Desarrollo Web en Entorno Cliente

AJAX

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

REPOSITORIO GITHUB
https://github.com/ITTDAW/AE-4.-AJAX.git

Actividad 4. AJAX

Requerimiento 1

Se pide hacer una aplicación AJAX que gestione una pizzería como la actividad 3. Puedes basarte en dicha actividad para reutilizar lo que consideres o para hacerte una idea de lo que tienes que implementar junto a tu compañero. El formulario que tendremos que usar será el mismo que en dicha actividad, consúltala para más información.

La idea es simular un entorno de acceso a servidor para traer toda la información que se necesite para cargar la página. La página cargará parte de los datos de manera dinámica, concretamente los tamaños y los ingredientes. Dichos datos estarán en formato JSON, por lo que habrá que tratarlos en el cliente para poder mostrarlos.

Nada más terminar de cargar la página, se accederá mediante AJAX a datos en un fichero del servidor para traer los tamaños de las pizzas y cargarlos dinámicamente. Al mismo tiempo, nos traeremos los ingredientes que tenemos disponibles para mostrarlos en nuestra pizzería.

También dispondremos de un botón de refrescar, de tal manera que cuando lo pulsemos haremos una llamada de nuevo a nuestro servidor para traernos los posibles cambios de los datos de nuestra aplicación.

No se puede utilizar la librería JQuery

Valoración: 5 puntos sobre 10

Requerimiento 2

Si pulsamos el botón de procesar el pedido, la web mostrará el resultado del precio total de la pizza. Para calcular dicho precio, el programa accederá mediante AJAX al servidor para traer la información sobre el precio del tamaño de la pizza escogido, así como el precio de los ingredientes escogidos. El precio de los ingredientes podrá ser diferente.

Valoración: 2 puntos sobre 10

Requerimiento 3

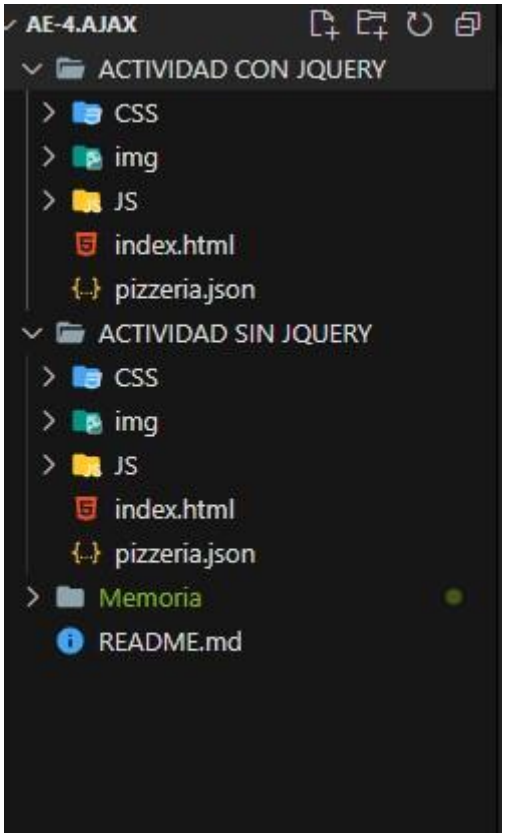
Cambiar todo lo que se considere oportuno para hacerlo con la librería JQuery

Valoración: 3 puntos sobre 10

Actividad 4. AJAX

ORGANIZACIÓN DE DIRECTORIOS

La organización de los directorios en nuestro proyecto será el siguiente:



- Tenemos dos carpetas con dos proyectos independientes los cuales deben de abrirse de forma individual en Visual Code por el tema de las rutas. Un proyecto está realizado **sin JQUERY** y otro **con JQUERY**.
- Después en cada proyecto tenemos dos carpetas llamadas **CSS** e **img** en donde almacenaremos nuestra hoja de estilos y nuestras imágenes respectivamente. Además, tenemos una carpeta llamada **JS** que es donde almacenamos nuestro archivo Javascript para las validaciones, nuestro archivo Ajax en donde haremos las peticiones Ajax con sus correspondientes funciones Callback y en el proyecto de JQUERY a mayores tendremos el archivo JS de nuestra librería JQUERY.
- En la raíz de cada 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.

Actividad 4. AJAX

La primera parte de la actividad nos pide que los datos del tamaño y los ingredientes disponibles para la pizza del formulario de la actividad anterior sean extraídos de un fichero externo y mediante peticiones AJAX transportemos esos datos del fichero a nuestra página.

Por lo tanto, nuestro HTML ya varia en varios aspectos, entre ellos, que los elementos que componen el bloque tamaños y el bloque ingredientes no serán fijos, sino que van a variar según los que nos devuelva el fichero, por lo tanto, tendrán que formarse mediante JS en una función. Nuestra primera parte del formulario en HTML será igual que la de la actividad anterior.

```
<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>
```

La segunda parte del formulario es diferente al ejemplo de la actividad 3 para introducir los elementos según la explicación anterior de que tienen que formarse según lo que nos devuelve el fichero:

```
        <div class="tamaño" id="tamaño">
            <legend for="tamaño">Tamaño de de la pizza</legend>
            <fieldset class="border p-5" id="cuadrocheck"> </fieldset>
        </div>

        <br>

        <div class="ingredientes">
            <legend for="ingredientes">Ingredientes de la pizza</legend>
            <fieldset class="border p-5" id="cuadroingredientes"> </fieldset>
        </div>

         <br>
         <!-- 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">
        <!-- Añadimos un boton para refrescar los datos de los bloques de tamaños e ingredientes segun varia el fichero -->
        <input type="button" class="btn btn-primary " name="refrescar" id="refrescar" value="Refrescar información" onclick="refrescarDatosJso
```

De esta forma, observamos que nuestro HTML queda bastante más reducido. Se ha introducido también el botón refrescar para que, si se pulsa, se actualicen los datos de los bloques correspondientes a tamaño e ingredientes. Solamente se actualiza esta parte, si tenemos información en los otros input, esta se mantiene.

Actividad 4. AJAX

Recordemos que en nuestro formulario tenemos el input de tipo **button** que será el que desencadena nuestro **evento** en el archivo JS.. 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 es momento de analizar nuestros archivos JavaScript.

Tenemos un primer archivo JavaScript llamado validaciones, que de la misma manera que en el anterior ejemplo empieza con nuestro evento onload

```
window.onload= function(){
    //Cuando pulsemos el input button con id=procesar 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 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;
    refrescar.onclick=refrescarDatosJson;
    /*Al mismo tiempo, vamos a realizar una petición AJAX al servidor para presentar los datos correspondientes a
    tamaño e ingredientes de la pizza, para ello ejecutamos la siguiente función que se encuentra en el archivo ajax.js */
    CargarPaginaConDatos();
```

Mediante este **evento onload**, cuando todos los objetos del DOM han terminado de cargarse, llamamos a las funciones que describimos.

La primera de la misma manera que antes determina que al hacer click en el botón del formulario con id=procesar, 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.

Las validaciones de los campos nombre, teléfono y correo electrónico ya están explicadas en la anterior memoria y en este caso sin JQUERY son exactamente las mismas.

La segunda, cuando hacemos click en el botón con id=refrescar nos llama a la función refrescarDatosJson que significará una llamada mediante AJAX al servidor, trayendo el fichero con su información y mediante su función callback nos va a actualizar los datos de los bloques de elementos de tamaño e ingredientes.

La tercera, nos llama a la función cargarPaginaConDatos que es otra llamada mediante AJAX al servidor para devolver nuestro fichero pizzería.json y según lo que contenga, formar nuestros bloques correspondientes a tamaño e ingredientes.

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

Como sabemos del ejemplo anterior, deben de estar seleccionado tamaño e ingredientes para que se pueda llevar a cabo el submit del formulario.

Para ello, definimos una variable llamada tamaño que es el grupo de input radio el cual seleccionamos mediante el name que definimos al formar el grupo de inputs tipo radio con la función JS. Este grupo se va a comportar como un “array” y por lo tanto podemos recorrerlo mediante un bucle for como tal.

.

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. Si check sigue en false, mostramos una alerta con información y devolvemos un false a la función.

```
//AQUI VALIDAMOS QUE ESCOGEMOS AL MENOS UN TAMAÑO DE LA PIZZA

//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 ipara el for que despues la utilizaremos para nuestro switch, por esa razon tiene que definirse aqui
let i;
//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 los ingredientes se hace de una forma parecida a la anterior, si no hay ningún checked en el grupo de ingredientes, mostramos un mensaje de error:

```
//AQUI VALIDAMOS QUE ALGUN INGREDIENTE SE ENCUENTRE SELECCIONADO
//Con checked obligamos a que algun ingrediente se encuentre seleccionado
ingredientes=document.getElementsByName("ingredientes");
let check2=false;
for(i=0;i<ingredientes.length;i++){
    //Cuando el elemento correspondiente se encuentre checked ponemos la variable check a true y salimos del bucle
    if (ingredientes[i].checked==true){
        check2=true;
        break;
    }
}
if(!check2){
    //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;

}

CargarPaginaParaPrecios();
```

Cuando hemos pasado todas las validaciones exigidas, llamamos a la función CargarPaginaParaPrecios, que es una función que vuelve a hacer una petición AJAX y mediante su función callback y el fichero que nos devuelve esta llamada calculamos los precios del tamaño de pizza elegida, así como de los ingredientes que hemos añadido.

A continuación, analizamos el segundo archivo JS que es el que contiene todas las llamadas mediante AJAX para devolver nuestro fichero pizzería.json.

```
//En una peticion asincrona el JS de nuestro navegador NO se queda bloqueado, por lo
//que tenemos que tener una funcion de callback que se ejecutará cuando el servidor
//nos mande la respuesta

function CargarPaginaConDatos(){
    //El puerto que utiliza el live Server de Visual Code, recuerda que si lo tienes ocupado, live Server abre otro puerto y lo deberias de cai
    const URL_DESTINO = "http://localhost:5500/"
    const RECURSO = "pizzeria.json"



    //Siempre hay que formar el objeto XMLHttpRequest
            let xmlHttp = new XMLHttpRequest()
            let procesar=document.querySelector("#procesar");
            xmlHttp.onreadystatechange = function () { //Esta funcion se va a ejecutar CADA VEZ que haya un cambio en la propiedad
                //"readyState"
                 //Solo cuando la respuesta este completa y su estado sea 200 (OK) leeremos el mensaje
        //del servidor y la procesamos
                if (this.readyState == 4) {
                    if (this.status == 200) {

                        //Llamamos a nuestra funcion procesarRespuesta y le enviamos el objeto en String por parametro
                        procesarRespuesta(this.responseText)



                    }
                     else {
                        alert("ALGO VA MAL! ESTAS CONECTADO AL SERVIDOR?. LOS DIRECTORIOS DE CON jQUERY Y SIN_JQUERY DEBEN DE ABRIRSE POR SEPAI
                    }
                }
            }
           // Para enviar una solicitud al servidor, se utiliza el método Open del objeto XMLHttpRequest () y send ():
            xmlHttp.open('GET', URL_DESTINO + RECURSO, true)
            xmlHttp.send(null)
```

En nuestras peticiones AJAX sin JQUERY siempre tenemos que formar el objeto xmlHttpRequest. Después mediante el método de este objeto onreadystatechange, analizamos si la respuesta está completa y el estado es 200(Ok), si esto sucede, ejecutamos nuestra función correspondiente pasándole por parámetro el objeto en formato String. Después debemos de parsearlo a json para trabajar con él. Para hacer la solicitud al servidor utilizamos el método open y send.

Ahora ya hemos hecho llamamiento a nuestra función a ejecutar con el fichero en formato String que le pasamos por parámetro. Para esta llamada en concreto, lo que buscamos es que cuando se cargue la página, los bloques de tamaño e ingredientes nos muestren la información correspondiente a sobre cuentos tamaños e ingredientes hay en el archivo json. Para ello, desarrollamos nuestra función para crear estos elementos:

Creamos una variable correspondiente e introducimos en ella el parámetro que traemos en string, pero ya parseado para que sea una variable de tipo JSON.

```
function procesarRespuesta(jsonDoc) {
    //Convertimos un String a un objeto JSON
    var objetoJson = JSON.parse(jsonDoc)


    /**
     * Ahora con nuestro objeto JSON, accedemos a los tamaños de la pizza y los introducimos en un array, tendremos un array de objetos
     * ya que tamaños esta formado por TAMAÑO y PRECIO
     */
    let arrayTamaños=objetoJson.PIZZA.TAMAÑOS;
    /**
     * Accedemos a los ingredientes de la pizza y los introducimos en un array, tendremos un array de objetos
     * ya que ingredientes esta formado por INGREDIENTES y PRECIO
     */
    let arrayIngredientes=objetoJson.PIZZA.INGREDIENTES;
```

Ahora que ya tenemos nuestro objeto JSON podemos sacar la información que nos interesa, en este caso, formamos un array de los tamaños y de los ingredientes. Estos van a ser un array de objetos, porque recordemos que estos objetos tienen dos parámetros.

Después tenemos que introducir dentro de nuestro elemento con id=cuadrocheck nuestros input radio correspondientes con sus labels. Para ello lo hacemos como se muestra en el código de la imagen.

```
/*A continuacion vamos a formar nuestros elementos del formulario que estan a la espera de recibir los datos del json. En este caso
estamos esperando a los valores de los tamaños y los ingredientes.
Para la parte de tamaños, tenemos que formar los radio button correspondientes e introducirlos en el fieldset existente.
Seleccionamos el elemento fieldset cuadrocheck y lo introducimos en una variable*/
let cuadrocheck=document.querySelector("#cuadrocheck");

//Con un for recorremos nuestro arrayTamaños que hemos definido anteriormente
for (let i = 0; i < arrayTamaños.length; i++) {
    //Para cada elemento del array formamos un div y le ponemos el nombre de clase para bootstrap
    let div=document.createElement("div");
    div.className="form-check";
    //Para cada elemento creamos un input, le decimos que es de tipo radio , le introducimos el name tamaño para que formen todos parte de
    //le introducimos el nombre de la clase correspondiente para formato bootstrap
    let input=document.createElement("input");
    input.type="radio";
    input.name="tamaño";
    input.className="form-check-input";
    //creamos una etiqueta para cada radio
    let label=document.createElement("label");
    //Creamos una variable texto que contiene el nombre de cada tamaño segun su indice
    let texto=arrayTamaños[i].TAMAÑO
    //Le introducimos el nombre a la etiqueta la primera en mayuscula y lo restante en minuscula
    label.textContent=texto[0].toUpperCase()+(texto.slice(1).toLowerCase());
    //le establecemos el for a la etiqueta
    label.htmlFor=texto;
    //le establecemos el id al input
    input.id=texto;
    //agregamos al div creado al principio el input y el label
    div.appendChild(input);
    div.appendChild(label);

    //agregamos al fieldset el div

    cuadrocheck.appendChild(div);

}
```


Para formar el bloque de ingredientes se hace de manera análoga a la anterior y se muestra a continuación:

Actividad 4. AJAX

```
//La forma de crear los checkbox para los ingredientes es de una manera analoga al anterior caso pero con los checkbox y los ingredientes
let cuadroingredientes=document.querySelector("#cuadroingredientes");

for (let i = 0; i < arrayIngredientes.length; i++) {
    let div=document.createElement("div");
    div.className="form-check form-switch";
    let input=document.createElement("input");
    input.type="checkbox";
    input.name="ingredientes";
    input.className="form-check-input"
    let label=document.createElement("label");
    let texto=arrayIngredientes[i].INGREDIENTE;
    label.textContent=texto[0].toUpperCase()+(texto.slice(1).toLowerCase());
    label.htmlFor=texto;
    input.id=texto;
    div.appendChild(input);
    div.appendChild(label);



    cuadroingredientes.appendChild(div);

}
```

De esta manera según se carga la página, se forman los elementos correspondientes tanto de input radio como de input checkbox correspondientes a los que se encuentran en el fichero pizzería.json.

La siguiente función, nos describe la llamada AJAX que tendrá como como propósito traer nuestro fichero para calcular mediante una función el precio de la pizza .

Para ello se vuelve a realizar una petición AJAX como la que explicamos anteriormente, y con una función callback diferente que nos va a llevar a ejecutar una función que recibe como parámetro el fichero de la pizzería.

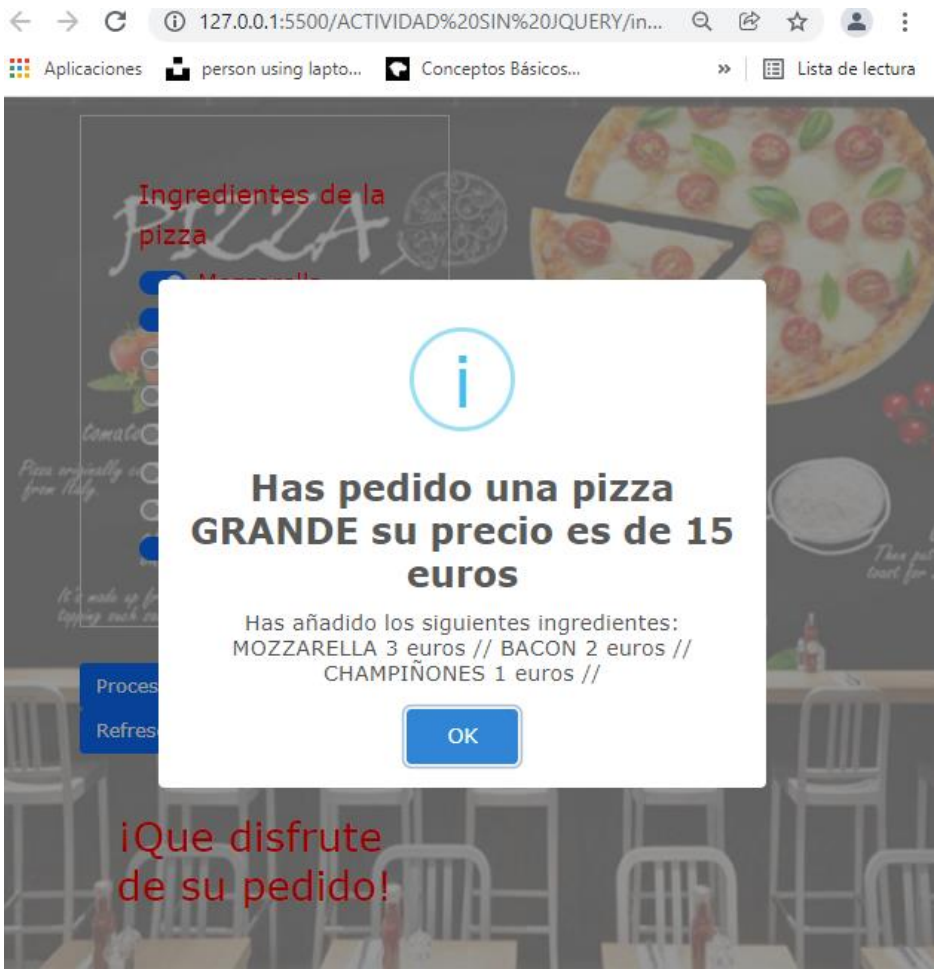
Esta función después de parsear nuestro objeto que recibimos por parámetro y de conseguir nuestro array de objetos de tamaños e ingredientes, realizamos las operaciones necesarias para sacar el precio correspondiente al tamaño conseguido y el precio correspondiente a cada ingrediente seleccionado. Para ello es necesario analizar que input está en estado checked:

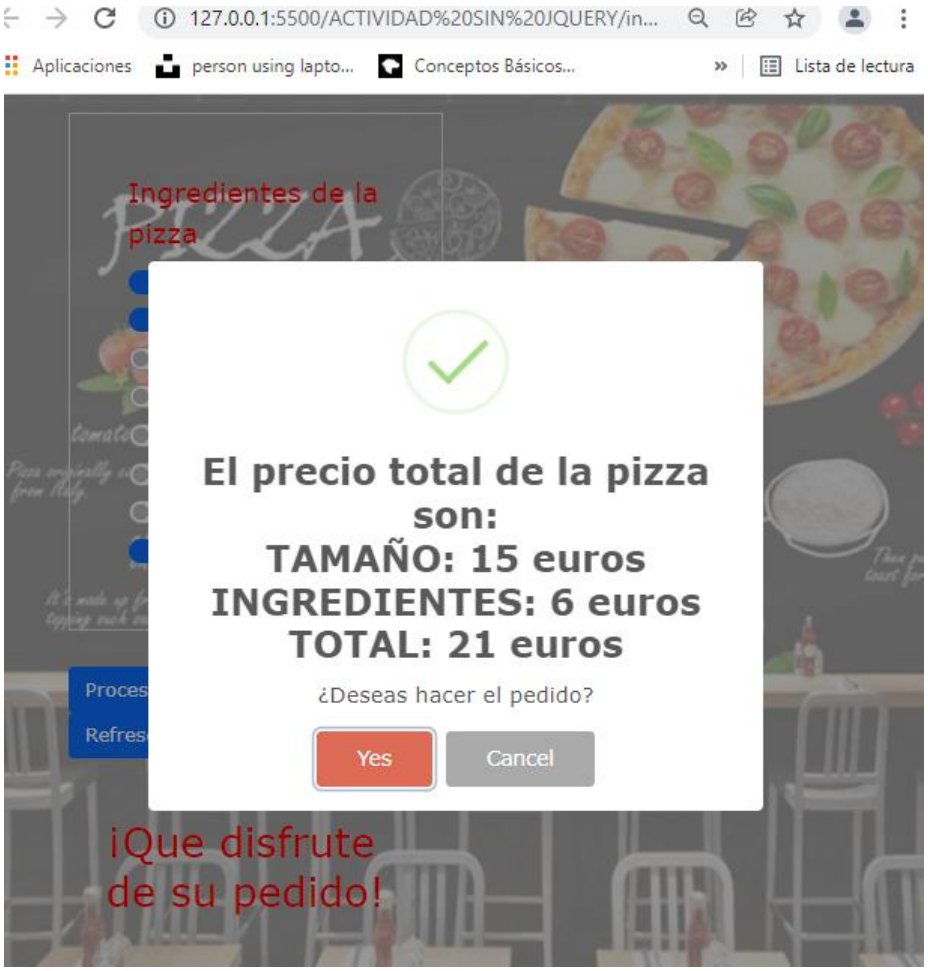
Actividad 4. AJAX

```
//Creamos el array de objetos arrayTamaños y arrayIngredientes
let arrayTamaños=objetoJson.PIZZA.TAMAÑOS;

let arrayIngredientes=objetoJson.PIZZA.INGREDIENTES;


let i=0;
//Creamos una variable preciotamaño donde vamos a introducir el precio del tamaño escogido
let preciotamaño=0;
//Recorremos el grupo de los radio, buscando cual esta checked, como todos tienen el mismo name, solamente habrá uno seleccionado,
//cuando lo encuentra sale del bucle.
for( i=0;i<tamaño.length;i++){

    if (tamaño[i].checked==true){
        //Establecemos el precio del tamaño escogido mediante buscando en nuestro array de objetos el objeto correspondiente al i y su pre
        preciotamaño=arrayTamaños[i].PRECIO;
        break;
    }
}

//Ahora vamos a hacer lo mismo con los ingredientes
//Creamos una variable en donde almacenaremos el precio de los ingredientes total
let precioingredientes=0;

//Creamos una variable en donde vamos a concatenar el texto ingrediente mas precio para la presentacion despues en el sweet alert
let cadenaIngredientesAñadidos="";

    //Recorremos el array formado por los elementos ingredientes que tienen el mismo name, se podria recorrer tambien el arrayIngredientes
     for(let j=0;j<ingredientes.length;j++){
    //Cada vez que un ingrediente se encuentra seleccionado...
    if(ingredientes[j].checked==true){
        //Sumamos a nuestra variable el precio de ese ingrediente que lo sacamos segun el indice j que ocupa en ese array
        precioingredientes=precioingredientes+arrayIngredientes[j].PRECIO
       //Añadimos a nuestra cadena el nombre del ingrediente añadido y su precio
        cadenaIngredientesAñadidos=cadenaIngredientesAñadidos.concat(arrayIngredientes[j].INGREDIENTE+
        " "+arrayIngredientes[j].PRECIO + " euros"+" // ");
```

Cuando ya disponemos de los precios correspondientes enviamos un mensaje de información del precio total con sus ingredientes correspondientes.

Actividad 4. AJAX





Actividad 4. AJAX

Nuestra última petición AJAX va asociada a la última función de nuestro requerimiento 1 que es la de disponer de un botón de refrescar que nos actualice la información de nuestros bloques con los elementos de tamaños e ingredientes. Para ello hacemos una petición AJAX al servidor como hemos hecho anteriormente, cuando la petición está completa y Ok, ejecutamos nuestra función que recibe como parámetro el objeto correspondiente en formato de String.

Primero le seleccionamos los bloques correspondientes y los vaciamos para después llamar a nuestra función que implementamos anteriormente y que nos construye los elementos necesarios según la información que recibe de nuestro archivo pizzería.json que en este caso le volvemos a pasar por parámetro.

```
//Esta es nuestra funcion callback para refrescar los datos de los bloques correspondientes sin refrescar los otros campos

function RefrescarDatos(jsonDoc) {

    //Convertimos un texto a un objeto JSON

    //Vaciamos nuestros bloques que contienen los tamaños y los ingredientes para no acumular los nuevos sobre los viejos
    let cuadrocheck=document.querySelector("#cuadrocheck");
    let cuadroingredientes=document.querySelector("#cuadroingredientes");
    cuadrocheck.innerHTML="";
    cuadroingredientes.innerHTML="";

    //Llamamos a nuestra funcion ya creada que recibe nuestro objeto json nuevo por parametro y vuelve a crear los elementos necesarios
    procesarRespuesta(jsonDoc);



}
```

El requerimiento 3 de esta actividad, nos solicita que todo lo que hemos hecho anteriormente se implemente, pero esta vez utilizando JQUERY en donde se considere.

Por lo tanto, realizamos una copia del anterior proyecto y procedemos a variar lo necesario para la realización del mismo con JQUERY.

Introducimos nuestra librería JQUERY en formato físico en nuestro directorio JS.

El HTML utilizamos el mismo sin ninguna variación.

El archivo JS hay que modificarlo para adaptarlo al formato JQUERY:

```
$(function(){

    //Cuando pulsemos el input button con id=procesar 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 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").click(validacion);
    //Cuando pulsemos el boton refrescar se llevará a cabo la funcion refrescarDatosJson.
    $("#refrescar").click(refrescarDatosJson);

    /*Al mismo tiempo, vamos a realizar una petición AJAX al servidor para presentar los datos correspondientes a
    tamaño e ingredientes de la pizza cada vez que se carga la pagina, para ello ejecutamos la siguiente función que se encuentra
    en el archivo ajax.js */

    $(CargarPaginaConDatos);
});
```

Actividad 4. AJAX

$(function(){} es la manera corta para $(document).ready que es lo mismo que cuando utilizamos window.onload, se ejecuta cuando el navegador ha terminado de cargar la página.

Después llevamos a cabo las mismas funciones, pero utilizando los selectores de JQUERY.

En la función validación es prácticamente igual, pero de la misma manera hay que utilizar selectores JQUERY, como ejemplo el de la validación del teléfono:

```
if (!$("#nombre").val().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;
  }
```

Con la validación de que un tamaño se encuentre seleccionado o al menos un ingrediente se haya marcado, el JQUERY si nos presenta una ventaja, ya que podemos formar un array de los objetos tamaño seleccionados añadiéndole la propiedad checked en el selector. De esta manera si el array tiene un length mayor de cero, ya sabemos que al menos uno esta seleccionado, si el array tiene un lenght de cero es que no hay ningún seleccionado y mostramos nuestro mensaje de error:

```
let tamaño=$("input[name=tamaño]:checked");

//Por lo tanto analizamos si ese array tiene una longitud de cero, si es asi, significa que ningun elemento esta seleccionado

if (tamaño.length==0){
    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;
}

//AQUI VALIDAMOS QUE ALGUN INGREDIENTE SE ENCUENTRE SELECCIONADO
//Utilizamos el selector de JQUERY que ya nos va a dar un objeto con los input con ese name seleccionados (checked)
let  ingredientes=$("input[name=ingredientes]:checked");

if(ingredientes.length==0){
    //Por lo tanto analizamos si ese array tiene una longitud de cero, si es asi, significa que ningun elemento esta seleccionado
    //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;

}
```

En nuestras peticiones AJAX también se nota una reducción del código:

Actividad 4. AJAX

```
function CargarPaginaConDatos(){
    //El puerto que utiliza el live Server de Visual Code, recuerda que si lo tienes ocupado, live Server abre otro puerto y lo deberias de ca
    const URL_DESTINO = "http://localhost:5500/"
    const RECURSO = "pizzeria.json"

        //Con JQuery se hace así lo cual es mucho mas rapido que de la forma tradicional ahorrando bastante codigo.
    $.ajax({
        'type'  : 'GET', //Por defecto es GET
        'url'   : URL_DESTINO + RECURSO,
        'async' : true, //Por defecto es true
    }
        ).done(procesarRespuesta)//funcion de callback que ejecutamos si todo ha ido bien
        .fail(procesarError)//funcion de callback que ejecutamos si ha ido mal, optativa

}
```

Para nuestra función asociada a la petición AJAX cuando cargamos la página y que se nos muestre la información de los tamaños e ingredientes existentes en el archivo pizzería.json, con JQUERY, el código se resume en menos líneas y por lo tanto se aligera nuestra implementación:

Recordemos que con JQUERY nuestro objeto que recibimos en la función del callback ya viene en formato JSON y no es necesario hacerle el parse.

```
for (let i = 0; i < arrayTamaños.length; i++) {

    let texto=arrayTamaños[i].TAMAÑO
    $("<div>", {
        'class': "form-check"
    }).append(
        $('<input>', {
            'class': "form-check-input",
            'type': 'radio',
            'name': 'tamaño',
        }),
        $('<label>', {
            'id': "texto",
            'text': texto[0].toUpperCase()+(texto.slice(1).toLowerCase()),

        })

    ).appendTo('#cuadrocheck');


}
```

La formación de los input radio es mucho más sencilla gracias a los append y appendTo.

La formación de los input checkbox se hace de forma análoga a esta:

```
for (let i = 0; i < arrayIngredientes.length; i++) {

    let texto=arrayIngredientes[i].INGREDIENTE;
    $("<div>", {
        'class': "form-check form-switch"
    }).append(
        $('<input>', {
            'class': "form-check-input",
            'type': 'checkbox',
            'name': 'ingredientes',
        }),
        $('<label>', {
            'id': "texto",
            'text': texto[0].toUpperCase()+(texto.slice(1).toLowerCase()),

        })

    ).appendTo('#cuadroingredientes');


}
```

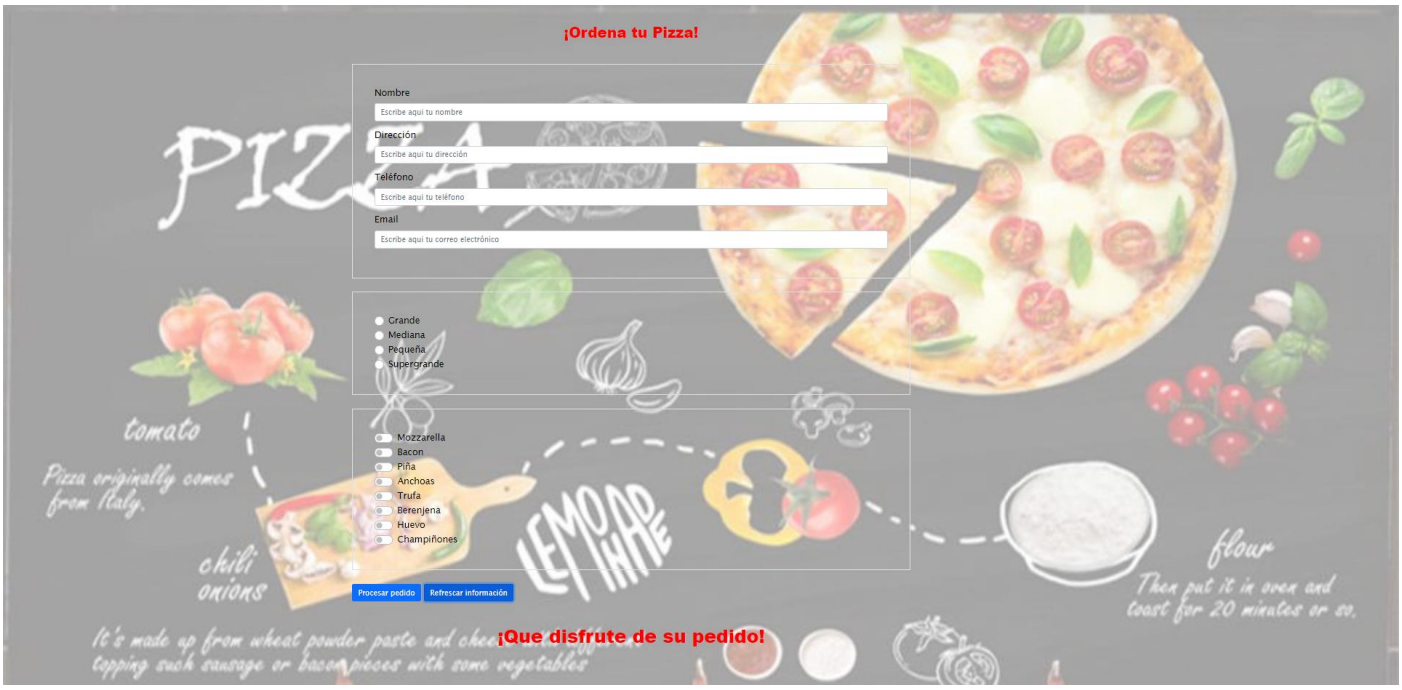
Nuestra función de calcular los precios después de la llamada mediante AJAX al servidor y que tengamos nuestra función de respuesta callback con el parámetro JSON correspondiente es muy similar a la forma de hacerlo sin JQUERY, solamente hay que tener en cuenta utilizar los selectores correctos para seleccionar bien nuestros elementos. Como por ejemplo para los precios según tamaño:

```
//Con JQUERY creamos una variable tamaño y le decimos que son los input con el name tamaño, por lo tanto sera una variable de tipo array
let tamaño=$("input[name=tamaño]");
//Recorremos el grupo de los radio, buscando cual esta checked, como todos tienen el mismo name, solamente habrá uno seleccionado,
//cuando lo encuentra sale del bucle.
for( i=0;i<tamaño.length;i++){

    if (tamaño[i].checked==true){
        //Establecemos el precio del tamaño escogido mediante buscando en nuestro array de objetos el objeto correspondiente al i y su pre
        preciotamaño=arrayTamaños[i].PRECIO;
        break;
    }
}
```

Por último tenemos nuestra función asociada al botón refrescar que de la misma forma que los anteriores, hay que emplear una correcta forma de utilizar los selectores JQUERY:

Actividad 4. AJAX

```
//Esta es nuestra funcion callback para refrescar los datos de los bloques correspondientes

function RefrescarDatos(objetoJSon) {


    //Vaciamos nuestros bloques que contienen los tamaños y los ingredientes para no acumular los nuevos sobre los viejos
    $("#cuadrocheck").empty();
    $("#cuadroingredientes").empty();

    //Llamamos a nuestra funcion ya creada que recibe nuestro objeto json nuevo por parametro y vuelve a crear los elementos necesarios
    procesarRespuesta(objetoJSon);



}
```



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.