

ACTIVIDAD 1

DOM y Formularios

Desarrollo web en Entorno Cliente

Andrea Adelaila

Ixchel López Alegre

David López Castellanos

Índice

Contextualización	2
Requerimiento 1.....	3
Estructura general del formulario	3
<i>Sobre la estructura organizativa</i>	4
fieldSetContacto: Datos personales	5
fieldsetServicio: Radio Buttons e imágenes	7
fieldSetCita: Select y TextArea	10
fieldSetNewsLetter: Checkbox	13
fieldSetAcciones: Acciones del formulario	17
Requerimiento 2.....	23
Estructura general	23
Validación formulario	26

Contextualización

Link al repositorio de github: <https://github.com/Davich7/GrupoTrabajo2-DAW>

En esta actividad todos hemos realizado todos los requerimientos, luego los hemos compartido y comparado.

Hemos tenido ciertos problemas iniciales con git y hemos creado un repositorio nuevo donde hemos vuelto a subir la actividad cuando estaba al 80%.

En ambos requerimientos hemos usado una estructura similar usando en el Requerimiento 1 **únicamente de JavaScript** y en el Requerimiento 2 una estructura con un esqueleto parecido pero a **través de HTML**.

Requerimiento 1

Hemos creado un formulario solo a través de la modificación del árbol dom, sin HTML a través del fichero script.js. A continuación, hemos realizado una validación simple solo con campos requeridos en el fichero validación.js. Hemos creado una hoja de estilo a partir de la cual hemos realizado el css de ambos requerimientos, aunque con modificaciones.

Estructura general del formulario

El formulario tiene como hijo un fieldset llamado fieldsetGeneral, el cual se divide en cinco secciones:

- **fieldsetContacto:** Aparecen los elementos y atributos para la sección de datos de contacto.
- **fieldsetServicio:** Se incluyen las diferentes opciones que se ofrecen en el servicio. Aquí encontraremos los Radio Buttons y las imágenes.
- **fieldsetCita:** Encontramos el select para seleccionar el lugar de la cita y el textarea para incluir alguna información adicional sobre la cita.
- **fieldsetNewsletter:** En esta sección se encuentran los checkbox, aunque no son obligatorios para el envío del formulario.
- **fieldsetAcciones:** Aquí encontramos el checkbox para aceptar términos y condiciones. Aparecen los botones para enviar y resetear el formulario.

```

/*
===== creacion de la estructura general del formulario =====
*/
const mainDiv = document.querySelector(".container");

//Formulario
const form = document.createElement("form");
form.setAttribute("action", "#");
form.setAttribute("method", "post");
mainDiv.appendChild(form);

//Fieldset general
const fieldsetGeneral = document.createElement("fieldset");
form.appendChild(fieldsetGeneral);

const leyendaFormulario = document.createElement("legend");
leyendaFormulario.textContent = "Pide tu cita";
leyendaFormulario.classList.add("form__titulo");
fieldsetGeneral.appendChild(leyendaFormulario);

//fieldset de datos personales para contacto
const fieldsetContacto = document.createElement("fieldset");
fieldsetContacto.setAttribute("id", "contacto");
fieldsetGeneral.appendChild(fieldsetContacto);

```

```

//fieldset con las opciones del servicio
const fieldsetServicio = document.createElement("fieldset");
fieldsetServicio.setAttribute("id", "servicio");
fieldsetGeneral.appendChild(fieldsetServicio);

//fieldset con las opciones de la cita
const fieldsetCita = document.createElement("fieldset");
fieldsetCita.setAttribute("id", "cita");
fieldsetGeneral.appendChild(fieldsetCita);

//fieldset para la subscripción al newsletter
const fieldsetNewsLetter = document.createElement("fieldset");
fieldsetNewsLetter.setAttribute("id", "newsletter");
fieldsetGeneral.appendChild(fieldsetNewsLetter);

//fieldset de los botones de Submit
const fieldsetAcciones = document.createElement("fieldset");
fieldsetAcciones.setAttribute("id", "acciones");
fieldsetGeneral.appendChild(fieldsetAcciones);

```

Sobre la estructura organizativa

Dentro de cada fieldset podemos encontrar varios divs que hacen la función de filas y columnas. Su objetivo es generar una estructura que ayude a la disposición de los elementos con el css. De esta manera estamos creando un esqueleto para cada sección que luego será tomada por el archivo style.css. Hemos seguido una lógica similar para el requerimiento 2.

Cada variable la hemos nombrado siguiendo esta nomenclatura:

- Filas: “row” + número de fila + tres primeras letras del id de la sección
- Columnas: “r” + número de fila + “c” + número de columna + tres primeras letras del id de la sección

Así, la primera fila de la sección Servicios se contendrá en la variable row1Serv. Las dos columnas de esa fila serán r1c1Serv y r1c2Serv.

Los campos del formulario se crearán dentro de su div correspondiente dependiendo de la disposición deseada.

fieldSetContacto: Datos personales

El fieldsetContacto es el nodo padre de esta estructura.

Tus datos de contacto

Introduce tus datos:

DNI

Nombre

Apellidos

Email

Teléfono

La estructura de la sección datos personales consta de un label general (“*Tus datos de contacto*”) y dos columnas compuestas por tres filas cada una. En la columna izquierda contamos con un label asignado a dicha columna, del cual hemos prescindido en la columna derecha por no ser necesario (“*Introduce tus datos*”). El contenedor llamado *r1c1Con* está asociado al label *labelContacto*.

```

//---Primer contenedor general---//

/* Crearemos primero los contenedores de la columna izquierda y luego la derecha.
 * Con un poco de css los situaremos
 */

//Creación de la Leyenda del FieldSet Contacto
const leyendaContacto = document.createElement("legend");
leyendaContacto.textContent = "Tus datos de contacto";
fieldsetContacto.appendChild(leyendaContacto);

// div de la 1ª fila que servirá de contenedor general
const row1Con = document.createElement("div");
row1Con.classList.add("fieldset-row");
fieldsetContacto.appendChild(row1Con);

// div de la 1ª columna que contendrá DNI, Nombre y Apellidos
const r1c1Con = document.createElement("div");
r1c1Con.classList.add("fieldset-column");
row1Con.appendChild(r1c1Con);

// Label de la 1ª columna que servirá de label para todos los datos
const labelContacto = document.createElement("label");
labelContacto.textContent = "Introduce tus datos:";
r1c1Con.appendChild(labelContacto);

```

Las otras dos filas de dicha columna constan con un label enlazado a su caja de texto correspondiente en cada una (en la primera caja de texto el label “DNI” va asociado a la caja de texto donde introduciremos nuestro DNI). De esta manera observamos como cada input, en este caso *inpCon1* es hijo del nodo *rowCon1*, que a su vez es hijo de *r1c1Con*. Esta estructura se repite en toda la sección de datos personales.

```
//---DNI---//

// Contenedor de la 1ª caja de texto
const rowCon1 = document.createElement("div");
rowCon1.classList.add("inline-column");
r1c1Con.appendChild(rowCon1);

// Label de la 1ª caja de texto
const labelCon1 = document.createElement("label");
labelCon1.setAttribute("id", "labelCon1");
labelCon1.setAttribute("for", "dni");
labelCon1.textContent = "DNI ";

// Input de la 1ª caja de texto
const inpCon1 = document.createElement("input");
inpCon1.setAttribute("type", "Text");
inpCon1.setAttribute("id", "dni");
inpCon1.setAttribute("placeholder", "DNI");
inpCon1.setAttribute("class", "form-control")
inpCon1.setAttribute("required", "true");

rowCon1.appendChild(labelCon1);
rowCon1.appendChild(inpCon1);
```

En el atributo del label de la caja de texto “for”, se asocia dicho label al id de la caja de texto correspondiente para que **al pulsar sobre el label nos dirija a la caja de texto adecuada**.

fieldsetServicio: Radio Buttons e imágenes

El fieldsetServicio el nodo padre de esta estructura.

¿Para que servicio quieres pedir cita?

Elige el tipo de servicio

- ☐ Primera cita de evaluación
- ☐ Terapia con psicólogo
- ☐ Sesión de coaching
- ☐ Cita con psiquiatra



Elige un centro para la cita

- ☐ Paseo de la Castellana 234
(Madrid)
- ☐ C/Santa Maria de la Cabeza 5
(Madrid)
- ☐ C/Doctor Cañadas (Cercedilla)
- ☐ C/Guindales 7 (Alcorcón)

Se ha creado una leyenda para la sección y una fila (*rowServ1*, *rowServ2*) para cada Radio Button con una imagen.

```
/*
===== Creación de la sección Servicio =====
*/

//----- Estructura general de Servicio -----//

//Creación de la Leyenda del FieldSet Servicio
const leyendaServicio = document.createElement("legend");
leyendaServicio.textContent = "¿Para que servicio quieres pedir cita?";
fieldsetServicio.appendChild(leyendaServicio);
```

Radio button

En este formulario se han incluido dos Radio Buttons similares, por lo que solo se va a describir uno de ellos.

Primero hemos creado un contenedor(*row1Serv*) que incluye una etiqueta, los radio button y las etiquetas respectivas.

Dentro de este div se ha creado otro (*r1c1Serv*) que contiene los radio button y la etiqueta *Elija el tipo de servicio*. Este contenedor es un nodo hijo del contenedor *row1Opc*.


```
//----- 1ª fila -----//
// div de la 1ª fila donde situaremos los radioButtons para elegir el tipo de Servicio
const row1Serv = document.createElement("div");
row1Serv.classList.add("fieldset-row");
fieldsetServicio.appendChild(row1Serv);

// div de la columna donde irá el Radio Button con los servicios a elegir
const r1c1Serv = document.createElement("div");
r1c1Serv.classList.add("fieldset-column");
row1Serv.appendChild(r1c1Serv);
```

Dentro de este div se ha creado otro (*r1c1Serv*) que contiene los radio button y la etiqueta *Elige el tipo de servicio*. Este contenedor es un nodo hijo del contenedor *row1Opc*.

```
/*-----*/
/---- Radio Button 1 ----/
/*-----*/

// Nodos de la etiqueta label (Tipos de servicio)
const labelRbTipo = document.createElement("label");
labelRbTipo.textContent = "Elige el tipo de servicio";
r1c1Serv.appendChild(labelRbTipo);

//----RB Opción 1 ----//
//Contenedor de la primera opción del RB
const rowRb1 = document.createElement("div");
rowRb1.classList.add("inline-column");
r1c1Serv.appendChild(rowRb1);

//Nodos
const inpRb1 = document.createElement("input");
inpRb1.setAttribute("type", "radio");
inpRb1.setAttribute("id", "rb1");
inpRb1.setAttribute("name", "rbtipo");
inpRb1.setAttribute("value", "primeracita");
inpRb1.setAttribute("required", "required");

//Etiqueta label
const labelRbTipo1 = document.createElement("label");
labelRbTipo1.setAttribute("id", "labelrb1");
labelRbTipo1.setAttribute("for", "rb1");
labelRbTipo1.textContent="Primera cita de evaluación";

//Vincular el radio button y label al div
rowRb1.appendChild(inpRb1);
rowRb1.appendChild(labelRbTipo1);
```

Para los radio button, se ha creado un contenedor para cada uno (nodos hijos del contenedor *r1c1Serv*) e incluye:

- Un elemento input de tipo radio button
- Una etiqueta asociada al radio button

Para evitar que se pueda seleccionar más de una opción dentro de un bloque de radio button (en este ejemplo, dentro de la categoría *Elija el tipo de servicio*), se ha asignado el mismo nombre a todos los radio button (*name= "rbtipo"*).

Imagen 1

```
//----- Imagen 1 -----//
// Div de la columna donde irá una imagen
const r1c2Serv = document.createElement("div");
r1c2Serv.classList.add("fieldset-column");
row1Serv.appendChild(r1c2Serv);
//Nodo de la img
const img1 = document.createElement("img");
img1.setAttribute("alt", "terapeuta_sentado_con_niño_al_lado");
img1.src = "images/terapeuta.jpg";
img1.width = "200";
//Vincular la img al div
r1c2Serv.appendChild(img1);
```

La primera imagen está situada en el *fieldsetServicio*. El contenedor que incluye la imagen (*r1c2Serv*) está situado al lado de la sección checkbox (*fila 1, columna 2*), y se encuentra dentro del mismo div de esta sección (*row1Serv*).

Como hemos indicado antes, la segunda fila con sus correspondientes dos columnas contienen otro radio button y otra imagen solo que con la disposición contraria (Dentro de la segunda columna(*row2Serv*): primero la imagen en la primera columna (*r2c1Serv*), luego el segundo Radio Button en la segunda columna(*r2c2Serv*)).

fieldsetCita: Select y TextArea

El *fieldsetCita* es el nodo padre de esta estructura.

¿Cuándo quieres tener tu cita?

Selecciona la franja que se adecúe más a tu horario:

Selecciona un horario ▼

¿Algún comentario que quieras añadir sobre tu cita?

Escribe aquí: 



Su estructura está compuesta de un elemento legend y de dos filas, todo asociado al *fieldsetCita* definido en la sección general del formulario. En la primera fila incluimos dos columnas. Dichas columnas son nodos padre de los nodos derivados del select y textarea.

```
/*
===== Creación de la sección Cita =====
*/

//leyenda de la sección Cita
const leyendaCita = document.createElement("legend");
leyendaCita.textContent = " ¿Cuándo quieres tener tu cita? ";
fieldsetCita.appendChild(leyendaCita);

//-----PRIMERA FILA DONDE IRAN EL SELECT Y EL TEXTAREA-----/

//primera fila (div para formato)
const row1Cit = document.createElement("div");
row1Cit.classList.add("fieldset-row");
fieldsetCita.appendChild(row1Cit);

//primera columna(div para formato)
const r1c1Cit = document.createElement("div");
r1c1Cit.classList.add("fieldset-column");
row1Cit.appendChild(r1c1Cit);
```

Select: Cita

La primera sección contiene el select. Hemos creado un contenedor *row1Cit* hijo de *fieldsetCita*. El contenedor *r1c1Cit* es hijo a su vez de *row1Cit* donde encontramos el select.

```
/*-----*/
/-----SELECT-----/
/*-----*/

// Seleccionador de la cita
const labelSelect = document.createElement("label");
labelSelect.setAttribute("for", "cita");
labelSelect.textContent = "Selecciona la franja que se adecúe más a tu horario:";
r1c1Cit.appendChild(labelSelect);

const selectCita = document.createElement("select");
selectCita.setAttribute("name", "cita");
selectCita.setAttribute("id", "cita");
selectCita.setAttribute("required", "required");
r1c1Cit.appendChild(selectCita);
```

Para crear el select hemos creado diversos nodos hijos de *r1c1Cit*, contenedor de sus elementos, textos y atributos.

```
// Opciones del Select
const placeholder = document.createElement("option");
placeholder.setAttribute("value", "");
placeholder.textContent = "Selecciona un horario";

const opcionA = document.createElement("option");
opcionA.setAttribute("value", "res1");
opcionA.textContent = "De 9:00 a 11:00";
```

A continuación relacionamos cada opción al contenedor del select *selectCita*.

```
selectCita.appendChild(placeholder);
selectCita.appendChild(opcionA);
selectCita.appendChild(opcionB);
selectCita.appendChild(opcionC);
selectCita.appendChild(opcionD);
```

TextArea

El nodo padre de los nodos de la sección del textarea es *row1Cit*. Creamos una segunda fila *r1c2Cit* que será contenedor de la caja de texto.

```
//----Segunda columna que va a contener el TEXTAREA ----//

//Segunda columna(div para formato)
const r1c2Cit = document.createElement("div");
r1c2Cit.classList.add("fieldset-column");
row1Cit.appendChild(r1c2Cit);
```

Imagen 3

La tercera imagen está en el contenedor *row2Cit* y ocupará toda la fila. Es hija del nodo padre *fieldsetCita*. Dentro de este contenedor encontramos la imagen y sus atributos.

```
//Div de la 2ª fila donde irá otra imagen

//----IMAGEN 3----//

//segunda fila (div para formato)
const row2Cit = document.createElement("div");
row2Cit.classList.add("fieldset-row");
fieldsetCita.appendChild(row2Cit);

//Nodo de la img
const img3 = document.createElement("img");
img3.setAttribute("alt", "dibujo_cabeza_psicologos");
img3.src = "images/cabeza-psi.jpg";
img3.width = "400";
//Vincular la img al div
row2Cit.appendChild(img3);
```

fieldsetNewsletter: Checkbox

En esta sección encontraremos el checkbox del newsletter. Su selección será opcional.

¿Te gustaría subscribirte a nuestro Newsletter?

¿Qué contenido quieres recibir?

- ☐ Avisos sobre ofertas y descuentos
- ☐ Información sobre cursos
- ☐ Noticias sobre Salud Mental
- ☐ Actualización de nuestro podcast
- ☐ Alerta de nuevos talleres y actividades



La estructura general de la sección de checkbox consta de una etiqueta y cinco checkbox. Se ha creado un contenedor que incluye la etiqueta (*¿Te gustaría subscribirte a nuestro newsletter?*) y los checkbox con sus etiquetas asociadas, declarado como constante (*row1New*). Este contenedor es un nodo hijo de *fieldsetNewsLetter*.

Checkbox: Newsletter

Dentro de este div se ha creado otro (*r1c1New*) que contiene los checkbox. Este contenedor es un nodo hijo del contenedor *row1New*.

Finalmente, se ha creado la etiqueta *¿Qué contenido quieres recibir?* (nodo hijo del contenedor *r1c1New*).

```

641  /*
642  ===== Creación de la sección Newsletter =====
643  */
644
645
646  //leyenda de la sección Newsletter
647  const leyendaNewsLetter = document.createElement("legend");
648  leyendaNewsLetter.textContent = "¿Te gustaría subscribirte a nuestro Newsletter?";
649  fieldsetNewsLetter.appendChild(leyendaNewsLetter);
650
651  /*-----/
652  /-----CheckBox-----/
653  /-----*/
654
655  // div de la 1ª fila que contiene las el checkbox para el newsletter
656  const row1New = document.createElement("div");
657  row1New.classList.add("fieldset-row");
658  fieldsetNewsLetter.appendChild(row1New);
659
660
661
662  // div de la 1ª columna donde irán las opciones
663  const r1c1New = document.createElement("div");
664  r1c1New.classList.add("fieldset-column");
665  row1New.appendChild(r1c1New);
666
667  // Nodos de la etiqueta de la columna donde se encuentran las opciones
668  const labelCheckbox = document.createElement("label");
669  labelCheckbox.textContent = "¿Qué contenido quieres recibir?";
670  r1c1New.appendChild(labelCheckbox);
671

```

Para los checkbox, se ha creado un contenedor para cada uno (nodos hijos del contenedor *r1c1New*) e incluye:

- Un elemento input de tipo checkbox
- Una etiqueta asociada al checkbox

Estos elementos son nodos hijos del contenedor respectivo:

- Elemento contenedor para el primer checkbox: *rowCbox1* hijo de *r1c1New*. Y así sucesivamente para cada checkbox.

```

// Contenedor del 1er Checkbox
const rowCbox1 = document.createElement("div");
rowCbox1.classList.add("inline-column");
r1c1New.appendChild(rowCbox1);

// Nodos del 1er Checkbox
const inpCbox1 = document.createElement("input");
inpCbox1.setAttribute("type", "checkbox");
inpCbox1.setAttribute("id", "cbox1");
inpCbox1.setAttribute("value", "ofertas");

//etiqueta label para el checkbox
const labelCbox1 = document.createElement("label");
labelCbox1.setAttribute("id", "labelcbox1");
labelCbox1.setAttribute("for", "cbox1");
labelCbox1.textContent = "Avisos sobre ofertas y descuentos";

//Vincular el checkbox y label al div
rowCbox1.appendChild(inpCbox1);
rowCbox1.appendChild(labelCbox1);

```

Imagen 4

Creamos el contenedor *r1c2New* que será el div de la segunda columna donde solo colocaremos la 4ª y última imagen. Este nodo es hijo de *row1New*.

```

//-----2ª Columna con imagen -----//

// div de la 2ª columna donde irá la imagen
const r1c2New = document.createElement("div");
r1c2New.classList.add("fieldset-column");
row1New.appendChild(r1c2New);

//----IMAGEN 4----//
//Nodo de la img
const img4 = document.createElement("img");
img4.setAttribute("alt", "newsletter_llegando_a_pc");
img4.src = "images/newsletter.jpg";
img4.width = "200";
//Vincular la img al div
r1c2New.appendChild(img4);

```


fieldsetAcciones: Acciones del formulario

El fieldsetAcciones es el nodo padre de esta estructura.

☐ Acepto los términos y condiciones

Enviar

Borrar

En este fieldSet encontraremos el checkbox obligatorio de Términos y condiciones y también los botones de acciones del formulario.

Checkbox:Términos y condiciones

Creamos un div (*row1Acc*) donde encontraremos el checkbox obligatorio para enviar la petición del formulario.

```
//  
// div | donde irán los términos y condiciones  
const row1Acc = document.createElement("div");  
row1Acc.classList.add("fieldset-row");  
fieldsetAcciones.appendChild(row1Acc);  
  
//checkbox terminos y condiciones  
const terminos = document.createElement("input");  
terminos.setAttribute("type", "checkbox");  
terminos.setAttribute("name", "terminos");  
terminos.setAttribute("id", "cbox");  
terminos.setAttribute("value", "s");  
terminos.setAttribute("class", "terminos");  
terminos.setAttribute("required", "required");  
row1Acc.appendChild(terminos);
```

Uno de los atributos del input *terminos* del que consta la checkbox, es *required* y tendrá que ser validado para poder validar el formulario.

```
const labelTerminos = document.createElement("label");
labelTerminos.setAttribute("for", "terminos");
labelTerminos.setAttribute("class", "terminos");
labelTerminos.textContent = "Acepto los términos y condiciones";
row1Acc.appendChild(labelTerminos);
```

Crearemos a continuación un label y lo asociaremos al checkbox.

Botones de Enviar y Borrar



Primero encontraremos el nodo *row2Acc* que es hijo del nodo padre *fieldsetAcciones*, constituirá la segunda fila de dicho *fieldset*.

A continuación creamos un div de tipo *fieldset-column* para crear una primera columna donde colocaremos el botón Enviar, *submit*.

```
// div donde irán los términos y condiciones
const row2Acc = document.createElement("div");
row2Acc.classList.add("fieldset-row");
fieldsetAcciones.appendChild(row2Acc);

//Primer div para el botón de envío
const r2c1Acc = document.createElement("div");
r2c1Acc.classList.add("fieldset-column");
row2Acc.appendChild(r2c1Acc);

//boton de envio
const submit = document.createElement("button");
submit.setAttribute("type", "submit");
submit.setAttribute("name", "submit");
submit.setAttribute("id", "submit");
submit.textContent = "Enviar";
submit.classList.add("cta");
r2c1Acc.appendChild(submit);
```

El botón de submit tiene asociado un evento que añade la clase “validar” a todos los elementos del formulario obligatorios. Esta clase tiene formatos asociados que muestran

claramente todos los campos obligatorios que no han superado la validación por defecto del formulario. Se encuentra en otro fichero JavaScript llamado validación.js. Este evento de tipo “click” se ha incluido en una función anónima que se ejecuta cuando carga la página web (window.onload). Realizaremos una validación más compleja en el Requerimiento 2.

```
1 window.onload = function () {  
2     submit.addEventListener("click", () => {  
3         const elementosObligatorios = document.querySelectorAll("[required]")  
4         for (let elemento of elementosObligatorios) {  
5             elemento.classList.add("validar");  
6         }  
7     });  
8 }
```

Esta acción la hacemos visible al usuario a través de css:

```

.validar:invalid {
  background: #f8aaaa;
  border: 1px solid #d60303;
  outline: 1px solid #d60303;
}

input.validar:invalid {
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  width: 0.8rem;
  height: 0.8rem;
  background: #f8aaaa;
  border: 1px solid #d60303;
  outline: 1px solid #d60303;
}

```

```

input[type="radio"].validar:invalid {
  outline: none;
  border-radius: 50%;
}

input[type="text"].validar:invalid {
  width: 100%;
  height: 1.4rem;
}

input[type="checkbox"].validar:invalid {
  width: 0.7rem;
  height: 0.7rem;
}

```

Tus datos de contacto

Introduce tus datos:

DNI

Nombre

Apellidos

Email

Teléfono



Completa este campo

¿Para que servicio quieres pedir cita?

Elige el tipo de servicio

- ☐ Primera cita de evaluación
- ☐ Terapia con psicólogo
- ☐ Sesión de coaching
- ☐ Cita con psiquiatra



Elige un centro para la cita

- ☐ Paseo de la Castellana 234 (Madrid)
- ☐ C/Santa Maria de la Cabeza 5 (Madrid)
- ☐ C/Doctor Cañadas (Cercedilla)
- ☐ C/Guindales 7 (Alcorcón)

¿Cuándo quieres tener tu cita?

Selecciona la franja que se adecúe más a tu horario:

Selecciona un horario ▼

¿Algún comentario que quieras añadir sobre tu cita?

Escribe aquí: 

Como hemos indicado, el campo de checkbox del newsletter no es obligatorio, a diferencia del checkbox de términos y condiciones.

¿Te gustaría subscribirte a nuestro Newsletter?

¿Qué contenido quieres recibir?

- ☐ Avisos sobre ofertas y descuentos
- ☐ Información sobre cursos
- ☐ Noticias sobre Salud Mental
- ☐ Actualización de nuestro podcast
- ☐ Alerta de nuevos talleres y actividades



☐ Acepto los términos y condiciones

Enviar

Borrar

Por último, el botón reset se ocupará de borrar todos los campos.

Se mostrará en el HTML final con el texto: "Borrar"

```
//Segundo div para el botón de reseteo
const r2c2Acc = document.createElement("div");
r2c2Acc.classList.add("fieldset-column");
row2Acc.appendChild(r2c2Acc);

//boton de reseteo
const reset = document.createElement("button");
reset.setAttribute("type", "reset");
reset.setAttribute("name", "reset");
reset.setAttribute("id", "reset");
reset.classList.add("res");
reset.textContent = "Borrar";
r2c2Acc.appendChild(reset);
```

Requerimiento 2

Estructura general

En esta actividad hemos practicado la validación de formularios trabajando en los diferentes archivos:

- El **HTML** (index.html) Como hemos indicado, hemos usado una estructura similar al esqueleto de la actividad 1. Estas son las principales diferencias con el Requerimiento1:
 - A la hora de la validación, nos hemos asegurado de que los “id” para los inputs sean sencillos para trabajar su validación.
 - Hemos incluido la etiqueta <p> vacía para mostrar mensajes de error cuando no se cumplen los criterios de validación.
 - Hemos colocado los RadioButtons necesarios para la actividad y hemos colocado el checkbox adecuado al requerimiento 2.
- El **JavaScript**, donde hemos incluido el código que impide el envío del formulario sin que se cumplan los criterios establecidos. Se ha utilizado un event listener para el envío del formulario, y varios para la validación instantanea del formulario ya que se comprueba si el input de ciertos campos es correcto a medida que se rellena el formulario. Todos los event listeners se han incluido en una función anónima que se ejecuta cuando carga la página web (*window.onload*). Más adelante analizamos en detalle cada función.

```
// Función window.onload

/*Creamos una función window.onload para asegurarnos de que el html se ha cargado completamente antes de
pasar a la validación del formulario.*/

window.onload = function () {

    //-----Event listeners-----//

    //validacion del formulario completo
    submit.addEventListener("click", validarFormulario);

    //validacion inmediata de nombre
    nombre.addEventListener("keyup", validarNombre);

    //validacion inmediata de apellidos
    apellidos.addEventListener("keyup", validarApellidos);

    //validacion inmediata de la direccion
    direccion.addEventListener("keyup", validarDireccion);

    //validacion inmediata del telefono
    telefono.addEventListener("keyup", validarTlf);

    //validacion inmediata del email
    email.addEventListener("keyup", validarEmail);

}
```



```

// validacion inmediata del minimo de ingredientes
// y actualizacion del precio
const ingredientesChkboxes = document.querySelectorAll(
  '#opciones-pizza input[type="checkbox"]'
);

ingredientesChkboxes.forEach((chkbox) => {
  chkbox.addEventListener("change", validarMinIngredientes);
  chkbox.onchange = calcularPrecio;
});

//validacion inmediata de los radio button MASA
const masaRadioButton = document.getElementsByName("masa");
for (var i = 0; i < masaRadioButton.length; i++) {
  masaRadioButton[i].addEventListener("click", validarMasa);
}

//validacion inmediata de los radio button TAMANIO
//y actualizacion del precio

const tamañoRadioButton = document.getElementsByName("tamaño");
for (var i = 0; i < tamañoRadioButton.length; i++) {
  tamañoRadioButton[i].addEventListener("click", validarTamaño);
  tamañoRadioButton[i].onchange = calcularPrecio;
}

// validacion inmediata de seleccion de restaurante
// valida cada vez que cambia la seleccion
restaurante.addEventListener("change", validarRestaurante);

//validacion inmediata de los terminos y condiciones
const terminos = document.getElementById("terminos");
terminos.addEventListener("click", validarTerminos);
};

```

- El **CSS**, además de aportar el diseño al HTML, es clave para obtener una buena UX, puesto que si el valor introducido en algún campo no coincide con el criterio establecido, la apariencia de dichos campos varía, y viceversa, es decir, cuando se corrige el input, la apariencia de error desaparece. Esto lo hemos conseguido creando una clase llamada `invalido`. Si la validación del elemento no es `true`, le añadimos la clase `invalido`, y en caso contrario se elimina si había sido previamente añadida en la función de validación de ese campo en JavaScript.

Lo hemos aplicado en todos los campos en los que hemos trabajado la validación, por lo que mostramos a modo representativo cómo se ha aplicado para el input de tipo radiobutton (Elige tu masa).

```
// Iteramos por los radio button para añadir o quitar la clase "invalido"
for (var j = 0; j < masaRadioButton.length; j++) {
    if (!valido) {
        masaRadioButton[j].classList.add("invalido");
    } else if (masaRadioButton[j].classList.contains("invalido")) {
        masaRadioButton[j].classList.remove("invalido");
    }
}

//Editamos el mensaje de error segun el resultado de la validacion
if (valido) mensajeErrorMasa.textContent = "";
else mensajeErrorMasa.textContent = "Elige el tipo de masa";

return valido;
```



Elige el tipo de masa

- ☐ Fina
- ☐ Normal
- ☐ Extra
- ☐ Extra con bordes

rellenos de queso

Elige el tipo de masa

Elige el tamaño

- ☐ Pequeña - 5€
- ☐ Mediana - 10€
- ☐ Grande - 15€

Elige el tamaño de la pizza



Elige el tipo de masa

- ☐ Fina
- ☐ Normal
- ☒ Extra
- ☐ Extra con bordes

rellenos de queso

Elige el tamaño

- ☐ Pequeña - 5€
- ☐ Mediana - 10€
- ☐ Grande - 15€

Elige el tamaño de la pizza

Para que el cambio de apariencia y el mensaje de error no aparezca solamente cuando se hace clic en el botón submit, hemos añadido disparadores de eventos adicionales sobre los input en cuestión: en el caso de los input de tipo text, cuando se levantan las teclas del teclado, y en los checkboxes y botones radio, al seleccionar o deseleccionar el elemento.

Validación formulario

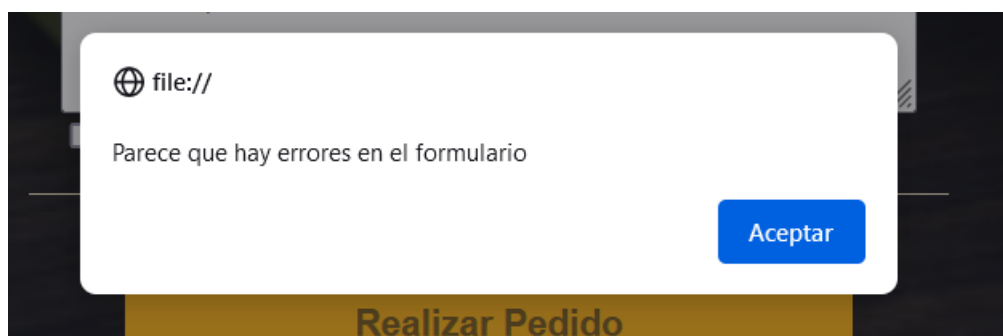
A diferencia del Requerimiento 1, se ha decidido atar la validación del formulario al evento click sobre el botón de envío en lugar de al evento submit del elemento formulario para mejorar la experiencia del usuario, ya que el evento click precede al evento submit. Si la validación estuviese ligada al envío del formulario, el usuario tendría que superar la validación por defecto de html5 antes que la nuestra (puesto que podría haber realizado cambios, para luego averiguar que no eran suficientes).

Se trata de una función general cuyo propósito es comprobar que el resto de las funciones de validación devuelven *true*. Si alguna de las funciones devuelve *false* el valor de la variable “valido” será *false*.

```
/*
 *===== Validacion formulario =====
 */
function validarFormulario(event) {
  let valido = true;
  if (!validarNombre()) valido = false;
  if (!validarApellidos()) valido = false;
  if (!validarDireccion()) valido = false;
  if (!validarTlf(event)) valido = false;
  if (!validarMinIngredientes()) valido = false;
  if (!validarEmail()) valido = false;
  if (!validarRestaurante()) valido = false;
  if (!validarMasa()) valido = false;
  if (!validarTamano()) valido = false;
  if (!validarTerminos()) valido = false;
```

A continuación, si la variable *valido* devuelve *false* se mostrará un pop up con un mensaje de error y se evitará el comportamiento por defecto del botón de submit (es decir, el envío del formulario) con el método `preventDefault()`. Si el valor de *valido* es *true* saltará una ventana de diálogo que mostrará el resultado de la función *calcularPrecio()*.

```
if (!valido) {
  alert("Parece que hay errores en el formulario");
  event.preventDefault();
}
```



Validación de Nombre y Apellidos

Los campos *Nombre* y *Apellidos* se validan a través de una función que comprueba que el input introducido por el usuario:

- Contiene algún carácter, es decir, no se compone solo de espacios.
- Comienza con mayúscula.
- No contiene caracteres no alfabéticos.

De esta manera si cumple todos los dichos requisitos, la función devuelve true y si no devolverá false.

Las funciones de *validarNombre* y *validarApellidos* son prácticamente iguales, así que comentaremos aquí solo *validarApellidos()*.

```
function validarApellidos() {  
  const mensajeErrorApellidos = document.querySelector(".apellidos-error");  
  const apellidosUsuario = apellidos.value.replace(/\s/g, "");  
  const pattern =  
    /^[A-Z][a-zA-ZÀ-ÿ\u00f1\u00d1]+(\s*[a-zA-ZÀ-ÿ\u00f1\u00d1]*)*[a-zA-ZÀ-ÿ\u00f1\u00d1]+$/;  
  const valido = pattern.test(apellidosUsuario);  
  if (!valido) {  
    apellidos.classList.add("invalido");  
    mensajeErrorApellidos.textContent = "Introduce un apellido válido";  
  } else {  
    if (apellidos.classList.contains("invalido"))  
      apellidos.classList.remove("invalido");  
    mensajeErrorApellidos.textContent = "";  
  }  
  return valido;  
}
```

En ambas funciones se ha usado el mismo patrón, que se comparará con el input sin los espacios. Al incluir primero `^[A-Z]` nos aseguramos que la primera letra introducida se trate de una mayúscula.

El resto del patrón se asegura de que puedan ser introducidos caracteres especiales como acentos y la ñ-Ñ, pero no otros caracteres como "\$", "@", etc...

(Este patrón ha sido obtenido de: <https://es.stackoverflow.com/questions/81041/expresion-regular-para-validar-letras-con-acentos-y-%C3%B1>)

Nombre

ana

Introduce un nombre válido

Apellidos

Apellidos

Introduce un apellido válido

Podemos observar que tanto si no introduces nada (en *apellidos*) como si la primera letra del input es en minúscula se mostrará el mensaje de error.

Introduce tus datos

Nombre

Nombre

Introduce un nombre válido

Apellidos

López

En esta imagen observamos cómo al introducir una mayúscula como primera letra no nos sale error, y si solo introducimos espacios en Apellidos nos salta un mensaje de error.

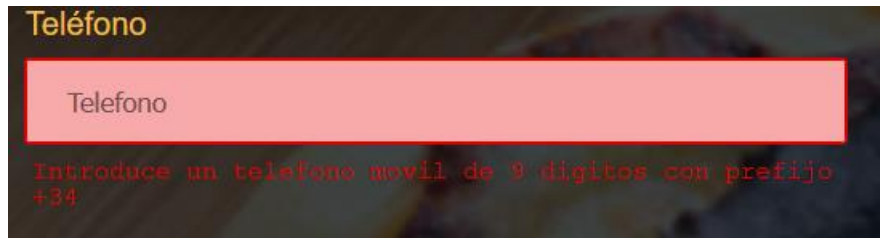
Validación del Teléfono

El campo teléfono verifica que el número introducido corresponde a un teléfono móvil español: se compone de un prefijo +34 seguido de nueve dígitos, el primero de los cuales ha de ser un 6. El campo acepta espacios entre los dígitos, pero no otros caracteres.

Entendemos que en un escenario real se deberían permitir números extranjeros, así como teléfonos fijos, pero nos ha parecido más útil como práctica aplicar un formato más restrictivo.

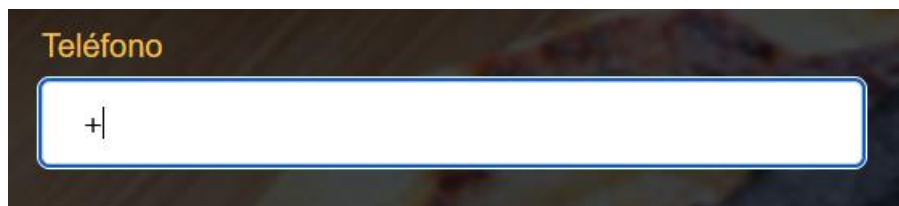
La validación se desata a raíz de dos tipos de evento:

- Al **enviar** el formulario

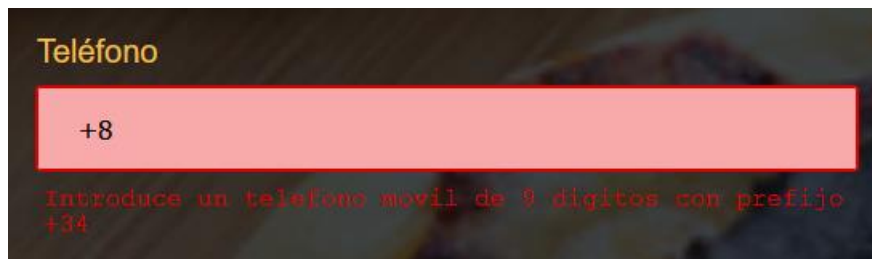


El formulario muestra el título "Teléfono" en amarillo. El campo de entrada, que contiene el texto "Telefono", está resaltado con un recuadro rojo. Debajo del campo, un mensaje de error en rojo indica: "Introduce un telefono movil de 9 digitos con prefijo +34".

- Al **soltar una tecla** dentro del campo

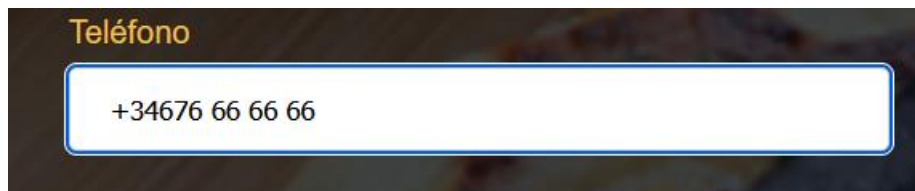


El formulario muestra el título "Teléfono" en amarillo. El campo de entrada, que contiene el texto "+", está resaltado con un recuadro azul.



El formulario muestra el título "Teléfono" en amarillo. El campo de entrada, que contiene el texto "+8", está resaltado con un recuadro rojo. Debajo del campo, un mensaje de error en rojo indica: "Introduce un telefono movil de 9 digitos con prefijo +34".

Admite además espacios de los que luego nos desharemos.



El formulario muestra el título "Teléfono" en amarillo. El campo de entrada, que contiene el texto "+34676 66 66 66", está resaltado con un recuadro azul.

La validación requerida cambia según el tipo de evento que la genera, por lo que necesitamos recoger el objeto del evento como argumento de la función (al llamar a un manejador de un evento, javascript incluye el objeto evento en la llamada, por lo que podemos acceder al él si lo nombramos en la definición de la función).

```
function validarTlf(evento) {
  const mensajeErrorTfno = document.querySelector(".telefono-error");

  let inputUsuario = telefono.value.replace(/\s/g, ""); //elimina todos los espacios del input
  const patternString = "^\\+346[0-9]{1,8}$";
  let pattern = new RegExp(patternString);

  // si estamos validando al teclear, modificamos el patron
  // para ajustarse a la longitud del input del usuario
  if (evento.type === "keyup") {
    if (inputUsuario.length < 1) null;
    else if (inputUsuario.length < 5) {
      pattern = new RegExp(patternString.substring(0, inputUsuario.length + 2));
    } else if (inputUsuario.length < 12) {
      pattern = new RegExp(patternString.slice(0, -1));
    }
  }

  //comparamos el telefono introducido con el formato esperado
  const valido = pattern.test(inputUsuario);
  if (!valido) {
    telefono.classList.add("invalido");
    mensajeErrorTfno.textContent =
      "Introduce un telefono movil de 9 digitos con prefijo +34";
  } else {
    if (telefono.classList.contains("invalido"))
      telefono.classList.remove("invalido");
    mensajeErrorTfno.textContent = "";
  }

  return valido;
}
```

Validación de la Dirección

El campo *dirección* se valida mediante una función que verifica que el input cumple los siguientes requisitos:

- Contiene caracteres, es decir, no se ha rellenado únicamente de espacios.
- Comienza con mayúscula y contiene, al menos, un número.
- La longitud mínima es de 20 caracteres. La longitud máxima es de 150, la cual se ha establecido mediante el atributo “maxLength” en HTML.

Esta función devuelve “true” si el input es válido, y “false” si no.

```
function validarDireccion() {
  //Seleccionamos el primer nodo hijo que deriva del nodo <p></p> cuya clase es "mensaje-error direccion-error"
  const mensajeErrorDireccion = document.querySelector(".direccion-error");
  let valido = false;
  const caracteresString = "[A-Z]{1,}[0-9]{1,}";
  let caracteres = new RegExp(caracteresString);
}
```

Para los caracteres que pueden incluirse en el input, se ha creado una expresión regular que indica que la dirección debe comenzar con mayúscula y debe contener al menos un número.

```
let direccion = document.getElementById("direccion");
let input = direccion.value.trim();
input.replace(/ {2,}/g, " ");
```

Se ha creado una variable “direccion” cuyo contenido es el elemento dirección del documento HTML. Se eliminan los espacios al comienzo y al final de la cadena de texto, y los espacios duplicados dentro de la misma.

```
if (input.length < 20 && caracteres.test(input) == false) {
    valido = false;
} else {
    valido = true;
}
```

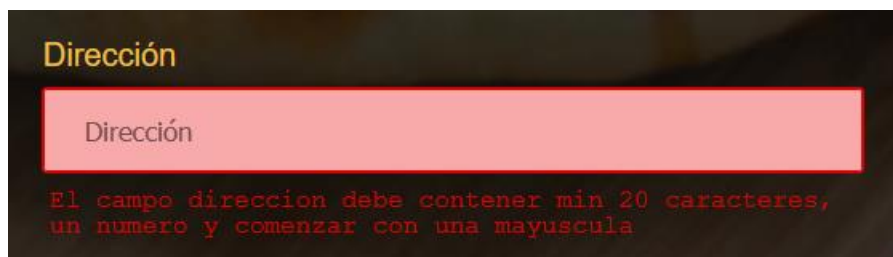
Para que la dirección sea válida debe contener más de 20 caracteres y el patrón establecido por la expresión regular. En el caso de que sea válida, se cambia el valor de la variable “valido” a true.

```
if (!valido) {
    direccion.classList.add("invalido");

    mensajeErrorDireccion.textContent =
        "El campo direccion debe contener min 20 caracteres, un numero y comenzar con una mayuscula";
} else {
    if (direccion.classList.contains("invalido")) {
        direccion.classList.remove("invalido");
        mensajeErrorDireccion.textContent = "";
    }
}

return valido;
}
```

En el caso de no ser válida, una vez se pulse el botón *Realizar pedido*, el campo *dirección* aparecerá así:



The screenshot shows a web form with a label "Dirección" in yellow. Below it is a text input field with a red border and a light red background. The text "Dirección" is faintly visible inside the field. Below the input field, there is a red error message: "El campo direccion debe contener min 20 caracteres, un numero y comenzar con una mayuscula".

Validación del Email

El campo *email* se valida mediante una función que verifica que el input cumple los siguientes requisitos:

- Elimina cualquier espacio introducido.
- Debe contener solo una "@".
- Permitir la inclusión: caracteres del abecedario, en mayúsculas o minúsculas, números, ".", "-" y "_"
- Debe contener al menos un punto.
- La "@" y el punto no pueden estar inmediatamente juntos.

Esta función devuelve "true" si el input es válido, y "false" si no.

```
const mensajeErrorEmail = document.querySelector(".email-error");
```

Además de la validación que se ejecuta al hacer clic en el submit, también se desencadena mientras se escribe en el campo y cuando se hace clic en el botón submit a través de la función `validarFormulario` como hemos explicado antes.

```
//validacion inmediata del telefono  
telefono.addEventListener("keyup", validarTlf);
```

Para realizar la validación hemos creado un patrón con expresiones regulares. Hemos considerado que una dirección sería válida si cumple con la siguiente estructura:

- Primer bloque: Desde el primer carácter hasta el que precede a la "@". Debe tener al menos un carácter en minúsculas, mayúsculas, numérico o un punto, guión o barra baja.
- Una "@"
- Segundo bloque: Desde el carácter que le sigue a la "@" y el anterior al ".": debe tener al menos un carácter en minúsculas, mayúsculas, numérico o un guión.
- Un ".".
- Tercer bloque: Detrás del punto puede haber 2, 3 o 4 caracteres en mayúsculas o minúsculas.

Eliminamos los espacios introducidos en el campo y comparamos el valor resultante con el patrón establecido mediante el **método test** de la clase RegExp.

```
const patternEmail = "[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,4}$";

let patronEmail = new RegExp(patternEmail);

//En primer lugar, eliminamos cualquier espacio introducido
let input = email.value.split(" ").join("");
const valido = patronEmail.test(input);

//Validamos el contenido final

if (!valido) {
  email.classList.add("invalido");
  mensajeErrorEmail.textContent =
    "El email introducido no tiene el formato correcto";
} else {
  if (email.classList.contains("invalido"))
    email.classList.remove("invalido");
  mensajeErrorEmail.textContent = "";
}

return valido;
}
```

Validación del CheckBox: Ingredientes

La validación de ésta sección consiste en verificar que al menos una de las 5 checkboxes de ingredientes está seleccionada. Se desata a raíz de dos eventos:

- Envío del formulario:

¿Cómo quieres tu RicoPizza?

Elige los ingredientes -
1€/ud

- ☐ Mezcla de quesos
- ☐ Champiñones
- ☐ Bacon
- ☐ Tomates cherry
- ☐ Pollo

Elige al menos un
ingrediente para tu
pizza



- Al hacer click en alguna de las checkboxes:

Elige los ingredientes -
1€/ud

- ☐ Mezcla de quesos
- ☐ Champiñones
- ☐ Bacon
- ☐ Tomates cherry
- ☒ Pollo

Elige los ingredientes -
1€/ud

- ☐ Mezcla de quesos
- ☐ Champiñones
- ☐ Bacon
- ☐ Tomates cherry
- ☐ Pollo

Elige al menos un
ingrediente para tu
pizza

```
*===== Validacion minimo ingredientes =====  
*/  
/**  
* Funcion que valida que hay al menos un ingrediente seleccionado  
* @returns true si al menos un ingrediente ha sido seleccionado, false si no  
*/  
  
function validarMinIngredientes() {  
  let valido = false;  
  const mensajeError = document.querySelector("#opciones-pizza p");  
  
  const ingredientesChkboxes = document.querySelectorAll(  
    '#opciones-pizza input[type="checkbox"]'  
  );  
  // iteramos por las checkboxes para ver si alguna esta marcada  
  // y actualizar el resultado de la validacion  
  for (let chkbox of ingredientesChkboxes) {  
    if (chkbox.checked) {  
      valido = true;  
      break;  
    }  
  }  
}
```

```

// iteramos por las checkboxes para añadir o quitar la clase "invalido"
// segun el resultado de la validacion
ingredientesChkboxes.forEach((checkbox) => {
  if (!valido) checkbox.classList.add("invalido");
  else if (checkbox.classList.contains("invalido"))
    checkbox.classList.remove("invalido");
});

// editamos el mensaje de error segun el resultado de la validacion
if (valido) mensajeError.textContent = "";
else mensajeError.textContent = "Elige al menos un ingrediente para tu pizza";

return valido; //devolvemos el resultado de la validacion
}

```

Validación Radio Button: Tipo de masa y Tamaño de la pizza

El formulario cuenta con dos listas de radio buttons, una corresponde con el *tipo de masa* y la otra, con el *tamaño de la pizza*. Debido a que la validación es igual para ambas, se va a explicar el código de una de ellas.

```

function validarTamanio() {
  let valido = false;
  const mensajeErrorTamanio = document.querySelector(".error-tamanio");
  const tamanioRadioButton = document.getElementsByName("tamanio");

```

Esta función verifica que se ha escogido un *tamaño de pizza*. Al igual que en el resto de validaciones, se ha creado un párrafo sin contenido en el que se insertará un mensaje de error en el caso de que no se haya seleccionado un radio button.

Se ha creado una variable de tipo Array que contiene aquellos radio buttons cuyo nombre es "tamanio" sin la ñ.

Iteramos por los radio buttons para comprobar si alguno está marcado. Para ello se ha utilizado un bucle for y la propiedad `.checked`.

```

for (var i = 0; i < tamanioRadioButton.length; i++) {
  if (tamanioRadioButton[i].checked) {
    valido = true;
    break;
  }
}

```

Si no hay algún radio button seleccionado, una vez se pulse el botón *Realizar pedido* aparecerá el mensaje de error y cambiará el color de los radio buttons:

Elige el tipo de masa	Elige el tamaño
<input type="radio"/> Fina	<input type="radio"/> Pequeña - 5€
<input type="radio"/> Normal	<input type="radio"/> Mediana - 10€
<input type="radio"/> Extra	<input type="radio"/> Grande - 15€
<input type="radio"/> Extra con bordes rellenos de queso	

Elige el tipo de masa	Elige el tamaño
<input checked="" type="radio"/> Fina	<input checked="" type="radio"/> Pequeña - 5€
<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Mediana - 10€
<input checked="" type="radio"/> Extra	<input checked="" type="radio"/> Grande - 15€
<input checked="" type="radio"/> Extra con bordes rellenos de queso	Elige el tamaño de la pizza
Elige el tipo de masa	

Cuando se selecciona uno de una lista:

Elige el tipo de masa	Elige el tamaño
<input type="radio"/> Fina	<input checked="" type="radio"/> Pequeña - 5€
<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Mediana - 10€
<input type="radio"/> Extra	<input checked="" type="radio"/> Grande - 15€
<input type="radio"/> Extra con bordes rellenos de queso	Elige el tamaño de la pizza

Cuando se selecciona uno de cada lista:

Elige el tipo de masa	Elige el tamaño
<input type="radio"/> Fina	<input type="radio"/> Pequeña - 5€
<input checked="" type="radio"/> Normal	<input checked="" type="radio"/> Mediana - 10€
<input type="radio"/> Extra	<input type="radio"/> Grande - 15€
<input type="radio"/> Extra con bordes rellenos de queso	

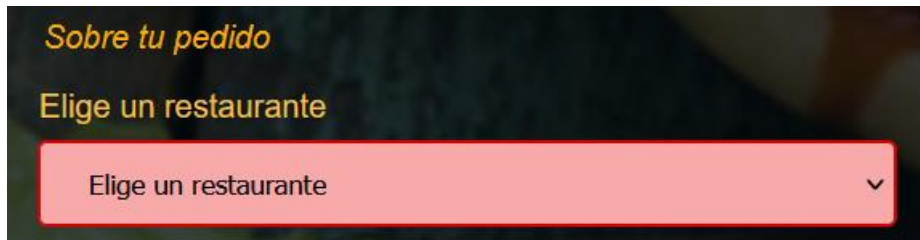
Precio: 10€

A diferencia del RB que recoge los tipos de masa, el radio button de tipo tamaño lleva asociado un evento que hace aparecer un mensaje con el precio. Esto lo explicaremos dentro del [cálculo de precio](#).

Validación Select: Elección del restaurante

La validación de esta sección consiste en verificar que al menos un restaurante ha sido seleccionado. Se desata a raíz de tipos de evento:

- Al enviar el formulario:

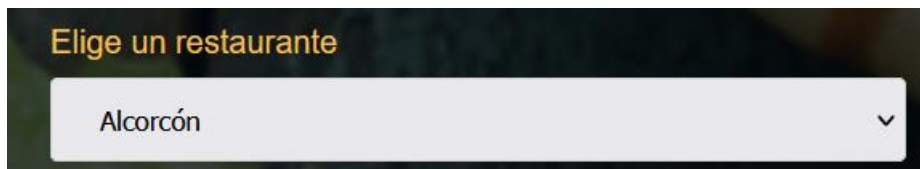


Sobre tu pedido

Elige un restaurante

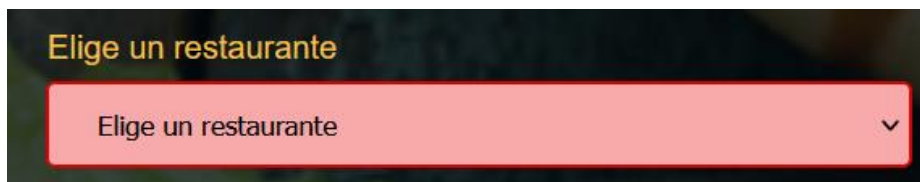
Elige un restaurante ▼

- Al cambiar la selección:



Elige un restaurante

Alcorcón ▼



Elige un restaurante

Elige un restaurante ▼

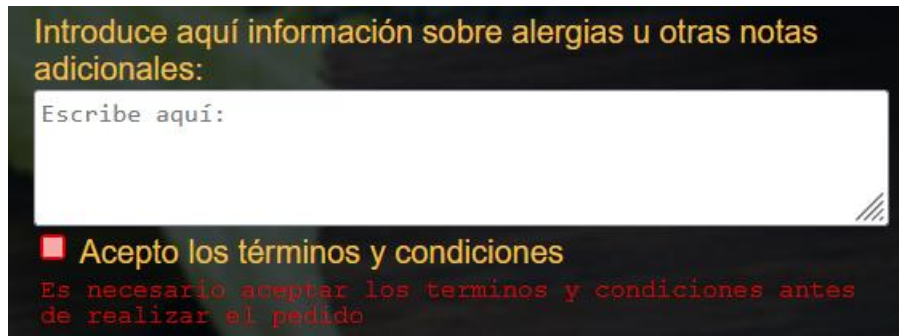
El código en VsCode:

```
function validarRestaurante() {  
  //descartamos la opcion por defecto  
  const opcionesRestaurante = restaurante.querySelectorAll(  
    'option:not([value=""])'  
  );  
  
  for (const rest of opcionesRestaurante) {  
    if (rest.selected) {  
      if (restaurante.classList.contains("invalido"))  
        restaurante.classList.remove("invalido");  
      return true;  
    }  
  }  
  restaurante.classList.add("invalido");  
  return false;  
}
```


Validación Checkbox: Acepto condiciones y términos

Antes de enviar el formulario, el usuario debe aceptar los términos y condiciones. El textarea situado justo encima es opcional y solo lo incluimos como parte de la prueba. Esta validación se desata a raíz de dos eventos:

- Envío del formulario:



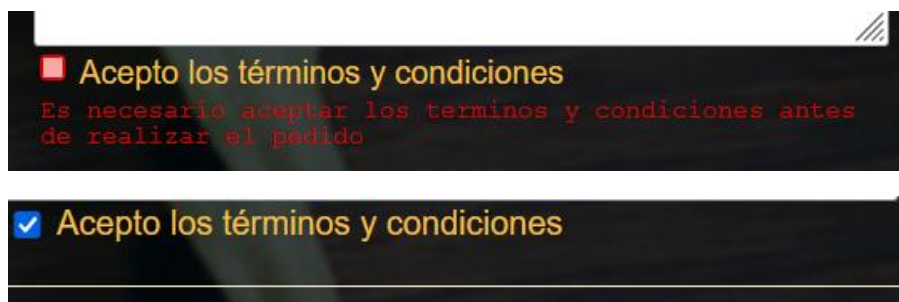
Introduce aquí información sobre alergias u otras notas adicionales:

Escribe aquí:

☐ Acepto los términos y condiciones

Es necesario aceptar los terminos y condiciones antes de realizar el pedido

- Al hacer click en la checkbox:



☒ Acepto los términos y condiciones

Es necesario aceptar los terminos y condiciones antes de realizar el pedido

El código en VSCode

```
function validarTerminos() {  
  const mensajeError = document.querySelector(".terminos__error");  
  
  if (!terminos.checked) {  
    terminos.classList.add("invalido");  
    mensajeError.textContent =  
      "Es necesario aceptar los terminos y condiciones antes de realizar el pedido";  
  } else {  
    if (terminos.classList.contains("invalido"))  
      terminos.classList.remove("invalido");  
    mensajeError.textContent = "";  
  }  
  
  return terminos.checked;  
}
```

Cálculo del precio total

La función `calcularPrecio()` devuelve el precio total de la pizza en base a dos criterios: tamaño (5€ si la pizza es pequeña, 10€ si es mediana y 15€ si es grande) y el número de ingredientes (1€ por ingrediente).

```
function calcularPrecio() {  
  let precio = 0;  
  
  //calculamos el precio del tamaño elegido  
  const tamaño = document.querySelector('input[name="tamaño"]:checked');  
  
  //para lidiar con el precio antes de que el usuario seleccione tamaño  
  const tamañoACobrar = tamaño === null ? "vacío" : tamaño.value;  
  switch (tamañoACobrar) {  
    case "pequeña":  
      precio += 5;  
      break;  
    case "mediana":  
      precio += 10;  
      break;  
    case "familiar":  
      precio += 15;  
      break;  
    default:  
      precio += 0;  
  }  
  
  //calculamos el precio de los ingredientes  
  const ingredientes = document.querySelectorAll(  
    '#opciones-pizza input[type="checkbox"]:checked'  
  );  
  ingredientes.forEach((ing) => (precio += 1));  
  
  //actualizamos el precio mostrado  
  const infoPrecio = document.getElementById("info-precio");  
  infoPrecio.textContent = `Precio: ${precio}\u20AC`;  
  infoPrecio.classList.add("visible");  
  
  return precio;  
}
```

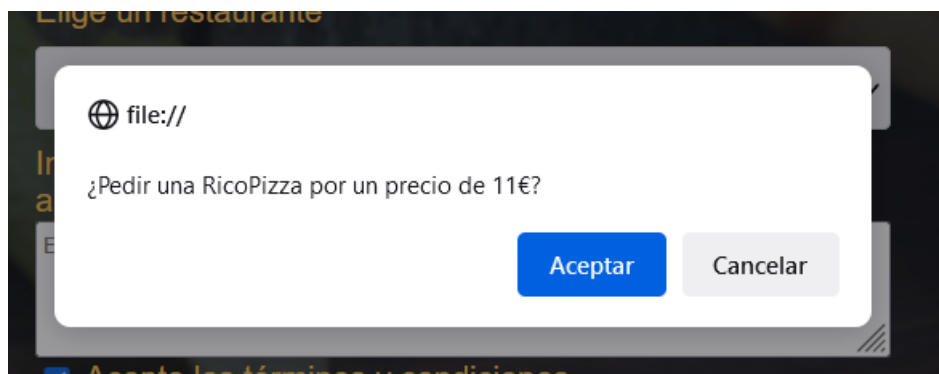
Al enviar el formulario, una vez se han validado los campos, se pide al usuario confirmación, informando del precio (el formulario solo se envía al servidor si el usuario confirma):


```

/*
 *===== Validacion formulario =====
 */
function validarFormulario(event) {
    let valido = true;
    if (!validarNombre()) valido = false;
    if (!validarApellidos()) valido = false;
    if (!validarDireccion()) valido = false;
    if (!validarTlf(event)) valido = false;
    if (!validarMinIngredientes()) valido = false;
    if (!validarEmail()) valido = false;
    if (!validarRestaurante()) valido = false;
    if (!validarMasa()) valido = false;
    if (!validarTamaño()) valido = false;
    if (!validarTerminos()) valido = false;

    if (!valido) {
        alert("Parece que hay errores en el formulario");
        event.preventDefault();
    } else if (
        !confirm(`¿Pedir una RicoPizza por un precio de ${calcularPrecio()}\u20AC?`)
    ) {
        event.preventDefault();
    }
}

```



El precio también se muestra debajo de los RadioButton en el campo opciones-pizza desde el momento en que el usuario altera un campo que afecta al precio, y se actualiza dinámicamente según el usuario interactúa con dichos campos, ya que están asociados a un event listener con esa función:

```
// validacion inmediata del minimo de ingredientes
// y actualizacion del precio
const ingredientesChkboxes = document.querySelectorAll(
  '#opciones-pizza input[type="checkbox"]'
);

ingredientesChkboxes.forEach((checkbox) => {
  checkbox.addEventListener("change", validarMinIngredientes);
  checkbox.onchange = calcularPrecio;
});
```

```
//validacion inmediata de los radio button TAMANIO
//y actualizacion del precio

const tamañoRadioButton = document.getElementsByName("tamaño");
for (var i = 0; i < tamañoRadioButton.length; i++) {
  tamañoRadioButton[i].addEventListener("click", validarTamaño);
  tamañoRadioButton[i].onchange = calcularPrecio;
}
```

Aquí la funcionalidad de la caja de precio, solo aparece cuando se selecciona alguno de los checkbox o RadioButton de tamaño:

¿Cómo quieres tu RicoPizza?

Elige los ingredientes - 1€/ud

- ☐ Mezcla de quesos
- ☐ Champiñones
- ☐ Bacon
- ☐ Tomates cherry
- ☐ Pollo

Elige el tipo de masa

- ☐ Fina
- ☒ Normal
- ☐ Extra
- ☐ Extra con bordes rellenos de queso

Elige el tamaño

- ☐ Pequeña - 5€
- ☐ Mediana - 10€
- ☐ Grande - 15€



¿Cómo quieres tu RicoPizza?

Elige los ingredientes -
1€/ud

- ☒ Mezcla de quesos
- ☒ Champiñones
- ☐ Bacon
- ☒ Tomates cherry
- ☐ Pollo



Elige el tipo de masa

- ☐ Fina
- ☒ Normal
- ☐ Extra
- ☐ Extra con bordes
rellenos de queso

Elige el tamaño

- ☐ Pequeña - 5€
- ☒ Mediana - 10€
- ☐ Grande - 15€

Precio: 13€