igual. Tampoco se puede acortar #CCCCCC a #C, sino que a #CCC.
- El último elemento de cada bloque de sentencias no necesariamente debe terminar con punto y coma.
- La extensión “Firebug” para Firefox ayuda muchísimo a la hora de depurar su código en CSS.
- NOTA: ¡La herencia es importante!

PHP

- PHP.NET es su amigo N° 1.

- OJO con los inputs: ¡Son **muy** vulnerables a inyección SQL! Si la consulta fuera:

  `SELECT id FROM usuarios WHERE nombre = 'unreal4u'`

  Con la inyección SQL podría quedar:

  `SELECT id FROM usuarios WHERE nombre = '';DELETE * FROM usuarios;`

  Lo cual es una consulta válida: en el input en vez de escribir

  `unreal4u`

  Sólo habría tenido que escribir:

  `';DELETE * FROM usuarios;`

- Aunque no es lo más recomendable, podemos "callar" errores o excepciones en nuestro script anteponiendo "@" (sin comillas) a la función que provoca el error a costa de rendimiento. Ejemplo:

```
$fp = @fopen('hola.txt','w');
```

  No tirará error si hola.txt no existe. Sin embargo, la buena práctica sería primero revisar si el archivo existe y sólo si existe tratar de abrirlo para su escritura:

```
if (file_exists('hola.txt')) $fp = fopen('hola.txt','w');
```

Bases de datos

- Planifiquen su consulta: primero créanla para un caso en específico, corriéndola directamente desde un programa gráfico de conexión directa como SQLYog o AquaDataStudio, luego la ordenan y finalmente la traspasan a PHP haciendo sólo los ajustes necesarios.
- Sólo rescaten los campos que necesitan, se gasta menos memoria y es más claro a la vista: se saben a priori los nombres de los campos a rescatarse. Ejemplo:

  `SELECT * FROM usuarios;`

  Es poco claro y estamos rescatando campos que no nos interesan.

  `SELECT id, nombre, grupo FROM usuarios;`

  Es más claro a la vista y sólo rescatamos campos que nos interesan.

- Al trabajar con fechas, es mucho más rápido y fácil formatear la fecha completa en la consulta que trabajándola con PHP. Revisar la función date_format() de MySQL.
- Ocupen JOIN, ya sea INNER, LEFT o RIGHT. Es más rápido que ejecutar 20 consultas que se podría hacer en uno.
- Para todos los campos de tipo varchar, char, text o cualquiera que involucre texto, deben llevar apóstrofe tanto al iniciar como al finalizar el texto.
- Si la base de datos no la creamos nosotros y no hay total certeza de la integridad del mismo y necesitamos buscar una cadena de texto, cambien a minúscula ambas cadenas: ocupen la función LOWER del estándar SQL-3. En PHP, se llama strtolower(). Sin embargo, es preferible ocupar la versión de MySQL puesto que PHP **no** convierte a minúsculas los caracteres “áéíóúñ”. MySQL **sí**.
- Importante para evitar dolores de cabeza: MySQL **no** diferencia entre “á” y “a”. **Sí** entre “n” y “ñ”.