Guía de Funciones PHP y Validaciones en JavaScript

Este documento explica las principales funciones y técnicas utilizadas en los ejercicios de PHP y JavaScript desarrollados, con especial atención a la sanitización de datos, el manejo de arrays y las validaciones del lado del cliente.

1. Funciones de PHP usadas en los ejercicios

htmlspecialchars()

Convierte caracteres especiales en entidades HTML. Evita ataques XSS (Cross Site Scripting) al mostrar texto del usuario. Ejemplo: <script>alert('x')</script> se convierte en texto visible, no en código ejecutable. Uso: htmlspecialchars($variable, ENT_QUOTES, 'UTF-8');

strip_tags()

Elimina etiquetas HTML o PHP de una cadena. Se usa para limpiar texto de entrada antes de guardarlo o mostrarlo. Uso: $texto_limpio = strip_tags($entrada);

filter_var()

Permite filtrar y validar datos. Aunque FILTER_SANITIZE_STRING está deprecado, se puede usar con otros filtros (ej. FILTER_VALIDATE_EMAIL). Uso: $email = filter_var($email_raw, FILTER_VALIDATE_EMAIL);

preg_match()

Realiza una comparación mediante expresiones regulares. Ideal para validar formatos personalizados, como nombres o contraseñas. Uso: preg_match('/^[\p{L}\s\-]+$/u', $nombre);

array_map()

Aplica una función a cada elemento de un array y devuelve un nuevo array. Uso: $cuadrados = array_map(fn($n) => $n*$n, $numeros);

array_filter()

Filtra un array usando una función callback. Solo conserva los elementos que devuelvan true. Uso: $primos = array_filter($numeros, fn($n) => es_primo($n));

array_walk()

Modifica un array directamente, aplicando una función a cada elemento (por referencia). Uso: array_walk($valores, fn(&$v) => $v *= 2);

implode()

Convierte un array en una cadena de texto, separando los elementos con un delimitador. Uso: implode(', ', $nombres);

htmlspecialchars($_SERVER['PHP_SELF'])

Protege el atributo action en formularios para evitar ataques XSS mediante rutas manipuladas. Uso: <form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>">

range()

Crea un array con una secuencia de números. Uso: $numeros = range(1, 100);

2. Validaciones en JavaScript

Las validaciones en JavaScript se realizan del lado del cliente, antes de enviar el formulario al servidor. Esto mejora la experiencia del usuario y reduce carga en el servidor, aunque no sustituye la validación en PHP.

addEventListener('submit', ...)

Permite interceptar el envío del formulario para validar los datos antes de que lleguen al servidor.

preventDefault()

Detiene el envío del formulario si no se cumplen las condiciones de validación.

checked

Propiedad booleana de los checkbox que indica si está seleccionado.

trim()

Elimina espacios al inicio y final de una cadena. Evita entradas vacías aparentes.

alert()

Muestra un mensaje emergente simple al usuario para indicar un error o confirmación.

textContent

Permite mostrar mensajes de error dentro del HTML sin recargar la página.

Ejemplo completo de validación en JavaScript:

```
document.addEventListener("DOMContentLoaded", () => { const form =
document.getElementById("textoForm"); const upper =
document.getElementById("upper"); const lower = document.getElementById("lower");
const mensajeError = document.getElementById("mensajeError");
form.addEventListener("submit", (e) => { if (!upper.checked && !lower.checked) {
e.preventDefault(); mensajeError.textContent = "Debes seleccionar al menos una
opción."; } else { mensajeError.textContent = ""; } }); });
```

Conclusión

Las funciones y validaciones vistas permiten crear aplicaciones PHP más seguras y eficientes. El uso de htmlspecialchars(), strip_tags() y las validaciones en el lado del cliente proporcionan una capa sólida de protección frente a ataques comunes y errores de usuario.