Programación

Formularios

Decimosexta semana

Febrero 2022

Cruz García, Iago

Introducción

Este último tema trabajamos la creación de eventos, en concreto en los ejercicios, añadiendo funciones a botones para dotar de dinamismo a nuestras páginas web. Esto nos obliga a cambiar el paradigma de diseño de nuestro código ligeramente, pues ya no hay una ejecución secuencial, si no que depende de las acciones que realiza el usuario.

Y de las funciones que más se realizan en entornos web es el envío, gestión o consulta de información, muchas veces mediante formularios. Como ya visteis en Lenguajes de Marcas cómo se crean y qué hacen, en este tema veremos cómo personalizarlos y utilizar los eventos para restringir la información de los usuarios y de paso veremos cómo controlar la entrada de datos en elementos como Textarea o Input.

Recomendación: A partir de ahora es necesario que a la hora de realizar ejercicios nos acostumbremos a buscar información en la documentación oficial. En caso de no poder resolver las dudas con la documentación ofrecida, el siguiente paso será preguntar las dudas en el foro de clase.

Documentación oficial de JavaScript

Foro de la asignatura

Eventos

Lista de disparadores

Formularios

Para comenzar, debemos conocer algunas de las etiquetas más utilizadas en formularios. Tenéis disponible en Lenguajes de Marcas la teoría sobre estos y en este tema lo que haremos será aplicar la teoría de eventos a diferentes campos de ellos y algunas estructuras que nos resultarán útiles en el futuro.

Input

Una de las tareas más realizadas en los formularios es la de rellenar campos de texto simples: nombre, apellidos, correo, contraseña… y si bien el propio HTML permite muchas opciones como por ejemplo que sea obligatorio el campo o que tenga que seguir un patrón determinado, podremos añadir funcionalidades extra en forma de eventos. Vamos a ver un ejemplo:

index.html

```
<html>
<head>
</head>
<body>
	<form name="DAW" action="#" method="post">
	<label for="user">Usuario: </label>
	<input type="text" name="user" id="user" value="" required>
	<br>
	<br>
	<label for="pass">Contraseña: </label>
	<input type="password" name="pass" id="pass" value="" required>
```

```
        </form>
</body>
<footer>
        <script src="main.js"></script>
</footer>
</html>
```

main.js

```
function __main__() {
        var user = document.getElementById("user");
        var pass = document.getElementById("pass");
        pass.addEventListener("keyup", (evt) => {
        comprobarPass(evt, user, pass);
        });

}

function comprobarPass(evt, user, pass) {
        var minusculas = /[a-z]/;
        var mayusculas = /[A-Z]/;
        var numeros = /[0-9]/;
        if (minusculas.test(pass.value)) {
        console.log("Tiene minusculas");
        }
        if (mayusculas.test(pass.value)) {
        console.log("Tiene mayusculas");
        }
        if (numeros.test(pass.value)) {
        console.log("Tiene numero");
        }
}

__main__();
```

En este ejemplo, recogemos el cuadro de texto de la contraseña "pass" y mediante el evento de "keyup", es decir, cuando la tecla termina de pulsarse, se comprueba si el valor del cuadro de texto tiene alguna minúscula, si tiene alguna mayúscula o algún número. Si es así, aparecerá en la consola el mensaje correspondiente. Para la comprobación de patrones en textos en JS, utilizaremos expresiones regulares.

Esto es solo un ejemplo de lo que podemos hacer con los campos de texto. Esto se puede trasladar a Textarea también.

Select

La modificación de los valores de un seleccionador es algo muy común en cualquier formulario. En el ejemplo siguiente vemos cómo modificar el campo de comunidad autónoma cambia las provincias que podemos seleccionar.

index.html

```
<html>

<head>
</head>

<body>
        <form name="DAW" action="#" method="post">
        <label for="comunidad"> Comunidad autónoma: </label>
        <select name="comunidad" id="comunidad">
        <option value="Galicia">Galicia</option>
        <option value="Asturias">Asturias</option>
        <option value="Cantabria" selected>Cantabria</option>
```

```
        <option value="Leon" >Castilla y León</option>
        </select>

        <label for="provincia">Provincia</label>
        <select name="provincia" id="provincia">
        </select>
        </form>
</body>
<footer>
        <script src="main.js"></script>
</footer>

</html>
```

main.js

```
function __main__() {
        var comunidad = document.getElementById("comunidad");
        var provincia = document.getElementById("provincia");
        comunidad.addEventListener("change", (evt) => {
        console.log(comunidad.value);
        cambiarComunidad(provincia, comunidad);
        });

}

function cambiarComunidad(provincia, comunidad) {
        var c_galicia = ["Pontevedra", "A Coruña", "Lugo", "Ourense"];
        var c_cantabria = ["Cantabria"];
        var c_asturias = ["Asturias"];
        var c_leon = ["Leon", "Zamora", "Palencia", "Burgos", "Valladolid",
"Soria", "Segovia", "Salamanca", "Ávila"];
        var comunidad_cambiar;
        console.log(comunidad.value);
        switch (comunidad.value) {
        case "Galicia":
        comunidad_cambiar = c_galicia;
        break;
```

```
        case "Asturias":
        comunidad_cambiar = c_asturias;
        break;
        case "Cantabria":
        comunidad_cambiar = c_cantabria;
        break;
        case "Leon":
        comunidad_cambiar = c_leon;
        break;
        default:
        break;
        }
        purgarProvincias(provincia);
        mostrarProvincias(provincia, comunidad_cambiar);
}

function purgarProvincias(provincia) {
        while (provincia.firstChild) {
        provincia.removeChild(provincia.lastChild);
        }
}

function mostrarProvincias(provincia, comunidad) {
        var opcion;
        for (var i = 0; i < comunidad.length; i++) {
        opcion = new Option(comunidad[i], i);
        provincia.appendChild(opcion);
        }
}

__main__();
```

Recomiendo copiar este ejemplo para visualizarlo mejor, pero lo principal es entender que dependiendo del elemento "Comunidad" cambiamos el

elemento “Provincia” en tiempo de ejecución para reflejar las opciones disponibles.

Otros elementos

Estos son los dos grandes elementos (Input y Select) que suelen ser modificados o comprobados. Otros elementos que a veces necesitarán un control desde JavaScript son los Radio Button o los Checkbox. A continuación os presento un ejemplo con Radio Button.

index.html

```
<html>

<head>
</head>

<body>
	<form name="DAW" action="#" method="post">
	<fieldset id="comunidad">
	<legend>Comunidad autónoma</legend>
	<label for="Galicia">Galicia<input type="radio" name="Comunidad" class="radio" value="Galicia"></label>
	<label for="Asturias">Asturias<input type="radio" name="Comunidad" class="radio" value="Asturias"></label>
	<br>
	<label for="Cantabria">Cantabria<input type="radio" name="Comunidad" class="radio" value="Cantabria"></label>
	<label for="Leon">Castilla y León<input type="radio" name="Comunidad" class="radio" value="Leon"></label>
	</fieldset>
```

```
        <fieldset id="provincia">
        <legend>Provincia</legend>
        </fieldset>
        </form>
</body>
<footer>
        <script src="main.js"></script>
</footer>
</html>
```

main.js

```
function __main__() {
        var comunidades = document.getElementsByClassName("radio");
        var provincia = document.getElementById("provincia");
        for (var i = 0; i < comunidades.length; i++) {
        comunidades[i].addEventListener("click", (evt) => {
        cambiarComunidad(provincia, evt.target);
        });
        }

}

function cambiarComunidad(provincia, comunidad) {
        var c_galicia = ["Pontevedra", "A Coruña", "Lugo", "Ourense"];
        var c_cantabria = ["Cantabria"];
        var c_asturias = ["Asturias"];
        var c_leon = ["Leon", "Zamora", "Palencia", "Burgos", "Valladolid",
"Soria", "Segovia", "Salamanca", "Ávila"];
        var comunidad_cambiar;
        console.log(comunidad.value);
        switch (comunidad.value) {
        case "Galicia":
        comunidad_cambiar = c_galicia;
        break;
        case "Asturias":
        comunidad_cambiar = c_asturias;
        break;
```

```
        case "Cantabria":
        comunidad_cambiar = c_cantabria;
        break;
        case "Leon":
        comunidad_cambiar = c_leon;
        break;
        default:
        break;
        }
        purgarProvincias(provincia);
        mostrarProvincias(provincia, comunidad_cambiar);
}
function purgarProvincias(provincia) {
        while (provincia.firstChild) {
        provincia.removeChild(provincia.lastChild);
        }
}
function mostrarProvincias(provincia, comunidad) {
        var etiqueta, opcion, salto;

        for (var i = 0; i < comunidad.length; i++) {

        etiqueta = document.createElement("Label");
        etiqueta.setAttribute("for", comunidad[i]);
        etiqueta.innerHTML = comunidad[i];
        opcion = document.createElement("input");
        opcion.setAttribute("type", "radio");

        opcion.setAttribute("value", comunidad[i]);
        opcion.setAttribute("name", "provincia");

        etiqueta.appendChild(opcion);
        provincia.appendChild(etiqueta);
        salto = document.createElement("br");
        provincia.appendChild(salto);
        }
}
__main__();
```

Como observamos, el código es muy parecido al del selector. Los cambios más significativos son:

- Como se recogen múltiples elementos: Al ser más de un elemento radio, tendremos que utilizar una clase para aunarlos todos en el mismo array o vector. Así mismo, para aplicar el evento a cada uno de ellos utilizaremos un bucle. Además, observamos que se envía como parámetro "evt.target" para no depender del elemento individual, si no con el que se haya interactuado.
- A la hora de añadir escoger las provincias no hay diferencia, ya que se trata de un método aislado.
- Sin embargo, cuando se crean los Radio Button cambia radicalmente. Ahora hay que crear también la etiqueta "Label" y esta añadirla al espacio de las provincias, dando como resultado la relación "fieldset"->"label"->"input". Además añado un salto de línea "br" para mejor legibilidad.

Probad a modificar los ejemplos para probar los diferentes comportamientos de cada elemento.