Funciones

Introducción a las funciones:

Cuando se desarrolla una aplicación compleja, es muy habitual utilizar una y otra vez las mismas instrucciones.

Por ejemplo, un script para una tienda de comercio electrónico ha de 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 mucho el código fuente de la aplicación ya que:

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

Las funciones son la solución a todos estos problemas, tanto en JavaScript, como en la resta de lenguajes de programación.

Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

```
<script type="text/javascript">
        var resultado;
        var numero1 = 3;
        var numero2 = 5;
        resultado = numero1 + numero2;
        alert("El resultado és " + resultado);

        numero1 = 10;
        numero2 = 7;
        resultado = numero1 + numero2;
        alert("El resultado és " + resultado);

        numero1 = 5;
        numero2 = 8;
        resultado = numero1 + numero2;
        alert("El resultado es " + resultado);
</script>
```

En el ejemplo anterior, a pesar de lo absurdo que parece, resulta más absurdo aún repetir constantemente el código teniendo en cuenta que ejecutan lo mismo.

La solución que proponen las funciones consiste en extraer las instrucciones que se repiten y sustituirlas por una instrucción del tipo “en este punto se ejecutan las instrucciones que se han extraido”:

```
<script type="text/javascript">

        var resultat;
        var numero1 = 3;
        var numero2 = 5;

        / * En este punto se le dice a la funcion que sume 2 numeros y muestre el resultado *
        /

        numero1 = 10;
        numero2 = 7;

        / * En este punto se le dice a la funcion que sume 2 numeros y muestre el resultado *
        /

        numero1 = 5;
        numero2 = 8;

        / * En este punto se le dice a la funcion que sume 2 numeros y muestre el resultado *
        /

</script>
```

Entonces, para que la solución anterior sea válida, las instrucciones comunes se deberían agrupar en una función a la cual se le pueda indicar los número que se han de sumar antes de mostrar el mensaje.

Por tanto, en primer lugar se ha de crear la función básica con las instrucciones comunes.

Las funciones en JavaScript se definen mediante la palabra reservada **function**, seguida del nombre de la función. Su definición formal seria la siguiente:

```
Function nombre_funcion(parámetros si los tiene)
{
        Instrucciones…
}
```

El nombre de la función se utiliza para llamar a esa función cuando sea necesario. El concepto es el mismo que con las variables, a las cuales se les asigna un nombre único para poder utilizarlas dentro del código.

Después del nombre de la función, se incluirían dos paréntesis entre los cuales vendrían los argumentos que recibiría esa función. Más adelante se explicarán.

Finalmente, los símbolos **{** i **}** se utilizan para cerrar todas las instrucciones que pertenecen a la función (de forma similar a como se cierran las instrucciones en las estructuras IF y FOR).

Volviendo al ejemplo anterior, se crea una función llamada **suma_y_muestra** de la manera siguiente:

```
<script type="text/javascript">

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

</script>
```

El código final quedaria así:

```
<script type="text/javascript">

        function suma_y_muestra()
        {
                resultat = numero1 + numero2;
                alert("El resultat és " + resultat);
        }

        var resultado;
        var numero1 = 3;
        var numero2 = 5;
        suma_y_muestra();

        numero1 = 10;
        numero2 = 7;
        suma_y_muestra();

        numero1 = 5;
        numero2 = 8;
        suma_y_muestra();
</script>
```

El código del ejemplo anterior es mucho más eficiente que el primer código mostrado, ya que no existen instrucciones repetidas.

Las instrucciones que suman y muestran mensajes se han agrupado bajo una función, lo que permite ejecutarla en cualquier punto del programa simplemente indicando el nombre de la función.

No obstante, aún faltaría un pequeño pero importante dato para que qué este código quedase totalmente optimizado; los argumentos.

Argumentos.

Las funciones más sencillas no necesitan ninguna información para producir sus resultados.

No obstante, la mayoría de funciones de las aplicaciones reales han de acceder al valor de algunas variables para producir sus resultados.

Las variables que necesitan las funciones se llaman argumentos. Antes de poder utilizarlos, la función ha de indicar cuantos argumentos necesita y cuales son los nombres de cada argumento.

Además, al invocar la función, se han de incluir los valores que se le pasaran a la función.

Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan por comas.

Siguiendo el ejemplo de la suma de números, la función ha de indicar que necesita dos argumentos, correspondientes a los dos números que ha de sumar:

```
<script type="text/javascript">

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

Dentro de la función, el valor de la variable numero1 será igual al primer valor que se pase a función y el valor de la variable numero2 es igual al segundo valor que se le pasa.

Para pasar valores a la función, se incluyen dentro de los paréntesis utilizados al llamar a la función:

```
<script type="text/javascript">

        function suma_y_muestra(num1,num2)
        {
                var resultado = numero1 + numero2;
                alert("El resultado és " + resultado);
        }
        var numero1 = 3;
        var numero2 = 5;
        suma_y_muestra(numero1,numero2);

</script>
```

En el código anterior se ha de tener en cuenta que:

- Aunque casi siempre se utilizan variables para pasar los datos a la función, se podría haber utilizado directamente el valor de estas variables: **suma_y_muestra(3,5);**
- El número de argumentos que se pasan a una función deberían ser los mismo que el número de argumentos que se han indicado en la función. No obstante, JavaScript no muestra ningún error si se le pasan más o menos argumentos.
- El orden de los argumentos es fundamental, ya que el primer dato que se indica en la llamada será el primer valor que espere la función, el segundo que se indica en la llamada será el segundo que espere la función y así sucesivamente.
- Se puede utilizar un número ilimitado de argumentos, aunque contra más argumentos más se complica la llamada a la función.
- No es obligatorio que coincidan el nombre de los argumentos que utiliza la función y el nombre de los argumentos que se le pasan.

Valores de retorno de una función

Las funciones no solo pueden tener parámetros de entrada sino que también pueden devolver valores que se pueden utilizar más adelante en el documento.

Pongamos un ejemplo para tenerlo más claro. A continuación se muestra otro ejemplo de una función que calcula el precio total de un producto a partir de su precio básico:

```
<script type="text/javascript">

// Definición de la función
function calculaPrecioTotal(precio)
{
        var impuestos = 1.16;
        var gastosEnvio = 10;
        var precioTotal = ( precio * impuestos ) + gastosEnvio;
}

// Llamada a la funcion
calculaPrecioTotal (23.34);

</script>
```

La función anterior toma como argumento una variable llamada precio y se le suma los impuestos y los gastos de envío para obtener el precio total. Al llamar a la función se le pasa directamente el valor del precio básico mediante el número 23,34.

No obstante el código, anterior no es muy útil puesto que lo ideal sería que la función pudiese devolver el resultado obtenido para guardarlo en otra variable y poder seguir trabajando con ella:

Afortunadamente, las funciones no solo pueden recibir variables y datos, sino que también pueden devolver los valores que han calculado. Para devolver los valores dentro de la función se utiliza la palabra clave **return**.

Aunque las funciones pueden devolver valores de cualquier tipo, solo pueden devolver uno cada vez que se ejecutan.

```
<script type="text/javascript">

// Definición de la función
function calculaPrecioTotal(precio)
{
        var impuestos = 1.16;
        var gestosEnvio = 10;
        var precioTotal = (precio * impuestos) + gestosEnvio;
        return precioTotal;
}

/ / El valor retornat per la funció, es guarda en una variable
var precioTotal = calculaPrecioTotal (23.34);

/ / Seguir treballant amb la variable "preuTotal"

</script>
```

Para que la función devuelva un valor, solo hace falta escribir la palabra reservada **return** junto al nombre de la variable que se quiere devolver. En el ejemplo anterior, la ejecución de la función llega a la instrucción **return** precioTotal, y en este momento, devuelve el valor que contiene la variable precioTotal.

Como la función devuelve un valor, en el punto en el que se realiza la llamada, hay que indicar el nombre de una variable en la que se guardara el valor retornado.

Si no se indica ninguna variable de recogida, JavaScript no producirá ningún error pero el valor retornado por la función se perderá.

En este caso tampoco es necesario que la variable que devuelve la función tenga el mismo nombre que la variable que la recoge.

Si la función llega a la instrucción del tipo return, se devuelve el valor y finaliza la ejecución de la función, lo que significa que todas las ordenes posteriores serán ignoradas.

Variables predefinidas

JavaScript posee una serie de funciones predefinidas.

Las más importantes son:

- **isNaN()** -> Determina si el valor introducido como argumento es un numero o no, devolviendo true o false.

- **parseInt()** -> Convierte a numérico una cadena.

- **parseFloat()**-> Convierte a decimal una cadena.