clase n°13
JavaScript

<animate a/> programar
revøluciøn* digital_

Funciones

Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones. Un script para una tienda de comercio electrónico por ejemplo, tiene que calcular el precio total de los productos varias veces, para añadir los impuestos y los gastos de envío

Cuando una serie de instrucciones se repiten una y otra vez, se complica demasiado el código fuente de la aplicación, ya que:

- El código de la aplicación es mucho más largo porque muchas instrucciones están repetidas.
- Si se quiere modificar alguna de las instrucciones repetidas, se deben hacer tantas modificaciones como veces se haya escrito esa instrucción, lo que se convierte en un trabajo muy pesado y muy propenso a cometer errores.

Generemos una función para saludar a una persona.

```
function  saludo(persona){
  console.log('Hola ' + persona);
}
saludo('Kiara');
```

```
Hola Kiara
>
```

Reutilizamos el código, y agregamos saludos a dos personas mas

```
saludo('Alfredo');
saludo('Diego');
```

```
Hola Kiara
Hola Alfredo
Hola Diego
>
```

Generemos una función para sumar dos números.

```
function sumar(n1, n2) {
  console.log(n1, n2);
}
sumar(3,2)
```

```
5
```

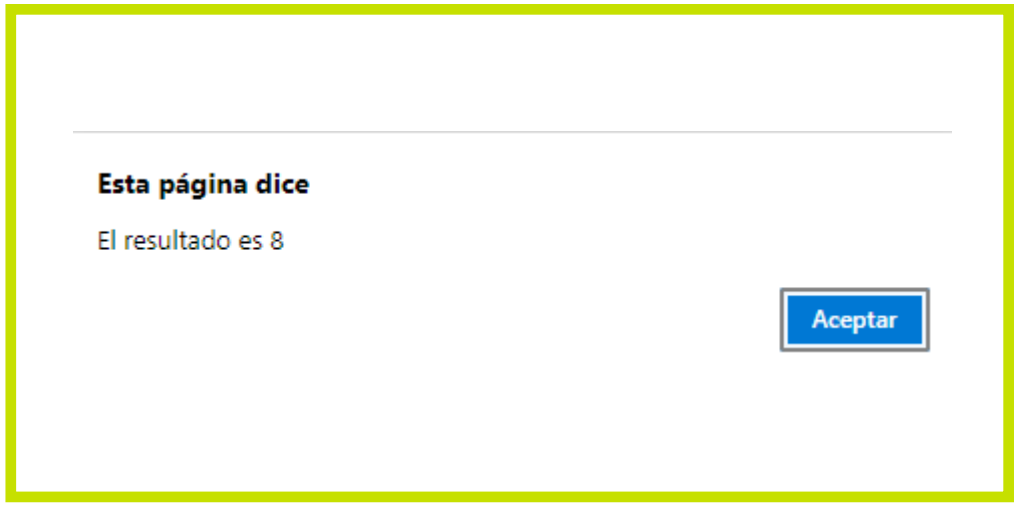
Reutilizamos el código, y sumamos mas numeros

```
sumar(5,5);
sumar(12,2);
```

```
5
10
14
```

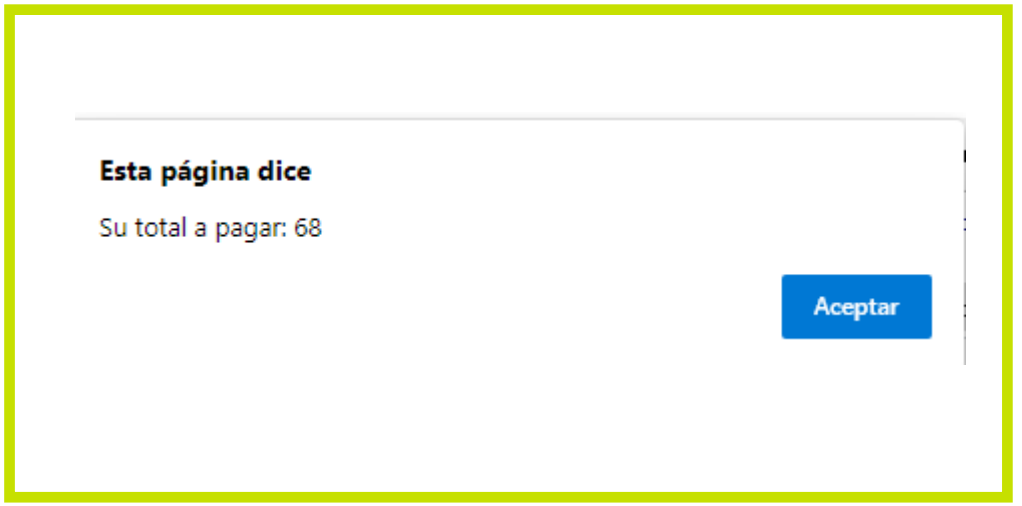
Generemos una función para sumar dos números, pero que nos muestre el resultado por pantalla.

```
var resultado;
var numero1 = 3;
var numero2 = 5;
function suma_y_muestra() {
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
}
suma_y_muestra();
```



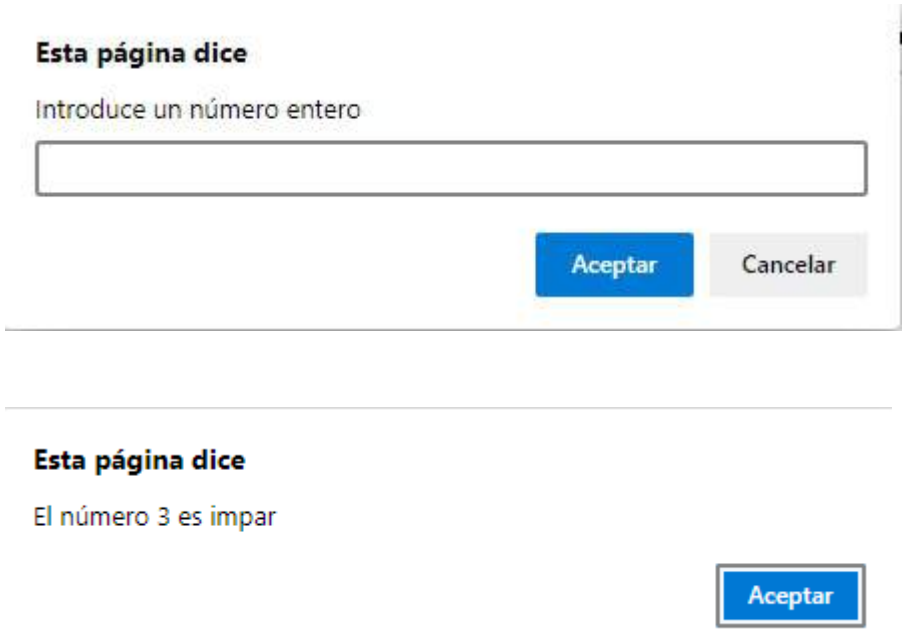

```
// Definición de la función
function calculaPrecioTotal(precio) {
  var impuestos = 1.16;
  var gastosEnvio = 10;
  var precioTotal = ( precio * impuestos ) + gastosEnvio;
  alert("Su total a pagar: " + precioTotal);
}

// Llamada a la función
calculaPrecioTotal(50);
```



Escribir el código de una función a la que se pasa como parámetro un número entero y devuelve como resultado una cadena de texto que indica si el número es par o impar. Mostrar por pantalla el resultado devuelto por la función.

```
<!DOCTYPE html>
<html>
<head>
<title>Ejercicio - Funciones básicas</title>
```




```
<script type="text/javascript">
var numero = prompt("Introduce un número entero");
var resultado = parImpar(numero);
alert("El número "+numero+" es "+resultado);
function parImpar(numero) {
if(numero % 2 == 0) {
return "par";
}
else {
return "impar";
}
}
</script>
</head>
<body>
</body>
</html>
```

ejercicio práctico

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"></head>
<body>

<script>
    let dato1, dato2, num1, num2;

    dato1 = window.prompt("Introduce primer número ?", "0");
    num1 = parseInt(dato1);
    dato2 = window.prompt("Introduce segundo número ?", "0")
    num2 = parseInt(dato2);

    let resultado = num1 + num2;

    document.write(`<br/> <br/> La suma es ${resultado} ` );

</script>


</body>
</html>
```

Agregar las 4 operaciones Básicas

Proyecto nuevo calculo de un promedio

Lista y funciones

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>

  <script type="text/javascript">
      function calcula()
      {
        let numero = document.fdatos.entrada.value;
        let dato = document.getElementById("lista");
        let valor = dato.options[lista.selectedIndex].value;

        let resultado = numero * valor;
        document.getElementById("salida").innerHTML = resultado;
      }

  </script>
</head>
<body>
  <form name ="fdatos">
    Introduce numero: <input type ="text" size ="10" name ="entrada"/>
    <b> X </b>
    <select id = "lista" onchange ="calcula()">
      <option value="1" selected>Uno</option>
      <option value="2" selected>Dos</option>
      <option value="3" selected>tres</option>
      <option value="4" selected>Cuatro</option>
    </select>
  </form>

  <div id ="salida"></div>

</body>
</html>
```

ejemplo de la interacción con el usuario

HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Tu peso en un lugar donde pesas menos</title>
    <link rel="stylesheet" href="peso.css">
  </head>
  <body>
    <h1>Tu peso en otro planeta</h1>
    <p>En la tierra pesas distinto que en Marte o Jupiter</p>
    <ul>
        <li>Ingresa tu peso en el espacio Indicado</li>
        <li>Haz click en el Botón calcular</li>
        <li>Descubre tu Peso en otros Planetas</li>
    </ul>
    <span>Digite tu Peso (KILOGRAMOS)</span>
    <p>
  <a class="botonestilo" href="#" role="button" id="boton">Calcule Aquí</a>
  </p>
  <script>
// seleccionamos el enlace
var boton = document.getElementById("boton");
// cuando se pulsa en el enlace
boton.onclick = function(e) {
```

JAVASCRIPT

```javascript
var usuario = prompt("Cual es tu peso?");
var peso = parseInt(usuario);
var g_tierra = 9.8;
var g_marte = 3.7;
var g_jupiter = 24.8;
var peso_final;
peso_final = peso * g_marte / g_tierra;
peso_final = parseInt(peso_final);
window.alert("Tu peso en marte es: " + peso_final + " kilos");
```

CSS

```css
.botonestilo {
text-decoration:none;
font-weight:600;
font-size:20px;
color:#ffffff;
padding-top:15px;
padding-bottom:15px;
padding-left:150px;
padding-right:150px;
background-color:#FFD27D;
}
```

Funciones Callback

Una función **callback** es aquella que es pasada como argumento a otra función para que sea "llamada de nuevo" (call back) en un momento posterior.

Una función que acepta otras funciones como argumentos es llamada función de orden-superior (High-Order), y contiene la lógica para determinar cuándo se ejecuta la función callback. Es la combinación de estas dos la que nos permite ampliar nuestra funcionalidad.

Para ilustrar las funciones callback, iniciemos con un ejemplo simple:

```
function crearCita(cita, callback){
  var miCita = "Como yo siempre digo, " + cita;
  callback(miCita); // 2
}

function logCita(cita){
  console.log(cita);
}

crearCita("come tus vegetales!", logCita); // 1

// Resultado en la consola:
// Como yo siempre digo, come tus vegetales!
```

En el ejemplo anterior, crearCita es la función de orden-superior, la cual acepta dos argumentos, el segundo es el callback. La función logCita se está pasando como nuestra función callback. Cuando ejecutamos la función crearCita (1), observa que no estamos agregando paréntesis a logCita al pasarla como argumento. Esto se debe a que no queremos ejecutar nuestra función callback de inmediato, simplemente queremos pasar la definición de la función a la función de orden-superior para que pueda ejecutarse más tarde

Además, podemos pasar funciones anónimas como callbacks. La siguiente llamada a crearCita tendrá el mismo resultado que el ejemplo anterior:

```
crearCita("come tus vegetales!", function(cita){
  console.log(cita);
});
```

Por cierto, tu no estás obligado a usar la palabra "callback" como el nombre de tu argumento, Javascript solo necesita saber que es el nombre correcto del argumento. Según el ejemplo anterior, la siguiente función se comportará exactamente de la misma manera

```
function crearCita(cita, funcionParaLlamar) {
  var miCita = "como siempre digo, " + cita;
  funcionParaLlamar(miCita);
}
```

¿Por qué usar funciones Callback?

La mayoría del tiempo estamos creando programas y aplicaciones que operan en una forma sincrónica. En otras palabras, algunos de nuestra operaciones comienzan solo después de que se hayan completado las anteriores. Usualmente, cuando solicitamos datos desde otras fuentes como una API externa, no siempre sabemos cuando nuestros datos serán devueltos. En estos casos queremos esperar la respuesta, pero no queremos que toda nuestra aplicación se detenga mientras se recuperan los datos. Estas son situaciones dónde las funciones callback resultan útiles.

- Veamos un ejemplo que simula una solicitud a un servidor:

```
function solicitudServidor(consulta, callback){
  setTimeout(function(){
    var respuesta = consulta + "lleno!";
    callback(respuesta);
  },5000);
}

function obtenerResultados(resultados){
  console.log("Respuesta del servidor: " + resultados);
}

solicitudServidor("El vaso está medio  ", obtenerResultados);

//Resultado en la consola luego de 5 segundos:
// El vaso está medio lleno!
```

- En el ejemplo anterior hacemos una solicitud simulada al servidor. Después de que pasen cinco segundos, la respuesta es modificada y luego nuestra función callback obtenerResultados se ejecuta. Para ver esto en acción, puedes copiar / pegar el código anterior en las herramientas de desarrollador de tu navegador y ejecutarlo.

revøluciøn*
digital_