Germán Martínez Cagigas

DEC

Documentando el DOM

Indice

Contenido

Introducción

Explicación de métodos y funciones utilizados en ejercicios DOM.

Ejercicio

Explicaciones DOM.

<button type="button" onclick="validarFormulario()">Verificar</button>

botón al que le hemos añadido el atributo onclick, el cual al presionar dicho boton lanzará una funcion que se llama validarFormulario.

lo primero que se hará al lanzar esta expresión será recoger el valor del campo de cada input que tenga nuestro formulario y lo guardaremos en una variable.

var nombre = document.getElementById("nombre");

capturo en una variable llamada nombre el valor introducido en el input con id nombre, en este caso no capturé el valor introducido en el campo porque quería que pasaseel input (entero por así decirlo) en vez de el valor puesto que iba a usar checkvalidity.

```
function refactorIf(variable, mensaje) {
      if (!variable.checkValidity()) {
        fallosArray.push(mensaje);
      }
    }
```

puesto que iba a usar un checkvalidity que valida los requisitos introducidos a la hora de crear el input como required o pattern y devuelve un true o false,

puesto que tengo varios input que van a pasar por este proceso, refactoricé para tener que escribirlo una vez pasando como parametro el input / field y usar un método ya creado por defecto de checkvalidity que devuelve un true o false

```
 <input
      type="text"
      id="direccion"
```

```
      name="direccion"
      required
      placeholder="Escribe la dirección"
      pattern="^[A-ZÑ]{1}[A-Z Ñ0-9ºª/:]*$"
    />
```

en caso de que no se hubiese escrito un numero de primer caracter en el campo de direccion el checkvalidity hubiera devuelto un false y se hubiese añadido el mensaje que pase como parametro:

al array de fallos.

```
function refactorP(pattern, variable, mensaje) {
      if (!pattern.test(variable)) {
        fallosArray.push(mensaje);
      }
    }
```

esta función sigue el mismo ejemplo de la de arriba pero en vez de trabajar sobre el campo con los requisitos que lo definimos como el required o patter, en esta funcion se trabaja sobre el dato. Introducido el .value si el valor introducido no cumple el patron definido, probado a traves del metodo test sobre la variabloe devolvía un false y añadía el fallo de mensaje capturado como parametro al array.

```
 if (fallosArray.length == 0) {
      bontonEnviar.removeAttribute("disabled");
      bontonEnviar.style.opacity = "1";

      fallosVerificacion.innerHTML = "";
    } else {
      bontonEnviar.setAttribute("disabled", "true");
      fallosVerificacion.innerHTML = letra + fallosArray.join("<br>* ");
    }
```

```
<button type="submit" id="enviar" style="opacity: 0.5" disabled>
    Enviar
</button>
<br />

<div id="fallosVerificacion" style="color: red"></div>
```

el botón de enviar estaba con un atributo disabled para que no se pudiese usar y una opacidad por darle estilo.

tras todas las comprobaciones si nuestro array de fallos esta vacío quitamos el atributo disabled a través de removeAtribuite al boton envía (capturado el boton a través del get Element). Se habilitaba y se podia enviar.

Si no al valor capturado del div a través del innerHTML le metemos a machete que escriba el array y junte cada hueco del array con un salto de linea y que empiece con el asterisco.

```
nombreInput.addEventListener("blur", () => validarCampo(nombreInput, s1, patternMayu));
```

método que se aplica sobre una variable, en este caso es un blur que hace que al perder el foco el campo pase algo, hay otro como change y onclick (en event) que no hemos usado aun.

pues a esta variable le añadimos un evento de cambio de foco el cual llamará a una función anónima que lo que sirve es para llamar directamente a la función pasando los parametros de la funcionvalidarCampo

además el eventlistenner en conjunto con función anónima hace un encapsulamiento de las variables que recibe, esto quiere decir que si tengo un generador de formularios y cada campo que se genera se llama input, esta variable distinguirá los requisitos asignados al primer input de los requisitos designados a los siguientes inputs, hace un encapsulamientos de los parámetros recibidos.

```
function validarCampo(input, mensaje, pattern) {

    input.value = input.value.toUpperCase();
      if (pattern.test(input.value)) {
        input.classList.remove("invalid");
```

```
        } else {
            input.classList.add("invalid");
            input.value = "";
            input.placeholder = mensaje;
        }
    }
```

Esta función hace lo que debe pero habría ido mejor haber capturado los datos de los campos como value para no hacerlo dentro de la función. (El .value digo).

La primera línea da al valor del campo introducido que todo sea mayusculas. con el toUpperCase.

Luego comprueba la variable en mayúsculas con el pattern introducido, si sale false se le añade una clase a ese campo, la cual esa clase sin mas pone el recuadro en rojo.

SI sale true, que no se le deje con la marca roja en caso de que la tuviese.

```
function validarVia(input) {
        if (input.value == "") {
        input.classList.add("rojo");
        } else {
        input.classList.remove("rojo");
        }
    }
```

es lo mismo que el anterior pero la manera de conseguirlo era distinta puesto que era una checkbox.

```
function addForm()

if (inputA.trim() == "" || patternA.trim() == "") {
    alert("Hay que rellenar los dos campos no pueden estar vacios");
    return;
  }
```

elimino espacios vacios de la variable y veo que no este validarCampo

const isRequired = document.getElementById("check").checked;

capturo true o falso si el check box esta seleccionado

const inputI = document.createElement('input');

metodo para crear elementos en este caso un campo input.

```
inputI.type = "text";
  inputI.value = "";
  inputI.placeholder = "Pattern: " + "\\" + patternA + "\\";
```

Asignación de atributos a este campo creado, como que sea de tipo texto, que tenga un valor interno vacío por defecto y que tenga de place holder pues el patrón introducido que se capturo en una variable.

const botonAbajo = document.createElement(`button`);

Lo mismo pero creo un botón.

```
botonEliminar.setAttribute("onclick","remove(this)");
  botonEliminar.textContent = "Eliminar";
  botonEliminar.type = `button`;
```

asignación de atributos, el primero hace que al presionar (onclick) se active la función remove, el (this) hace referencia que se aplique solo sobre ese botón, es una manera de encapsular la variable y hacerla distinta del resto puesto que se pueden crear tantos botones eliminar como se quiera.

El textcontent es escribir sobre la variable el contenido que quieres que muestre en este caso se verá un boton eliminar

si fuese una p el objeto creado, el text content sera el parrafo que veamos dentro de la p.

```
if (isRequired) {
    inputI.required = true;
  }
```

añadimos al campo en caso de sea true esta variable el atributo required

```
divContainer.appendChild(labelL)
```

Hacemos que cuelgue algo, es como crear un hijo del padre, en este caso a nuestra variable divContainer que es un div, hacemos que dentro de el tenga un labelL

```
<div>
  <Label>Hola</label>
</div>
```

Además el label que creamos tendria un textcontent de hola.

```
forma.appendChild(divContainer);
```

moveríamos el div anterior con sus hijos a que cuelgue de un form por lo cual el div pasaría a estar dentro de un form.

```
const mensaje = `El pattern para validar es ${patternA}`;
```

${} forma de mostrar valores de variables dentro de un texto comentado en `` con "" no serviria

```
function remove(button) {
        const taskItem = button.parentNode;
        const form = document.getElementById('lista-form');
```

```
        form.removeChild(taskItem);
    }
```

Parentnode coge un escalon por encima, en el ejemplo anterior label parentnode seria el divContaner por tanto en este ejemplo el taskItem habria sido el divContainer

en este ejemplo del boton también sería el div

como el botón tenía el this a ese botón que hace referencia el this borramos del formulario el bloque donde esta contenido la estructura seria la siguiente.

```
<form>
  <div>
    <boton>Eliminar</label>
  </div>

<form>
```

taskItem sería el div

form el formulario

borro del formulario el taskitem(div)

```
function arriba(button) {
    var elemento = button.parentElement;
    const form = document.getElementById('lista-form');
    var indice = Array.prototype.indexOf.call(form.children, elemento);

    if (indice > 0) {
     form.insertBefore(elemento, form.children[indice - 1]);
    }
  }
```

esta función realiza un array con todos los hijos del padre, coje el index del elemento contenido en el padre mediante el metodo indexOf y una vez obtenido el indice pues en este caso lo mueve una posición hacia arriba, también podría borrarle o moverle hacia abajo.

Bibliografía

https://auladecroly.com/mod/assign/view.php?id=33627