FORMACIÓN PROFESIONAL
MONTECASTELO

Validación de formularios

Desarrollo Web en Entorno Cliente

Ciclo Superior de Desarrollo de Aplicaciones Web

2020/2021

Formularios HTML

- Las páginas HTML pueden incluir formularios que permiten que el usuario introduzca información y la envíe al servidor.
- Un formulario consiste en un conjunto de campos de entrada agrupados dentro de una etiqueta <form>.

Formularios y el DOM

- Puesto que forman parte de la página, los elementos de un formulario también forman parte del DOM.
- Los elementos del DOM que representan campos de formulario ofrecen una serie de propiedades, métodos y eventos que no están presentes en otros elementos.
- De ese modo pueden ser inspeccionados y controlados mediante JavaScript para facilitar y gestionar la entrada de información por parte del usuario.

Validación

- La validación de datos consiste en comprobar que los datos introducidos en la aplicación cumplen el formato.
- Si la validación es correcta, la aplicación pasa a procesar los datos.
- En caso contrario se debe notificar al usuario con un mensaje indicativo acerca de los errores que se han producido en el formato de la información.

Validación

- La validación de datos es una de las tareas fundamentales que JavaScript implementa en la mayoría de las aplicaciones web con entrada de datos por parte del usuario.

Motivos para validar

- Se debe implementar validación en los formularios web, a pesar de que la validación pueda perjudicar la experiencia de usuario.
- La aplicación no funcionará adecuadamente si el usuario no introduce la información correcta, si los datos no se almacenan en el formato adecuado, o directamente si se omite la introducción de la información.
- La validación también contribuye a la seguridad del sistema, ya que existen varios modos por los que usuarios malintencionados pueden emplear formularios no validados para dañar la aplicación.

Motivos para validar

- En cualquier caso, NUNCA se debe confiar en la información enviada por el cliente al servidor.
- Aunque el entorno cliente esté realizando la validación correctamente, un usuario malicioso puede alterar la petición a través de la red.
- **Por tanto una aplicación debe validar tanto en entorno cliente como en entorno servidor.**

Tipos de validación

- La validación puede dividirse en dos tipos en función del entorno en el que se lleve a cabo: validación en el cliente y validación en el servidor.
- Ambos tipos de validación deben convivir en una aplicación web, tal como se acaba de justificar.

Validación en el servidor

- Ocurre en el servidor tras recibir la información enviada por el cliente.
- La validación del servidor se encarga de validar los datos antes de procesarlos o almacenarlos en la base de datos .
- Si la validación falla, se envía una respuesta al cliente con los errores detectados.
- La validación del servidor es la última línea de defensa contra datos incorrectos o malintencionados.

Validación en el cliente

- Ocurre en el navegador antes de que la información sea enviada al servidor.
- Es más amigable que la validación de servidor porque genera un mensaje en cuanto se intenta enviar el formulario.
- El presente módulo se centra en la validación en el cliente, que puede realizarse mediante HTML5, JavaScript o una combinación de ambos.

Validación en el cliente

- Existen dos grandes tipos de validación en el cliente:
  - Validación nativa mediante HTML5.
  - Validación mediante JavaScript.

Validación HTML5

- HTML5 ofrece la posibilidad de validar ciertos datos de usuario sin necesidad de scripts.
- Esta validación se lleva a cabo a través de atributos de validación, lo cuales permiten establecer reglas para los campos del formulario.
- Las reglas permiten establecer si el campo es obligatorio, si la información es numérica o de otro tipo, valores máximo y mínimo, etc.

Validación HTML5

- La validación de HTML5 se basa fundamentalmente en los siguientes dos aspectos.
- Por un lado, el tipo de campo, especificado a través del atributo `type` de la etiqueta `<input>`.
- Por otro lado, determinados atributos de la etiqueta `<input>` que establecen restricciones adicionales.

Validación HTML5

- La validación de HTML5 tiene mejor rendimiento que la de JavaScript.
- No obstante, si bien es personalizable, no es tan personalizable como la validación mediante JavaScript.

Validación HTML5

- Ejemplo 1:

```
<input type="email“ id=“correo“ required>
```

- Cuando se intente enviar un formulario que contiene la etiqueta anterior:
  - El atributo `type` provoca que el navegador rechace entradas que no tengan formato de correo electrónico.
  - El atributo `required` provoca que se cancele el envío si dicha entrada no está presente.

Validación HTML5

- Ejemplo 2:

```
<input type="number" id=“porcentaje“ max=“100">
```

- Cuando se intente enviar un formulario que contiene la etiqueta anterior:
  - El atributo `type` provoca que el navegador rechace entradas que no tengan formato numérico.
  - El atributo `max` provoca que se rechacen números mayores que 100.

Validación HTML5

- Si bien se presenta aquí la validación basada en HTML5, en el presente módulo el foco estará en la validación basada en JavaScript.

Validación con JavaScript

- La validación mediante JavaScript puede complementar o incluso sustituir a la validación mediante HTML5.
- Las validaciones mediante JavaScript permiten una mayor flexibilidad, a cambio de requerir más código.

Validación con JavaScript

- Es recomendable basar este tipo de validación en una función para cada una de las validaciones requeridas.
- De ese modo se puede llamar a cada una de forma independiente.
- Las funciones deberían devolver `true` si la validación ha sido correcta o `false` (junto con los mensajes de error solicitados) si la validación ha sido incorrecta, tal como se muestra en los ejemplos.

Expresiones regulares

- Las expresiones regulares representan patrones en cadenas de texto.
- Son construcciones muy útiles en la validación de formularios para reconocer patrones.
- Están definidas en base a una sintaxis propia que, hasta cierto punto, es independiente del lenguaje de programación.

Sintaxis de expresiones regulares

- **abc** Una secuencia de caracteres
- **[abc]** Un carácter del conjunto
- **[^abc]** Un carácter que *no* sea del conjunto
- **[0-9]** Un carácter del rango
- **x+** Una o más ocurrencias
- **x*** Cero o más ocurrencias
- **x?** Cero o una ocurrencia
- **x{2,4}** De dos a cuatro ocurrencias
- **(abc)** Un grupo

Sintaxis de expresiones regulares

- **a|b|c** Operación OR
- **\d** Cualquier cifra
- **\w** Cualquier valor alfanumérico
- **\s** Espacio
- **.** Cualquier caracter (salvo cambio de línea)
- **^** Marca el comienzo de la cadena
- **$** Marca el fin de la cadena

Expresiones regulares

- Ejemplo: código postal

  ```
  ^\d{5}$
  ```

- Ejemplo: DNI

  ```
  ^\d{8}[A-Z]$
  ```

- Ejemplo: matrícula española

  ```
  ^\d{4}[A-Z]{3}$
  ```

Visualización

- Se puede comprobar visualmente el funcionamiento de las expresiones regulares en el siguiente sitio web:

https://www.debuggex.com/

https://regexr.com/

Expresiones regulares en JS

- JavaScript ofrece un objeto nativo llamado `RegExp` para trabajar con expresiones regulares.
- Así mismo, puesto que las expresiones regulares se aplican sobre cadenas, los objetos `String` también ofrecen métodos que permiten aplicar expresiones regulares.

RegExp

- RegExp cuenta con un método para comprobrar (`test`) si una determinada cadena cumple la expresión regular.
- También tiene un método de ejecución (`exec`) que devuelve un array con todas las coincidencias de la expresión regular que existen en la cadena.

String

- Las cadenas tienen un método de coincidencia (`match`) para compararlas con una expresión regular y un método de búsqueda (`search`) para buscar una cadena, devolviendo solo la posición inicial de la coincidencia.
- También cuentan con un método de reemplazo (`replace`) puede reemplazar coincidencias de un patrón con una cadena.

Expresiones regulares

- Las expresiones regulares son de gran utilidad en la validación de campos de formulario.
- Simplifican enormemente algunas tareas, sin embargo pueden volverse difíciles de manejar si se aplican a problemas demasiado complejos.
- En tal caso es recomendable recurrir a la validación mediante instrucciones nativas de JavaScript.

FORMACIÓN PROFESIONAL

MONTECASTELO