PROGRAMACIÓN WEB

JavaScript

PROGRAMACIÓN WEB

PERO ANTES...

- Que vamos a ver en este Curso de Programación Web???
  - Introducción
  - HTML y XHTML
  - CSS (Cascading Style Sheets)
  - Javascript
  - XML
  - Web 1.0

JAVASCRIPT

OBJETIVOS

- Aprenderemos a:
  - Escribir un programa simple en JavaScript
  - Usar sentencias de entrada y salida
  - Entender conceptos de memoria básicos
  - Usar operadores aritméticos y entender su precedencia
  - Escribir sentencias condicionales
  - Usar operadores relacionales y de igualdad

JAVASCRIPT

- Lenguaje de Secuencia de Comandos (Scripting)
  - Mejora la funcionalidad y la apariencia
  - Scripting del lado del cliente
    - Hace que las paginas sean mas interactivas y dinamicas
  - Fundamento para Scripting del lado del servidor mas complejo
  - Desarrollo de Programas
  - Control de Programas

JAVASCRIPT – PROGRAMA SIMPLE

- Inline Scripting
  - Escrito en el cuerpo <*body*> de un documento
  - Etiqueta <*script*>
    - Indica que el texto es parte de un script
    - Atributo *type*
      - Especifica el tipo de archivo y el tipo de lenguaje Script usado
    - Método *writeln*
      - Escribe una línea en el documento
    - Carácter de escape (\)
      - Indica que un carácter especial es usado en una cadena
    - Método *alert*
      - Cuadro de Dialogo

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- welcome.html                -->
6  <!-- Displaying a line of text -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Mi primer programa en JavaScript</title>
11
12       <script type = "text/javascript">
13          <!--
14          document.writeln(
15             "<h1>Bienvenido a la Programacion en JavaScript!</h1>" );
16          // -->
17       </script>
18
19    </head><body></body>
20 </html>
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- welcome2.html                                  -->
6  <!-- Printing a Line with Multiple Statements -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Escribiendo una linea con varias sentencias</title>
11
12       <script type = "text/javascript">
13          <!--
14          document.write( "<h1 style = \"color: magenta\">" );
15          document.write( "Bienvenido a la Programacion en JavaScript " +
16             "!</h1>" );
17          // -->
18       </script>
19
20    </head><body></body>
21 </html>
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!--           welcome4.html                  -->
6  <!-- Printing multiple lines in a dialog box -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head><title>Imprimiendo Varias Lineas en un Cuadro de Dialogo</title>
10
11       <script type = "text/javascript">
12          <!--
13             window.alert( "Bienvenido a \nla Programacion \nen JavaScript!" );
14          // -->
15       </script>
16
17    </head>
18
19    <body>
20       <p>Presione Refresh (o Reload) para ejecutar este script de nuevo.</p>
21    </body>
22 </html>
```

Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Backslash. Used to represent a backslash character in a string.
`\"`	Double quote. Used to represent a double quote character in a string contained in double quotes. For example, `window.alert( "\"in quotes\"" );` displays `"in quotes"` in an `alert` dialog.
`\'`	Single quote. Used to represent a single quote character in a string. For example, `window.alert( '\'in quotes\'' );` displays `'in quotes'` in an `alert` dialog.

Some common escape sequences.

JAVASCRIPT – PAGINA DE BIENVENIDA DINAMICA

- Un Script puede adaptar el contenido a partir de la entrada de datos de un usuario o de otras variables

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- welcome5.html            -->
6  <!-- Using Prompt Boxes       -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Usando Cuadros de Alerta y de Ingreso de Datos</title>
11
12       <script type = "text/javascript">
13          <!--
14          var name; // string entered by the user
15
16          // read the name from the prompt box as a string
17          name = window.prompt( "Por favor ingrese su nombre", "GalAnt" );
18
19          document.writeln( "<h1>Hola, " + name +
20             ", bienvenido a la programacion en JavaScript!</h1>" );
21          // -->
22       </script>
```

```
23
24      </head>
25
26      <body>
27      <p>Presione Refresh (o Reload) para ejecutar este script de nuevo.</p>
28      </body>
29  </html>
```

JAVASCRIPT – SUMA DE ENTEROS

- Preguntar al usuario por dos numeros enteros y calcular la suma
- NaN (Not a Number)
- parseInt
  - Convierte el argumento string en un entero

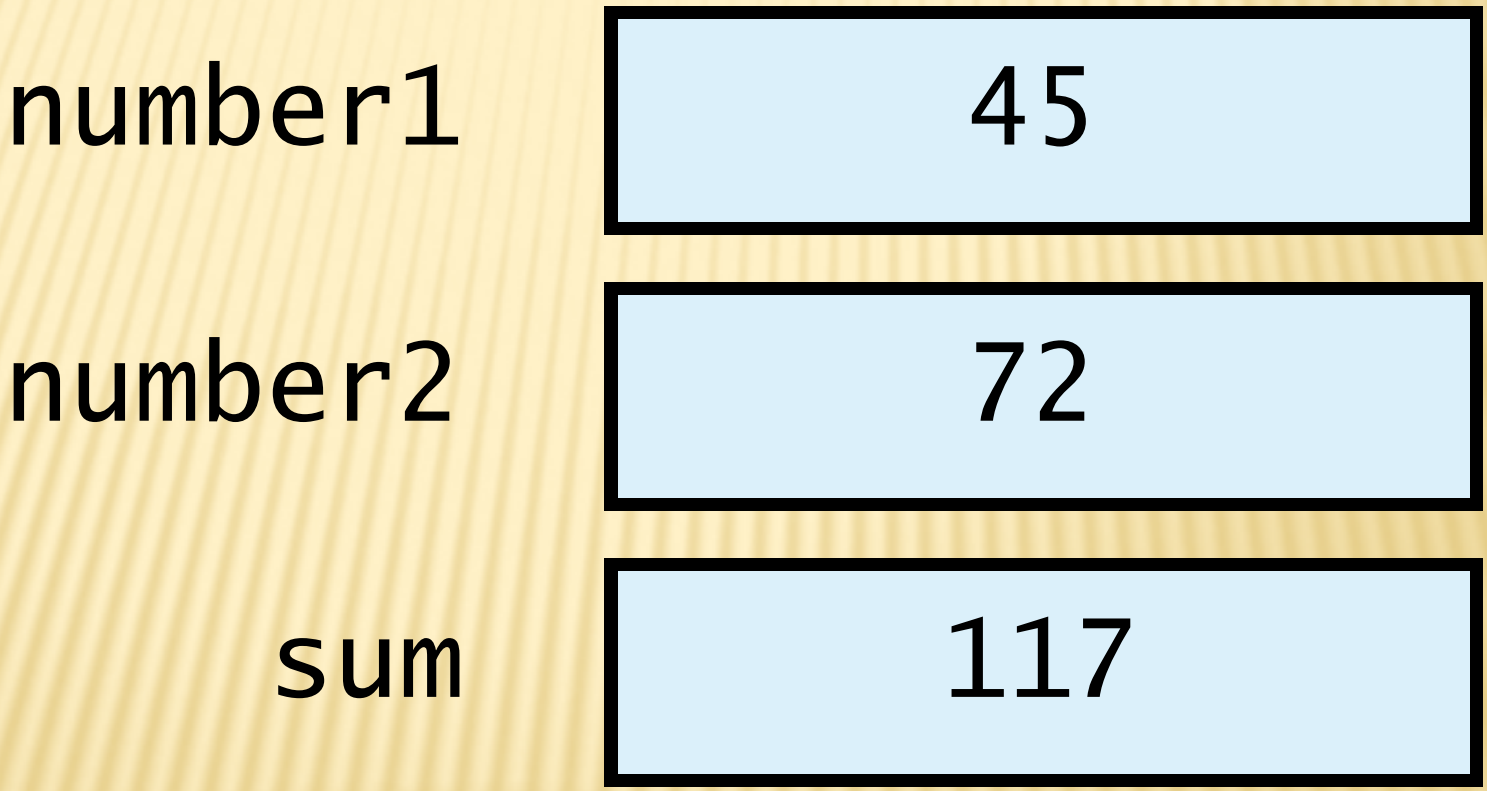
```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Addition.html -->
6  <!-- Addition Program        -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Un Programa de Suma</title>
11
12       <script type = "text/javascript">
13          <!--
14          var firstNumber,   // first string entered by user
15              secondNumber,  // second string entered by user
16              number1,       // first number to add
17              number2,       // second number to add
18              sum;           // sum of number1 and number2
19
20          // read in first number from user as a string
21          firstNumber =
22             window.prompt( "Ingrese el primer entero", "0" );
23
```

```
24          // read in second number from user as a string
25          secondNumber =
26             window.prompt( "Ingrese el segundo entero", "0" );
27
28          // convert numbers from strings to integers
29          number1 = parseInt( firstNumber );
30          number2 = parseInt( secondNumber );
31
32          // add the numbers
33          sum = number1 + number2;
34
35          // display the results
36          document.writeln( "<h1>La suma es " + sum + "</h1>" );
37          // -->
38       </script>
39
40    </head>
41    <body>
42       <p>Presione Refresh (o Reload) para ejecutar este script de nuevo</p>
43    </body>
44 </html>
```

JAVASCRIPT - MEMORIA

- Los nombres de las variables son equivalentes a direcciones en la memoria de la computadora
- Cada variable tiene un nombre, un tipo y un valor
- El valor se lee de un espacio de memoria
  - nondestructive

JAVASCRIPT - MEMORIA



JAVASCRIPT - ARITMETICA

- Varios scripts ejecutan cálculos aritméticos
  - Las expresiones en Javascript deben ser escritos en la forma de una línea recta

JAVASCRIPT - ARITMETICA

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or or $x y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Arithmetic operators.

Operator(s)	Operation(s)	Order of evaluation (precedence)
*, / or %	Multiplication Division Modulus	Evaluated second. If there are several such operations, they are evaluated from left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several such operations, they are evaluated from left to right.

Precedence of arithmetic operators.

JAVASCRIPT - ARITMETICA

Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;`

`2 * 5 is 10` *(Leftmost multiplication)*

Step 2. `y = 10 * 5 + 3 * 5 + 7;`

`10 * 5 is 50` *(Leftmost multiplication)*

Step 3. `y = 50 + 3 * 5 + 7;`

`3 * 5 is 15` *(Multiplication before addition)*

Step 4. `y = 50 + 15 + 7;`

`50 + 15 is 65` *(Leftmost addition)*

Step 5. `y = 65 + 7;`

`65 + 7 is 72` *(Last addition)*

Step 6. `y = 72;` *(Last operation—place 72 into y)*

JAVASCRIPT – TOMA DE DECISIONES

- Operadores de igualdad y operadores relacionales
  - Decisión basada en la verdad o falsedad de una condición

JAVASCRIPT – TOMA DE DECISIONES

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
Equality operators			
=	==	x == y	x is equal to y
□	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
	>=	x >= y	x is greater than or equal to y
	<=	x <= y	x is less than or equal to y

Equality and relational operators.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5  <!-- welcome6.html    -->
6  <!-- Using Relational Operators -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Usando Operadores Relacionales</title>
11
12       <script type = "text/javascript">
13          <!--
14          var name, // string entered by the user
15             now = new Date(),      // current date and time
16             hour = now.getHours(); // current hour (0-23)
17
18          // read the name from the prompt box as a string
19          name = window.prompt( "Por favor ingrese su nombre", "GalAnt" );
20
21          // determine whether it is morning
22          if ( hour < 12 )
23             document.write( "<h1>Buenos Dias, " );
24
```

```
25          // determine whether the time is PM
26          if ( hour >= 12 )
27          {
28             // convert to a 12 hour clock
29             hour = hour - 12;
30
31             // determine whether it is before 6 PM
32             if ( hour < 6 )
33                document.write( "<h1>Buenas Tardes, " );
34
35             // determine whether it is after 6 PM
36             if ( hour >= 6 )
37                document.write( "<h1>Buenas Noches, " );
38          }
39
40          document.writeln( name +
41             ", bienvenido a la programacion en JavaScritp!</h1>" );
42          // -->
43       </script>
44
45    </head>
46
47    <body>
48       <p>Presione Refresh (o Reload) para ejecutar este script de nuevo.</p>
49    </body>
50    </html>
```

JAVASCRIPT – TOMA DE DECISIONES

Operators	Associativity	Type
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
=	right to left	assignment
Precedence and associativity of the operators discussed so far.		

ESTRUCTURAS DE CONTROL

JAVASCRIPT

JAVASCRIPT – ESTRUCTURAS DE CONTROL

- Ejecucion secuencial
  - Las sentencias se ejecutan en el orden en las que fueron escritas
- Transferencia de control
  - La siguiente sentencia a ejecutar puede que no sea la siguiente en la secuencia
- Tres estructuras de control
  - Estructura de secuencia
  - Estructura de selección
    - `if`
    - `if… else`
    - `switch`
  - Estructura de repeticion
    - `while`
    - `do … while`
    - `for`
    - `for … in`

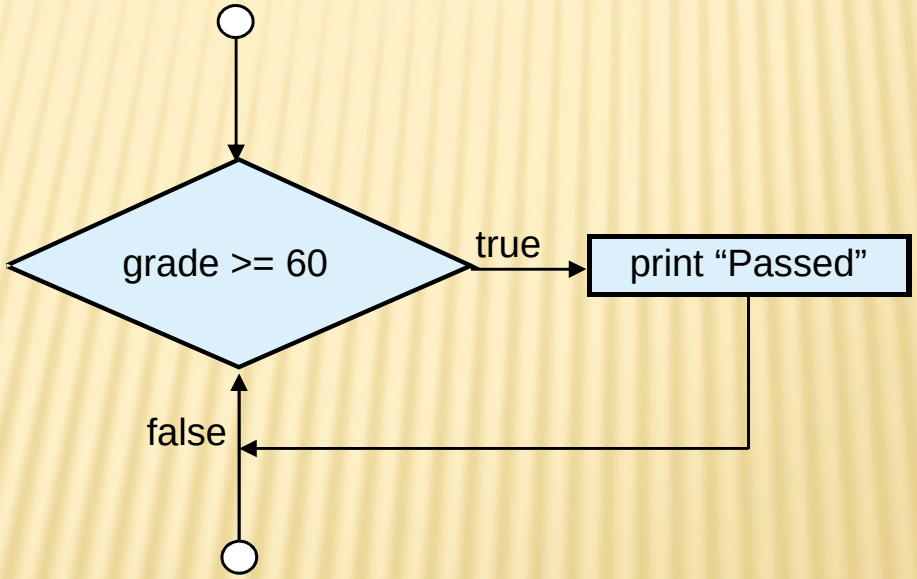
JAVASCRIPT – ESTRUCTURAS DE CONTROL

JavaScript Keywords				
break	case	catch	continue	default
delete	do	else	finally	for
function	if	in	instanceof	new
return	switch	this	throw	try
typeof	var	void	while	with
Keywords that are reserved but not currently used by JavaScript				
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile				

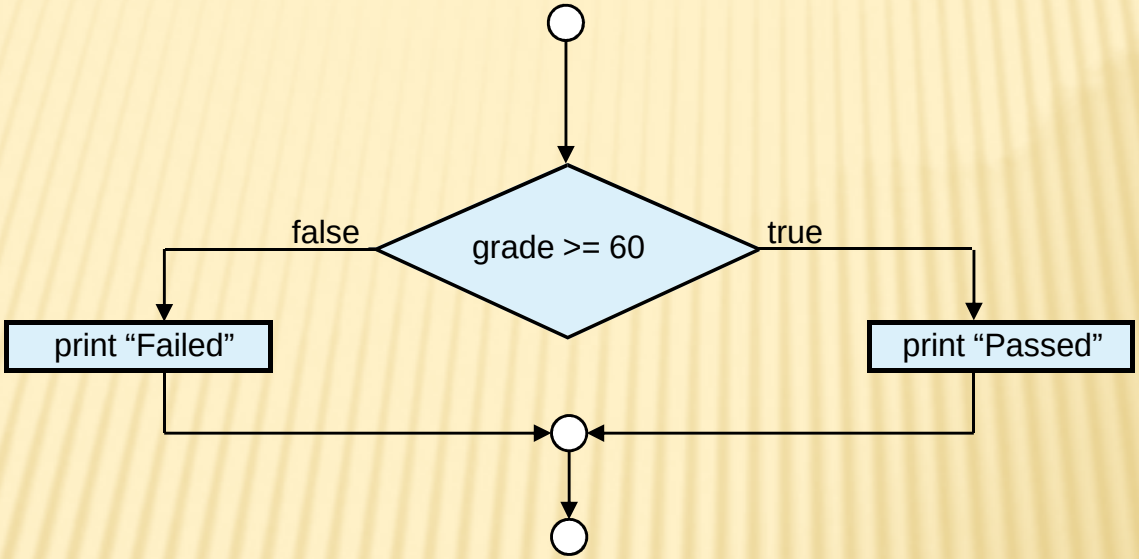
JavaScript keywords.

JAVASCRIPT - SENTENCIAS DE SELECCION

- Realiza una acción solo cuando la condición se evalúa en verdadero

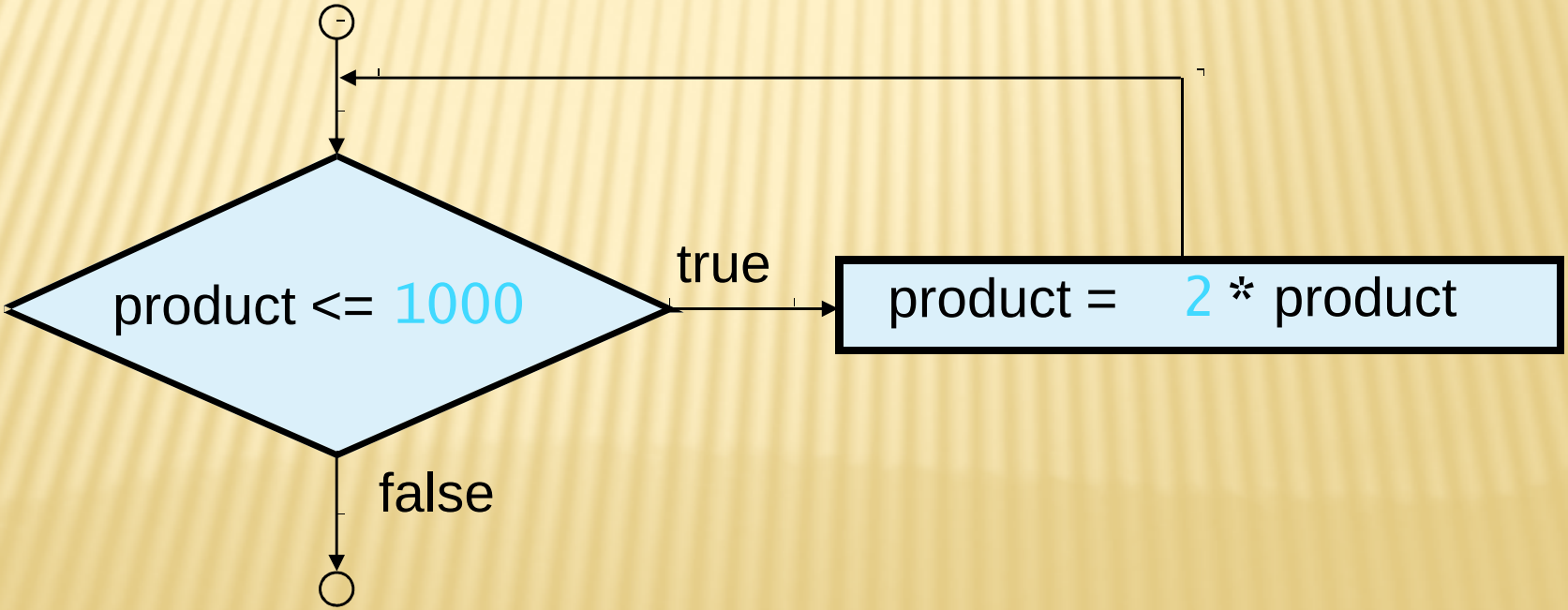


JAVASCRIPT – SENTENCIAS DE SELECCION



JAVASCRIPT - WHILE

- Estructura de repeticion
  - Repite una accion mientras la condicion continue en verdadero



```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.7: average.html -->
6  <!-- Class Average Program  -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Class Average Program</title>
11
12       <script type = "text/javascript">
13          <!--
14          var total,          // sum of grades
15              gradeCounter,   // number of grades entered
16              gradeValue,     // grade value
17              average,        // average of all grades
18              grade;          // grade typed by user
19
20          // Initialization Phase
21          total = 0;          // clear total
22          gradeCounter = 1;   // prepare to loop
23
```

```
24          // Processing Phase
25          while ( gradeCounter <= 10 ) {  // loop 10 times
26
27             // prompt for input and read grade from user
28             grade = window.prompt( "Enter integer grade:", "0" );
29
30             // convert grade from a string to an integer
31             gradeValue = parseInt( grade );
32
33             // add gradeValue to total
34             total = total + gradeValue;
35
36             // add 1 to gradeCounter
37             gradeCounter = gradeCounter + 1;
38          }
39
40          // Termination Phase
41          average = total / 10;  // calculate the average
42
43          // display average of exam grades
44          document.writeln(
45             "<h1>Class average is " + average + "</h1>" );
46          // -->
47       </script>
```

```
47    <body>
48       <p>Presione Refresh (o Reload) para ejecutar este script de nuevo.</p>
49    </body>
50    </html>
```

JAVASCRIPT - WHILE

- Repeticion controlada por un contador
  - Contador
    - Controla el numero de veces que un conjunto de sentencias se ejecutan
  - Repeticion definida
- Repeticion controlada por un centinela
  - Repeticion indefinida

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.9: average2.html        -->
6  <!-- Sentinel-controlled Repetition -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Class Average Program:
11           Sentinel-controlled Repetition</title>
12
13       <script type = "text/javascript">
14          <!--
15          var gradeCounter,  // number of grades entered
16              gradeValue,    // grade value
17              total,         // sum of grades
18              average,       // average of all grades
19              grade;         // grade typed by user
20
21          // Initialization phase
22          total = 0;         // clear total
23          gradeCounter = 0;  // prepare to loop
24
```

```
25          // Processing phase
26          // prompt for input and read grade from user
27          grade = window.prompt(
28               "Ingrese una nota, -1 para Salir:", "0" );
29
30          // convert grade from a string to an integer
31          gradeValue = parseInt( grade );
32
33          while ( gradeValue != -1 ) {
34             // add gradeValue to total
35             total = total + gradeValue;
36
37             // add 1 to gradeCounter
38             gradeCounter = gradeCounter + 1;
39
40             // prompt for input and read grade from user
41             grade = window.prompt(
42                  "Ingrese una nota, -1 para Salir:", "0" );
43
44             // convert grade from a string to an integer
45             gradeValue = parseInt( grade );
46          }
47
```

```
48          // Termination phase
49          if ( gradeCounter != 0 ) {
50             average = total / gradeCounter;
51
52             // display average of exam grades
53             document.writeln(
54                "<h1>El promedio de la Clase es " + average + "</h1>" );
55          }
56          else
57             document.writeln( "<p>Ninguna nota ha sido ingresada</p>" );
58          // -->
59       </script>
60    </head>
61
62    <body>
63       <p> Presione Refresh (o Reload) para ejecutar este script de nuevo </p>
64    </body>
65 </html>
```

JAVASCRIPT – ESTRUCTURAS DE CONTROL

- Estructuras de Control Anidadas

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.11: analysis.html -->
6  <!-- Analyzing Exam Results   -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Analisis de los Resultados del Examen</title>
11
12       <script type = "text/javascript">
13          <!--
14          // initializing variables in declarations
15          var passes = 0,      // number of passes
16              failures = 0,    // number of failures
17              student = 1,     // student counter
18              result;          // one exam result
19
20          // process 10 students; counter-controlled loop
21          while ( student <= 10 ) {
22             result = window.prompt(
23                "Ingrese el resultado (1=paso,2=fallo)", "0" );
24
```

```
25             if ( result == "1" )
26                passes = passes + 1;
27             else
28                failures = failures + 1;
29
30             student = student + 1;
31          }
32
33          // termination phase
34          document.writeln( "<h1>Resultados del Examen</h1>" );
35          document.writeln(
36             "Pasaron: " + passes + "<br />Aplazados: " + failures );
37
38          if ( passes > 8 )
39             document.writeln( "<br />Pedir Aumento" );
40          // -->
41       </script>
42
43    </head>
44    <body>
45       <p> Presione Refresh (o Reload) para ejecutar este script de nuevo </p>
46    </body>
47 </html>
```

JAVASCRIPT – OPERADORES DE ASIGNACIÓN

- Operadores de asignación compuestos
  - Expresiones de asignación abreviadas

Assignment operator	Initial value of variable	Sample expression	Explanation	Assigns
+=	c = 3	c += 7	c = c + 7	10 to c
-=	d = 5	d -= 4	d = d - 4	1 to d
*=	e = 4	e *= 5	e = e * 5	20 to e
/=	f = 6	f /= 3	f = f / 3	2 to f
%=	g = 12	g %= 9	g = g % 9	3 to g

Arithmetic assignment operators.

JAVASCRIPT – MAS OPERADORES

- Operadores de incremento y de disminución
  - Operadores PRE
    - Operador colocado antes de una variable
  - Operadores POST
    - Operador colocado después de una variable

JAVASCRIPT – MAS OPERADORES

Operator	Called	Sample expression	Explanation
`++`	preincrement	`++a`	Increment `a` by 1, then use the new value of `a` in the expression in which `a` resides.
`++`	postincrement	`a++`	Use the current value of `a` in the expression in which `a` resides, then increment `a` by 1.
`--`	predecrement	`--b`	Decrement `b` by 1, then use the new value of `b` in the expression in which `b` resides.
`--`	postdecrement	`b--`	Use the current value of `b` in the expression in which `b` resides, then decrement `b` by 1.

increment and decrement operators.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.14: increment.html            -->
6  <!-- Preincrementing and Postincrementing -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Preincrementing and Postincrementing</title>
11
12       <script type = "text/javascript">
13          <!--
14          var c;
15
16          c = 5;
17          document.writeln( "<h3>Postincrementing</h3>" );
18          document.writeln( c );            // print 5
19          // print 5 then increment
20          document.writeln( "<br />" + c++ );
21          document.writeln( "<br />" + c );   // print 6
22
23          c = 5;
24          document.writeln( "<h3>Preincrementing</h3>" );
25          document.writeln( c );            // print 5
```

```
26          // increment then print 6
27          document.writeln( "<br />" + ++c );
28          document.writeln( "<br />" + c );   // print 6
29          // -->
30       </script>
31
32    </head><body></body>
33 </html>
```

JAVASCRIPT

Operator	Associativity	Type
++ --	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment
Precedence and associativity of the operators discussed so far.		

JAVASCRIPT – TIPOS DE DATOS

- Loosely typed
  - Automáticamente convierte entre valores de diferentes tipos
  - Cuando la variable es declarada esta tienen un valor no definido (*undefined*)
  - Para indicar que una variable no contiene un valor se puede asignar el valor *null* a la variable

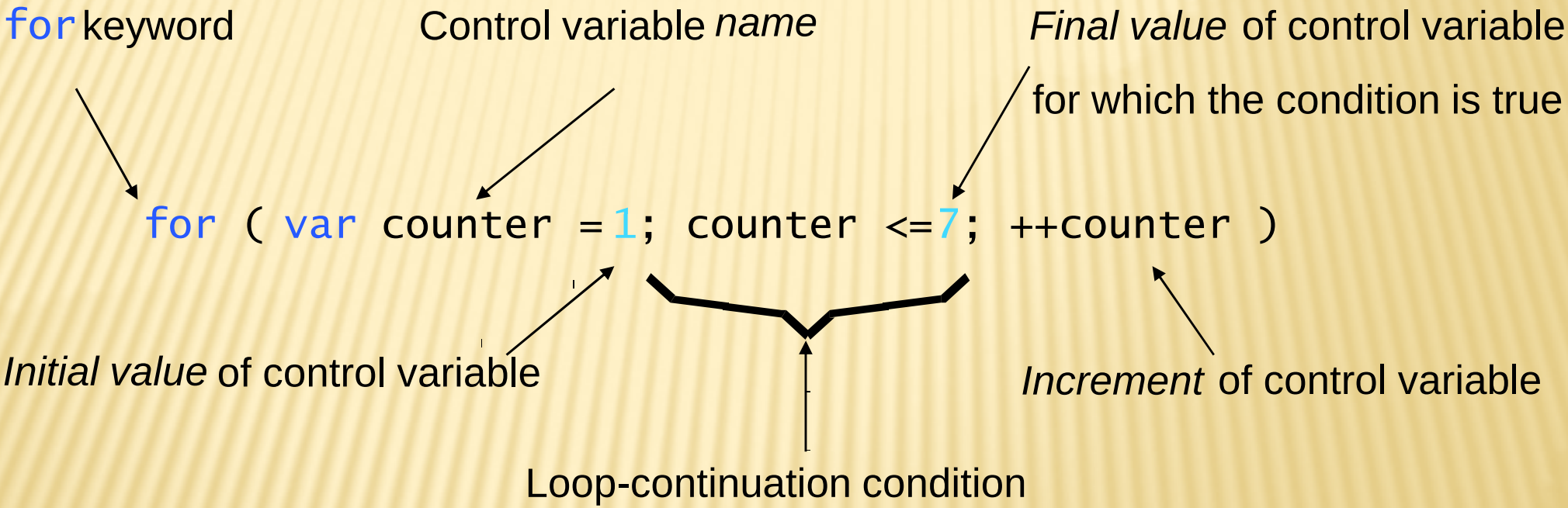
Estructuras de Control 2

JAVASCRIPT

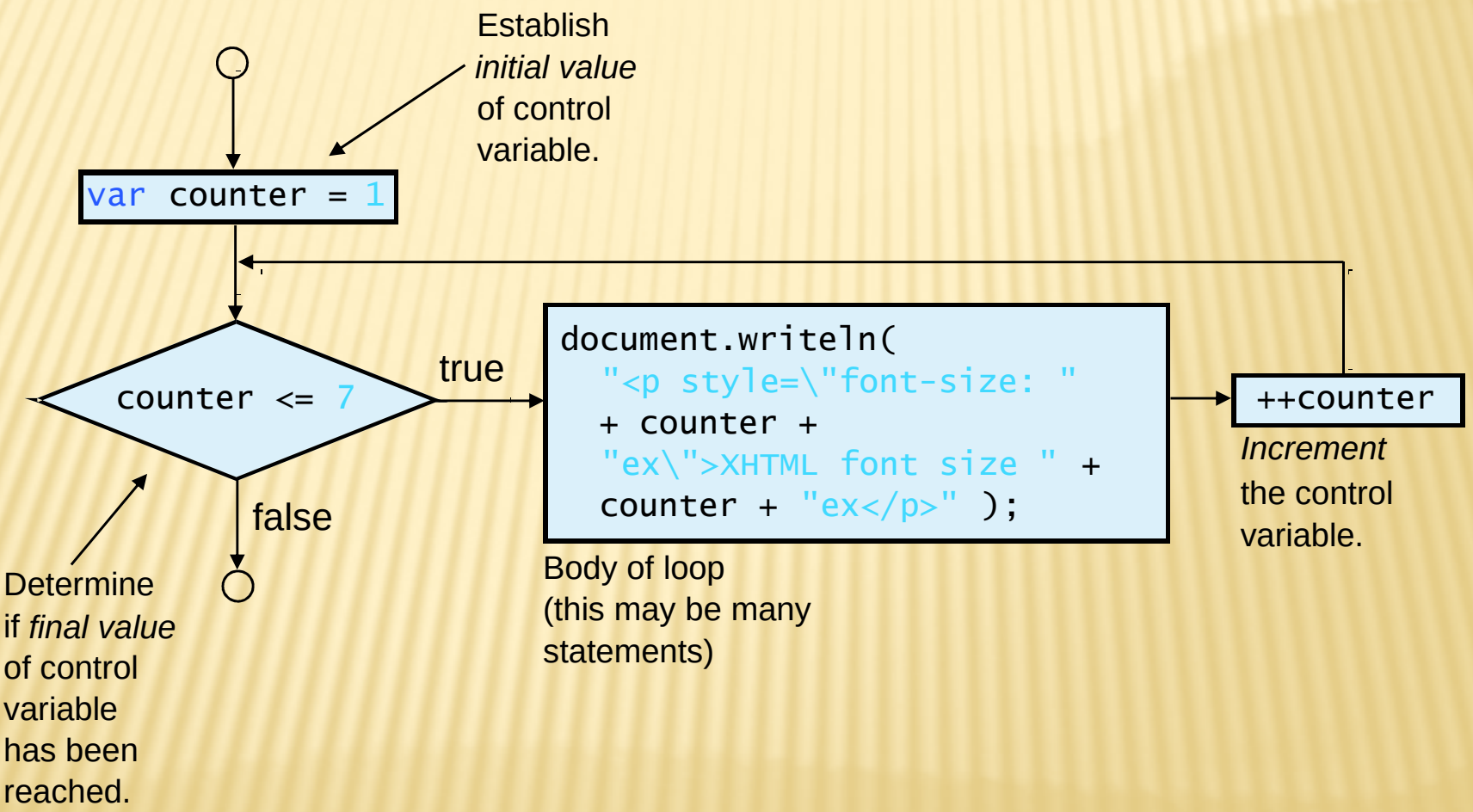
JAVASCRIPT – REPETICION (FOR)

- Estructura de repetición *for*
  - Maneja todos los detalles de una repetición controlada por un contador
  - Cabecera de la estructura for
    - La primera línea

JAVASCRIPT – REPETICION (FOR)



JAVASCRIPT – REPETICION (FOR)



```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.2: ForCounter.html                          -->
6  <!-- Counter-Controlled Repetition with for statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Counter-Controlled Repetition</title>
11
12       <script type = "text/javascript">
13          <!--
14          // Initialization, repetition condition and
15          // incrementing are all included in the for
16          // statement header.
17          for ( var counter = 1; counter <= 7; ++counter )
18             document.writeln( "<p style = \"font-size: " +
19                counter + "ex\">XHTML font size " + counter +
20                "ex</p>" );
21          // -->
22       </script>
23
24    </head><body></body>
25 </html>
```

JAVASCRIPT – EJEMPLOS (FOR)

- Sumas con la estructura *for*
- Calculo de interés compuesto
  - Objeto *Math*
    - Método *pow*
    - Método *round*

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.5: Sum.html                    -->
6  <!-- Using the for repetition statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Sum the Even Integers from 2 to 100</title>
11
12       <script type = "text/javascript">
13          <!--
14          var sum = 0;
15
16          for ( var number = 2; number <= 100; number += 2 )
17             sum += number;
18
19          document.writeln( "The sum of the even integers " +
20             "from 2 to 100 is " + sum );
21          // -->
22       </script>
23
24    </head><body></body>
25 </html>
```

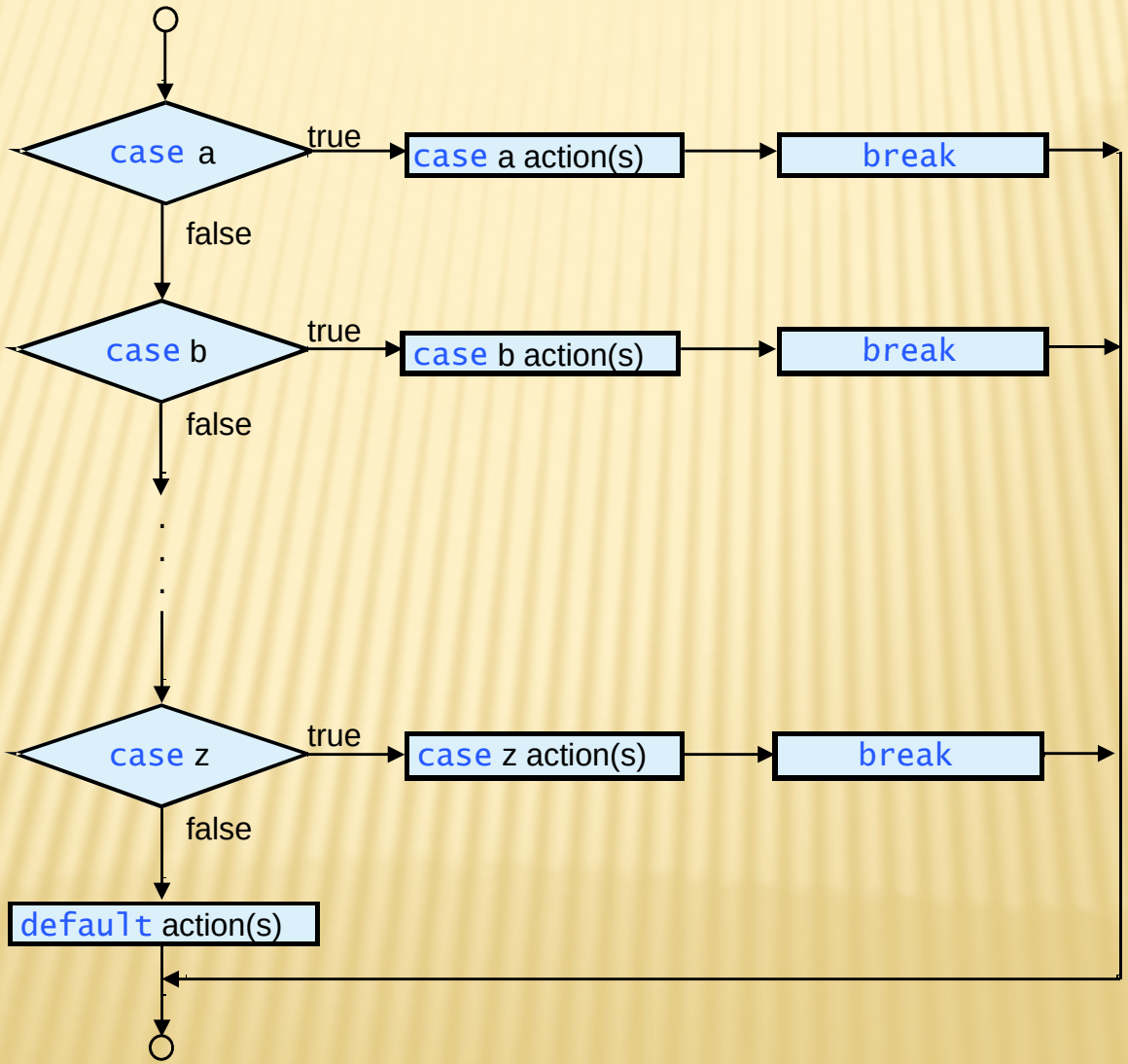
```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.6: Interest.html              -->
6  <!-- Using the for repetition statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Calculating Compound Interest</title>
11
12       <script type = "text/javascript">
13          <!--
14          var amount, principal = 1000.0, rate = .05;
15
16          document.writeln(
17             "<table border = \"1\" width = \"100%\">" );
18          document.writeln(
19             "<caption>Calculating Compound Interest</caption>" );
20          document.writeln(
21             "<thead><tr><th align = \"left\">Year</th>" );
22          document.writeln(
23             "<th align = \"left\">Amount on deposit</th>" );
24          document.writeln( "</tr></thead>" );
25
```

```
26          for ( var year = 1; year <= 10; ++year ) {
27             amount = principal * Math.pow( 1.0 + rate, year );
28             document.writeln( "<tbody><tr><td>" + year +
29                "</td><td>" + Math.round( amount * 100 ) / 100 +
30                "</td></tr>" );
31          }
32
33          document.writeln( "</tbody></table>" );
34          // -->
35       </script>
36
37    </head><body></body>
38 </html>
```

JAVASCRIPT – MULTIPLE SELECCIÓN (SWITCH)

- Expresión de Control
- Etiquetas *case*
- Caso por defecto

JAVASCRIPT – MULTIPLE SELECCIÓN (SWITCH)



```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- SwitchTest.html              -->
6  <!-- Using the switch statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Cambiando entre formatos de lista XHTML</title>
11
12       <script type = "text/javascript">
13          <!--
14          var choice,              // user's choice
15              startTag,            // starting list item tag
16              endTag,              // ending list item tag
17              validInput = true,   // indicates if input is valid
18              listType;            // list type as a string
19
20          choice = window.prompt( "Seleccione un estilo de lista:\n" +
21              "1 (bullet), 2 (numbered), 3 (lettered)", "1" );
22
```

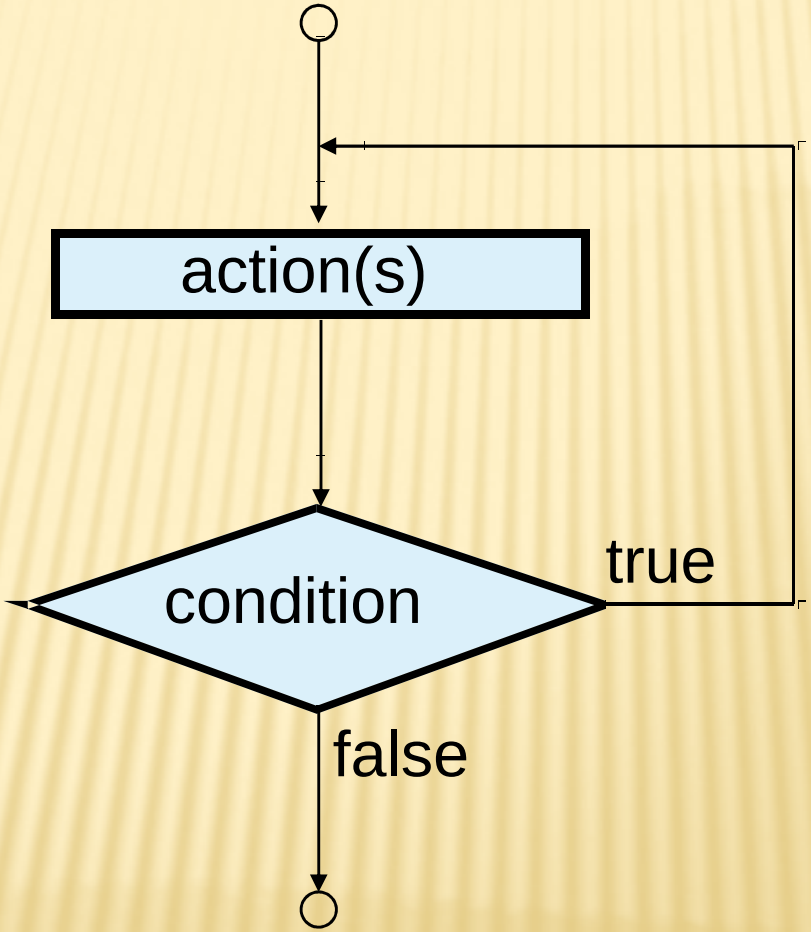
```
23          switch ( choice ) {
24             case "1":
25                startTag = "<ul>";
26                endTag = "</ul>";
27                listType = "<h1>Bullet List</h1>";
28                break;
29             case "2":
30                startTag = "<ol>";
31                endTag = "</ol>";
32                listType = "<h1>Ordered List: Numbered</h1>";
33                break;
34             case "3":
35                startTag = "<ol type = \"A\">";
36                endTag = "</ol>";
37                listType = "<h1>Ordered List: Lettered</h1>";
38                break;
39            default:
40                validInput = false;
41          }
42
43          if ( validInput == true ) {
44             document.writeln( listType + startTag );
45
46             for ( var i = 1; i <= 3; ++i )
47                document.writeln( "<li>List item " + i + "</li>" );
```

```
48
49             document.writeln( endTag );
50          }
51          else
52             document.writeln( "Opcion invalida: " + choice );
53          // -->
54       </script>
55
56    </head>
57    <body>
58       <p> Presione Refresh (o Reload) para ejecutar este script de nuevo </p>
59    </body>
60 </html>
```

JAVASCRIPT – REPETICION (DO...WHILE)

- Similar al *while*
- Evalúa la condición de repetición luego de que se ejecute el cuerpo de la misma
- El cuerpo de la repetición siempre se ejecuta al menos una vez

JAVASCRIPT – REPETICION (DO...WHILE)



```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.9: DoWhileTest.html     -->
6  <!-- Using the do...while statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Using the do...while Repetition Statement</title>
11
12       <script type = "text/javascript">
13          <!--
14          var counter = 1;
15
16          do {
17             document.writeln( "<h" + counter + ">This is " +
18                "an h" + counter + " level head" + "</h" +
19                counter + ">" );
20
21             ++counter;
22          } while ( counter <= 6 );
23          // -->
24       </script>
25
26    </head><body></body>
27 </html>
```

JAVASCRIPT – BREAK Y CONTINUE

- *break*
  - Salir inmediatamente de una estructura
  - Usado comúnmente para salir de un bucle
  - Salta el resto de una declaración *switch*
- *continue*
  - Salta las declaraciones restantes en el cuerpo de una estructura
  - Procede con la siguiente iteración de un bucle

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.11: BreakTest.html   -->
6  <!-- Using the break statement   -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>
11          Using the break Statement in a for Structure
12       </title>
13
14       <script type = "text/javascript">
15          <!--
16          for ( var count = 1; count <= 10; ++count ) {
17             if ( count == 5 )
18                break;  // break loop only if count == 5
19
20             document.writeln( "Count is: " + count + "<br />" );
21          }
22
```

```
23          document.writeln(
24             "Broke out of loop at count = " + count );
25          // -->
26       </script>
27
28    </head><body></body>
29 </html>
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.12: ContinueTest.html  -->
6  <!-- Using the break statement     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>
11          Using the continue Statement in a for Structure
12       </title>
13
14       <script type = "text/javascript">
15          <!--
16          for ( var count = 1; count <= 10; ++count ) {
17             if ( count == 5 )
18                continue;  // skip remaining code in loop
19                           // only if count == 5
20
21             document.writeln( "Count is: " + count + "<br />" );
22          }
23
```

```
24            document.writeln( "Used continue to skip printing 5" );
25            // -->
26         </script>
27
28      </head><body></body>
29   </html>
```

JAVASCRIPT – ETIQUETAS BREAK Y CONTINUE

- Declaración de etiquetas break
  - Salir de un conjunto de estructuras anidadas
  - Salida inmediata de la estructura en la que se encuentra y las estructuras que la encierran
  - La ejecución continua con la primera declaración después de la etiqueta
- Declaración de etiquetas continue
  - Salta las declaraciones restantes en el cuerpo de la estructura
  - Procede con la siguiente iteración de la estructura de repetición encerrada por la etiqueta

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.13: BreakLabelTest.html         -->
6  <!-- Using the break statement with a Label -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Using the break Statement with a Label</title>
11
12       <script type = "text/javascript">
13          <!--
14          stop: {   // labeled block
15             for ( var row = 1; row <= 10; ++row ) {
16                for ( var column = 1; column <= 5 ; ++column ) {
17
18                   if ( row == 5 )
19                      break stop; // jump to end of stop block
20
21                   document.write( "* " );
22                }
23
24                document.writeln( "<br />" );
25             }
```

```
26
27             // the following line is skipped
28             document.writeln( "This line should not print" );
29          }
30
31          document.writeln( "End of script" );
32          // -->
33       </script>
34
35    </head><body></body>
36 </html>
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- ContinueLabelTest.html            -->
6  <!-- Using the continue statement      -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Usando la declaracion continue con una etiqueta</title>
11
12       <script type = "text/javascript">
13          <!--
14          nextRow:   // target label of continue statement
15             for ( var row = 1; row <= 5; ++row ) {
16                document.writeln( "<br />" );
17
18                for ( var column = 1; column <= 10; ++column ) {
19
20                   if ( column > row )
21                      continue nextRow; // next iteration of
22                                        // labeled loop
23
24                   document.write( "* " );
25                }
```

```
26              }
27            // -->
28        </script>
29
30    </head><body></body>
31 </html>
```

JAVASCRIPT – OPERADORES LOGICOS

- Logical AND (&&)
- Logical OR (||)
- Logical NOT (!)

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- LogicalOperators.html               -->
6  <!-- Demonstrating Logical Operators     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Demostrando los operadores logicos</title>
11
12       <script type = "text/javascript">
13          <!--
14          document.writeln(
15             "<table border = \"1\" width = \"100%\">" );
16
17          document.writeln(
18             "<caption>Demostrando los Operadores " +
19             "Logicos</caption" );
20
21          document.writeln(
22             "<tr><td width = \"25%\">Logical AND (&&)</td>" +
23             "<td>false && false: " + ( false && false ) +
24             "<br />false && true: " + ( false && true ) +
25             "<br />true && false: " + ( true && false ) +
```

```
26             "<br />true && true: " + ( true && true ) +
27             "</td>" );
28
29          document.writeln(
30             "<tr><td width = \"25%\">Logical OR (||)</td>" +
31             "<td>false || false: " + ( false || false ) +
32             "<br />false || true: " + ( false || true ) +
33             "<br />true || false: " + ( true || false ) +
34             "<br />true || true: " + ( true || true ) +
35             "</td>" );
36
37          document.writeln(
38             "<tr><td width = \"25%\">Logical NOT (!)</td>" +
39             "<td>!false: " + ( !false ) +
40             "<br />!true: " + ( !true ) + "</td>" );
41
42          document.writeln( "</table>" );
43          // -->
44       </script>
45
46    </head><body></body>
47 </html>
```

JAVASCRIPT – OPERADORES LOGICOS

Operator	Associativity	Type
++ -- !	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical AND
||	left to right	logical OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Fig. 9.19 Precedence and associativity of the operators discussed so far.

Funciones y Metodos

JAVASCRIPT

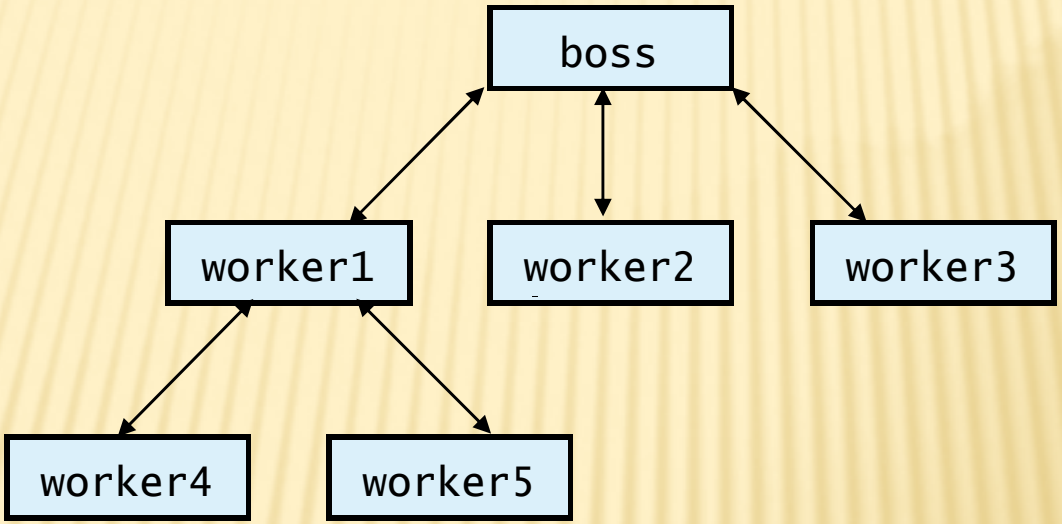
JAVASCRIPT

- Módulos en Javascript
  - Funciones
  - Métodos
    - Pertenecen a un objeto
  - Javascript incluye varios métodos predefinidos
    - Combinar con métodos propios para hacer un programa

JAVASCRIPT - FUNCIONES

- Funciones
  - Iniciada por una llamada a una función
  - Recibe información necesaria via argumentos (parametros)
  - Relación jefe-empleado
    - La función que llama
    - Función llamada
    - Retornar valor cuando termine
    - Puede tener varias capas

JAVASCRIPT - FUNCIONES



JAVASCRIPT - FUNCIONES

- Llamando a una función
  - Nombre
  - Paréntesis izquierdo
  - Argumentos separados por comas
    - Constantes, variables o expresiones
  - Paréntesis derecho
  - Ejemplos:

```
total += parseFloat( inputValue );
total += parseFloat( s1 + s2 );
```

JAVASCRIPT - FUNCIONES

- Definiendo funciones
  - Todas las variables declaradas en una función son llamadas variables locales
    - No existen fuera de la función
  - Parametros
    - También son variables locales
  - Promueve reusabilidad
    - Nombre claro

JAVASCRIPT - FUNCIONES

- Formato de una definicion de funcion

```
function function-name( parameter-list )
{
    declarations and statements
}
```

  - Function-name, cualquier identificador valido
  - Parameter-list, nombres de variables que recibiran argumentos
    - Debe tener la misma cantidad como la funcion que llama
    - Puede ser vacio
  - Declaraciones
    - Cuerpo de la funcion (bloque de codigo)

JAVASCRIPT - FUNCIONES

- Retornando el control
  - Sentencia *return*
  - Puede retornar un valor o nada

    `return` *expression*`;`
  - Si no hay una declaración de *return* es lo mismo que *return;*
  - Si no se retorna un valor cuando es esperado da un error

JAVASCRIPT - FUNCIONES

- Ejemplo: Escribiendo una funcion que eleve al cuadrado un numero
  - Bucle for de 1 a 10
  - Pasar cada numero como argumento de la funcion
  - Retornar (return) el valor del argumento multiplicado por el mismo
  - Mostrar el resultado

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- SquareInt.html            -->
6  <!-- Square function           -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>A Programmer-Defined square Function</title>
11
12       <script type = "text/javascript">
13          <!--
14          document.writeln(
15             "<h1>Square the numbers from 1 to 10</h1>" );
16
17          // square the numbers from 1 to 10
18          for ( var x = 1; x <= 10; ++x )
19             document.writeln( "El cuadrado de " + x + " es " +
20                square( x ) + "<br />" );
21
```

Llamando a la funcion square y psando el valor de x.

```
22          // The following square function's body is executed
23          // only when the fu
24
25          // square function definition
26          function square( y )
27          {
28             return y * y;
29          }
30          // -->
31       </script>
32
33    </head><body></body>
34 </html>
```

Variable y obtiene el valor de la variable x.

La sentencia `return` pasa el valor de `y * y` a la funcion que origino la llamada.

JAVASCRIPT – NUMEROS ALEATORIOS

- Generacion de numeros aleatorios
  - Math.random

```
var randomValue = Math.random();
```

  - El valor entre 0 y 1
  - Ajustar el rango
  - Math.floor
    - Redondear hacia abajo

```
Math.floor( 1 + Math.random() * 6 )
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- RandomInt.html                    -->
6  <!-- Demonstrating the Random method -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Shifted and Scaled Random Integers</title>
11
12       <script type = "text/javascript">
13          <!--
14          var value;
15
16          document.writeln(
17             "<table border = \"1\" width = \"50%\">" );
18          document.writeln(
19             "<caption>Numeros Aleatorios</caption><tr>" );
20
```

```
21          for ( var i = 1; i <= 20; i++ ) {
22             value = Math.floor( 1 + Math.random() * 6 );
23             document.writeln( "<td>" + value + "</td>" );
24
25             // write end and start <tr> tags when
26             // i is a multiple of 5 and not 20
27             & i != 20 )
28             ln( "</tr><tr>" );
29          }
30
31          document.writeln( "</tr></table>" );
32          // -->
33       </script>
34
35    </head>
36    <body>
37       <p>Click Refresh (or Reload) to run the script again</p>
38    </body>
39 </html>
```

El metodo floor redondea el numero generado por el metodo random.

El bucle for crea 20 celdas (4 filas x 5 columnas).

Cada celda es llenada con un numero aleatorio generado por el metodo random.

JAVASCRIPT – REGLAS DEL ALCANCE

- Alcance (Scope)
  - Porcion del programa donde el identificador puede ser referenciado
  - Dentro de la funcion es local
    - Identificadores existen solo entre llaves
    - Las variables locales ocultan las globales

JAVASCRIPT – REGLAS DEL ALCANCE

- Demostracion del alcance
  - Variable Global x inicializada en 1
  - *start* tiene una variable local x inicializada en 5
  - *functionA* tiene la variable local x inicializada en 25
  - *functionB* no tiene ninguna variable local x
  - Observar el resultado de cada funcion

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 10.8: scoping.html    -->
6  <!-- Local and Global Variables -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>A Scoping Example</title
11
12       <script type = "text/javascript" >
13          <!--
14          var x = 1;       // global vari
15
16          function start()
17          {
18             var x = 5;    // variable local to function start
19
20             document.writeln( "local x in start is " + x );
21
22             functionA(); // functionA has local x
23             functionB(); // functionB uses global variable x
24             functionA(); // functionA reinitializes local x
25             functionB(); // global variable x retains its value
```

Para empezar el programa, la variable x es inicializado en 1.

La funcion `start` cambia el valor de x a 5.

```
26
27          document.writeln(
28             "<p>local x in start is " + x + "</p>" );
29       }
30
31       function functionA()
32       {
33          var x = 25;  // initialized each time
34                       // functionA is called
35
36          document.writeln( "<p>lo
37                            x + " after entering functionA" );
38          ++x;
39          document.writeln( "<br />local x in functionA is " +
40             x + "                                          );
41       }
42
```

La Funcion `functionA` cambia el valor de x a 25.

El valor de x es incrementado.

```
43          function functionB()
44          {
45             document.writeln( "<p>global variable x is " + x +
46                " on entering functionB" );
47             x *= 10;
48             document.writeln( "<br />global variable x is " +
49                x + " on exiting functionB"  + "</p>" );
50          }
51          // -->
52       </script>
53
54    </head>
55    <body onload = "start()"></body>
56 </html>
```

La Funcion `functionB` multiplica el valor de x por 10.

JAVASCRIPT – FUNCIONES GLOBALES

- Objeto Global
  - Siempre disponible
  - Provee 7 métodos
  - No necesita que se haga referencia de forma explícita antes de la llamada a un metodo
  - Tambien contiene todas las variables globales y funciones definidas por el usuario

JAVASCRIPT – FUNCIONES GLOBALES

Global function	Description
`escape`	This function takes a string argument and returns a string in which all spaces, punctuation, accent characters and any other character that is not in the ASCII character set (see Appendix D, ASCII Character Set) are encoded in a hexadecimal format (see Appendix E, Number Systems) that can be represented on all platforms.
`eval`	This function takes a string argument representing JavaScript code to execute. The JavaScript interpreter evaluates the code and executes it when the `eval` function is called. This function allows JavaScript code to be stored as strings and executed dynamically.
`isFinite`	This function takes a numeric argument and returns `true` if the value of the argument is not `NaN`, `Number.POSITIVE_INFINITY` or `Number.NEGATIVE_INFINITY`; otherwise, the function returns `false`.
`isNaN`	This function takes a numeric argument and returns `true` if the value of the argument is not a number; otherwise, it returns `false`. The function is commonly used with the return value of `parseInt` or `parseFloat` to determine whether the result is a proper numeric value.

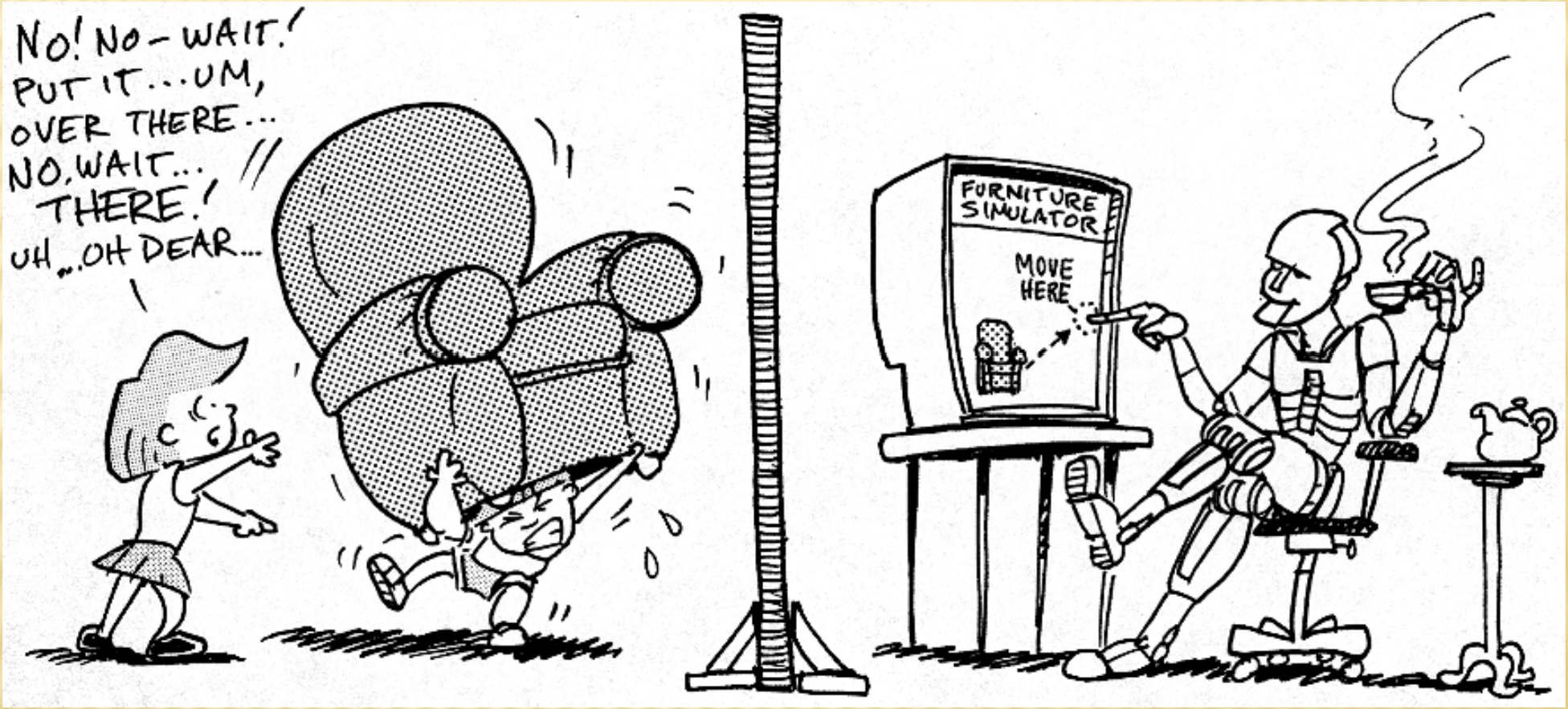
Fig. 10.9 JavaScript global functions.

JAVASCRIPT – FUNCIONES GLOBALES

Global function	Description
`parseFloat`	This function takes a string argument and attempts to convert the beginning of the string into a floating-point value. If the conversion is unsuccessful, the function returns `NaN`; otherwise, it returns the converted value (e.g., `parseFloat( "abc123.45" )` returns `NaN`, and `parseFloat( "123.45abc" )` returns the value `123.45`).
`parseInt`	This function takes a string argument and attempts to convert the beginning of the string into an integer value. If the conversion is unsuccessful, the function returns `NaN`; otherwise, it returns the converted value (e.g., `parseInt( "abc123" )` returns `NaN`, and `parseInt( "123abc" )` returns the integer value `123`). This function takes an optional second argument, from 2 to 36, specifying the **radix** (or **base**) of the number. Base 2 indicates that the first argument string is in **binary** format, base 8 indicates that the first argument string is in **octal** format and base 16 indicates that the first argument string is in **hexadecimal** format. See see Appendex E, Number Systems, for more information on binary, octal and hexadecimal numbers.
`unescape`	This function takes a string as its argument and returns a string in which all characters previously encoded with `escape` are decoded.

Fig. 10.9 JavaScript global functions.

PRACTICA


NO! NO - WAIT.
PUT IT ... UM,
OVER THERE ...
NO, WAIT ...
THERE!
UH .. OH DEAR ...
FURNITURE SIMULATOR
MOVE HERE