Desarrollo Web Responsive con HTML5 y CSS3 + PHP, MySQL, JavaScript-JQuery y AJAX

- HTML y CSS
- Diseño Responsive con HTML5 y CSS3
- PHP: Variables, Operadores y Estructuras de Control
- Tratamiento de Cadenas, Arrays, Fechas… con Funciones PHP
- Base de Datos MySQL
- JavaScript y JQuery
- AJAX

Luis Orlando Lázaro Medrano

Introducción a HTML

- HTML es un lenguaje de **marcas**, que basa su sintaxis en el uso de **etiquetas**
- Las etiquetas normalmente se presentan en 2 partes: apertura **<etiqueta>** y cierre **</etiqueta>**, aunque podemos encontrar elementos que prescindan del cierre y lo integren en la misma etiqueta **<etiqueta />**
- Un Documento HTML está divido en varias partes, delimitadas por etiquetas:

```
<html>
    <head> Encabezado, etiquetas de índole informativo: título, palabras clave, estilos...
    </head> Final del Encabezado
    <body> Cuerpo, parte del documento que será mostrada en el navegador
        ……
    </body> Final del Cuerpo

</html>
```

- Las mayúsculas o minúsculas son indiferentes al escribir etiquetas HTML, pero como otros lenguajes no son tan permisivos (por ejemplo PHP) nos acostumbraremos a escribir las etiquetas con la misma sintaxis.
- Existen etiquetas que por su complejidad necesitan atributos, que se incluirán dentro de la apertura de la etiqueta, por ejemplo si queremos darle formato a un texto la etiqueta quedaría así ** … **
- Guardaremos los archivos con extensión .html o .htm y aunque podemos usar un abanico amplio de caracteres para darle nombre a los archivos, no usaremos ni acentos, ni caracteres de puntuación, ni espacios, ni ningún otro carácter que no sean letras de la a-z en minúsculas, números de 0-9 y el guion medio “-” o bajo “_”

Etiquetas Básicas HTML

<title> </title>	Titulo de la página	Es el título que aparece en la pestaña del navegador. Va siempre dentro de la cabecera <head> … </head>
<body > </body>	Cuerpo de la Página	Atributos: bgcolor (color de fondo), text (color por defecto del texto), link (color enlaces), vlink (color enlaces visitados), alink (color enlaces activos), background="imagen.gif" (imagen de fondo)
<!– -->	Comentarios	Sirve para indicar comentarios, anotar aclaraciones 'privadas'... Lo que se escribe dentro de esta etiqueta es ignorado por el navegador y no se muestra en la página
	Párrafo	Atributos: align=center, left o right
  o 	Salto de Línea	
	Centrar el texto	No deja tanto espacio como el párrafo
<blockquote> </blockquote>	Sangría	
o	Negrita	
o	Cursiva	
<u> </u>	subrayado	
<h1> </h1>, <h2>…	Encabezados	Dentro de los estilos HTML por defecto existen predefinidos 6 encabezados identificados como h1..h6
...	Lista sin Numerar	Dentro del metemos tantos como líneas queramos crear. Lista numerada sería y lista de definición <dl><dt>término a definir</dt><dd>definición</dd></dl>
	Línea Horizontal	Atributos: width (anchura) size (altura) align (center, left, right. Alineacion) noshade (línea sin efecto de sombra)
	Imagen	Atributos: src (ruta de la imagen), border (tamaño del borde) height y width (alto y ancho) alt (texto que se muestra un texto al pasar el cursor) align (bottom,middle,top,left,right) hspace…
<table><tr>... </table>	Tabla	Atributos: width (ancho) height (alto) border (tamaño del borde) align (alineación de la tabla) cellspacing (espaciado entre celdas) cellpadding (relleno de celdas)
<tr> </tr>	Fila	Tanto las filas <tr>, como las celdas-columnas normales <td> o de encabezado <th> van dentro de la etiqueta anterior <table> .. </table> y podemos insertar tantas como deseemos, también podemos combinarlas simplemente indicando el numero de filas **rowspan** o columnas **colspan** que juntamos. Los Atributos vienen a ser comunes a todos: align (alineación horizontal) valign (alienación vertical), width y height (alto y ancho), bgcolor (color de fondo)…
<th> </th>	Encabezado Tabla	
<td> </td>	Celda (columna)	
 	Enlace	El Enlace puede ser a una página: http://, a un correo: mailto:..@.. a un marcador: #marcador y el atributo target marca el destino target: _blank (The response is displayed in a new window or tab), _self (The response is displayed in the same frame, this is default), _parent (The response is displayed in the parent frame), _top (The response is displayed in the full body of the window), Framename (The response is displayed in a named iframe)
<pre> </pre>	Pre formateado	Muestra el texto en su formato original, tal cual está escrito. Respeta espacios, saltos de línea y los retornos utilizados

Introducción a HTML. Página de Prueba

```
<html>
    <head>
        <title>Pagina de Prueba Curso PHP</title>
    </head>
    <body>
        <p><h1>Bienvenido: Encabezado 1</h1></p>
        <p>Estás en la página <b>Curso Desarrollo Aplicaciones Web 2.0</b>.</p>
        <p>Esto es </br>salto de línea dentro de un <i>párrafo</i></p>
        <p align="left">Texto alineado a la <b><i>izquierda</i></b></p>
        <div align=“center">
            <p>División de 3 párrafos centrados</p>
            <p>Texto con <sup>Superíndice</sup></p>
            <p>Texto con <sub>Subíndice</sub></p>
        </div>
        <p align=“right”><font size=4 color="red">Este texto está en rojo</font></p>
        <p>Letra a acentuada: &aacute;</p>
     </body>
</html>
```

Caracteres especiales

Caracteres especiales del HTML 2.0							
Á	Á	À	À	ö	ö	ä	ä
É	É	È	È	ü	ü	ë	ë
Í	Í	Ì	Ì	Ã	Ã	ï	ï
Ó	Ó	Ò	Ò	Ñ	Ñ	â	â
Ú	Ú	Ù	Ù	Õ	Õ	ê	ê
á	á	à	à	ã	ã	î	î
é	é	è	è	ñ	ñ	Ã	Ã
í	í	ì	ì	õ	õ	Ñ	Ñ
ó	ó	ò	ò	Ø	Ø	Õ	Õ
ú	ú	ù	ù	ø	ø	Ø	Ø
Ä	Ä	Â	Â	Ð	Đ	ø	ø
Ë	Ë	Ê	Ê	ð	ð	Ð	Đ
Ï	Ï	Î	Î	ß	ß	ð	ð
Ö	Ö	Ô	Ô	ö	ö	ß	ß
Ü	Ü	Û	Û	ü	ü	ÿ	ÿ
Þ	Þ	þ	þ	Æ	Æ	æ	æ

Básicos			
<	<	>	>
&	&	"	"

Otros caracteres especiales			
×	×	¢	¢
÷	÷	€	€
“	“	™	™
”	”	‰	‰
Œ	Œ	ƒ	ƒ
‡	‡	†	†
¨	¨	º	º
±	±	´	´
»	»	«	«

Del HTML 3.2	
¼	¼
½	½
¾	¾
©	©
®	®
ª	ª
²	²
³	³
¹	¹
¯	¯
µ	µ
¶	¶
·	·
°	°
¸	¸
¿	¿
	Tabulador
¡	¡
£	£
¥	¥
§	§
¤	¤
¦	¦
¬	¬

Estilos CSS

- Un estilo, es un conjunto de características identificadas con un nombre y que aplicamos a un elemento. En este caso los Estilos CSS los utilizamos para dar formato a documentos HTML y XML, independizando el contenido del formato.
- Funciona a base de reglas, y tiene dos partes: el selector y la declaración.
- Tipos de selectores:
  - Clase: pueden aplicarse como atributo class a cualquier elemento HTML, su nombre comienza por un punto y luego cualquier combinación alfanumérica
  - ID: se aplica al elemento cuya ID coincide con este nombre, se aplica a un atributo de una etiqueta, debe comenzar con almohadilla
  - Etiqueta: se aplica a una etiqueta completa
  - Compuesta: Afecta a 2 o más etiquetas, lo emplearemos por ejemplo para los estados de los botones, vínculos…:
    - a -> el selector para el vínculo, le afecta a todos los estados del vínculo si no se especifican por separado.
    - a:link ->igual que el anterior
    - a:hover -> es el estado de cuando el cursor se posa sobre el vínculo, lo que pongas aquí se verá al poner el cursor en el enlace.
    - a:visited -> afecta a los enlaces que ya fueron visitados. Si haces clic sobre algún enlace queda marcado como visitado siempre
    - a:active -> es mientras tienes presionado el botón del ratón con el cursor sobre el enlace.
    - a:focus -> esta no es una propiedad de los enlaces, pero en tal caso sirve para cuando un enlace es seleccionado (sea por mouse o con la tecla TAB)

Referencias CSS:

- Las unidades de medida en CSS pueden ser:
  Unidades absolutas:
  - in: pulgadas (inches). Una pulgada=2.54 cm
  - cm: centímetros
  - mm: milímetros
  - pt: puntos. Un punto=1/72 de pulgada.
  - pc: picas. Una pica=12 puntos

  Unidades relativas:
  - em, igual a la altura (font-size) de la letra del elemento en el que se usa. Por ejemplo si queremos doble línea sería 2em
  - ex, igual a la altura de la letra x minúscula del elemento en el que se usa. Normalmente em=1.5ex
  - px, depende de la resolución y el tamaño de la pantalla
- Los posibles valores de color son:
  Valores: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow,
  Por ejemplo: <DIV STYLE="color: red;">Texto cualquiera</DIV>
- **Hexadecimal**: <DIV STYLE="color: #ff0000;">Texto cualquiera</DIV>
- **RGB (enteros)**: <DIV STYLE="color: rgb(255,0,0);">Texto</DIV>
  (porcentaje): <DIV STYLE="color: rgb(100%,0%,0%);">Texto</DIV>
- *A partir de CSS 3 se pueden usar colores con transparencia:*
  RGBA (255, 125, 0, 0.5) dónde los tres primeros valores son números que corresponden con los valores de rojo, verde y azul. Y el cuarto es un número entre 0 y 1. 0 totalmente transparente, 1 sería totalmente opaco y 0.5 sería 50

Estilos CSS en las tablas:

- Para dejar espacios entre las celdas de las tablas existen las instrucciones HTML cellpadding (relleno de celda) y cellspacing (margen de la celda). Dado que no existe equivalente CSS de esas propiedades la solución es usar:
  - table {
    border-collapse:collapse;
    }
  - td {
    padding:0px; margin:0px;
    }

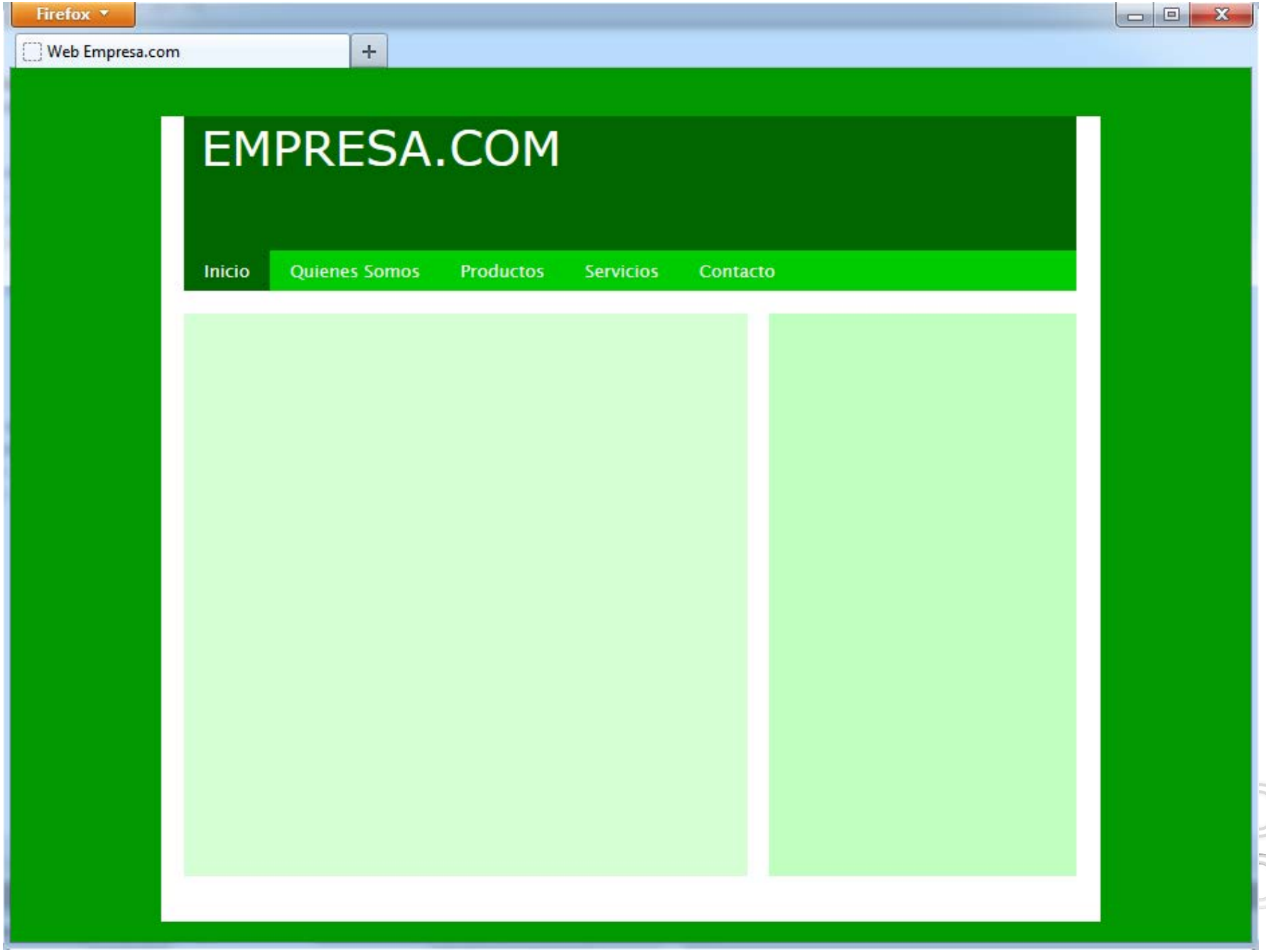
** Tenéis un listado completo de los selectores, clases y elementos HTML en los apuntes*

Pseudo-clases

- La pseudo-clase **:first-child** selecciona el primer elemento hijo de un elemento. Por ejemplo:
  - p em:first-child { color: red; }
    * Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent odio sem, tempor quis, auctor eu, tempus at, enim. Praesent nulla ante, ultricies id, porttitor ut, pulvinar quis, dui. *
    El selector p em:first-child selecciona el primer elemento que sea hijo de un elemento y que se encuentre dentro de un elemento . Por tanto, en el ejemplo anterior sólo el primer se ve de color rojo.
- La pseudo-clase **:lang** En el siguiente ejemplo sólo el texto en español estaría en rojo
  Lorem ipsum dolor sit amet... Lorem ipsum dolor sit amet...
  - p:lang(es) { color: red; }
- Otras pseudo-clases :link, :visited, :hover, :active, :focus

Pseudo-elementos:

- El pseudo-elemento **:first-line** permite seleccionar la primera línea de texto de un elemento. La siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:
  - p:first-line { text-transform: uppercase; }
- Se pueden combinar varios pseudo-elementos de tipo :first-line, el siguiente ejemplo pone la primera linea de un DIV en rojo y la linea de un parrafo en mayúsculas si el parrafo está dentro del DIV aplicamos los estilos al mismo elemento (Primera Línea… 2ª…):
  - div:first-line { color: red; }
  - p:first-line { text-transform: uppercase; }
- **:first-letter** permite seleccionar la primera letra de la primera línea de texto de un elemento. La siguiente regla CSS muestra en mayúsculas la primera letra del texto de cada párrafo:
  - p:first-letter { text-transform: uppercase; }
- Los pseudo-elementos **:before y :after** se utilizan en combinación con la propiedad content de CSS para añadir contenidos antes o después del contenido original de un elemento. Las siguientes reglas CSS añaden el texto ***Capítulo -*** delante de cada título de sección <h1> y el carácter **.** detrás de cada párrafo de la página:
  - h1:before { content: "Capítulo - "; }
  - p:after { content: "."; }


Firefox
Web Empresa.com
EMPRESA.COM
Inicio
Quienes Somos
Productos
Servicios
Contacto

Código HTML de Web Empresa

```
<html>
  <head>
        <title>Web Empresa.com</title>
        <link href="HojaEstilos.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
      <div id="cuerpo">
          <div id="Encabezado">
              <p> EMPRESA.COM</p>
          </div>
          <div id="menu">
              <ul>
                  <li class="menu_seleccionado"><a href="index.html">Inicio</a></li>
                  <li><a href="quienes_somos.html">Quienes Somos</a></li>
                  <li><a href="productos.html">Productos</a></li>
                  <li><a href="servicios.html">Servicios</a></li>
                  <li><a href="contacto.html">Contacto</a></li>
              </ul>
          </div>
          <div id="bloque1"></div>
          <div id="bloque2"></div>
      </div>
  </body>
</html>
```

Estilos CSS de Web Empresa

```
body {
	font-family: "Lucida Sans Unicode", "Lucida Grande", sans-serif;
	font-size: small;
	color: #333;
	background-color: #090;
}
#cuerpo {
	height: 600px;
	width: 700px;
	margin-right: auto;  margin-left: auto;
	background-color: #FFF;
}
#Encabezado {
	background-color: #060;
	height: 100px;
	width: 95%;
	margin-right: auto; margin-left: 17px;
	font-family: Verdana, Geneva, sans-serif;
	font-size: 36px;  color: #FFF;
}
#bloque1 {
	background-color: #D5FFD5;
	float: left;
	height: 419px;
	width: 60%;
	margin-top: 17px; margin-left: 17px;
}
#bloque2 {
	background-color: #BFFFBF;
	height: 419px;
	width: 229px;
	float: right;
	margin-top: 17px; margin-right: 18px;
}
```

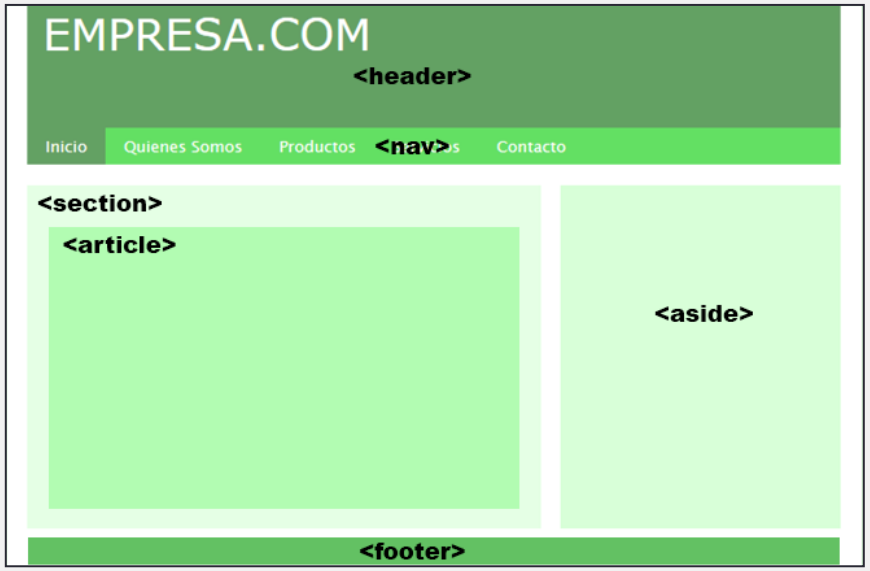
```
#menu {
	background-color: #0C0;
	height: 30px;
	width: 95%;
	margin-right: auto;
	margin-left: 17px;
}
#menu ul {
	margin: 0px;
	padding: 0px;
}
#menu ul li {
	float: left;
	list-style-type: none;
	display: block;
	height: 30px;
}
#menu ul li a {
	text-decoration: none;
	color: #FFF;
	padding-top: 5px;
	padding-right: 15px;
	padding-left: 15px;
	display: block;
	padding-bottom: 5px;
	line-height: 20px;
}
#menu ul li a:hover {
	background-color: #030;
}
.menu_seleccionado {
	background-color: #060;
}
```

HTML5

- El HTML5 es la quinta revisión del lenguaje "*básico de programación de las Páginas Web"* el HTML. Nace de la necesidad de estandarizar la incorporación de nuevos elementos a las páginas web (audio, sonido…), las nuevas capacidades dinámicas de las páginas (JavaScript, BD…) y para facilitar el diseño de webs.
- HTML5 también es un termino de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y JavaScript. Tiene hasta su logo
- Para decirle a nuestra página web es HTML tendremos indicarle el DOCTYPE (una declaración para hacer que el navegador, entienda qué es lo que va a recibir, y cómo debe procesarlo). Tendremos que escribir **<!DOCTYPE html>** antes de la etiqueta <html>.
- Las etiquetas de HTML5 van a ser semánticas (van a tener nombre propio) y se van a comportar como o . Es decir en vez de poner pondremos sólo <header>. Aún así podemos seguir empleando también.
- Existen muchas etiquetas HTML5 https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos y la mayoría las emplearemos durante todo el curso. De momento vamos a empezar con las más usadas:
  - <header> Se refiere al encabezado de la página Web
  - <nav> Está diseñado para que coloquemos el menú principal
  - <section> Se usa para dividir el contenido de las páginas por conceptos: texto, video
  - <article> Se usa para agrupar los datos dentro de las secciones
  - <aside> Se usa para anuncios, noticias… cualquier contenido prescindible de la web
  - <footer> Aquí colocaremos el pie de página y todo lo que contenga.

HTML

HTML5



Código y Estilos HTML5

```
<!DOCTYPE html> <!-- HTML5 -->
<html>
  <head>
    <title>Web Empresa.com</title>
    <link href="HojaEstilos.css" rel="stylesheet" type="text/css" />
    <meta charset="utf-8" />
  </head>
  <body>
    <header>
        <p> EMPRESA.COM</p>
    </header>
    <nav>
        <ul>
            <li class="menu_seleccionado"><a href="index.html">Inicio</a></li>
            <li><a href="quienes_somos.html">Quienes Somos</a></li>
            <li><a href="productos.html">Productos</a></li>
            <li><a href="servicios.html">Servicios</a></li>
            <li><a href="contacto.html">Contacto</a></li>
        </ul>
    </nav>
    <section id="izquierda">
        <article>Contenido de la parte izquierda</article>
    </section>
    <aside>Banner publicitario</aside>
    <footer>Curso Logroño IFES &copy;</footer>
  </body>
</html>
```

```
body {          font-family: Lucida;font-size: small; color: #333;
                background: url(imagenes/verde.jpg) #FFF;
                height: 600px; width: 700px;
                margin-right: auto;  margin-left: auto; }
header {        background-color: #060;
                height: 100px; width: 95%;
                margin-right: auto; margin-left: 17px;
                font-family:Verdana; font-size:36px;  color:#FFF; }
nav {           background-color: #0C0;
                height: 30px; width: 95%;
                margin-right: auto; margin-left: 17px; }
nav ul {        margin: 0px; padding: 0px; }
nav ul li {     float: left; list-style-type: none; display: block;
                height: 30px; }
nav ul li a {   text-decoration: none; color: #FFF;
                padding:top:5px;right:15px;left:15px;bottom:5px;
                display: block; line-height: 20px; }
nav ul li a:hover { background-color: #030; }
.menu_seleccionado { background-color: #060; }
#izquierda {    background-color: #D5FFD5;
                float: left;
                height: 219px; width: 60%;
                margin:top:17px; left:17px; bottom:17px;
                padding:5px;}
```

```
aside {  background: #BFFFBF;
         height:219px;width:209px;
         float: right;
         margin:top:17px;
         m-right:18px; bottom:17px;
         padding:5px; }
footer{  clear:both;
         background-color: #060;
         height:20px; width:95%;
         text-align:center;
         margin-right:auto;left:17px;
}
```

Diseño Responsive

- El Diseño Web **Responsive** o Diseño Web **Adaptativo** (Responsive Web Design) es una filosofía de diseño web que tiene como objetivo la correcta visualización de una misma página en distintos dispositivos: PCs, tablets, móviles…
- Se basa en adaptar dinámicamente el diseño web en función de la resolución de la pantalla del visitante. De esta forma adaptamos nuestras webs a dispositivos móviles sin necesidad de tener dos sitios separados y también podemos adaptar la web a resoluciones grandes para ocupar toda la pantalla.
- Algunos ejemplos de webs Responsive: http://mediaqueri.es/

Diseño Responsive

- Por lo tanto para crear nuestra web Responsive hablaremos siempre de medidas relativas. Tanto para los contenedores: header, nav, section, div… en los que indicaremos anchuras en % (a partir de ahora hablaremos también de anchura máxima), como para los tipos de letra, margin y padding… dónde emplearemos como medidas píxeles, em (relación con la letra m) o ex (con la letra x), por ejemplo: 1em es el tamaño por defecto =16px=12pt=100%, igualmente podemos usar 0.8em, 1.5em…
- Por lo tanto para adaptar nuestra web pondremos como medidas:
  - **header**: max-width:95%; height: 100px; margin-left: 0.2em; bottom: 0.1em; font-size: 3em;
  - **nav**, será como header pero con 25px de altura: max-width:95%; height: 25px; margin- bottom: 0.1em y left: 0.6em (en el caso de margen izquierdo ponemos 0.6em que equivale a 0.2em del header por que el encabezado tiene una separación por defecto mayor con respecto al margen)
  - **section#izquierda** y **aside**, están flotando a izquierda y derecha. Bueno pues en HTML5 y CSS3 vamos a acostumbrarnos a usar display:inline-block; que viene a decir que vamos a colocar los contenedores en línea tratándolos como bloques, por lo tanto si tienen espacio se posicionarán unos al lado de los otros. Y además de eso les indicaré medidas de margin como al nav y en cuanto al tamaño les diré que ocupen la parte izquierda el 65% y la derecha el 30%
  - **footer**: tendrá medidas como nav pero una altura de 20px
- Con todo esto, tendríamos estructurada la página web, vamos a seguir con el menú:
  - Si queríamos un menú horizontal, hasta ahora estábamos acostumbrados a crearlo empleando listas sin ordenar (<ul>) y a indicar que sus elementos (<li>) flotasen a la izquierda, a partir de ahora emplearemos display:inline-block; como en el caso de izquierda y aside.
  - Y en el estilo que define el vínculo: nav ul li a, le diremos más o menos lo mismo que hasta ahora pero cambiando las medidas de padding, margin… a relativas: padding-top: 0.1em; right: 0.2em; left: 0.2em; bottom: 0.2em; display: block;
- El resto de los aspectos que se refieren al tipo de letra, colores de letra y fondo… se hace como hasta ahora: font-family:…, background-color: #030; color: #FFF; …

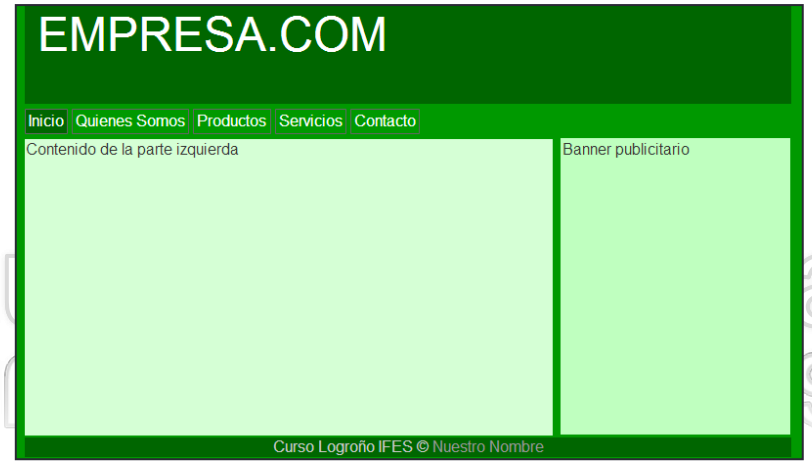
Diseño Responsive

- Otro elemento básico que tenemos que indicar el nuestro código HTML para hacerlo responsive es decirle que la web se adapte a lo que mida la pantalla de nuestro dispositivo, y eso lo hacemos empleando la etiqueta **meta viewport** (la cual pondremos en la cabecera, junto a las otras etiqueta meta)
  `<meta name="viewport" content="width=device-width, initial-scale=1" />`
  ** podríamos añadirle otras opciones como: maximum-scale=1… si lo considerásemos conveniente.*

con el scale=1 desactivamos el zoom automático, por ejemplo el de los iPhone

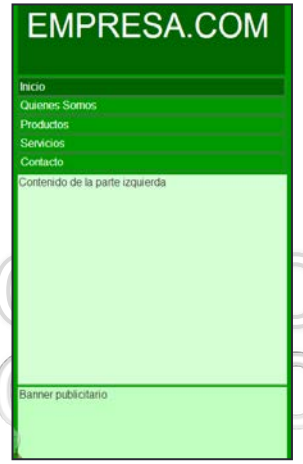
- Y otro aspecto que tenemos que tener muy presente es que por mucho que indique las medidas en porcentajes.. Habrá tamaños de pantalla en los que no quepa el menú horizontal… o las zonas izquierda y derecha. Por ejemplo el menú horizontal mide al menos 600 píxeles con lo que cualquier dispositivo con una resolución menor no podrá ver los menús al completo, perderá *servicios*, *contactos*… toda la parte derecha. Para evitarlo necesitamos decirle de alguna forma que cuando la resolución sea menor de 600 píxeles cambie el menú a vertical y que cuando la resolución sea menor de 460px ponga derecha debajo de izda.

Aspecto Normal



Aspecto para un resolución menor de 600 píxeles

Aspecto resolución menor de 460px



Diseño Responsive

- Para poder especificar distintas características CSS dependiendo de las medidas debemos emplear **Media Queries** que es un módulo de CSS3 que limita las hojas de estilo utilizando características del medio como ancho, alto y color.
- En nuestro caso para decirle que trate de forma distinta las medidas menores de 600px, pondremos: @media screen and (max-width:600px) { …. } y dentro indicaremos que cambia en los estilos, por ejemplo en el caso de los menús ya no queremos display:inline-block; para que los menús *floten* sino que los queremos que se coloquen en vertical con lo que cambiaremos el atributo a display:block;
  Y si también queremos que a partir de 460px, el aside (derecha) se coloque debajo de izquierda y ocupen ambos el 95% del tamaño, pondremos:


```
@media screen and (max-width:600px){
        nav {
                height: 100%;
        }
        nav ul li {
                margin-bottom: 0.1em;
                display:block;
        }
}
@media screen and (max-width:460px){
        #izquierda, aside {
                display:block;
                width: 95%;
                margin-left: 0.6em;
        }
}
```

Diseño Responsive. Código HTML de Web Empresa

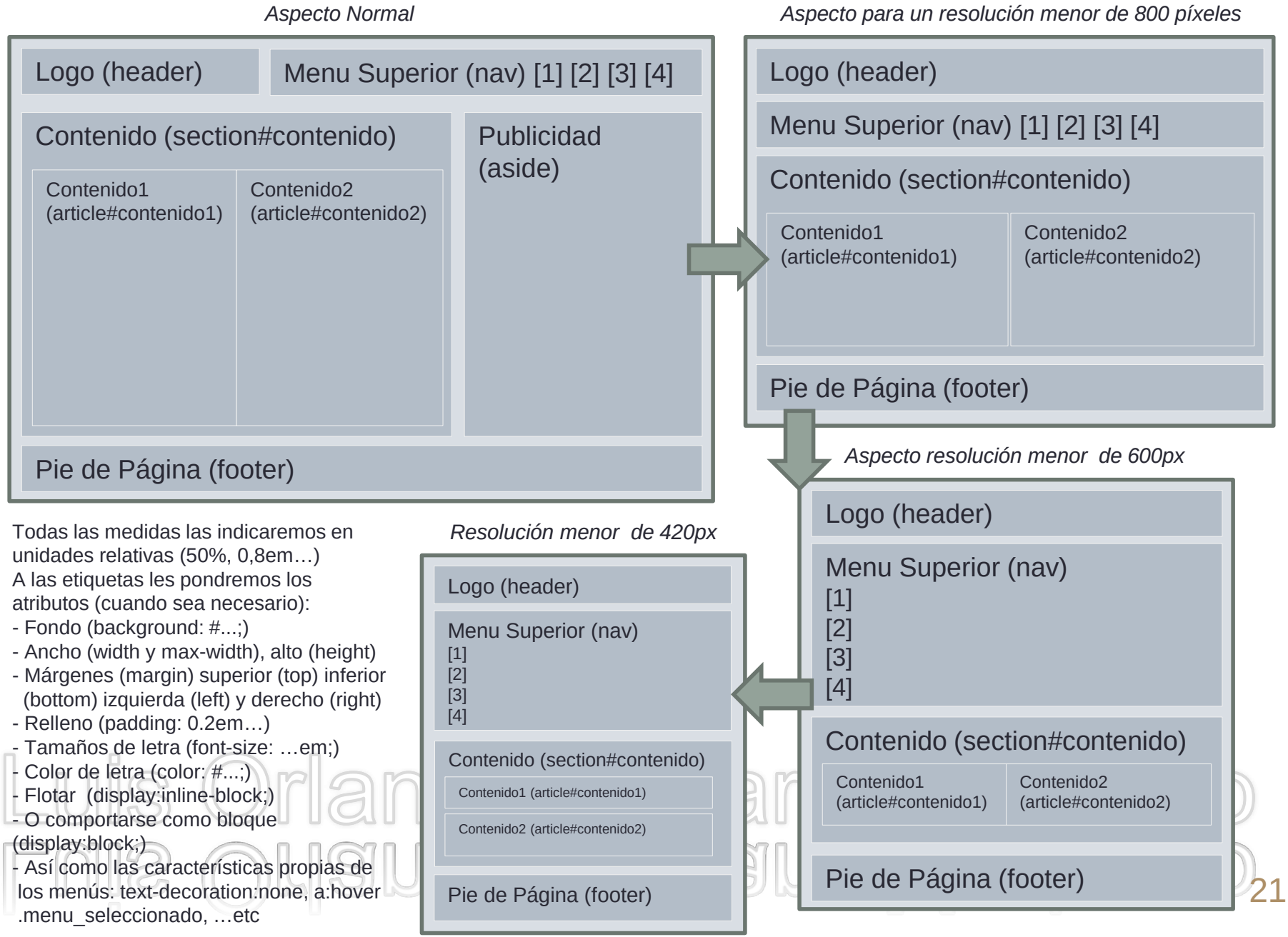
** Si el navegador es un Internet Explorer antiguo, hay alguna etiqueta de HTML5 que no funcionan, para evitarlo existe una web **html5shiv**, que permite generar las etiquetas de HTML5 como Javascript y como cada día aparece una cosa nueva mejor enlazar directamente con el archivo js y no descargarlo.*

```
<!DOCTYPE html> <!-- HTML5 -->
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Web Empresa.com</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link href="favicon.ico" rel="shortcut icon" type="image/x-icon">
    <link href="HojaEstilosResponsive.css" rel="stylesheet" type="text/css" />
    <!--[if lt IE 9]>
        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
  </head>
  <body>
    <header> EMPRESA.COM</header><!-- Encabezado en HTML5 -->
    <nav><!-- Menu en HTML5 -->
        <ul>
            <li class="menu_seleccionado"><a href="index.html">Inicio</a></li>
            <li><a href="quienes_somos.html">Quienes Somos</a></li>
            <li><a href="productos.html">Productos</a></li>
            <li><a href="servicios.html">Servicios</a></li>
            <li><a href="contacto.html">Contacto</a></li>
        </ul>
    </nav>
    <!-- Un contenido en HTML5 compuesto por 1 sección y un banner aside -->
    <section id="izquierda">
        <article>Contenido de la parte izquierda</article>
    </section>
    <aside>Banner publicitario</aside>
    <!-- Pie -->
    <footer>Curso Logroño IFES &copy;
        <a rel="author" href="http://www.nuestra-pagina.com" target="_blank">Web</a>
    </footer>
  </body>
</html>
```

Diseño Responsive. Estilos CSS de Web Empresa

```
* {       border:0; margin:0; padding:0;
}
/* Decimos que todos los elementos HTML5 se comporten como bloque */
header, nav, section, article, footer {  display:block;
}
body {  font-family: Verdana;font-size:1em; color: #333;
        background: url(imagenes/fondo_verde.jpg) #FFF ;
}
a { color: #999; text-decoration:none;
}
header { background-color: #060;
        height: 100px; max-width:95%;
        margin-left: 0.2em; margin-bottom: 0.1em;
        font-size: 3em;  color: #FFF;
}
nav {   height: 25px; max-width:95%;
        margin-left: 0.6em; margin-bottom: 0.1em;
}
section#izquierda, aside { background-color: #D5FFD5;
        display:inline-block;
        max-width:100%; height: 300px; width:65%;
        margin-left: 0.6em; margin-top: 0.2em; margin-bottom: 0.2em;
        padding:0.1em;
}
aside { background-color: #BFFFBF;
        margin-left: 0.3em;
        width:28%;
}
footer{ background-color: #060;
        height: 20px; width: 95%;
        text-align:center; color:#CCC;
        margin-left: 0.6em;
}
```

```
nav ul li { display:inline-block;
        list-style-type: none;
}
nav ul li a { border:0.1em solid #666;
        border-radius: 0.1em;
        color: #FFF;
        padding-top: 0.1em; right,left,bottom : 0.2em;
        display: block;
}
nav ul li a:hover { background-color: #030;
}
.menu_seleccionado { background-color: #060;
}

@media screen and (max-width:600px){
        nav {
                height: 100%;
        }
        nav ul li {
                margin-bottom: 0.1em;
                display:block;
        }
        aside {
                background-color: #BFFFBF;
                margin-left: 0.1em;
                width:28%;
        }
}
@media screen and (max-width:460px){
        #izquierda, aside {
                display:block;
                width: 95%;
                margin-left: 0.6em;
        }
}
```



Todas las medidas las indicaremos en unidades relativas (50%, 0,8em…)
A las etiquetas les pondremos los atributos (cuando sea necesario):
- Fondo (background: #...;)
- Ancho (width y max-width), alto (height)
- Márgenes (margin) superior (top) inferior (bottom) izquierda (left) y derecho (right)
- Relleno (padding: 0.2em…)
- Tamaños de letra (font-size: …em;)
- Color de letra (color: #...;)
- Flotar (display:inline-block;)
- O comportarse como bloque (display:block;)
- Así como las características propias de los menús: text-decoration:none, a:hover .menu_seleccionado, …etc

Diseño Responsive. Solución Base a la práctica anterior sin márgenes, ni relleno (padding)

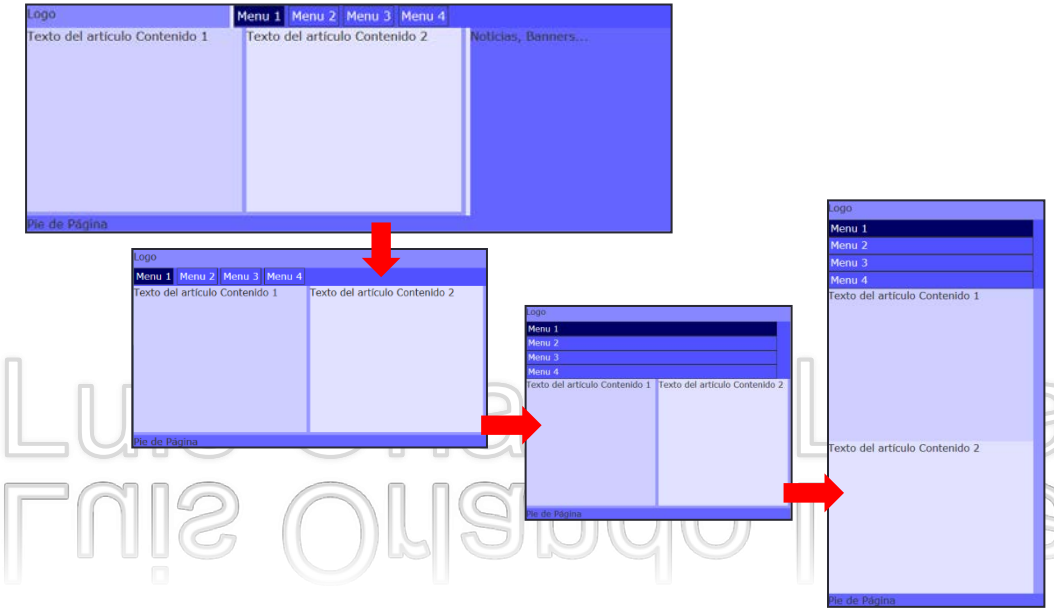
Index_practica3.html (Página Web HTML5)

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="estilos/EstilosPractica3.css">
    <title>Practica 3 - Responsive</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
</head>
<body>
    <header>Logo</header>
    <nav>
        <ul>
          <li><a href="#" class="menu_seleccionado">Menu 1</a></li>
          <li><a href="#">Menu 2</a></li>
          <li><a href="#">Menu 3</a></li>
          <li><a href="#">Menu 4</a></li>
        </ul>
    </nav>
    <section id="contenido">
        <article id="contenido1">Texto del artículo Contenido 1</article>
        <article id="contenido2">Texto del artículo Contenido 2</article>
    </section>
    <aside>
        Noticias, Banners...
    </aside>
    <footer>Pie de Página</footer>
    </body>
</html>
```

EstilosPractica3.css (Hoja de Estilos CSS)

```
@charset "utf-8";
*{  border:0; margin:0; padding:0;  }
body {
    font-family: Verdana, sans-serif;
    font-size: 1em; color: #333; background-color: #E1E1FF;
}
header, nav, section, aside, footer { display:block; }
header, nav{
    max-width:98%; width:30%;
    height:30px;
    background:#88F;
    display:inline-block;
    vertical-align:top;
}
nav{
    background:#5151FF;
    width:65%;
}
nav ul ...
section#contenido, aside{
    background:#9797FF;
    width:65%;max-width:95%;
    height:250px;
    display:inline-block;
    vertical-align:top;
}
aside{
    background:#6464FF;
    width:30%;
}
article#contenido1, article#contenido2{
    background:#CECEFF;
    width:49%;
    height:245px;
    display:inline-block;
}
article#contenido2{
    background:#E1E1FF;
}
footer{
    background:#6464FF;
    max-width:95.4%;heigh:20px;
}
@media screen and (max-width:800px){
    header, nav {display:block;width:95%;}
    aside {display:none;}
    section#contenido{width:95%;}
    footer{width:95%;}
}
@media screen and (max-width:600px){
    nav {height:100%;}
    nav ul li {display:block;width:95%;}
}
@media screen and (max-width:420px){
    section#contenido{height:100%;}
    article#contenido1, article#contenido2{
        display:block;
        width:95%;
    }
}
```

Omito los estilos del menú (ul li , ul li a…)



Diseño Responsive. Fuentes en HTML5

- Como hemos visto hasta ahora empleamos un reducido conjunto de fuentes por que el usuario debe tener instaladas las fuentes en su equipo, pda, móvil…
  Pero a partir de CSS2 (aunque se estandarizó con CSS3) podemos incorporar nuestras fuentes a nuestra página web, empleando el atributo @font-face
- Pero para hacerlo debemos tener nuestra fuente convertida en los formatos de todos los navegadores:
  - Fuentes WOFF (Web Open Font Format) Para casi todos los navegadores
  - Fuentes EOT (Embedded OpenType) Para Internet Explorer
  - Fuentes TTF (True Type) Para Firefox y Chrome
  - Y Fuentes SVG (Scalable Vector Graphics) Para Safari Mobile (iPhone, iPad)
- Por lo tanto para que funcionen nuestras fuentes sin problemas subiremos con nuestra web los archivos de fuentes en los 4 formatos, indicando en nuestra hoja de estilos dónde se encuentran y el nombre que queremos darle para referirnos a el:

```
@font-face {
    font-family: 'allerregular';
    src: url('../fuentes/aller_rg-webfont.eot');
    src: url('../fuentes/aller_rg-webfont.eot?#iefix') format('embedded-opentype'),
        url('../fuentes/aller_rg-webfont.woff') format('woff'),
        url('../fuentes/aller_rg-webfont.ttf') format('truetype'),
        url('../fuentes/aller_rg-webfont.svg#allerregular') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

** Podemos ponerle el nombre que queramos (sin espacios…)*

- Podemos crear y convertir nuestras fuentes desde varios sitios, los más populares http://www.google.com/fonts y http://www.fontsquirrel.com/tools/webfont-generator

Diseño Responsive. Audio y Video en HTML5

- Para incorporar Audio y Video en nuestras web, HTML incorpora 2 nuevas etiquetas que facilitan su uso, sin necesidad de recurrir a plugins externos como flash… <video></video> y <audio></audio>
- Y al igual que con las fuentes para que los videos funcionen en todos los navegadores debemos convertir los archivos de vídeo y audio en los formatos:
  - WebM: para Firefox, Chrome y Opera. Se puede agregar a Internet Explorer y Safari con un plugin
  - Ogg: para Firefox, Chrome y Opera. Se puede agregar a Safari con un plugin (no a IE)
  - MP4 y MP3: compatible con Internet Explorer, Safari y Chrome
  - WAVE: para Firefox y Safari
- Y además podemos acompañarlos de atributos, los más usados:
  - controls : Muestra los controles estándar de HTML5 (Play, Stop, volumen…)
  - autoplay : Hace que el audio se reproduzca automáticamente.
  - loop : Hace que el audio se repita automáticamente.
  - preload: Hace una precarga temporal (none, auto o metadata sólo los metadatos)
  - Otros atributos: muted, poster (imagen mientras carga el video) , width, height…
- Ejemplos:

```
<audio id="ejemplo_audio" controls>
    <source src="audio/ej-audio.mp3" type="audio/mpeg"/>
    <source src="audio/ej-audio.ogg" type="audio/ogg"/>
</audio>
<video id="ejemplo_video" controls preload>
    <source src="videos/ejemplo_video.mp4" type="video/mp4"/>
    <source src="videos/ejemplo_video.webm" type="video/webm"/>
    <source src="videos/ejemplo_video.ogv" type="video/ogg"/>
</video>
```

SEO (Search Engine Optimization) Posicionamiento en Buscadores

- Son un conjunto de técnicas que nos permiten mejorar la posición de la lista en la que nos encontramos tras una búsqueda (por ejemplo con google). El concepto es crear una estructura y contenidos óptimos para que resulten más amigables a los motores de búsqueda para que en vez de penalizarnos nos premien.
- Antes de nada tenemos que entender que existen 2 grandes tipos de buscadores:
  - Por **índices** (yahoo…) la información se divide en un árbol temático de categorías, subcategorías, administrado por personas con lo que para estar incluido en estos listados hay que solicitar el alta… por lo tanto no vamos a profundizar en este punto.
  - Por **motores de búsqueda** (google…) en este caso, aunque puede existir supervisión humana, la información, posición… se genera de forma automática, normalmente usando programas automáticos: robots, software spider (arañas) que escanean todas las páginas web. Y aquí es donde nos centraremos, sin profundizar ya que este tema daría para otro curso entero.
- Cada motor de búsqueda, funciona de una manera distinta (aunque con conceptos comunes). Por ejemplo Google premia las páginas que apuntan hacía la nuestra y otros motores como Bing valoran los Meta Tags, que son líneas de código, en el HEAD de nuestra página, que informan al motor acerca de nuestro contenido.
  Por ejemplo <meta name="description" content="Descripción de la web: …." /> esta etiqueta incluye la descripción que nosotros queremos dar a nuestra página.

* Esto es la teoría pero como se abusa tanto de las palabras clave incluidas en las etiquetas meta, ahora los buscadores penalizan nuestra página si encuentran demasiadas palabras repetidas o un texto de tamaño muy pequeño o con el mismo color que el fondo, etc, hay veces que incluso las eliminarán de la base de datos por pensar que son técnicas de promoción spam (malintencionadas).
* Y el uso de Acentos, Mayúsculas… es importante ya que hay buscadores como Altavista que busca palabras tal cual las escribe el usuario, otras en cambio como google no discrimina por acentos….

SEO. MetaTags

- Existen muchas reglas para posicionar nuestra página correctamente. Vamos a empezar con las básicas: añadir en la cabecera, en el <head>
  - El **Título**: debe describir correctamente la página, no debe ser muy largo entre 50 y 100 caracteres y ha de incluir palabras clave (logroño, desarrollo web…) ya que hay buscadores que sólo se fijan en el título para buscar. <title>….</title>
  - Las **Palabras Clave**: son palabras sueltas que describen una página web: las que pensamos que van a escribir los usuarios en los buscadores para encontrarnos.
  - Además de palabras también tenemos que indicar **Frases Clave**, son frases que van a escribir los usuarios para encontrarnos: desarrollo paginas web logroño… <meta name="keywords" content="palabra clave 1,palabra 2, frase 1, frase 2…">
  - **Descripción**, frase que describe el contenido de la página. Además será la descripción que muestre el buscador cuando nos encuentre. <meta name="description" content="Descripción de la web: …." />

  Todas ellas tienen que describir correctamente nuestra página pero sin abusar, sin repetirlas excesivamente ya que el motor de búsqueda podría pensar que estamos haciendo trampa (ponemos muchas palabras sin que tengan que ver con nuestra web, simplemente para que nuestra página salga la primera) cosa que nos penalizará
- Otras etiquetas que podemos incluir en el encabezado:
  - author: Indica el nombre del autor de la página (persona, empresa..)
  - copyright: Similar a la anterior pero indicamos quien posee los derechos de autor
  - subjetc: Viene a ser como el resumen de la descripción
  - generator: El programa con el que se ha hecho la página (editor HTML) Dreamweaver, Joomla, WordPress
  - language: Idioma del contenido. Esta etiqueta está ya obsoleta, en html5 se incluye junto a la etiqueta de apertura html: <html lang="es">

SEO. Otras Etiquetas

- Y otras etiquetas meta, no tan sencillas de comprender son:
  - revisit-after: esta etiqueta le indica al robot que vuelva a visitar la página pasados x días, meses… (weeks, days, months…) por que vamos a cambiar la página, o por que nos interesa…: <meta name="revisit-after" content="5 days" />
  - robots: mediante esta etiqueta le indicamos varias cosas al robot del buscador:
    - index – noindex: sirve para indicar si queremos permitir la indexación (incluirnos en índices)
    - follow – nofollow: para indicar si permitimos a los motores de búsqueda recorrer la web a través de los enlaces que encuentre del cuerpo de la pagina.
    - archive – noarchive: esto permite decir si queremos que el motor de búsqueda archive el contenido del sitio web en su caché interna. (*no lo muestra aunque lo google lo sigue haciendo*)
    - snippet – nosnippet: esta directriz no resulta muy útil, sirve para que el motor de búsqueda no muestre ninguna descripción de un sitio, sólo su título. Si utilizas nosnippet automáticamente defines un noarchive, por lo que la página tampoco se mostrará en caché.
    - all - none equivale a index, follow o al reves noindex, nofollow * Es HTML da igual mayúsculas-minúsculas

    Ejemplo: <META name="robots" content="INDEX,NOFOLLOW">
- Las etiquetas vistas hasta ahora eran las llamadas *meta names*, otras que ya hemos visto en alguna ocasión son las *meta http-equiv*, las cuales usamos para decirle al navegador como tiene que mostrar la información.
  Por ejemplo para decirle que conjunto de caracteres (símbolos latinos, arabes...) usar:
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  Y otra etiqueta, q podemos usarla para redirigir nuestra página pasados 30 segundos:
  <meta http-equiv="refresh" content="30;URL="http://www.otrapagina.com">

* Cosas que NO debemos hacer nunca:
- Poner un párrafo entre comentarios HTML con la descripción del sitio, o repitiendo palabras clave de la página
- Poner un texto del mismo color que el fondo (con lo que no se ve) con las palabras clave del sitio
- Repetir excesivamente las palabras clave en el texto.
- Y utilizar palabras clave que luego no figuren por ningún sitio en la página

SEO. Otras Consideraciones

- Los motores de búsqueda también tienen en cuenta otras cosas, entre ellas:
  - Es importante el texto del primer encabezado de nuestro sitio <h1>…</h1>
  - y el primer texto que aparece debajo. Que debe estar colocado cerca del <body> si es posible sin etiquetas en medio (como imágenes…).
  - También se valora que las imágenes siempre aparezcan con atributos alt="…" que usamos para indicar el texto a mostrar si falla la imagen
- Para solucionar este problema en HTML5 tenemos una nueva etiqueta llamada <hgroup> que usaremos cuando queramos incluir mas de un encabezado h1 (por ejemplo como título de los artículos de la página)

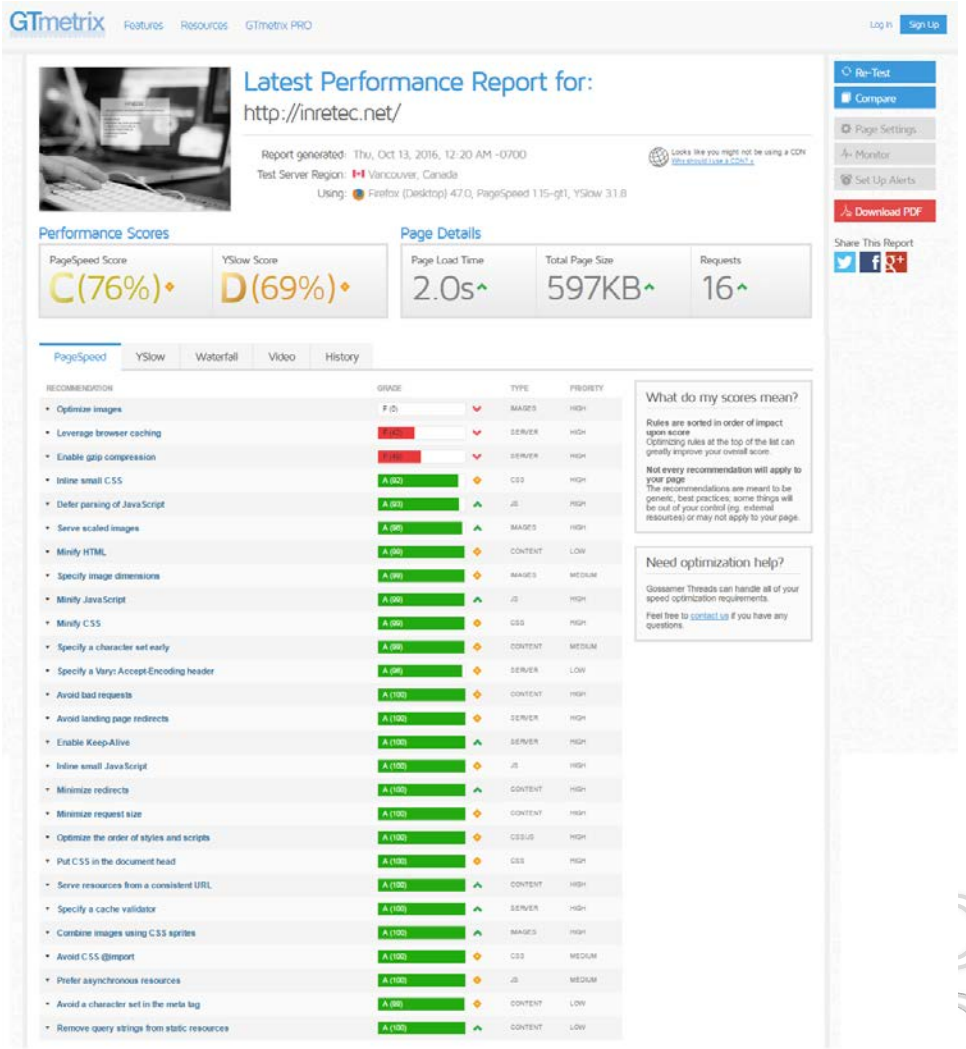
  <hgroup> <h1>Encabezado secundario</h1> </hgroup>

  De esta forma evitaremos ser penalizados por los buscadores
- Un ejemplo de una página web teniendo en cuenta las etiquetas meta…:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Empresa SL: Desarrollo de Páginas Web en Logroño con HTML5 y PHP </title>
   <link rel="stylesheet" type="text/css" href="estilos.css">
   <meta name="description" content="Empresa SL: Desarrollo de Páginas Web en…" />
   <meta name="keywords" content= "PHP, HTML5, Logroño, Desarrollo Web…" />
   <meta name="author" content= "Empresa SL" />
</head>
<body>
   <header>
   <h1> Empresa SL: Desarrollo de Páginas Web en Logroño con HTML5 y PHP </h1>
   Somos una empresa de Desarrollo de Páginas Web, ubicada en Logroño….
```

SEO. Informe de nuestra web

- Existen muchas aplicaciones para evaluar nuestra web, su velocidad de carga y varios aspectos necesarios para corregir nuestro posicionamiento SEO. Una de las más interesantes es: gtmetrix.com
- Y dado que google es el motor de búsqueda mas empleado, posee una serie de herramientas que nos permiten comprobar, que posición ocupamos en su buscador con respecto a webs de la competencia, que palabras de búsqueda suelen emplear los usuarios que acceden a nuestra página y similares… Para poder acceder a esa información, debemos registrarnos en google Analytics y Google Search Console para comprobar todas estas opciones.
- Una vez registrados Google nos dará un numero de Identificación (ID) y deberemos incluir un código javascript en nuestra web para que Google registre el tráfico de nuestro sitio web después de verificar la autoría del mismo.



SEO. Google Search Console

- Trafico de Búsqueda
  - ⇨ Análisis de Búsqueda:
    Activamos: ☑ Clics ☑ Impresiones ☑ CTR y ☑ Posición
    * **Clics**: recuento de los clics desde una página de resultados de búsqueda de Google , **Impresiones**: cantidad de enlaces a tu sitio web que ha visto un usuario en los resultados de búsqueda de Google, **CTR** (porcentaje de clics): recuento de clics dividido entre el recuento de impresiones y **Posición**: la posición media del primer resultado de tu sitio
    Revisar al menos 1 vez al mes, por que solo guarda los 3 últimos meses
  - ⇨ Enlaces a tu sitio: Si en enlaces de tu sitio, tenemos muchos enlaces no deseados anuncios... hay que tener cuidado por que nos pueden penalizar
  - ⇨ Enlaces Internos: Vínculos de nuestras páginas
  - ⇨ Acciones manuales: Penalizaciones que nos ha echo google a mano
  - ⇨ Segmentación internacional: Cuando tenemos un dominio .es google solo lo tiene en cuenta en España
  - ⇨ Usabilidad móvil: Nos dice los problemas del responsive
- Índice de Google
  - ⇨ Estado de indexación: Tiene que ir creciendo por que si no es que nos están penalizando
  - ⇨ Palabras clave de contenido
  - ⇨ Recursos bloqueados
  - ⇨ Eliminación de URL, para decirle a google que nos elimine una página por que nos hemos equivocado en algún precio, contenido...
- Rastreo
  - ⇨ Errores de URL: OJOOOOO!!!!!! si tenemos muchos errores 404 hay que solucionarlo modificando el archivo htaccess (si nuestro servidor es linux) y poniendo *Redirect 301 /contacto http://pepito.com/contactanos*
  - ⇨ Estadísticas de rastreo
  - ⇨ Parámetros de URL: cuando pasamos parámetros por GET piensa que es la misma página (Contenido duplicado) entonces hay que decirle al Google que no tenga en cuenta esos parámetros.

SEO. URL Amigables

- Lo vamos a estudiarlo mejor con un ejemplo. Abrimos el sitio de estaciones. Esta formado por una única página index.php y le pasamos como parámetro GET el nombre de la estación: index.php?estacion=invierno.
- Esta dirección no esta considerada como URL Amigable ya que por los buscadores como Google piensan que es una única página index.php y en algunos casos incluso piensa que es contenido duplicado. Entonces una solución es *"medio-engañarle"* diciendo que nuestra página en vez de ser index.php?estacion=invierno es invierno.html y lo mismo con el resto de las estaciones: verano.html…
- Si nuestro servidor es Apache (que será lo habitual) tenemos que modificar el archivo .htaccess (se llama punto htaccess) que nos permite configurar el acceso a nuestro sitio web. Desde este archivo podemos bloquear páginas, directorios, redireccionar (como hemos visto en la diapositiva anterior) y podemos "reescribir" direcciones URL para que vayan a otro sitio, empleando **Expresiones regulares.** En nuestro caso queremos que cuando la página se llama invierno.html le reeinviamos a la página index.php?estacion=invierno. Es decir reescribimos (Rewrite) la dirección de la página. Pasos que debemos seguir:
  1. los vínculos de nuestra página ya no van a ser index.php?estacion=invierno… sino que los cambiamos por invierno.html, verano.html…
  2. Creamos un nuevo documento llamado .htaccess (punto htaccess). Si estamos trabajando con Dreamweaver: con el botón derecho encima de la pestaña Archivos, le damos a Nuevo Archivo
  3. Dentro de este archivo escribimos :
     RewriteEngine On ➔ Esto activa el renombramiento de páginas web (url)
  4. Y debajo escribimos la expresión regular:
     RewriteRule ^([a-zA-Z]*)\.html$ index.php?estacion=$1
     El tejado (acento circunflejo) **^** marca el Principio de la expresión regular y el **$** el Final
     En nuestro caso queremos que capture las palabras: invierno, primavera… que son letras de la A a la Z
     Así que lo ponemos entre corchetes dentro dentro de los paréntesis **[a-zA-Z]**. Si en además hubiésemos metido números añadiríamos 0-9 y si metemos cualquier otro carácter como guion medio (-) o bajo (_) también lo incluiríamos (**[a-zA-Z0-9-]**)
     El ***** que viene después significa que guarde esas letras en la primera variable ($1)
     El **\.html** dice que tenemos que leer hasta el .html (\ es el caracteres de escape, que hay que anteponerlo a los caracteres reservados)
     Así que, si la página es **invierno.html** lo que hace es coger las letras que hay hasta el .html y guardarlas en una variable llamada $1. Y lo que viene después es la URL que tiene que reescribir: **index.php?estacion=invierno**. Y si la página se llama verano.html la reescribe como index.php?estacion=verano…

Introducción a PHP

- PHP es un lenguaje de **scripting** que permite la generación dinámica de contenidos en un servidor web. El significado de sus siglas es "**Pre-Hypertext Processor**" (Procesador Previo al Hipertexto).
- PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP necesitamos especificar cuáles son las partes constitutivas del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando nuestro código por etiquetas. Podemos hacerlo de varias formas:
  - Usando las etiquetas **<?php** y **?>**
  - Usando las etiquetas **<?** y **?>**
  - Mediante **<script languaje="php"> </script>**
- PHP es un lenguaje de lado de servidor:
  Es decir el servidor va a reconocer la extensión correspondiente a la página PHP (phtml, php, php4,...) y antes de enviarla al navegador va a encargarse de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. El resto, lo enviara sin más ya que, asumirá que se trata de código HTML absolutamente comprensible por el navegador
- La forma de separar las instrucciones es mediante punto y coma **;**
  Para la última expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario
- Comentarios de una línea usamos doble barra ***//*** o el símbolo # y de varias líneas mediante ***/**** y ****/***.

Pedido HTTP
http://pagina.com

Servidor web

Toma la página

```
<html>
   <?php echo “Hola” ?>
</html>
```

Interpreta el código PHP

```
<html>
            Hola
</html>
```

Respuesta del Servidor

Cliente (Navegador)

El Navegador Muestra la Web

Las Variables

- Una variable consiste en un elemento al cual le damos un **nombre** y le atribuimos un determinado tipo de información
- En PHP se definen anteponiendo el símbolo dólar **$**
- Y podemos usarlos para guardar:
  - **Números**: $entero=2002; $real=3.14159;
  - **Textos:** $cadena="Hola!!!";
  - **Arrays** o Tablas: $dato[1]="Pepito"; $dato[2]="Juanito"; $dato[3]="Jorgito";
  - Y otros tipos como objetos… que veremos más adelante
- A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables:

```
< ?php
        $cadena="5"; //esto es una cadena
        $entero=3; //esto es un entero
        //Para mostrar un resultado usamos print o echo
        echo $cadena+$entero
?>
```

La diferencia entre **print** y **echo** es que *print* tarda mas tiempo y puede devolver valores tipo **true** y **false** y *echo* nos permitiría ejecutar funciones mientras imprime

- Sin embargo, en contraste, hay que tener cuidado en no cambiar mayúsculas por minúsculas ya que, en este sentido, PHP es sensible
- Podemos **Forzar** una variable para que cambie de tipo con la función setType($variable,"nuevo_tipo");
  Los tipos pueden ser: integer, double, string, array y object
- También podemos forzarla para que se comporte como un tipo determinado:
  $variable = (int) $variable; (int), (integer), (real), (double), (float), (string), (array), (object)

** Un **array** es una colección **ordenada** de elementos de un mismo tipo de datos, agrupados de forma consecutiva en memoria. Cada elemento del **array** tiene asociado un **índice**, que no es más que un número natural que lo identifica inequívocamente y permite al programador acceder a él.*

Las Operadores

- **Operadores aritméticos:**
  - **+** Suma
  - **-** Resta
  - ***** Multiplicación
  - **/** División
  - **%** Devuelve el resto de la división
- **Operadores de comparación:**
  - == Igual
  - != Distinto
  - < Menor que
  - <= Menor igual que
  - > Mayor que
  - >= Mayor igual que
- **Operadores lógicos**
  - And Y &&
  - Or O ||
  - ! No
- **Operadores de incremento** Sirven para aumentar o disminuir de una unidad el valor de una variable
  - ++$variable Aumenta de 1 el valor de $variable
  - --$variable Reduce de uno el valor de $variable
- **Operadores combinados**
  - $variable += 10 Suma 10 a $variable es lo mismo que $variable = $variable+10
  - $variable -= 10 Resta 10 a $variable
  - $variable .= "añado" Concatena las cadenas $variable y "añado"

Variables Globales y Locales

- Las variables, en cualquier lenguaje de programación, tienen un ámbito. Que es el lugar o lugares dónde tienen validez.
- En PHP todas las variables creadas en la página, fuera de funciones, son variables **Globales** de la página.
- Y las variables creadas dentro de una función son variables **Locales** de esa función
- A las variables Globales se puede acceder desde cualquier lugar de la página, mientras que las Locales sólo tienen validez dentro de la función.
- Para evitar confusiones en las funciones, cuando llamemos a una variable global dentro, especificaremos que es global y lo podemos hacer llamando al array $GLOBALS o utilizando la palabra global para especificar que variables queremos emplear:

```php
$numero1=2;
$numero2=7;

function Suma(){
        global $numero1, $numero2;
        return $numero1+$numero2;
}

echo Suma();
```

```php
function Suma(){
    return $GLOBALS["numero1"]+$GLOBALS["numero2"];
}
```

*Una **función** podemos definirla como un conjunto de instrucciones, que permiten procesar las variables para obtener un resultado. Para acceder a una función en PHP simplemente tenemos que llamarla (indicando su nombre) y especificar los parámetros (si los tuviese)*

Variables SuperGlobales

- Las variables SuperGlobales se denominan así por que se puede hacer referencia a ellas en toda la página, dentro y fuera de funciones sin necesidad de indicar que son global.
- Un ejemplo es la que hemos visto antes **$GLOBALS** y otras que explicaremos con detenimiento mas adelante son:
- **$_SERVER**: Variables definidas por el servidor web o directamente relacionadas con el entorno en donde el script se esta ejecutando y contiene información, como cabeceras, rutas y localizaciones del código (algunos servidores pueden omitirlas)
- **$_GET:** Variables proporcionadas al script por medio de HTTP GET y vía parámetros URL.
- **$_POST:** Variables proporcionadas al script por medio de HTTP POST.
- **$_COOKIE:** Variables proporcionadas al script por medio de HTTP cookies
- **$_FILES:** Variables proporcionadas al script por medio de la subida de ficheros vía HTTP, a través del método POST
- **$_ENV:** Variables proporcionadas al script por medio del entorno (por ejemplo HOSTNAME, COMPUTERNAME, USER..)
- **$_REQUEST:** Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario: contiene el contenido de $_GET, $_POST y $_COOKIE
- **$_SESSION:** Variables registradas en la sesión del script.

Control del flujo en PHP: Condiciones IF

- La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución
- Cuando queremos que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, nos servimos del conjunto de instrucciones if, else y elseif.
  La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición) {
    Instrucción 1;
    Instrucción 2;
}  else {
    Instrucción A;
    Instrucción B;
}
```

- La estructura se puede complicar mediante condiciones anidadas:

```
if (condición) {
      Instrucción 1;
}  else {
      if (condición2) {
            Instrucción A;
            Instrucción B;
      } else {
            Instrucción X
      }
}
```

```
if (condición1) {
      Instrucción 1;
}  elseif (condición2) {
      Instrucción A;
      Instrucción B;
} else {
      Instrucción X
}
```

Para comprenderlo mejor, estudiemos un ejemplo -> Queremos que dependiendo del idioma del navegador nos muestre un saludo en ese idioma:

```
<?php
	//Antes de nada introducimos mensajes en forma de variables
		$espanol="Hola";
		$ingles="Hello";
		$frances="Bonjour";

	//Ahora leemos del navegador cuál es su lengua principal
		$idioma=substr($_SERVER["HTTP_ACCEPT_LANGUAGE"],0,2);
		echo $idioma."<br>";
	 //Formulamos las posibilidades que se pueden dar
		 if ($idioma == "es") {
				echo "$espanol";
		} elseif ($idioma == "fr")  {
				echo "$frances";
		} elseif ($idioma == "en")  {
				echo "$ingles";
		} else {
				echo “Otro idioma";
		}
?>
```

*La función **substr** (string $string , int $start [, int $length]) recorta la cadena de texto (**string**) usando los parámetros **start** (inicio) y **length** (numero de caracteres que queremos como resultado)*
$_SERVER: Variables Suplerglobales, definidas por el servidor web o directamente relacionadas con el entorno en donde el script se esta ejecutando

Envío de datos mediante GET y POST

- GET y POST son dos métodos diferentes de comunicación con el servidor definidos en HTTP. Sin entrar en más detalles, que completaremos más adelante, podemos decir que el método GET envía los datos usando la URL, el método POST los envía por la entrada estándar STDIO y en nuestro caso a través de formularios.
- **GET**: envía las variables dentro de la propia URL (dirección) de la página, concatenadas(unidas) por símbolos **&** quedando algo así:
  http://www.mipagina.com/index.php?page=mia&variable2=valor2&variable3=valor3.....
- Cuando enviemos datos usando el método GET a través de la URL habrá caracteres extraños que no podremos pasar por ejemplo el símbolo *… Para evitar estos problemas podemos usar la función **urlenconde**() que viene en la librería de funciones de PHP y limpia todos los caracteres no permitidos en la URL
- Y el método **POST** sólo se puede usar con formularios, y la información no se envía por la URL sino que es invisible al usuario (a no ser que empleemos un proxy…)
- También podemos emplear el método GET para el envío de información a través de formularios aunque nosotros durante el curso vamos a emplear GET dentro de la URL para mandar información "*corta*" y "*seudo-codificada*" y el método POST para enviar grandes campos de texto, subir imágenes… a través de formularios:

```
<form name="miformulario" action="mipagina.php" method="POST">
    <input type="text" size="20" name="mivariable1">
</form>
```

También podemos combinar ambos métodos:

```
<form name="miformulario" action="index.php?pagina=mipagina" method="POST">
```

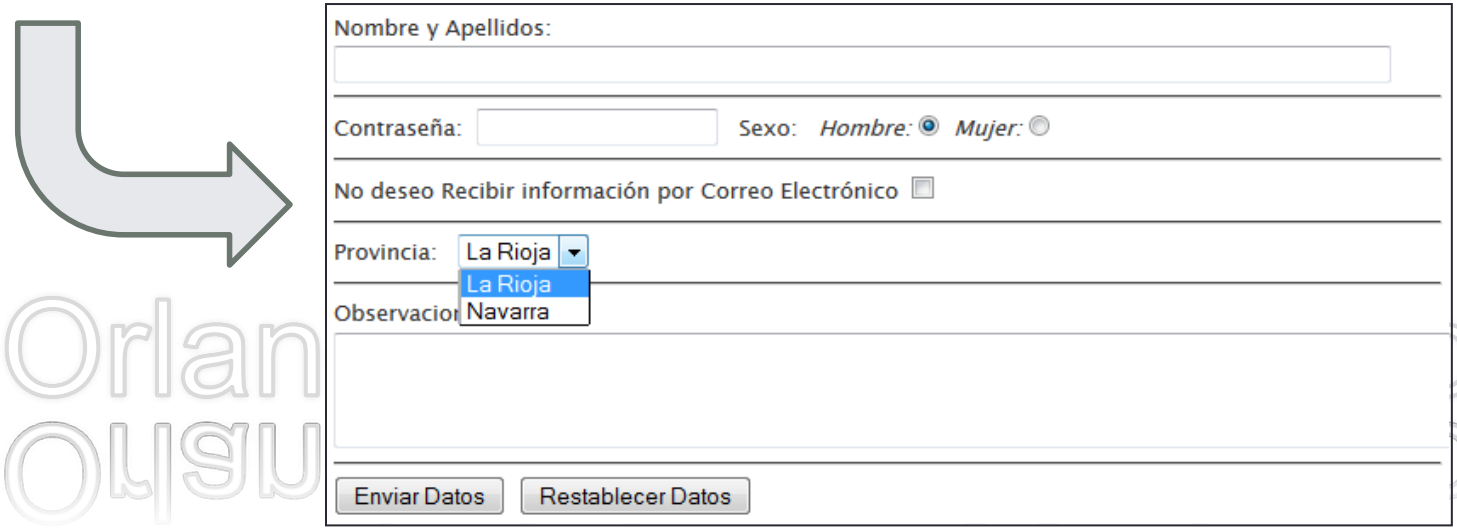
Diferencias entre GET y POST

- Otra de las diferencia entre GET y POST es la limitación de tamaño en cuanto a la información que podemos enviar.
- Por ejemplo, Microsoft Internet Explorer tiene una longitud máxima para el localizador uniforme de recursos (dirección URL) de 2.083 caracteres. Internet Explorer también tiene una longitud máxima para la ruta de acceso de 2.048 caracteres.
- Este límite se aplica tanto a las direcciones URL de las solicitudes POST como a las de las solicitudes GET
- Con lo que si está utilizando el método GET, se aplica la limitación de 2.048 caracteres menos el número de caracteres de la ruta de acceso real
- Sin embargo, el método POST no está limitado por el tamaño de la dirección URL al enviar pares de nombre y valor. Estos pares se transfieren en el encabezado y no en la dirección URL
- En otros navegadores el límite de caracteres es mayor:
  - Firefox: 65.536
  - Safari: 80.000
  - Opera: 190.000
- Pero no solo los navegadores usan URL's, los servidores web también las gestionan y sus tamaños también varían:
  - Apache: 4.000 caracteres
  - Microsoft Internet Information Server (IIS): 16.384 caracteres
- Así que para evitar problemas usaremos GET para pasar información corta (hasta 2.000 caracteres) y sin símbolos extraños y usaremos POST para el resto de datos. De momento solo hemos visto como pasar texto pero mas adelante veremos como enviar archivos… y todo esto lo haremos empleando POST.

Envío de datos mediante GET y POST

- Mediante el uso de **Formularios** podemos incluir elementos interactivos en la página Web, para obtener información del cliente, el cual la escribe rellenando los campos y luego presiona un botón para enviarla. Están delimitados por etiquetas **<form>…</form>** y entre los atributos más destacados tenemos:
  - El Nombre **name** y el identificador **id** del formulario (ambos pueden ser iguales)
  - **method**, indica como se enviaran las respuestas mediante GET o POST
  - **action**, indica la dirección a la que se enviará la información, puede ser una web una dirección de correo electrónico…
  - Y otros atributos pueden ser **enctype**: tipo de codificación, **accept**: Lista de tipos MIME de archivo que se pueden enviar, **accept-charset**: lista de codificación de caracteres… y otros como **onsubmit**, **onreset**…
- Dentro de las etiquetas form, podemos especificar los campos que queremos enviar y esos campos, a los que se les denomina controles, pueden ser de varios tipos:
  - Texto: de una línea <input type="text"/> o de varias líneas <textarea></textarea>
  - Casillas de Verificación ☐
  - Botones de Opción ☐
  - Archivos <input type="file"/> e Imágenes <input type="image"/>
  - Campos Ocultos <input type="hidden"/>
  - Listas/Menús <select></ select >
  - Botones que pueden ser de tipo Enviar <input type="submit"/> Restablecer <input type="reset"/>, botón <input type="button"/>
  - ……
- Estos campos van ha tener elementos comunes como son name (nombre), id… y otros específicos como son size (tamaño: cuando es texto se refiere a caracteres y sino a píxeles), widht, height, maxlength (lóngitud máxima de los datos de entrada puede ser mayor que size)… que veremos con detenimiento en ejemplos prácticos

```
<form action="enviar.php" method="post" id="EJEMPLO">
    <label>Nombre y Apellidos:</label><br />
    <input name="NOMBRE" type="text" id="NOMBRE" size="100"/>
    <br /><hr />
    Contraseña:  <input name="PWD" type="password" id="PWD" size="20" maxlength="50"/>
      Sexo:  
    <i>Hombre:</i><input name="Sexo" id="Sexo_0" type="radio" value="Hombre" checked="CHECKED" />
    <i>Mujer:</i><input name="Sexo" id="Sexo_1" type="radio" value="Mujer"/>
    <br /><hr />
    No deseo Recibir información por Correo Electrónico <input name="no_email" type="checkbox" value="1" />
    <br /><hr />
    Provincia:   <select name="Provincia" size="1">
                        <option value="26">La Rioja</option>
                        <option value="31">Navarra</option>
                      </select>
    <br /><hr />
    Observaciones<br />
    <textarea name="observaciones" cols="100" rows="3"></textarea>
    <br /><hr />
    <input name="Usuario" type="hidden" value="<?php echo $_SESSION["USUARIO"] ;?>" />
    <input name="enviar" type="submit" id="enviar" value="Enviar Datos" />
     <input name="cancelar" type="reset" value="Restablecer Datos" />
</form>
```



Envío de datos mediante GET y POST

- Bueno ya sabemos como se envían ahora vamos a ver como las recibimos:

```
$variable = $_GET['mivariable'];
$variable = $_POST['mivariable'];
```

- Aunque en algunos casos deberemos tratar la información para guardar el valor correcto. Por ejemplo en el caso de los checkbox: cuando está activo (marcado) el valor del campo es el valor definido en su *value* y cuando no, vale *vacío*.
  Si no hubiésemos puesto valor en el *value* el campo vale ***on*** cuando esta activo.
  Por lo tanto en el ejemplo anterior cuando recibamos el valor del campo ***no_email*** el código para resolver el valor sería:

```
$no_email= $_POST['no_email'];
if ( $no_email == “1" ) { echo “No quiere recibir Mails”; }
--------------
if ( $no_email == "on" ) { echo “No quiere recibir Mails”; }
```

- Y aunque podamos evaluar directamente el valor que viene del POST:
  `if ($_POST['no_email']== “1" ) { echo “No quiere recibir Mails”; }`
  por seguridad es siempre conveniente asociarlo a una variable. Más adelante veremos como implementar seguridad para evitar ataques de inyección de código… a través del envío de variables

Otras variables *SuperGlobales* Servidor ($_SERVER['HTTP_USER_AGENT']) interesantes son:
'*HTTP_REFERER*' La dirección de la página que se refirió a la página actual (si la hay).
'*HTTP_USER_AGENT*' Información del cliente con el que se navega la página: el nombre del explorador y su versión.
'PHP_SELF' La dirección relativa a la raíz del sitio de la página actual.
'REMOTE_ADDR' Dirección IP de donde se está navegando la página actual.
'DOCUMENT_ROOT' La dirección (del sistema de archivos) de la carpeta raíz del sitio. Sólo funciona si es Apache.

Formularios HTML5

- Como ya hemos adelantado, HTML5 incorpora nuevos elementos y atributos semánticos para los formularios, que nos van a permitir eliminar gran parte de programación.
  Estos nuevos atributos van a tener algunas particularidades dependiendo del navegador.
- El elemento <input> tiene nuevos valores para el atributo type:
  - email: valida que los datos que introduzcamos correspondan a la estructura de una dirección de correo electrónico (si ponemos el atributo *multiple*, podemos meter mas de una dirección de correo, pero de momento no funciona en Firefox)
  - tel: valida que los datos sean teléfonos
  - url: valida que los datos sean una dirección web
  - search: funciona como el texto pero quita los saltos de línea y podemos añadir el atributo results=x que nos guarda el histórico de búsquedas (x es el numero de búsquedas guardadas).
    El aspecto de estos 4 elementos vistos hasta ahora es una caja de texto, pero en Chrome, Safari… éste último aparece con una lupa para identificarlo 
    Además en las Tablets y Smartphones, cambia el tipo de teclado:

email

tel (teléfono)

url (dirección web)

text, search ...

Formularios HTML5 II

- Otros tipos del elemento input:
  - date, time: se usan para validar que los datos introducidos sean fecha, hora… dependiendo del navegador nos puede salir un ayudante para el calendario (chrome, safari, opera…) lo mismo para la hora, en Firefox e iexplorer sólo valida que sea fecha. Otros tipos válidos son datetime (año, mes, día, hora, minuto, segundos e información del *timezone*), datetime-local para lo mismo que el anterior pero sin la información del timezone, month para fijar el año y el mes pero sin el día, week se utiliza para fijar el número de semana del año (entre 1 y 53)
  - number: valida que los datos sean numéricos, además en algunos navegadores sale una flecha arriba-abajo para incrementar-decrementar. Si queremos meter decimales pondremos usar la propiedad step (pasos) por ejemplo para incrementos de 0,1-> step=0.1, de 2 en 2-> step=2…
    (si queremos cualquier número pondremos step=any, y para numeros sólo positivos pondremos min=0)
  - range: muestra una barra (slider) con la cual podemos marcar-arrastrar un valor de los indicados en los atributos max y min. También podemos usar step, para marcar los incrementos.
  - color: nos permite seleccionar un color como en los programas de diseño…
- Además de los visto antes el elemento <input> tiene nuevos atributos:
  - placeholder: texto por defecto o descriptivo del valor que queremos introducir
  - required: obligar a meter el valor
  - pattern: atributo usado para indicar un patrón para validar los datos, por ejemplo si el campo es teléfono (tel) cada país tiene su propio formato: 9 números… admite expresiones regulares.
  - list: Esto permite al usuario seleccionar un valor de la lista (de un datalist) o escribir uno que no esté en ella. Digamos que nos sugiere una lista de valores, de un datalist, que coincidan con lo que voy escribiendo:

```
<input type="text" name="diasemana" id="diasemana" list="dias"/>
    <datalist id="dias">
        <option value="Lunes" />
        <option value="Martes" />
        <option value="Miércoles" />
        …
</datalist>
```

Formularios HTML5. Etiquetas oninput y output

- Una etiqueta nueva y muy interesante es la etiqueta output, la cual según la definición, representa el resultado de una cálculo. Por lo tanto podremos usarla para calcular cantidades… o por ejemplo para mostrar el numero seleccionado de un range.
  - En el primer caso-ejemplo tenemos 2 etiquetas input numéricas y lo que vamos a decir al formulario que cuando cambiemos alguna valor de ellas, es decir cuando toquemos un input: oninput, realice una operación matemática, que en este caso va a ser una suma, empleando una sencilla formula en *javascript* que nos dice que resultado es igual a la suma de los números 1 y 2:

```
<form method="POST"
    oninput="resultado.value=parseInt(numero1.value)+parseInt(numero2.value)">
<!-- Si lo ponemos así:oninput="resultado.value=numero1.value+numero2.value"
dice que 1 + 3 = 13 por eso los convertimos en enteros mediante la función parseInt -->
<input type="number" id="numero1" value="0"/> +
<input type="number" id="numero2" value="0"/> =
<output name="resultado" for="numero1 numero2"></output>
</form>
```

  - En este otro caso vamos a decirle, que cuando seleccione el valor del rango, es decir cuando hagamos algo en el input de tipo range: oninput que nos muestre el valor a su derecha, en un elemento llamado valor_precios:

```
<form oninput="valor_precios.value = precios.value">
    Intervalo (0 a 100 €):
    <input type="range" name="precios" min="0" max="100" value="0" step="5"/>
    <output for="precios" name="valor_precios" >0</output>
</form>
```

Control del flujo en PHP: Bucles I

- Un bucle es una estructura que empleamos en programación para realizar una determinada acciones un número específico de veces.
- Bucle **While**, los usamos para ejecutar las instrucciones contenidas en su interior, siempre y cuando la condición definida sea verdadera:

```
while (condición) {
   instruccion1;
   instruccion2;
   ...
}
```

```
$i=1; //Iniciamos una variable a 1
while ($i<10) { //Que lo repita mientras la variable no llegue a 10
      /*  Coloreamos DIVs dependiendo de si la variable $i es múltiplo de 2
          para hacerlo usamos el Operador división Resto que nos devuelve el
          resto de una división 4/2=2 sin resto y 3/2=1 y resto 1
      */
      if ($i%2 == 0) { //Si es línea impar color verde oliva
          echo "<div style='background-color:olive'>".$i."</div>";
      } else { //Sino color Lima
          echo "<div style='background-color:lime'>".$i."</div>";
      }
      $i++;
} //Final While
```

Control del flujo en PHP: Bucles II

- Bucle **do/While** es similar al anterior con la diferencia de que la condición se ejecuta al final con lo que incluso siendo falsa se ejecuta al menos una vez:

```
do {
    instruccion1;
    instruccion2;
    ...
}  while (condición)
```

```
$i=1; //Iniciamos una variable a 1
do {
        if ($i%2 == 0) { //Si es línea impar color verde oliva
            echo "<div style='background-color:olive'>".$i."</div>";
        } else { //Sino color Lima
            echo "<div style='background-color:lime'>".$i."</div>";
        }
        $i++;
} while ($i<10) { //Que lo repita mientras la variable no llegue a 10
```

- Bucle **for** es similar a los anteriores la diferencia radica en como se plantea la finalización del bucle:

```
for ($i=1; $i<10; $i++) {
    if ($i%2 == 0) { //Si es línea impar color verde oliva
        echo "<div style='background-color:olive'>".$i."</div>";
    } else { //Sino color Lima
        echo "<div style='background-color:lime'>".$i."</div>";
    }
}
```

Tablas o Arrays en PHP

- Ya hemos definido antes el concepto de Array, completándolo podemos decir que son variables con filas y columnas, a las que podemos acceder mediante un índice que puede ser el id de la fila (1,2,3…) o el nombre de la misma, dependiendo de como la hayamos definido:

```
$moneda["espana"]=“Euro";
$moneda[“inglaterra"]=“Libra";
$moneda["usa"]="Dolar";
```

```
$sentido[1]=“vista";
$sentido[2]=“tacto";
$sentido[3]=“ido";
$sentido[4]=“gusto";
$sentido[5]=“olfato";
```

```
$moneda=array("espana"=> “Euro", “inglaterra" => “Libra", "usa" => "Dolar")
```

- Sino se especifica, el primer índice es el cero, pero podemos utilizar el operador => para especificar el indice inicial. $ciudad = array(1=>"París", "Roma", "Sevilla”);
- Un Array también puede ser multidimensional llamado también tabla dónde podemos incluir arrays dentro de arrays…

```
$pais=array ("espana" =>array ("nombre"=>"España",
                               "lengua"=>"Castellano",
                               "moneda"=>“Euro"),
             “inglaterra" =>array ("nombre"=>“Inglaterra",
                               "lengua"=>“Ingles",
                               "moneda"=>“Libra")
            );
echo $pais["espana"]["moneda"] //Saca en pantalla: “Euro”
```

Tablas o Arrays en PHP

- Para manejar estos arrays existen muchas funciones en PHP vemos algunas de ellas:
  - **print_r**($varible) y **var_dump**($varible) Nos sirven para ver el contenido de una variable, en el caso de los arrays nos saca la estructura interna con todo su contenido
  - **array_values** (*$mi_array*): Lista los valores contenidos en *mi_array*
  - **asort** (*$mi_array*) y **arsort** (*$mi_array*): Ordena por orden alfabético A-Z o Z-A en función de los valores
  - **count** (*$mi_array*): Nos da el número de elementos de nuestro array
  - **ksort** (*$mi_array*) y **krsort** (*$mi_array*): Ordena por orden alfabético 0-9 o 9-0 en función de las claves
  - **array_multisort**, ordenar arrays multidimensionales, explicado a continuación..
  - **list** ($variable1, $variable2...)=*$mi_array* Asigna cada una variable a cada uno de los valores del array (no funciona con cadenas)
  - **next** (*$mi_array*), **prev** (*$mi_array*), **reset** (*$mi_array*) y **end** (*$mi_array*): Nos permiten movernos por dentro del array con un puntero hacia delante, atrás y al principio y al final.
  - **each** (*$mi_array*) Nos da el valor y la clave del elemento en el que nos encontramos y mueve al puntero al siguiente elemento.
  - **array_slice** (*$mi_array*, nº): Corta un array desde el 0 hasta el número indicado
  - **array_shift** (*$mi_array*): devuelve el primer valor del array
  - **unset** (*$mi_array*["valor" o nº]): Elimina ese valor del array
  - **array_push** (*$mi_array*, "Valor1", "Valor2"...): Inserta al final del array los valores
  - **array_merge** (*$mi_array1*, *$mi_array2*...): une 2 o mas arrays

Tablas o Arrays en PHP

array_multisort, ordenar arrays multidimensionales. Imaginemos que tenemos un array con el nombre y la edad de una lista de personas:

```
$personas = array('nombre' => 'Carmen', 'edad' => 22);
$personas = array('nombre' => 'Pablo', 'edad' => 29);
$personas = array('nombre' => 'Susana', 'edad' => 24);
$personas = array('nombre' => 'Cristina', 'edad' => 21);
```

Y queremos que nos ordene los datos por edad.

Si imprimo el array mediante print_r($personas); para ver como están los datos vemos que Carmen de 22 años tiene el índice 0, Pablo el 1, Susana el 2…:

```
Array ( [0] => Array ( [nombre] => Carmen [edad] => 22 ) [1] => Array ( [nombre] => Pablo [edad] => 29 )
[2] => Array ( [nombre] => Susana [edad] => 24 ) [3] => Array ( [nombre] => Cristina [edad] => 21 ) )
```

Entonces lo que hacemos es crear un nuevo array (le llamamos por ejemplo auxiliar) con los datos que nos sirven de orden, en este caso edad:

```
foreach ($personas as $indice => $fila) {
    $auxiliar[$indice ] = $ fila['edad'];
}
```

Si hacemos un print_r($auxiliar); veremos que saca sólo las edades sin ordenar, pero tienen el mismo índice que las personas: Array ([0] => 22 [1] => 29 [2] => 24 [3] => 21)

Ahora lo que tengo que hacer es ordenar el array de $personas y para hacerlo voy a decirle que ordene $auxiliar en orden ascendente (SORT_ASC, SORT_DESC para orden descendente) y que los índices resultantes se los aplique igual a $personas

```
array_multisort($auxiliar, SORT_ASC, $personas );
```

Quedando el array ordenado como vemos a continuación:

```
Array ( [0] => Array ( [nombre] => Cristina [edad] => 21 ) [1] => Array ( [nombre] => Carmen [edad] => 22 )
[2] => Array ( [nombre] => Susana [edad] => 24 ) [3] => Array ( [nombre] => Pablo [edad] => 29 ) )
```

Movernos por Arrays en PHP

- Además de las funciones anterior vamos a ver las siguientes que no permiten recorrer un array y posicionarnos donde nos interese:
  - **current** o **pos** (*$mi_array*): devuelve el valor del elemento que indica el puntero
  - **reset**(*$mi_array*): mueve el puntero al **primer** elemento de la tabla
  - **end**(*$mi_array*): mueve el puntero al **último** elemento de la tabla
- Por ejemplo vamos a ver como crear un array con los días de las semana y recorrerlo:

```
$semana = array("lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo");
echo count ($semana); //7
//situamos el puntero en el primer elemento
reset($semana);
echo current ($semana); //lunes
next($semana);
echo pos ($semana); //martes
end($semana) ;
echo pos ($semana); //domingo
prev($semana);
echo current($semana); //sábado
```

- Como podemos ver aunque efectivo es muy lento, por lo tanto sabiendo que la función **each**() devuelve el valor del elemento actual, podemos usar **while** para recorrerlo:

```
reset($semana);
while (list($clave, $valor) = each($semana)) {
    echo "el día $clave de la semana es $valor <br>";
}
```

Control del flujo en PHP: Bucles III

- Bucle **foreach**, nacido en las versiones de PHP4, nos ayuda a recorrer los valores de un array lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Recordamos que un array es una variable que guarda un conjunto de elementos (valores) catalogados por claves. La estructura general es:

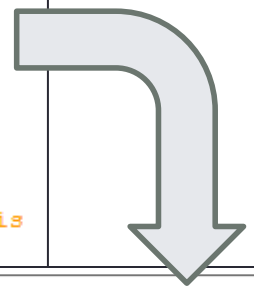
```
foreach ($array as $clave=>$valor) {
   instruccion1;
   …
}
```

```
$moneda= array("España"=>"Euro" ,"Inglaterra" =>"Libra", "USA" => "Dolar");
foreach ($moneda as $clave=>$valor) {
   echo "Pais: ". $clave . " Moneda: ". $valor. " <br>";
}
```

- **switch** es otra estructura que evalúa una expresión (que puede ser una variable) y dependiendo de su valor va hasta el **case** correspondiente y ejecuta lo que se encuentre dentro hasta el **break**, si no encuentra ninguna correspondencia va hasta el **default**, el cual podemos omitir:

```
switch (expresión) {
        case "a": //Código a ejecutar;
                break;
        case "b": //Código a ejecutar;
                break;
        …
        default: //Código a ejecutar por default.
}
```

```php
$moneda=array("España"=>"Euro" ,"Inglaterra" =>"Libra", "USA" => "Dolar");
echo "<hr>";
echo var_dump($moneda);
echo "<hr>";
echo print_r($moneda);
echo "<hr>";
echo print_r(array_values($moneda));
echo "<hr>";
asort($moneda);//Cambiamos el orden A-Z por Moneda
foreach ($moneda as $clave=>$valor) {
   echo "Pais: ". $clave . ", Moneda: ". $valor. " <br>";
}
echo "<hr>";//******************
ksort($moneda);//Cambiamos el orden A-Z por el Indice (en este caso por Pais
foreach ($moneda as $clave=>$valor) {
   echo "Pais: ". $clave . ", Moneda: ". $valor. " <br>";
}
echo "<hr>";//******************
$Paises = array('España', 'Inglaterra', 'USA');
list($pais1, $pais2, $pais3) = $Paises;//Pasamos el Array a variables
echo $pais1.", ".$pais2.", ".$pais3."<br>";
echo "<hr>";//******************
$Solo2Paises=array_slice($Paises,1);//Cortamos el Array del 0 al 1
echo print_r(array_values($Solo2Paises));
echo "<hr>";//******************
echo array_shift($Solo2Paises);//Mostramos el Primer valor
echo "<hr>";//******************
$Solo2Paises=$Paises;
unset($Solo2Paises[2]);//Eliminamos USA
echo print_r(array_values($Solo2Paises));
echo "<hr>";//******************
array_push ($Paises,"Italia","Alemania");//Añadimos Italia y Alemania
echo print_r(array_values($Paises));
echo "<hr>";//******************
$OtroArray=array_merge($Paises,$sentido);//Juntamos 2 Arrays
echo print_r(array_values($OtroArray));
```



```
array(3) { ["España"]=> string(4) "Euro" ["Inglaterra"]=> string(5) "Libra" ["USA"]=> string(5) "Dolar" }
Array ( [España] => Euro [Inglaterra] => Libra [USA] => Dolar ) 1
Array ( [0] => Euro [1] => Libra [2] => Dolar ) 1
Pais: USA, Moneda: Dolar
Pais: España, Moneda: Euro
Pais: Inglaterra, Moneda: Libra
Pais: España, Moneda: Euro
Pais: Inglaterra, Moneda: Libra
Pais: USA, Moneda: Dolar
España, Inglaterra, USA
Array ( [0] => Inglaterra [1] => USA ) 1
Inglaterra
Array ( [0] => España [1] => Inglaterra ) 1
Array ( [0] => España [1] => Inglaterra [2] => USA [3] => Italia [4] => Alemania ) 1
Array ( [0] => España [1] => Inglaterra [2] => USA [3] => Italia [4] => Alemania [5] => ver [6] => tocar [7] => oir [8] => gustar [9] => oler ) 1
```

```php
//Comprobamos si ha seleccionado orden y si no establecemos que es AZ
$Orden="AZ";
if (isset($_POST["Orden"])) {
    $Orden=$_POST["Orden"];
}
?>
<form action="ArrayClientesTablaOrdenar.php" method="post">
    Seleccione el Orden en el que desea mostrar los Datos:  A-Z
    <input name="Orden" type="radio" value="AZ" onclick="this.form.submit()"
        <?php  if ($Orden=="AZ") { echo "checked=checked"; asort($clientes);}?>/>
           Z-A
    <input name="Orden" type="radio" value="ZA" onclick="this.form.submit()"
        <?php  if ($Orden=="ZA") { echo "checked=checked"; arsort ($clientes);}?>/>
</form>
<!-- Creamos una tabla para mostrar los datos -->
<table width="90%" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <th>Id</th>
    <th>Nombre</th>
    <th>Teléfono</th>
    <th>Localidad</th>
  </tr>
<?php
$i=1;
    foreach ($clientes as $clave=>$valor) {
        if ($i%2==0) {//Comprobamos si la fila es par o impar
            echo "<tr>";
        } else {
            echo "<tr class='fila_impar'>";
        }
        //Y ponemos los valores de las celdas
        echo "<td>".$clave."</td>";
        echo "<td>".$valor["nombre"]."</td>";
        echo "<td>".$valor["telefono"]."</td>";
        echo "<td>".$valor["localidad"]."</td>";
        echo"</tr>";//Final de la Fila
        $i++;
    }
?>
</table>
```

```css
<style type="text/css">
    body {
        font-family: Verdana, Geneva, sans-serif;
        font-size: 14px;
    }
    table {
        font-size: 10px;
        color: #0006;
        border-collapse: collapse;
        border-top-width: 1px;
        border-top-style: solid;
        border-top-color: #006;
    }
    th {
        font-size: 14px;
        color:#FFF;
        text-transform: capitalize;
        background-color:#999;
        border-bottom-width: 1px;
        border-bottom-style: solid;
        border-bottom-color: #006;
    }
    td {
        text-align: center;
        padding-top: 5px;
        padding-left: 2px;
        border-top-width: 1px;
        border-top-style: dashed;
        border-top-color: #9AA8BD;
    }
    .fila_impar {
        background-color: #eaf1f4;
    }
</style>
```

```php
//Definimos el Array
$clientes=array("juan" =>array("nombre"=>"Juan SL",
                               "telefono"=>"941123456",
                               "localidad"=>"Logroño"),
                "carlos" =>array("nombre"=>"Carlos SA",
                               "telefono"=>"917894567",
                               "localidad"=>"Madrid"),
                "paco" =>array("nombre"=>"Paco SC",
                               "telefono"=>"948456789",
                               "localidad"=>"Pamplona")
                );
```

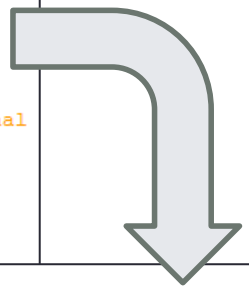
Seleccione el Orden en el que desea mostrar los Datos: A-Z Z-A

Id	Nombre	Teléfono	Localidad
carlos	Carlos SA	917894567	Madrid
juan	Juan SL	941123456	Logroño
paco	Paco SC	948456789	Pamplona

Cadenas de Texto (strings) en PHP

- Otro de los fundamentos que necesitamos conocer en PHP es el manejo de cadenas de texto, y para hacerlo tenemos múltiples funciones, algunas de ellas:
  - **strtolower**($variable) Pone Todo en minúsculas
  - **strtoupper**($variable) Pone Todo en mayúsculas
  - **ucfirst**($variable) Pone la Primera letra en mayúscula
  - **ucwords**($$variable) Pone Primeras Palabras de la frase en mayúsculas
  - **trim**($variable) Quita los espacios al principio y al final
    (otras variantes son **ltrim**->Quita los espacios del Principio, **rtrim**-> Del Final)
  - **str_repeat**($$variable, nº) Repetimos la cadena el nº de veces indicado
  - **strlen**($variable) Cuenta los caracteres y nos devuelve el número
  - **str_word_count**($variable) Cuenta las Palabras
  - **strstr**($variable, “QueBuscar”) Busca la cadena dentro de la variable
  - **str_replace**(“Palabra", “Otra", $variable) Remplaza una palabra con otra. Si queremos que sea insensible a mayúsculas y minúsculas usaremos **str_ireplace**
  - **substr**($variable, nº inicio, nº caracteres) Recorta la cadena desde el nº inicio, el numero de caracteres indicado
  - **strpos**($variable, “donde") Dice la posición dónde se encuentra la palabra
  - **explode**(caracter, $variable): Convierte una cadena en una matriz, tomando como referencia un caracter para separar una cadena en diferentes elementos
  - **implode**(caracter, $array) Lo contrario, separa un array y pone el caracter entre uno y otro (por ejemplo $separado_por_comas = implode(",", $array))
  - **htmlentities**(): convierte todos los caracteres especiales en HTML
  - **number_format** ($variable, nº decimales, símbolo decimal, símbolo miles)
  - Además si necesitamos incluir espacios o saltos de línea en una cadena de texto plana (no HTML) podemos escribir **\t** para tabulación, **\n** para salto de línea o **\r** para retorno de carro (intro), todo esto lo veremos más adelante,

```php
$variable="hola Texto de prueba";
echo strtolower($variable); //Pone Todo en minúsculas
echo "<hr>";//******************
echo strtoupper($variable); // Pone Todo en mayúsculas
echo "<hr>";//******************
echo ucfirst($variable); // Pone la Primera letra en mayúscula
echo "<hr>";//******************
echo ucwords($variable); // Pone Primeras Palabras de la frase en mayúsculas
echo "<hr>";//******************
echo trim("     Muchos Espacios al Principio y al Final       "); // Quita los espacios al principio y al final
echo "<hr>";//******************
echo str_repeat($variable, 3); // Repetimos la cadena el nº de veces indicado
echo "<hr>";//******************
echo strlen($variable); // Cuenta los caracteres y nos devuelve el número
echo "<hr>";//******************
echo str_word_count($variable); // Cuenta las Palabras
echo "<hr>";//******************
if (strstr($variable, "Texto")) {// Busca la cadena dentro de la variable
    echo "Encontrado";
}
echo "<hr>";//******************
echo str_replace("Texto", "String", $variable); // Remplaza una palabra con otra
echo "<hr>";//******************
echo substr($variable, 2, 4); // Recorta la cadena desde el nº inicio, el numero de
echo "<hr>";//******************
echo strpos($variable, "Texto"); // Dice la posición dónde se encuentra la palabra
echo "<hr>";//******************
$datos=("Juan;Luis Orlando;Pepito de Los Palotes;Carlos");
$Listado=explode(";", $datos); //Convierte una cadena en una matriz
echo print_r(array_values($Listado));
echo "<hr>";//******************
$datos=implode(" <> ", $Listado); // Lo contrario, separa un array y pone el caracte
echo $datos;
echo "<hr>";//******************
echo "<font color='#000066' size='+6'>Texto HTML</font><br>";
$texto="<font color='#000066' size='+6'>Texto HTML</font>";
echo htmlentities($texto);
```



hola texto de prueba

HOLA TEXTO DE PRUEBA

Hola Texto de prueba

Hola Texto De Prueba

Muchos Espacios al Principio y al Final

hola Texto de pruebahola Texto de pruebahola Texto de prueba

20

4

Encontrado

hola String de prueba

la T

5

Array ([0] => Juan [1] => Luis Orlando [2] => Pepito de Los Palotes [3] => Carlos) 1

Juan <> Luis Orlando <> Pepito de Los Palotes <> Carlos

Texto HTML

Texto HTMLOtra Linea

```php
$hora = "09:24:38";
list($horas, $minutos, $segundos) = explode(':', $hora);
$hora_en_segundos = ($horas * 3600 ) + ($minutos * 60 ) + $segundos;
echo $hora_en_segundos;
```

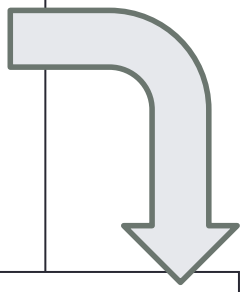
Fechas en PHP

- Trabajar con fechas es algo que suele dar bastantes quebraderos de cabeza. Primero tenemos que saber que PHP trabaja con fechas tomando como referencia la fecha UNIX *timestamp* (segundos que han pasado desde el 1/1/1970).
  La función que me dice el *timestamp* actual es **time()**. Esta fecha nos la devuelve como un numero entero y podré emplearla para sumar, restar días, horas, fechas...
  - Por ejemplo, para calcular la fecha y hora de mañana: **time()+1*24*60*60**
- Otra función muy empleada es **date()** que devuelve la fecha/hora actual del servidor (***Ojo con esto***) dónde se encuentra alojada la página Formateada según parámetros. Los parámetros más comunes son:
  - Para indicar el día: **d** (01-31) **D** (Mon-Sun) **j** (1-31) **N** (1-7) *desde el lunes*
  - Para el Mes: **m** (01-12) **M** (Jan-Dec) **n** (1-12) **F** (January-December)
  - Para el Año: **y** (2 dígitos, por ejemplo 00-13) **Y** (4 dígitos, 2000-2013)
  - Para la Hora: **H** (00:23) **h** (formato de 12 horas, 00-12) **a** (am-pm) **A** (AM-PM)
  - Para los Minutos: **i** (00:59) y para Segundos: **s** (00:59)
  - Existen otros parámetros como **w** que es el día de la semana (0 Domingo a 6 Sábado)… que podemos consultar cuando necesitemos en la ayuda de PHP
- También se puede usar para formatear otro fecha que no sea la actual, esta fecha tiene que estar en formato *timestmap* y la sintaxis es: **date(formato, fecha)**
  Ejemplos de esta función son los siguientes:
  - date (“d/m/Y”) Esto mostraría la fecha de hoy 28/05/2021
  - date("D j \d\e F \d\e Y", 1356994800) Esto mostraría Tue 1 de January de 2013

Fechas en PHP

- **strtotime(fecha)** convierte un texto en fecha expresada en *timestamp*. El único problema es que la fecha tiene que estar en un formato determinado:
  - Para fechas: mm/dd/yyyy mm/dd/yy yyyy/mm/dd dd-mm-yyyy yy-mm-dd yyyy-mm-dd (si usamos / asume fecha en ingles y – formato europeo) con lo que si ponemos una fecha en español 11/09/2013 se pensará que le pasamos el 9 de Noviembre en vez del 11 de Septiembre la solución podría pasar por hacer un str_replace de / por - (str_replace("/", "-", $fecha))
  - Para Horas hh:mm:ss o hh:mm
  - strtotime("01/01/2013") Devuelve el número anterior 1356994800
- **mktime**(hora,minuto,segundo,mes,dia,año,horario_verano) devuelve el tiempo *timestmap* de los parámetros pasados. Y nosotros solemos emplearlo para hacer operaciones con fechas. Por ejemplo:
  - $mañana = mktime(0, 0, 0, date("m") , date("d")+1, date("Y"));
  - $mes_anterior = mktime(0, 0, 0, date("m")-1, date("d"), date("Y"));
  - $año_siguiente = mktime(0, 0, 0, date("m"), date("d"), date("Y")+1);
- **getdate()** devuelve un array con los datos de la fecha actual, también podemos usar ***getdate(nº timestamp)*** para indicarle la fecha. Los índices de ese array son: ***mday*** *(para el día: de 1 a 31),* ***mon*** *(para el mes: 1 a 12),* ***year*** *(año 4 digitos),* ***wday*** *(día de la semana 0:Domingo a 6:Sábado),* ***yday*** *(día del año: de 1 a 365),* ***seconds***, ***minutes*** *y* ***hours****. Luego en representación texto están* ***weekday*** *(Sunday-Saturday) y* ***month*** *(January-December)*.
  - Por ejemplo: $ArrayFecha=getdate(1356994800); echo $ArrayFecha["year"]; devolvería 2013 o echo $ArrayFecha[“month"]; devolvería January
- **chekdate** (mes, dia, año) Comprueba si una fecha es válida, si es así devuelve TRUE y si no lo es FALSE.

```php
echo $hoy=time();//timestamp fecha actual
echo "<hr>";//*****************;
echo date("d/m/y");//Fecha actual en formato Dia/Mes/Año
echo "<hr>";//*****************
$mañana=time()+1*24*60*60;
echo date("d/m/y",$mañana);//Fecha en formato Dia/Mes/Año
echo "<hr>";//*****************
$fecha="28/11/2013";
/*Esto da error y así lo comprobamos, por que strtotime sólo
acepta fecha con formatos mm/dd/yyyy mm/dd/yy yyyy/mm/dd dd-mm-yyyy yy-mm-dd yyyy-mm-dd*/
if (strtotime($fecha)){ echo strtotime($fecha)."<br>";} else {echo "Error en la fecha<br>";}
//Soluciones posibles, reemplazamos la / con -
$fecha=str_replace("/", "-", $fecha);
echo date("d/m/y", strtotime($fecha))."<hr>";//*****************
//O en otros casos nos interesará sacar día, mes y año para componerla
$Otrafecha="31/12/2013";
$dia=substr($Otrafecha, 0, strpos ($Otrafecha,"/"));
$resto=substr($Otrafecha, strpos ($Otrafecha,"/")+1);
$mes=substr($Otrafecha, strpos ($Otrafecha,"/")+1, strpos ($resto,"/"));
$anio=substr($resto, strpos ($resto,"/")+1);
//Y con los 3 datos sacados la componemos como queramos
echo "Día: ".$dia." Mes:".$mes." y Año: ".$anio."<hr>";//*****************
echo "Formato MySQL: ".$anio."-".$mes."-".$dia."<hr>";//*****************
//Para hacer operaciones con ella
$TimeStamp_SemanaQueViene=mktime(0,0,0,$mes,$dia+7,$anio);
$SemanaQueViene=date("d/m/y", mktime(0,0,0,$mes,$dia+7,$anio));
echo $SemanaQueViene."<hr>";//*****************
//Por que esto no se puede hacer
$MalSumar7Dias=$Otrafecha+7;
echo "Esta Fecha: ".$Otrafecha." + 7 =".$MalSumar7Dias;//En este caso suma 7 al primer número
echo "<hr>";//*****************
//Otra forma de sacar el Mes, Día, Año para hacer lo que queramos es usando getdate()
$ArrayFecha=getdate($hoy);
echo "Día: ".$ArrayFecha["mday"]." Mes:".$ArrayFecha["mon"]." y Año: ".$ArrayFecha["year"];
```



1385652604

28/11/13

29/11/13

Error en la fecha
28/11/13

Día: 31 Mes:12 y Año: 2013

Formato MySQL: 2013-12-31

07/01/14

Esta Fecha: 31/12/2013 + 7 =38

Día: 28 Mes:11 y Año: 2013

Crear nuestras propias Funciones en PHP

- Hasta ahora hemos visto como emplear funciones definidas en la biblioteca de PHP, bueno pues el siguiente paso es crear las nuestras propias. Ya hemos definido antes una función como un conjunto de instrucciones, englobadas bajo un nombre y en la cual podemos definir unos parámetros. Estas instrucciones puede ser cualquier código válido, incluso otras funciones y definiciones de clases. Para llamar a esta función basta con nombrarla y para definirla, la sintaxis es:

```
function NombreDeLaFuncion ($parámetros) {
        Instrucciones;
        …
        }
```

- Los nombre de las funciones siguen las mismas reglas que otras etiquetas PHP: tienen que empezar con una letra o guion bajo, seguido de letras, números o guiones bajos.
- Todas las funciones tienen ámbito global y no tienen por que ser declaras antes de usarla, siempre y cuando el flujo de programa lo permita, es decir una función definida dentro de otra, depende de la ejecución de la primera y no puede ser llamada desde fuera hasta que evaluamos la primera.
- También podemos llamar a funciones recursivas dentro de las mismas, lo que no permite PHP es redefinir o *desdefinir* una función.
- Podemos definir funciones con o sin parámetros y estos sólo tienen validez durante la ejecución de la función (local). Y estos parámetros pueden tener valores por defecto `function Nombre ($parametro1="Juan";$parametro2=5.8) { }`

Crear nuestras propias Funciones en PHP

- Las funciones pueden ejecutar código directamente (echo …;) o retornar valores para ello se emplea la palabra return este devuelve un único valor, si queremos que devuelva más deberemos emplear un array. Por ejemplo:

```
function CalcularIVA($BaseImponible,$Porcentaje=21){
   return $BaseImponible*(,$Porcentaje/100);
}
$BI=$_Post["BaseImponible"];
$TotalIVA_incluido= $BI+CalcularIVA($BI);
echo "Base Imponible = ". $BI."<br>";
echo "Importe IVA = ".CalcularIVA($BI)."<br>";
echo  "Total = ". $ TotalIVA_incluido."<br>";
echo “IVA Reducido= ". CalcularIVA($BI,10) . "<br>";
```

- También podemos crear librerías de funciones, las cuales llamaremos usando require(librería.php) o include(librería.php). Mediante estos comandos podemos insertar código de otro fichero en el actual. La diferencia básica entre ambos es que si llamamos a un archivo y este no existe, include da un mensaje de error y continua mientras que require interrumpe el flujo de la página, es decir para que funcione el programa es **requerido** que se encuentre el archivo (en versiones anteriores:4.0.2 su diferencia radicaba en que require siempre leía el archivo origen e include sólo si el código pasaba por ahí (por ejemplo en sentencias if…)
- Existen otras versiones de las llamadas que son require_once e include_once las cuales funcionan de la misma manera que la anteriores pero sólo se cargan una vez aunque llamemos varias veces a la misma librería. El problema de usar estas versiones es que son mas pesadas, por lo que sólo las usaremos cuando sea obligatorio, cuando necesitamos que no exista duplicidad de variables…

Concepto de Base de Datos

- Una Base de Datos podemos decir que es una Cantidad de Información (Conjunto de Datos) que se encuentra organizada mediante una estructura común y que guarda una relación entre sí. Los datos del mismo tipo se encuentran en un mismo sitio, lo cual hace mucho más fácil acceder a esos datos, y acelera el proceso de trabajo con ellos
- Un ejemplo claro y que vamos a asociar perfectamente de una base de datos es una Agenda Telefónica. Estas almacenan Información (Datos) referentes a números de Teléfono y Direcciones asociadas a una determinada Persona, Empresa... datos que guardan relación entre sí. Por lo tanto una de las características de la definición ya está cumplida. La otra es que posean una estructura común, o sea, los datos han de estar organizados. Por lo tanto una agenda en la que vamos apuntando nos de teléfono asociados a nombres pero que carecen de un orden, o los nos de teléfono escritos en el tablón de la cocina, no podemos considerarlos como una Base de Datos por que no poseen una estructura común, y los datos del mismo tipo no se encuentran en el mismo sitio

Nombre	Teléfono	Dirección	Empresa
Juan	**94112345678**	Logroño	**S.L.**
Pepe	**94845678912**	Navarra	**S.A.**

- La estructura principal de una base de datos es la **Tabla** donde los datos se organizan en Filas y Columnas. Las filas reciben el nombre de **Registros** y las columnas el de **Campos**. Las filas o registros contienen los datos referentes a una Persona o un Elemento Particular, mientras que las columnas o campos contienen los datos de un mismo Tipo. Y la intersección de una fila (registro) y una columna (campo), recibe el nombre de dato, y nos indica el valor de una registro en un determinado Tipo de Dato.
- Una Base de Datos puede estar compuesta por una única Tabla, como en el ejemplo anterior, en ese caso se dice que es una base de datos **Simple**. Pero cuando necesitamos tener mas de una tabla, con datos relacionados entre ellos decimos que es una base de datos **Relacional**.

Base de Datos Relacional

- Imaginémonos la Base de Datos de un Vídeo Club. En esa Base de Datos tenemos los datos de nuestros Clientes con su nº de Cliente, Dirección, Teléfono, D.N.I.
  Y donde cada día introducimos las películas Alquiladas, cuándo es la fecha de Devolución y qué Cliente la ha cogido. Si se retrasa más de un día en devolver la película, tendremos que llamarle por teléfono para decirle que, si se retrasa más le cobraremos una penalización. Por lo tanto vamos a centrarnos primero en los datos necesarios de cada Cliente: Nombre, DNI, Dirección y Teléfono. Y vamos a centrarnos en los datos que manejamos cada Día con respecto a cada Cliente y la Película que ha cogido: Titulo de la Película, Fecha de Alquiler, y Fecha de Devolución. Si juntamos todos estos datos tendremos que cada día manejaremos los siguientes datos todos a la vez. Nombre Cliente, Dirección, Teléfono, DNI, Titulo de la Película Alquilada, Fecha Alquiler, Fecha Devolución.
  Es de Suponer que un cliente Alquile más de una película a lo largo del tiempo, es más, sería lógico que lo hiciese como mínimo cada 7, 14 días. Pues si cada vez que viene un Cliente tenemos que indicar todos los datos vistos anteriormente en una sola tabla estamos ante dos grandes problemas:
  - 1º es que tenemos que escribir los datos repetidos: Duplicidad de Información
  - Y el otro problema proviene de esa duplicidad de Información, y es que si modificamos un dato por ejemplo el Teléfono o la Dirección de un cliente, a la hora de consultar datos referentes a ese Cliente va a existir un conflicto. Ya que asociados a un mismo individuo vamos a tener dos datos distintos en el mismo sitio (campo) y eso dará lugar a un grave Error en la Base de Datos.
- Para evitar esto se ha optado en la actualidad por trabajar con otro tipo de Base de Datos llamadas Relacionales. Es decir compuesta por un conjunto de Tablas relacionadas entre sí por los datos comunes. Para solucionar el ejemplo anterior crearíamos como mínimo 3 tablas: Clientes, Películas y Alquileres, relacionadas por campos en común, por ejemplo el DNI o un nuevo campo Código de Cliente, que repetiremos en la tabla de Clientes y Alquileres y lo mismo con las Películas, Código de Película que repetiremos en la tabla de Alquileres y Películas.

Otros conceptos relacionados con las de Bases de Datos

- Además de los conceptos vistos antes de Base de Datos Simple, y Relacional, Tabla, Registro, Campo y Dato… necesitamos conocer otra serie de conceptos que emplearemos a la hora de definir las tablas en MySQL.
- **Nombre** de Base de Datos, Tabla, Campo, Índice: El nombre de tabla, campo… puede ser cualquiera, pero hay una serie de normas y consejos a seguir para crearlos:
  - Hemos de procurar que el nombre describa la información que contendrá el campo, tabla…
  - Este nombre puede tener como máximo 64 caracteres (letras, números y símbolos) y puede incluir espacios
  - No puede empezar ni terminar con espacios
  - y no puede haber dos campos con el mismo nombre en una tabla.

  Estas son las reglas aplicables a casi todas las Bases de Datos incluida MySQL y aunque podemos poner un nombre tan extenso como queramos, incluyendo espacios, es mucho mejor simplificarlo todo lo que podamos aunque resulte un poco ilegible, para evitar problemas propios de nombres demasiado largos (P. Ej.: al tener que referirnos a esos campos con el Nombre Literalmente escrito, podemos cometer errores de escritura y es mucho más fácil dirigirnos a él cuanto más simplificado este el nombre) lo más indicado es no usar espacios y usar nombres de como mucho 15 caracteres. Y nunca emplear signos de puntuación, ni caracteres extraños… algunos no están ni permitidos: / \ .

  Tenemos que tener en cuenta que la internamente MySQL creará un directorio con la Base de Datos y ficheros con las Tablas con lo que a las restricciones lógicas de crear un nombre le sumamos las permitidas al crear un directorio/archivo
- **Índice y Clave Principal**. Un índice es una *marca* que ponemos en un campo para indicarle a la base de datos que ordene los datos por ese campo. Y una clave principal además de ordenar los datos, obliga a que los registros incluidos en su interior sean Únicos y no Nulos, con lo que incluiremos siempre una clave principal en todas las tabla y la usaremos para relacionar las tablas (DNI, Código Película…) y los índices para marcar aquellos campos que son consultados habitualmente.

Otros conceptos relacionados con las de Bases de Datos

- **Tipos de Datos** (Campos) antes de profundizar en los tipos de datos-campos de una base de datos, tenemos que preguntarnos una para de cosas:
  - ¿Qué **Variedad** de Datos debemos introducir en el campo (letras, números, dibujos, fechas...)?. Ya que en un campo cuyo tipo de datos sea numérico no podremos introducir letras, y al introducir números en un campo de tipo texto no podremos tratar esos números como tales y no podremos por ejemplo realizar operaciones matemáticas...
  - ¿Qué **Espacio** queremos Ocupar? Por ejemplo un campo numérico suele ocupar entre 8-16 bytes. Mientras que un campo texto, puede llegar a ocupar hasta 64000 bytes. En cuanto al espacio lo lógico es ajustar el tamaño lo máximo posible, simplemente por ahorrar capacidad y aumentar la rapidez.
  - ¿Qué **Operaciones** vamos a Realizar con ese campo?. Ya hemos visto antes que al introducir un dato numérico en un campo reservado a texto, ese nº será tratado como texto y no se podrán realizar operaciones matemáticas con él.
  - Y por último si, ¿Podremos usar ese campo como **Índice** de esa tabla de datos?

  Casi siempre será muy fácil definir los tipos de campos, pero existen casos en los que tendremos que evaluar una serie de condicionantes, por ejemplo: El número de Cuenta del Banco es un número de 20 dígitos, pero no existe un tipo de campo numérico que acepte 20 enteros con lo que si le definimos este campo como numérico, la Base de Datos redondeará los números según la precisión. Con lo que perderemos, en la mayoría de los casos, los 6 últimos dígitos, con todo lo que conlleva esta metedura de pata, cuando genere un cargo a quien se lo voy a hacer.
  Igual ocurre con el DNI, Código Postal… Por lo tanto existen números que trataremos como textos, por que nunca haremos operaciones matemáticas con ellos (Sumar DNI, Multiplicar Números de Cuenta…)

Introducción a MySQL

- **MySQL** es un sistema para gestionar Bases de Datos Relacionales usando un lenguaje de consulta estructurado SQL (a partir de una oración SQL llevará una determinada acción sobre la base de datos). Es una aplicación de código abierto y por lo tanto gratuita, aunque posee soporte de pago si fuese necesario.
- Podemos manejar MySQL desde una ventana de consola, o desde herramientas externas, desarrolladas en código abierto o de pago, como pueden ser phpMyAdmin o MySQL Administrator. Nosotros dado que hemos instalado el paquete completo XAMPP bajo Windows, que incluye phpMyAdmin utilizaremos éste, aunque antes tenemos que saber como realizar todos los procesos desde línea de comandos.
- Por lo tanto accedemos a la ventana de consola de Windows escribiendo **cmd** o **command** en *Buscar Programas y Archivos* o desde Inicio > Ejecutar. Una vez accedamos a la consola localizamos la carpeta **bin** de la instalación de MySQL, normalmente **c:\xampp\mysql\bin** por lo tanto escribimos en la consola **cd** para ir al raíz del directorio y luego **cd c:\xampp\mysql\bin** una vez en el directorio, mediante el comando **mysql –help** podemos ver la ayuda de MySQL con todas las sentencias que podemos emplear.
- **MySQL** crear por defecto el usuario **root** con todos los permisos posibles habilitados y sin clave, con lo que en un servidor en producción del cual seamos administradores lo primero sería asignarle un password. Para hacerlo vamos a usar el comando mysqladmin y los parámetros –u Usuario y password o -p (si usamos -p la clave irá seguida de la p sin espacios):

  mysqladmin -u root password miclave o **mysqladmin -u root -pmiclave**
- **Conexión/Desconexión** a la Base de Datos:

  mysql -h NombreDelServidor -u NombreDeUsuario –p **sino ponemos clave nos la pide*

  Si estamos corriendo en una máquina local (como la nuestra) podemos conectarnos de forma anónima simplemente poniendo **mysql**. Entonces vemos como el la ventana de comando nos aparece **mysql>** esperando que le dictemos alguna orden. Una vez conectados podemos desconectarnos poniendo **quit** o **exit** o pulsando Ctrl+D

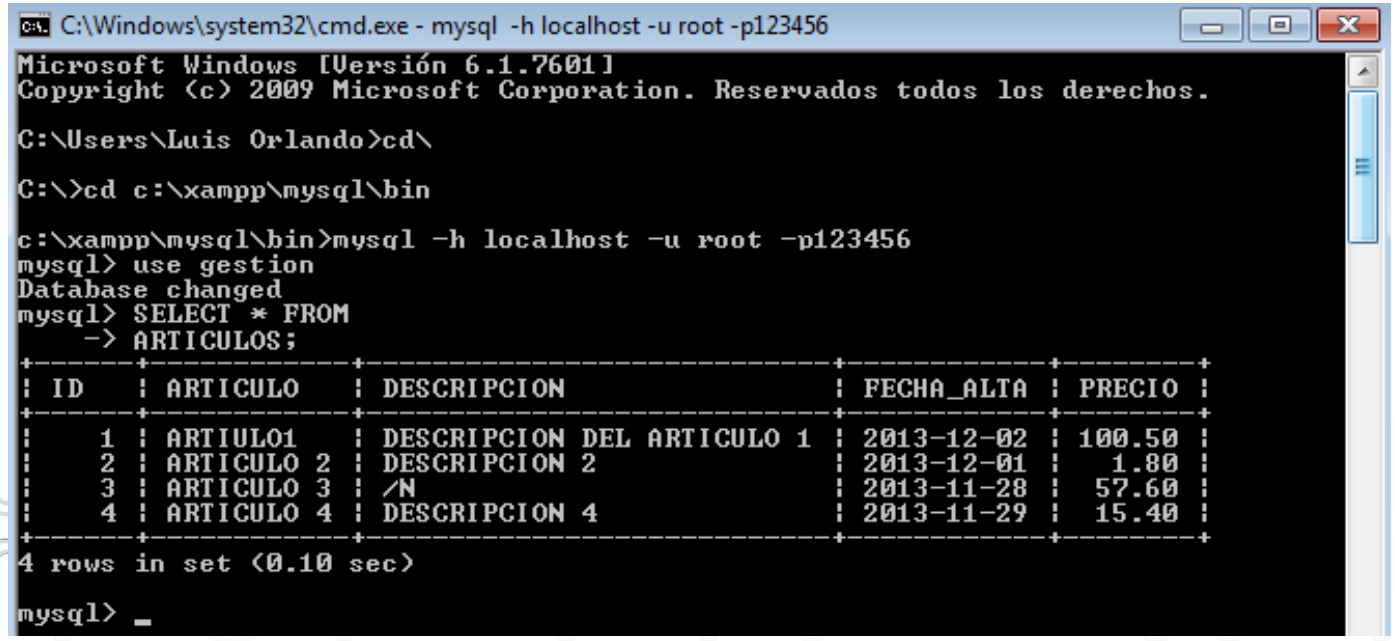
Estructura de MySQL

- En el directorio /share están los mensajes de error del servidor para los diferentes idiomas, los directorios /include y /lib incluyen las librerías necesarias… Bueno lo que nos interesan son 2 directorios: **/bin** dónde están los ejecutables para manejar la base de datos: conectar, desconectar, copias de seguridad… y en **/data** encontraremos un subdirectorio por cada base de datos que creemos y dentro encontraremos 3 archivos por cada tabla: **mitabla.ISD** (que contiene los datos de la tabla) **mitabla.ISM** (El archivo de índices y claves) y **mitabla.frm** que contiene la estructura de la tabla.
- Por defecto se crea la base de datos **mysql** con 5 tablas: **host** (indica que maquinas pueden acceder a nuestro sistema), **user** (usuarios, son sus diferentes permisos) y **db**, **tables_priv** y **columns_priv** nos indican las Base de Datos de nuestro sistema, tablas y columnas… En estas tablas veremos campos que nos marcarán los permisos…
  - Select_priv (Permite utilizar la sentencia SELECT)
  - Insert_priv (Permite utilizar la sentencia INSERT)
  - Update_priv (Permite utilizar la sentencia UPDATE)
  - Delete_priv (Permite utilizar la sentencia DELETE)
  - Create_priv (Permite utilizar la sentencia CREATE o crear bases de datos)
  - Drop_priv (Permite utilizar la sentencia DROP o eliminar bases de datos)
  - Reload_priv (Permite recargar el sistema mediante mysqladmin reload)
  - Shutdown_priv (Permite parar el servidor mediante mysqladmin shutdown)
  - Process_priv (Permite manejar procesos del servidor)
  - File_priv (Permite leer y escribir ficheros usando comandos como SELECT INTO OUTFILE y LOAD DATA INFILE)
  - Grant_priv (Permite otorgar permisos a otros usuarios)
  - Index_priv (Permite crear o borrar índices)
  - Alter_priv (Permite utilizar la sentencia ALTER TABLE)

  Dado que sería muy extenso explicar la función de todas estas sentencias, las iremos viendo sobre la marcha a medida que vamos necesitándolas

Sintaxis de MySQL

- Un comando normalmente consiste de un sentencia **SQL** seguida por un **punto y coma**.
- Cuando emitimos un comando, mysql lo manda al servidor para que lo ejecute, nos muestra los resultados y regresa el prompt indicando que está listo para recibir más consultas.
- mysql muestra los resultados de la consulta como una tabla (filas y columnas). La primera fila contiene etiquetas para las columnas. Las filas siguientes muestran los resultados de la consulta. Normalmente las etiquetas de las columnas son los nombres de los campos de las tablas que estamos usando en alguna consulta.
- mysql muestra cuántas filas dan como resultado y cuanto tiempo tardó en ejecutarse la consulta *(estos valores pueden ser un tanto imprecisos ya que no se muestra la hora del CPU, y porque pueden verse afectados por otros factores, como la carga del servidor y la velocidad de la red)*
- Las palabras clave pueden ser escritas usando MAYÚSCULAS y minúsculas
- Un comando no necesita ser escrito en una sola línea, mysql determinará en dónde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea.

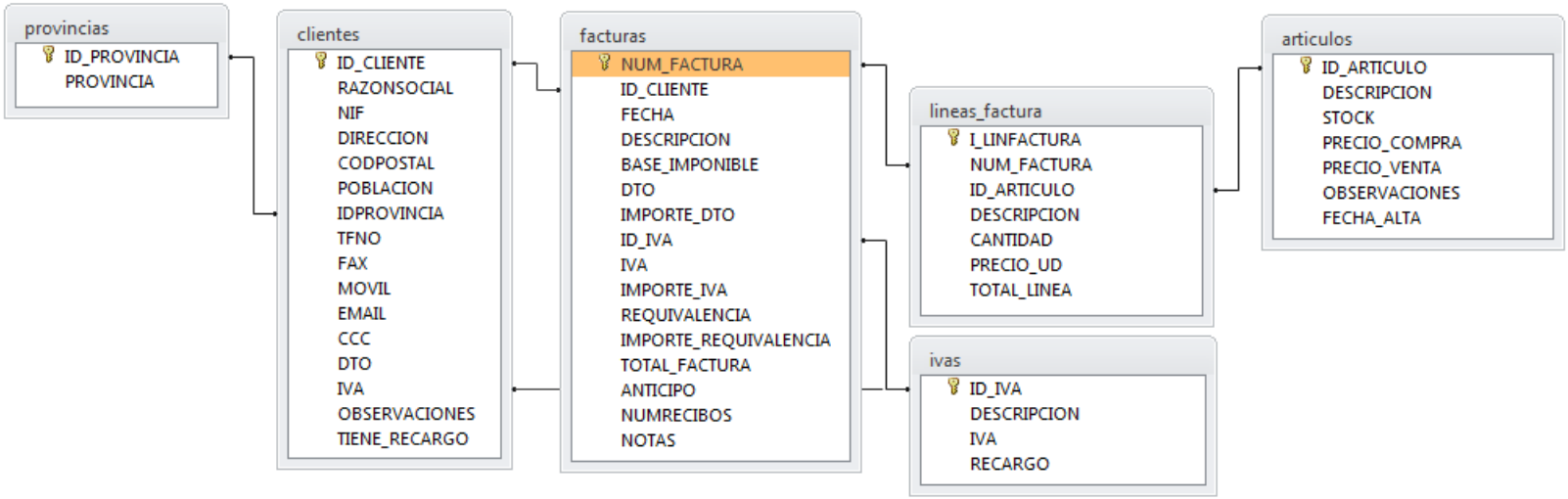


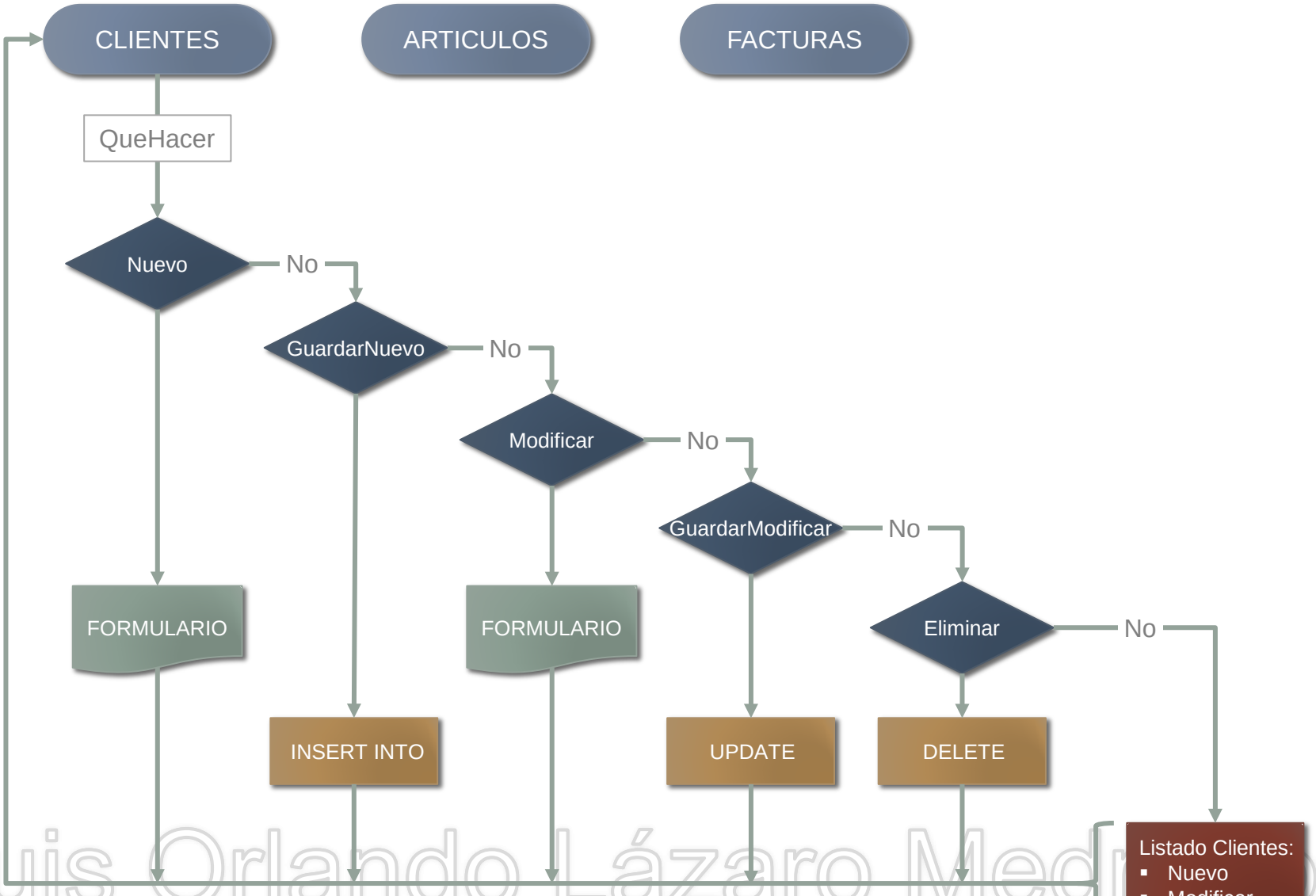
Un ejemplo muy sencillo que analizaremos mas tarde, consiste en seleccionar todos los elementos de una tabla llamada ARTICULOS

Base de Datos GESTION

- Para realizar las prácticas del curso vamos a crear una Base de Datos Relacional (RDB) para la gestión de las Facturas de una empresa ficticia. Siguiendo el esquema de las RDBs vamos a separar las tablas por conceptos y luego as conectaremos por campos comunes:
  - Tabla de CLIENTES con un campo clave principal llamado ID_CLIENTE, y los datos habituales como RAZON_SOCIAL, NIF_CIF, TELEFONO, DIRECCION, CCC, TIPOIVA…
  - Tabla de ARTICULOS con un campo clave principal llamado ID_ARTICULO, y los datos ARTICULO, DESCRIPCION, STOCK, PRECIO_COMPRA, PRECIO_VENTA, FECHA…
  - Estas 2 tablas las conectaremos a la de FACTURAS que a su vez dividimos en 2 la cabecera conectada a la tabla de CLIENTES y las líneas conectada a los ARTICULOS:
    - Tabla FACTURAS con un campo clave principal llamado NUM_FACTURA, y otros campos como FECHA, BASE_IMPONIBLE, IVA, IMPORTE_IVA, TOTAL_FACTURA… y por supuesto el campo de conexión con la tabla de CLIENTES -> ID_CLIENTE
    - Tabla LINEAS_FACTURA con una campo clave principal llamado ID_LINEA_FACTURA y conectada a la tabla de FACTURAS por el campos NUM_FACTURA y a la tabla de ARTICULOS por el campo ID_ARTICULO, así como los datos CANTIDAD, TOTAL…

Dado que es un ejemplo práctico para el curso, no profundizaremos en los campos necesarios para la gestión real de un programa de facturación (Retenciones…) y gestión de clientes (Formas y Fechas de Pago…) y Artículos (Movimiento de Almacén…)





Tipos de Campos MySQL

De forma genérica podemos dividir los tipos de datos posibles en 3: Numéricos, Texto y Fechas:

- **Numéricos**: Pueden ser enteros (exactos) o decimales (reales):
  - **Enteros** dependiendo de los valores que puede alcanzar y su tamaño en Bytes:

Tipo	Bytes	Valor Mínimo (Con signo/Sin signo)	Valor Máximo (Con signo/Sin signo)
TINYINT	1	-128 0	127 255
SMALLINT	2	-32768 0	32767 65535
MEDIUMINT	3	-8388608 0	8388607 16777215
INT	4	-2147483648 0	2147483647 4294967295
BIGINT	8	-9223372036854775808 0	9223372036854775807 18446744073709551615

Además también podemos indicar la anchura visual que tendrá la columna cuando mostremos los datos, este ancho no restringe el rango de valores y es sólo estético INT(10) y si además añadimos el atributo ZEROFILL rellena el espacio en blanco con ceros: INT(5) ZEROFILL -> 00002. Otro atributo opcional es UNSIGNED para no permitir valores negativos.

  - **Decimales**: En este caso es un poco más complejo evaluar el tamaño, ya que va depender de varias opciones que podemos especificar entre paréntesis:
    - Si ponemos un solo número es la precisión en bits FLOAT(8)
      Si ponemos un nº de 0-23 usamos un FLOAT de 4 bytes y hasta 53 usamos un DOUBLE de 8 bytes
    - Pero si ponemos 2 valores, se trata de la anchura del numero completo y el número de decimales, por ejemplo DOUBLE (6,2) dice que el número tendrá una anchura máxima de 6 dígitos de los cuales 2 son la parte decimal y por tanto 4 la parte entera
    - Otro concepto que debemos tener en cuenta es si el número es coma flotante (aproximado) o fija (exacto). Éste último lo emplearemos para datos dónde necesitamos guardar una precisión exacta, por ejemplo datos monetarios…

Tipos de Campos MySQL

- **Numéricos**
  - **Decimales** intentando resumir los visto en la página anterior quedaría la tabla así:

Tipo	Bytes	Coma	Valor Mínimo	Valor Máximo
FLOAT	4	Flotante	1.175494351E-38	3.402823466E+38
FLOAT(X)	4-8	Flotante		
DOUBLE o **REAL**	8	Flotante	2.2250738585072014E-308	1.7976931348623157E+308
DECIMAL(M,D) **NUMERIC** (M,D)	M+2 o M+1	Fija	M+2 si D>0 y M+1 si D=0, el rango de valores de M es de 1 a 64 (por defecto 10) y el D rango de 0-34	

  - **Bool** o **Bit** sólo permite un número entero que puede ser 0 o 1. Ocupa 1 bit
- **Fecha**
  - **DATETIME** en este caso podemos guardar la Fecha y la Hora juntas (8 bytes)
  - **DATE** sólo guardamos la Fecha (3 bytes) YYYY-MM-DD
  - **TIME** sólo guardamos la Hora (3 bytes) HH:MM:SS
  - **YEAR** sólo guardamos el año (1 byte) YYYY
  - **TIMESTAMP** guarda los segundos que han pasado desde el 1/enero/1970 pero el rango máximo es hasta el año 2037 (4 bytes)
- **Texto**
  - **Char(n)**: guarda una cadena de longitud fija, por ejemplo char(4) = 4 bytes (da igual que pongamos a, ab ó abc) y n puede ir de 0 a 255
  - **VarChar(n)**: longitud variable, varchar(4): ‘’->1 byte, ‘ab’->3 bytes y ‘abcd’ -> 5 bytes. n puede ir de 0 hasta 65.535
  - **Binary** y **VarBinary** funcionan como Char y VarChar pero cuando guardan los valores los guardan como binarios y no como caracteres

Tipos de Campos MySQL

- **Texto**
  - Luego existen 2 tipos nuevos de grupos campos para almacenar Textos, que son los **TEXT** y los **BLOB** (Binary Large Object) que difieren en la forma de almacenar y ordenar los valores (ordenan los valores sin tener en cuenta Mayúsculas o minúsculas), los TEXT almacenan los datos como cadenas de caracteres mientras que los BLOB lo hacen como cadenas binarias (bytes) por lo que se suelen usar para almacenar ficheros, imágenes, sonidos, y ambos son de longitud variable.

Tipo	Bytes	Valor Máximo
TINYBLOB, **TINYTEXT**	Longitud+1 bytes	255 caracteres
BLOB, **TEXT**	Longitud+2 bytes	65.535 caracteres
MEDIUMBLOB, **MEDIUMTEXT**	Longitud+3 bytes	16.777.215 caracteres
LONGBLOB, **LONGTEXT**	Longitud+4 bytes	4.294.967.295 caracteres

  - Y por último podemos encontrarnos con los tipos **ENUM** y **SET** que permiten guardar valores de una Lista (por ejemplo Provincias…):
    - El campo ENUM puede contener un único valor de una lista que se especifica, acepta hasta 65535 valores distintos
    - Y el campo SET puede guardar ninguno, uno ó varios valores de una Lista de 64 posibles
- Tipos de datos **Espaciales**, a partir de la versión 4.1 se introducen un nuevo tipo de campo para generar, almacenar y analizar elementos geográficos. Según especificaciones del consorcio Open GIS (empresas, agencias gubernamentales, universidades…). Las especificaciones dependerán del tipo geográfico que acepte pero podemos definir un elemento geográfico como cualquier cosa que tenga una ubicación: Una montaña, una Ciudad, un Código Postal, Carretera, intersección de calles…

Usar MySQL

- Ya hemos visto en las páginas anterior como acceder a la Base de Datos (en adelante BD) y Manejarla, a continuación vamos a ver sentencias que podemos usar para crear una BD, Usarla, Crear Tablas, Insertar Datos… todo utilizando la línea de comandos: **cmd.exe** desde la dirección **c:\xampp\mysql\bin**
  - Conectar con mysql: **mysql -h NombreDelServidor -u NombreDeUsuario –p**
  - Mostrar todas las BD del servidor: mysql> **SHOW DATABASES**;
  - Crear nueva BD llamada gestion: mysql> **CREATE DATABASE gestion;**
  - Entrar en la BD para poder usarla: mysql> **USE gestion;**
  - Mostrar las tablas de la BD: mysql> **SHOW TABLES;**
  - Crear nueva tabla: mysql> **CREATE TABLE clientes(**
    ** lo podemos hacer en varias líneas ya que no ejecuta el código hasta que ve un punto y coma*
    -> **id INT(10), razon_social VARCHAR(50),**
    -> **nif VARCHAR(9), fecha_alta DATE);**
  - Mostrar la estructura de una tabla: mysql> **DESCRIBE clientes;**
  - Insertar datos desde archivo de texto: mysql> **LOAD DATA LOCAL**
    -> **INFILE “articulos.txt" INTO TABLE articulos;**
    El Fichero separado por tabuladores en el mismo orden de las columnas, fechas en el formato YYYY-MM-DD y \N para NULL
  - Insertar datos en la Tabla: mysql> **INSERT INTO clientes**
    -> **VALUES(1,‘Cliente 1',‘B26123456','2013-11-30’);**
    O podemos insertar datos en campos específicos: mysql> **INSERT INTO clientes**
    -> **(id, razon_social, nif, fecha_alta)**
    -> **VALUES(1,’CLIENTE 1',‘B26123456','2013-11-30’);**
  - Seleccionar datos de una tabla: mysql> **SELECT * FROM clientes;**
  - Eliminar todos los datos de una tabla: mysql> **DELETE FROM clientes;**
  - Modificar datos de una tabla: mysql> **UPDATE clientes**
    -> **SET fecha_alta= "2013-12-2" WHERE id=1;**

Estructura de las Sentencias en MySQL

- Una vez visto por encima como manejar la base de datos vamos a concretar la sintaxis de varias sentencias SQL que podemos usar en MySQL:
  - Para Insertar datos la estructura de la sentencia sería:
    INSERT INTO tabla (columna1, columna2..) VALUES (valorCol1, ValorCol2);
    Podemos omitir las columnas si insertamos datos en todos los campos y en los values cuando el campo sea texto hay que encerrarlo entre comillas (simples o dobles)
  - Para actualizar los datos de una Tabla:
    UPDATE tabla SET columna = valor WHERE condición;
  - Para borrar los datos de una Tabla:
    DELETE FROM tabla WHERE condición;
  - Y para seleccionar datos:
    SELECT columna1, colum2 FROM tabla WHERE condición ORDER BY col1 ASC o DESC;
    Si queremos seleccionar todos los campos podemos omitir columna y poner en su lugar * y las clausulas WHERE y ORDER BY (ASCendente o DESCendente) son opcionales
- Para personalizar las sentencias podemos emplear operadores de comparación, aritméticos… algunos son:

Operador	Significado	Ejemplo
=	Igualdad	SELECT * FROM CLIENTES WHERE ID= 100;
!=, <>, ^=	Desigualdad	SELECT * FROM CLIENTES WHERE ID!= 100;
<	Menor que	SELECT * FROM CLIENTES WHERE ID< 200;
>	Mayor que	SELECT * FROM CLIENTES WHERE ID> 200;
<=	Menor o igual	SELECT * FROM CLIENTES WHERE ID<= 200;
>=	Mayor o igual	SELECT * FROM CLIENTES WHERE ID>= 200;
in	Igual a cualquiera ()	SELECT * FROM CLIENTES WHERE IDIN (100, 300);
not in	Distinto a cualquiera ()	SELECT * FROM CLIENTES WHERE IDNOT IN (200);
between	Contenido en el rango	SELECT * FROM CLIENTES WHERE ID BETWEEN 100 AND 199;
not between	Fuera del rango	SELECT * FROM CLIENTES WHERE ID NOT BETWEEN 100 AND 199;
like	Contiene (% equivale a *)	SELECT * FROM CLIENTES WHERE NOMBRE LIKE 'LU%' OR LIKE '%OR%'
+ -	Suma y Resta	SELECT BASE+IVA FROM FACTURAS WHERE NUMFACTURA=13-0048';
* /	Producto y División	SELECT NUMFACTURA, BASE*1.21 FROM FACTURAS;
||	Concatenación	SELECT TFNO||MOVIL AS TELEFONOS FROM CLIENTES;

Estructura de las Sentencias en MySQL

- Otros ejemplos:
  SELECT * FROM clientes WHERE razon_social="CLIENTE 1";
  SELECT * FROM clientes WHERE fecha_alta>= “2013-1-1";
  SELECT * FROM clientes WHERE (nif LIKE =“%b26%" AND fecha_alta>= “2013-1-1“)
  OR (nif LIKE =“%b31%" AND fecha_alta>= “2013-1-1“) ;
- Seleccionando columnas particulares: mysql> **SELECT razon_social, nif FROM clientes;**
- Para que sólo salga una vez el nombre de cada cliente:
  mysql> **SELECT DISTINCT cliente FROM facturas;**
- Ordenando registros
  mysql> **SELECT nombre, nif FROM clientes ORDER BY provincia;**
  mysql> **SELECT nombre, nif FROM clientes ORDER BY provincia DESC;**

- También podemos especificar las filas que queremos seleccionar utilizando la cláusula **LIMIT**. Esta cláusula tiene 2 parámetros, el nº de fila desde la que empezar y el nº de filas que devolver:
  Para seleccionar los 10 primeros clientes: **SELECT * FROM CLIENTES LIMIT 0, 10;**
- Juntar los datos de 2 tablas, por ejemplo dado que en la ficha de cliente sólo guardamos el ID_PROVINCIA para sacar el nombre de la provincia que está en la tabla de provincias, juntamos los datos de CLIENTES Y PROVINCIAS a través de la clausula INNER JOIN:
  SELECT clientes.razonsocial, provincias.provincia FROM provincias INNER JOIN clientes ON provincias.id_provincia = clientes.idprovincia;
  INNER JOIN sólo seleccionará clientes que tengan provincia, otras versiones son LEFT JOIN y RIGHT JOIN dónde podemos seleccionar todos los datos de una tabla y los que coincidan de otra. Left y Right indican dónde se encuentra la tabla de la que sacamos todos los datos, en la segunda la tabla los datos que no existan serán NULL
- Otras Sentencias Importantes pueden ser:
  - Copia de Seguridad
    $archivo = "backup_".date("Y_m_d_H_i_s").".sql"; //Definimos el nombre del archivo
    mysqldump --opt -h localhost gestion -u root –p123456 -r \"".$archivo."\“
    ** Desde ventana de comandos poner junto el –u y el nombre de usuario –uroot –p123456*
  - Restaurar copia de Seguridad
    mysql -h localhost -u root -p123456 gestion < archivo.sql

Dar Formato a las Sentencias de MySQL

- Existen muchísimas funciones para formatear un resultado de una Consulta (query) en MySQL, nosotros vamos a estudiar con detenimiento algunas de ellas y otras simplemente las explicaremos a medida que vayamos usándolas.
  Una de ellas muy empleada es DATE_FORMAT(fecha,formato): dónde indicaremos el campo fecha o directamente la fecha deseada y su formato, sus opciones de formato son:

%M	Nombre del mes (January..December)	%m	Mes, numérico (01..12)	%c	Mes, numérico (1..12)
%W	Nombre de día (Sunday..Saturday)	%d	Día, numérico (01..31)	%e	Día, numérico (1..31)
%b	Nombre del mes abreviado (Jan..Dec)	%a	Día abreviado (Sun..Sat)	%w	(0=Domingo..6=Lunes)
%Y	Año, numérico con 4 dígitos	%y	Año, numérico con 2 dígitos	%j	Día del año (001..366)
%H	Hora (00..23)	%k	Hora (0..23)	%h	Hora (01..12)
%i	Minutos, numérico (00..59)	%s	Segundos (00..59)	%f	Microsegundos
%r	Tiempo, 12-horas (hh:mm:ss AM o PM)	%T	24-horas (hh:mm:ss)	%p	AM o PM
%U	Semana (00..53), domingo el primer día	%u	Semana (00..53), Lunes 1º	%l	Hora (1..12)

  SELECT NOMBRE, DATE_FORMAT(fecha_alta,'%d/%m/%Y') as fecha_sp FROM CLIENTES
  SELECT ID, DATE_FORMAT(hora_HMS,'%H:%i') as hora_HM FROM ENTRADAS
- Otras funciones, llamadas de Control de Flujo, las podemos emplear para especificar el resultado dependiendo de un condicional: IF(condicion, “Valor SI”, “Valor NO”) o para evaluar si el valor de una campo es NULO: IFNULL (“Valor SI”, “Valor NO”)
  IF (recibos.pagado<>0,'Si','No') AS recibo_pagado
- **CONCAT** lo empleamos para juntar cadenas, que pueden ser texto, numeros… Devuelve NULL si alguno de los argumentos es NULL. Puede haber más de 2 argumentos y un argumento numérico se convierte a su cadena equivalente
  SELECT CONCAT(nombre, ‘ ‘ , apellidos) AS nombre_completo FROM contacos;
- **FORMAT(Nº,D)** Formatea el número a un formato como '#,###,###.##', (separación de miles y decimales) redondeado a D decimales, y devuelve el resultado como una cadena de texto.
  Si ponemos D=0 no devuelve decimales: **SELECT FORMAT(PRECIO,0) AS PRECIO FROM…**
- Otra sentencia parecida pero **PHP** es number_format (Nº, Decimales, Separador Decimal, Separador de Miles), acepta 2 (n1 Decimales) o 4 argumentos (separador Decimal y Miles):
  $nombre_formato_español = number_format($numero, 2, ',', ‘.');

Otras funciones de MySQL

- **CAST** o **CONVERT** convierten un valor en otro los valores posibles de conversión son: BINARY, CHAR, DATE, DATETIME, TIME, SIGNED, UNSIGNED. Por ejemplo si queremos tratar el ID_ARTICULO como texto: **SELECT CAST (ID_ARTICULO AS CHAR)**
- Existen muchísimas mas funciones que no tenemos tiempo de estudiar, algunas irán apareciendo a lo largo del curso, pero dado que ya estamos interactuando con la BD MySQL desde PHP conviene que dotemos a nuestras aplicaciones de **Seguridad**.
- Podemos encontrarnos con muchos tipos de ataque a nuestro Sitio Web, pero uno de los más habituales cuando hablamos de BD y PHP, es el de Inyección de Código SQL: **SQL-Injection**. Éste ataque se basa en intentar introducir sentencias SQL dentro de las nuestras a través de los elementos de entrada de nuestro sitio: por ejemplo los campos de texto de los formularios (<input type="text">). Para hacerlo intentar romper el funcionamiento normal del programa para que emita interrupciones, excepciones… que pueden aprovechar para leer estos errores y obtener, por ejemplo los usuarios, claves… almacenados en la base de datos. Más adelante, cuando empecemos a ver Sesiones, Autentificación… veremos un ejemplo detallado, de momento lo que vamos a emplear es una serie de funciones que antepondremos a las lecturas de nuestros "*envíos*", tanto si usamos el método POST o GET.
- Estas funciones realizar validaciones del lado del servidor y otorgan más confianza que otras validaciones javascript que pueden ser desactivadas por el cliente, y algunas de ellas son:
  - **strip_tags**($variable): Esta función devuelve una cadena sin etiquetas HTML y PHP, con lo que evitamos que inserten código malicioso que interrumpa nuestro programa.
    Por ejemplo *(no es muy peligroso, pero nos sirve de ejemplo)* si alguien intenta meter un salto de línea en medio de nuestro código: un
 mediante esta función eliminaría estas etiquetas:
    Texto: *Una
 Línea* -> $variable=strip_tags($_POST["Texto"]) devolvería: *Una Línea*
  - **addslashes**($variable): Pone una barra invertida antes de las comillas simples, dobles…
  - Y aunque está marcado como obsoleta a partir de PHP 5.5 se sigue usando **mysql_real_escape_string**($variable): Quita los caracteres especiales en una Consulta para que sea seguro usarla en mysql_query().
    Esta función hay que usarla después de conectar con la Base de Datos sino no funciona.

JavaScript. Introducción

- Podemos definir JavaScript como un lenguaje de **scripting** del lado del **cliente**. Es decir, scripting por que vamos a integrar el código como scripts dentro del código HTML y del lado del cliente, por que el script se va a ejecutar en el propio navegador sin necesidad de realizar peticiones al servidor, lo que nos proporciona mayor agilidad al no depender del ancho de banda.
- Y éste es el fundamento de su **nacimiento**, no depender del ancho de banda ni de las peticiones al servidor para su ejecución, su mayor **virtud** y el porqué vamos a emplearlo nosotros, acompañando a HTML y a PHP: para realizar validaciones del lado de cliente, por ejemplo en los formularios y dinamizar nuestras páginas: mensajes, abrir ventanas… Por supuesto existen muchísimas otras funciones como crear efectos especiales, menús interactivos… en las que no vamos a profundizar en este curso.
- JavaScript **no es Java**, aunque tienen una sintaxis común, y sus orígenes son similares aunque son muchísimas las diferencias y sólo voy a citar alguna de ellas: JavaScript no se compila simplemente se interpreta cuando el navegador lee la página, JavaScript no es un lenguaje orientado a objetos y podemos decir que Java es mucho más potente y robusto que JavaScript.
- JavaScript podemos ejecutarlo como un **script**, a medida que va **leyendo** la página, o como **respuesta a un evento**, por ejemplo como hacíamos para realizar el submit de un formulario mediante “this.form.sumit()” en el evento OnClick, OnBlur, OnChange…
- Y, podemos incluir el script en la página mediante etiquetas:
  <script type="text/javascript"> … </script>
  o crear un archivo externo *.js con funciones a las que llamaremos cuando necesitemos indicando su dirección: **<script type="text/javascript" src="funciones.js"></script>**
- Para que una página ejecute los JavaScript tiene que tenerlos activado y soportarlos (en la actualidad todos los navegadores) por lo que se suele incluir un mensaje encerrado entre las etiquetas <noscript> que se ejecuta sólo si tiene los scripts desactivados en el navegador. Dentro de esta etiqueta podemos encerrar todo el código HTML que nos interese, por ejemplo:
  <noscript> Bienvenido Activa los JavaScript para el correcto… </noscript>
  Y ésta etiqueta la meteremos justo al empezar el <body> (no es obligatorio pero sí conveniente)

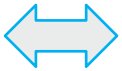
JavaScript. Sintaxis

- Como ya vimos una introducción detallada de la sintaxis, comentarios, manejo de variables y estructuras de control en PHP hay conceptos similares que daremos por entendidos.
- **//** Comentario de una línea y **/*** Comentario de varias líneas ***/**
- Distingue **Mayúsculas** y **minúsculas**
- Las instrucciones se separan por "**;**" o por **saltos de línea** (mejor acostumbrarnos al ;)
- Las variables se declaran anteponiendo **var** y se pueden asignar valores en la declaración, pueden estar o no **tipadas** y su ámbito puede ser **global** o **local** (dentro de las funciones), y podemos no declararlas antes de usarlas, es decir podemos usarlas y automáticamente se declaran aunque es conveniente declararlas.
- Entre los tipos de datos que podemos almacenar en la variables:
  - Todos los **Números** son de tipo numérico: 15 85.6 4.3e12 (tb octal anteponiendo 0 y hex anteponiendo 0x) no como en otros lenguajes en los que se especifica real, entero…
  - Texto: encerrado entre comillas "" y si necesitamos algún carácter especial anteponemos la contrabarra -> Salto de línea: \n Comilla simple: \' Comilla doble: \" Tabulador: \t Retorno de carro: \r Avance de página: \f Retroceder espacio: \b Contrabarra: \\
  - Booleanos: true o false (a veces se traducen como 0=false o cualquier otro valor=true)
  - Y otros como Objetos, Funciones o NULL que estudiaremos sobre la marcha
- Operadores:
  - + - * / % Suma, Resta, Multiplicación, División, Resto de la división (Módulo)
  - ++ Incremento en una unidad
  - -- Decremento en una unidad
  - += -= *= /= %= Incremento Sumando, Restando, Multiplicando… 1000+= 500 -> 1500
  - Concatenar Textos mediante +
  - Lógicos: > (Mayor) >= (Mayor o Igual) <= (Menor o igual) != (Distinto) ! (No) && (Y) II (O)
  - == compara los valores y === compara los tipos de datos y los valores

JavaScript. Sintaxis

- Estructuras de control
  - if ( ) } else { }.. Se puede simplificar como en C: variable = (condición) ? valorSi: ValorNo
  - switch (expresión) { case valor1: break … default: }
  - for (i=0;i<=10;i++) { … }
  - while (condición){ … }
  - do { … } while (condición)
- JavaScript además incluye 2 instrucciones que podemos usar dentro de los bucles que son break y continue. Break detiene el bucle:

```
for (i=0;i<10;i++){
    document.write (i);
    Pregunta = prompt("Seguimos SI o NO??", "SI");
    if (Pregunta == "NO") {
        break;
    }
}
```



```
for (i=0;i<10;i++){
    document.write (i)
    Pregunta = prompt("Seguimos SI o NO??", "SI")
    if (Pregunta == "NO")
       break
}
```

y Continue vuelve al principio del bucle, sin ejecutar las líneas que hay debajo:

```
var i=0;
while (i<10){
    Pregunta = prompt("Estamo en " + i + ", Seguimos SI o NO??", "SI");
     if (Pregunta == "NO"){
       continue;
     }
    i++;
}
```

- Algunas funciones útiles que emplearemos a los largo de curso:
  - typeof variable, devuelve el tipo de datos: boolean, number, string, object
  - escape(TXT) Devuelve el TXT que recibe en codificación ISO Latin 1.
  - unescape(carácter) lo opuesto a la función escape.

JavaScript. Funciones

- Similar a todo lo visto hasta ahora en PHP para definir las funciones:
  function NombreFuncion(parametros){ ... }
  Y para llamarla simplemente hacer referencia al nombre: NombreFuncion(valor)
  El único problema sería el **orden** a la hora de declararla si se generan en bloques <script> distintos, ya que tendríamos que declararla antes de llamarla:

```
<script type="text/javascript">
        NombreFuncion ()

        function NombreFuncion (){
                …
        }
< /script >
```

```
<script type="text/javascript">
        NombreFuncion ()
< /script >

<script type="text/javascript">
        function NombreFuncion (){
                …
        }
< /script >
```

- Y si necesitamos que devuelva un valor lo hacemos mediante **return**
- Otro caso particular javascript es cuando usamos y no definimos una variable dentro de una función esta será global y podremos acceder a ella en toda la página:

```
function glogales_o_locales(){
        var variableLocal = “Hola”
        variableGlobal = “Don Pepito"
}
```

- eval(string) Ejecuta el texto de su interior como si fuese una sentencia de Javascript:
  var VariableTXT = "2 * 3"; eval("document.write(" + VariableTXT +")") -> 6
- parseInt(TXT,base) Convierte un TXT en un nº entero dependiendo de la base (opcional):
  parseInt("10") -> devuelve el valor nº 10 parseInt("101011",2) -> de binario a decimal: 43
- parseFloat(TXT) TXT en número decimal
- isNaN(número) true o false si es un nº

JavaScript. Arrays

- Similar a lo visto en PHP y por destacar alguna cosa:
  - Es un objeto* con lo que para instanciarlo añadimos la sentencia new.
  - Los datos pueden ser de distinto tipo
  - Podemos introducir datos en cualquier columna incluso dejando otras vacías
  - Podemos decirle con cuantas columnas queremos empezar
  - y NombreArray.length nos dice el nº de columnas

  Var Listado = new Array ()

  Var Listado10Uds= new Array(10)

  Var ArrayLleno=[“Enero”, “Febrero”, 3, 4.5, true ….] … ArrayLleno[1] -> “Febrero”
- También podemos crearlos Multidimesionales creando Arrays dentro de Arrays…

```
var Tipico= new Array(2);
Tipico[0]="Logroño"
Tipico[1]="Rincón de Soto"
var TipicoLogroño=new Array("Pinchos", "Vinos", "San Mateo");
Tipico[0]=TipicoLogroño;//Ahora Tipico[0] a no es Logroño sino que es un array
var TipicoRincon=new Array("Peras", "Llorente y Pardo");Tipico[1]=TipicoRincon;

document.write("<table width=300 border=1 cellpadding=1 cellspacing=1>");
for (i=0;i<Tipico.length;i++){
    document.write("<tr>")
    document.write("<td><b>Ciudad " + i + "</b></td>")
    for (j=0;j<Tipico[i].length;j++){
       document.write("<td>" + Tipico[i][j] + "</td>")
    }
    document.write("</tr>")
}
document.write("</table>")

//O los creamos con datos
var Tipico=[["Logroño",["Pinchos", "Vinos", "San Mateo"]],["Rincon",["Futbolistas", ["Llorente","Pardo"]]]];
document.write("Array Completo de 0:"+Tipico[0]+"<br>")
document.write("Segundo Array de 0:"+Tipico[0][1]+"<br>")
document.write("1º valor de 4º Array de 1:"+Tipico[1][1][1][0]+"<br>")
```

Ciudad 0	Pinchos	Vinos	San Mateo
Ciudad 1	Peras	Llorente y Pardo	

Array Completo de 0:Logroño,Pinchos,Vinos,San Mateo
Segundo Array de 0:Pinchos,Vinos,San Mateo
1º valor de 4º Array de 1:Llorente

* *Sin profundizar en lo que es un Objeto, una Clase o la Programación Orientada a Objetos (POO) baste decir que además de asociarle un valor como a las variables, los objetos incluyen métodos propios, propiedades…. a las cuales podemos acceder también. Javascript pese a no ser un lenguaje POO como Java si que usa objetos.*

JavaScript. Cadenas de texto. Clase String

String es otro objeto de Javascript y sus propiedades y métodos son similares a las vistas en PHP:

- Length: Indica el Nº de caracteres
- indexOf(carácter,desde) Dice la posición de la primera vez que aparece el carácter
- lastIndexOf(carácter,desde) Busca como indexOf pero desde el final en lugar del principio
- replace(substring_a_buscar,nuevoStr) Reemplaza un txt por otro
- split(separador) Crea un Array a partir de un separador.
- substring(inicio,fin) Corta un texto desde el inicio hasta el fin.
- toLowerCase() Pone el Texto en minúsculas.
- toUpperCase() Pone el Texto en mayúsculas.
- toString() Este método lo tienen todos los objetos y se usa para convertirlos en texto.
- bold() Texto en Negrita
- fontColor(color) Pone la letra a ese color
- fontSize(tamaño) Pone la letra a ese tamaño
- italics() Pone la letra en cursiva
- link(url) Pone el texto como un enlace a la URL indicada.
- strike() Sirve para que el texto aparezca tachado.
- sub() subíndice.
- sup() superíndice.

```
var Texto = "Hola Hola"
//Partimos la cadena en 2 por la mitad
Posicion_Mitad = Texto.length / 2
Mitad1 = Texto.substring(0,Posicion_Mitad)
Mitad2 = Texto.substring(Posicion_Mitad,Texto.length)
//Y ponemos la primera mitad en negrita y la segunda a 50
document.write(Mitad1.bold() + "<br>" + Mitad2.fontsize(50))
```

JavaScript. Cadenas de texto. Clase Date

- Fecha = new Date -> Muestra la fecha actual, aunque podemos decirle que use otra fecha: Fecha = new Date(año,mes,dia) o Fecha = new Date(año,mes,dia,hora,minutos,segundos) Teniendo en cuenta que para JavaScript Enero es el mes 0 y Diciembre el 11
- getDate() Devuelve el día del mes.
- getDay() Devuelve el día de la semana.
- getHours() Devuelve la hora.
- getMinutes() Devuelve los minutos.
- getMonth() Devuelve el mes (el mes que empieza por 0).
- getSeconds() Devuelve los segundos.
- getTime() Devuelve los milisegundos transcurridos desde el 1/enero/1970 hasta la fecha
- getYear() Devuelve 2 cifras del año
- getFullYear() Devuelve 4 cifras del año
- setDate() Cambia el día.
- setHours() Cambia la hora.
- setMinutes() Cambia los minutos.
- setMonth() Cambia el mes (el mes que empieza por 0).
- setSeconds() Cambia los segundos.
- setTime() Cambia la fecha completa
- setFullYear() Cambia el año de la fecha al número que recibe por parámetro

```
//Cambiar el formato de la fecha Actual
Fecha = new Date()
//sacamos el día mes y año
dia = Fecha.getDate()
mes = parseInt(Fecha.getMonth()) + 1
ano = Fecha.getFullYear()
//escribimos la fecha en un formato Español
document.write (dia + "/" + mes + "/" + ano)
document.write ("<br>")
//Le decimos que hoy es 25
Fecha.setDate(25)
document.write (Fecha)
```

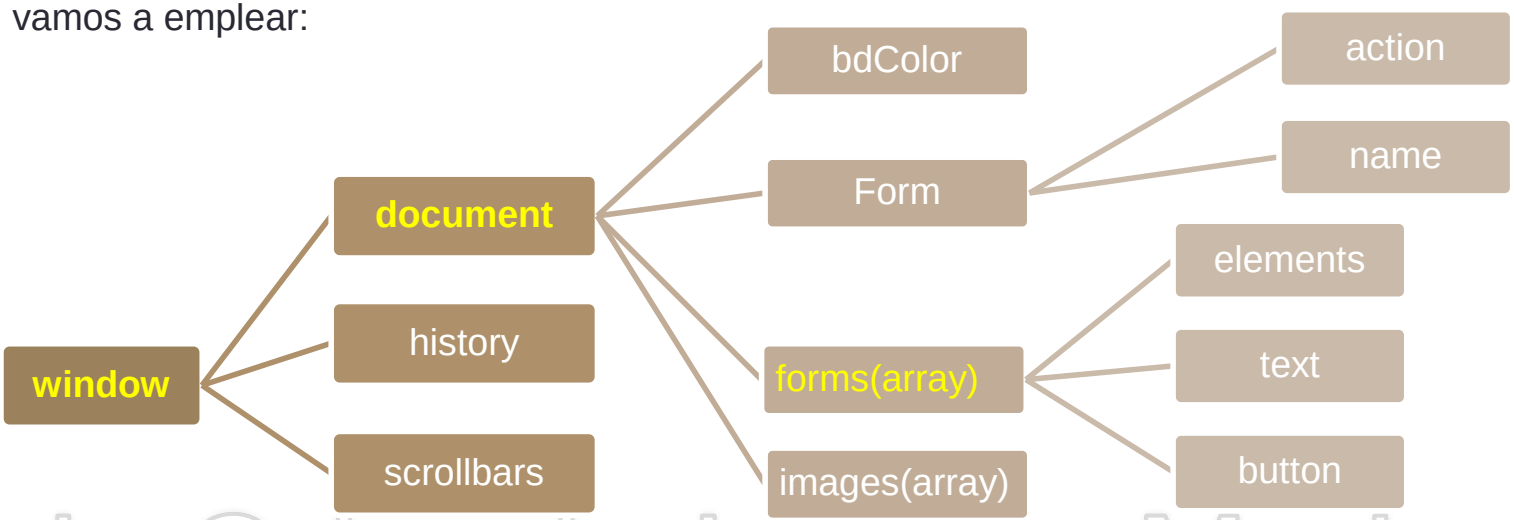
```
12/12/2013
Wed Dec 25 2013 14:23:08 GMT+0100
```

JavaScript. Operaciones Matemáticas. Clase Math y Number

- La clase **Math** es una de las clases nativas de Javascript y se emplea para hacer operaciones matemáticas, más complejas que las vistas hasta ahora de suma, multiplicación…
- Propiedades:
  E Número E, LN2 Logaritmo neperiano de 2, LN10 Logaritmo neperiano de 10,
  LOG2E Logaritmo en base 2 de E, LOG10E Logaritmo en base 10 de E,
  PI cálculo con círculos: π, SQRT1_2 Raiz cuadrada de un medio, SQRT2 Raiz cuadrada de 2.
- Y los Métodos de Math:
  abs() Valor absoluto de un número. El valor después de quitarle el signo.
  ceil() Redondeo Superior.
  floor() Lo contrario de ceil() Redondeo Inferior
  max() Devuelve el mayor de 2 números (Math.max(5,10) da como resultado 10)
  min() Devuelve el menor de 2 números.
  pow() Recibe dos números como parámetros y devuelve el 1º elevado al 2º.
  random() Devuelve un número aleatorio entre 0 y 1
  round() Redondea al entero más próximo.
- Otros Métodos:
  acos() Devuelve el arcocoseno en radianes, asin() el arcoseno, atan() la arcotangente
  cos() El coseno, exp() Devuelve el resultado de elevar el número E por un número
  log() Devuelve el logaritmo neperian,o sin() el seno, sqrt() Devuelve la raiz cuadrada y
  tan() Calcula y devuelve la tangente de un número en radianes
- La clase **Number** la usamos para forzar a que un valor sea numérico si no lo es devuelve NaN (Not a Number).
  Sus propiedades MAX_VALUE y MIN_VALUE nos dicen cual es el número máximo y mínimo que se puede representar en JavaScript
  Y NEGATIVE_INFINITY y POSITIVE_INFINITY los valores negativos y positivos antes de que se produzca el desbordamiento

JavaScript. DOM

- El DOM, Document Objet Model, podemos traducirlo como el mapa del documento, la lista de objetos que tiene la página. Y a los cuales podemos acceder por medio de JavaScript para cambiar sus propiedades o acceder a sus métodos, para modificarlos, suprimirlos, obtener sus valores, colocar nuevos… Por ejemplo para acceder al valor de una campo de un formulario su código sería similar a este: document.forms[0].campo_formulario.value
  Y éste es el verdadero valor de JavaScript, poder acceder al DOM y modificarlo.
- Cuando se carga una página el navegador crea una jerarquía de objetos en memoria por lo tanto necesitamos conocerla para poder acceder correctamente.
  Y además cuando accedamos a ella tendremos que saber que reconoce mayúsculas/minúsculas con lo que no será igual: document.bgColor="red“ con la c minúscula que no funcionará.
- Si bien la jerarquía es muy extensa vamos a ver una pequeña aproximación a los objetos que más vamos a emplear:



- Como vemos todos los objetos comienzan a partir del objeto window, que es objeto con el que controlamos la ventana del navegador, abrimos ventanas nuevas, las cerramos, imprimir…: window.open, window.close, window.print…
- A partir de éste, el segundo en importancia es document con el que trabajaremos detenidamente, y por ejemplo dentro de este se crea un array con todas las imágenes, un array con formularios…

JavaScript. Objeto window

Como ya hemos dicho éste es el objeto principal y de él dependen todos los demás objetos del navegador y sus propiedades son:

- **closed** Indica si se ha cerrado la ventana
- **opener** Hace referencia a la ventana de navegador que abrió la ventana donde estamos
- **defaultStatus** Texto que se escribe por defecto en la barra de estado del navegador.
- **document** Objeto que contiene la página web que se está mostrando.
- **history** Objeto historial de páginas visitadas.
- **innerHeight** Tamaño vertical en pixeles del espacio donde se visualiza la página
- **innerWidth** Tamaño horizontal en pixel del espacio donde se visualiza la página
- **location** La URL del documento que se está visualizando. Podemos cambiar el valor de esta propiedad para movernos a otra página (la tenemos también en el objeto document)
- **locationbar** Objeto barra de direcciones de la ventana
- **menubar** Objeto barra de menús de la ventana
- **name** Nombre de la ventana. Lo asignamos cuando abrimos una nueva ventana.
- **outherHeight** Tamaño vertical de toda la ventana, incluye barras desplazamiento, botones, etc
- **outherWidth** Tamaño horizontal de toda la ventana, incluye las barras de desplazamiento
- **personalbar** Objeto barra personal del navegador
- **scrollbars** Objeto de las barras de desplazamiento de la ventana.
- **status** Texto de la barra de estado.
- **statusbar** Objeto barra de estado del navegador
- **toolbar** Objeto barra de herramientas
- Otros objetos que ya no se usan:

  Frame Un objeto frame de una página web. Se accede por su nombre, ***frames*** array contiene todos los frames de la página, ***length*** Numero de frames de la ventana, ***parent*** Hace referencia a la ventana donde está situada el frame donde estamos trabajando, ***top*** Hace referencia a la ventana donde está situada el frame donde estamos trabajando. Como la propiedad parent, ***window*** Hace referencia a la ventana actual, igual que la propiedad self, ***self*** Ventana o frame actual.

```
<form>
        < input type="Button" value="Pulsa"
        onclick="window.status='Hola!!!'">
< /form>
```

JavaScript. Objeto window

Antes hemos visto sus propiedades, pues sus métodos son:

- **alert**(texto) Abre una ventana de alerta donde se puede leer el texto que recibe por parámetro
- **back**() Ir una página atrás en el historial de páginas visitadas.
- **blur**() Quitar el foco de la ventana actual
- **captureEvents**(eventos) Captura los eventos que se indiquen por parámetro
- **clearInterval**() Elimina la ejecución de sentencias indicadas con el método setInterval()
- **clearTimeout**() Elimina la ejecución de sentencias indicadas con el método setTimeout().
- **close**() Cierra la ventana
- **confirm**(texto) Muestra una ventana de confirmación y permite aceptar o rechazar.
- **find**() Muestra una ventanita de búsqueda.
- **focus**() Coloca el foco de la aplicación en la ventana
- **forward**() Ir una página adelante en el historial de páginas visitadas
- **home**() Ir a la página de inicio que haya configurada en el explorador.
- **moveBy**(pixelsX,pixelsY) Mueve la ventana del navegador hacia la derecha y abajo.
- **moveTo**(pixelsX, pixelsY) Mueve la ventana del navegador a las coordenadas indicadas
- **open**() Abre una ventana secundaria del navegador.
- **print**() Como si pulsásemos el botón de imprimir del navegador.
- **prompt**(pregunta,valor_por_defecto) Muestra una caja de diálogo para pedir un dato.
- **releaseEvents**(eventos) Deja de capturar eventos del tipo que se indique por parámetro
- **resizeBy**(pixelsAncho,pixelsAlto) Redimensiona el tamaño de la ventana, añadiendo a su tamaño actual los valores indicados en los parámetros. Admite valores negativos para reducir la ventana.
- **resizeTo**(pixelsAncho,pixelsAlto) Redimensiona la ventana para que ocupe el espacio indicado
- **routeEvent**() Dirije el evento capturado a un objeto determinado.
- **scroll**(pixelsX,pixelsY) Este método está desaconsejado, ahora se utiliza scrollTo()
- **scrollBy**(pixelsX,pixelsY) Hace un scroll del contenido de la ventana relativo a la posición actual.
- **scrollTo**(pixelsX,pixelsY) Hace un scroll de la ventana a la posición indicada por el parámetro
- **setInterval**() Define un script para que sea ejecutado indefinidamente en cada x tiempo
- **setTimeout**(sentencia,milisegundos) Define un script para que sea ejecutado una vez después de un tiempo de espera determinado.
- **stop**() Como pulsar el botón de stop de la ventana del navegador.

Ya hemos visto que Mediante las funciones setInterval() y setTimeout() podemos ejecutar una función pasado un cierto intervalo de tiempo. La única diferencia entre ellas es que "setInterval" se ejecutará una y otra vez en intervalos de x segundos, en cambio, "setTimeout" se ejecutará una sola vez pasados x segundos. Estas funciones se pueden utilizan todo para realizar alguna acción cada cierto tiempo o por ejemplo para redireccionar una página transcurridos x segundos.
Nosotros vamos a ver un ejemplo que muestre la hora y que se actualice cada segundo::

1. Utilizando la función **setInterval**():

```
 <script>
        function reloj() {
                // Cogemos la fecha actual
                var Fecha= new Date() ;
                // Sacamos la hora, los Minutos y segundos
                var horas = Fecha.getHours() ;
                var minutos = Fecha.getMinutes() ;
                var segundos = Fecha.getSeconds() ;

                // Si es menor o igual a 9 le concatena un 0
                if (horas <= 9) horas = "0" + horas;
                if (minutos <= 9) minutos = "0" + minutos;
                if (segundos <= 9) segundos = "0" + segundos;

        // Asigna la hora actual a la caja de texto reloj
        document.frmReloj.reloj.value = horas+":"+minutos+":"+segundos;
        }
        // Cada segundo invoca la funcion reloj()
        setInterval("reloj()",1000);
</script>
<body>
        <form name=“frmReloj">
                <input type="text" name="reloj" size="10">
        </form>
 </body>
```

2. Utilizando la función **setTimeout**():

```
<script>
        function reloj() {
                // Cogemos la fecha actual
                var Fecha = new Date() ;
                // Sacamos la hora, los Minutos y segundos
                var horas = Fecha.getHours() ;
                var minutos = Fecha.getMinutes() ;
                var segundos = Fecha.getSeconds() ;

                // Si es menor o igual a 9 le concatena un 0
                if (horas <= 9) horas = "0" + horas;
                if (minutos <= 9) minutos = "0" + minutos;
                if (segundos <= 9) segundos = "0" + segundos;

          // Asigna la hora actual a la caja de texto reloj
          document.frmReloj.reloj.value = horas+":"+minutos+":"+segundos;
        // Cada segundo invoca a si mismo
        setTimeout("reloj()",1000);
         }
        // Llamo a la funcion en el primer segundo
        setTimeout("reloj()",1000);
</script>
<body>
        <form name=“frmReloj">
                <input type="text" name="reloj" size="10">
        </form>
 </body>
```

Como os he dicho la única diferencia entre estas funciones, es que "setInterval" se ejecutará una y otra vez en intervalos de x segundos, en cambio, "setTimeout" se ejecutará una sola vez pasados x segundos. Por eso, la función setTimeout se ejecuta dentro de la función reloj() para que de esta forma se ejecute una y otra vez.

JavaScript. Objeto document

Éste es el 2º objeto en jerarquía depende del objeto windows o del frame (en desuso). Propiedades:

- **alinkColor** Color de los enlaces activos
- **Anchor** Llamar a un ancla de la página. Se accede por su nombre.
- **anchors** Array de las anclas del documento.
- **Applet** Llamar a un applet de la página. Se accede por su nombre (un applet es un programa java)
- **applets** Array con todos los applets de la página
- **Area** Una etiqueta <AREA>, de las que están vinculadas a los mapas de imágenes (Etiqueta)
- **bgColor** El color de fondo del documento.
- **classes** Las clases definidas en la declaración de estilos CSS
- **cookie** Llamar a una cookie
- **domain** Nombre del dominio del servidor de la página.
- **Embed** Un elemento de la pagina incrustado con la etiqueta <EMBED>
- **embeds** Array con todos los elementos de la página incrustados con <EMBED>
- **fgColor** El color del texto. Para ver los cambios hay que reescribir la página.
- **Form** Un formulario de la página. Se accede por su nombre.
- **forms** Array con todos los formularios de la página.
- **ids** Para acceder a estilos CSS
- **Image** Una imagen de la página web. Se accede por su nombre
- **images** Array con cada una de las imágenes de la página
- **lastModified** La fecha de última modificación del documento.
- **linkColor** El color de los enlaces.
- **Link** Un enlace de los de la página. Se accede por su nombre.
- **links** Array con cada uno de los enlaces de la página.
- **location** La URL del documento que se está visualizando. Es de solo lectura.
- **referrer** La página de la que viene el usuario.
- **tags** Estilos definidos a las etiquetas de HTML en la página web
- **title** El titulo de la página.
- **vlinkColor** El color de los enlaces visitados.

JavaScript. Objeto document

- **captureEvents**() Para capturar los eventos que ocurran en la página web. Recibe como parámetro el evento que se desea capturar.
- **close**() Cierra el flujo del documento.
- **contextual**() Línea de control de estilos de la página. Si queremos especificarlos con Javascript.
- **getSelection**() Devuelve un string que contiene el texto que se ha seleccionado. Sólo funciona en Firefox.
- **handleEvent**() Invocas el manejador de eventos del elemento especificado.
- **open**() Abre el flujo del documento.
- **releaseEvents**() Liberar los eventos capturados del tipo que se especifique, enviándolos a los objetos siguientes en la jerarquía.
- **routeEvent**() Pasa un evento capturado a través de la jerarquía de eventos habitual.
- **write**() Escribe dentro de la página web. Podemos escribir etiquetas HTML y texto normal.
- **writeln**() Escribe igual que el método write(), aunque coloca un salto de línea al final.

El **flujo del documento** es el orden de escritura de la página en el navegador, una vez cerrado no podemos volver a escribir y si lo hacemos se recarga la página y todo lo anterior se borra. Si por ejemplo después de cargar una página pulsamos sobre el siguiente botón:

```
<INPUT type=button value=escribir onclick="window.document.write('hola')">
```

La página se queda cargando por que ya se ha cerrado el flujo y nosotros hemos vuelto a abrirlo. Pero no lo hemos abierto correctamente (además de no cerrarlo al final). Con lo que si realmente es necesario, usaremos open y close para controlarlo, para el ejemplo anterior mejor:

```
<script>
    function re_escribir(){
        document.open()
        window.document.write('Hola')
        document.close()
    }
< /script>
< form>< INPUT type=button value="Volver a Escribir“ onclick=" re_escribir()">< /form>
```

JavaScript. Formularios

Ya hemos visto que para acceder a los elementos de los formularios podemos hacerlo por **Nombre** o por la posición que ocupa en el **array** de document: document.Nombrefrm.campo2 , document.Nombrefrm.elements[1], document.forms[0].campo2 o document.forms[0].elements[1]. Cualquiera de las cuatro formas es correcta.

En cuanto a las propiedades del objeto form:

- **action** Es la acción que queremos realizar cuando ejecutamos (submit) un formulario: document.miFormulario.action = "miPagina.php"
- **elements** El array de elementos contiene cada uno de los campos del formulario.
- **encoding** El tipo de codificación del formulario
- **length** El número de campos del formulario.
- **method** El método por el que mandamos la información. POST o GET
- **name** El nombre del formulario (también el ID).
- **target** La ventana o frame en la que está dirigido el formulario

Y los eventos para ejecutar el formulario ya hemos visto que son **submit()** o **reset()**

A continuación vemos un pequeño ejemplo de como emplear los eventos y las propiedades:

```
function validarDatos() {
        if (document. NombreFrm. DNI.value == "") {
            alert("Obligatorio: Debe indicar el DNI")
        } else {
            document. NombreFrm.action="enviar.php“ // o mailto:enviarcorreo@dominio.com
            document. NombreFrm.submit()
        }
}
<form name="NombreFrm"  enctype="text/plain">
    Indique DNI*: <input type="Text" name="DNI" value="" size=“10">
    <input type="button" value="Enviar" onclick="validarDatos()">
</form>
```

JavaScript. Campos de los Formularios

En la página 20 cuando estudiamos como enviar variables a través de formularios ya vimos una pequeña introducción a los tipos de campo con lo que ahora sólo vamos a ver que propiedades y métodos podemos usar en javascript para controlarlos:

- Campo **text**: <input type="text"/> . Propiedades:
  - **defaultValue** Valor por defecto
  - **name** Nombre de este campo en el formulario
  - **type** Tipo de campo
  - **value** El texto visible=escrito

  Y sus métodos:
  - **blur**() Retira el foco
  - **focus**() Pone el foco
  - **select**() Selecciona el texto del campo

```
function IrACampo(){
        document.NombreFrm.NombreCampo.focus()
        document.NombreFrm.NombreCampo.select()
}
```

- Los campos **password** tienen las mismas propiedades y métodos, y los **hidden** como están ocultos sólo las propiedades (no tiene sentido hacer un focus a un campo oculto)
- **checkbox:** ☐ .Propiedades:
  - **checked** Informa sobre el estado del checkbox. Puede ser true o false.
  - **defaultChecked** Si está chequeada por defecto o no.
  - **value** El valor actual del checkbox.

  Métodos del checkbox
  - **click**() Es como si hiciésemos un click sobre el checkbox: marca/desmarca el checkbox.
  - **blur**() Quita el foco
  - **focus**() Pone el foco

JavaScript. Campos de los Formularios

- Campo **radio**: ☐ . En este campo podemos quedar otro array nombrando todos los radio button igual. Propiedades:
  - checked Indica si está chekeado o no un botón de radio.
  - defaultChecked Su estado por defecto.
  - value El valor del campo de radio
  - Length (tamaño del array) El número de botones de radio que forman parte en el grupo

  Y sus métodos:
  - **click**() Es como si hiciésemos un click sobre el radio: marca/desmarca
  - **blur**() Quita el foco
  - **focus**() Pone el foco

  En el ejemplo siguiente vemos como recorrer el array de radios y poner de color de fondo el que esté marcado:

```
function ColorFondo(){
        var i
        for (i=0;i<document.FrmColores.QueColor.length;i++){
                if (document.FrmColores.QueColor [i].checked)
                        break;
        }
        document.bgColor = document.FrmColores.QueColor[i].value
}
<form name=FrmColores>
        <input type="Radio" name="QueColor" value="ffffff" checked>Blanco<br>
        <input type="Radio" name="QueColor" value="ff0000">Rojo<br>
        <input type="Radio" name="QueColor" value="00ff00">Verde<br>
        <input type="Radio" name="QueColor" value="0000ff">Azul<br>
        <input type="Radio" name="QueColor" value="ffff00">Amarillo<br>
        <input type="Radio" name="QueColor" value="000000">Negro<br>
        <input type="Button" name="" value="Cambia Color" onclick=" ColorFondo()">
</form>
```

JavaScript. Campos de los Formularios

- Campo **select**: <select></ select >. En este campo creamos tantas <option> </option> como valores queremos en la lista. Propiedades de select:
  - **length** Guarda la cantidad de opciones del campo select. Cantidad de etiquetas <OPTION>
  - **Option** Hace referencia a cada una de sus opciones. Son por si mismas objetos.
  - **options** Un array con cada una de las opciones del select.
  - **selectedIndex** Es el índice de la opción que se encuentra seleccionada

  Y sus métodos:
  - **blur**() Quita el foco
  - **focus**() Pone el foco

  Como ya hemos visto en este campo además de select también tenemos **option**, sus propiedades son:
  - defaultSelected Indica con un true o un false si esa opción es la opción por defecto
  - index El índice de esa opción dentro del select.
  - selected Indica si esa opción se encuentra seleccionada o no.
  - text Es el texto de la opción. Lo que puede ver el usuario en el select
  - value Indica el valor de la opción, que se introduce con el atributo VALUE

  Y también podemos acceder al valor seleccionado de un select a traves de su value: document.MiFormulario.MiListaSelec.value
- Campo **textarea**: <textarea></textarea>. Propiedades:
  - **defaultValue** Que contiene el valor por defecto del textarea.
  - **value** Que contiene el texto que hay escrito en el textarea

  Y sus métodos:
  - **blur**() Quita el foco
  - **focus**() Pone el foco
  - **select**() Selecciona el texto del textarea

JavaScript. Eventos

Los eventos son los métodos con los que controlamos las acciones del usuario, para determinar el comportamiento de una página. Y podemos colocarlos en muchas etiquetas HTML. Si queremos asociar más de una acción a un evento tenemos que separarlos con **;** y los eventos más importantes que podemos utilizar son los siguientes:

- **onabort** Este evento se produce cuando un usuario detiene la carga de una imagen, ya sea porque detiene la carga de la página o porque realiza una acción que la detiene, Por ejemplo: irse de la página.
- **onblur** Se desata un evento onblur cuando un elemento pierde el foco (el lugar donde está el cursor)
- **onchange** Se desata este evento cuando cambia el estado de un elemento de formulario
- **onclick** Se produce cuando se da una pulsación (clic) al botón del ratón sobre un elemento de la página
- **ondragdrop** Se produce cuando un usuario suelta algo que había arrastrado sobre la página web
- **onerror** Se produce cuando no se puede cargar un documento o una imagen y esta queda rota
- **onfocus** El evento onfocus es lo contrario de onblur
- **onkeydown** Este evento se produce en el instante que un usuario presiona una tecla, independientemente que la suelte o no. Se produce en el momento de la pulsación.
- **onkeypress** Ocurre un evento onkeypress cuando el usuario deja pulsada una tecla un tiempo determinado. Antes de este evento se produce un onkeydown en el momento que se pulsa la tecla
- **onkeyup** Se produce cuando el usuario deja de apretar una tecla. Cuando se libera la tecla.
- **onload** Este evento se desata cuando la página, o las imágenes, ha terminado de cargarse.
- **onmousedown** Se produce el evento onmousedown cuando el usuario pulsa sobre un elemento de la página. Se produce en el momento de pulsar el botón, se suelte o no.
- **onmousemove** Se produce cuando el ratón se mueve por la página.
- **onmouseout** cuando el puntero del ratón sale del área ocupada por un elemento de la página
- **onmouseover** cuando el puntero del ratón entra en el área ocupada por un elemento de la página
- **onmouseup** Este evento se produce en el momento que el usuario suelta el botón del ratón
- **onmove** Evento que se ejecuta cuando se mueve la ventana del navegador, o un frame.
- **onresize** Evento que se produce cuando se redimensiona la ventana del navegador, o el frame
- **onreset** Este evento está asociado a los formularios cuando se hace clic en el botón de reset
- **onselect** Se ejecuta cuando un usuario realiza una selección de un elemento de un formulario
- **onsubmit** Cuando pulsamos sobre el botón de enviar el formulario. Se ejecuta antes del envío
- **onunload** Al abandonar una página, porque se pulse sobre un enlace o porque se cierre la ventana

JavaScript. Objeto Images

Otro de los objetos mas modificados desde javascript es el objeto **images**, dependiente de window.document. Podemos acceder a el por medio de su nombre o por medio del array de imágenes: document.nombre_imagen.propiedad, o document.images[indice].propiedad

- **name** Nombre de la imagen
- **src** ruta de la imagen asociada al objeto
- **width** / **height** anchura y la altura de la imagen
- **border** atributo border de la imagen asociada al objeto
- **hspace** / **vspace**, espacio horizontal o vertical de una imagen respecto al texto que la rodea
- **lowsrc** ruta de la miniatura
- **prototype** permite añadir propiedades y métodos adicionales a un objeto Image, es decir, permite ampliar lo que es el objeto en sí, aumentando sus propiedades por defecto
- **complete** propiedad del objeto image de tipo booleano, cierta cuando la imagen se ha acabado de cargar en memoria, y falsa en caso contrario

```
<img src="imagenes/eliminar.gif" width="32" height="32" name="eliminar"/>
<img src="imagenes/modificar.gif" width="32" height="32" name="modificar" />
<img src="imagenes/nuevo.gif" width="32" height="32" name="nuevo" />
<script language="JavaScript" type="text/javascript">
    function ListadoNombresImagenes(){
        var nombres = "";
        for (i=0;i<document.images.length;i++){
            nombres+=document.images[i].name+"\n";
         }
         alert(nombres);
    }
</script>
<input type="button" value="Lista de los Nombres de la imagenes" onClick="ListadoNombresImagenes()">
<br />
<a href= "#" onmouseOver="document.EliminarRollOver.src='imagenes/eliminar_.gif';"
    onClick= "return false;"
    onmouseOut="document.EliminarRollOver.src='imagenes/eliminar.gif';">
    <img src="imagenes/eliminar.gif" width="32" height="32" name="EliminarRollOver" border=0/>
</a>
```

JavaScript. Manejar Divs

- **getElementById** y **getElementByName**, para acceder a todos los objetos del DOM (imágenes, campos, tablas, divs…) por su nombre o id
- **innerHTML**, lo poseen todos los objetos HTML e indica su contenido, podemos cambiarlo, por ejemplo rellenar con una tabla, lista o toda una página HTML un DIV

En el siguiente ejemplo vamos a mostrar ocultar divs a través de JavaScript y de los eventos onmouseout y onmouseover.

Para ello dependiendo de si pasa por el ratón por encima o nó buscaremos por getElementById su nombre y le cambiaremos la propiedad visibilty a visible o hidden:

```
<script language="Javascript">
    function mostrar(nombreDiv){
        document.getElementById(nombreDiv).style.visibility="visible";
    }
    function ocultar(nombreDiv){
        document.getElementById(nombreDiv).style.visibility="hidden";
    }
</script>
<div id="Pestaña1" style="position:absolute;width:100px;height:30px;top:10px;left:10px;background-color:blue"
    onmouseout="ocultar('Contenido1')"
    onmouseover="mostrar('Contenido1')">Pestaña 1</div>
<div id="Pestaña2" style="position:absolute;width:100px;height:30px;top:10px;left:110px;background-color:cyan"
    onmouseout="ocultar('Contenido2')"
    onmouseover="mostrar('Contenido2')">Pestaña 2</div>
<div id="Contenido1" style="position:absolute;width:200px;height:100px;top:40px;left:10px;background-color:cyan;
    visibility:hidden">Contenido 1</div>
<div id="Contenido2" style="position:absolute;width:200px;height:100px;top:40px;left:10px;background-color:blue;
    visibility:hidden">Contenido 2</div>
```

También podíamos hacerlo cambiando la clase (className) para que se comportase de otra forma, o cambiase de tamaño, posición…

JavaScript. Obtener información del Evento

Cuando queremos procesar tareas más complejas con JavaScript que las vistas hasta ahora, utilizando los mismos u otros eventos: onclick, onmouse… casi siempre necesitamos conocer más información como puede ser que tecla ha pulsado o dónde estaba el ratón en el momento de hacerlo.
Para hacerlo JavaScript cuenta con un objeto especial llamado **event**.
Y dependiendo del navegador podemos acceder a el de una forma u otra:

- En **Internet Explorer** depende del objeto window:
  var evento = **window.event**;
- Y en el resto aparece por *"inducción divina"* como argumento:
  function LoQueSea(e) { var evento = e; }
- Combinando ambas opciones podemos obtener el valor de evento en cualquier navegador:
  function LoQueSea(e) { var evento = e || window.event; }

Una vez que tengamos el evento podemos acceder a toda la información relacionada con el mismo:

- **type**, devuelve el tipo de evento producido
- Y otros para el teclado **keyCode**, código interno de la tecla
- **charCode**, carácter de la tecla pulsada
- Posición del cursor con respecto a la ventana del navegador **clientX** y **clientY**
- Posición del cursor con respecto a la pantalla completa **screenX** y **screenY**
- **pageX**, **pageY**…

```
<script type="text/javascript">
function ReMarcaBorde(e) {
  var evento = e || window.event;
  switch(evento.type) {
    case 'mouseover':
      this.style.borderColor = 'black';
      break;
    case 'mouseout':
      this.style.borderColor = 'silver';
      break;
  }
}

window.onload = function() {
  document.getElementById("Contenedor").onmouseover = ReMarcaBorde;
  document.getElementById("Contenedor").onmouseout = ReMarcaBorde;
}
</script>
<div id="Contenedor" style="width:150px; height:60px; border:thin solid silver">
  Aquí escribimos lo que queramos....
</div>
```

JavaScript. Detectar navegador

Existen muchas diferencias entre los distintos navegadores: Opera, Chrome, Internet Explorer, Mozilla… pero sobre todo existen diferencias entre Internet Explorer y el resto, por lo tanto algo importante para que nuestro código funcione es distinguir si es Internet Explorer o No.
Internet Explorer tiene su propio DOM, para poder acceder a los objetos se les puede llamar por document.getElementById("LoQueSea") o directamente por document.LoQueSea, por lo tanto tiene una propiedad única que es **document.all** que abarca todos los objetos de la página, entonces una manera de reconocerlo es preguntando si existe, o al contrario si no existe. Por ejemplo para saber si no es explorer:

var NoIE= document.getElementById && !document.all;

Y otra forma de reconocerlo es preguntando al cliente como se llama el navegador:

var ie = navigator.userAgent.toLowerCase().indexOf('msie')!=-1;

```
<script type="text/javascript">
    //Esto detecta si No es Interne Explorer
    var NoIe = document.getElementById && !document.all;
    // Y esto si es Internet Explorer
    var Ie = navigator.userAgent.toLowerCase().indexOf('msie')!=-1;

    function QuePosicion(e) {
        if (Ie) { // O podríamos poner if (NoIe) y hacerlo al reves
            coordenadaX = event.clientX;
            coordenadaY = event.clientY;
        } else {
            coordenadaX = e.clientX;
            coordenadaY = e.clientY;
        }
        alert("Has pulsado el ratón en la posición: " + coordenadaX + ", " + coordenadaY);
    }

    document.onclick=QuePosicion;
</script>
```

parentNode

Por medio de parentNode podemos seleccionar el elemento padre de otro elemento.

Por ejemplo, si tenemos el siguiente código:

```
<div>
   <p id="hijo">Texto de Ejemplo.</p>
</div>
```

Si escribimos `document.getElementById("hijo").parentNode;`

Selecciona el elemento padre del elemento identificado como hijo, en este caso el div.

firstChild

Con firstChild lo que seleccionamos es el primer hijo de un elemento.

Por ejemplo, si tenemos el siguiente código:

```
<div id="contenido">
   <p>Un párrafo.</p>
   <p>Otro párrafo.</p>
</div>
```

Parece, que document.getElementById('contenido').firstChild debería seleccionarnos el primer párrafo. Sin embargo, sólo ocurre así en Internet Explorer. El problema es que navegadores como Opera o Firefox interpretan también como hijos de un elemento los posibles espacios en blanco y saltos de línea que el elemento pueda contener. Estos, por tanto considerarían que el salto de línea entre el cierre del div y la apertura del primer p es el primer hijo.

lastChild

La propiedad lastChild funciona exactamente como firstChild, pero se refiere el último de los hijos de un elemento. Se aplican, por tanto, las mismas indicaciones anteriores.

nextSibling

Gracias a nextSibling, lo que podemos seleccionar es el siguiente hermano de un elemento.

Se aplican las mismas limitaciones que para las dos propiedades anteriores.

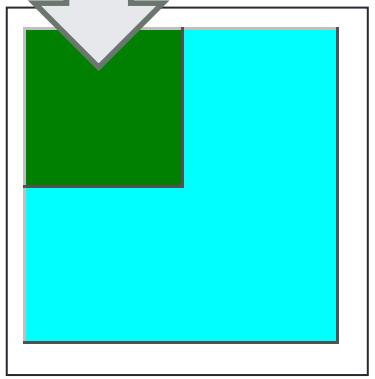
previousSibling

previousSibling funciona igual que nextSibling, pero selecciona el hermano anterior de un elemento.

```
<script type="text/javascript">
    //Esto detecta si No es Interne Explorer
    var NoIe = document.getElementById && !document.all;
    //Indica si estamos o no arrastrando el ratón
    var estoyArrastrando = false;
    //Variable para Guardar el Objeto que estamos moviendo
    var Objeto;

    function arrastrarRaton(e){
        if (estoyArrastrando) {
            //Sacamos la posición Actual del Raton
            PosX = NoIe ? e.clientX : event.clientX;
            PosY = NoIe ? e.clientY : event.clientY;
            // Y se la restamos al Objeto
            Objeto.style.left = PosX - parseInt(Objeto.style.width)/2;
            Objeto.style.top = PosY - parseInt(Objeto.style.height)/2;
            return false;
        }
    }

    function soltarBoton(e) {estoyArrastrando = false;}

    function presionarBoton(e) {
        //Vamos a arrastrar todos los objetos que tengan la clase
        //Buscamos cual es el elemento sobre el que se ha presionado el botón del ratón
        var ObjetoActivo = NoIe ? e.target : event.srcElement;

        // Buscamos el primer elemento que pertenezca a la clase.
        while (ObjetoActivo.tagName.toLowerCase() != "html" && ObjetoActivo.className != "Clase") {
            ObjetoActivo = NoIe ? ObjetoActivo.parentNode : ObjetoActivo.parentElement;
        }

        // Si hemos obtenido un objeto con esa clase...
        if (ObjetoActivo.className == "Clase") {
            //Indicamos que se empieza a arrastrar
            estoyArrastrando = true;
            // Guardamos los datos del ObjetoActivo al objeto que se está moviendo en la variable global
            Objeto = ObjetoActivo;
            // Devolvemos false para no realizar ninguna acción posterior
            return false;
        }
    }

    document.onmousedown = presionarBoton;
    document.onmouseup = soltarBoton;
    document.onmousemove = arrastrarRaton;

    document.oncontextmenu=new Function("return false");
</script>
```

```
<style type="text/css">
    /*Tiene que tener posición absoluta para poder arrastrarlo*/
    .Clase{position:absolute;cursor:move}
</style>
```

```
<body>
    <div class="Clase" style="width:100px;height:100px;background-color:cyan"></div>
    <div class="Clase" style="width:50px;height:50px;background-color:green"></div>
</body>
```



AJAX. Objeto XMLHttpRequest

AJAX es el acrónimo de *Asynchronous Javascript and XML (*Javascript y XML Asíncrono), es decir vamos a realizar peticiones Asíncronas empleando Javascript y XML.
Estás peticiones se denominan asíncronas por que el cliente realiza una petición al servidor y sigue trabajando hasta que le llegue la petición del servidor, no detiene la página y se queda en blanco hasta que llega. Aquí radica su utilidad, ya que haremos peticiones al servidor y le diremos que publique la respuesta en un DIV determinado, mientras en el resto de la página podemos hacer cualquier otra cosa.
Es decir, **con AJAX no será necesario recargar toda la página**. Ya que haremos peticiones desde dentro de la página usando Javascript, y si la petición es simple y no requiere intervención del servidor la respuesta será inmediata, y si requiere intervención del servidor la petición será asíncrona, y cuando llegue se mostrará sin que se interrumpa ni se tenga que recargar toda la web.
(veremos que en Ajax también podemos hacer peticiones síncronas y esperar a la respuesta)

El objeto que se suele utilizar para realizar estas peticiones es **XMLHttpRequest**, que sirve para hacer peticiones que llegarán en formato XML a través de HTTP.
Este objeto tiene una serie de Propiedades y métodos que necesitamos conocer. Métodos:

- **abort**() Cancela la petición actual (muy útil si vemos que tarda mucho)
- **getAllResponseHeaders**() Devuelve todos los encabezados HTTP: hora, servidor, tipo, tamaño..
- **getResponseHeader**("Nombre_Encabezado") Devuelve el valor de ese encabezado
- **open**("método", "URL", "async", "Usuario", "Pwd") Indicamos los atributos para establecer una conexión con el servidor: método:GET/POST… URL:que página web… No todos los atributos son necesarios, normalmente con 2 (método, URL) o 3 (método, URL, asíncrona) sobran.
- **setRequestHeader**("Etiqueta", "Valor") Para enviar valores de cabeceras personalizadas. Hay que llamarle siempre después de open y antes de send
- **send**(contenido) Envía la petición, se puede acompañar de argumentos o sino se indica null

AJAX. Objeto XMLHttpRequest

Las propiedades de **XMLHttpRequest**:

- **onreadystatechange** Se utiliza como manejador de eventos para ver cambios en el estado
- **readyState** Contiene el estado actual del objeto:
  - 0 Sin incializar (objeto creado, pero no se ha invocado el método open)
  - 1 Cargándose (objeto creado, pero no se ha invocado el método send)
  - 2 Cargado (se ha invocado el método send, pero el servidor aún no ha respondido)
  - 3 en Interacción (se han recibido algunos datos, pero no se puede emplear la responseText)
  - 4 Completado (se han recibido todos los datos de la respuesta del servidor)
- **responseText** Contiene la respuesta del servidor en formato String (Cadena de Texto)
- **responseXML** Contiene la respuesta del servidor en formato XML
- **status** El código de estado HTTP devuelto por el servidor:
  - 200 (OK) para una respuesta correcta
  - 304 (Not Modified) no modificado
  - 401 (Unauthorized) Sin autorización
  - 403 (Forbiden) cuando no encuentra la página, No existe
  - 404 (Not Found) No encontrado
  - 500 (Internal Server Error) Error del servidor
  - 503 (Service Unavialable) Cuando el servidor está demasiado saturado
- **statusText** El código de estado HTTP devuelto por el servidor, en texto (paréntesis anteriores)

AJAX. Ejemplo XMLHttpRequest

A. Creamos el Objeto XMLHttpRequest y le llamamos HTTPRequest (actuará como mini navegador dentro del cliente)

```
// 1. Instanciamos el objeto XMLHttpRequest
if(window.XMLHttpRequest) {  // Navegadores que siguen los estándares
  HTTPRequest = new XMLHttpRequest();
}
else if(window.ActiveXObject) {  // Navegadores obsoletos
  HTTPRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

B. Abrimos una conexión y le enviamos los datos y el método, en este caso GET y la página Ejemplo.html (URL del servidor)

```
// 3. Hacemos la llamada
HTTPRequest.open("GET", "Ejemplo.html" );
HTTPRequest.send(null);
```

C. Y por último (aunque este paso en el código debe ir antes que open) le decimos que cada vez que cambie el estado del objeto conexión, cuando el estado sea 4 que llene el contenido del DIV

```
// 2. Indicamos un manejador de eventos para cuando llegue la información
HTTPRequest.onreadystatechange=function() {
   if (HTTPRequest.readyState == 4 ) { // Por acortar podemos poner this.readyState
                document.getElementById("Aqui").innerHTML=HTTPRequest.responseText;
            }
}
```

Faltaría crear un DIV para mostrar los datos:

```
<body>
<div id="Aqui"></div>
</body>
```

AJAX. Ejemplo Menú Simple

```
<script>
    // Documento JavaScript
    function QuePagina (url, id_contenedor){
        // 1. Instanciamos el objeto XMLHttpRequest
        if(window.XMLHttpRequest) {  // Navegadores que siguen los estándares
          HTTPRequest = new XMLHttpRequest();
        }
        else if(window.ActiveXObject) {  // Navegadores obsoletos
          HTTPRequest = new ActiveXObject("Microsoft.XMLHTTP");
        }
        //2
        HTTPRequest.onreadystatechange=function() {
            if (this.readyState == 4 ) {
                CargarPagina(HTTPRequest, id_contenedor)
            }
        }
        function CargarPagina(HTTPRequest, id_contenedor){
                document.getElementById(id_contenedor).innerHTML=HTTPRequest.responseText;
        }
        //3
        HTTPRequest.open("GET",url);
        HTTPRequest.send(null);
    }
</script>
<body>
    <div id="menu">
        <a href="javascript:QuePagina('Ejemplo.html', 'Contenido');">Ejemplo</a>
        <a href="javascript:QuePagina('provincias.php', 'Contenido');">Provincias</a>
    </div>
    <div id="Contenido"></div>
</body>
```

AJAX. Mostrar Ocultar

```
<script>
    // 1. Instanciamos el objeto XMLHttpRequest
    if(window.XMLHttpRequest) {  // Navegadores que siguen los estándares
      HTTPRequest = new XMLHttpRequest();
    }
    else if(window.ActiveXObject) {  // Navegadores obsoletos
      HTTPRequest = new ActiveXObject("Microsoft.XMLHTTP");
    }

    var SignoMas=true; //Definimos una variable para mostrar el mas o el menos

    function MostrarOcultar(){
        if (SignoMas==true){
            document.getElementById("Signo").src="imagenes/menos.png";
            SignoMas=false;
            HTTPRequest.open("GET", "Ejemplo.html" );
            //
            HTTPRequest.onreadystatechange=function() {
                if (this.readyState == 4 && this.status == 200) {
                    document.getElementById("Aqui").innerHTML=this.responseText;
                }
            }
            HTTPRequest.send(null);
        } else {
            document.getElementById("Signo").src="imagenes/mas.png";
            SignoMas=true;
            document.getElementById("Aqui").innerHTML="";
        }
    }
</script>
<body>
<!-- tambien se puede poner <a href="javascript:void(0)" Que devuelve vacío-->
<a href="javascript://" onclick="MostrarOcultar()">
    <img id="Signo" src="imagenes/mas.png" width="30" height="30" />
</a>
<div id="Aqui"></div>
</body>
```

AJAX. Mostrar Ayuda cuando pasas el ratón por encima

```
<script>
    // 1. Instanciamos el objeto XMLHttpRequest
    if(window.XMLHttpRequest) {  // Navegadores que siguen los estándares
      HTTPRequest = new XMLHttpRequest();
    }
    else if(window.ActiveXObject) {  // Navegadores obsoletos
      HTTPRequest = new ActiveXObject("Microsoft.XMLHTTP");
    }

    function MostrarAyuda(IdAyuda, e){
        QueID=document.getElementById("Ayuda");
        QueID.style.visibility="visible";
        QueID.style.height="400px";
        QueID.style.width="400px";

        //Sacamos la posición Actual del Raton
        var NoIe = document.getElementById && !document.all;
        PosX = NoIe ? e.clientX : event.clientX;
        PosY = NoIe ? e.clientY : event.clientY;
        QueID.style.left=PosX+"px";
        QueID.style.top=PosY+"px";

        HTTPRequest.open("GET", IdAyuda+".html" );
        HTTPRequest.onreadystatechange=function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("Ayuda").innerHTML=this.responseText;
            }
        }
        HTTPRequest.send(null);
    }
    function OcultarAyuda(){
        QueID=document.getElementById("Ayuda")
        QueID.style.visibility="hidden";
        QueID.style.height="0px";
        QueID.style.width="0px";
    }
</script>
<body>
<a href="javascript://" onmouseover="MostrarAyuda('1', event)" onmouseout="OcultarAyuda()"><img src="imagenes/eliminar.gif" width="32" height="32" /></a>
<a href="javascript://" onmouseover="MostrarAyuda('2', event)" onmouseout="OcultarAyuda()"><img src="imagenes/modificar.gif" width="32" height="32" /></a>
<a href="javascript://" onmouseover="MostrarAyuda('3', event)" onmouseout="OcultarAyuda()"><img src="imagenes/nuevo.gif" width="32" height="32" /></a>
<div id="Ayuda" style="position:absolute"></div>
</body>
```

AJAX. Envío de parámetros al servidor

Hasta ahora hemos empleado el objeto XMLHttpRequest para realizar peticiones usando el método **GET**. Y ya sabemos que GET envía los parámetros concatenados en la URL: www.dominio.com/pagina.php?variable1=valor1&variable2=valor2.... pero dado que GET tiene un límite máximo de 512 bytes, vamos a estudiar como hacerlo a través de **POST**.

Cuando pulsamos sobre enviar en un Formulario, éste crea automáticamente una cadena con los nombres de los campos y su valor separados por el símbolo & (si pasamos los datos por un proxy como **Paros** lo vemos) .

El objeto XMLHttpRequest no tiene esa posibilidad vamos a ser nosotros los que creemos la cadena con los valores de los campos.
Y además de eso cuando enviamos datos a través del método POST tenemos que acompañarlos de una cabecera **Conten-Type** correcta sino los datos serán descartados por el servidor.
Para hacerlo emplearemos el método

setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

Y si además queremos ya pasarle el charset sería:

setRequestHeader("Content-Type", "application/x-www-form-urlencoded; charset=UTF-8");

Y meteremos como parámetro al método **send**() en el que hasta ahora pasábamos null, la cadena con los campos y sus valores.

Además limpiaremos la cadena con la función **encodeURIComponent**() que limpia los caracteres no válidos y para evitar problemas con la cache de los navegadores metemos numero aleatorio dentro de un campo inventado

```
<script>
    function CrearObjeto(){
        //Instanciamos el objeto XMLHttpRequest
        if(window.XMLHttpRequest) {  // Navegadores que siguen los estándares
          return new XMLHttpRequest();
        }
        else if(window.ActiveXObject) {  // Navegadores obsoletos
          return new ActiveXObject("Microsoft.XMLHTTP");
        }
    }

    function CreaCadenaCamposForm(){
        var Nombre = document.getElementById("Nombre");
        var Apellidos = document.getElementById("Apellidos");
        var Telefono = document.getElementById("Telefono");

        return "Nombre=" + encodeURIComponent(Nombre.value) +
            "&Apellidos=" + encodeURIComponent(Apellidos.value) +
            "&Telefono=" + encodeURIComponent(Telefono.value) +
            "&evitarcache=" + Math.random();

    }

    function EnviarObjeto() {
        Objeto = CrearObjeto();
        if(Objeto) {
            Objeto.onreadystatechange = MostrarResultado;
            Objeto.open("POST", "AjaxPOST_validar.php", true);

            Objeto.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
            var CadenaCamposForm = CreaCadenaCamposForm();
            Objeto.send(CadenaCamposForm);
      }
    }

    function MostrarResultado() {
      if(Objeto.readyState == 4) {
        if(Objeto.status == 200) {
          document.getElementById("Aqui").innerHTML = Objeto.responseText;
        }
      }
    }

</script>
```

```
<form>
    <table width="300" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td align="left"> Nombre:</td>
        <td align="left"> <input type="text" id="Nombre" name="Nombre"/></td>
      </tr>
      <tr>
        <td align="left"> Apellidos:</td>
        <td align="left"> <input type="text" id="Apellidos" name="Apellidos"/></td>
      </tr>
      <tr>
        <td align="left"> Telefono:</td>
        <td align="left"> <input type="text" id="Telefono" name="Telefono" /></td>
      </tr>
      <tr>
        <td colspan="2" align="center"> 
            <input type="button" value="Enviar Datos" onclick="EnviarObjeto()"/>
        </td>
      </tr>
    </table>
</form>
<div id="Aqui"></div>
```

Nombre:
Apellidos:
Telefono:
Enviar Datos

```
<?php
    echo "Valor del Campo Nombre=".$_POST["Nombre"]."<br>";
    echo "Valor del Campo Apellidos=".$_POST["Nombre"]."<br>";
    echo "Valor del Campo Telefono=".$_POST["Nombre"]."<br>";
?>
```

AJAX. encodeURIComponent

La función **encodeURIComponent**() reemplaza todos los caracteres que no se pueden utilizar de forma directa en las URL por su representación hexadecimal. Las letras, números y los caracteres - _ . ! ~ * ' ( ) no se modifican, pero todos los demás caracteres se sustituyen por su equivalente hexadecimal.
Por ejemplo reemplaza los espacios en blanco por %20, y el símbolo & por %26.
Y también se sustituyen todos los acentos y cualquier otro carácter que no se puede incluir directamente en una URL:

```
var cadena = "cadena de texto";
var cadena_limpia = encodeURIComponent(cadena);
// cadena_limpia= "cadena%20de%20texto";
var cadena = "cadena & con caracteres problemáticos / : =";
var cadena_limpia = encodeURIComponent(cadena);
//cadena_limpia =
cadena%20%26%20caracteres%20problem%C3%A1ticos%20%2F%20%3A%20%3D
```

JavaScript incluye una función contraria llamada **decodeURIComponent**() y que realiza la transformación inversa. Además, también existen las funciones **encodeURI**() y **decodeURI**() que codifican/decodifican una URL completa. La principal diferencia entre encodeURIComponent() y encodeURI() es que esta última no codifica los caracteres ; / ? : @ & = + $, #:

```
var cadena = "http://www.ejemplo.com/index.php?parametro=valor con ñ y &";
var cadena_limpia = encodeURIComponent(cadena);
// cadena_limpia =
http%3A%2F%2Fwww.ejemplo.com%2Findex.php%3Fparametro%3Dvalor%20con%20%C3%B1%20y%20%26
var cadena_limpia = encodeURI(cadena);
// cadena_limpia =
http://www.ejemplo.com/index.php?parametro=valor%20con%20%C3%B1%20y%20"
```

Validación de Datos. Expresiones regulares

Las Expresiones Regulares, podemos definirlas como un Patrón de caracteres que compararemos con los datos escritos para ver coincidencias. Podemos emplearlas en muchos lenguajes en nuestro caso vamos a evaluarlas con JavaScript, en el evento OnBlur antes de enviar los datos por AJAX. Para hacerlo tenemos que indicarle, Qué caracteres queremos emplear y de que tipo, para hacerlo tenemos la siguiente tabla:

^	Principio de entrada o línea.
$	Fin de entrada o línea.
*****	El carácter anterior 0 o más veces.
+	El carácter anterior 1 o más veces.
?	El carácter anterior una vez como máximo (es decir, indica que el carácter anterior es opcional).
.	Cualquier carácter individual, salvo el de salto de línea.
x|y	x o y.
{n}	Exactamente n apariciones del carácter anterior.
{n,m}	Como mínimo n y como máximo m apariciones del carácter anterior.
[abc]	Cualquiera de los caracteres entre corchetes. Podemos emplear un guión para un rango ([a-f] es [abcdef]).
[^abc]	Cualquier carácter que no esté entre corchetes. Podemos emplear un guión para un rango ([^a-f] es [^abcdef]).
\b	Límite de palabra (como un espacio o un retorno de carro).
\B	Cualquiera que no sea un límite de palabra.
\d	Cualquier carácter de dígito. Equivalente a [0-9].
\D	Cualquier carácter que no sea de dígito. Equivalente a [^0-9].
\f	Salto de página.
\n	Salto de línea.
\r	Retorno de carro.
\s	Cualquier carácter individual de espacio en blanco (espacios, tabulaciones, saltos de página o saltos de línea).
\S	Cualquier carácter individual que no sea un espacio en blanco.
\t	Tabulación.
\w	Cualquier carácter alfanumérico, incluido el de subrayado. Equivalente a [A-Za-z0-9_].
\W	Cualquier carácter que no sea alfanumérico. Equivalente a [^A-Za-z0-9_].

Validación de Datos. Expresiones regulares

Por ejemplo para validar un campo fecha *(sin entrar a que la fecha sea correcta: Febrero máximo 28 días, bisiesto…)*

1. Primero definimos el patrón:
   Queremos 2 números para el día luego un separador , luego 2 números para el mes, separador y luego 4 números para el año. Traducido sería una expresión regular como la que sigue:
   ^\d{2}\/\d{2}\/\d{4}$
   - Empieza y termina con el carácter **^ …. $** con esto decimos que la comparación debe empezar desde el principio y llegar hasta el final (como son caracteres especiales les anteponemos la contrabarra /)
   - Luego vienen 2 números **\d{2}** de esta forma obligamos a que metan 2 digitos (de 01 a 31) si queremos de 1-31 la expresión sería **\d{1,2}** -> que significa 1 o 2 caracteres
   - Luego el separador "**/**" que como es un carácter especial le anteponemos la contrabarra **\/**
   - Y así seguimos … hasta los 4 caracteres del año
2. Asociamos esa expresión regular a una variable: var FechaRegEx = /^\d{2}\/\d{2}\/\d{4}$/;
3. Y por último mediante la función match la comparamos al valor del campo:
   if(!fvalor_campo.match(FechaRegEx)) { …..

```
<tr>
  <td>Fecha Nacimiento:</td>
  <td><input type="text" name="Fecha" id="Fecha" onBlur="Validar_Fecha(this.value,'Fecha')"></td>
</tr>
```

```
function Validar_Fecha(Quefecha,QueCampo){
    var FechaRegEx = /^\d{2}\/\d{2}\/\d{4}$/;
    if(!Quefecha.match(FechaRegEx)) {
        alert("formato de fecha no valido (dd/mm/aaaa)");
        setTimeout("document.getElementById('"+ QueCampo + "').className='TextoError'",75);
        setTimeout("document.getElementById('"+ QueCampo + "').focus()",75);
        return false;
    } else {
        setTimeout("document.getElementById('"+ QueCampo + "').className='TextoBien'",75);
    }
}
```

jQuery

jQuery es una **librería** JavaScript, libre y de código abierto, que nos permite realizar casi todo lo visto hasta en JavaScript pero de forma más sencilla (acceder a elementos del DOM *id=…,* manejar eventos *onclick=..* y usar AJAX)

- Dado que es una librería **js**, cuando queramos emplearla lo primero que hay que hacer es llamarla, para ello la meteremos en el <head> de la página web con un enlace:
  <script type="text/javascript" src="jquery.js"></script>
- Como son funciones JS cuando queramos emplearlas lo haremos dentro de los bloques de código <script type="text/javascript"> </script > o simplemente <script > Y para acceder a este conjunto de funciones debemos identificarlas anteponiendo **$()** o **jQuery()**
- Normalmente antes de indicarle al documento que hacer, tenemos que ver si esta listo, y esto se consigue con la función **$(document).ready()** o sólo **$()** por ejemplo para que nos muestre un *alert* cuando podamos empezar pondríamos el código:

```
$(document).ready( function() {
        alert ('documento preparado'); //Aquí metemos todo lo que queremos hacer
} );
```

- Y para seleccionar los elementos del DOM podemos acceder a ellos:
  - Mediante su ID — $('#id');
  - Por el nombre de su clase — $('.nombre_de_la_clase');
  - Por algún atributo del elemento — $('input[name=FECHA]'); *esto puede ser muy lento*
    *además también podemos emplear : *= (que contenga) ~= (que contenga exacta)*
  - Por su selector CSS — $('#menu ul li');
  - O mezclando selectores:
    $('a.QueClase:first'); -> selecciona el primer elemento <a> con la clase 'QueClase'
    $('tr:odd'); -> selecciona todos los elementos <tr> impares de una tabla
    $('#Form :input'); -> selecciona todos los elementos input dentro del formulario #Form
    $('div:visible'); -> selecciona todos los divs visibles (JQuery considera un objeto oculto cuando es *hidden* y cuando el elemento no se ve por que su altura y anchura es 0, siempre que no sea un <tr>
    $('div:gt(2)'); -> selecciona todos los divs excepto los tres primeros

Tenemos una descripción (en ingles) de todos los selectores en: http://api.jquery.com/category/selectors/

jQuery

- Otras posibilidades de selección:
  $('div.cuerpo').has('p'); los elementos <p> del div cuerpo
  $('div.menu li').not('.menu_seleccionado'); los elementos de menu que no sean la clase …
  $('ul li').filter('.normal'); los li cuya clase sea normal
  $('ul li').first(); $('ul li').eq(5); el primero y 6 item de la lista li

- Para seleccionar los elementos de un formulario podemos acceder a ellos usando su tipo (anteponiendo '**:**' , por ejemplo $('#Form :input');)
  :button (todos los type='button') :checkbox (los type='checkbox')
  :checked (los checkbox seleccionados) :disabled (todos los elementos deshabitados)
  :enabled (los habilitados) :file (los type='file') :image :input (los <input>,<textarea> y <select>)
  :password :radio :reset :selected (los <options> que están seleccionados) :submit y :text

- Para evaluar que haga algo si ha encontrado un elemento no basta con poner if ($('div.contenido')) { ... } Ya que esto siempre devuelve un objeto con lo que será true, para hacerlo mejor usamos la función *lenght* que ya vimos if ($('div.contenido ').length) { ... }

- Una vez realizada una búsqueda podemos guardar el resultado en una variable var divs = $('div');

- Y también podemos seguir seleccionando… Por ejemplo mediante la propiedad ***.html*** cambiamos el contenido de un elemento, siempre que el elemento ya este cargado por flujo normal, por lo tanto para reemplazar el texto del primer h1 de contenido pondríamos después del elemento (para evitar problemas lo ponemos al final y listo):
  $('#contenido').find('h1').eq(0).html('Texto nuevo para el primer h1 de contenido');
- Otra cosa que tenemos que tener en cuenta es que por ejemplo en el caso anterior .html() obtiene el valor del elemento y .html('Texto') reemplaza el valor del texto

jQuery

- Y en cuanto a las propiedades **CSS, Estilos y Dimensiones** de los elementos:
  - Mediante $('#contenido').find('h1').css('fontSize'); obtenemos los valores, en este caso el tamaño de la fuente
  - Y si queremos cambiarlos sería $('h1').css('fontSize', '100px');
  - O si necesitamos cambiar más de un valor

```
$('h1').css({  'fontSize' : '100px',
               'color' : 'red'
            });
```

  - También podemos hacer cambios relativos (sumamos 15px al tamaño de letra) 'fontSize' : '+=15px' o cualquier otra propiedad por ejemplo paddingTop' : '+=20px'
- Añadir o quitar clases: var $h1 = $('h1'); $h1.addClass('grande'); $h1.removeClass('grande');
- Cambiar-Obtener tamaños.. $('h1').width('50px'); $('h1').height('50px'); $('h1').position();
- Añadir atributos (cualquiera de los vistos hasta ahora)
  $('a').attr({ 'title' : 'Texto de Vínculo', 'href' : 'www.google.es' });
  $("#ImagenTal").attr("src", "imagenes/NombreImagen.png");
- Recorrer el DOM: $('h1').next('p'); ...last(); O si buscamos padres-hijos-hermanos: .parent() .children() siblings(); la documentación completa: http://api.jquery.com/category/traversing/
- Modificar objetos, por ejemplo su contenido como hemos visto antes con el .html("Texto") igualmente con .text(..) .val(…) la documentación: http://api.jquery.com/category/manipulation/.
- También podemos crear objetos mediante: insertAfter y after, insertBefore y before, appendTo y append, prependTo y prepend, clone(), remove (), detach()

```
var $NuevoElemento = $('<p>Nuevo elemento</p>');
$NuevoElemento.appendTo('#contenido');
$NuevoElemento.insertAfter('ul:last'); // eliminará al elemento <p> existente en #contenido
$('ul').last().after($NuevoElemento.clone()); // clonar al elemento <p> // para tener los dos
```

jQuery. Eventos

- Otro ejemplo de mover elementos del DOM

```
// hacer que el primer item de la lista sea el último
var $li = $('#menu li:first').appendTo('#menu');
//Otra forma de hacerlo
$('#menu').append($('#menu li:first'));
```

- Y de Copiar elementos:

```
// copiar el primer elemento de la lista y moverlo al final de la misma
$('# menu li:first').clone().appendTo('#menu');
```

- La documentación completa está en: http://api.jquery.com/category/events/. Y http://api.jquery.com/category/events/event-object/.
- Siguiendo lo visto hasta ahora podemos agregar evento onclick, blur change…

```
$('div.tal').click(function() {
                    alert('Has hecho click');
                  });
$('p').on('click', function() {
                  alert('Has hecho click');
                  });
```

- Para desvíncular una acción de un elemento `$('p').off('click');`
- O para indicar que se ejecute un elemento una sola vez:

```
$('#contenido h1').one('click', function(){
                                  alert('Has echo click por primera vez');
                                  $(this).click(function() { alert('Y has vuelto a hacer click'); });
                                  }
                                );
```

jQuery. Eventos

- Y para meter mas de un evento a la vez

```
$('#contenido h1').on({
        'click': function() { alert('Has echo click');},
        'mouseover': function() {alert('Y te has hido con el ratón');}
        });
```

- Objeto evento, también como hacíamos en JavaScript con la e podemos obtener datos del evento:
  - pageX, pageY La posición del puntero del ratón en el momento que el evento ocurrió, relativo a las zonas superiores e izquierda de la página.
  - type El tipo de evento (por ejemplo “click”).
  - which El botón o tecla presionada.
  - data Alguna información pasada cuando el evento es ejecutado.
  - target El elemento DOM que inicializó el evento.
  - preventDefault() Cancela la acción predeterminada del evento.
  - stopPropagation() Detiene la propagación del evento sobre otros elementos.

Funcionalidad Ajax mediante jQuery

Conociendo Javascript veremos que darle funcionalidad Ajax a una web mediante jQuery es muy sencillo. Para entenderlo, vamos a preparar un ejemplo muy simple:

- Suponiendo que tenemos una web con un menu, y un enlace como este:

```
<a href="#" id="enlace_ajax">Menu Ajax</a>
```

- Creamos un id dónde mostraremos el enlace de la página:

```
<div id="destino"></div>
```

- Y Mediante jQuery le asignamos un evento al enlace:

```
$(document).ready(function(){
    $("#enlace_ajax").click(function(evento){
      //elimino el comportamiento por defecto del enlace
      evento.preventDefault();
      //Y le digo que muestre la página… en el div llamado destino
      $("#destino").load("web_destino.html");
    });
 })
```

- Además si necesitamos pasar parámetros con la URL podemos hacerlo con la tipica notación de propiedades y valores jQuery: {idioma: “ESP", id: 45}. Y además podemos llamar a una función cuando se termine de ejecutar el método (por ejemplo ocultar mensaje cargando):

```
$(document).ready(function(){
    $("#enlaceajax").click(function(evento){
      evento.preventDefault();
      $("#destino").load("web_destino.php", {idioma: “ESP", id: 45}, function(){
        alert(“Datos recibidos correctamente por ajax"); // o div visibility hidden…
      });
    });
 })
```

- En la web destino leeríamos los datos que recibimos por POST como siempre:

```
Recibido el siguiente dato:< br>Idioma: <?php echo $_POST["idioma"];?>
```