

UT 6_2 - Interacción con el usuario: Formularios

JS

Marina Navarro Pina



Formularios: Validación

Se trata de una verificación útil porque evita enviar datos al servidor que sabemos que no son válidos pero **NUNCA** puede sustituir a la validación en el lado servidor ya que en el lado cliente se puede manipular el código desde la consola para que se salte las validaciones que le pongamos.

Básicamente tenemos 2 maneras de validar un formulario en el lado cliente:

- usar la **validación incorporada en HTML5** y dejar que sea el navegador quien se encargue de todo
- realizar nosotros la validación mediante Javascript

Formularios: Validación incorporada en HTML5

Esta opción permite que el navegador valide automáticamente los campos del formulario mediante atributos HTML.

Algunos de los atributos más comunes son:

- ❑ **required**: Indica que el campo es obligatorio.
- ❑ **pattern**: Define una expresión regular que el contenido del campo debe cumplir.
- ❑ **minlength** y **maxlength**: Establecen la longitud mínima y máxima del contenido.
- ❑ **min** y **max**: Especifican los valores mínimo y máximo para campos numéricos.

También producen errores de validación si el contenido de un campo no se adapta al **type** indicado (email, number, ...) o si el valor de un campo numérico no cumple con el *step* indicado.

Formularios: Validación incorporada en HTML5

Cuando el contenido de un campo es valido dicho campo obtiene automáticamente la pseudoclase `:valid` y si no lo es tendrá la pseudoclase `:invalid` lo que nos permite poner reglas en nuestro CSS para destacar dichos campos, por ejemplo:

```
input:invalid {  
  border: 2px dashed red;  
}
```

Además de las anteriores tenemos las pseudoclases `:required`, `:optional`, que nos permiten poner reglas de estilo a los campos obligatorios o no, y `:focus` para el campo con el foco.

La validación del navegador se realiza al enviar el formulario. Si encuentra un error lo muestra, se detiene la validación del resto de campos y no se envía el formulario.

Formularios: Validación mediante API JavaScript

Este enfoque permite tener un mayor control sobre el proceso de validación. Utilizando JavaScript, se pueden implementar validaciones personalizadas y mostrar mensajes de error más claros.

La API proporciona métodos y propiedades útiles, como:

- ❑ **checkValidity()**: Verifica si un campo (o formulario) es válido.
- ❑ **validationMessage**: Contiene el mensaje de error si el campo no es válido.
- ❑ **setCustomValidity(mensaje)**: Permite establecer un mensaje de error personalizado.

Formularios: Validación mediante API JavaScript

- ❑ **validity**: es un objeto que tiene propiedades booleanas para saber qué requisito del campo es el que falla:
 - *valueMissing*: **true** si el campo tiene el atributo **required** y no se ha introducido ningún valor en él. Es útil para verificar si el usuario ha dejado un campo obligatorio vacío.
 - *typeMismatch*: **true** si el contenido del campo no coincide con el tipo especificado en el atributo **type**. Por ejemplo, si un campo tiene `type="email"` y el usuario introduce un texto que no es una dirección de correo electrónico válida, esta propiedad será **true**.
 - *patternMismatch*: **true** si el valor del campo no cumple con la expresión regular definida en el atributo **pattern**. Esto permite validar formatos específicos, como códigos postales o números de teléfono.
 - *tooShort* / *tooLong*: indican si el valor del campo no cumple con los requisitos de longitud establecidos por los atributos **minlength** o **maxlength**, respectivamente. *tooShort* será **true** si el valor es *más corto* de lo permitido, y *tooLong* será **true** si es *más largo* de lo permitido.

Formularios: Validación mediante API JavaScript

- *rangeUnderflow / rangeOverflow*: se aplican a campos numéricos. *rangeUnderflow* es **true** si el valor ingresado es *menor que el mínimo* especificado por el atributo *min*, mientras que *rangeOverflow* es **true** si el valor es *mayor que el máximo* especificado por el atributo *max*.
- *stepMismatch*: es **true** si el valor del campo no cumple con el paso especificado por el atributo *step*. Por ejemplo, si un campo numérico tiene un *step* de 0.5 y el usuario introduce un número que no es múltiplo de 0.5, esta propiedad será **true**.
- *customError*: es **true** si se ha establecido un error personalizado en el campo utilizando el método *setCustomValidity()*. Esto permite definir mensajes de error específicos que no están cubiertos por las validaciones estándar.
- *valid*: es **true** si el campo es válido, es decir, si no hay errores de validación. Si alguna de las propiedades anteriores es **true**, entonces *valid* será **false**.
- ...

Ejemplo1

Formularios: Validación mediante API JavaScript - Referencias

- https://developer.mozilla.org/es/docs/Learn/Forms/Form_validation
- https://www.w3schools.com/js/js_validation_api.asp

Formularios: Expresiones Regulares

Las expresiones regulares son herramientas que permiten **buscar y manipular texto mediante patrones específicos**. Se utilizan comúnmente para *validar formularios, buscar y reemplazar texto*, y realizar otras operaciones relacionadas con cadenas de caracteres.

En JavaScript, las expresiones regulares se pueden crear de dos maneras:

- **Literal:** Se define entre barras inclinadas /

```
let cadena='Hola mundo';  
let expr=/mundo/;  
expr.test(cadena);           // devuelve true porque en la cadena se encuentra la expresión 'mundo'
```

- **Instanciación:** Usando la clase RegExp, aunque la forma literal es más común y sencilla.

Formularios: Expresiones Regulares - Patrones

Las expresiones regulares utilizan **patrones** que permiten definir qué tipo de texto se busca.

Algunos de los componentes más comunes son:

- ❑ **Corchetes [...]:** Permiten especificar un conjunto de caracteres. Por ejemplo:
 - [abc]: Coincide con 'a', 'b' o 'c'.
 - [^abc]: Coincide con cualquier carácter excepto 'a', 'b' o 'c'.
 - [a-z]: Coincide con cualquier letra minúscula (el carácter '-' indica el rango)
 - [a-zA-Z]: cualquier letra
- ❑ **Pipe |:** Indica una opción entre varias. Por ejemplo:
 - (x|y): Coincide con 'x' o 'y' = [xy]
 - (http|https): cualquiera de las 2 palabras

Formularios: Expresiones Regulares - Patrones

- ❑ **Metacaracteres:** Símbolos que tienen un significado especial en las expresiones regulares:
 - (punto) . : Coincide con cualquier carácter.
 - \d: Coincide con un dígito (0-9).
 - \D: No es un dígito.
 - \s: Coincide con un espacio en blanco. (\S: No es un espacio en blanco)
 - \w: Coincide con una palabra o carácter alfanumérico. (\W: lo contrario)
 - \n: Nueva línea.
- ❑ **Cuantificadores:** Especifican cuántas veces debe aparecer un carácter o grupo de caracteres:
 - +: Al menos una vez. ([0-9]+ al menos un dígito)
 - *: Cero o más veces.
 - ?: Cero o una vez.
 - {n}: Exactamente n veces. ([0-9]{5} = 5 dígitos)
 - {n,}: Al menos n veces.
 - {n,m}: Entre n y m veces.
 - ^: al principio de la cadena (ej.: ^[a-zA-Z] = empieza por letra)
 - \$: al final de la cadena (ej.: [0-9]\$ = que acabe en dígito)

Formularios: Expresiones Regulares - Patrones

- ❑ **Modificadores:** Alteran el comportamiento de la búsqueda:
 - /i: Ignora mayúsculas y minúsculas. (/html/i = buscará html, Html, HTML, ...)
 - /g: Realiza una búsqueda global, encontrando todas las coincidencias.
 - /m: Permite buscar en múltiples líneas.

Formularios: Expresiones Regulares - Métodos

Existen varios métodos para trabajar con expresiones regulares en JavaScript:

- ❑ `expr.test(cadena)`: Devuelve true si la cadena coincide con la expresión regular.
- ❑ `expr.exec(cadena)`: Devuelve un objeto con información sobre la coincidencia encontrada.
- ❑ `cadena.match(expr)`: Aplica la expresión a la cadena y devuelve un array con todas las coincidencias.
- ❑ `cadena.search(expr)`: Devuelve la posición de la primera coincidencia o -1 si no se encuentra.
- ❑ `cadena.replace(expr, cadena2)`: Reemplaza las coincidencias en la cadena con otra cadena.

Ejemplo2

Formularios: Expresiones Regulares - Referencias

- https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_expressions
- https://www.w3schools.com/jsref/jsref_obj_regexp.asp
- <https://www.regular-expressions.info/>
- Probar expresiones: <https://regexr.com/>

Formularios: Ficheros en formularios

Hay infinidad de páginas donde poder consultar cómo validar ficheros e imágenes en un formulario.

Recomendable:

- https://xxjcaxx.github.io/libro_dwec/dom.html#ficheros-en-formularios

Ejemplo3

*Existen múltiples **librerías** que facilitan enormemente el tedioso trabajo de validar un formulario. Un ejemplo es **yup**:*

- <https://www.npmjs.com/package/yup>