Desarrollo Web en Entorno Cliente

AJAX

Una lámpara en el agua

Descripción generada automáticamente con confianza media

|  |
| --- |
| 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> |
|  |

Formularios

**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*

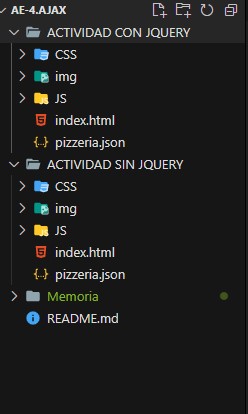
**Requerimiento 3**

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

*Valoración: 3 puntos sobre 10*

**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.

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.

Texto

Descripción generada automáticamente

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:



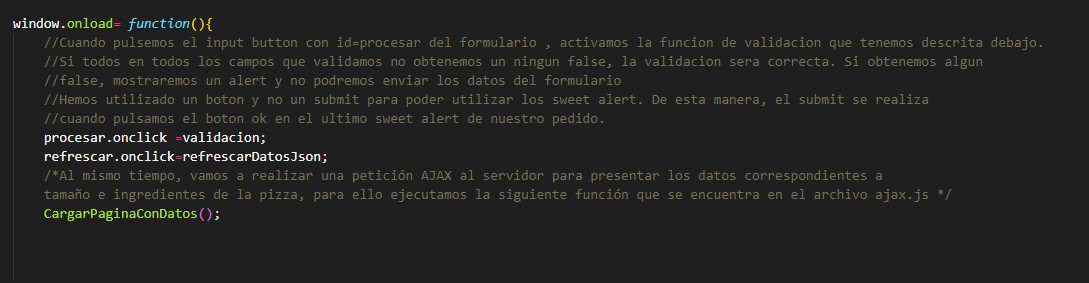
De esta forma, observamos que nuestro HTML queda bastante más reducido. Se ha introducido también el botón refrescar para que, si se pulsa, se actualicen los datos de los bloques correspondientes a tamaño e ingredientes. Solamente se actualiza esta parte, si tenemos información en los otros input, esta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

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



|  |
| --- |
|  |

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.

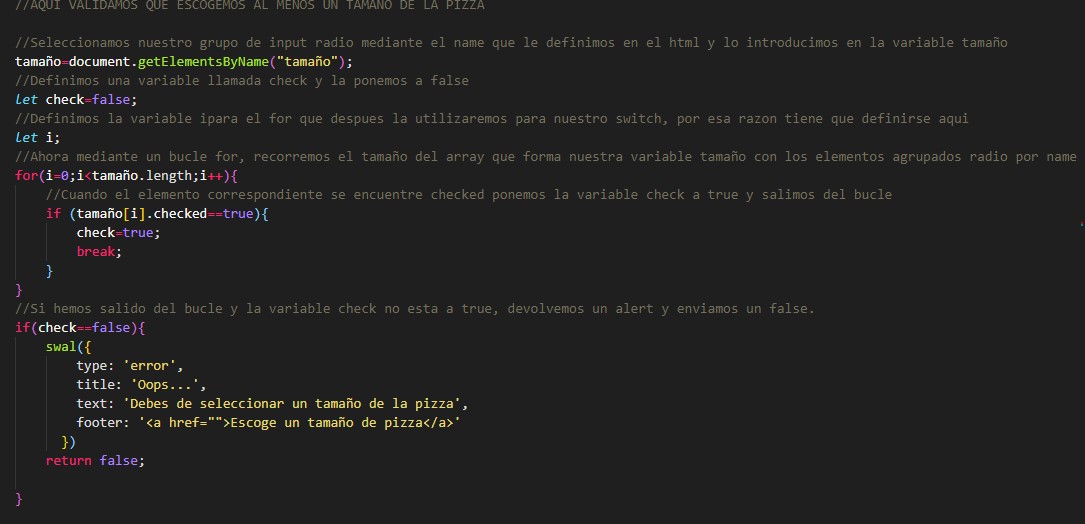
Cuando ya hemos analizado los inputs tipo texto, el siguiente objetivo es analizar los campos input tipo **radio** e input tipo **checkbox**.

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

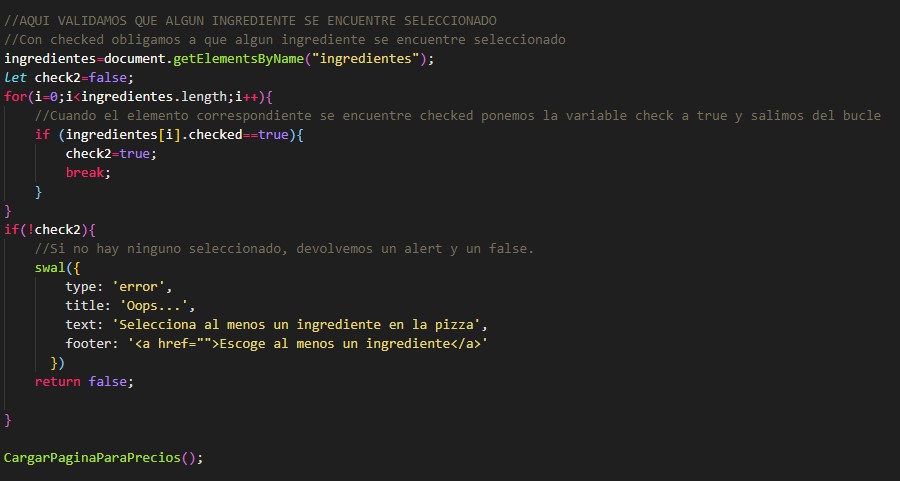
Para ello, definimos una variable llamada tamaño que es el grupo de input radio el cual seleccionamos mediante el name que definimos al formar el grupo de inputs tipo radio con la función JS. Este grupo se va a comportar como un “array” y por lo tanto podemos recorrerlo mediante un bucle for como tal.

.

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.

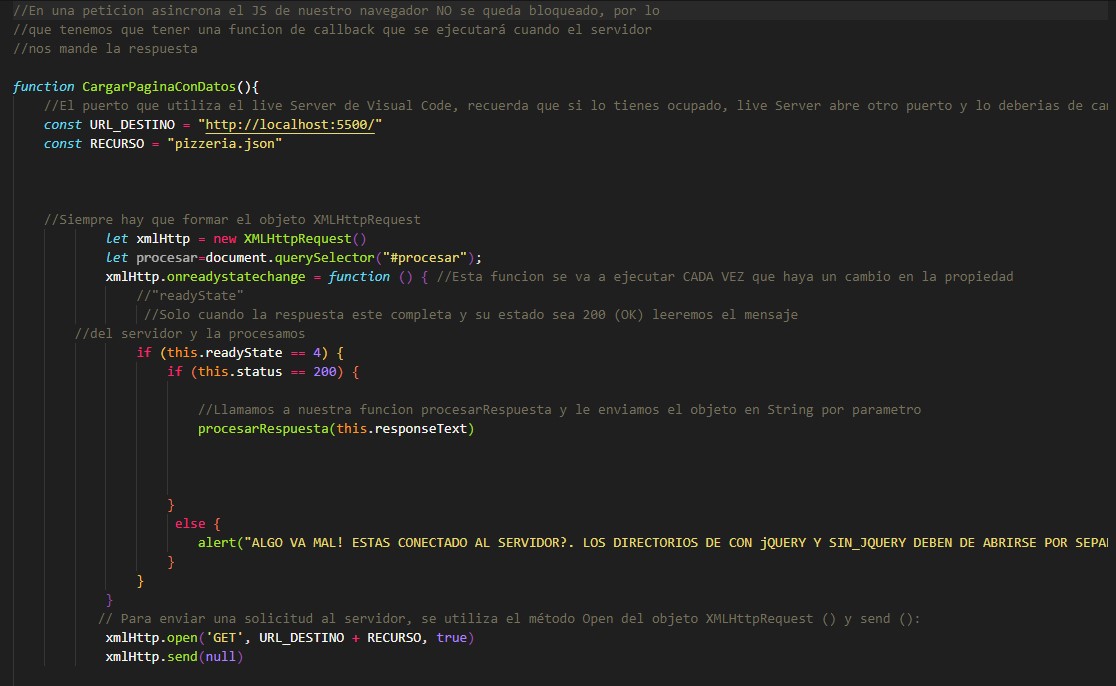


Para los ingredientes se hace de una forma parecida a la anterior, si no hay ningún checked en el grupo de ingredientes, mostramos un mensaje de error:



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.

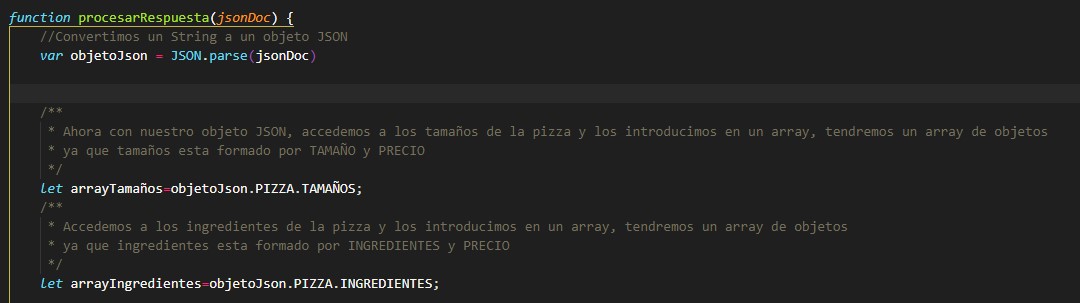
A continuación, analizamos el segundo archivo JS que es el que contiene todas las llamadas mediante AJAX para devolver nuestro fichero pizzería.json.



En 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.

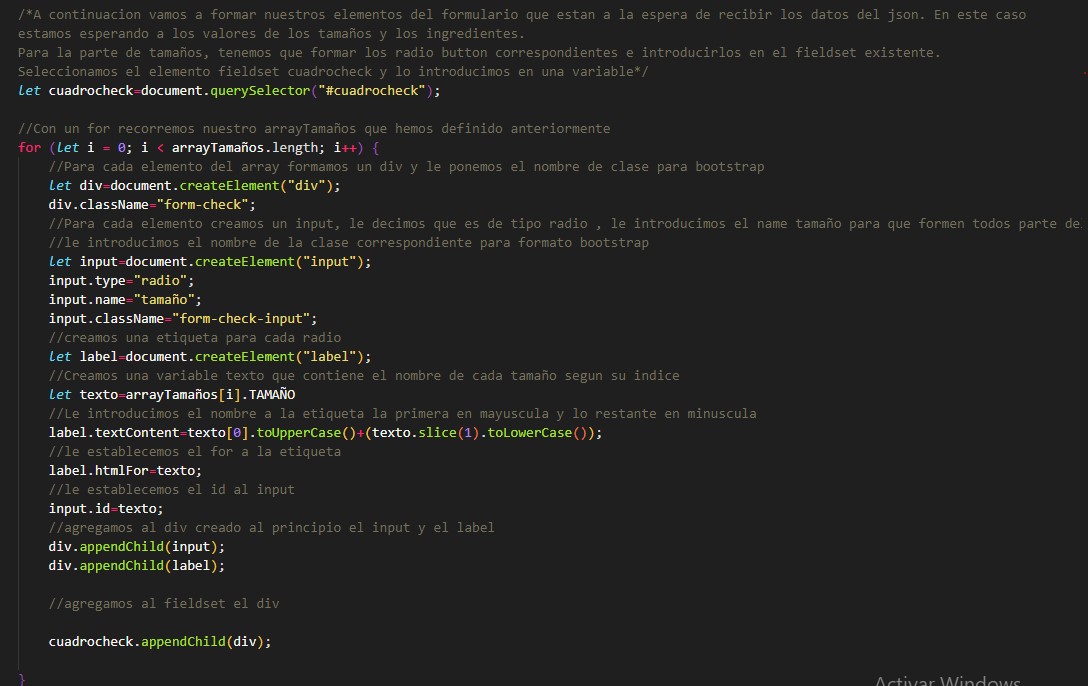
Ahora ya hemos hecho llamamiento a nuestra función a ejecutar con el fichero en formato String que le pasamos por parámetro. Para esta llamada en concreto, lo que buscamos es que cuando se cargue la página, los bloques de tamaño e ingredientes nos muestren la información correspondiente a sobre cuentos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

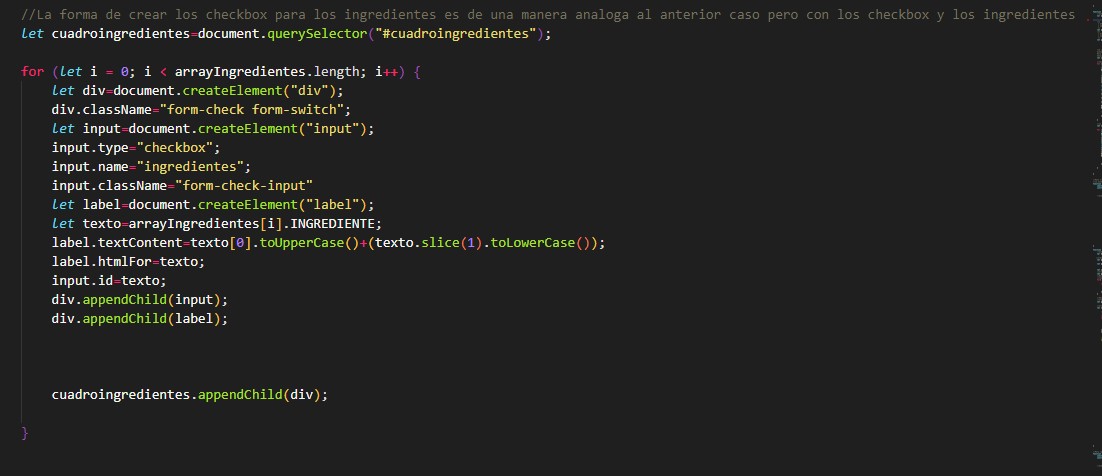


Ahora que ya tenemos nuestro objeto JSON podemos sacar la información que nos interesa, en este caso, formamos un array de los tamaños y de los ingredientes. Estos van a ser un array de objetos, porque recordemos que estos objetos tienen dos parámetros.

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.



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

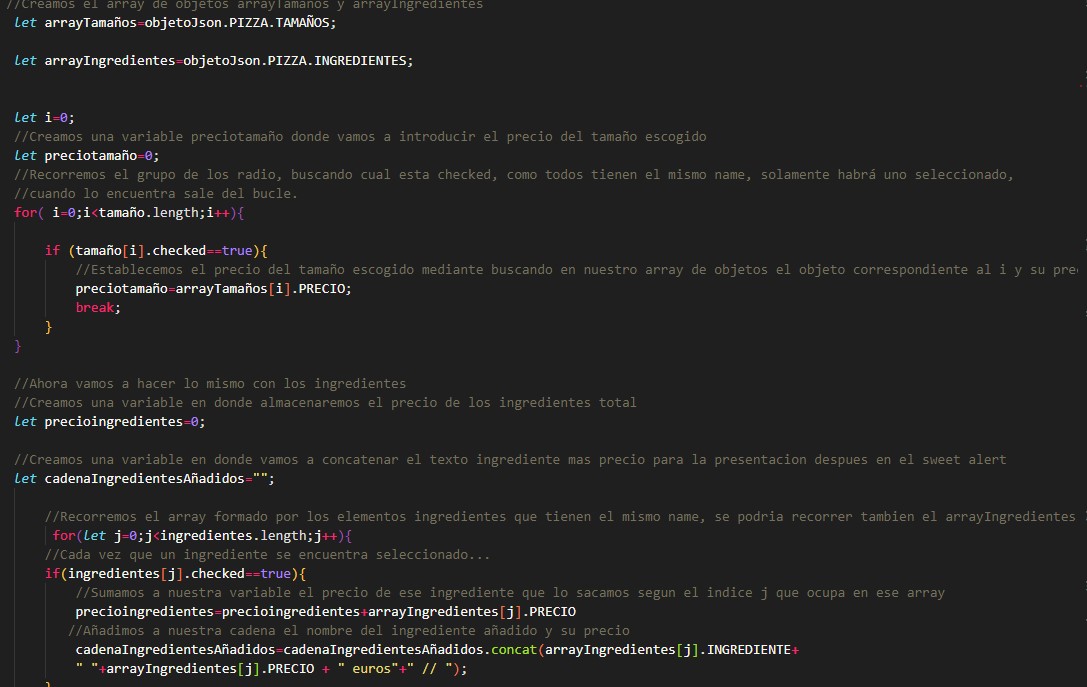


De esta manera según se carga la página, se forman los elementos correspondientes tanto de input radio como de input checkbox correspondientes a los que se encuentran en el fichero pizzería.json.

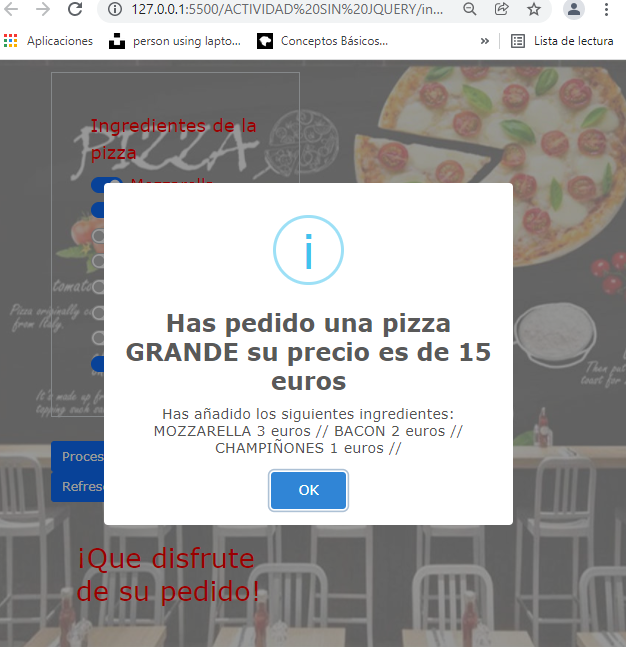
La siguiente función, nos describe la llamada AJAX que tendrá como como propósito traer nuestro fichero para calcular mediante una función el precio de la pizza .

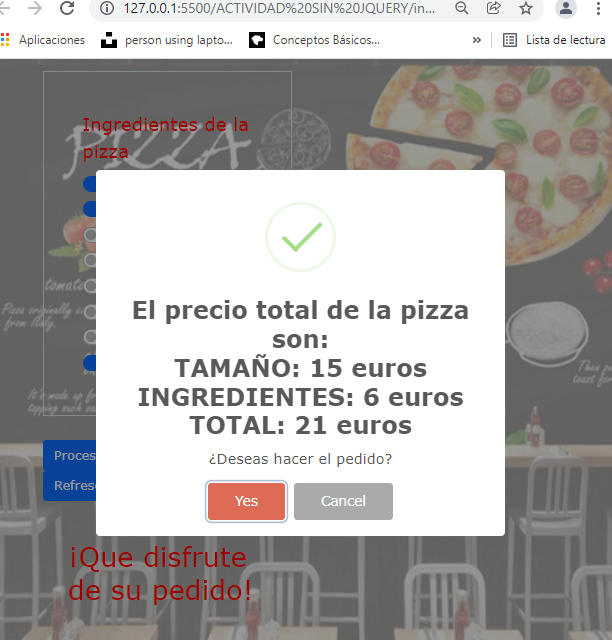
Para ello se vuelve a realizar una petición AJAX como la que explicamos anteriormente, y con una función callback diferente que nos va a llevar a ejecutar una función que recibe como parámetro el fichero de la pizzería.

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:



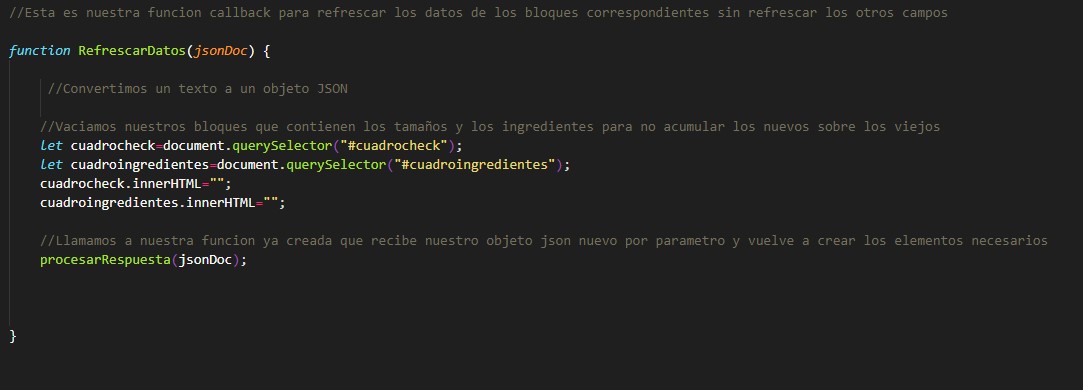
Cuando ya disponemos de los precios correspondientes enviamos un mensaje de información del precio total con sus ingredientes correspondientes.





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.



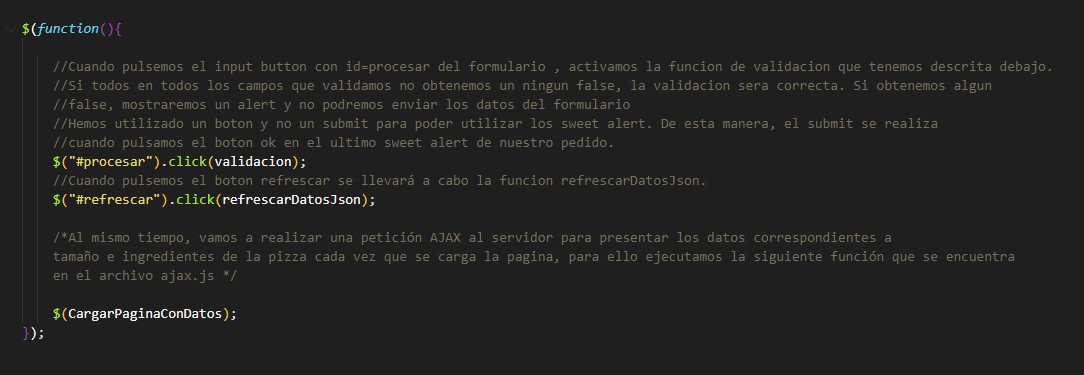
El requerimiento 3 de esta actividad, nos solicita que todo lo que hemos hecho anteriormente se implemente, pero esta vez utilizando JQUERY en donde se considere.

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.

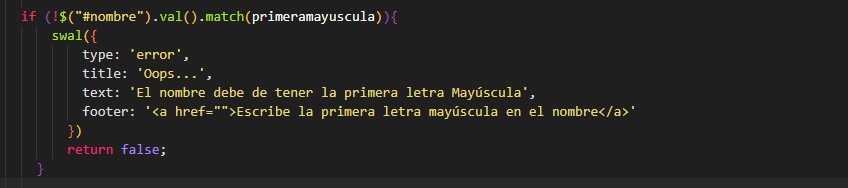
El archivo JS hay que modificarlo para adaptarlo al formato JQUERY:



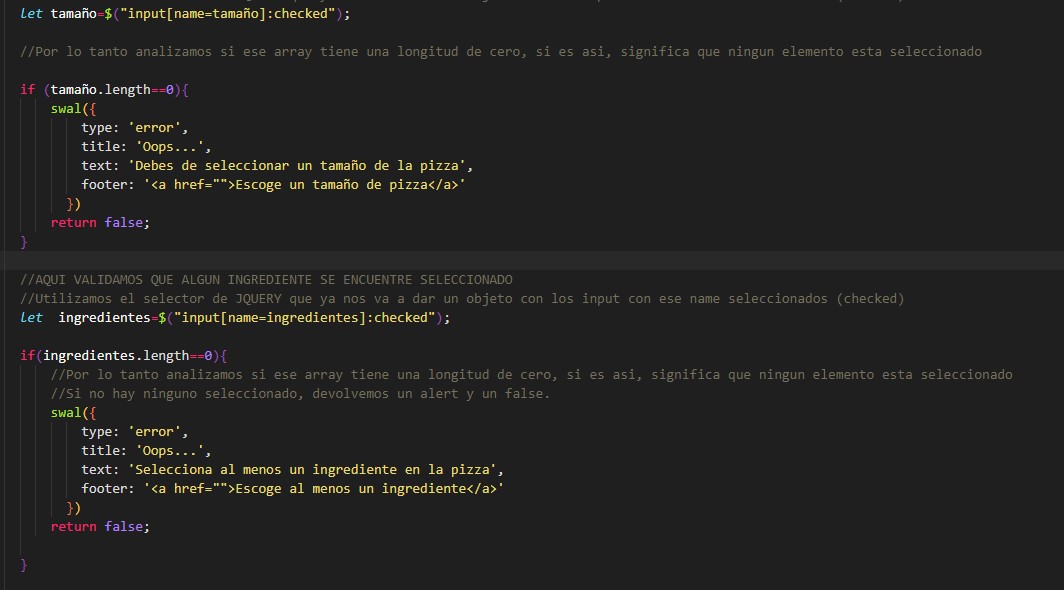
$(function(){} es la manera corta para $(document).ready que es lo mismo que cuando utilizamos window.onload, se ejecuta cuando el navegador ha terminado de cargar la página.

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

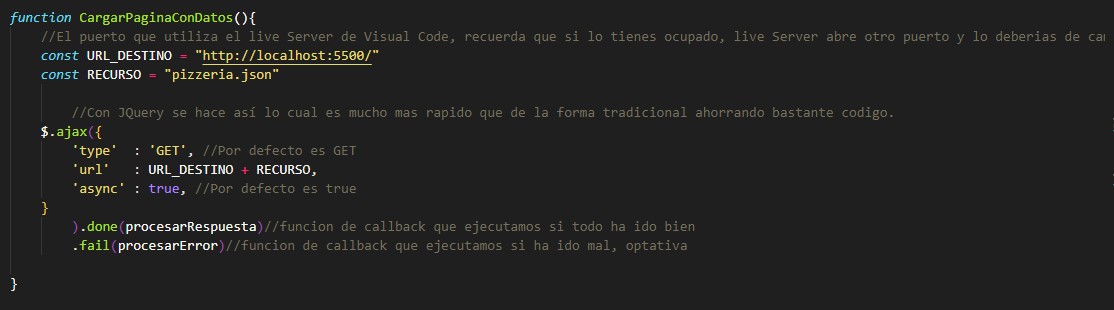
En la función validación es prácticamente igual, pero de la misma manera hay que utilizar selectores JQUERY, como ejemplo el de la validación del teléfono:



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:



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



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.

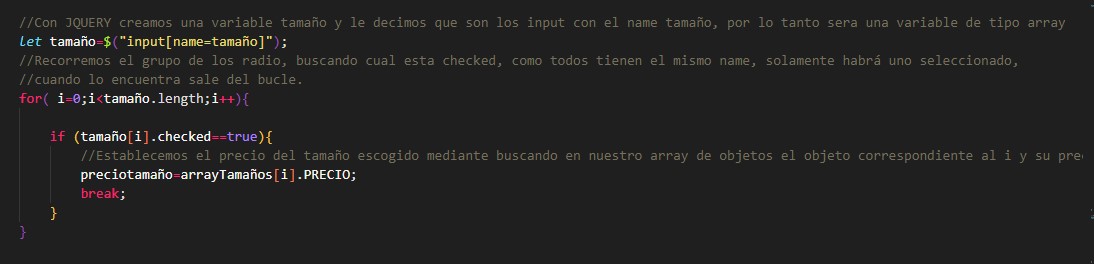


La formación de los input radio es mucho más sencilla gracias a los append y appendTo.

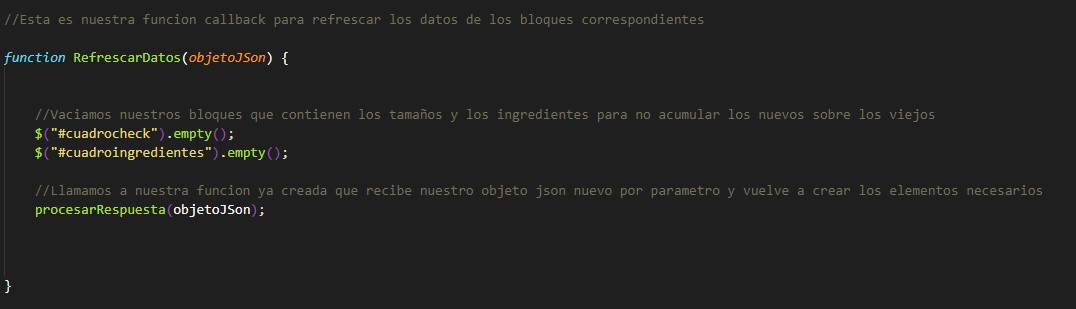
La formación de los input checkbox se hace de forma análoga a esta:



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:



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:



Interfaz de usuario gráfica

Descripción generada automáticamente

**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.**