Desarrollo Web en Entorno Cliente

VERÓNICA BONIS MARTÍN
MARIA CARMEN CORREA HERAS
ÁNGEL SÁNCHEZ-SIERRA CRUZ
JOSÉ MARÍA TENREIRO EIRANOVA

REPOSITORIO GITHUB
https://github.com/ITTDAW/AE-3.-Formularios.git

Actividad 3. Formularios

Requerimiento 1

En esta actividad debes desarrollar un formulario completo, sin envío al servidor, que sirva para dar de alta un pedido de una pizza dentro de una web de una pizzería.

Los campos del formulario son los siguientes:

- Nombre
- Dirección
- Teléfono
- Email
- Un radio button con el tamaño de la pizza, pudiendo ser pequeña, mediana o grande
- 4 Checkbox con los diferentes ingredientes de la pizza
- Un botón de procesar el pedido

Todos los campos tienen que estar rellenos de tipo texto deben de estar rellenos para que sean válidos, además debe de elegir obligatoriamente un tamaño de la pizza y al menos un ingrediente para ella.

Requerimiento 2

El campo teléfono y email deben de tener un formato adecuado, deben de hacerse las validaciones oportunas con JavaScript (Expresiones regulares).

Además, el nombre no puede empezar por minúsculas.

Requerimiento 3

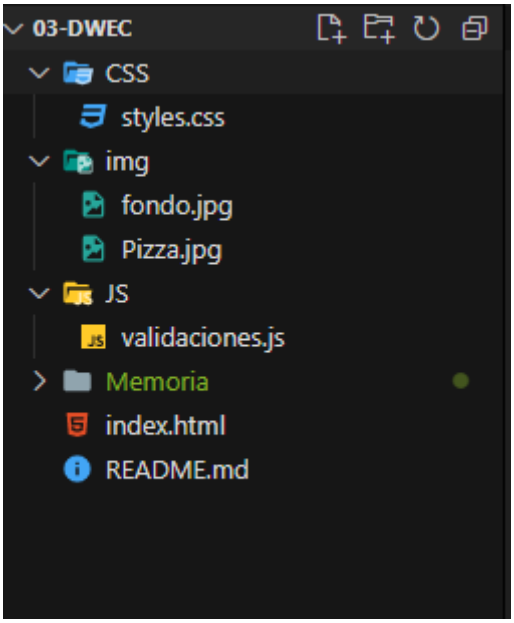
Al pulsar el botón de procesar el pedido, se mostrará el precio total del pedido calculándolo en base a los siguientes parámetros:

- 5€ para la pizza pequeña
- 10€ para la pizza mediana
- 15€ para la pizza grande
- Cada ingrediente elegido tendrá un valor de 1€

ORGANIZACIÓN DE DIRECTORIOS

La organización de los directorios en nuestro proyecto del formulario de pedido de una pizzería será el siguiente:

Actividad 3. Formularios



- Tenemos dos carpetas llamadas CSS e img en donde almacenaremos nuestra hoja de estilos y nuestras imágenes respectivamente.
- Después tenemos una carpeta llamada JS que es donde almacenamos nuestro archivo Javascript para las validaciones.
- En la raíz del proyecto se encuentra nuestro archivo HTML index. En el index implementamos nuestro formulario de la pizzería con los requisitos exigidos en el enunciado.
- Hay dos cuestiones clave siempre en todo archivo HTML:

1. La primera, hay que intentar que nuestro código HTML sea limpio, es decir, que no encontremos rastro de código JavaScript en éste. Esto debe de hacerse así, porque el cliente podría copiar nuestro código desde el navegador. Es cierto, que aun con esta medida de seguridad sigue siendo posible copiarlo, pero al menos se añade un nivel de dificultad.
2. La segunda cuestión clave es que las validaciones siempre deben de hacerse tanto en el cliente como en el servidor. En la parte del servidor es donde el cliente no va a poder alterar nada de nuestro código. Es por eso que con unas validaciones solo en el cliente, nos exponemos a un problema de seguridad.

REQUERIMIENTO 1

En la primera parte del formulario nos encontramos con los inputs de tipo **text**. Para poder llamarlos desde el archivo JS, nos vale definirlos con un **id**.

Si fuéramos a trabajar con servidor, hay que definir también un **name**.

Actividad 3. Formularios

```
<form id="formulario"   method="get">

    <h2>¡Ordena tu Pizza!</h2>


        <fieldset class="border p-5">
            <!-- Cuando son input text nos vale con localizarlos con un id para despues llamarlos desde el archivo JS -->
            <div class="mb-3">
                <label for="nombre" class="form-label">Nombre</label>
                <input type="text" id="nombre" placeholder="Escribe aqui tu nombre" class="form-control">
            </div>
            <div class="mb-3">
                <label for="direccion" class="form-label">Dirección</label>
                <input type="text" id="direccion" placeholder="Escribe aqui tu dirección" class="form-control">
            </div>
            <div class="mb-3">
                <label for="telefono" class="form-label">Teléfono</label>
                <input type="text" id="telefono" placeholder="Escribe aqui tu teléfono" class="form-control">
            </div>
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="text" id="email" placeholder="Escribe aqui tu correo electrónico" class="form-control">
            </div>
        </fieldset>

    <br>
```

En nuestra segunda parte del formulario, definimos el grupo de input tipo **radio**. Aquí si debemos de definir un **name** para llamar a todo el grupo y que formen uno solo. Además, podemos utilizar el name para seleccionarlo desde JS para recorrer los elementos.

```
<div class="tamaño">

    <fieldset class="border p-5">

    <label for="tamaño">Tamaño de de la pizza</label>
    <div class="form-check">
        <!-- Cuando tenemos un input radio, es importante identificarlos con name para despues poder tratarlo como grupo
        desde el archivo de JS -->
        <input type="radio" name="tamaño" id="grande" >
        <label for="grande" >Grande</label>
    </div>
    <div class="form-check">

        <input type="radio" name="tamaño" id="mediana" >
        <label for="mediana" >Mediana</label>
    </div>
    <div class="form-check">
        <input type="radio" name="tamaño" id="pequeña" >
        <label for="pequeña" >Pequeña</label>
    </div>

    </fieldset>

</div>
```

Nuestro grupo de ingredientes, formados por **checkbox**, los agruparemos también bajo un name, porque después nos servirá para contar cuantos hemos añadido y tratarlo como grupo.

```
<div class="ingredientes">
    <fieldset class="border p-5">
<label for="ingredientes">Ingredientes de la pizza</label>


<!-- Para los checkbox igual que con radio, es importante el name para despues poder trabajar con ellos como grupo desde la parte
de JS -->
<div class="form-check form-switch">
    <label for="mozzarella" class="form-check-label">Mozzarella</label>
    <input type="checkbox" name="ingredientes" id="mozzarella" class="form-check-input">
</div>

<div class="form-check form-switch">
    <label for="bacon" class="form-check-label">Bacon</label>
    <input type="checkbox" name="ingredientes" id="bacon" class="form-check-input">
</div>

<div class="form-check form-switch">
    <label for="piña" class="form-check-label">Piña</label>
    <input type="checkbox" name="ingredientes"  id="piña" class="form-check-input">
</div>
<div class="form-check form-switch">
    <label for="anchoas" class="form-check-label">Anchoas</label>
    <input type="checkbox" name="ingredientes" id="anchoas" class="form-check-input">
</div>
<div class="form-check form-switch">
    <label for="berenjena" class="form-check-label">Berengena</label>
    <input type="checkbox" name="ingredientes" id="berenjena" class="form-check-input">
</div>
<div class="form-check form-switch">
    <label for="trufa" class="form-check-label">Trufa</label>
    <input type="checkbox" name="ingredientes" id="trufa" class="form-check-input">
</div>


 </div>
```

Por último en nuestro formulario tenemos el input de tipo **button** que será el que desencadena nuestro **evento** en el archivo JS. Pero esto lo analizamos a continuación en este archivo. Hemos utilizado un tipo button para poder utilizar **Sweet Alert** en nuestro documento Javascript.

Activaremos el submit del formulario al aceptar nuestro pedido con **sweet alert**, un plugin de JQuery que mejora la estética del alert clásico.

```
    <!-- Ahora definimos nuestro boton  que va a ser el que active la funcion de validacion en el archivo JS -->
    <!-- No utilizamos el boton submit para poder utilizar sweet alerts en nuestro archivo JS. El submit lo accionaremos desde el JS -->
   <input type="button" class="btn btn-primary " name="procesar" id="procesar" value="Procesar pedido">
```

Actividad 3. Formularios

REQUERIMIENTO 2

Nuestro archivo JS el cual hemos enlazado en el head de nuestro HTML, está estructurado de la siguiente manera:

```
/*El evento onload, se dispara al final del proceso de carga del documento. Es decir, cuando todos los objetos del DOM
(imágenes, flash, scripts, frames) han terminado de cargarse. Una excepción son las hojas de estilo, que no siempre están
cargadas al momento de lanzarse este evento. */

window.onload= function(){
    //Cuando pulsemos el input button del formulario , activamos la funcion de validacion que tenemos descrita debajo.
    //Si todos en todos los campos que validamos no obtenemos un ningun false, la validacion sera correcta. Si obtenemos algun
    //false, mostraremos un sweet alert y no podremos enviar los datos del formulario
    //Hemos utilizado un boton y no un submit para poder utilizar los sweet alert. De esta manera, el submit se realiza
    //cuando pulsamos el boton ok en el ultimo sweet alert de nuestro pedido.
    procesar.onclick =validacion;


}
```

Primero establecemos mediante el **evento onload** que cuando todos los objetos del DOM han terminado de cargarse, llamamos a la función que describimos.

Esta función determina que al hacer click en el botón del formulario, se lleva a cabo el análisis de la función validación que describimos abajo en el código. Por ahora sabemos que para que eso se lleve a cabo, esta función debe de devolver un **true**, si devuelve false, la validación será **false** y por lo tanto el **submit** al final de la función no se llevara a cabo.

A continuación, analizamos la función validación:

```
function validacion(){


    //AQUI VALIDAMOS QUE TODOS LOS CAMPOS TIENEN QUE ESTAR CUBIERTOS
    /*Con el  método trim( ) nos devuelve la cadena de texto despojada de los espacios en blanco en ambos extremos.
    Además el método no afecta al valor de la cadena de texto. Por lo tanto si el valor de los campos que tenemos definidos
    a continuacion quitando los espacios en blanco, estan vacios mostraremos el alert y devolveremos el false en la funcion */
    if (nombre.value.trim() == "" || telefono.value.trim()=="" || direccion.value.trim()=="" || email.value.trim()=="") {
        // Si no se cumple la condicion...
        swal({
            type: 'error',
            title: 'Oops...',
            text: 'Los campos Nombre, Direccion, Telefono y Email deben de estar cubiertos',
            footer: '<a href="">Rellena todos los campos para hacer un pedido</a>'
          })

        return false;
    }
```

Lo primero que analizamos en nuestra función es que todos los campos deben de estar completos, ninguno debe de encontrarse en blanco.

Para eso, utilizamos la propiedad **value** con el método **trim()** el cual nos devuelve la cadena despojada de los espacios. Por lo tanto, si todos los campos están vacíos, devolvemos un **false** en la función para que no se lleve a cabo el **submit**, y se muestre una ventana emergente con información.

Lo segundo que analizamos es que el nombre empiece por letra mayúscula:

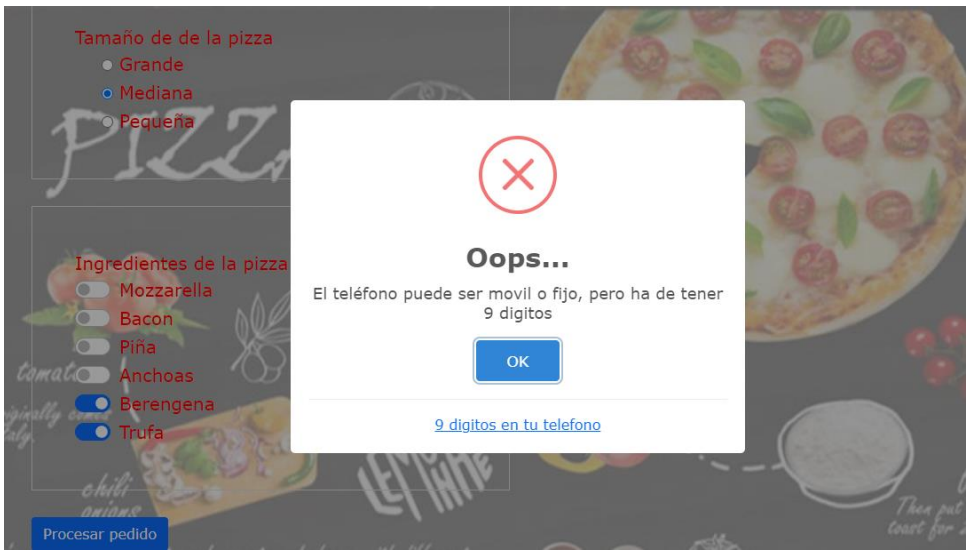
```
//AQUI IMPLEMENTAMOS CON EXPRESIONES REGULARES QUE LA PRIMERA LETRA DEL NOMBRE EMPIECE POR MAYUSCULAS
/*Definimos nuestra variable primeramayuscula de tal forma que nuestra palabra tiene que empezar por una letra(^) del rango que
establecemos [A-Z]*/
let primeramayuscula = /^[A-Z]/

/*Con el metodo match nos devuelve todas las ocurrencias que haya dentro de una cadena teniendo en cuenta una expresión regular
establecida. Nosotros mediante el condicional if, le decimos que si no existe la ocurrencia de que nuestra palabra empiece por
mayuscula, enviamos un alert y devolvemos false en la funcion*/
if (!nombre.value.match(primeramayuscula)){
   swal({
       type: 'error',
       title: 'Oops...',
       text: 'El nombre debe de tener la primera letra Mayúscula',
       footer: '<a href="">Escribe la primera letra mayúscula en el nombre</a>'
     })
     return false;
 }
```

Para ello, utilizamos expresiones regulares y con el método match observamos las ocurrencias y lo analizamos con un condicional if para ver si se producen. Si no obtenemos el resultado que deseamos, mostramos un cuadro con información tipo sweet alert y devolvemos un false para que no se lleve a cabo el submit en el formulario.

Para el teléfono y email utilizamos **expresiones regulares** de la misma manera que para el nombre.

Lo difícil en esta cuestión es dar con la expresión regular correcta para conseguir el objetivo:

```
  //AQUI IMPLEMENTAMOS QUE EL TELEFONO SEA NUMERICO Y SOLO TENGA 9 CARACTERES

  /*De la misma forma que antes, definimos la expresion regular que queremos y la introducimos en nuestra variable
  En este caso le decimos que tiene que empezar por un caracter comprendido entre [0-9] y que la longitud debe de ser de 9 numeros exactos
let tel=/^[0-9]{9,9}$/;
  /*El metodo match nos devuele las ocurrencias que haya dentro de la cadena si cumple con nuestra expresion regular, con nuestro
  condicional if, comprobamos si es falsa esta ocurrencia, pues devolvemos el alert correspondiente y retornamos false en la funcion */
if (!telefono.value.match(tel)){
    swal({
        type: 'error',
        title: 'Oops...',
        text: 'El teléfono puede ser movil o fijo, pero ha de tener 9 digitos',
        footer: '<a href="">9 digitos en tu telefono</a>'
      })
     return false;
 }
```



Actividad 3. Formularios

Cuando ya hemos analizado los inputs tipo texto, el siguiente objetivo es analizar los campos input tipo **radio** e input tipo **checkbox**.

Lo primero que nos dice el enunciado es que hay que escoger un tamaño de pizza para que se envíen los datos:

```
//Seleccionamos nuestro grupo de input radio mediante el name que le definimos en el html y lo introducimos en la variable tamaño
tamaño=document.getElementsByName("tamaño");
//Definimos una variable llamada check y la ponemos a false
Let check=false;
//Definimos la variable i para el for que despues la utilizaremos para nuestro switch, por esa razon tiene que definirse aqui
Let i;
//Definimos una variable llamada preciotamaño
Let preciotamaño;
//Definimos una variable llamada tam
Let tam;

//Ahora mediante un bucle for, recorremos el tamaño del array que forma nuestra variable tamaño con los elementos agrupados radio por name.
for(i=0;i<tamaño.length;i++){
    //Cuando el elemento correspondiente se encuentre checked ponemos la variable check a true y salimos del bucle
    if (tamaño[i].checked==true){
        check=true;
        break;
    }
}
//Si hemos salido del bucle y la variable check no esta a true, devolvemos un alert y enviamos un false.
if(check==false){
    swal({
        type: 'error',
        title: 'Oops...',
        text: 'Debes de seleccionar un tamaño de la pizza',
        footer: '<a href="">Escoge un tamaño de pizza</a>'
      })
    return false;
```

Para ello, definimos una variable llamada tamaño que es el grupo de input radio el cual seleccionamos mediante el name que habíamos definido en el html. Este grupo se va a comportar como un “array” y por lo tanto podemos recorrerlo mediante un bucle for como tal.

También definimos unas variables que después vamos a ir cargando con valores según los resultados.

Con el bucle for recorremos nuestro grupo de input radio y con un condicional if, analizamos si alguno de los elementos de ese grupo se encuentra en estado checked. Si esto se cumple, ponemos el check a true y salimos del bucle. Analizamos esa variable check y si es false, mostramos una alerta con información y devolvemos un false a la función.

Si el check es true:

```
}else{
   switch (i){
    case 0:
        //Para el caso de i=0, precio es de 15 y el tam es grande, sucesivamente establecemos los valores para las variables segun el valo
         preciotamaño=15;
         tam="grande";
        break;
     case 1:
          preciotamaño=10;
          tam="mediana";
          break;
     case 2:
          preciotamaño=5;
          tam="pequeña";
          break;
 }
}
```

Analizamos mediante un switch que elemento de nuestro grupo input **radio** se encontraba en **checked** e introducimos en las variables correspondientes sus valores para después presentar el precio correctamente.

El siguiente paso es validar que al menos un ingrediente se encuentra seleccionado. Este paso es muy sencillo con la propiedad checked que hemos analizado antes y con un condicional **if**:

```
//AQUI VALIDAMOS QUE ALGUN INGREDIENTE SE ENCUENTRE SELECCIONADO
//Con checked obligamos a que algun ingrediente se encuentre seleccionado
if(mozzarella.checked==false && piña.checked==false && bacon.checked==false && anchoas.checked==false && trufa.checked==false && berenjena
    //Si no hay ninguno seleccionado, devolvemos un alert y un false.
    swal({
        type: 'error',
        title: 'Oops...',
        text: 'Selecciona al menos un ingrediente en la pizza',
        footer: '<a href="">Escoge al menos un ingrediente</a>'
      })
    return false;

}

//SI TODAS LAS VALIDACIONES NO DEVUELVEN NINGUN FALSE, PODEMOS CALCULAR EL PRECIO DE NUESTRA PIZZA
```

Analizamos que, si ninguno de nuestros ingredientes están checked, mostramos un alert y devolvemos un false a la función.

REQUERIMIENTO 3

Cuando tenemos todas nuestras validaciones comprobadas y ninguna devuelve un false, es el momento de analizar el precio de nuestra pizza con su tamaño e ingredientes correspondientes:

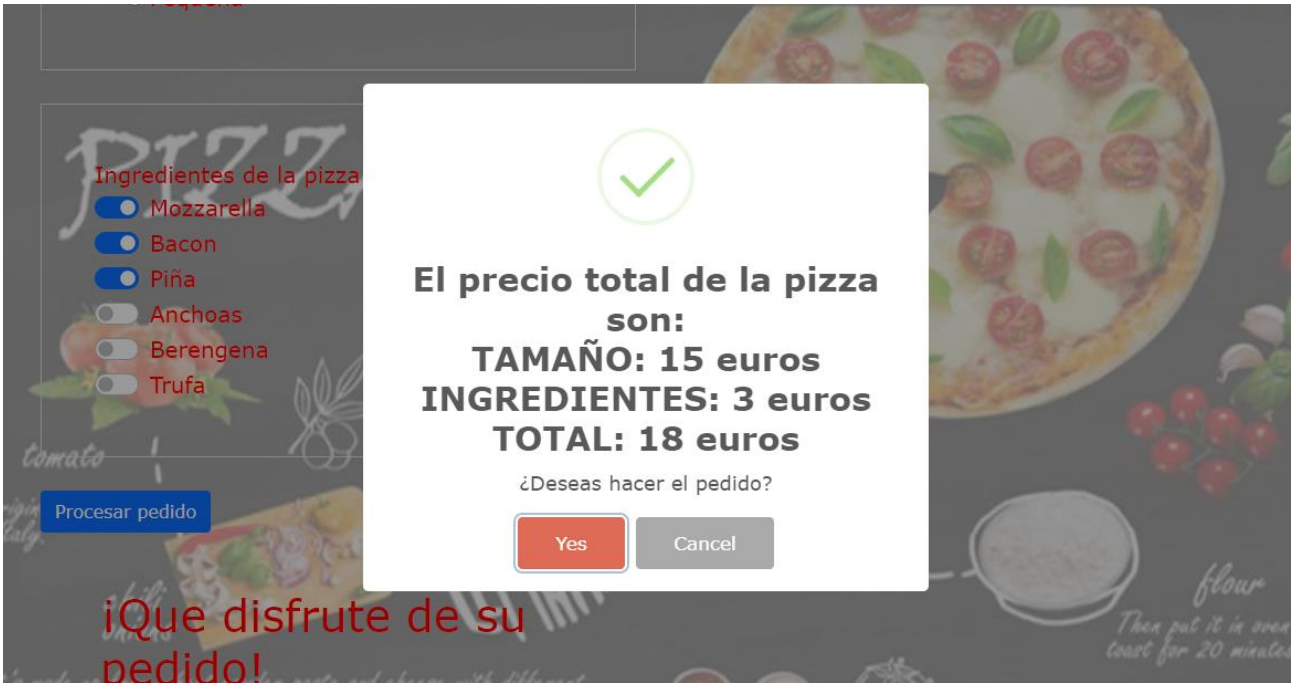
```
/*Para saber cuantos ingredientes hemos seleccionado, seleccionamos nuestro grupo checkbox con el name que le establecimos
  en el html y lo recorremos con un bucle for */
  ingredientes=document.getElementsByName("ingredientes");
  let j;
  let precioingredientes=0;
  for(j=0;j<ingredientes.length;j++){
    //Cada vez que un ingrediente se encuentra seleccionado sumamos un ingrediente a precioingredientes
    if(ingredientes[j].checked==true){
      precioingredientes++;

    }
  }

  //Y en el ultimo sweet alert devolvemos el valor total de la pizza desglosando sus apartados

  swal({
    title:"La pizza pedida es: \n"
     + "Tamaño : "+ tam.toUpperCase() + " " + preciotamaño + " euros",
    text: "Has añadido : \n"+
    precioingredientes + " ingredientes: " + precioingredientes + "euros.",
```

```
    type: 'info',

    confirmButtonColor: '#3085d6',

    confirmButtonText: 'OK'
  }).then((result) => {

      swal(

        {
          title            : "El precio total de la pizza es: \n" +
          "TAMAÑO: "+ preciotamaño+ "EUROS\n" +
          "INGREDIENTES: " + precioingredientes + "EUROS\n"+
          "TOTAL: " +(preciotamaño+precioingredientes) + " EUROS.",
          text             : "¿Deseas hacer el pedido?",
          type             : "success",
          allowEscapeKey   : false,
          allowOutsideClick : false,
          showCancelButton  : true,
          confirmButtonColor : "#DD6B55",
          confirmButtonText  : "Yes",
          showLoaderOnConfirm: true,
          closeOnConfirm    : false,
          //en sweet alert para el submit es necesario el parametro preconfirm, sino no envia formulario,
ya que tiene que ser async
          preConfirm:  function (isConfirm) {

            if (isConfirm) {
              //Aqui es donde hacemos el submit del formulario, cuando ya tenemos todo preparado
              document.formulario.submit();

              return true;
            }else{
              return false;
            }



          }
        }


      )
```

El precio según el tamaño ya lo habíamos guardado en una variable cuando lo analizamos con el bucle for y el switch. Para saber cuántos ingredientes hemos seleccionado, utilizamos un bucle for y según los checked que nos encontremos sumamos una unidad.

Después mediante alert solo nos queda presentar nuestro resultado del precio total.

Esta será nuestra imagen de nuestro formulario con la solución:



Todo el proyecto se encuentra comentado para explicar las decisiones que se toman en cada momento.

Esta memoria es un resumen de cómo funciona el proyecto.