

PRUEBA PRÁCTICA

Web de Envío de Emails

NOMBRE ALUMNO:

CONTENIDOS: bloque 2, bloque 3, bloque 4, bloque 5

Enviar mails

Email:

El Campo email es obligatorio

Asunto:

El Campo asunto es obligatorio

Mensaje:



El Campo mensaje es obligatorio

➤ Enviar

🗑 Reset

Partes a evaluar (Pista: ****Pasos a seguir****):

- **Primeros pasos:**
 - Cuando el DOM este cargado
 - Crea el elemento inputEmail, inputAsunto e inputMensaje
- **Eventos ideales para validar un formulario:**
 - Usa el evento blur para comprobar el contenido del input
- **Crea una función para validar un formulario:**
 - Crea una función para comprobar todos los inputs, y que esta sea llamada con los diferentes eventos.
- **Detecta cuando un campo esta vacío:**
 - Crea un condicional dentro de la función validar para comprobar si el input esta vacío y usa el metodo .trim() para evitar los espacios en blanco.
- **Crea una alerta de error de validación:**
 - Ahora tendremos que crear una función alertaError, a la cual llamaremos si los inputs están vacíos.
 - Crea un nuevo elemento llamado error, para poder mostrar el error por consola.
- **Añade la alerta al HTML:**
 - Añade el elemento error al html y muéstralo, ya que el usuario no suele ver la consola.

-
- Inserta el error en el formulario, para ello crea el elemento formulario y añadelo como hijo.
 - añade el color de fondo rojo al mensaje.
- **Valida cada campo:**
 - Ahora tienes que mostrar un mensaje diferente dependiendo del campo que has dejado vacío. "El campo _____ es obligatorio".
 - Para ello necesitas una variable con el id de cada elemento, su contenido se le pasará a la función que teníamos ya creada al llamarla.
- **Muestra alertas junto al campo:**
 - Anteriormente se añadían los mensajes al final del formulario, ahora tienes que añadirlo debajo de cada input o del input correspondiente.
 - Para esto debes pasarle a la función además del parametro id, lugar donde quieres añadirlo usando TRANSVERSING DOM (padres, hijos, etc.)
- **Prevenir que se generen multiples alertas:**
 - Crea un elemento alerta con querySelector y comprueba si existe.
 - Si existe elimina este elemento antes de volver a escribirlo, dentro de la función que ya teníamos creada.
- **Ocultar alertas si pasa validación:**
 - Pero si tenemos las alertas y rellenamos los inputs se mantendrían por ello, tenemos que eliminar la alerta en caso de que el input contenga texto.
 - Te puedes ayudar de un return para en caso de entrar al if, acabar la función y en caso de no entrar llamas a una nueva función limpiarAlerta();
 - Hay que pasarle la referencia del elemento de la alerta para saber cual eliminar.

- **Validar email con una expresión regular:**

- Pero en el input email introducimos lo que sea y no nos da error de formato o expresión regular, por lo tanto tenemos que:
- Crear funcion validarEmail
- Dentro poner esto:
- `var validEmail = /^w+([_+]?w+)*@w+([_]?w+)*(\.w{2,10})+$/;`
- Esto se conoce como expresión regular y es necesario para validar cualquier patrón. Emails, codigo postal, dni, etc...
- La función tiene el argumento email, el cual debe ser validado con el metodo .test(), investiga un poco en w3school.
- Crea un condicional para en caso de que no se introduzca el mail correctamente y el id es email entonces muestre el email no es valido en el alert

- **Crea el objeto principal para validar y sincronizar datos:**

- Crear un objeto con 3 propiedades: email, asunto, contenido. Vacío al principio.
- Usando el dom introduce valores en ese objeto despues de pasar la validación del formulario. Y usa trim(), para quitar los espacios al final y al principio.
- usa tambien el metodo toLowerCase().
- Todo esto lo puedes meter dentro de la función validar ya que, le entra el evento con la información del input, entonces sería objeto.propiedad=valor,
- para obtener tanto la propiedad como el valor usa e.target.algo
- Crear una función comprobarObjeto, para comprobar que el objeto no esta tiene algun campo vacío , usa .values y .includes metodos, dentro de esta función. A esta función la llamas al final de la función validar.

- **Habilitar o deshabilitar el botón de enviar:**

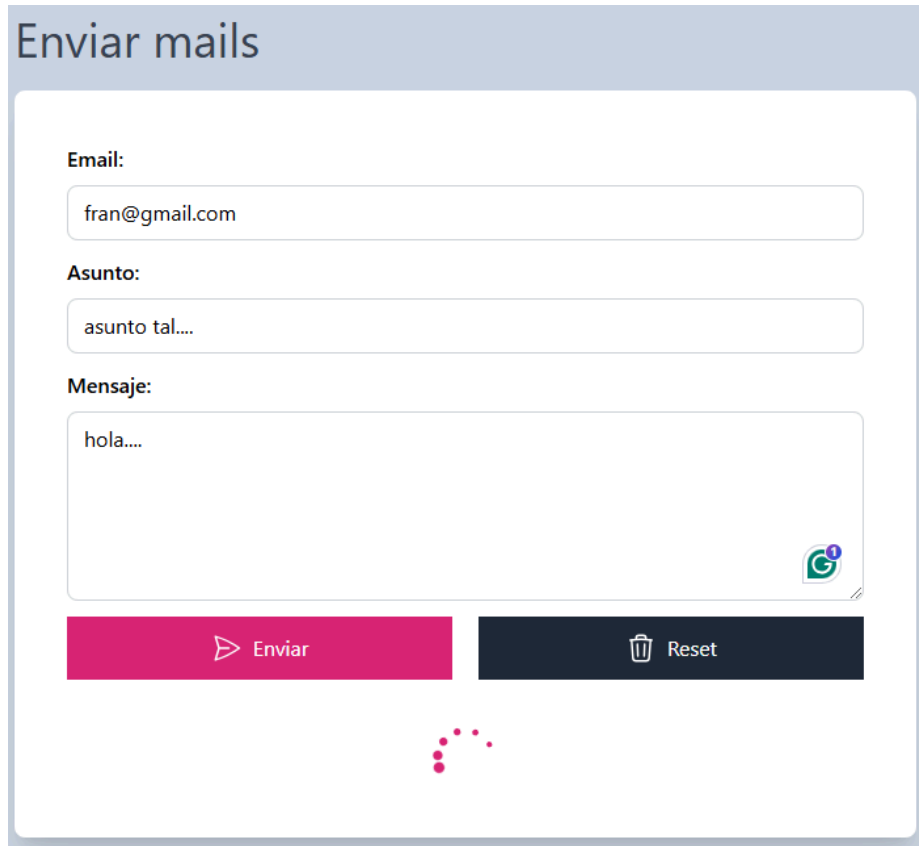
- Si al menos uno esta vacio nos devuelve un true el metodo .includes(' ').
- creamos el elemento botonSubmit
- Este boton tiene 2 propiedades la de opacity y la de disabled, esto tenemos que cambiarlo dinamicamente, es decir , si detectamos que la respuesta del metodo de antes es FALSE , entonces el boton se activa(boton.disabled = false) y se pone visible(elimnando la clase opacity-50 del html ==> classlist.remove).
- Pero claro, ahora si despues de activarse, pruebas a borrar un campo, no se volvería a desactivar. Esto hay que arreglarlo:
- Esto se debe a que el objeto donde escribimos las propiedades aunque borremos los inputs ya contiene los valores, entonces en el condicional de la funcion validar() debemos poner que si esta vacío el campo le escriba el valor vacio al objeto y despues llame a la funcion comprobarObjeto (esta función siempre tiene que ser comprobada)
- despues volver a añadir la opacity y disabled en la funcion comprobarObjeto.
- para hacer mas rapido el codigo cambiar el evento blur por input, usados al principio del codigo

- **Resetear el formulario:**

- Crea el elemento botonFormulario
- Creamos su evenListeners con el evento click
- usamos e.preventDefault(), para prevenir su accion por defecto
- formulario.reset() sería una forma, pero seguimos teniendo el objeto con contenido, por lo tanto hay que llamar a las diferentes propiedades del objeto creado e introducirle un valor vacío.
- Despues llamamos a la funcion comprobarObjeto() de nuevo

- **Añadiendo un Spinner debajo de los botones del formulario:**

- Crear funcion enviarEmail
- Llamar a la función enviarEmail cuando escuchemos el evento submit en el elemento formulario



The screenshot shows a web form titled "Enviar mails" with a light blue header. Below the header, there are three input fields: "Email:" with the value "fran@gmail.com", "Asunto:" with the value "asunto tal...", and "Mensaje:" with the value "hola...". At the bottom of the form, there are two buttons: a pink button labeled "Enviar" with a right-pointing triangle icon, and a dark blue button labeled "Reset" with a trash can icon. Below the buttons, there is a small pink spinner icon consisting of several dots arranged in a semi-circle.

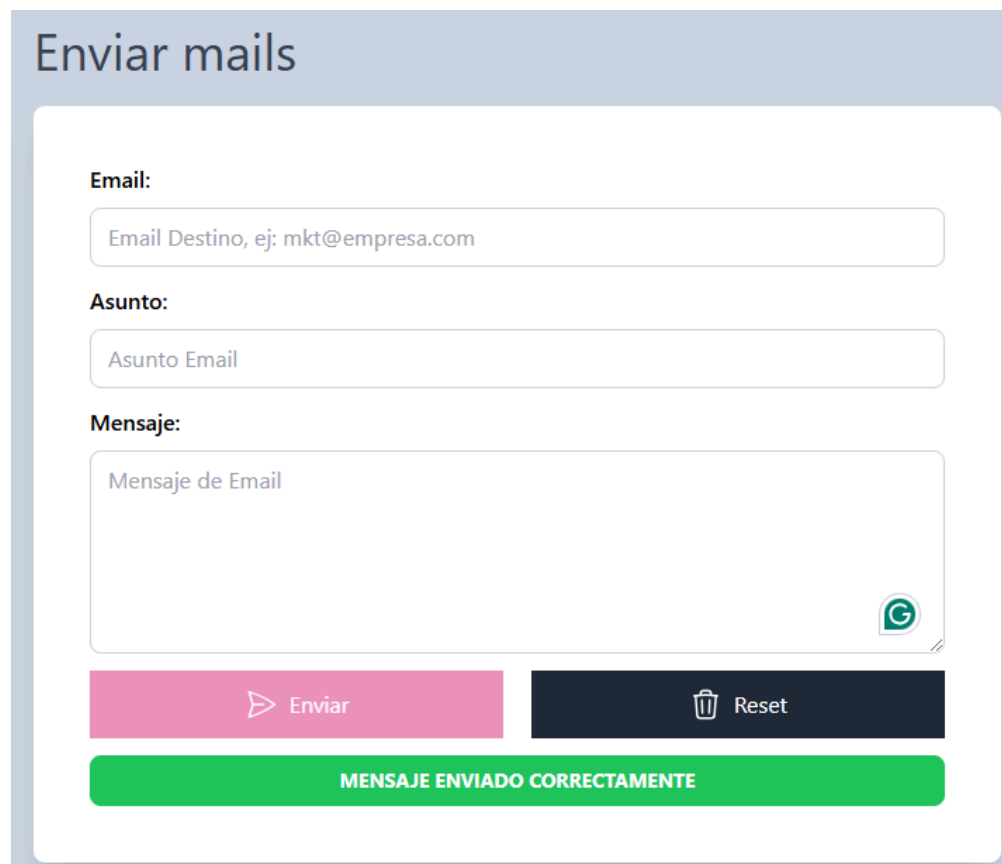
- **Mostrar el Spinner cuando se envía el email:**

- Crear elemento Spinner
- Eliminar su clase hidden y añadir la clase flex dentro de la función enviarEmail. ESTA PARTE MUESTRA EL SPINNER
- Ahora con la función setTimeout hacemos que solo se muestre 3 segundos el spinner para ello: ESTA PARTE ESCONDE EL SPINNER
- Dentro añadimos el código del punto anterior pero al contrario, es decir el que elimina la clase flex y añade la hidden cuando pasan 3 segundos. setTimeout es fácil de usar, mirar un ejemplo hecho en internet.

- Crear función `resetearFormulario()`, este código ya lo teníamos creado, pero esta función la vamos a llamar cuando pulsamos enviar o cuando pulsamos reset, es decir desde dos sitios diferentes.

- **Mostrar alerta de éxito:**

- Crea un elemento alerta y añádelo al final del formulario cuando acabe de mostrarse tu spinner
- Esta alerta debe mostrar mensaje enviado correctamente durante 3 segundos solamente.



The screenshot shows a web form titled "Enviar mails" with a light blue header. The form contains three input fields: "Email:" with placeholder text "Email Destino, ej: mkt@empresa.com", "Asunto:" with placeholder text "Asunto Email", and "Mensaje:" with placeholder text "Mensaje de Email". Below the inputs are two buttons: a pink "Enviar" button with a right-pointing arrow icon and a dark blue "Reset" button with a trash can icon. At the bottom of the form is a green banner with the text "MENSAJE ENVIADO CORRECTAMENTE".

○