**PAC1 - JavaScript: Estructures bàsiques.**

**1.- Explica utilitzant un exemple de codi, els distints àmbits de les variables a JavaScript, indicant com i on es declaren. (1 punt)**

Fixem-nos en el següent codi:

*var variable\_1 = «variable 1»;*

*variable\_2 = «variable 2»;*

*function mes\_variables() {*

*var variable\_3 = «variable 3»;*

*variable\_4 = «variable 4»;*

*}*

En aquest codi es donen els 4 casos que es poden donar en referència a l’àmbit de les variables. Les variables *variable\_1* i *variable\_2* es defineixen a fora de cap funció i per tant són variables globals, independentment de que es declarin o no amb la paraula reservada *var*. La variable *variable\_3* es defineix a dintre d’una funció i es fa servir la paraula reservada *var*, per tant, es tracta d’un variable local, és a dir, que es pot fer servir a dintre de la funció però no a fora. En canvi, la variable *variable\_4*, encara que es defineix a dintre d’una funció, com que en la seva definició no es fa servir la paraula clau *var,* es tracta d’una variable global, que es pot fer servir a fora de l’àmbit de la funció. Per tant, resumint, tenim que només són variables locals les que es definèixen a dintre d’una funció (o qualsevol altra estructura que en defineixi l’àmbit) i ho fan utilitzant la paraula reservada *var*; els demés casos es tracta de variables globals.

**2.- Descriu el principal ús que es donen als operadors de comparació. Explica la utilitat dels operadors estrictes. (1,5 punt)**

El seu principal ús és en les estructures iteratives i de decisió. Com a estructures de control iteratives tenim el *for*, el *while* i el *do ... while*. En la sintaxi dels bucles *for* hi intervenen tres expressions, de les quals la segona és una condició que es sol definir amb un operador de comparació. En els bucles *while* i *do ... whil*e hi intervé una expressió, que es tracta també d’una condició que també es sol definir amb un operador de comparació.

Pel cas de les estructures de decisió o altrament dites de control condicional, tenim l’*if* i el *switch*. En el cas de l’*if*, es porten a terme una o vàries accions si es compleix una condició, que sol ser una expressió amb algun operador de comparació. En el cas del *switch* no es verifica si es compleix una condició, i per tant no es fan servir operadors de comparació.

En referència als operadors de comparació estrictes, en tenim dos: la igualtat estricta (===) i la desigualtat estricta (!==). La diferència respecte als operadors d’igualtat i desigualtat no estrictes (== i !=) és que els comparadors estrictes fan una comparació estricta de tipus, és a dir, que a més de comprovar la igualtat o desigualtat dels valors també es comprova la igualtat o desigualtat en els tipus. Per tant, per exemple, tindriem que *2018 == ‘’2018’’* serà cert, ja que els valors són iguals, malgrat els tipus no ho són (2018 és un nombre i ‘’2018’’ és una cadena), però *2018 === ‘’2018’*’ serà fals, ja que per un costat tenim un nombre i per l’atre una cadena, que són dos tipus diferents.

**3.- La sentència with pot ajudar a simplificar el codi JavaScript en certes estructures. Identifica amb dos exemples diferents aquesta possibilitat. (1,5 punt)**

Vegem el primer exemple:

*edats = new Array(34, 76, 5, 45, 23, 55, 46);*

*edats.push(44);*

*edats.push(57);*

*edats.sort();*

*edats.reverse();*

*edats.pop();*

*edats.join();*

Emprant la sentència *with* això es podria simplificar al següent codi:

*edats = new Array(34, 76, 5, 45, 23, 55, 46);*

*with(edats) {*

*push(44);*

*push(57);*

*sort();*

*reverse();*

*pop();*

*join();*

*}*

Anem a veure un altre exemple utilitzant un altre tipus d’objecte:

*var area, x, y;*

*var radi = 35;*

*with(Math) {*

*x = radi \* cos(PI);*

*y = radi \* sin(PI);*

*area = PI \* r \* r;*

*}*

En aquest cas el que tenim és que amb la sentència *with* especifiquem l’objecte Math. Sense utilitzar *with* aquest codi seria així:

*var area, x, y;*

*var radi = 35;*

*x = radi \* Math.cos(Math.PI);*

*y = radi \* Math.sin(Math.PI);*

*area = Math.PI \* r \* r;*

Com podem veure, amb el *with* el codi sembla més ordenat i fàcil d’entendre.

**4.- Explica quines avantatges té l’ús de funcions dins el nostre codi i les principals característiques dels seus paràmetres. (1,5 punt)**

Una funció és un bloc de codi dissenyat per a realitzar una tasca particular. Una funció s’executa quan se la invoca. El principal avantatge que té utilitzar funcions és la reutilització del codi. Amb les funcions, el codi es defineix una vegada però s’utilitza moltes vegades. Amés, es pot utilitzar el mateix codi moltes vegades amb diferents arguments, per tal de produir resultats diferents.

Els paràmetres d’una funció són unes ‘’variables’’ que es defineixen per a poder passar dades d’entrada que la funció farà servir per a realitzar la tasca que implementa la pròpia funció. Una funció pot tenir zero o qualsevol nombre de paràmetres. Els paràmetres s’utilitzen a les funcions com a variables locals, és a dir, fora de la funció no tenen sentit. Els paràmetres es passen per valor, és a dir, que la variable a l’exterior de la funció que es passa com a paràmetre no canviarà el seu valor degut a les manipulacions que pateixi dintre de la funció; és com si el que es passa a la funció sigui una còpia de la variable externa. Ara bé, aquesta norma no es compleix per a tots els tipus de paràmetres, ja que els arrays i els objectes són passats per referència i d’aquesta manera les manipulacions que pateixen a dintre de la funció sí que afecten als propis arrays o objectes. Una altra característica dels paràmetres és que si en una crida a una funció no es passen certs paràmetres, aquests passaran a tenir el valor *null* a la funció.

**5.- Explica la funcionalitat de les sentències break i continue, el seu objectiu i les recomanacions d’us. (1,5 punt)**

Són dues sentències que es poden fer servir a dintre del bucles, per modificar el comportament o fluxe normal del bucle.

En el cas de la sentència *break*, es fa servir quan es vol que l’execució del bucle s’interrompi. La sentència *continue*, en canvi, no es tan dràstica, ja que el que fa és interrompre només l’execució de l’actual iteració, transferint el control a la següent iteració però sense sortir del bucle.

La utilització d’ambdues sentències està desaconsellada en el que es ve a conèixer com a ‘’programació elegant’’, ja que la seva utilització provoca que el manteniment dels programes sigui molt dificultosa. Sempre hi ha alguna manera d’aconseguir els mateixos resultats sense haver d’utilitzarles, i aquestes opcions normalment són més desitjables.

**6.- Escriu un programa en JavaScript que et demane un nombre inicial. A partir d’aquest nombre es crearà un bucle que anirà demanat nombres sencers i es emmagatzemaran en una estructura del tipus array.**

**Quan s’introdueixca l’últim nombre, es crearà el codi JavaScript amb estructures adients per ordenar els nombres de l’array i es mostrarà el resultat en una finestra modal. (3 punts)**

A continuació es mostren els codis en HTML i JavaScript corresponents a aquest exercici:

*<html>*

*<head>*

*<title>PAC1 - Exercici 6</title>*

*<script type="text/javascript" src="pac1\_ex6.js"></script>*

*</head>*

*<body>*

*<p>Pitja el botó per començar...</p>*

*<form name="form\_boto">*

*<p>*

*<input type="button" name="boto" value="Comença" onclick="javascript:mostra\_vector\_ordenat()">*

*</p>*

*</form>*

*</body>*

*</html>*

*// Crea una funció que demana el nombre de vegades que es demanarà un nombre*

*// sencer, va guardant aquests nombre en un vector, i després de demanar el darrer*

*// nombre ordena el vector i el mostra.*

*function mostra\_vector\_ordenat() {*

*// Mostrem un prompt que demana la quantitat de nombre sencers que tindrà*

*// el vector, convertim el valor introduit a un nombre sencer i el guardem en una*

*// variable.*

*var iteracions = parseInt(prompt("Introdueix la quantitat de nombres sencers que tindrà el vector:",""));*

*// Definim un vector i el guardem en una variable.*

*var vector = new Array(iteracions);*

*// Bucle que demana els nombre.*

*for (var i = 0; i < iteracions; i++) {*

*// Mostrem un prompt que demana un número sencer que tindrà el vector,*

*// convertim el valor introduit a un nombre sencer i el guardem en una variable.*

*var num = parseInt(prompt("Introdueix el nombre sencer " + (i + 1) + " d'un total de " + iteracions + " nombres:"));*

*// Afegim el nombre sencer en el vector.*

*vector[i] = num;*

*}*

*// Ordenem els valors que conté el vector.*

*vector.sort();*

*// Mostrem el vector ordenat.*

*alert("El vector ordenat és: " + vector);*

*}*