

UT8 GENERACIÓN DE PÁGINAS WEB INTERACTIVAS

Índice

- [UT8 GENERACIÓN DE PÁGINAS WEB INTERACTIVAS](#)
 - [Índice](#)
 - [Programación del cliente web](#)
 - [Comunicación asíncrona con el servidor web: AJAX](#)
 - [PHP y Javascript](#)
 - [Ajax](#)

Programación del cliente web

Muchas de las aplicaciones web que existen en la actualidad tienen dos componentes una parte de la aplicación, generalmente la que contiene la lógica de negocio, que se ejecuta en el servidor y otra parte de la aplicación, de menor peso, se ejecuta en el cliente.

Existen incluso cierto tipo de aplicaciones web, como Google Docs, en las que gran parte de las funcionalidades que ofrecen se implementan utilizando programación del cliente web.

Ahora aprenderemos a integrar estos dos componentes de una misma aplicación web: el código PHP que se ejecutará en el servidor, con el código que se enviará al cliente para que éste lo ejecute.

La etiqueta que se utiliza para integrar el código ejecutable por el navegador junto al resto de etiquetas es "script".

La ejecución de código en el navegador encaja perfectamente con cierto tipo de tareas:

- Como comprobar y/o procesar los datos que introduce el usuario en los formularios, como paso previo a su envío al servidor web.
- Gestionar diferentes ventanas del navegador.
- Modificar de forma dinámica los elementos que componen la página web, ajustando sus propiedades o estilos en respuesta a la interacción del usuario.

Por ejemplo:

```
<form action="usuario.php" method="get" name="datos_usuario" onsubmit="return
validar_email()">
<input type="text" id="email" />
</form>

//Definimos la función:
function validar_email (){
valor =document.getElementById("email").value;
pos_arroba = valor.indexOf("@");
pos_punto = valor.lastIndexOf(".");
if (pos_arroba < 1 || pos_punto < pos_arroba+2 || pos_punto+ valor.length ){
    alert("Dirección de correo no valida");
    return false; }
return true; }
```

El lenguaje de guiones que se utiliza mayoritariamente hoy en día para la programación de clientes web es **JavaScript**.

Su sintaxis está basada en la del lenguaje C, parecida a la que conocemos del lenguaje PHP.

Si bien, la gran mayoría de navegadores web soportan código en lenguaje JavaScript debes tener en cuenta que:

- La ejecución de JavaScript en el navegador puede haber sido deshabilitada por el usuario.
- La implementación de JavaScript puede variar de un navegador a otro. Lo mismo sucede con el interface de programación que usa JavaScript para acceder a la estructura de las páginas web el DOM. Por este motivo, hay que verificar la funcionalidad del código en diversos navegadores antes de publicarlo.

Comunicación asíncrona con el servidor web: AJAX

Una de las principales causas de la evolución de JavaScript en los últimos tiempos es, sin duda, la tecnología **AJAX**.

Entre las tareas que puedes llevar a cabo gracias a AJAX están:

- Actualizar el contenido de una página web sin necesidad de recargarla
- Pedir y recibir información desde un servidor web manteniendo la página cargada en el navegador
- Enviar información de la página a un servidor web en segundo plano

PHP y Javascript

Vamos a realizar un ejercicio con una de las tareas que con más frecuencia hacen uso de Javascript la validación de formularios.

Si no utilizas código ejecutable en el navegador, la única forma de validar un formulario consiste en enviarlo al servidor web para comprobar si existen errores en los datos. En caso de que así sea, habrá que volver a enviar el formulario al navegador del usuario mostrando las advertencias oportunas.

Obviamente, si utilizas JavaScript en tus aplicaciones, la validación se puede realizar en el cliente web. De esta forma el proceso es mucho más rápido. No es necesario enviar la información al servidor hasta que se haya comprobado que no existen errores de validación.

Sin embargo, aunque parecen claras las ventajas de la validación de formularios en el cliente web, hay ocasiones en las que ésta no es posible. A veces el navegador que utiliza el usuario no tiene capacidad para ejecutar código JavaScript o incluso puede suceder que se haya deshabilitado por motivos de seguridad.

En los casos que no sea posible asegurar la capacidad de ejecución de código de los clientes, la solución óptima sería utilizar un escenario dual.

Supongamos que queremos validar los datos de un formulario que contiene los siguientes campos:

- Nombre
- DNI
- Contraseña
- Repita la contraseña

Si hacemos la validación en PHP, podemos generar junto con la página web el texto con las advertencias de validación. Y si el formulario se valida también utilizando JavaScript tendremos que crear los mismos textos o similares.

Una posibilidad para no repetir el código que introduce esos textos en el cliente y en el servidor, es introducir los textos de validación en las etiquetas HTML de la página web, y utilizar estilos para mostrarlos, o no, según sea oportuno. Por ejemplo:

```
<span class = 'error'>El nombre debe tener más de 3 caracteres </span>  
<span class = 'error'>El DNI es incorrecto </span>  
<span class = 'error'>La contraseña debe ser mayor de 5 caracteres o no coinciden  
</span>
```

El código PHP y JavaScript deberá ocultar cada uno de los textos cuando la validación de su elemento respectivo sea correcta. Por ejemplo, si el nombre tiene más de tres letras, validará correctamente y no se deberá mostrar el primer mensaje.



Hoja08_PHPJavaScript_01



Hoja08_PHPJavaScript_02

Ajax

La tecnología AJAX se utiliza desde el cliente web para permitir comunicaciones asíncronas con el servidor web, sin necesidad de recargar la página web que se muestra en el navegador. Se basa en la utilización de código en lenguaje JavaScript.

jQuery posee varios métodos para trabajar con Ajax. Sin embargo, todos están basados en el método **\$.ajax**.

El método **\$.ajax** es configurado a través de un objeto y contiene todas las instrucciones que necesita jQuery para completar la petición. Es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no.

Algunas opciones del método son:

- **url**: establece la URL en donde se realiza la petición. La opción url es obligatoria para el método \$.ajax.
- **data**: establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo foo=bar&&baz=bim)
- **type**: de forma predeterminada su valor es GET. Otros tipos de peticiones también pueden ser utilizadas (como PUT y DELETE), sin embargo pueden no estar soportados por todos los navegadores.
- **dataType**: establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME que posee la respuesta.
- **success**: establece una función a ejecutar si la petición a sido satisfactoria. Dicha función recibe como argumentos la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma y el objeto de la petición en crudo.
- **error**: establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición. Ejemplo:

```
<script>
    $(document).ready(function(){
        $("button[name='add']").click(function(){
            $.ajax({
                type: "POST",
                url: "index.php?option=productos",
                dataType: "json",
                data: {"accion":"add","producto":this.id},
                success : function(data){
                    $("#cesta").html(data.contenido_cesta);
                    $("#botonVaciar").prop('disabled',data.botones);
                    $("#botonComprar").prop('disabled',data.botones);
                }
            });
            window.location="index.php?option=productos";
        });
    });
});
```


```


    //... debajo se podría realizar el de borrar y por último , se cerraria el
script
</script>
//Otro ejemplo puede ser el buscador :

<script>
    $(document).ready(function(){
        $("#resultadoBusqueda").html('<p>Buscador</p>');
    });
    function buscar(){
        var textoBusqueda=$("#input#busqueda").val();
        if (textoBusqueda != ""){
            $.post("index.php?option=buscador",
{valorBusqueda:textoBusqueda},function(mensaje){
                $("#resultadoBusqueda").html(mensaje);
            });
        } else{
            $("#resultadoBusqueda").html('<p>JQUERY VACIO</p>');
        }
    };
};
</script>

```

Para comunicarse con el servidores, se puede utilizar la notación **JSON**. Json es un formato de intercambio de información más sencillo de procesar que XML (especialmente al utilizar el lenguaje JavaScript) para transmitir la información con el servidor

 Hoja08_AJAX_01

 Hoja08_AJAX_02