# Formularios

# Introdución

Neste documente vaise ver como realizar a validación de formularios no lado cliente. Trátase dunha verificación útil porque evita enviar datos incorrectos ao servidor. Sen embargo, isto non pode substituír a validación do lado do servidor, xa que dende o lado cliente se pode manipular maliciosamente o código para saltar as validacións establecidas.

En realidade hai dous tipos de validación do lado cliente:

- Validación propia de HTML. Neste caso é o navegador o que se encarga de todo. Non hai que facer moita programación, mais pola contra non é moi personalizable.
- Validación usando JavaScript. É completamente personalizable mediante código.

A vantaxe da primeira opción é que non hai que escribir código adicional, simplemente engadir os atributos necesarios no HTML. A desvantaxe é que non hai ningún control sobre o proceso de validación, polo que se dan as seguintes desvantaxes:

- cando se envía o formulario, o navegador valida campo a campo e cando encontra un erro, móstrao.
- as mensaxes mostradas son as predeterminadas polo navegador e en ocasións pode non ser claras. Ademais, non hai unha forma estándar de cambiar o CSS das mensaxes de erro.
- as mensaxes móstranse no idioma en que está configurado o navegador, non no idioma da páxina.

# Validación propia de HTML

Hoxe en día é posible validar moitos dos datos dos formularios sen necesidade de JavaScript. Isto é posible usando atributos dos compoñentes dos formularios como:

- required: especifica que hai que cubrir un campo do formulario.
- minlength e maxlength: especifica a lonxitude mínima e máxima do texto.
- min e max: especifica o valor mínimo e máximo numérico.
- type: especifica o tipo de dato contido nun cambo (email, number, …).
- pattern: especifica unha expresión regular que define un patrón a cumprir.

Se os datos do formulario cumpren as regras establecidas polos anteriores atributos, son considerados válidos, en caso contrario serán considerados inválidos.

Cando un elemento dun formulario ten datos considerados **válidos**:

- aplícase a pseudo-clase CSS :valid, que permite aplicar estilos específicos.
- se se envía o formulario, o navegador permíteo.

Cando un elemento dun formulario ten datos considerados **inválidos**:

- aplícase a pseudo-clase CSS :invalid, que permite aplicar estilos específicos.
- o navegador non permite o envío do formulario.

Exemplo:

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

```
<form>
  <label for="choose">Would you prefer a banana or a cherry?</label><br><br>
  <input id="choose" name="i_like" required><br><br>
  <input id="choose2" name="i-like" required pattern="[Bb]anana|[Cc]herry" /><br><br>
  <input type="text" id="choose3" name="i-like" required minlength="6" maxlength="6" />
  <br><br>
  <label for="number">How many would you like?</label>
  <input type="number" id="number" name="amount" value="1" min="1" max="10" />
  <br><br>
  <label for="t2">What's your e-mail address?</label>
  <input type="email" id="t2" name="email"><br><br>
  <button>Submit</button>
</form>
```

# Validación usando JavaScript

No apartado anterior viuse que é posible validar os datos dun formulario usando atributos HTML. Sen embargo, as mensaxes de erro son as que proporciona o navegador, non podendo configuralas. Para poder ter control sobre as mensaxes de erro e facer as configuracións necesarias, hai que facer a validación usando JavaScript.

É posible facer toda a validación de formularios manualmente e personalizar as mensaxes de erros. Como todo o traballo debe ser realizado manualmente, hai que comprobar no evento “submit” se existen erros nalgún campo e en caso afirmativo non debe enviarse o formulario (preventDefault).

Unha alternativa a realizar todo o traballo de forma manual é utilizar un API, como se explica no seguinte apartado.

# API de validación de restricións

A API de validación de restricións consiste nun conxunto de métodos e propiedades dispoñibles nas seguintes interfaces de elementos DOM:

- HTMLButtonElement (representa un elemento <button>)
- HTMLFieldSetElement (representa un elemento <fieldset>)
- HTMLInputElement (representa un elemento <input>)
- HTMLOutputElement (representa un elemento <output>)
- HTMLSelectElement (representa un elemento <select>)
- HTMLTextAreaElement (representa un elemento <textarea>)

A API proporciona as seguinte propiedades nos citados elementos:

- **validationMessage**: devolve a mensaxe describindo as restricións que o elemento non cumpre. Se non hai que validar o elemento (**willValidate** é falso) ou cumpre as restricións (é **valid**), devolve unha cadea baleira.
- **validity**: obxecto ValidityState que contén varias propiedades describindo o estado das restricións que incumpre o elemento. A continuación hai unha lista das propiedades de ValidityState máis utilizadas:
    - patternMismatch: devolve true se o valor non cumpre o patrón e false en caso contrario.
    - tooLong: devolve true se o valor é máis grande que o valor especificado no atributo maxlength ou false en caso contrario.
    - tooShort: devolve true se o valor é máis pequeno que o valor especificado no atributo minlength ou false en caso contrario.
    - rangeOverflow: devolve true se o valor é máis grande que o valor especificado no atributo max ou false en caso contrario.
    - rangeUnderflow: devolve true se o valor é menor que o valor especificado no atributo min ou falso en caso contrario.
    - typeMismatch: devolve true se o valor non coincide co tipo de dato indicado no atributo type ou false en caso contrario.
    - valid: devolve true se o elemento cumpre todas as restricións e é considerado válido ou false en caso contrario.
    - valueMissing: devolve true se o elemento ten o atributo required e non se lle asignou ningún valor. En caso contrario devolve false.
- **willValidate**: de volve true se hai que validar o elemento cando se envíe o formulario e se devolve falso non haberá que validalo.

Os atributos anteriores correspóndense con diferentes restricións. En función de se a restrición se cumpre ou non, aplicaranse ao elemento as clases CSS :valid, :invalid, :out-of-range, etc.

Ademais das propiedades anteriores, a API de validación de restricións proporciona os seguintes métodos no formulario e nos seus elementos:

- **checkValidity()**: devolve true se o valor elemento é válido e false en caso contrario. Se o elemento é invalido, este método lanza o invalid event no elemento.
- **reportValidity()**: comproba se o elemento cumpre as restricións de validación. Se non se cumpren, lanza o invalid event e envía o validationMessage ao navegador para que sexa mostrado á persoa usuaria.
- **setCustomValidity(message)**: engade unha mensaxe de erro personalizada ao elemento. Se se establece unha mensaxe de erro personalizada, o elemento é considerado inválido e o erro especificado é mostrado. Para eliminar este erro hai que usar setCustomValidity(‘’).

Exemplo:

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

  input:valid {
    border: 2px solid black;
  }
```

```
<form>
  <label for="mail">I would like you to provide me with an e-mail address:</label>
  <input type="email" id="mail" name="mail">
  <button>Submit</button>
</form>
```

```
const email = document.getElementById("mail");

email.addEventListener("input", function (event) {
  if (email.validity.typeMismatch) {
    email.setCustomValidity("I am expecting an e-mail address!");
    // a seguinte liña permite mostrar a mensaxe de erro.
    // Se está comentada a mensaxe só se mostrará ao enviar o formulario.
    email.reportValidity();
  } else {
    email.setCustomValidity("");
  }
});
```

O evento anterior execútase cada vez que se modifica o valor do campo input. No manexador do evento compróbase se o correo é válido e en caso de que sexa incorrecto establécese a mensaxe personalizada de erro usando **setCustomValidity**. Se o correo é válido, establécese a mensaxe de erro a “”.

Ligazón a un exemplo máis detallado e a explicación detallada.

# Compoñentes DOM

A interface HTMLFormElement representa un elemento <form> no DOM que permite acceder e, nalgúns casos modificar, aspectos do formulario así como acceder aos seus elementos.

Para obter o HTMLFormElement pode usarse o método querySelector(), ou obter a lista con todos os formularios usando Document.forms.

Document.forms devolve unha colección cos formularios do documento. Pode accederse aos formularios individuais dalgunha das seguintes maneiras:

- document.forms[index]: devolve o formulario especificado no index.
- document.forms[id]: devolve o formulario co id especificado.
- document.forms[name]: devolve o formulario co valor especificado no atributo *name*.

Para acceder aos elementos do formulario pode usarse a propiedade elements, que devolve unha HTMLFormControlsCollection cos **controis** do formulario. Esta lista só conterá elementos dos seguintes tipos: <button>, <fieldset>, <input>, <object>, <output>, <select>, <textarea>.

Para acceder a un control particular usarase o índice ou o valor do atributo **name** ou **id**.

```
const controis = document.getElementById("my-form").elements;
const inputByIndex = controis[0];
const inputByName = controis["nameValue"];
const inputById = controis.idValue;
```

Se hai varios controis co mesmo valor no atributo name, controis[“nameValue”] devolverá unha colección de elementos.

Tamén é posible acceder ao **formulario** (ancestro) dende calquera dos seus controis usando o atributo **form.**

```
const inputByIndex = controis[0];
cont form = inputByIndex.form;
```

## Elementos do formulario

Os elementos do formulario teñen dispoñibles un conxunto de atributos para acceder á información que almacenan.

Así, para acceder aos valores almacenados nos elementos **input** e **textarea** poden usarse as propiedades **input.value** ou **input.checked**.

**NOTA:** non usar textarea.innerHTML para acceder ao valor dun área de texto, xa que innerHTML garda o HTML que había inicialmente na páxina, non o valor actual.

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname" value="Nome" /><br />
<textarea name="areaTexto" id="textArea" cols="30" rows="10">Área de texto
</textarea>
<input type="checkbox" name="vehicle3" value="Boat" id="checkbox" checked />
<label for="vehicle3"> I have a boat</label><br /><br />
```

```
console.log(document.getElementById("fname").value);
console.log(document.getElementById("textArea").value);
console.log(document.getElementById("checkbox").value);
console.log(document.getElementById("checkbox").checked);
```

Un elemento <select> ten 3 propiedades importantes:

- **select.options**: colección de subelementos de <option>
- **select.value**: valor seleccionado actualmente.
- **select.selectedIndex**: o número de opción seleccionada actualmente.

```
<select name="select" id="seleccion">
  <option value="value1">Value 1</option>
  <option value="value2" selected>Value 2</option>
  <option value="value3">Value 3</option>
</select>
```

```
let seleccion = document.getElementById("seleccion");
console.log(seleccion.options);
console.log(seleccion.value);
console.log(seleccion.selectedIndex);

// Seleccionar un valor mediante JavaScript
seleccion.value = "value1";
seleccion.selectedIndex = 0;
seleccion.options[0].selected = true;
```

Os formularios e os elementos que o compoñen poden xerar unha serie de eventos:

- **focus**: prodúcese cando se enfoca un compoñente.
- **blur**: prodúcese cando o compoñente perde o foco.
- **change**: prodúcese cando se remata de cambiar un compoñente. Nun compoñente de texto sucede cando o elemento perde o foco. Nos elementos select e checkbox sucede inmediatamente ao cambiar a opción seleccionada.
- **input**: prodúcese cada vez que o valor dun elemento é modificado. Nun campo de texto sucede cada vez que este se modifica. A diferencia dos eventos de teclado, este evento lánzase ao copiar e pegar co rato no elemento.

- **cut**, **copy** e **paste**: suceden ao cortar, copiar e pegar un valor.
- **submit**: prodúcese cando se envía o formulario. Un formulario pode enviarse facendo clic no campo “submit” ou pulsando **Enter** nun campo do formulario.

**Exercicios:**

1. Crea unha páxina web que conteña un elemento <select> con dúas opcións.

   Utiliza JavaScript para:

   - Mostrar por consola o valor seleccionado
   - Engadir unha nova opción.

2. Crea unha páxina web que conteña un campo onde haxa que escribir un número co seguinte formato DDD-DD-DDD. O número estará formado por grupos de tres, dúas e tres cifras separadas, de forma opcional, por un guión (-). É dicir, os seguintes valores deben ser válidos: 123-45-678 e 12345678.

   Fai as tres versións de validación do formulario: só usar HTML, JavaScript manual e API de validación de restricións.

# Referencias

Para a elaboración deste material utilizáronse, entre outros, os recursos que se enumeran a continuación:


- JavaScript | MDN
- Client-side web APIs - Learn web development | MDN
- JavaScript Tutorial - w3schools
- Desarrollo Web en Entorno Cliente | materials
- Javascript en español - Lenguaje JS
- The Modern JavaScript Tutorial
- Eloquent JavaScript
- Client-side form validation - Learn web development | MDN