**Conceptos de validación en formularios**

Al momento de crear formularios, podemos ingresar distintos tipos de datos, algunos son más importantes que otros, pero unos de ellos deben de ser obligatorios para que el formulario tenga sentido.

Para hacer obligatorio un campo ***input*** podemos declarar la propiedad ***required*** dentro del input en HTML que queremos que sea obligatorio:

<input name="name" id="name" class="input" type="text" placeholder="Nombre" required>

Para hacer obligatorios campos de correo electrónico, y que no sea posible escribir cualquier texto, sino obligatoriamente un correo electrónico, podemos ingresar el tipo de input que es ***type=”email”***

<input name="email" id="email" class="input" type="email" placeholder="Email" required>

Dentro del apartado de las contraseñas podemos hacer que no sean visibles agregando el **type=”password”** y no solo eso, también podemos declarar el mínimo y máximo de caracteres, con ***minlength=”numero de caracteres” y maxlength=”numero de caracteres”***.

<input name="password" id="password" class="input" type="password" placeholder="Contraseña" required minlength="6" maxlength="12">

El atributo ***type=”date”*** nos permite validar que la información digitada esta en el formato de fecha correcto.

**Expresión regular:**

Es un estilo de nomenclatura que nos ayuda a generar patrones para aplicar condiciones en nuestro código.

La que veremos a continuación es una expresión regular para contraseñas:

^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?!.\*[ !@#$%^&\*\_=+-]).{6,12}$

Para declarar una expresión regular, primero se empieza con un acento ***^*** y termina con un ***$***, dentro de estos signos estaremos declarando las condiciones que queremos que se cumplan. En el ejemplo son un total de 4.

Cuando digitamos “***?***=” estamos diciendo que aceptamos caracteres, luego el punto “**.”** Que quiere decir siempre y cuando no sean enters ni saltos de línea. Luego viene el asterisco, con el cual decimos que tiene que venir al menos una vez el carácter. Los caracteres dentro de los corchetes vienen las condiciones del tipo de carácter, el primero son letras minúsculas, el otro son letras mayúsculas, el siguiente números y por ultimo, declarado con los signos “**?!”** quiere decir que nosotros no vamos aceptar, “**.**” Saltos de línea “**\*”** o almenos, [ !@#$%^&\*\_=+-] un espacio, un signo de interrogación, etc. Y por ultimo declaramos que esas condiciones deben cumplirse al menos “**.”** De 6 a 12 veces, entre llaves. Lo que queremos decir aquí, es declarar el mínimo y máximo de caracteres.

**Para declarar esta expresión regular en una contraseña,** dentro de la etiqueta declaramos la propiedad ***pattern= “la expresión regular”***

**Ahora para avisar al usuario que condiciones son obligatorias para la contraseña**, podemos agregar la propiedad ***“tittle”*** en la cual podemos escribir un mensaje que vera el usuario y podrá tener claridad acerca de las condiciones para la contraseña.

**Para los campos donde necesitamos fechas** usamos el tipo “***date***” en nuestro código html y saldrá un campo con un recuadro para poder digitar las fechas con el formato correcto.

**Declaración de *data-set***: Para declarar un data set, primero debemos localizar la etiqueta, luego agregar ***data-nombre-nombre*** o ***data-nombre=”nombre”***

De la siguiente manera:

data-archivo-formulario

o

data-tipo="nacimiento"

Ahora podremos llamar estos ***datas*** dentro de nuestro archivo JavaScript, para ello agregamos la propiedad ***.dataset*** para adquirir la colección de todos los datas, posterior a ello, agregamos la propiedad ***.nombredeldata*** es decir el data que esta después del guión “***data-nombre”***

    const tipoDeInput = input.dataset.tipo

**Conceptos en JavaScript**

***New*** cuando declaramos esta palabra lo que queremos decir es que estamos creando una nueva instancia, la cual es un objeto nuevo e independiente el cual puede ser modificado según las plantilla de la clase o constructo en el que fue declarado.

**Desde JS podemos agregar un mensaje de validación,** como el ***“tittle”*** de HTML, la propiedad es :

***.setCustomValidity(mensaje);*** el mensaje puede ser una variable o texto.

**Funciones de fecha:** Con las siguientes funciones podemos obtener datos específicos de la fecha:

        fecha.getUTCFullYear(), Devuelve el año.

        fecha.getUTCMonth(), Devuelve el mes, el 0 es enero y 11 diciembre

        fecha.getUTCDate(), Devuelve el día del mes 1 al 31

Estos valores están en UTC (Tiempo Universal Coordinado).

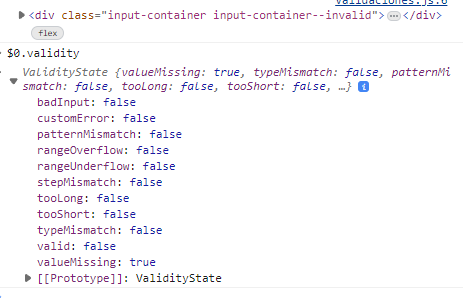
const inputs = document.querySelectorAll("input");

Sirve para seleccionar todos los tipos de datos, según sean los declarados dentro del paréntesis.

inputs.forEach();

Significa para cada uno.

Dentro de la consola de nuestro compilador, podemos obtener una información, acerca de nuestras validaciones en nuestro input. Dicho objeto se llama ***validity*** y se accede, primero seleccionando el input y despues con el comando: ***$0.validity***



El valor de valid, es false, por lo tanto no es valido. Y podemos ajustar ciertos cambios desde JS con estos valores, por ejemplo, si **valid** es false, podemos agregar una clase en la cual tengamos un estilo de “campo no valido”. En cambio es si es true, podemos quitar esta clase y dejar el campo tal y como esta.

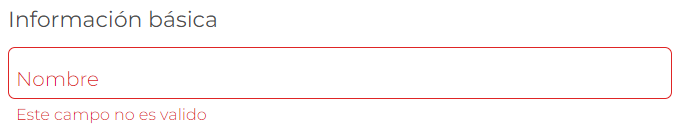
    if(input.validity.valid){

        input.parentElement.classList.remove("input-container--invalid");

    }else{

        input.parentElement.classList.add("input-container--invalid");

    }



Si no es válido quedara con este estilo, de lo contrario, quedara normal.

const mensajeDeErro= {

    nombre: {

        valueMissing: "Este campo no puede estar vacío",

    },

    email: {

        valueMissing: "Este campo no puede estar vacío",

        typeMismatch: "Este campo no puede estar vacío",

    },

    password: {

        valueMissing: "Este campo no puede estar vacío",

        patternMismatch: "La contraseña debe contener al menos 6 caracteres y máximo 12, pueden ser mayúscual o minúscula, debe contener un número, y un carácter especial.",

    },

    nacimiento: {

        valueMissing: "Este campo no puede estar vacío",

        customError: "Debes tener al menos 18 años de edad."

    }

}

El ***valueMissing*** es una propiedad que esta esperando un valor, si lo esta buscando será true, si ya lo encontró será false.

El ***typeMismatch*** nos permite encontrar el mismo tipo de input que se esta esperando, por ejemplo un correo electrónico, el cual tiene una estructura definida. Este parámetro nos ayuda a validarlo.

El ***patternMismatch*** Sirve por ejemplo, para validar un patrón con expresión regular que hemos definido, por ejemplo el patrón que queremos en una contraseña.

El ***customError*** como su nombre lo dice, nos sirve para validar un error que nosotros hemos definido.