AE-1. Carrito de la compra

Grupo 12
23/10/2022

Indice

Integrantes

Adalberto Martinez
Patricia Calzado
Victor Manuel Monzon

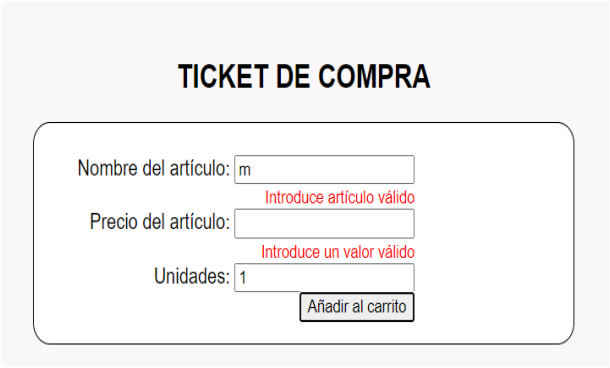
INTRODUCCIÓN

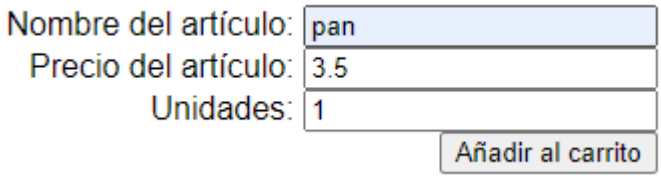
Para la realización de la práctica ha sido necesario la creación de un documento html para crear la estructura de los formularios enlazados con el documento js junto con el css el cual albergará los estilos necesarios de la página. Finalmente, se elaboró el programa en JavaScript para desarrollar el correcto funcionamiento de almacenamiento de artículos y la extracción del ticket de compra.

FUNCIONAMIENTO

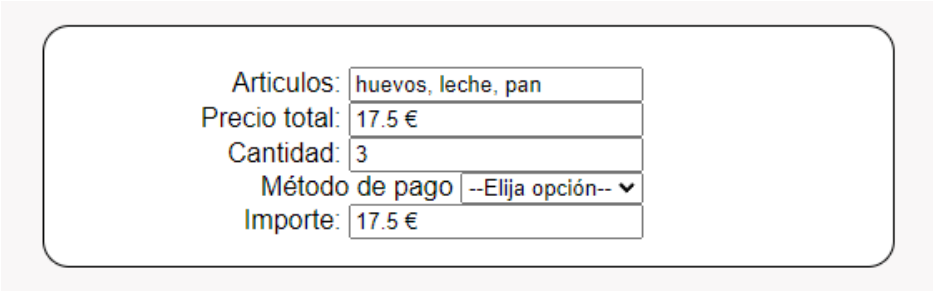
- Hacer que el usuario rellene las 3 primeras cajas y que haga click a un botón el cual acumular los artículos al carrito y nos devolverá un precio total.
- Cada vez que se pulsa el botón “añadir al carrito” se resetean las 3 primeras cajas para poder introducir de nuevo articulo, precio y cantidad.
- Si en alguna caja de articulo y precio no se introduce los datos nos indicará el error: el nombre del artículo no puede encontrase vacío y el precio del artículo debe ser un número real positivo.

```
var validPrice = /^\d+([.]\d*)?$/;
```

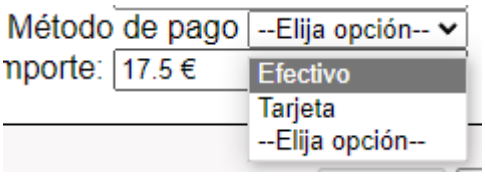




- El carrito almacenará los artículos y hará la suma total del precio:

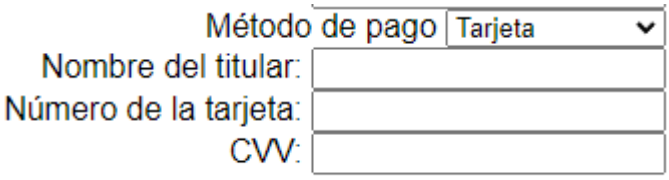


- Seguidamente se procederá a la selección del método de pago: efectivo y tarjeta.

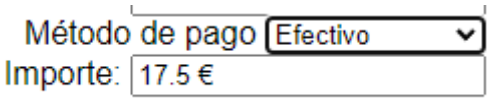


- Si el pago es mediante tarjeta aparecerán 3 nuevas cajas para añadir los datos de la tarjeta bancaria: nombre del titular, número de la tarjeta, cvv. Los cuales deben ser correctos para no mostrar un error: número de 16 dígitos y CVV de tres dígitos:

```
var validCard= /^\d{16}$/;
var validCVV= /^\d{3}$/;
```

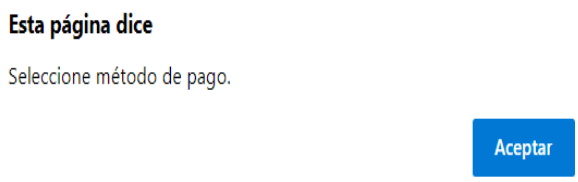


- Si seleccionamos efectivo aparecerá una nueva caja de texto con el importe final.



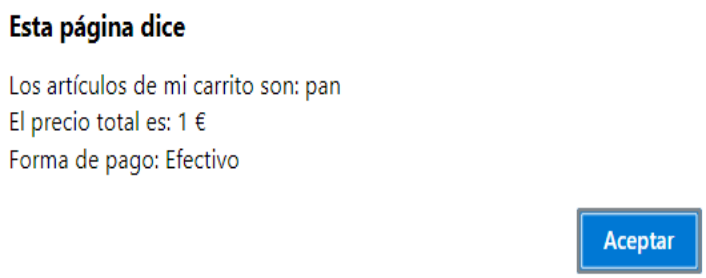
- Para finalizar la compra habrá que seleccionar el checkbox para aceptar las condiciones y finalmente se habilitará el botón de Imprimir:

- Si no introducimos método de pago nos saltara otra venta modal con el texto “introduce de método de pago”.

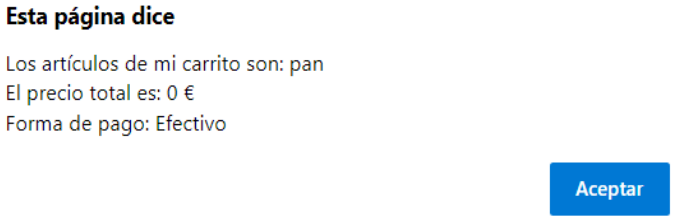


- El botón restablecer nos resetea todas las cajas para poder volver a introducir nuevos productos.

- Finalmente, si todo es correcto, se procederá a mostrar el ticket en un nueva ventana con el monto total y el método de pago elegido:



- Hemos contemplado la posibilidad de tener ofertas o regalos y que se lleven productos gratis bastará que en la caja de precio se ponga como valor “0”.



CÓDIGO

Para inicializar todos los procesos correctamente durante la carga debemos indicarlo definiendo los métodos durante el evento “load”: así podremos usar sin problemas el programa ya que las variables y los eventos se cargan con anterioridad:

```
//Inicializamos las variables y los eventos en carga
window.addEventListener("load", () => {
    inicializarVariables();
    inicializarEventos();
});
```

Variables:

```
//Variables
//variables que almacena la entrada de datos del usuario
var artName, artPrice, artCount;
//variables que recopilan y almacen los datos introducidos
var artText;
var artArray;
//Botones y otros métodos de selección
var submitButton, printButton, resetButton;
var cardBlock, moneyBlock;
var paySelect;
var checkBox;
//Errores
var error1, error2, error3, error4, error5;
//Validaciones
var validPrice = /^\d+([.]\d*)?$/;
var validCard= /^\d{16}$/;
var validCVV= /^\d{3}$/;
```

Inicialización de variables:

```
function inicializarVariables(){
    error1 = document.getElementById("error1");
    error2 = document.getElementById("error2");
    error3 = document.getElementById("error3");
    error4 = document.getElementById("error4");
    error5 = document.getElementById("error5");
    submitButton = document.getElementById("submit");
    artText = document.getElementById("lart");
    price = document.getElementsByName("price");
    artTotal = document.getElementById("lcount");
    artArray = [];
    moneyBlock = document.getElementById("money");
    cardBlock = document.getElementById("card");
    paySelect = document.getElementById("spay");
    resetButton = document.getElementById("reset");
    checkBox = document.getElementById("cbox1");
    printButton = document.getElementById("print");
    printButton.disabled = true;
}
```

Inicializamos todas las variables necesarias , principalmente para los elementos de la página gracias

a .getElementById y .getElementsByName cuando se debe mostrar el mismo valor en varias celdas del formulario (p.ej. Precio). Tambien otros como el array o la disponibilidad del botón “Imprimir”.

Métodos y eventos principales:

Tras rellenar el formulario con el artículo, el botón “Añadir” incluirá dicho producto en el listado. Esta función ligada al evento controlará los errores y llamará al método en caso de que esté todo correcto:

```
//Botón submit guarda los valores y los imprime por pantalla
submitButton.addEventListener("click", () =>{
    artName = document.getElementById("fname").value;
    artPrice = parseFloat(document.getElementById("fprice").value);
    artCount = parseInt(document.getElementById("fcount").value);
    if (artName==undefined || artName==""){
        error1.style.display="inline";
    } else if (!document.getElementById("fprice").value.match(validPrice)) {
        error2.style.display="inline";
    } else {
        guardarValores();
        borrarErrores();
    }
```

Para ello se ha declarado un array de objetos Artículo:

```
//Declaración de clase Artículo
class Article {

    constructor (name, price, count){
        this.name = name;
        this.price = price;
        this.count = count;
    }

}
```

y con el método guardarValores() irá alamacenando todos los artículos del carrito:

```
//Guarda valores en array y muestra valores en el segundo formulario
function guardarValores(){
    var art = new Article(artName, artPrice, artCount);
    artArray.push(art);
    var total = arrayMostrar(artArray);
    document.getElementById("form1").reset();
    artText.value=total.name;
    for (var i=0; i<price.length; i++){

        price[i].value=total.price + " €";
    }
    artTotal.value=total.count;
}
/*Concatena los datos del array y hace el sumatorio del precio y las unidades (Parámetro de entrada: Array y devuelve un
diccionario con tres claves)*/
function arrayMostrar(array){
    var string1 = {name:"", price:0, count:0};
    for (var i = 0; i<array.length; i++){
        if (i==0){
            string1.name=array[i].name + string1.name;
        }else{
            string1.name = array[i].name.concat(", ", string1.name);
        }
    string1.price = array[i].price * array[i].count + string1.price;
    string1.count = array[i].count + string1.count;
    }
    return string1;
}
```

Además, para mostrar los objetos del carrito se desarrolla la función arraymostrar() para pasar a String todos los valores de la clase mediante un bucle FOR y el método .concat(). En ella se almacenan los tres valores en un diccionario para más facilidad de acceso ya que tendríamos los tres Strings necesarios en una sola salida.

Para elegir la visibilidad de las opciones del método de pago, se eligió controlarlo con un SWITCH y variar la visibilidad de los bloques ( ) mediante su ID gracias a las modificaciones en el CSS:

```
//Selección de métodos de pago y muestra diferentes opciones
paySelect.addEventListener('change', () => {
    switch (paySelect.value){
        case "tarjeta":
        cardBlock.style.display = "block";
        moneyBlock.style.display ="none";
        break;
        case "efectivo":
        cardBlock.style.display ="none";
        moneyBlock.style.display = "block";
        break;
        default:
        cardBlock.style.display ="none";
        moneyBlock.style.display="none";
        break;
    }
});
```

Para borrar errores se utilizó getElementsByTagName (ya que todos los errores tienen la etiqueta <small> y así con un simple bucle FOR podríamos desactivar su visibilidad que fue añadida en la hoja de estilos CSS:

```
//Borra errores, quita la visibilidad de los errores
function borrarErrores(){
    var errors = document.getElementsByTagName("small");
    for (var i=0; i<errors.length; i++){
        errors[i].style.display="none";
    }
}
```