educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 2 - Java Web Developer

Módulo 19

Formularios

Validaciones HTML

Aunque se pueden **validar datos en el servidor**, tiene muchos beneficios que cuentes con validación adicional en tu página web. Existen usuarios a los que les resulta molesto llenar un formulario.

Si se validan los datos mientras el usuario lo completa, podría saber inmediatamente si ha cometido algún error; esto le **ahorra tiempo de espera a una respuesta HTTP** y le evita, al servidor, lidiar con entradas incorrectas.

Una de las características de HTML5 es la habilidad de validar la mayoría de datos del usuario **sin depender de scripts.** Esto se hace usando **atributos de validación en elementos de formulario**.

Desde el punto de vista de seguridad, respaldarnos únicamente en esta técnica no es bueno ya que pueden ser anulados por el usuario en cualquier momento, sin siquiera tener conocimientos avanzados en HTML, simplemente editando nuestro HTML desde una consola de desarrollo a la que pueden acceder libremente solo con apretar **F12.**

Es por esto que el **uso del método `checkValidity` desde un Nodo de HTML no es buena práctica a la hora de validar formularios** ya que nos basamos en que los elementos de validación de HTML5 están activos.

Ejemplo

El **checkValidty**, como toda la API de Validaciones de HTML5 es un atajo fácil, por eso no es conveniente a largo plazo:

```
<form id="validation">
   <fieldset>
       <legend>Validation Test</legend>
       <div>
           <label for="name">Name</label>
           <input type="text" name="name" id="name" required>
       </div>
       <button>Enviar</button>
   </fieldset>
</form>
```

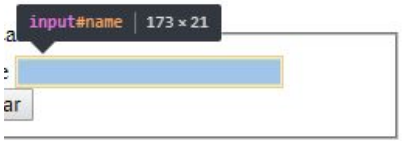
Validaciones HTML - Ejemplo

```
const validator = document.querySelector("#validation")
validator.addEventListener("submit", e => {
        e.preventDefault()
        e.stopPropagation()

        let name = e.target[1]

        console.log(name.checkValidity())
        console.log(name.validity)
})
```

Si el usuario edita el HTML desde la consola de desarrollo:




```
<html lang="en">
▶#shadow-root (open)
  <link type="text/css" id="dark-mode" rel="stylesheet" href
  <style type="text/css" id="dark-mode-custom-style"></style
▶<head>…</head>
▼<body cz-shortcut-listen="true">
    <h2>Validation</h2>
  ▼<form id="validation">
    ▼<fieldset>
        <legend>Validation Test</legend>
      ▼<div>
          <label for="name">Name</label>
          <input type="text" name="name" id="name" required>
        </div>
```

Nuestro **`checkValidity`** podría retornar true haciendo que el usuario pueda enviar cualquier información sin sanitizar.

Por otro lado, podríamos **usar el método `setCustomValidity`** para proporcionar una validación customizada.

Método setCustomValidity

Permite **configurar un mensaje de error customizado** para un elemento del DOM.

```
Element.setCustomValidity(string);
```

Si quisiéramos anular la validación en un caso exitoso deberíamos **declarar un string vacío como valor del parámetro** para indicar al nodo que no tiene errores:

```
Element.setCustomValidity("");
```

Incluso, se pueden usar los **pseudo-selectores de CSS** para declarar estilos customizados, en caso de que nuestros controles sean inválidos:

```
input:valid{
   border : 1px green solid;
}

input:invalid{
   border : 1px solid red;
}
```

Evento Submit

En el DOM, los formularios HTML (HTMLFormElement) tienen propiedades particulares. Un ejemplo de esto es **el evento `submit`, que se dispara al enviar el formulario**.

El evento submit **sirve para ejecutar una función**, después de que el usuario hace clic en el botón *Enviar* y antes de que los datos salgan de la página.

Esta función puede **contener formateo del envío, validaciones, verificaciones, seguimientos**, entre otros.

Para la gestión de este evento tenemos dos partes:

1. **`HTMLFormElement.addEventListener ('submit', ...)`**

   Agrega una función a ser ejecutada en el `submit`.

2. **`HTMLFormElement.submit()`**

   Dispara los `event listeners` registrados para el evento `submit` y, si es posible, luego envía el formulario.

Ejemplo

```
<div>
    <form id="frmLogin">
        <div>
            <label for="username">Usuario:</label>
            <input type="text" name="username" id="username" />
        </div>
        <div>
            <label for="password">Contraseña:</label>
            <input type="password" name="password" id="password" />
        </div>
        <div>
            <button>Enviar</button>
        </div>
    </form>
    <div>
        <button id="enviarDeNuevo">Volver a enviar</button>
    </div>
</div>
```

Continúa el ejemplo

```
20  const form = document.getElementById('frmLogin');
21  const enviarDeNuevo = document.getElementById('enviarDeNuevo');
22
23  form.addEventListener('submit', function()
24  {
25      console.log('Enviando formulario');
26      console.log(form);
27  });
28
29  enviarDeNuevo.addEventListener('click', function()
30  {
31      form.submit();
32  });
33
```

¡Sigamos trabajando!