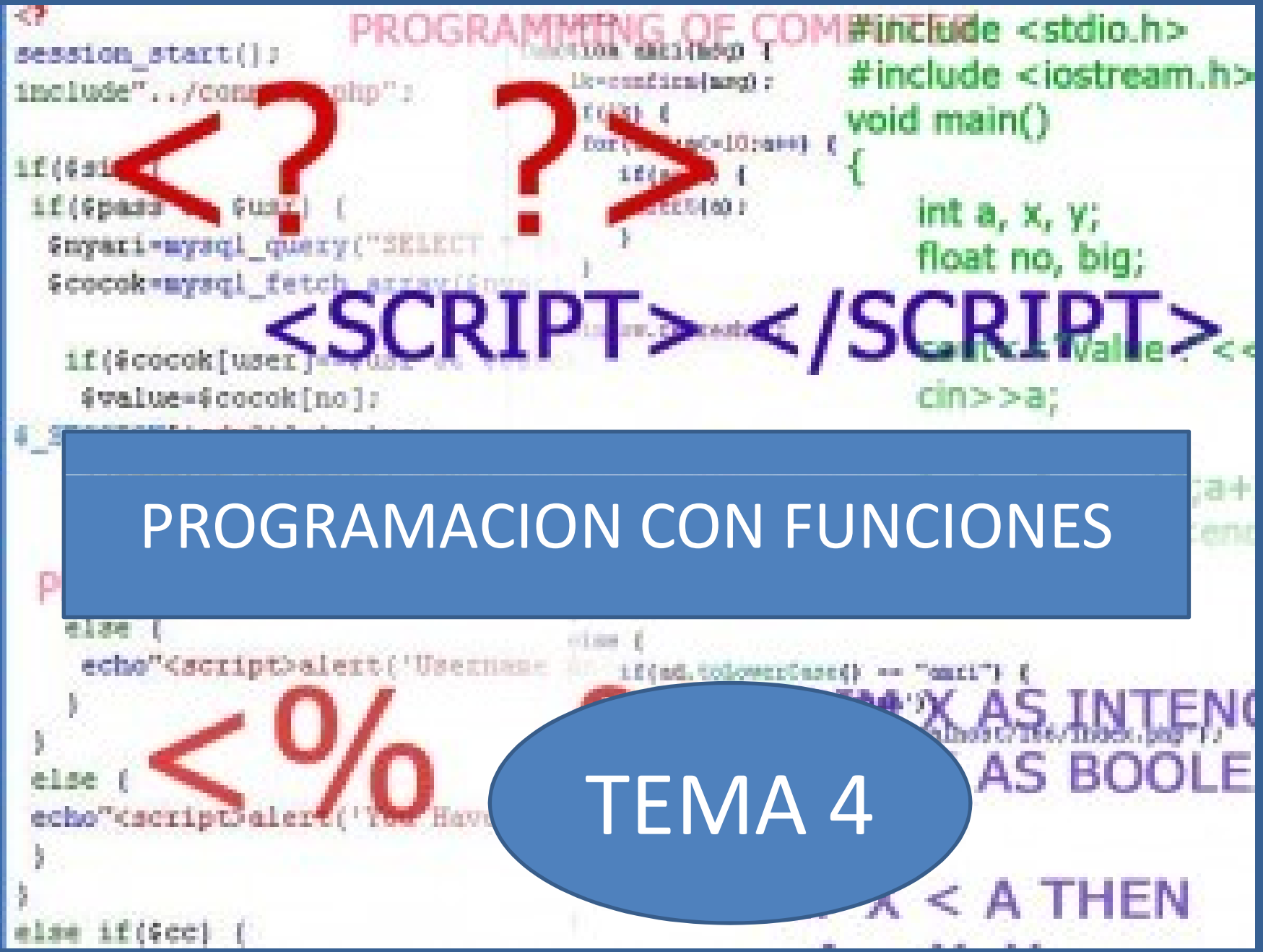
<? ?>
<SCRIPT></SCRIPT>
<%
#include <stdio.h>
#include <iostream.h>
void main()
{
int a, x, y;
float no, big;
cin>>a;
PROGRAMACION CON FUNCIONES
TEMA 4

Funciones del Usuario

- Es posible crear funciones personalizadas diferentes a las funciones predefinidas por el lenguaje.
- Con estas funciones se pueden realizar las tareas que queramos.
- Una tarea se realiza mediante un grupo de instrucciones relacionadas a las cuales debemos dar un nombre.

- En términos generales, una función es un "subprograma" que puede ser *llamado* por código externo (o interno en caso de recursión) a la función.
- Al igual que el programa en sí mismo, una función se compone de una secuencia de declaraciones, que conforman el llamado *cuerpo de la función*.
- Se pueden pasar valores a una función, y la función puede *devolver* un valor.
- En JavaScript, las funciones son objetos de primera clase, es decir, son objetos y se pueden manipular y transmitir al igual que cualquier otro objeto. Concretamente son objetos Function.

Se utilizan de tres formas:

- Funciones Globales
- Métodos de objetos
- Constructores

Funciones del Usuario

Definición de funciones:

- El mejor lugar para definir las funciones es dentro de las etiquetas HTML <head>y </head>.
- El motivo es que el navegador carga siempre todo lo que se encuentra entre estas etiquetas.
- **Lo más correcto sin duda es crear un fichero .js**

Funciones del Usuario

- La definición de una función consta de cinco partes:
  - La palabra clave **function**.
  - El nombre de la función.
  - Los argumentos utilizados.
  - El grupo de instrucciones.
  - La palabra clave **return**

Funciones del Usuario

Definición de funciones –Sintaxis:

```
function nombre_función ([argumentos]) {
  grupo_de_instrucciones;
  [return valor o expresión;]
}
```

Funciones del Usuario

- Definición de funciones –function:

Es la palabra clave que se debe utilizar antes de definir cualquier función

Funciones del Usuario

- Definición de funciones –Nombre:

El nombre de la función se sitúa al inicio de la definición y antes del paréntesis que contiene los posibles argumentos.

  - Deben usarse sólo letras, números o el carácter de subrayado.
  - Debe ser único en el código JavaScript de la página web.
  - No pueden empezar por un número.
  - No puede ser una de las palabras reservadas del lenguaje.

Funciones del Usuario

Definición de funciones –Argumento:

- Los argumentos se definen dentro del paréntesis situado después del nombre de la función.
- Los parámetros en la llamada a una función son los argumentos de la función.
- Los argumentos se pasan a las funciones *por valor*

- No todas las funciones requieren argumentos, con lo cual el paréntesis se deja vacío.

Funciones del Usuario

Definición de funciones –Grupo de instrucciones:

- El grupo de instrucciones es el bloque de código JavaScript que se ejecuta cuando invocamos a la función desde otra parte de la aplicación.
- Las llaves ({}) delimitan el inicio y el fin de las instrucciones.

Funciones del Usuario

Definición de funciones –Return:

- La palabra clave return es opcional en la definición de una función.
- Indica al navegador que devuelva un valor a la sentencia que haya invocado a la función

Funciones del Usuario

Ejemplo –Función que calcula el importe de un producto después de haberle aplicado el IVA:

```
function aplicar_IVA(valorProducto, IVA){
var productoConIVA= valorProducto* IVA;
alert("El precio del producto, aplicando el IVA
del " + IVA + " es: " + productoConIVA);
}
```

Funciones del Usuario

Invocación de funciones:

- Una vez definida la función es necesaria llamarla para que el navegador ejecute el grupo de instrucciones.
- Se invoca usando su nombre seguido del paréntesis.
- Si tiene argumentos, se deben especificar en el mismo orden en el que se han definido en la función.

Funciones del Usuario

- Ejemplo:

```
aplicar_IVA(300, 1.18);
```

Funciones del Usuario

- Realizar el ejemplo de la presentación :
- Introducir los valores con prompt

1. Colocar la función en head y llamarla desde un script.

3. Crear un fichero .js incorporarlo y probar los dos tipos de llamada

Funciones del Usuario

- Devolver un valor desde una función RETURN
- function media(valor1,valor2){
  var resultado
  resultado = (valor1 + valor2) / 2
  return resultado
  }
  Código de la llamada
- var miMedia;
  miMedia = media(12,8);
  document.write (miMedia);

Ámbito -Scope

- **El ámbito global**: yo puedo definir variables fuera de mis funciones que estas variables se encuentran disponibles internamente.

NO RECOMENDABLE

```
const x = 5;
function printX() {
    console.log(x);
}
printX(); // pinta 5
```

Ámbito -Scope

- **El ámbito función**: dentro de una función podemos declarar variables que estas no pueden ser accedidas desde fuera de su ámbito.

```
function foo() {
    const x = 5;
}
console.log(x); // undefined.
```

- **El ámbito de bloque**: podemos declarar variables que solo son disponibles dentro de un if, while, for o switch. Esto podemos conseguirlo gracias a la palabra reservada ‘let’.

Mas sobre function

- Las funciones son definidas como tal, pero también pueden ser empleadas como datos.

```
function saluda(){
        console.log("hola");
}

function ejecuta(func){
        func();
}

ejecuta(saluda)
```

Funciones anónimas

```
var saluda = function(quien){
        console.log("hola " + quien);
}

saluda("mundo");
```

Funciones autoejecutables

```
function(quien){
        console.log("hola " + quien);
}
```

Función anónima sin asignar a variable. NO se puede ejecutar

```
(function() { console.log("hola mundo") }) ()
```

La podemos convertir autoejecutable

```
( function(quien){
    console.log("hola " + quien);
})("mundo");
```

Autoejecutable pasando parámetros

Una función puede devolver una función anónima

```
function saludator(quien){
        return function(){ //acá se crea la funcion anónima a retornar
                alert("hola " + quien);
        }
}

var saluda = saludator("mundo");

saluda();
```

O de esta otra forma
saludator("mundo")()