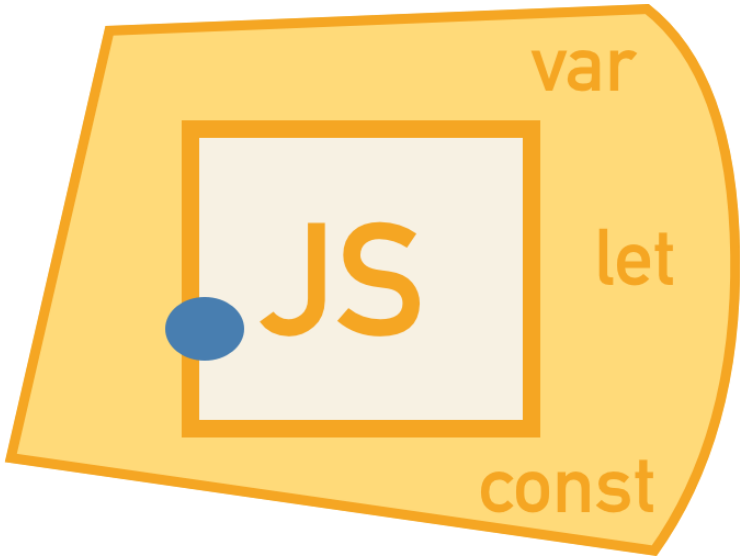
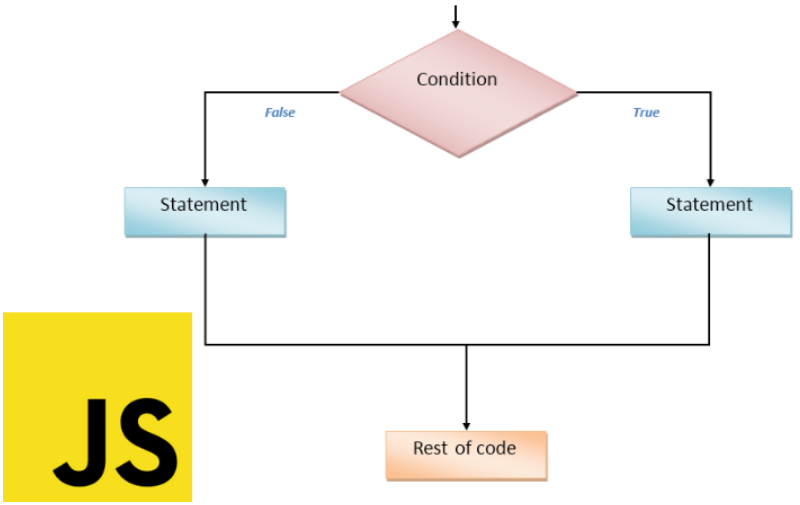
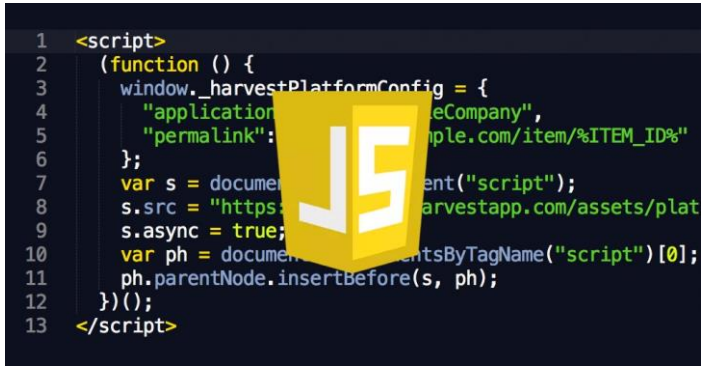
Curso JavaScript 6.0

Rafael Herrera García

Octubre 2019







Tema 2

Construcciones básicas

Tipos de datos básicos

- En Javascript tenemos los siguientes tipos de datos básicos:
  - Numéricos (Enteros y flotantes)
  - Carácter
  - Booleanos
  - Arreglos
  - Objetos
  - Valores especiales

Literales

Tipo de dato	Formato	Ejemplo
Numérico entero	Decimal	738,989,918929389,81
Numérico entero	Octal	045,03721,0342
Numérico entero	Hexadecimal	0x4f67d, 0X32A4F, 0x45BFC
Numérico entero	Binario	0b01001011, 0B011110
Numérico flotante	Decimal	32.234, 32.4, 3.1212e-12, 1.432e5
Carácter	Un solo carácter	‘a’, ‘\\’, ‘\n’, “f”
Carácter	Varios caracteres	“hola mundo”, ‘Esto es otra cadena’
Booleanos		true, false
Arreglos		[32,43,54,12], [“maría”,”ana”,”luisa”]
Especiales		undefined, null, NaN

Variables y constantes

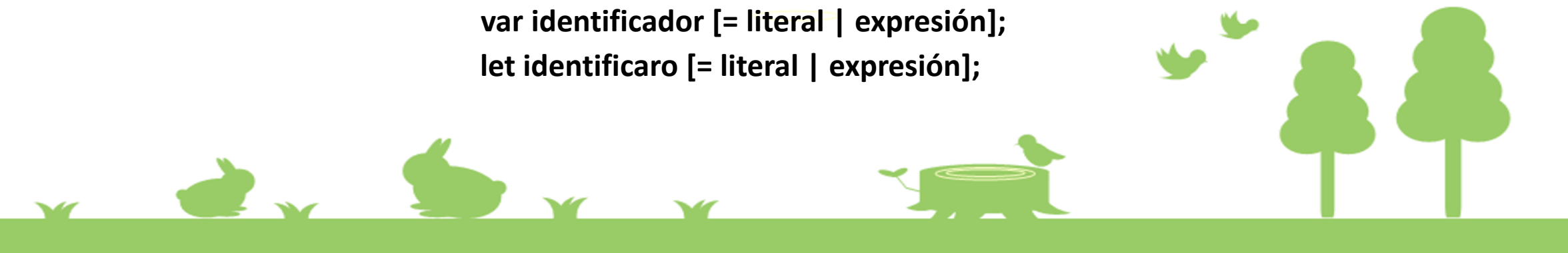
- Al igual que cualquier lenguaje moderno, Javascript soporta inmutabilidad y mutabilidad de los datos.
- Constantes son variables inmutables
  - Se declaran con la palabra reservada const

const identificador= literal|expresión;

- Variables son mutables
  - Se declaran con las palabras reservadas var o let.

var identificador [= literal | expresión];

let identificaro [= literal | expresión];



Expresiones

- Están formadas por operandos (variables, literales, constantes, funciones) y operadores.
- Los operadores, definen los tipos de datos de los operandos.
- Las expresiones pueden ser muy simples desde el uso de un operador hasta muy complejas con combinación de operadores.

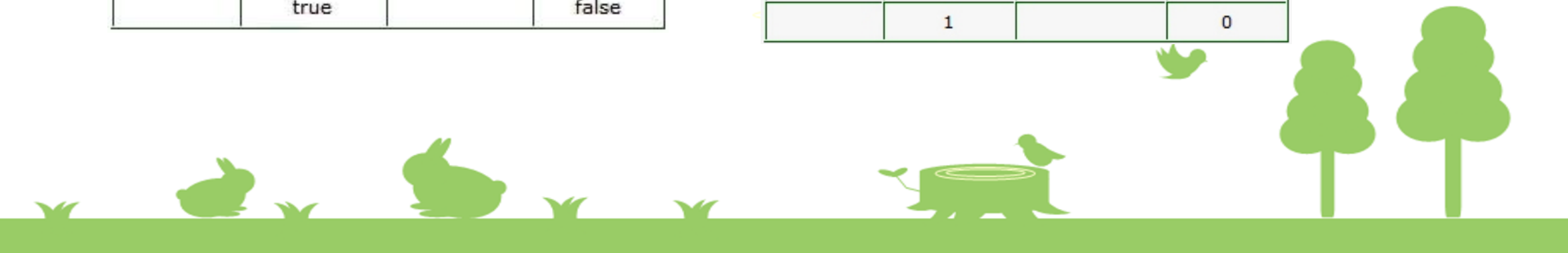
Operadores

Aritméticos		Lógicos		Bit a Bit		Asignación		Otros	
Cambio de signo	-	Negación	!	Negación	~	Asignación	=	borrar	delete
Incremento	++	Menor que	<	Rotación izquierda	<<	Con adición	+=	tipo	typeof
Decremento	--	Mayor que	>	Rotación derecha	>>	Con sustracción	-=	obviar valor	void
Multiplicación	*	Menor o igual que	<=	Rotación derecha sin signo	>>>	Con multiplicación	*=		
División	/	Mayor o igual que	>=	AND	&	Con división	/=		
Módulo	%	Igual a	==	XOR	^	Con módulo	%=		
Suma	+	Distinto a	!=	OR	|	Con op. bit a bit	<<=,>>=, etc.		
Resta	-	AND	&&						
		OR	||						
		Condicional	?:						
		Coma	,						
		Igualdad estricta	===						
		Desigualdad estricta	!==						

Operadores

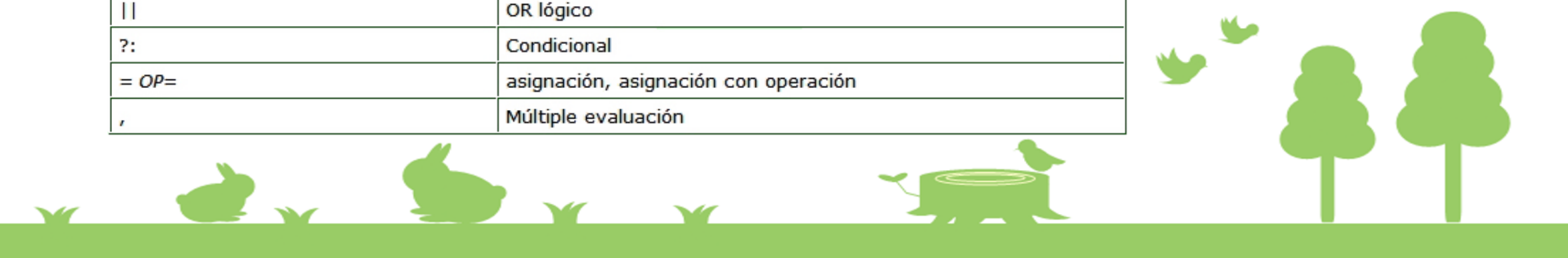
Operador	Operando1	Operando2	Resultado
AND (&&)	false	false	false
	false	true	false
	true	false	false
	true	true	true
OR (||)	false	false	false
	false	true	true
	true	false	true
	true	true	true
NOT (!)	false		true
	true		false

Operador	Primer_Bit	Segundo_Bit	Resultado
AND (&)	0	0	0
	1	0	0
	0	1	0
	1	1	1
OR (|)	0	0	0
	1	0	1
	0	1	1
	1	1	1
XOR (^)	0	0	0
	1	0	1
	0	1	1
	1	1	0
NOT (~)	0		1
	1		0



Operadores

Operador	Descripción
. [] ()	Acceso a miembro, índice de array, llamada a función
++ -- - ~ ! typeof new void delete	Operadores unarios, tipo de dato, constructor, valores indefinidos
* / %	Multiplicación, división, módulo
+ - +	Adición, sustracción, concatenación de cadenas
<< >> >>>	Desplazamiento de bits
< <= > >=	Menor que, menor o igual que, mayor que, mayor o igual que
== != === !==	Igualdad, desigualdad, igualdad estricta, desigualdad estricta
&	AND
^	XOR
|	OR
&&	AND lógico
||	OR lógico
?:	Condicional
= *OP*=	asignación, asignación con operación
,	Múltiple evaluación



Operadores

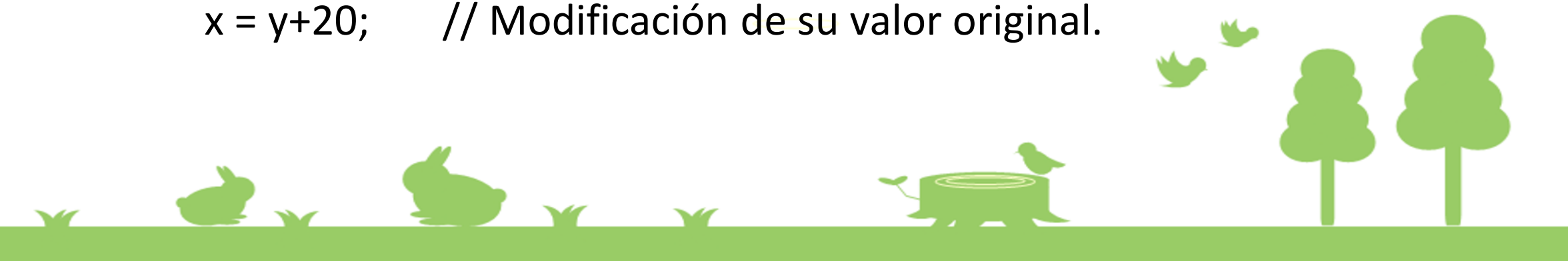
Operador	Ejemplo	Significado
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y
**=	x **= y	x = x ** y

Asignación

- La operación de asignación permite que las variables mutables puedan modificar su contenido en tiempo de ejecución.

```
let   x=10;       // Declaración e inicialización
var  y=20;

x = y+20;       // Modificación de su valor original.
```



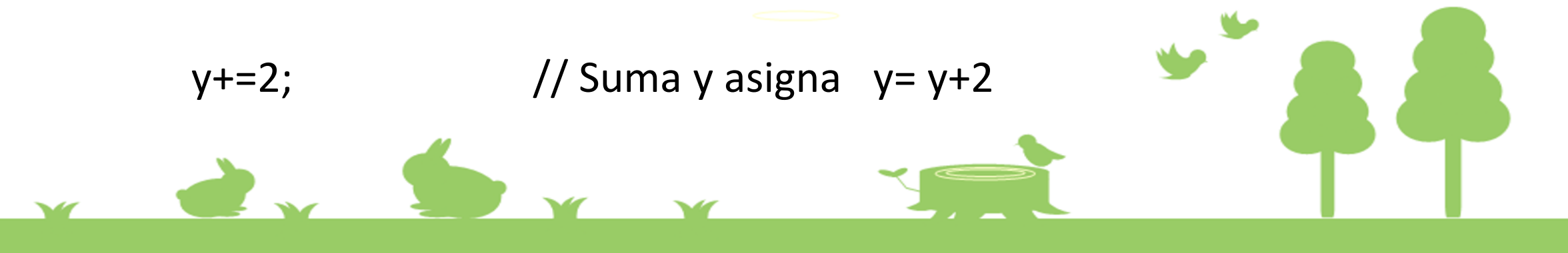
Asignación

- El operador = así como los operadores resumidos (operador=) cumplen esta acción.

```
let y;                    // Declaración de variable

y= 100;                   // Asigna valor directamente modificando su estado

y+=2;                     // Suma y asigna   y= y+2
```



Funciones

- A subrutinas o subprogramas en todo lenguaje son muy importantes, en Javascript lo es más, independiente de cual sea el tipo de aplicación se quiera desarrollar, a los subprogramas se les llama funciones.

```
function nombre([argumentos]){
    sentencia(s);
}
```

Funciones

- Ejemplos:

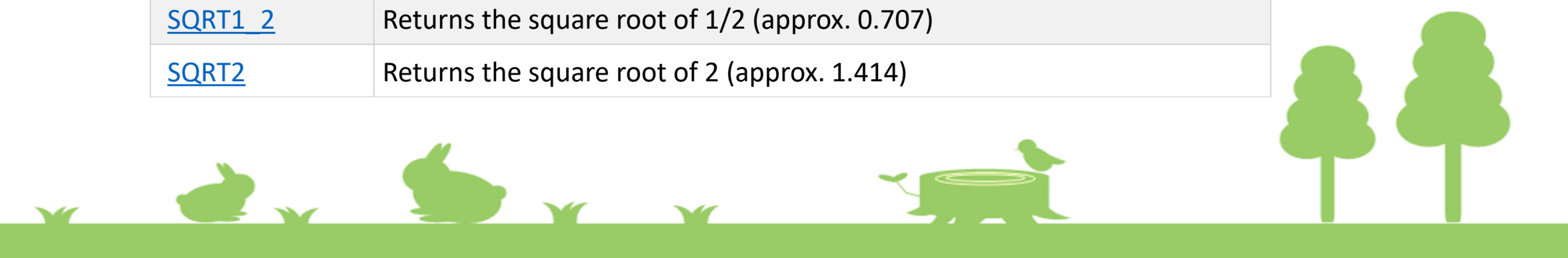
Ejemplos sin argumentos	Ejemplos con argumentos

function inicializa(){
i=j=v=0;
}
	function mensaje(texto){
	alert(texto);
	}

function generaAleatorio(){
return Math.floor(Math.random()*10);
}
	function cubo(x){
	return Math.pow(x,3);
	}

Funciones Math

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)



Funciones Math

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
acosh(x)	Returns the hyperbolic arccosine of x
asin(x)	Returns the arcsine of x, in radians
asinh(x)	Returns the hyperbolic arcsine of x
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
atanh(x)	Returns the hyperbolic arctangent of x
cbrt(x)	Returns the cubic root of x
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
cosh(x)	Returns the hyperbolic cosine of x
cos(x)	Returns the cosine of x (x is in radians)
cosh(x)	Returns the hyperbolic cosine of x

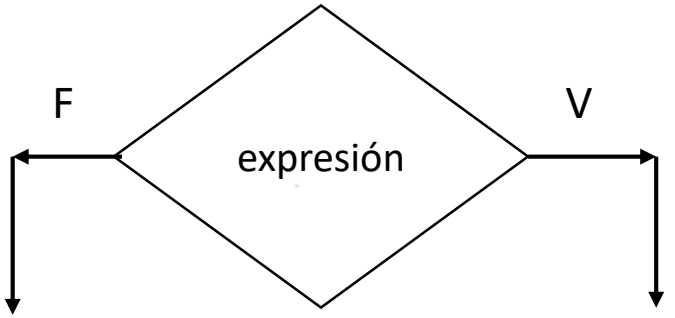
Funciones Math

Method	Description
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a number
trunc(x)	Returns the integer part of a number (x)

Selección

- La bifurcación o selección es un condicionante de ejecución de sentencias.

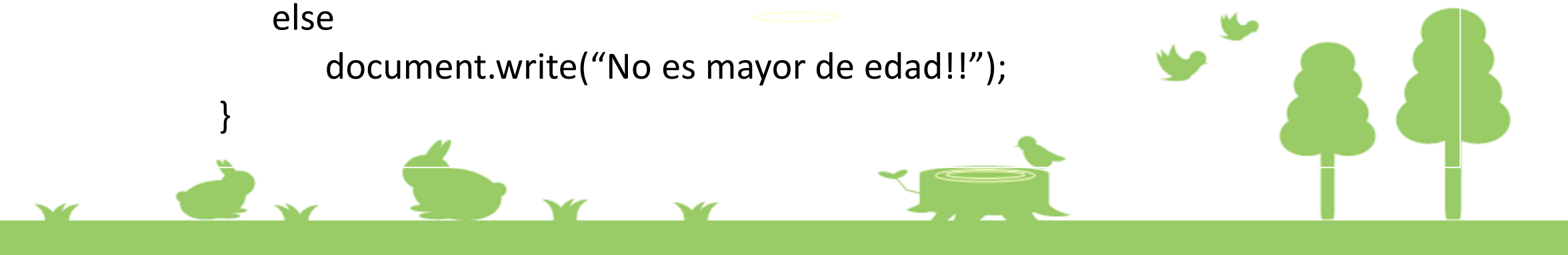
```
if (expresión)
        sentencia;
[else
      sentencia;
]
```



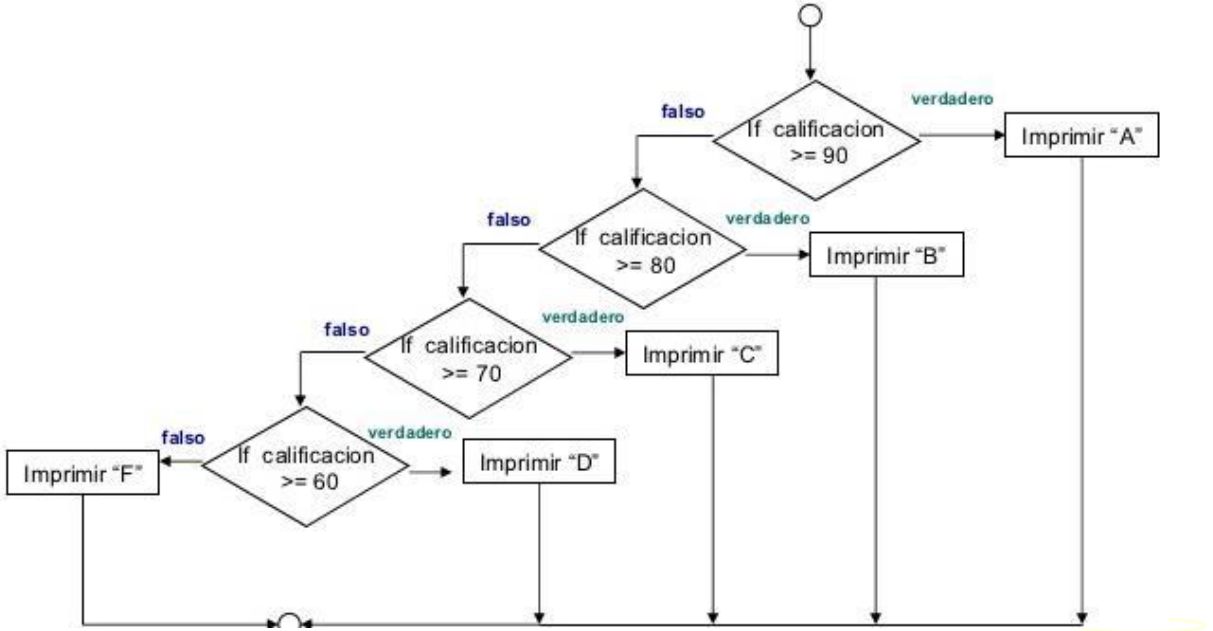
Selección

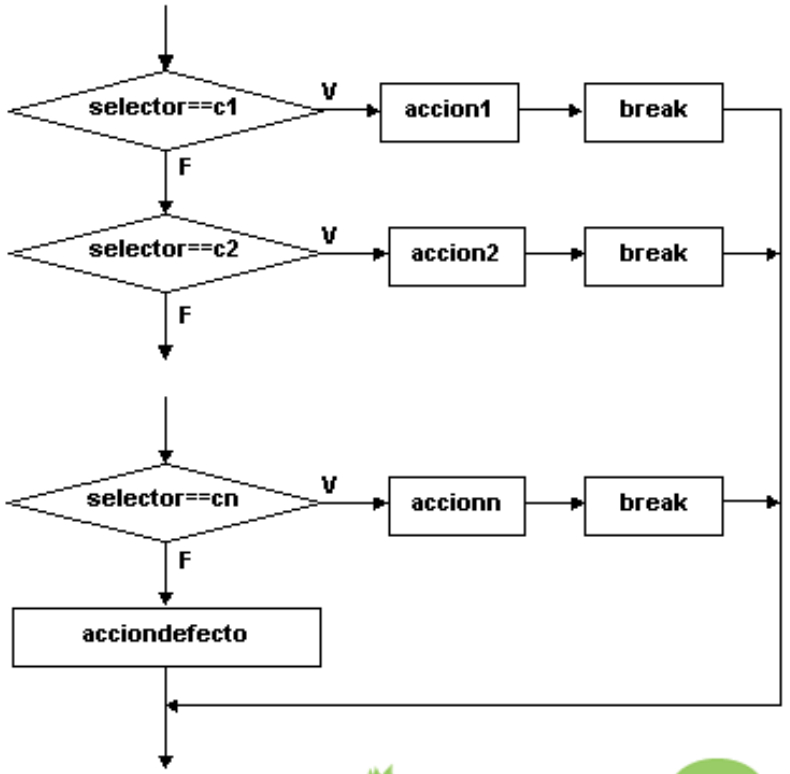
```
function esMayorEdad(edad){
        return edad>=18
}

function imprime(edad){
      if (esMayorEdad(edad))
            document.write(“Es mayor de edad!!”);
      else
            document.write(“No es mayor de edad!!”);
}
```



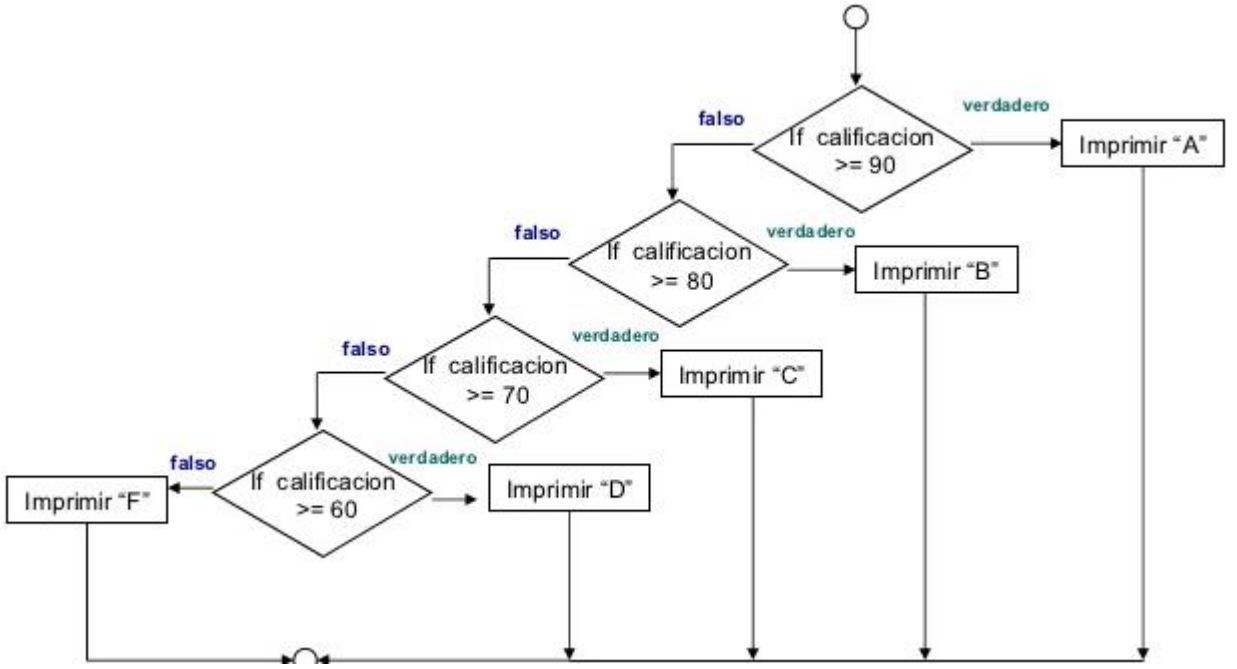
Selección múltiple





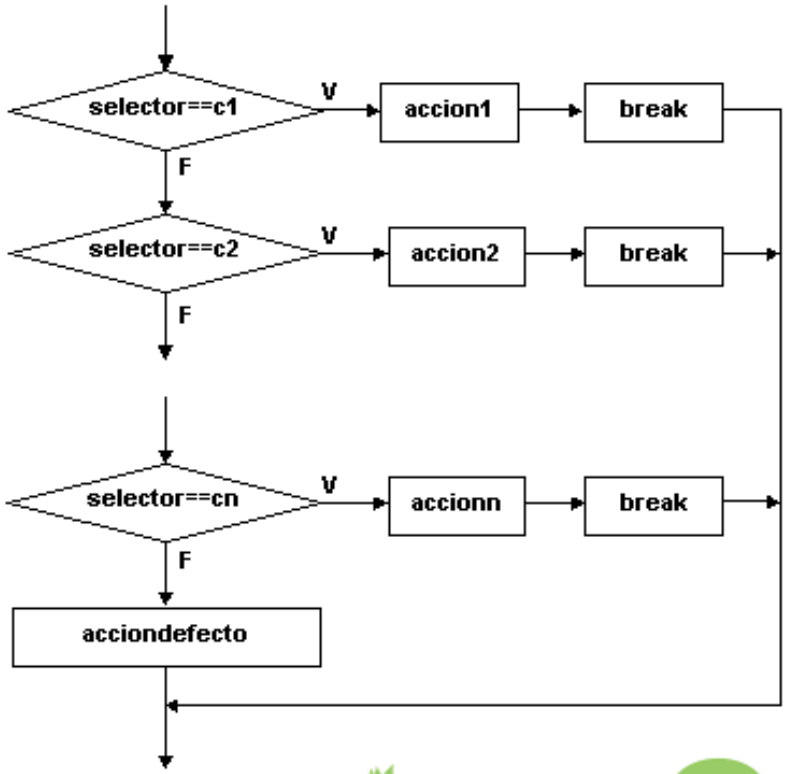
Selección múltiple

```
if(calificacion>=90)
    document.writeln("A");
else if(calificacion>=80)
    document.writeln("B");
else if(calificacion>=70)
    document.writeln("C");
else if(calificacion>=60)
    document.writeln("D");
else
    document.writeln("F");
```



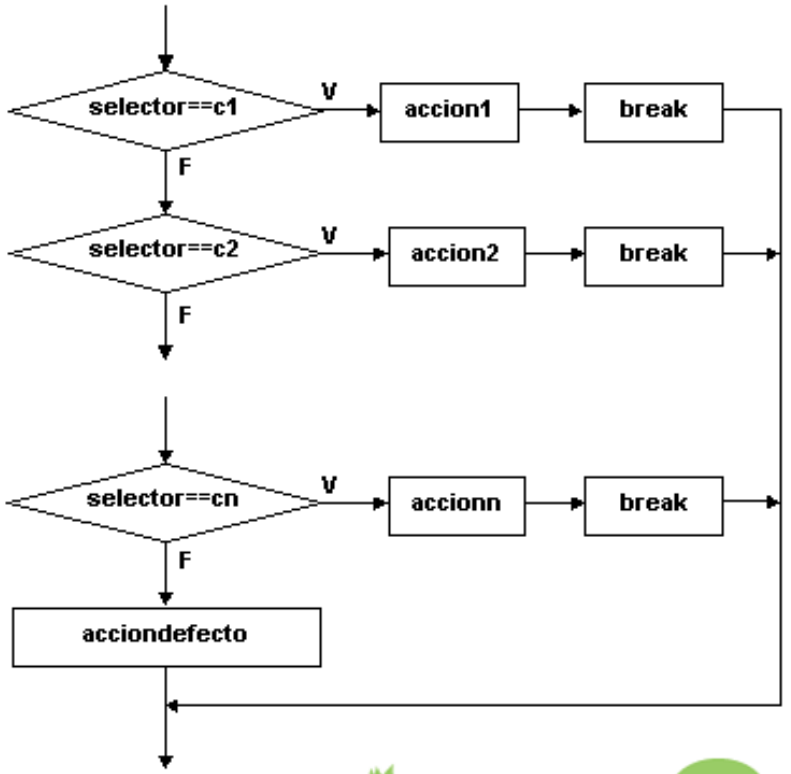
Selección múltiple

```
switch(expresion) {
  case x:
        // bloque de código
        break;
  case y:
        // bloque de código
        break;
  default:
        // bloque de código
}
```



Selección múltiple

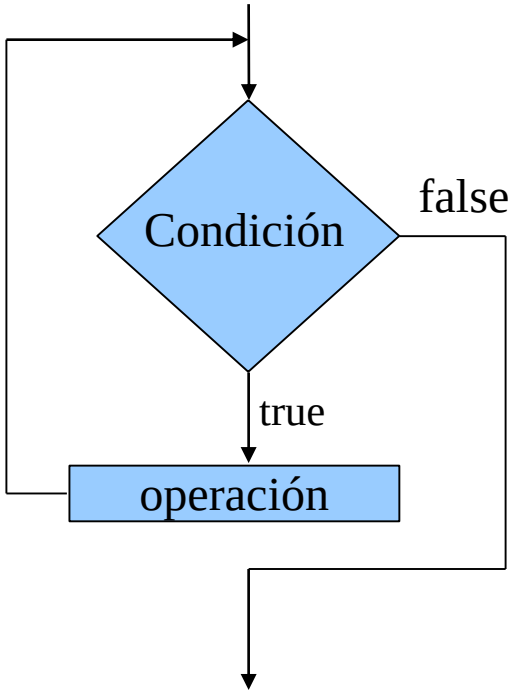
```
switch(selector%5) {
    case 0: document.writeln("Cero");
            break;
    case 1: document.writeln("Uno");
            break;
    default: document.writeln("Otro valor");
 }
```



Iteración

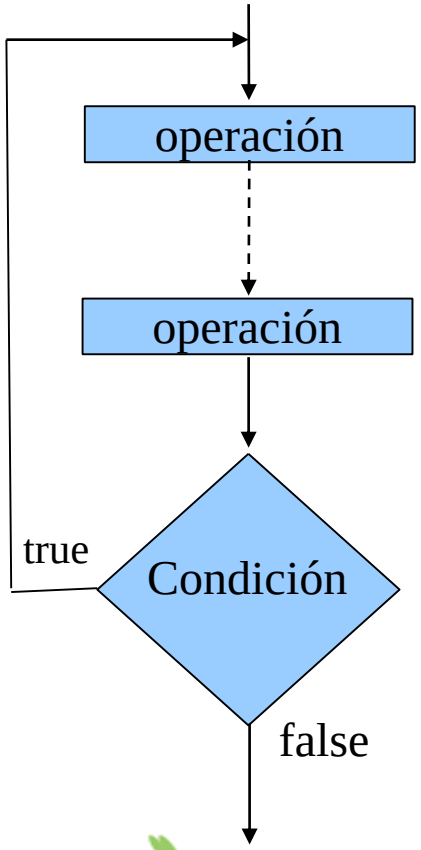
- Javascript como muchos lenguajes basados en sentencias, provee los mecanismos de iteración más conocidos:
  - Con condición al inicio (while)
  - Con condición al final (do)
  - Con contador (for)

Iteración con condición al inicio



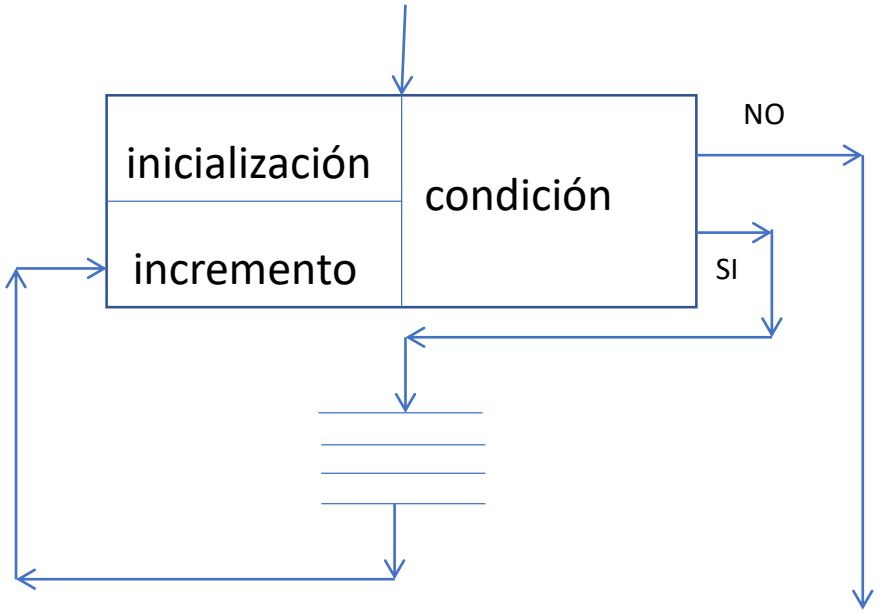

```
while(val<10){
        document.write("<img src='../image/im.png'/>");
        val+=1;
 }
```

Iteración con condición al final




```
do{
        document.write("<img src='../image/im.png'/>");
        val+=1;
 }while(val<10);
```

Iteración con contador




```
for(i=1;i<=5;i++)
        sentencia;
```

```
for(i=1;i<=5;i++){
    sentencias;
}
```

Break y continua

- La sentencia ***break***, permite en el switch, evitar la ejecución de sentencias de más de un caso.
- En el caso de las sentencias, while, for y do, su uso es evitar que las sentencias dentro del ciclo se ejecuten mas. Se dice que rompe el ciclo de forma abrupta.
- Por otro lado ***continue*** es una sentencia que fuerza a que se ejecute en un ciclo while, for o do, la condición que hace que el ciclo se ejecute.

Arreglos

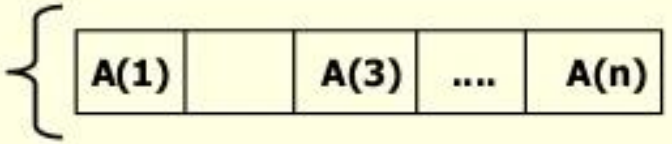
- Un arreglo es un conjunto de elementos almacenados continaumente representados mediante una variable.
- Un arreglo puede ser de una o más dimensiones.

A(1)		A(3)	....	A(n)

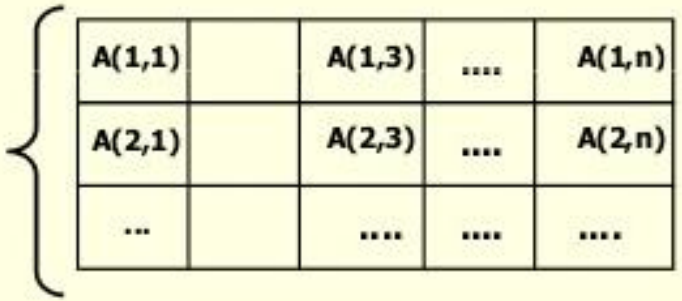
A(1,1)		A(1,3)	....	A(1,n)
A(2,1)		A(2,3)	....	A(2,n)
...		....	....	....

Arreglos

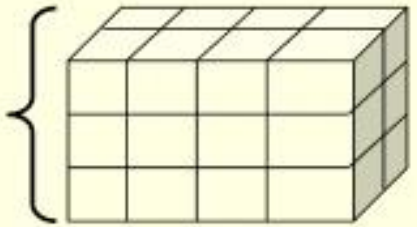
- Los arreglos de una dimensión se les llama vector o lista.



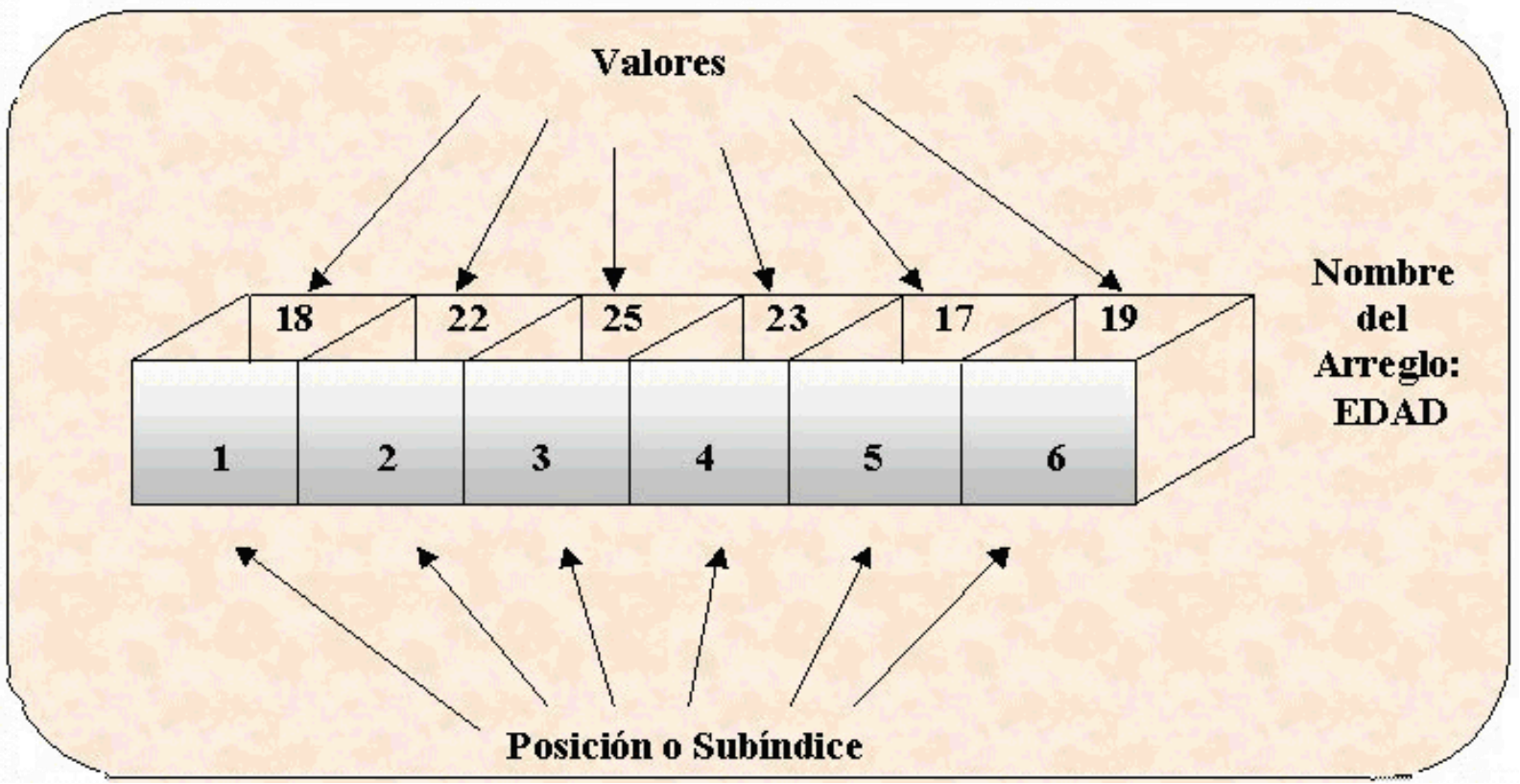
- Los arreglos de dos dimensiones se les conoce como matriz o tabla (Anxn).



- Los demás se les conoce como arreglos multidimensionales.



Arreglos

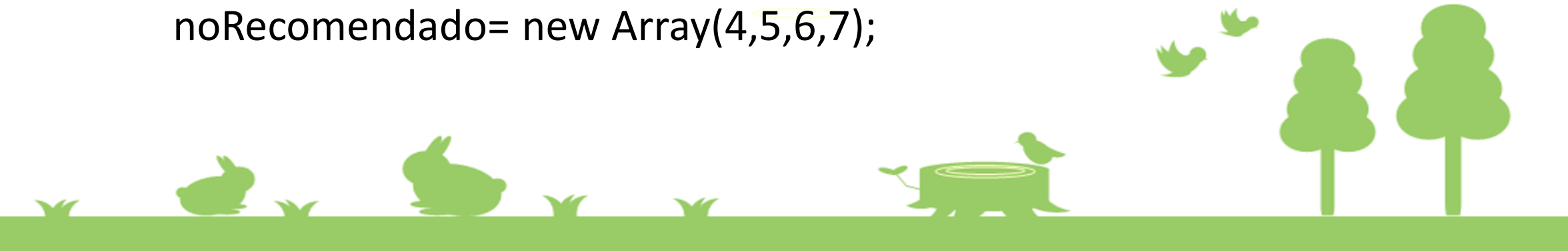

Valores
18
22
25
23
17
19
1
2
3
4
5
6
Nombre del Arreglo: EDAD
Posición o Subíndice

Arreglos

- En Javascript se declaran de la siguiente forma

```
a=[3,4,5,6,7];
v=[“Ana”,”Luis”,”Mario”];
t=[[3,4,5],[3,5,6],[9,2,3]];

noRecomendado= new Array(4,5,6,7);
```

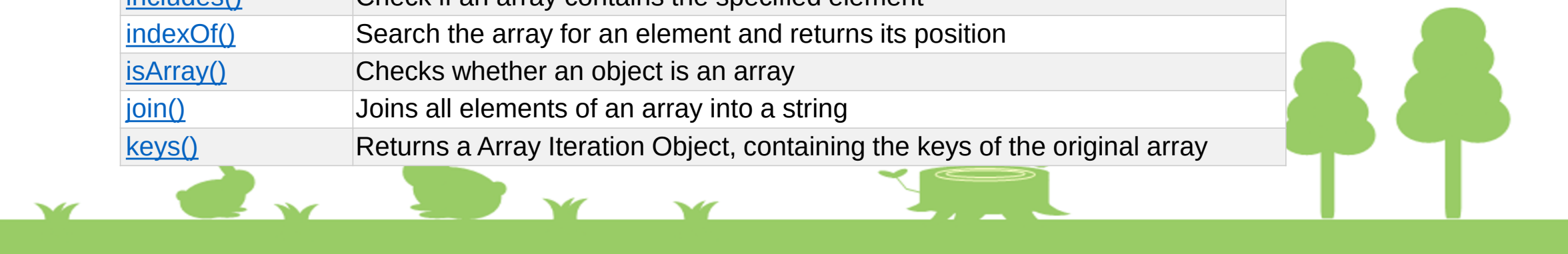


Arreglos

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an Array object

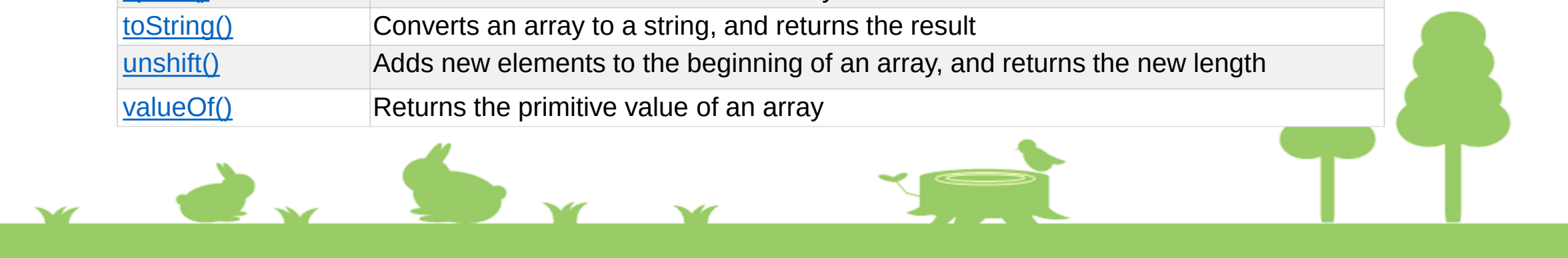
Arreglos

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if all element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element
indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array
join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array



Arreglos

Method	Description
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
some()	Checks if any of the elements in an array pass a test
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array



Cadenas (String)

- En Javascript se declaran de la siguiente forma

```
a="hola"
v=new String("hola");
```

typeof(a) ➔ String

typeof(v) ➔ Object

Cadenas (String)

Propiedades	Descripción
constructor	Returns the string's constructor function
length	Returns the length of a string
prototype	Allows you to add properties and methods to an object

Cadenas (String)

Metodo	Descripción
charAt()	Returns the character at the specified index (position)
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a new joined strings
endsWith()	Checks whether a string ends with specified string/characters
fromCharCode()	Converts Unicode values to characters
includes()	Checks whether a string contains the specified string/characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
localeCompare()	Compares two strings in the current locale
match()	Searches a string for a match against a regular expression, and returns the matches
repeat()	Returns a new string with a specified number of copies of an existing string
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced
search()	Searches a string for a specified value, or regular expression, and returns the position of the match

Cadenas (String)

Metodo	Descripción
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
startsWith()	Checks whether a string begins with specified characters
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLocaleLowerCase()	Converts a string to lowercase letters, according to the host's locale
toLocaleUpperCase()	Converts a string to uppercase letters, according to the host's locale
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string
valueOf()	Returns the primitive value of a String object

Objetos

- Es una variable en Javascript que tiene más de un valor y estos pueden ser de distintos tipos.

```
let materia={
        id: 1,
        nombre:”Química”,
        creditos: 5
};
```

Objetos

- Para acceder a las propiedades se realiza utilizando el formato de arreglo o de campos.

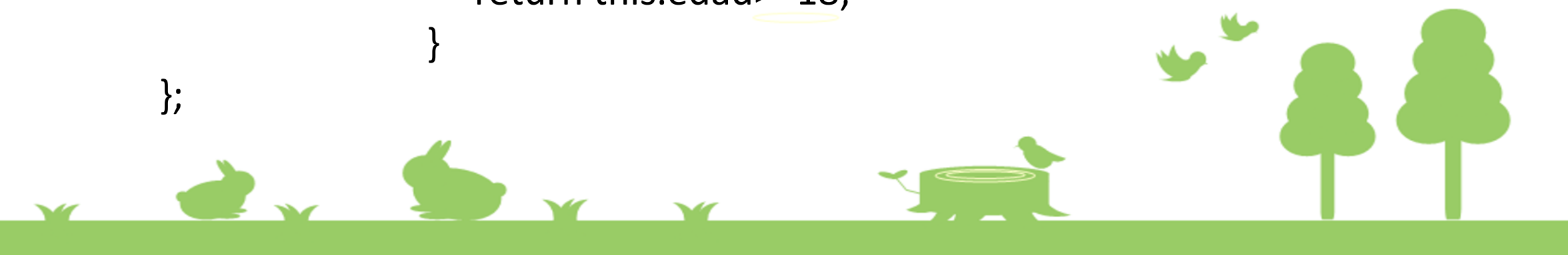
materia[“id”] // Accediendo a propiedad en formato de arreglo

materia.id // Accediendo a propiedad en formato campo

Objetos

- Los objetos pueden tener propiedades y métodos.

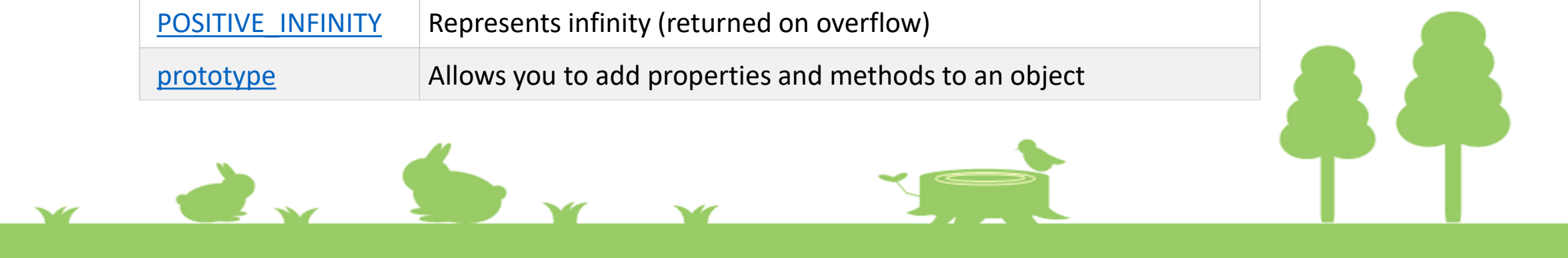
```
var persona={
        id: 34,
        nombre:”Pedro Pérez”,
        edad:25,
        esMayor:function(){
                return this.edad>=18;
            }
};
```



Clase Number

- Es una clase que define propiedades y métodos

Property	Description
constructor	Returns the function that created JavaScript's Number prototype
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)
prototype	Allows you to add properties and methods to an object



Clase Number

Method	Description
isFinite()	Checks whether a value is a finite number
isInteger()	Checks whether a value is an integer
isNaN()	Checks whether a value is Number.NaN
isSafeInteger()	Checks whether a value is a safe integer
toExponential(x)	Converts a number into an exponential notation
toFixed(x)	Formats a number with x numbers of digits after the decimal point
toLocaleString()	Converts a number into a string, based on the locale settings
toPrecision(x)	Formats a number to x length
toString()	Converts a number to a string
valueOf()	Returns the primitive value of a number

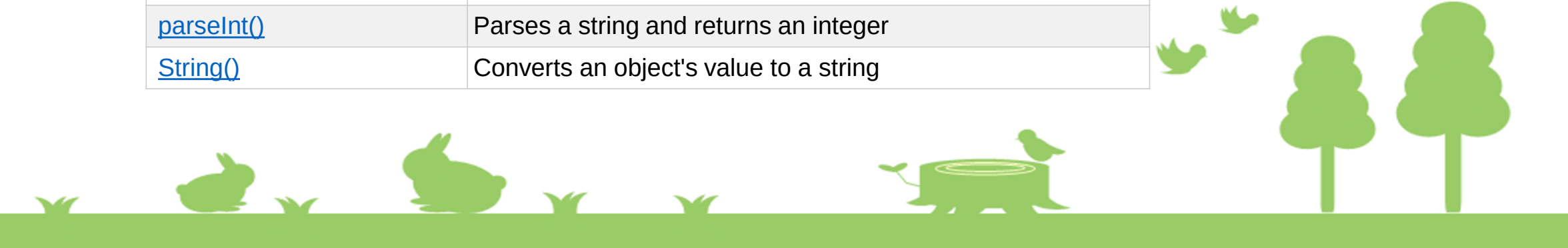
Propiedades y métodos globales

- Estas propiedades y métodos están disponibles por defecto

Property	Description
Infinity	A numeric value that represents positive/negative infinity
NaN	"Not-a-Number" value
undefined	Indicates that a variable has not been assigned a value

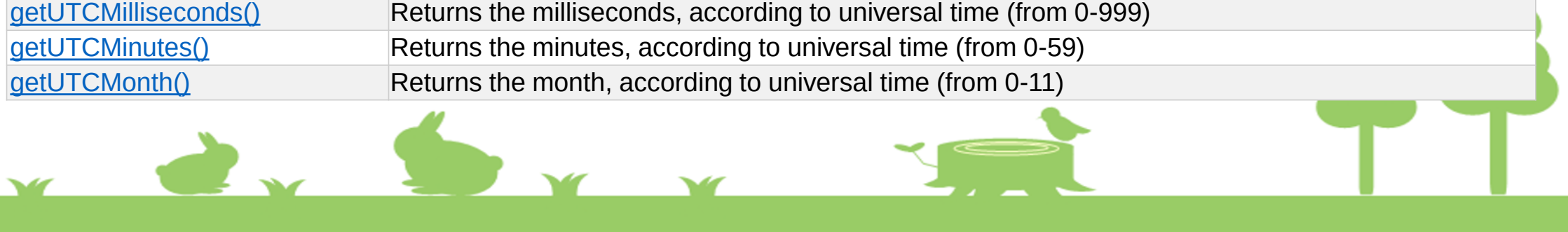
Propiedades y métodos globales

Function	Description
decodeURI()	Decodes a URI
decodeURIComponent()	Decodes a URI component
encodeURI()	Encodes a URI
encodeURIComponent()	Encodes a URI component
eval()	Evaluates a string and executes it as if it was script code
isFinite()	Determines whether a value is a finite, legal number
isNaN()	Determines whether a value is an illegal number
Number()	Converts an object's value to a number
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer
String()	Converts an object's value to a string



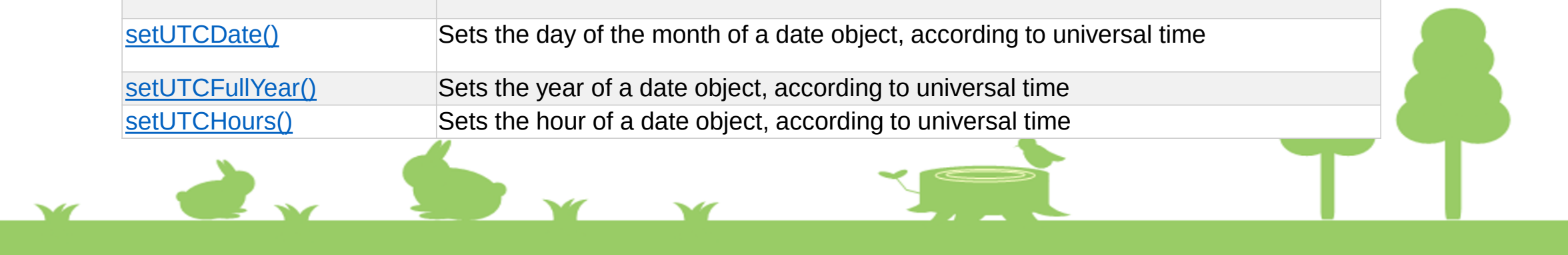
Clase Date

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes
getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)



Clase Date

Method	Description
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)
getYear()	Deprecated. Use the getFullYear() method instead
now()	Returns the number of milliseconds since midnight Jan 1, 1970
parse()	Parses a date string and returns the number of milliseconds since January 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date to a specified number of milliseconds after/before January 1, 1970
setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time
setUTCHours()	Sets the hour of a date object, according to universal time



Clase Date

Method	Description
setUTCHours()	Sets the hour of a date object, according to universal time
setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
setYear()	Deprecated. Use the setFullYear() method instead
toDateString()	Converts the date portion of a Date object into a readable string
toGMTString()	Deprecated. Use the toUTCString() method instead
toISOString()	Returns the date as a string, using the ISO standard
toJSON()	Returns the date as a string, formatted as a JSON date
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date since midnight of January 1, 1970, according to UTC time
valueOf()	Returns the primitive value of a Date object