**Ingreso a ALURA**

<https://app.aluracursos.com/dashboard>

<https://app.aluracursos.com/loginForm?urlAfterLogin=https://app.aluracursos.com/dashboard>

**Compilador de javaScript tener en cuenta**

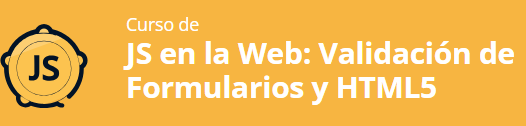
[**https://www.programiz.com/javascript/online-compiler/**](https://www.programiz.com/javascript/online-compiler/)

Compilador de Python

<https://www.programiz.com/python-programming/online-compiler/>

Curso de

JS en la Web: Validación de Formularios y HTML5



**Presentación**

[00:00] Hola, mi nombre es Harland Lohora y soy instructor aquí en Alura cursos. Durante este curso vas a aprender cómo trabajar con formularios en HTML, cómo poder agregar clases y quitarlas. Por ejemplo, en este caso yo tengo aquí mi formulario. Si yo me posiciono dentro de mi input de nombre, le doy clic y salgo, me comenta que el campo no puede estar vacío.

[00:23] Yo lo que voy a tener que hacer es voy a tener que poner mi nombre. Lo mismo va a pasar con mi correo electrónico. En este caso pues bueno, voy a poner aquí uno, y así va a seguir sucesivamente con todos los inputs que tengo aquí en mi formulario.

[00:41] Vamos a aprender cómo poder trabajar con las validaciones, cómo trabajar o cómo sería una aplicación normal de expresiones regulares, por lo cual te recomiendo que tengas al menos un conocimiento base, no tienes que ser un experto, pero para que puedas entender bien el curso, te lo recomiendo.

[01:00] Si no, de cualquier manera puedes hacerlo. Yo hago la explicación de más o menos qué es lo que hace la expresión regular que nosotros utilizamos dentro del input y al final del curso no solamente vas a tener este proyecto en tu computadora sino que también lo vas a tener en una URL que tú vas a poder compartir esta URL con tus amigos o poderlo poner en tu portafolio de proyectos o inclusive poderlo compartir en LinkedIn.

[01:27] Entonces espero que te agrade el curso, espero que tengas interés y si lo tienes, nos vemos en el curso.

**Preparando el ambiente**

Para empezar con este curso es necesario hacer la descarga de los archivos HTML y CSS, además de algunos íconos utilizados en esta clase.

Te dejo el enlace para descarga: Proyecto inicial.

**Atributos de validación**

[00:00] Muy bien. Entonces vamos a comenzar y lo primero que hice o lo primero que tienes que hacer es descargar el proyecto en tu computadora, ya sea a través de GitHub o bien descargando el punto zip, lo descomprimes o lo clonas el repositorio y deberías de tener acceso ya a estos documentos.

[00:18] Muy bien. Entonces a simple vista, pues bueno, es un simple código HTML, donde al final de cuenta, si nosotros abrimos en nuestro navegador, vamos a tener esta vista, esta de aquí. Entonces, todo este HTML nos ayuda a generar este formulario, lo que vamos a trabajar entonces es con que se realice el registro pero de una manera en la cual nosotros tengamos el control de qué cosas sí pueden ingresar los usuarios y qué cosas no.

[00:54] Por ejemplo, si yo en este momento le doy clic en registrar, lo que hizo fue redireccionarme a una vista en el cual dice registro completado, pero realmente nunca rellené nada de esta información como el nombre, el correo electrónico, contraseña, etcétera. ¿Entonces, qué es lo que necesitamos? Pues bueno, lo primero con lo que vamos a trabajar es con el input de nombre.

[01:20] Este input de nombre, lo que yo necesito es que el usuario escriba al menos su nombre. Ahora lo que nosotros necesitamos decirle a nuestro código es que ese campo es requerido. No podemos registrar un usuario o un cliente si no tiene un nombre. ¿Y cómo hacemos eso?

[01:43] Bueno. Aquí, en la línea 27 yo tengo el input. Y lo que necesitaría agregar sería una nueva propiedad de los elementos input, el cual es llamado required. Aquí está. Voy a guardar, voy a regresar a mi navegador, le voy a dar clic en registrar y ahora, ¿qué es lo que comenta? Bueno, ya no hizo la redirección automáticamente y más bien lo que dice es completa este campo.

[02:11] Si tú tienes tu navegador en otro idioma, posiblemente te aparezca en el idioma con el cual lo instalaste. En este caso, pues bueno, es español. Muy bien, entonces, dice: completa este campo. Okay, yo le voy a poner entonces un nombre Harland. Ahora tenemos la parte de email aquí, ¿cuál sería la estructura que nosotros necesitamos?

[02:35] Le podemos poner también si nosotros quisiéramos qué es requerido. Aquí en el input de la línea 32, le voy a decir que es required. Voy a rellenar de nuevo aquí con Harland, y mi correo le voy a poner simplemente harland. Le voy a dar clic en registrar, de nuevo me llevó a la vista como si el registro ya se hubiera completado.

[02:57] Aunque nosotros sabemos que esta estructura, solamente harland, no hace referencia a un correo electrónico para eso los navegadores ya saben de qué manera tratarlo, saben que primero tiene que venir una serie de caracteres, después un @, después otra serie de caracteres y por último.com, .mx, punto lo que sea, pero la estructura es esa.

[03:21] ¿Entonces nosotros, como le decimos a este input que no va a ser texto normal, sino que va a ser un input de tipo correo electrónico? Muy bien. Lo que vamos a hacer entonces aquí tenemos type. Aquí este type le voy a decir que va a ser de tipo email. Voy a guardar. Voy a regresar de nuevo aquí a mi navegador, voy a volver a llenar con la información.

[03:48] Aquí le voy a poner simplemente harland, lo voy a dar clic en registrar y dice: incluye un signo @ en la dirección de correo electrónico. La dirección “harland”, que es lo que yo puse, no incluye ese signo. ¿Entonces qué pasa si yo le pongo ahora solo el @? Le voy a dar clic en registrar y me dice: ingresa texto después del signo @.

[04:08] Okay, entonces está incompleto nuestro correo electrónico. Entonces bueno, lo voy a poner aquí a Alura.com Aluracursos.com. Le voy a dar clic en registrar y ahora sí, ¿qué es lo que obtenemos? Pues bueno, nos está mandando a nuestra visa de registro. Vamos a ir entonces con nuestro siguiente input, que en este caso va a ser el de la contraseña.

[04:33] Y bueno, así como nosotros tenemos nuestro tipo email, también existe un tipo contraseña, o en inglés password, entonces nosotros aquí en la línea 37 le vamos a decir “el tipo de este input va a ser password”. Vamos a poner que también es requerido. Voy a guardar, vamos a regresar a nuestro navegador. De nuevo voy a poner harland.

[05:01] harland@aluracursos.com. Y la contraseña ahora vamos a ver. Bueno, voy a poner aquí una contraseña. Y si te puedes dar cuenta ya hay una diferencia. Al momento que yo le dije type password, lo que va a hacer automáticamente es, en lugar de mostrarme lo que yo escribí, rellenarlo con puntos. De esta manera no queda visible la contraseña.

[05:27] Pero aún así. Nosotros pusimos que iba a ser requerido la contraseña, hasta ahí está funcionando. ¿Pero qué pasa por ejemplo, si yo solo le pongo un carácter y le doy clic en registrar? Para el formulario, la contraseña pues fue válida, o sea, sí se llenó con información, pero nosotros nunca le mencionamos cuál era el formato que nosotros necesitábamos o tal vez un mínimo de caracteres.

[05:53] Entonces nosotros tenemos también aquí, en nuestra parte de nuestro proyecto, en la línea 37 vamos a poder pasarle otra propiedad al input, que es minlength, es decir, una longitud mínima que necesita tener este input de caracteres para la cual va a funcionar como si estuviera completado.

[06:19] Por ejemplo, si yo le pongo aquí 6, es decir, aquí el input espera que, al menos se ingresen 6 caracteres en el campo. Y así como tenemos un minlength, también podemos tener un maxlength. Entonces vamos a decirle la longitud máxima que nosotros estamos requiriendo. En este caso podría ponerle tal vez 12.

[06:46] Voy a guardar. Voy a regresar a mi proyecto. Recuerda que se refresca con Live Server, aquí, con esta extensión. En caso de que no la tengas, puedes abrir el archivo y simplemente vas recargando cada vez que vayas haciendo algún cambio.

[07:04] Muy bien, entonces voy a rellenar aquí de nuevo con mi nombre Harland, de nuevo mi correo harland@aluracursos.com, la contraseña voy a poner 5 caracteres, clic en registrar, entonces aquí yo ya puse más caracteres. Si yo quiero seguir escribiendo ya no me deja el input. ¿Por qué?

[07:25] Porque yo le especifiqué que la longitud máxima de este input era de 12 caracteres. Entonces bien, hasta aquí está funcionando. Le voy a dar clic de nuevo en registrar. Está funcionando como nosotros esperamos, no está registrando realmente la información, pero al menos está llevando esas validaciones de cuáles campos son requeridos, cuáles campos son de tipo email, cuáles campos son de tipo contraseña.

[07:52] Y a pesar de que nosotros ya logramos decir un parámetro que es la longitud mínima y máxima para nuestra contraseña, ¿realmente qué pasaría si nosotros necesitamos al menos un carácter especial, al menos una mayúscula, al menos un número?

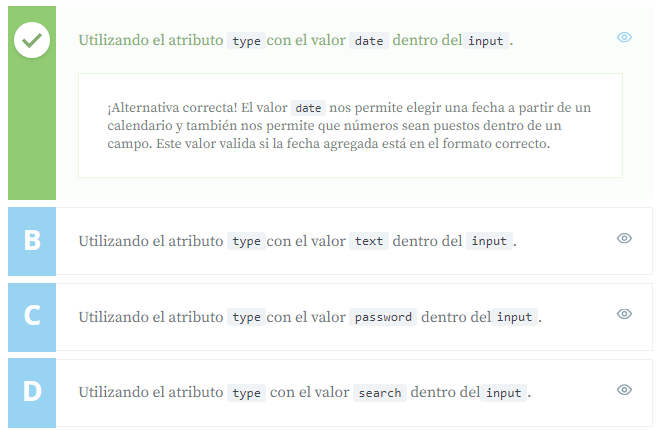
[08:11] Entonces, lo que nosotros vamos a ver en la siguiente clase, es justamente cómo trabajar ese formato o esas validaciones de una manera apropiada. Entonces te veo en la siguiente clase.

**Atributo type**

HTML nos regala algunas herramientas para la validación de formularios. Una de ellas es el atributo type. Este atributo hace validaciones, por ejemplo, type=email, que verifica si la estructura que fue escrita está en un formato válido.

¿De las alternativas abajo, cuál de ellas presenta un escenario en el que el uso del atributo type hace una validación?

Seleccione una alternativa



**Haga lo que hicimos en aula**

Practicar ayuda mucho en el aprendizaje de un nuevo concepto. Así siendo, es muy importante que implementes lo que fue presentado en esta clase.

**VER OPINIÓN DEL INSTRUCTOR**

Opinión del instructor

No dejes de quitar tus dudas antes de dar continuidad al curso. En el caso de que tengas alguna duda esa podrá ser contestada por nuestra comunidad en el foro de Alura Latam.

**Lo que aprendimos**

Lo que aprendimos en esta aula:

Hacer validación en el propio HTML utilizando los atributos required y type.

Utilizar reglas del regex dentro del atributo pattern para validar el campo de contraseña.

Mandar un mensaje customizado de error en el navegador

**Proyecto del aula anterior**

¿Comenzando en esta etapa? Aquí puedes descargar los archivos del proyecto que hemos avanzado hasta el aula anterior.

Descargue los archivos en Github o haga clic aquí para descargarlos directamente..

**Regex para contraseña**

[00:00] Muy bien, entonces a como vimos en la clase anterior, ya logramos poner como requerido el input de nombre, el input de correo electrónico, definiendo que va a ser de tipo correo electrónico y también la contraseña, y solamente formateamos cuál iba a ser la longitud mínima y la longitud máxima.

[00:19] Pero puede que nuestra regla de negocio o en lo que nosotros queremos, tenga otras propiedades, por ejemplo, al menos un número, al menos una letra en minúscula, al menos una letra en mayúscula.

[00:35] Entonces, si nosotros quisiéramos pasarle propiedades como lo hicimos aquí, de decirle cuál iba a ser la longitud mínima y la longitud máxima, realmente no sería tan conveniente. Lo que nosotros en cambio vamos a poder utilizar es algo llamado the pattern, que sería patrón.

[00:53] Voy a eliminar estos dos de aquí, voy a escribir aquí pattern. Este pattern lo que va a recibir es una expresión regular. Entonces, en caso de que no hayas visto o no sepas muy bien qué es una expresión regular, aquí en Alura tenemos un curso igual impartido por mí o si no igual puedes simplemente venir al navegador, poner regex contraseña, regex es password. Vamos a ver aquí, por ejemplo, de aquí regexlib.

[01:36] Aquí bueno, ya tienen diferentes opciones o expresiones que han desarrollado otras personas, en las cuales o te puedes basar o simplemente puedes copiarlos y pegarlos. Esta sería una opción. También está el sitio más popular creo que de todos los programadores que es esta Cover Flow, y pues igual aquí, en este caso, pues puedes encontrar cuál sería la respuesta.

[02:01] Entonces aquí como comenta, pues bueno, al menos 8 caracteres y al menos una letra y un número, entonces, toda esta expresión regular genera ese patrón que nosotros buscamos dentro de nuestro input de contraseña. Pero bueno, yo no voy a tomar ninguno de estos, yo ya tengo aquí uno definido, que es esta expresión regular de aquí. Y vamos a pegarlo también aquí abajo para ver de qué se trata.

[02:40] Lo que nosotros tenemos es estos símbolos. Uno, que es como un acento, lo voy a hacer un poco más grande este acento y cierra con el símbolo de peso. Lo que prácticamente le estoy diciendo aquí es que quiero que evalúe todo nuestro texto con esta lógica.

[03:04] Después, lo que tenemos aquí son diferentes conjuntos de reglas que queremos que se apliquen a este texto. En este caso son cuatro. El primero es ?=. Lo que estamos diciendo prácticamente es que vamos a aceptar caracteres siempre y cuando, cuando el punto, este punto lo que hace referencia es que no sean saltos de línea o enters.

[03:31] Entonces lo que estamos diciendo, vamos a aceptar dígitos o números o cualquier carácter realmente. Y ahora con \* lo que le estamos diciendo es que tiene que venir al menos una vez y este al menos una vez, pues bueno, va a ser sobre este conjunto, es decir, desde la A minúscula hasta la Z minúscula.

[03:49] Ahora, si te das cuenta realmente tanto el primero, el segundo y el tercer conjunto de reglas, pues es prácticamente lo mismo. Lo único que va a cambiar es qué es lo que nosotros estamos aceptando. En el primero de la A minúscula a la Z minúscula, el segundo, de la A mayúscula, a la Z Mayúscula, el tercero del 0 al 9 para poder aceptar números.

[04:15] Y el último que nosotros tenemos aquí, lo que le estamos diciendo es prácticamente nosotros no vamos a aceptar, igual de nuevo, saltos de línea o al menos un espacio en blanco, un símbolo de admiración, @, #, gato o almohadilla, como le digan, símbolo de peso, porcentaje y así sucesivamente.

[04:40] Y por último, lo que le estamos definiendo aquí es que queremos que se cumplan esta serie de reglas, y tiene que ser entre 6 a 12 veces. Es de esta manera en la cual nosotros le estamos definiendo que nuestra contraseña tiene que tener al menos 6 caracteres y máximo, 12 caracteres.

[04:59] Muy bien, voy a eliminar entonces esto de aquí. Voy a regresar de nuevo aquí a mi input. Si recuerdas, pues bueno, simplemente lo que hicimos fue poner entre comillas la expresión regular y le estamos asignando a pattern. Lo voy a guardar, voy a regresar de nuevo a mi navegador. Voy a cerrar esta de aquí, voy a poner mi nombre.

[05:24] harland@aluracursos.com y la contraseña pues bueno, la voy a poner ahorita solo 5 caracteres. Le voy a dar clic en registrar. Y ahora me comenta: Haz coincidir el formato solicitado. Yo sé que se necesitan al menos 6 caracteres. Voy a ver qué pasa si sigo escribiendo 6 caracteres, le voy a dar clic en registrar, en este caso me dejó escribir, ya no me puso un máximo de caracteres como tal, pero pues de nuevo me está comentando que necesito coincidir el formato solicitado.

[06:00] Entonces nosotros sabemos, porque lo estamos programando, cuál es ese formato que nosotros queremos en ese input, pero necesitamos decirle al usuario cuál es ese formato. Entonces para poder llegar a eso, lo que voy a hacer aquí es igual dentro del mismo input de la contraseña, lo que voy a escribir entonces es algo llamado title.

[06:25] En este title voy a decir un texto que le explique al usuario cuál es este formato. Le voy a poner al menos 6 caracteres, un máximo 12, debe contener una letra minúscula, una letra mayúscula, un número y no puede contener caracteres especiales. Muy bien, entonces voy a guardar.

[07:13] Le voy a poner un punto final, voy a guardar. Voy a regresar aquí a mi proyecto, voy a poner entonces mi nombre, mi correo electrónico, la contraseña y ahora, como puedes darte cuenta, yo escribí solo 5 caracteres y ya ahora sí me está dando un texto explicativo de cuál es el formato o el patrón que nosotros estamos esperando.

[07:37] Entonces es: Al menos 6 caracteres máximo 12, debe contener una letra minúscula, una letra mayúscula, un número y no puede contener caracteres especiales. Entonces bueno, solo para recapitular, si nosotros quisiéramos justamente tener ciertas reglas respecto a la información que ingrese el usuario, nosotros vamos a utilizar una expresión regular y la vamos a asignar aquí a nuestro pattern.

[08:06] Al momento de hacer eso, nosotros ya no vamos a tener un texto, por así decirlo que le diga al usuario qué es lo que le está faltando o cuál es el formato esperado y para eso vamos a utilizar la propiedad title. Muy bien, entonces vamos a dejar hasta aquí el video y te veo en la siguiente clase.

Archivo registro.html

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Doguito Petshop | Crear cuenta</title>

    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400&display=swap" rel="stylesheet">

    <link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="./assets/css/base/base.css">

    <link rel="stylesheet" href="./assets/css/registro.css">

    <link rel="stylesheet" href="./assets/css/componentes/card.css">

    <link rel="stylesheet" href="./assets/css/componentes/inputs.css">

    <link rel="stylesheet" href="./assets/css/componentes/button.css">

</head>

<body>

    <main class="container flex flex--column flex--center">

        <div class="cadastro-cabecalho">

            <img src="./assets/img/doguito.svg" alt="Logo Doguito" class="registry-header\_\_logo">

            <h1 class="registry-header\_\_title">PetShop</h1>

        </div>

        <section class="cartao">

            <h2 class="card\_\_title">Completa tu registro</h2>

            <form action="./registro\_concluido.html" class="formulario flex flex--column">

                <fieldset>

                    <legend class="form\_\_label">Información básica</legend>

                    <div class="input-container">

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

                        <label class="input-label" for="name">Nombre</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                    <div class="input-container">

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

                        <label class="input-label" for="email">Email</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                    <div class="input-container">

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

                        <label class="input-label" for="password">Contraseña</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                </fieldset>

                <fieldset>

                    <legend class="form\_\_label">Información personal</legend>

                    <div class="input-container">

                        <input name="birth" id="birth" class="input" type="text" placeholder="Fecha de nacimento">

                        <label class="input-label" for="birth">Fecha de nacimiento</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                    <div class="input-container">

                        <input name="phoneNumber" id="phoneNumber" class="input" type="text" placeholder="Número telefónico">

                        <label class="input-label" for="phoneNumber">Número telefónico</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                </fieldset>

                <fieldset>

                    <legend class="form\_\_label">Dirección</legend>

                    <div class="input-container">

                        <input name="cep" id="cep" class="input" type="text" placeholder="CEP">

                        <label class="input-label" for="cep">Dirección completa</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                    <div class="input-container">

                        <input name="city" id="city" class="input" type="text" placeholder="Ciudad">

                        <label class="input-label" for="city">Ciudad</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                    <div class="input-container">

                        <input name="state" id="state" class="input" type="text" placeholder="Estado">

                        <label class="input-label" for="state">Estado</label>

                        <span class="input-message-error">Este campo no es valido</span>

                    </div>

                </fieldset>

                <button class="button">Registrar</a>

            </form>

        </section>

    </main>

</body>

</html>

<https://regexlib.com/Search.aspx?k=password&AspxAutoDetectCookieSupport=1>

<https://stackoverflow.com/questions/19605150/regex-for-password-must-contain-at-least-eight-characters-at-least-one-number-a>

<div class="input-container">

                        <input name="password" id="password" class="input" type="password" placeholder="Contraseña" required pattern = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?!.\*[ !@#$%^&\*\_=+\-]).{6,12}$" title="Al menos seis caracteres maximo 12 , debe contener una letra minuscula , 1 letra mayuscula. 1 numero , no puede contener caracteres especiales">

                        <label class="input-label" for="password">Contraseña</label>

                        <span class="input-message-error">Este campo no es valido</span>

<div class="input-container">

<input name="password" id="password" class="input" type="password" placeholder="Contraseña" required pattern = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?!.\*[ !@#$%^&\*\_=+\-]).{6,12}$" title="Al menos seis caracteres maximo 12 , debe contener una letra minuscula , 1 letra mayuscula. 1 numero , no puede contener caracteres especiales">

<label class="input-label" for="password">Contraseña</label>

<span class="input-message-error">Este campo no es valido</span>

**Haga lo que hicimos en aula**

Practicar ayuda mucho en el aprendizaje de un nuevo concepto. Así siendo, es muy importante que implementes lo que fue presentado en esta clase.

**VER OPINIÓN DEL INSTRUCTOR**

Opinión del instructor

No dejes de quitar tus dudas antes de dar continuidad al curso. En el caso de que tengas alguna duda esa podrá ser contestada por nuestra comunidad en el foro de Alura Latam.

**Lo que aprendimos**

Lo que aprendimos en esta aula:

Crear funciones para validaciones del formulario.

Definir mensajes customizados para validaciones fuera del HTML.

Utilizar data attributes para trabajar con JavaScript.

Dejar la función de validación más genérica para funcionar con cualquier input

**Proyecto del aula anterior**

¿Comenzando en esta etapa? Aquí puedes descargar los archivos del proyecto que hemos avanzado hasta el aula anterior.

Descargue los archivos en Github o haga clic aquí para descargarlos directamente.

**Fecha de nacimiento**

[00:00] Bueno, ya conseguimos entonces definir que el nombre es requerido, el correo electrónico también es requerido y tiene que tener al menos el @. También ya insertamos a un pattern o un patrón que nosotros queremos en el campo de la contraseña. Vamos a continuar entonces con el input de fecha de nacimiento.

[00:21] Ahorita como está actualmente yo realmente puedo escribir lo que quiera, si te das cuenta. Esto no es un formato válido para una fecha de nacimiento. Entonces, lo primero que voy a hacer es voy a venir a mi formulario y lo que voy a hacer entonces aquí en fecha de nacimiento tenemos aquí el input.

[00:43] Lo que voy a hacer es escribirle que es requerido y el tipo ahora lo voy a cambiar para que sea de tipo date. Voy a guardar, voy a regresar a mi proyecto, y ahora ya cambió. Anteriormente era simplemente un input, pero ahora ya nos aparece un icono que hace referencia a un calendario y el formato de la fecha.

[01:07] Entonces yo puedo escribir aquí la fecha, por ejemplo, esta de aquí, o bien, pues bueno, puedo darle clic al iconito y pues puedo ir aquí jugando con las fechas. Muy bien. Entonces imagina que nosotros tenemos una regla en la cual solamente vamos a dejar que se complete el formulario o que este input sea válido, siempre y cuando sea mayor de 18 años.

[01:40] Es decir, que al menos haya nacido hace 18 años. Lo que voy a hacer entonces es voy a venir al código, voy a generar un archivo JavaScript, porque toda esta lógica ya la vamos a hacer con código de JavaScript. Ya no vamos a utilizar como tal a pattern porque realmente no podemos definir todo ese tipo de reglas tan específicas.

[02:07] Entonces lo que voy a hacer es voy a crear una carpeta que se llame JS, aquí dentro de JS le voy a poner el nombre de validaciones.js. Muy bien. Lo que voy a hacer entonces aquí, solo para corroborar que tenga mi archivo bien vinculado, voy a poner un console.log, voy a guardar.

[02:36] Aquí en mi archivo de registro.html, antes de que cierre el body, lo que voy a hacer es escribir una etiqueta script, abre y cierra y le voy a decir cuál es el source. En este caso va a ser js.validaciones. Voy a guardar. Voy a regresar a aquel proyecto, le voy a dar clic derecho inspeccionar aquí en la parte de la consola y aquí tengo mi consola de validaciones.

[03:06] Ya tenemos entonces funcionando la conexión entre el archivo HTML y el archivo JS. Ahora lo que yo quiero es que cuando yo salga de este input, es decir, cuando yo termine de completarlo y salga, verifique si la fecha que yo ingresé es mayor o hace referencia a que tengo más de 18 años. Entonces vamos a hacer eso.

[03:40] Lo primero que voy a hacer es encontrar dentro de todo mi DOM este input, en este caso este input puedo acceder a él a través del id “birth”, que hace referencia a la fecha de nacimiento. Voy a hacer entonces aquí una constante que se llame inputNacimiento = document.querySelector. Y entre paréntesis le voy a decir cuál es el id de mi input.

[04:21] Es este de aquí. Voy entonces a poner, a este input voy a agregarle un addEventListener. InputNacimiento.addEventListener. ¿Y cuál es el tipo de evento que yo quiero escuchar? Pues bueno, cuando salga, entonces va a ser “blur” o cuando quite el input o el foco de ese espacio o de ese input.

[04:45] Después voy a ponerle aquí una función anónima, voy a recibir mi evento, voy a poner entonces aquí solo un console.log(evento.target); voy a guardar, voy a regresar aquí. Voy a seleccionar aquí alguna fecha, le voy a dar clic en registrar y aquí entonces ya estamos obteniendo nuestro input.

[05:14] Ahora vamos a crear una función que se va a llamar validarNacimiento. Esta función lo que va a hacer es recibir nuestro input. Muy bien. Es más, vamos a agregarlo aquí de una vez, en lugar del console.log, lo que voy a hacer es validar nacimiento. Una vez que yo recibo el input, a mí como tal no me interesa en este momento este input de aquí, sino lo que yo quiero acceder, es al valor.

[05:50] ¿Entonces, cómo accedo a ese valor? Pues bueno, vamos a crear una nueva constante que se llame fecha y esta fecha va a ser input.value. Un console.log(fecha); voy a guardar, voy a venir de nuevo aquí a mi proyecto, voy a seleccionar alguna fecha y aquí tenemos la fecha. En este caso, por el formato que recibe, es año, mes y día, por eso es 2021-08-01.

[06:24] Ahora, entonces, ¿qué más vamos a hacer? Bueno, necesitamos verificar de alguna manera si esta fecha que nosotros estamos ingresando es mayor o tiene una diferencia de al menos 18 años, para corroborar que nosotros o el cliente que está llenando esta información es mayor de edad.

[06:45] Para eso, bueno, voy a crear una función que se va a llamar mayorDeEdad. Lo que va a recibir entonces aquí es la fecha. Muy bien, entonces voy a tener esta función, que es mayorDeEdad, y lo que voy a hacer aquí primero es crear una nueva instancia de nuestra clase date y lo que voy a hacer es mandarle este input.value.

[07:22] Vamos a guardar, vamos a regresar a ver qué es lo que obtenemos. Seleccionamos una fecha, registrar y aquí ya tenemos nuestra fecha. ¿Ahora qué más necesitamos? Bueno, necesitamos verificar si es mayor de edad, por eso vamos a recibir esta fecha.

[07:42] Vamos entonces a tener que hacer la comparación entre la fecha de hoy y la fecha que ingresó el usuario. Voy a crear entonces aquí una nueva fecha que se llama fechaActual, que simplemente lo que vamos a hacer es crear una nueva instancia de la clase date y al momento de simplemente definirla, no necesitamos especificarle nada, ya lo va a hacer automáticamente.

[08:11] Vamos a poner aquí mi console.log de fecha actual y esta fecha le voy a cambiar el nombre, tal vez como fechaCliente. Lo voy a poner aquí y en lugar de hacer el console.log, voy a poner aquí mayorDeEdad. Vamos a guardar este, vamos al navegador, entonces y voy a poner aquí la fecha que estamos recibiendo, que es fechaCliente y la fecha actual.

[08:51] Voy a poner aquí solo para separarlo algunos espacios. Voy a seleccionar aquí otra fecha, tal vez aquí. Bueno, aquí. Voy a seleccionar otra fecha. Bien. ¿Qué es lo que tenemos aquí? Tenemos la fecha que puso el cliente, que es 31 de diciembre del 2008, y la fecha de hoy, que es primero de agosto del 2021.

[09:31] Entonces ahora lo que vamos a necesitar es hacer la comparación. Vamos a ver cómo lo vamos a realizar. Muy bien, lo que vamos a hacer entonces es crear una nueva constante que se llame diferenciaFechas, va a ser igual a una nueva instancia de la clase date, en la cual le voy a poner la fecha que estamos recibiendo aquí, punto getUTCFullDate() + 18 años.

[10:10] Como segundo parámetro va a recibir el mes, que en este caso va a ser fecha.getUTCMonth(), y por último, fecha.getUTCFull. Aquí, perdón es FullYear y aquí es FullDate. Y vamos ahora a hacer el console.log(diferenciaFechas); vamos a seleccionar aquí alguna fecha.

[11:07] Bueno. Voy entonces a buscar cuáles son los métodos. Aquí tenemos getUTCDATE, getUTCFullYear. Entonces aquí tenemos la referencia de todos los métodos a los cuales vamos a poder acceder. En este caso, me parece que getUTCFullYearm, getUTCMonth y getUTCDate. Este FUllDate no debería de venir aquí.

[11:48] Vamos a guardar, vamos a volver a verificar aquí. Voy a seleccionar una nueva fecha. Entonces, ¿cuál es la diferencia o qué es lo que estamos obteniendo aquí? Lo que estamos obteniendo es la fecha que está ingresando el usuario como fecha de nacimiento y le estamos agregando 18 años. ¿Para qué?

[12:13] Para obtener por ejemplo este 2031. 2013 + 18, nos da 31, entonces por eso la fecha nos está mostrando 2031. Ahora lo que necesitamos regresar o verificar es si la fecha actual es menor o igual a la diferencia de fechas. Vamos a ver qué obtenemos. Voy a seleccionar aquí una fecha entonces.

[12:50] Me está diciendo true. Entonces al nosotros obtener este true, ¿qué es lo que nos está comentando? O sea que esta fecha actual, que era la fecha en la que estamos ahorita es menor, entonces nosotros tenemos al revés nuestra validación. Lo que puedo hacer aquí es simplemente fechaActual.

[13:28] Vamos a ver qué es lo que obtenemos, voy a seleccionar aquí de nuevo otra fecha. False. ¿Entonces en este caso esta fecha tiene al menos 18 años o esta persona que nació en este día o en esta fecha tiene al menos 18 años? Falso. ¿Qué pasa si yo vengo y lo voy a cambiar ahora por alguien que nació en el 2012 en enero?

[13:58] True. Entonces, la persona que nació el primero de enero del 2002 ya tiene al menos 18 años. Entonces, con esta función ya lo que hicimos fue obtener o saber si una persona es mayor o no de edad. Lo único que vamos a hacer en lugar de hacer el console.log es vamos a regresar esto de aquí.

[14:19] Al momento de regresarlo, pues bueno, ¿qué es lo que necesitamos? Necesitamos verificar si es o no mayor de edad a través de lo que nos está regresando. Y aquí viene algo interesante, nosotros tenemos nuestro input, pero vamos a necesitar algo que se llama setCustomValidity.

[14:44] Vamos a ver, al buscarlo, qué es lo que tenemos. El método SetCustomValidity define el mensaje de validación personalizado para el elemento seleccionado. Aquí con la contraseña, ves que si yo le doy clic aquí en registrar, bueno, tengo el nombre, disculpa.

[15:15] Aquí me aparece este texto, pues bueno, eso lo vamos a poder rellenar de una manera dinámica con Este setCustomValidity. Entonces es una función que va a recibir un mensaje. Este mensaje voy a hacer una variable let, donde el mensaje primero va a estar vacío, y si es mayor de edad, simplemente no nos importa darle ningún mensaje, porque esa es la idea que nosotros queremos, que sea mayor de edad.

[15:50] Entonces lo que voy a hacer es negarlo. Si no es verdadero, en este caso false, si es false, quiero que el mensaje sea: “Debes tener al menos 18 años de edad”. Voy a guardar. Vamos a ver qué es lo que obtenemos. De nuevo voy a poner aquí mi nombre, mi correo, la contraseña, le voy a dar clic en registrar. Tengo que seguir el patrón.

[16:33] Okay, completa este campo, le voy a decir que este usuario nació en el 2014 en enero. Clic en registrar y, como puedes darte cuenta, aquí ya me está pareciendo, debes tener al menos 18 años de edad. Entonces vamos a hacer ahora la prueba con alguien que pues sí ya tiene más de 18 años.

[17:00] Aquí ya el formulario sigue trabajando, ya me redireccionó a esta vista. No le hagan caso a esto de aquí, es una extensión. Pero bueno, lo importante es que ya nosotros estamos avanzando con nuestro formulario, ya aprendimos que es el SetCustomValidaty, ya también estuvimos checando esta parte de las fechas.

[17:21] Si te quedó alguna duda como habíamos buscado aquí, o sea puedes buscarlo en internet para verificar cuáles son todos los métodos a los cuales tienes acceso. Posiblemente haya mejores maneras de trabajar con todo esto, tal vez con alguna otra librería como Moment, alguna cosa así, pero bueno, esto es para nuestro ejercicio. Y pues bueno, vamos a dejar hasta aquí la clase. Te veo en la siguiente.

**Validando fecha de nacimiento**

**Validando fecha de nacimiento**

<legend class="form\_\_label">Información personal</legend>

<div class="input-container">

<input name="birth" id="birth" class="input" type="date" placeholder="Fecha de nacimento" required>

<label class="input-label" for="birth">Fecha de nacimiento</label>

<span class="input-message-error">Este campo no es valido</span>

</div>

Función que va a recibir un mensaje

setCustomValidity

<https://developer.mozilla.org/en-US/docs/Web/API/HTMLObjectElement/setCustomValidity>

Archivo validaciones.js

const inputNacimiento = document.querySelector("#birth")

//evento blur, es cuando quita el foco de ese espacio o ese input

inputNacimiento.addEventListener("blur", (evento) => {

validarNacimiento(evento.target);

});

//se crear función para recibir el input

function validarNacimiento(input) {

let mensaje = ""

//acceder al valor de fecha, formato año, mes y dia

const fechaUsuario = new Date(input.value);

if (!mayorDeEdad(fechaUsuario)) {

mensaje = "Debes tener al menos 18 años de edad";

}

input.setCustomValidity(mensaje)

}

//verificar si es mayor de edad

function mayorDeEdad(fecha) {

const fechaActual = new Date();

const diferenciaFechas = new Date(

fecha.getUTCFullYear() + 18,

fecha.getUTCMonth(),

fecha.getUTCDate()

)

return diferenciaFechas <= fechaActual;

}

**Trabajando con validity**

Hemos visto que para enseñar mensajes de error podemos utilizar el atributo title. En JavaScript también es posible tratar los errores de validación utilizando la propiedad validity del input, como utilizamos en la validación customizada del campo “fecha de nacimiento”.

¿Cómo adicionar un mensaje customizado de error para validaciones que no estén en los padrones de HTML?

Seleccione una alternativa

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Mejorando el código**

[00:00] Bueno, vamos a continuar. Nosotros ya logramos saber si la fecha de nacimiento es mayor a 18 años, pero lo que nosotros hicimos aquí de esta constante de input nacimiento, el querySelector, yendo solamente a una id no es tan buena práctica.

[00:21] Además, nosotros queremos que este addEventListener lo podamos reutilizar, es decir, no solamente utilizarlo para el input nacimiento o tener que escribir un addEventListener por cada uno de nuestros inputs. Lo que queremos es que sea lo más sencillo o reutilizable nuestro código. Lo que voy a hacer entonces aquí es simplemente voy a eliminar esto de aquí. Voy a ir a mi registro HTML.

[00:49] Y lo que nosotros posiblemente vamos a utilizar, y ya tal vez lo has visto en otros cursos es son los data attributes. Entonces le voy a poner data-tipo= “nacimiento”. Voy a guardar, voy a venir aquí a mi archivo de validaciones.js y lo que voy a hacer es crear una función que se llama válida() y esta función lo que va a recibir va a ser nuestro input.

[01:29] Aquí lo que vamos a tener es una constante que se va a llamar tipoDeInput que va a ser igual a nuestro input.dataset.tipo. ¿Qué es este dataset? Lo que nosotros estamos poniendo aquí, data-tipo, bien pudiéramos poner tal vez un data, guión otra cosa. Entonces con este dataset lo que nosotros estamos obteniendo es la colección de todos los datas y el .tipo es para obtener justamente este de aquí.

[02:10] Si le quieres poner otro nombre, pues bueno, tendrás que cambiarlo. Voy a eliminarlo entonces aquí. Ahora, lo que yo quiero es tener una constante que se va a llamar validaciones o validadores, va a ser un objeto. Y en el cual les voy a poner para que coincidan el nombre del tipo con la llave dentro de este objeto.

[02:37] En este caso, como yo le puse aquí nacimiento, pues bueno. Voy a poner aquí nacimiento. Va a ser una función que va a recibir input y lo que va a hacer es mandar a llamar validarNacimiento. Es aquí simplemente un arrow function y le voy a mandar entonces aquí mi input, aquí ya lo tenemos.

[03:02] Lo que nosotros entonces necesitamos es por cada uno de los tipos de input o por el tipo de input, verificar si existe dentro de los validadores, es decir, si es nacimiento yo quiero que se le asigne entonces nacimiento. Lo que voy a hacer aquí es primero verificar si dentro de validadores existe el tipo de input.

[03:33] Y si existe, lo que voy a hacer entonces es hacer validadores[tipoDeInput] y voy a mandar a llamar aquí o le voy a pasar como parámetro input. Entonces este input es el que va a recibir o el que va a estar vinculado con otro archivo que ahorita vamos a ir a generando. Pero bueno, más bien lo que voy a hacer es guardar esto de aquí.

[04:00] Voy a crear un nuevo archivo que se va a llamar app.js para tenerlo más separado. Y lo que voy a hacer entonces es exportar esta función: export function valida. ¿Para qué? Para que yo pueda utilizarlo en otros archivos. Voy a importar, vamos a ver si el autocompletado está aquí, aquí está. Importó valida de validaciones.js.

[04:32] Muy bien, ya tenemos entonces aquí nuestra función de valida. Ahora lo que nosotros necesitamos es agregar el addEventListener que va a mandar a llamar esta función cada vez que el input sufra el cambio o tenga el cambio de blur, cada vez que salga el usuario de ese input.

[04:53] Entonces lo primero que vamos a hacer es hacer una constante que se llame inputs. Este inputs va a ser igual a document.querySelectorAll() para que seleccione todos nuestros inputs, y lo que voy a hacer entonces aquí es voy a ponerle todos mis elementos que sean de tipo input.

[05:16] Después, inputs. Yo cuando digo document.querySelectorAll lo que me va a regresar es un arreglo. Y como recuerdas, los arreglos los podemos iterar, lo cual voy a hacer aquí. Lo que vamos a recibir aquí es nuestro input, y lo que voy a hacer entonces es para cada uno de los inputs vamos a obtener input, lo que vamos a tener entonces es a nuestro input.addEventListener le vamos a decir cuál es el tipo de evento que queremos escuchar.

[05:51] En este caso va a ser blur. Va a recibir una función anónima, un arrow function, un callback. Le vamos a decir entonces que nosotros vamos a recibir aquí nuestro input. Vamos a checar entonces aquí nuestro valida y le vamos a mandar este input, pero el .target, entonces vamos a hacer input.target, vamos a guardar y bueno, como te comentaba, la idea es que primero vamos a seleccionar todos los inputs, todos los inputs que nosotros tenemos aquí.

[06:24] Después, lo que va a pasar es le va a agregar a cada uno de estos inputs el addEventListener de blur cuando salga de foco, y cuando salga de foco va a mandar a llamar esta función la cual es valida. Esta función valida vive aquí en validaciones, va a recibir el input, va a verificar cuál es el tipo de input a través de dataset.tipo y a partir de ahí lo que vamos a tener que ir armando, pues bueno, es aquí todo un objeto con los diferentes tipos de input que vamos a tener.

[07:03] Vamos a tener uno para nacimiento, otro para nombre, otro para correo electrónico o número telefónico, y así sucesivamente. Entonces vamos a dejar hasta aquí la clase y te veo en la siguiente.

**FORO:**

En cuanto a la función ForEach, te puedo explicar que se utiliza para recorrer los elementos de un arreglo y ejecutar una función para cada uno de ellos. Por ejemplo, si tienes un arreglo de números y quieres imprimir cada uno de ellos, puedes utilizar la función ForEach para recorrer el arreglo y llamar a la función console.log para imprimirlos.

NO ENTENDÍ BIEN

¡Hola Edwin!

Entiendo que te sientas confundido con esta clase. Permíteme explicarte un poco más sobre lo que está sucediendo en el código.

En la función valida, el parámetro input representa el elemento HTML que se está validando. La constante tipoDeInput almacena el valor del atributo data-tipo de ese elemento. Luego, se verifica si existe una función de validación correspondiente a ese tipo en el objeto validadores. En este caso, la única función de validación existente es validarNacimiento, por lo que se llama a esa función pasando el input como parámetro.

En el archivo app.js, la constante inputs almacena todos los elementos input del HTML. Esto se hace para poder agregar el evento blur a cada uno de ellos. El método forEach se utiliza para iterar sobre cada elemento input y agregar el evento. Dentro de la función de callback del forEach, input representa cada uno de los elementos input en cada iteración.

En el ejemplo que muestras en app.js, se está utilizando el evento blur para llamar a la función valida y pasar el elemento input como parámetro. Esto es similar a lo que se hace en el otro ejemplo que muestras en el archivo app.js, donde se llama a la función validarNacimiento pasando el evento.target como parámetro.

En resumen, el parámetro input se utiliza para representar el elemento HTML que se está validando y se pasa como argumento a las funciones de validación correspondientes.

Espero que esta explicación haya aclarado tus dudas. Si tienes alguna otra pregunta, estaré encantado de ayudarte.

¡Espero haber ayudado y buenos estudio

**PREGUNTA**

Hola, un gran saludo a la comunidad, hoy necesito ayuda para entender un metodo. Se trata del método valida.

Exactamente, desde la línea 3 hasta la 5 no entiendo. Un desarrollador "berraco" que lo pueda explicar, fuera genial.

¡Hola Estudiante!

¡Saludos a ti también! Estoy aquí para ayudarte a entender el método valida del archivo validaciones.js.

En las líneas 3 a 5 del código, se está declarando la función valida. Esta función toma dos parámetros: campo y tipoValidacion.

El parámetro campo representa el campo de entrada del formulario que se desea validar. Por ejemplo, si tienes un formulario con un campo de nombre, podrías pasar el elemento de ese campo como argumento a la función valida.

El parámetro tipoValidacion representa el tipo de validación que se desea realizar en el campo. Por ejemplo, si quieres validar que el campo de nombre no esté vacío, podrías pasar el valor "noVacio" como argumento a la función valida.

Dentro de la función valida, se utiliza un condicional if para verificar el valor del parámetro tipoValidacion. Dependiendo del valor, se ejecutará el código correspondiente para realizar la validación específica.

Por ejemplo, si tipoValidacion es igual a "noVacio", se verifica si el campo está vacío utilizando la propiedad value del elemento campo. Si el campo está vacío, se muestra un mensaje de error utilizando alert.

Aquí tienes un ejemplo práctico de cómo podrías utilizar la función valida:

PREGUNTA MIA

Buenas noches a toda la comunidad, este tema donde se supone que es la base ha sido la más dificil de entender y la que se supone que es la especialización, que frustración, pienso que son conceptos complejos y deberia haber una explicación mas profunda , ya que la metodologia impartida es como si el instructor estuviera explicando el codigo a una persona que ya domina la programación. Señores Alura este tema ha sido un matadero según veo por los comentarios. Igualemente agradezco el esfuerzo. Hay dos conceptos en especial que me gustaria que por favor me explicaran (blur y target) según entendí blur es cuando pierde el foco pero entonces cual es la diferencia con focus? y target no lo tengo claro; respecto al resto de codigo no entendí nada. El instructor quito el codigo

¡Hola Graciela!

Lamento mucho que estés teniendo dificultades con este tema en el curso. Entiendo que puede ser frustrante cuando los conceptos no se explican de manera clara y profunda. Permíteme ayudarte a entender los conceptos de "blur" y "target".

En el contexto de la programación web, "blur" se refiere a un evento que ocurre cuando un elemento pierde el foco. Por ejemplo, si tienes un campo de texto y haces clic en otro lugar de la página, el campo de texto perderá el foco y se disparará el evento "blur". Esto puede ser útil para realizar validaciones o ejecutar ciertas acciones cuando el usuario deja de interactuar con un elemento.

Por otro lado, "target" se refiere al elemento específico que disparó un evento. En el código que compartiste, el evento "blur" está asociado al elemento con el id "birth" (un campo de fecha de nacimiento). Cuando este campo pierde el foco, se ejecuta la función "validarNacimiento" pasando como argumento el elemento que disparó el evento, es decir, el campo de fecha de nacimiento en este caso.

En cuanto a la validación de si es menor de edad, es posible que haya algún error en el código que no esté funcionando correctamente. Sin embargo, sin ver el resto del código y la función "validarNacimiento", es difícil determinar cuál podría ser el problema. Te sugiero revisar el código en busca de posibles errores o compartir más detalles para que podamos ayudarte de manera más precisa.

Recuerda que la programación puede ser desafiante, pero con práctica y paciencia, podrás comprender y dominar estos conceptos. ¡No te desanimes! Si tienes alguna otra pregunta o necesitas más ayuda, no dudes en preguntar.

¡Espero haber ayudado y buenos estudios!

**Para saber más: Regex**

Existen diversas reglas para la escritura de una regex. Por eso, el entendimiento de cada una de ellas exige mucha atención. Una herramienta que recomiendo bastante para entrenar la escrita de regex es la página <https://regexr.com/>. Esta página nos regala una tabla con ejemplos, un cheatsheet con algunas reglas y también un validador explicando todo lo que está pasando durante la escritura.

Les comento que tengo un curso aquí en Alura Latam sobre las regex, les dejo el link aquí en el caso de que no lo hayan visto.

Una sugerencia es buscar ejemplos de regex en internet y pegarlos en ese validador para ver las explicaciones. Tenemos que recordar que una regex puede estar formada por diversas reglas, entender partes de la escritura de algunas de estas expresiones puede facilitar el entendimiento en la hora de escribir tus propias reglas.

**Haga lo que hicimos en aula**

Practicar ayuda mucho en el aprendizaje de un nuevo concepto. Así siendo, es muy importante que implementes lo que fue presentado en esta clase.

VER OPINIÓN DEL INSTRUCTOR

Opinión del instructor

No dejes de quitar tus dudas antes de dar continuidad al curso. En el caso de que tengas alguna duda esa podrá ser contestada por nuestra comunidad en el foro de Alura Latam.

**Lo que aprendimos**

Lo que aprendimos en esta aula:

A validar fechas en nuestro formulario.

A utilizar expresiones regulares para mejorar la validación de nuestro formulario.

**Proyecto del aula anterior**

¿Comenzando en esta etapa? Aquí puedes descargar los archivos del proyecto que hemos avanzado hasta el aula anterior.

Descargue los archivos en Github o haga clic aquí para descargarlos directamente.

**Live Server y Browser Sync**

[00:00] Muy bien. Vamos a continuar. Si recuerdas, lo que nosotros hicimos aquí fue crear un archivo nuevo, app.js, en el cual importábamos nuestra función valida que vive dentro de nuestro archivo validaciones. Pero nosotros al momento de hacer ese cambio, lo que nosotros necesitamos entonces ahora en lugar de utilizar el archivo, validaciones.js vamos a utilizar app.js.

[00:28] Voy a guardar, vamos a regresar aquí. Y nos comenta que no se puede utilizar import statement fuera de un módulo, entonces necesitamos definir aquí que el tipo va a ser un módulo, voy a guardar. Voy a poner aquí harland, mi correo harland@aluracursos, mi contraseña, mi fecha de nacimiento, voy a poner la fecha del primero de febrero del 2019. Les voy a dar clic en registrar.

[01:08] Aquí. Al parecer hubo algún tema con la contraseña. Clic en registrar y bueno, sigue funcionando, me dice: Debes tener al menos 18 años de edad. Ahora, si yo vengo acá, voy a poner una fecha más antigua. Clic en registrar. Está funcionando. Ahora posiblemente si tú no utilizaste la extensión de Live Server, este Open with Live Server, sería buen momento para que lo hicieras.

[01:44] ¿Esto por qué? Porque bueno, si yo vengo aquí a mi explorador de archivos y voy a abrir aquí entonces el proyecto, le voy a dar clic en inspeccionar y ahora vamos a la consola. Aquí me comenta que no puede como tal encontrar este archivo debido a políticas de curso. Prácticamente lo que está esperando es que este archivo viva en un servidor http.

[02:15] Como no está viviendo o como no lo tenemos levantado en un servidor por eso nos está marcando este error. La recomendación sería bueno aquí en la parte de extensiones, busca simplemente Live Server, este de aquí y lo instalas. En caso de que no estés trabajando con visual Studio Code también podemos utilizar algo que se llama browser sync. Vamos a darle enter. Este de aquí.

[02:56] Y seguimos las instrucciones. En este caso lo instalas, tienes que tener instalado Node, y bueno, una vez que ya lo instalas, lo que vas a tener que hacer es venir a una terminal. En este caso voy a utilizar la de Visual Studio Code. Esperamos a que cargue. Y tendrás que escribir el siguiente comando: browser-sync start -s -f . - -directory. Le voy a dar enter.

[03:37] Y lo que va a hacer esta extensión es levantarnos un servidor. Aquí, nosotros ya entonces vamos a obtener nuestro árbol de todos nuestros documentos, le voy a dar clic en registro\_html, inspeccionar. En la consola no obtenemos ningún error. Voy a poner aquí mi correo que sería harland@aluracursos.com, mi contraseña, fecha de nacimiento, mi nombre, clic en registrar. Okay, está funcionando.

[04:26] Lo único que voy a hacer, voy a seleccionar otra fecha. Bueno, esta sería otra manera en la cual pudieras ir trabajando con este proyecto. La verdad, yo prefiero más utilizar Live Server sin necesidad de tener que levantar browser-sync, pero de nuevo si estás utilizando algún otro editor de código, pues bueno, puedes utilizar simplemente este comando.

[04:52] Recuerda instalarlo antes, si no, no te va a funcionar, y vas a poder levantar entonces un servidor http, para que no obtengas el problema que vimos anteriormente.

**NOTAS**

[01:29] Aquí lo que vamos a tener es una constante que se va a llamar tipoDeInput que va a ser igual a nuestro input.dataset.tipo. ¿Qué es este dataset? Lo que nosotros estamos poniendo aquí, data-tipo, bien pudiéramos poner tal vez un data, guión otra cosa. Entonces con este dataset lo que nosotros estamos obteniendo es la colección de todos los datas y el .tipo es para obtener justamente este de aquí.

<input name="birth" id="birth" class="input" type="date" placeholder="Fecha de nacimento" required data-tipo="nacimento">

FORO

**Para saber más: Live Server y Browser Sync**

Live Server

(<https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>)

Browser Sync

<https://browsersync.io/>

**Mostrando mensajes de error**

[00:00] Bueno, continuando con todo esto sobre verificar si la información que está poniendo el usuario es o no correcta, lo que nosotros hemos visto ahorita es que si yo por ejemplo le doy clic aquí en registrar, me aparece este mensaje de completa este campo, pero realmente este mensaje y el formato o el estilo puede cambiar de navegador a navegador.

[00:26] Para lo cual, imagínate que tú en tu empresa o en tu lugar de trabajo te dicen: “mira, yo quiero”, el equipo de UX o de UI, quieren que cuando exista ese error o exista algún error, nuestro input que presenta ese error se ponga con rojito. Lo que voy a poner aquí es input-container--invalid. ¿Qué es lo que estamos teniendo aquí?

[00:56] Bueno, estamos aplicando una clase de CCS en la cual, pues bueno, se está poniendo el estilo en rojo y resalta a la vista, en el cual nosotros estamos viendo que este campo no es válido. Sin embargo, lo que nosotros hicimos fue escribirlo directamente, no es lo que queremos.

[01:16] Voy entonces a eliminar esta clase que acabo de poner. Y, antes de proseguir a codificarlo, mira. Lo que quiero enseñarte aquí es que nosotros tenemos, estamos seleccionando aquí nuestro input y después lo que voy a hacer es .validity. Este validity es prácticamente un objeto en el cual nos va a ir diciendo si nuestro input presenta ciertas características, por ejemplo.

[01:58] Nos dice aquí que sí es válido. En este caso está false. Aquí dice que si está faltando nuestro valor, en este caso es true. ¿Qué es lo que queremos entonces? Que a partir de este valid verificar si está o no válido y dependiendo de eso agregar o quitar la clase. Lo que voy a hacer es venir de nuevo aquí a validaciones, aquí en mi función valida. Recuerda que esta función se manda llamar cada vez que el usuario sale del input que estaba rellenando.

[02:30] Lo que voy a hacer entonces es verificar if(input.validity.valid), si es válido, o sea, si está en true, quiero que quite la clase. En todo caso quiero que ponga la clase CCS para que represente que está mal. ¿Entonces, cómo vamos a hacer eso? Bueno, lo que vamos a hacer entonces es primero verificar aquí nuestro input.

[03:16] Voy a guardar, vamos a esperar a que recargue. Aquí tengo mi input. Pero si nosotros vemos nuestro HTML no es al input al cual le tenemos que poner la clase, sino es al padre. Y si recuerdas las clases de HTML con Dom, bueno, sabes que lo que necesitamos hacer es input.parentElement. Vamos a ver qué obtenemos aquí.

[03:53] Vamos de nuevo aquí. Aquí tenemos nuestro div. Entonces es a partir a este div al que nosotros vamos a querer agregarle esta clase. Y bueno, lo único que tendremos que hacer es decir Input.parentElement.classList. Recuerda la L en mayúscula. Y en caso de que sea true, lo que voy a querer es que lo remueva.

[04:21] Remove. ¿Cuál es la clase que quiero que quite? Vamos a ver. Este “input-container--invalid”. En caso de que no sea válido, voy a querer entonces que mi input.parentElement.classList.add y voy a querer que agregue esta clase. Vamos a probar. Aquí está el nombre. Vengo afuera, aquí ya me agregó la clase de CCS. Voy a poner mi nombre, salgo y ya la quitpo.

[05:10] Entonces, bueno, recuerda. Nosotros utilizamos este if simplemente para verificar si es válido o si el validity es válido. Si es true, va a quitar la clase. Si es false significa que no es válido, por lo cual va a agregarla. Voy entonces aquí a eliminar este console.log.

[05:29] Y ahora si nosotros regresamos aquí, dice: este campo no es válido y así, si seguimos probando con los demás, va a seguir teniendo el mismo texto. Lo que nosotros queremos es que dependiendo del error, que es este de aquí, .validity, dependiendo de estos errores, o al menos algunos de ellos, va a ser el texto que nosotros vamos a mostrarle al usuario.

[06:06] Entonces lo que voy a hacer aquí, voy a crear un objeto y bueno, y pues bueno se va a llamar mensajes de error. Esto, como te comentaba, es un objeto y dentro de este objeto vamos a ir definiendo cuáles van a ser los tipos de errores de cada uno de los inputs. Si recuerdas, nosotros teníamos aquí el input de nacimiento que mandaba llamar esta función, vamos a hacer prácticamente lo mismo.

[06:35] Nuestro primer input que tenemos, bueno lo voy a poner en español. Tenemos aquí nuestro mensaje de error de nombre. Esto de nuevo va a ser un objeto. En este caso, como es el nombre, yo lo que quiero verificar primero es verificar si el valor está faltando, valueMissing. Este valueMissing, lo que voy a poner como texto de error es: “Este campo no puede estar vacío”. Muy bien.

[07:15] Vamos entonces con el que sigue, que es email. Aquí vamos a tener nuestro email, que de nuevo son objeto. Voy a copiar este valueMissing, de nuevo, esto sí es necesario y lo que vamos a querer después entonces es nuestro typeMissmatch. Este tipo hace referencia a que es correo electrónico. Entonces vamos a poner typeMissmatch.

[08:01] Voy a poner entonces: “El correo no es válido”. Vamos con el que sigue. Tenemos el de contraseña. Lo voy a poner en inglés para que no tengamos problema. De nuevo, valueMissing, en caso de que este campo está vacío. Y lo que vamos a tener ahora es nuestro patternMismatch.

[08:33] Si recuerdas nosotros en este input lo que hicimos fue agregar un patrón que nosotros esperábamos con una regex. Entonces lo que voy a hacer es venir aquí a mi archivo HTML. Voy a venir a la parte de title, este de aquí, lo voy a copiar. Voy a venir de nuevo aquí a validaciones y ahora lo que voy a querer es patternMissing me parece, vamos a verificar, patternMismatch y voy a pegar el texto. Muy bien, ya tenemos ahí el de contraseña, vamos con el siguiente. Es el de fecha de nacimiento.

[09:20] Entonces lo que voy a hacer es utilizar el mismo nombre que era nacimiento. De nuevo va a ser un objeto, voy a poner cuál es el valueMissing, en este caso este de aquí. Y lo que vamos a tener después es nuestro customError. Este customError es este de aquí, este mensaje de aquí que nosotros agregamos con setCustomValidity. Entonces lo voy a copiar. Voy a ver qué es customError y lo voy a pegar.

[10:04] Entonces bueno, vamos a esperar aquí a que recargue. Y ahora lo que nosotros necesitamos ver es, dependiendo del tipo de input, que el usuario está interactuando es verificar si está válido o no está válido. En caso de que no sea válido, vamos a necesitar acceder a este objeto y a cada una de sus llaves, dependiendo del tipo de input, pero bueno, eso lo vamos a ver en el siguiente video.

Archivo validaciones.js

export function valida(input) {

    //required data-tipo="nacimento

    const tipoDeInput = input.dataset.tipo;

    if (validadores[tipoDeInput]) {

        validadores[tipoDeInput](input); // vinculado con el otro archivo app.js

    }

    console.log(input.parentElement);

    if (input.validity.valid) {

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

    } else {

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

      }

}

const mensajesDeError = {

    nombre: {

        valueMissing: "Este campo no puede estar vacio",

    },

    email: {

        valueMissing: "Este campo no puede estar vacio",

        typeMismatch: "El correo no es valido",

    },

    password: {

        valueMissing: "Este campo no puede estar vacio",

        patternMismatch: "Al menos seis caracteres maximo 12 , debe contener una letra minuscula , 1 letra mayuscula. 1 numero , no puede contener caracteres especiales",

    },

    nacimiento: {

        valueMissing: "Este campo no puede estar vacio",

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

    },

},

const validadores = {

    // en registro usamos la palabra nacimiento ( required data-tipo="nacimento")

    nacimiento: input => validarNacimiento(input),

};

//se crear función para recibir el input

function validarNacimiento(input) {

    const fechaCliente = new Date(input.value);

    let mensaje = "";

    //acceder al valor de fecha, formato año, mes y dia

    const fechaUsuario = new Date(input.value);

    if (!mayorDeEdad(fechaCliente)) {

        mensaje = "Debes tener al menos 18 años de edad";

    }

    input.setCustomValidity(mensaje);

}

//verificar si es mayor de edad

function mayorDeEdad(fecha) {

    const fechaActual = new Date();

    const diferenciaFechas = new Date(

        fecha.getUTCFullYear() + 18,

        fecha.getUTCMonth(),

        fecha.getUTCDate()

    );

    return diferenciaFechas <= fechaActual;

}

**Tipos de errores**

[00:00] Bueno, vamos a continuar. Primero que nada, si recuerdas, cuando nosotros mostramos aquí un error, aparece este texto, este campo no es válido. Lo que vamos a necesitar entonces es en nuestra función aquí de valida, en caso de que no existe algún error o de que sea válido, lo que vamos a necesitar es seleccionar ese spam, este de aquí, y simplemente ponerle el innerHTML como vacío.

[00:41] Entonces lo que voy a hacer aquí es input.parentElement.querySelector, y es una clase, por lo tanto es punto y el nombre de la clase. O bien podría ser spam, pero en este caso voy a utilizar la clase .innerHTML va a ser igual a un string vacío. Voy a copiar esto de aquí, y ahora en caso de que si existe algún error, lo que quiero es mostrar error, mostrar mensaje de error, para que se entienda mejor.

[01:28] Va a ser una función. En esta función lo que voy a recibir yo va a ser el tipo de input como primer parámetro y el input. Ahorita vamos a ver para qué. Lo que voy a hacer entonces aquí en esta función de mostrar mensaje de error, voy a generarla, no existe. La voy a generar aquí. Va a recibir tipo de input y el input como tal.

[02:00] Muy bien, lo que nosotros vamos a tener entonces aquí es un mensaje que es el que queremos decirle, o sea, dependiendo de si el campo está vacío, si no cumple con ciertas características, prácticamente lo que nosotros necesitamos es acceder a estos diferentes mensajes que escribimos en el video anterior.

[02:21] Entonces, bueno, aquí ahora voy a poner un return de nuestro mensaje. Muy bien. ¿Ahora qué es lo que necesitamos? Bueno, lo que necesitamos es de alguna manera interceptar dependiendo del tipo de input, bueno, si el tipo de input es nombre, tendría que acceder aquí y nos debería demostrar nuestro error de value missing.

[02:53] ¿Pero qué pasa, por ejemplo, con este de email que tiene un typeMismatch, o este de aquí que tiene un patternMismatch? Nosotros tendríamos que hacer muchos ifs y diferentes tipos de lógica para poder acceder a cada una de estas llaves de cada uno de los objetos.

[03:13] Lo que vamos a hacer mejor vamos a crear una función que se llame tipo de errores, que va a ser un arreglo. Este arreglo, lo que va a tener es el valiueMissing. Va a tener typeMismatch, va a tener pattern, y por último, va a tener el customError. Muy bien, ahora lo que nosotros tenemos aquí ya es el tipo de input.

[03:48] Lo que vamos a necesitar después es recorrer este arreglo y dependiendo, lo que voy a hacer entonces es para cada tipo de errores, lo voy a recorrer, si recuerdas, es un arreglo: tipoDeErrores.forEach. En este tipo de errores voy a recibir cada uno de estos errores.

[04:10] Es decir, la primera vez que pase va a acceder a valueMissing, después a typeMismatch, después patternMismatch, después customError. Ahora lo que yo voy a querer es si dentro de mi input, recuerdas que trabajamos con validity, y yo quiero buscar este error, en caso de que este error valiueMissing por ejemplo sea true, yo lo que voy a querer entonces es que el mensaje sea, por ejemplo, el campo no puede estar vacío.

[05:02] Lo que voy a hacer aquí es un console.log, mejor para que quede un poco más claro, de esto de aquí. Vamos a guardar, vamos a regresar, vamos aquí con el nombre. Y es más, aquí me regreso que true, pero lo que voy a poner entonces aquí es para verificar cuál era el error. Vamos a ver.

[05:29] Recargo ahí, salgo y me dice valueMissing. Entonces está faltando ese valor. Por lo tanto, yo lo que necesito es, ya sé que es valueMissing el que está faltando. Ahora lo que necesitamos es ver el tipo, eso es con este tipo de input que nosotros le estamos mandando. Siguiendo toda esta estructura, yo lo que quiero hacer aquí es un console.log(mensajesDeError[tipoDeInput])

[06:04] Este tipo de input puede tener valores de nombre, email, password, nacimiento y después voy a querer acceder a cada uno de estos valores. Puede ser valueMissing, typeMismatch. ¿Entonces cómo voy a acceder a ese? Pues bueno, voy a poner aquí el error. Voy a guardar, voy a regresar aquí. Selecciono, salgo y al ocurrir un error, vamos a ver qué fue lo que pasó.

[06:33] Nuestro tipoDeInput. Vamos a esperar aquí. Vamos a ver qué es lo que obtenemos. Y me aparece un error. ¿Por qué dice undefined? Estamos recibiendo aquí undefined. Vamos a verificar aquí. Teníamos el tipo, vamos a ver si nuestro input tiene el data-attribute y no lo tiene, por eso nos está mandando un error.

[07:03] Entonces lo que nosotros necesitamos decirle aquí nuestro input es data- tipo. Si recuerdas, pues bueno, tiene que ser nombre. Vamos a poner aquí nombre, voy a guardar, vamos a verificar. Ahora sí, ya no me mandó el error. ¿Qué es lo que dice? Este campo no puede estar vacío. Ya estoy teniendo acceso entonces a este texto.

[07:34] Lo que voy a hacer antes de eso, voy a terminar de rellenar todos estos data-tipo. Entonces, vamos a tener data-tipo. Este de aquí. Este data-tipo va a ser email. Después vamos a poner nuestro otro data-tipo, que pues bueno, va a ser password, que es la contraseña. Y vamos ahorita con el último que hemos trabajado que es fecha de nacimiento, y este sí lo tiene, entonces no tenemos ningún tema.

[08:14] Vamos a esperar aquí, vamos a seleccionar. Vamos a verificar por qué no le está quitando este texto, vamos a ver, perdón, estos estilos. Classlist remove. Al parecer debería de quitarse. Vamos a ver por qué no lo está haciendo. Input-container-invalid. Vamos a verificar.

[08:50] Bueno, lo que nosotros entonces necesitamos aquí es, ya tenemos aquí nuestro console.log. Aquí ya nos está mostrando cuál es el error que se está generando. Lo que necesitamos, por último, es simplemente decirle aquí mensaje va a ser igual a mensajes de error, entre corchetes, el tipo de error para poder acceder a si es nombre, email, password, nacimiento, el tipo de input y ahora sí por último, nuestro error.

[09:23] Vamos a guardar. Vamos a recargar. Tenemos nombre, clic afuera, este input no puede estar vacío. Okay, le voy a poner aquí mi nombre, Harland. Voy a poner aquí el correo electrónico, este campo no puede estar vacío, que es este de aquí. Entonces, si quisiéramos ser más específicos, pudiéramos decir tal vez el campo nombre no puede estar vacío, el campo correo no puede estar vacío.

[10:05] El campo contraseña no puede estar vacío y así sucesivamente. Vamos a verificar. Vamos aquí a nuestro correo, salimos, el campo correo no puede estar vacío, el campo nombre no puede estar vacío, entonces al parecer está funcionando. Entonces Harland, mi correo, vamos a ver si solo le pongo mi nombre.

[10:29] Ahora dice que el correo no es válido, que entonces sería porque el typeMismatch no está en true. Sin embargo, valueMissing está en false porque ya tiene un texto. Entonces bueno, vamos a seguir viendo, tenemos nuestra contraseña. Vamos a ver tres números. Ahí me dice que hay un error. El campo contraseña no puede estar vacío, eso está bien. Aquí ya tengo la contraseña.

[11:05] Vamos a ver, ahí ya se ve la contraseña, mi correo, ahí también ya está bien. Fecha de nacimiento. Vamos a ver, voy a seleccionar este de aquí. Debes tener al menos 18 años de edad. Voy a ver si cambiándolo. 2005, 18. Necesitamos todavía más. Entonces bueno, como pudiste darte cuenta, utilizamos diferentes objetos o ir trabajando con objetos, justamente para tener el código un poco más organizado.

[11:52] Si nosotros conforme fuéramos realizando tal vez algún formulario, queremos manejar algún otro tipo de input, perdón, algún otro tipo de errores, o bien lo agregamos aquí y simplemente en este objeto ya lo único que tendremos que hacer es irlos agregando.

[12:09] La ventaja que tiene trabajar con este tipo de objetos es que, imagínate, tu página puede estar en portugués, puede estar en español, puede estar en inglés, en cualquier idioma, y realmente lo único que tendrías que hacer es tal vez desde un API o desde algún otro lado, simplemente traer estos valores sin necesidad que estén escritos necesariamente en tu código HTML.

[12:35] Entonces de esta manera tienes una mejor organización de código. Y pues bueno, la vamos a dejar hasta aquí esta clase. Te veo en la próxima.

<https://developer.mozilla.org/es/docs/Learn/Forms/Form_validation>

<https://www.w3schools.com/js/js_validation_api.asp>

**Mensajes del navegador**

Cuando existe algún error de validación en el formulario aparecerá un mensaje arriba del campo explicando o dando algún tip para corregirlo. Pero en el proyecto existe una estilización y una tag <span> para enseñar un mensaje de error directo en el HTML.

Elije las opciones que mejor definen las motivaciones para no utilizar el mensaje padrón del navegador:

Selecciona 3 alternativas



**Haga lo que hicimos en aula**

Practicar ayuda mucho en el aprendizaje de un nuevo concepto. Así siendo, es muy importante que implementes lo que fue presentado en esta clase.

VER OPINIÓN DEL INSTRUCTOR

Opinión del instructor

No dejes de quitar tus dudas antes de dar continuidad al curso. En el caso de que tengas alguna duda esa podrá ser contestada por nuestra comunidad en el foro de Alura Latam.

**Lo que aprendimos**

Lo que aprendimos en esta aula:

A instalar y utilizar el plugin Liveserver.

Como descargar y correr el Browser-Sync.

Como enseñar el mensaje de error directamente en el HTML.

Como customizar los mensajes de error de validación

**Proyecto del aula anterior**

¿Comenzando en esta etapa? Aquí puedes descargar los archivos del proyecto que hemos avanzado hasta el aula anterior.

Descargue los archivos en Github o haga clic aquí para descargarlos directamente.

**Completando formulario I**

[00:00] Muy bien. Vamos a continuar. Si recuerdas, ya tenemos nosotros los inputs con las validaciones de nombre, correo electrónico, contraseña y fecha de nacimiento. Vamos entonces con el de número telefónico. El número telefónico va a variar dependiendo del país e inclusive hasta de mismo de ciudad.

[00:20] Entonces ahorita para este ejemplo lo único que vamos a hacer es generar una expresión regular muy sencilla de 10 dígitos, y si tú quieres desarrollar un proyecto en el cual solamente sea para tu área local o para tu país, puedes tú buscar la expresión regular del número telefónico de tu país.

[00:42] Vamos entonces a nuestro código. Aquí, en la línea 50 yo lo que tengo es un input. ¿Cuál es el tema con esto? Nosotros tenemos el tipo texto. Si yo lo cambio esto por type number, vamos a ver qué es lo que tenemos. Tenemos aquí nuestro input de número telefónico. Yo no voy a poder escribir ninguna letra a excepción de la E porque en matemáticas la E hace referencia al exponencial.

[01:12] También, pues voy a poder poner un punto, voy a poder poner un + y puedo seguir así sucesivamente con caracteres especiales. Vamos a querer que siga el formato de 10 dígitos, solo como ejemplo. Lo que voy a hacer entonces es escribir la expresión regular aquí en pattern, me faltó la n, y le voy a poner que quiero que los dígitos que pueden ir del 0 al 9, se repitan 10 veces.

[01:43] Lo voy a guardar, le voy a poner también que este tipo requerido y vamos a probar. De nuevo estoy probando aquí con las letras y no me deja. Voy a ponerle entonces aquí los números. Y sin embargo, a pesar de que nosotros pusimos la expresión regular que sean solamente 10 dígitos, la verdad es que me deja escribir más números. ¿Esto por qué?

[02:06] Porque nosotros necesitamos definirle aquí en el tipo, en lugar de number que sea de tipo texto. Vamos a ver qué sucede. Lo cambio, esperamos a que recargue, voy a escribir los números, en este caso, ¿qué fue lo que me faltó? Me faltó decirle cuál era el maxlength. Este maxlength es de 10.

[02:25] Vamos a ponerle, aquí vamos a poder poner nuestros números y aquí ya me deja. Aquí ya solamente me deja ingresar 10 números. Si yo le cambió el tipo de nuevo al number, vamos a ver qué sucede. Voy de nuevo a escribir números y me dejó escribir más. ¿Esto por qué? Porque este maxlength sirve solamente para el tipo text.

[02:52] Entonces lo vamos a dejar como si fuera de tipo texto y con la expresión regular solamente deberíamos de poder obtener o poder ingresar números telefónicos. Muy bien, vamos entonces, a pesar de que nosotros estamos poniendo aquí nuestro pattern y nuestro maxlength, inclusive también pudiéramos ponerle un minlength de de 10 igual.

[03:26] Lo que vamos a hacer es, recuerdas en la parte de validaciones, tenemos un objeto en donde nosotros vamos agregando todos los mensajes de error además como las validaciones. ¿Entonces, qué es lo que vamos a hacer? Vamos a venir aquí. Y si recuerdas hemos estado trabajando con este data-tipo. Vamos entonces a agregárselo de nuevo: data-tipo = “numero”, que va a ser el de número telefónico.

[04:05] Lo voy a guardar. Vamos a venir aquí a nuestro archivo de validaciones. Aquí, en la parte de mensajes de error, vamos a agregar uno nuevo que va a ser número. Este iba a ser el tipo de input que nosotros estamos obteniendo aquí en este tipo de input. Muy bien. Lo que vamos a hacer entonces es crear un objeto y en este caso lo que podemos poner simplemente es un valueMissing.

[04:39] Vamos a poner: “Este campo no puede estar vacío.” Y en nuestro patternMismatch vamos a poner: “El formato requerido es xxxxxxxxxx”. Lo podemos poner esto en tren. Vamos a dejarlo así, 10 números. Vamos a guardar, vamos a regresar aquí. Vamos a ver número telefónico, salgo y dice: “Este campo no puede estar vacío”.

[05:35] Voy a ingresar un número. El formato requerido es de 10 números. Entonces, bueno, ¿qué pasa si yo le pongo aquí ya nuestros 10 números? Ya no tenemos el error. ¿Y ahora qué pasa por ejemplo si yo agrego aquí la E? A pesar de que al usuario sí le deja ingresar la letra E, ya cuando entra a la función esta de aquí de validar y verificar que el tipo de input, que es número, el patternMismatch está como false entonces se da cuenta de que no está siguiendo la expresión regular.

[06:13] Si recuerdas, nosotros le indicamos que solamente iban a ser los 10 dígitos y solamente pueden ser números. Por eso es que está apareciendo el error.

**Completando formulario II**

[00:00] Muy bien, entonces lo que vamos a querer aquí en nuestro input de dirección completa es al menos 10 caracteres y tal vez máximo 40. Pueden incluir números, pueden incluir texto o caracteres como tal letras, inclusive hasta nuestro símbolo de gato. Hay algunas direcciones donde le suelen poner el número.

[00:23] Vamos a ver entonces, vamos a tener que ponerle aquí nuestro pattern entre corchetes “[\s\S]”{10, 40} Vamos a ver qué sucede. Vamos a poner aquí Calle. Son solamente 5 caracteres, bueno, 6 con el espacio. Vamos a ponerle aquí principal, ya nos está entonces aquí marcando que no hay ningún tema.

[01:04] ¿Qué pasa si yo le pongo nuestro gato? Sigue funcionando, Calle principal número 12, por decir. Entonces así como nosotros lo agregamos para nuestra dirección completa también hay que hacerlo para la ciudad y el estado. En este caso lo que pudiéramos hacer tal vez para no tener que estar repitiendo en nuestro objeto, tener que agregar ciudad y estado, como van a seguir el mismo patrón, pudiéramos dejarlo con este tipo data-dirección.

[01:44] Sin embargo, pues bueno, lo que voy a hacer aquí es simplemente copiarlo. Más bien voy a copiar todo este required, lo voy a poner en nuestro input de ciudad, el data-tipo le voy a poner ciudad y aquí le voy a poner estado. Tal vez en tu país se llaman provincias. Déjalo a tu criterio, pero bueno, al menos en México, así se les conoce como estados.

[02:15] Lo voy a guardar, vamos entonces a la parte de nuestras validaciones. Yo voy a hacer lo mismo de aquí, simplemente lo voy a copiar y lo voy a pegar. Aquí el primero es ciudad y el segundo es estado. Le voy a poner aquí: “La ciudad debe contener entre 10 a 40 caracteres” y “El estado debe contener entre 10 a 40 caracteres”.

[02:48] Vamos a ver entonces. Nuestro formulario al parecer ya está completo. Si nosotros accedemos y salimos, pues bueno, ya nos está marcando errores en todo nuestro input. Tienes que checar, por ejemplo, aquí algo curioso. Cuando yo le di clic aquí al fecha de nacimiento y salí, me dice que debes de tener al menos 18 años de edad.

[03:11] Esto prácticamente es porque en nuestra función para verificar si es mayor de edad, está marcando un error, que en este caso es debes tener al menos 18 años de edad, pero bueno, pudiéramos manejarlo ahorita por simplicidad, pues bueno, vamos a dejarlo así. Entonces ya tenemos todos nuestros formularios. Si yo le pongo aquí ahora sí, mi información completa, me voy a poner aquí.

[03:55] Fecha de nacimiento, el número telefónico, la dirección, Calle principal, ciudad CDMX. Va a haber aquí algún tema. Posiblemente la parte de la ciudad pudiéramos bajarlo de caracteres, tal vez pudiéramos manejarlo con 4 y tal vez el máximo pudieran ser 30. Y lo mismo, tal vez con el estado, de 4 a 30.

[04:34] De nuevo mi nombre, mi correo electrónico, la contraseña, la fecha de nacimiento. Recuerda que tiene que ser mayor de 18 años, número telefónico, dirección completa, CDMX y el estado bueno, CDMX, igual en México así se conoce. Vamos a cambiarlo. Voy a ponerle Ecatepec, el estado de México, no tenemos mucha imaginación con el nombre de los estados. México.

[05:15] Okay, entonces ya le puedo dar clic en registrar. Y me comenta que el registro ya está completado. Entonces prácticamente ya tenemos cómo es la validación. Ya aprendimos cómo poder trabajar con todo este tipo de verificar que esté el patrón, verificar que no esté vacío, un mínimo, un máximo de caracteres.

[05:36] Y a pesar de que nosotros no nos estamos conectando con un servicio al cual estemos mandando la información, creo que pudiera ser muy buen ejercicio. Si ya sigues o si ya tienes conocimientos de back-end pudieras tratar tú de crear un end point para recibir esta información.

[06:00] Pues bueno. Ahora lo que vamos a ver en nuestro siguiente módulo es ir un poco más allá de solamente tener nuestro proyecto aquí en nuestra computadora, te voy a enseñar cómo utilizar dos servicios en los cuales tú vas a poder publicar tu página y vas a tener una URL única que vas a poder compartir, ya sea con tus amigos, ya sea en LinkedIn y, pues bueno, vamos entonces al siguiente módulo.

**Haga lo que hicimos en aula**

Practicar ayuda mucho en el aprendizaje de un nuevo concepto. Así siendo, es muy importante que implementes lo que fue presentado en esta clase.

**VER OPINIÓN DEL INSTRUCTOR**

Opinión del instructor

No dejes de quitar tus dudas antes de dar continuidad al curso. En el caso de que tengas alguna duda esa podrá ser contestada por nuestra comunidad en el foro de Alura Latam.

**Lo que aprendimos**

Lo que aprendimos en esta aula:

Como validar formularios

**Proyecto del aula anterior**

¿Comenzando en esta etapa? Aquí puedes descargar los archivos del proyecto que hemos avanzado hasta el aula anterior.

Descargue los archivos en Github o haga clic aquí para descargarlos directamente.

**Para saber más: Proveedores para sitios estáticos**

Aquí encontrarás los accesos directos a las plataformas que utilizaremos en esta sección:

Vercel <https://vercel.com/>

GitHub Pages <https://pages.github.com/>

**GitHub Pages**

[00:00] Muy bien, vamos entonces a ver cómo poder subir nuestro proyecto a un servicio para que quede disponible en internet. Si tú copiaste este proyecto o si tú lo descargaste solamente desde el punto zip o bien desde el repositorio, el link tal cual que aparece en GitHub, hay dos cosas que puedes hacer.

[00:22] El primero o más bien lo que tendrías que hacer es hacer fork. Prácticamente lo que es va a hacer una copia del código que tú quieres clonar en tu propia cuenta para que tú puedas tener justamente una copia de ese código para que tú lo puedas modificar, puedas hacer adecuaciones.

[00:41] En caso de que hayas descargado el punto zip, lo único que tendrías que hacer es aquí en GitHub crear un nuevo repositorio y vas a tener esta vista. Vas entonces a tener que copiar esta línea de aquí, el git remote add. Este de aquí. Vas a venir a Visual Studio Code en terminal, nueva terminal para que cargue.

[01:13] Tienes entonces que inicializar git. En este caso le das git init y le das enter. Esto va a inicializar tu proyecto git para que puedas ir llevando un control de versiones de tu código. En este caso, como ya lo tengo, lo único que voy a hacer es pegar lo que copié, el git remote add origin y toda esta URL.

[010:38] Bueno, le voy a dar enter. Y ahora sí, lo checo, con el comando git remote -v, aquí ya dice que tengo mi origin y hacia dónde está apuntando. En este caso es mi repositorio. Lo que voy a hacer entonces es aquí, ya cuando tengo todos mis cambios, quieres hacer git add -a para tener todos los cambios, un git commit -m para guardarlos.

[02:01] Puedes poner realmente el texto que tú quieras, lo que sea. En este caso, como no tengo ningún cambio, no me deja. Pues si tú tienes, pues bueno, te va a dejar. Y por último, un git push origin master. Este origin, si recuerdas es el nombre que nosotros le asignamos al momento de copiar esto de aquí. Es este mismo origin.

[02:23] Si tú le quisieras poner aquí algún otro nombre, lo puedes poner y solamente tendrás que cambiar aquí. Y ahora lo que estamos diciendo aquí con master es la rama que nosotros queremos subir. En este caso es master. Y de hecho, para que puedas ver, yo aquí con mis ramas, cuando bueno, cuando puse mi comando de git branch me muestra todas las ramas que yo tengo en este proyecto o versiones, por así decirlo.

[02:51] En este caso master ya es la rama que tiene todo el código que yo quiero subir a mi sitio web. Entonces, ya lo que tendría que hacer es un git push origin master. Le voy a dar enter, si tú no tienes configurado tu terminal, posiblemente te va a decir que inicies sesión, simplemente lo inicias con tu correo electrónico y tu contraseña y listo.

[03:15] En este caso me comenta que ya lo subió, vamos aquí a GitHub. Si yo recargo, aquí ya aparece mi todo mi código, con el cual yo he estado trabajando. Entonces lo que tendríamos que hacer prácticamente es hacer un cambio. Bueno, vamos a ver. Vamos entonces aquí nuestra sección de settings. Voy a checar aquí donde dice pages, que es páginas en inglés.

[03:44] Y aquí lo que me deja es poder seleccionar de qué ramas yo quiero tomar. Recuerdas la copia que te había comentado. En este caso es master. Me va a decir aquí en dónde se encuentra, le voy a dar route, le voy a dar clic en salvar. Okay, me dice aquí que él selecciona algún tema.

[04:05] Esto realmente puedes seleccionar el que quieras. No hay ningún tema con el tema. Bueno. Aquí ya lo seleccioné, tenemos ya este link y tenemos que acceder a /registro.html, justo porque así se llama nuestro archivo aquí, registro.html. Nos muestra un error. Este error dice que no puede encontrar las siguientes o los siguientes archivos.

[04:36] Y cómo prácticamente este de variables es donde se contienen todos los colores, el tamaño de las fuentes y en general, todo lo que le da la estructura a nuestra página está marcando un error y esto efectivamente no lo está encontrando. Entonces lo que tuve que hacer es aquí, en nuestro import, le quité guión bajo que antes estaban así.

[05:04] Se los quité y al mismo tiempo también le cambié el nombre a mis archivos, en este caso guión bajo reset, guión bajo variables. Entonces para que funcione deberías de tener también esos cambios. Vuelves a hacer un git add –a, un git commit –m y un git push origin master para que puedas obtener ahora sí el proyecto.

[05:31] Tienes que esperar a que guarde. Mira, voy a acceder a una página de incógnito, problemas de caché, le voy a dar aquí enter y aquí, ahora sí ya tengo mis estilos. Entonces, como puedes verificar aquí pues bueno, sigo teniendo todo mi formulario y ahora sí en una URL en la cual podrías compartir con tus amigos, ponerlo en LinkedIn, poner un link directo en tu portafolio, en tu blog, etcétera.

[06:00] Entonces de esta manera es como tú puedes ir trabajando o subiendo tus proyectos que vas realizando aquí en Alura. Esta fue una primera manera, una primera opción en la cual tú puedes poner tus proyectos personales. Vamos a ver la siguiente.

Git init

Git status

git remote add origin <https://github.com/Shela95/JS-en-la-web-validacion-de-formularios-y-HTML5.git>

<https://shela95.github.io/JS-en-la-web-validacion-de-formularios-y-HTML5/>

**Vercel**

[00:00] Muy bien. Vamos a continuar entonces el otro servicio que nosotros vamos a utilizar se llama Vercel, de hecho, es muy conocido en el mundo de React, porque es la empresa que está por detrás de un proyecto llamado Next.js que justamente utiliza React y si en algún futuro te interesa, puedes aprenderlo.

[00:18] En fin, lo que voy a hacer aquí es login o sign up, dependiendo. En este caso, como ya tengo una cuenta, voy a hacer login, voy a seleccionar, voy a continuar con GitHub. Puede que a ti durante todo este proceso, si tú no tienes una cuenta, te pida permisos de conexión con tu cuenta de GitHub y simplemente lo autorices.

[00:40] Lo que voy a hacer entonces es crear un nuevo proyecto. Como te comentaba, tiene para poder subir aplicaciones con Next.js, uno muy parecido que es Gatsby, Nuxt.js que está con View. Pero bueno, lo que voy a hacer aquí es seleccionar el repositorio que yo quiero subir a esta plataforma, en este caso es html-validaciones.

[01:05] Tú buscas solamente tu repositorio y listo. Le voy a dar clic aquí en importar. Me pregunta si quiero crear un equipo, le voy a decir que en este caso no, que lo salte. Aquí en framework preset lo que voy a hacer es seleccionar otro. Y en build y output settings lo voy a deseleccionar, le voy a dar clic entonces aquí en deploy.

[01:30] Voy a esperar a que primero descargue el código que tenemos en GitHub, después siga con el siguiente paso, construya la aplicación y lo suba ya una URL o producción. Vamos a esperar a que termine todo este proceso. Aquí me comenta que ya está listo el proyecto, le voy a dar clic en go to dashboard.

[01:51] Okay, aquí ya me aparece un botón para poderlo visitar, le voy a dar clic y voy a ver aquí mi página. Igual, el mismo tema que teníamos con GitHub pages. Me dice que no existe. ¿Por qué? Porque mi archivo es registro-html. Le voy a dar enter y listo, aquí ya tenemos nuestro proyecto.

[02:14] En caso de que tú quieras que no tengan que escribir registro y que aparezca luego, luego, lo que tendrías que hacer es el lugar que se llame registro, simplemente escribes index.html y posiblemente lo has visto o aprendido en algún otro curso, pero cada vez que un servidor busca dentro de todos los archivos por defecto, va a buscar el archivo index.html, y es el que va a mostrar cuando nuestra URL no tiene nada después que la base, es decir esto de aquí.

[02:49] Muy bien. Entonces, pues bueno, y tienes tu proyecto, tanto en GitHub pages como en Vercel. Hay algunas otras opciones como pueden ser Netlify o esta empresa de aquí, también es prácticamente lo mismo.

[03:10] Entonces, puedes ir probando las tres plataformas, me parece que incluso hay más, estas al menos son las que yo he utilizado y creo que son de las más comunes, al menos para este tipo de proyectos que no tienen la responsabilidad de manejar rutas del lado del back end o guardar la información, entonces me parece que esta es una muy buena opción. Entonces, vamos a dejar hasta aquí el video.