Nuestros clientes están muy felices de poder calcular el precio final de sus productos después del descuento, pero la tienda tiene una nueva solicitud para nosotros: **implementar descuentos con cupones**.

Déjame tu solución en los comentarios y luego, sí, evalúa mis 3 propuestas.

**Solución #1: arrays y switch**

Analicemos el problema y nuestra propuesta de solución:

* Crearemos un array con cada uno de nuestros cupones.
* Cambiaremos el input de descuento en HTML por un input de cupones.
* Usaremos un condicional switch para aplicar cierto porcentaje de descuento en nuestros productos dependiendo del cupón que se haya elegido al presionar el botón del formulario.

Vamos paso a paso:

* **Crear un array de cupones**

**const** coupons = [

"JuanDC\_es\_Batman",

"pero\_no\_le\_digas\_a\_nadie",

"es\_un\_secreto",

];

* **Cambiar el input de descuento por uno de cupones**

Cambiemos los IDs en HTML:

<label for="InputCoupon">Escribe el cupón de descuento para tu producto:</label>

<input id="InputCoupon" type="text" />

Y actualicemos nuestras referencias en el código JavaScript:

**const** inputCoupon = document.getElementById("InputCoupon");

**const** couponValue = inputCoupon.value;

* **Crear una variable descuento y asignarle un valor distinto con un switch**

**function** **onClickButtonPriceDiscount**() {

**const** inputPrice = document.getElementById("InputPrice");

**const** priceValue = inputPrice.value;

**const** inputCoupon = document.getElementById("InputCoupon");

**const** couponValue = inputCoupon.value;

**let** descuento;

**switch**(couponValue) {

**case** coupons[0]: // "JuanDC\_es\_Batman"

descuento = 15;

**break**;

**case** coupons[1]: // "pero\_no\_le\_digas\_a\_nadie"

descuento = 30;

**break**;

**case** coupons[2]: // "es\_un\_secreto"

descuento = 25;

**break**;

}

**const** precioConDescuento = calcularPrecioConDescuento(priceValue, descuento);

**const** resultP = document.getElementById("ResultP");

resultP.innerText = "El precio con descuento son: $" + precioConDescuento;

}

**Solución #2: legibilidad, error first y muerte al switch**

**Legibilidad** es la facilidad que ofrece nuestro código para entenderlo sin tropiezos cuando lo leemos. Mejorar la legibilidad es mucho más importante que reducir la cantidad de líneas de código.

Además, **error-first** es un patrón de programación donde atrapamos los errores antes de escribir el resto de la lógica de nuestros programas. Aplicar error-first nos puede ayudar a mejorar la legibilidad de nuestro código (es completamente válido tener otra opinión, no te preocupes, cuéntame en los comentarios).

* **Error first con array.includes**

La función array.includes() nos permite validar si un elemento ha sido almacenado en el array. Vamos a usar esta función para ejecutar un error si el cupón de los usuarios es inválido (no fue guardado en el array coupons):

**if** (!coupons.includes(couponValue)) {

alert("El cupón " + couponValue + "no es válido");

}

Ya que estamos usando un condicional if, podemos reescribir el resto del condicional switch, pero en este caso usaremos condicionales else if:

**if** (!coupons.includes(couponValue)) {

alert("El cupón " + couponValue + "no es válido");

} **else** **if** (couponValue === "JuanDC\_es\_Batman") {

descuento = 15;

} **else** **if** (couponValue === "pero\_no\_le\_digas\_a\_nadie") {

descuento = 30;

} **else** **if** (couponValue === "es\_un\_secreto") {

descuento = 25;

}

**Solución #3: arrays y condicionales mucho más inteligentes**

Ya usamos uno de los muchos métodos de los arrays para aplicar error first. Pero podemos usar algunos más para simplificar aún más nuestro código.

💡 Recuerda que mejorar la legibilidad no (necesariamente) significa reducir líneas de código.

* **Array de cupones 2.0: ahora también con los descuentos**

Por ahora tenemos muy separada la lógica de nuestros cupones. Por un lado tenemos la lista de cupones, pero en otro lugar distinto definimos el descuento de cada cupón.

Vamos a cambiar eso convirtiendo nuestro array de strings en un array de objetos (con el nombre y descuento de cada cupón en el mismo lugar):

**const** coupons = [

{

name: "JuanDC\_es\_Batman",

discount: 15,

},

{

name: "pero\_no\_le\_digas\_a\_nadie",

discount: 30,

},

{

name: "es\_un\_secreto",

discount: 25,

},

];

Ahora la función array.includes dejó de funcionar, pero podemos reemplazarla por la función array.find() para validar si el cupón que escribió el usuario está guardado en la propiedad name de alguno de los objetos en la lista de cupones.

Vamos a crear una variable isUserCouponValueValid con una función que valide si su parámetro coupon es igual a nuestra variable userCouponValue. Y esta variable la vamos a enviar como parámetro a nuestra función array.find().

**const** isCouponValueValid = **function** (coupon) {

**return** coupon.name === couponValue;

};

**const** userCoupon = coupons.find(isCouponValueValid);

**if** (!userCoupon) {

alert("El cupón " + userCouponValue + "no es válido");

}

¡Muy bien!

Pero fíjate que ya tenemos nuestro cupón (con nombre y **descuento**) disponible en la nueva variable userCoupon.

Podemos aprovechar esta variable para eliminar todos nuestros condicionales else if y cambiarlos por un condicional else que nos proporcione el porcentaje de descuento automáticamente, sin importar el cupón (válido) que estén usando nuestros usuarios.

**if** (!userCoupon) {

alert("El cupón " + userCouponValue + "no es válido");

} **else** {

**const** descuento = userCoupon.discount;

**const** precioConDescuento = calcularPrecioConDescuento(priceValue, descuento);

**const** resultP = document.getElementById("ResultP");

resultP.innerText = "El precio con descuento son: $" + precioConDescuento;

}

De esta forma, no solo logramos reducir las líneas de código de nuestro programa, sino también mejorar su legibilidad.

¡Te espero en la siguiente clase para que iniciemos nuestro tercer taller práctico del curso!