ParqueSoft.TI®

Talent Training & Improvement

TIPOS DE DATOS Y OPERADORES

Tutora: Diana García

www.parquesoft.com/ti

dianagarco@gmail.com

VIDEO

TIPOS DE DATOS

TIPOS DE DATOS

Variable	Explicación	Ejemplo
String	Esto es una secuencia de texo conocida como cadena. Para indicar que la variable es una cadena, debes escribirlo entre comillas.	`let miVariable = 'Bob';`
Number	Esto es un número. Los números no tienen comillas.	`let miVariable = 10;`
Boolean	Tienen valor verdadero/falso. `true` / `false` son palabras especiales en JS, y no necesitan comillas.	`let miVariable = true;`
Array	Una estructura que te permite almacenar varios valores en una sola referencia.	`let miVariable = [1,'Bob','Steve',10];` Llama a cada miembro del array así: `miVariable[0]`, `miVariable[1]`, etc.
Object	Básicamente cualquier cosa. Todo en JavaScript es un objeto y puede ser almacenado en una variable. Mantén esto en mente mientras aprendes.	`let miVariable = document.querySelector('h1');` Todos los ejemplos anteriores también.

TIPOS DE DATOS

JavaScript es un lenguaje ***débilmente tipado y dinámico.***

Las variables en JavaScript no están asociadas directamente con ningún tipo de valor en particular, y a cualquier variable se le puede asignar (y reasignar) valores de todos los tipos:

```
let foo = 42;    // foo ahora es un número
foo     = 'bar'; // foo ahora es un string
foo     = true;  // foo ahora es un booleano
```

TIPOS DE DATOS

Variable	Explicación	Ejemplo
String	Esto es una secuencia de texto conocida como cadena. Para indicar que la variable es una cadena, debes escribirlo entre comillas.	let miVariable = 'Bob';
Number	Esto es un número. Los números no tienen comillas.	let miVariable = 10;
Boolean	Tienen valor verdadero/falso. true/false son palabras especiales en JS, y no necesitan comillas.	let miVariable = true;
Array	Una estructura que te permite almacenar varios valores en una sola referencia.	let miVariable = [1,'Bob','Steve',10];
		Llama a cada miembro del array así: miVariable[0], miVariable[1], etc.
Object	Básicamente cualquier cosa. Todo en JavaScript es un objeto y puede ser almacenado en una variable. Mantén esto en mente mientras aprendes.	let miVariable = document.querySelector('h1');
		Todos los ejemplos anteriores también.

Un objeto JavaScript es una asociación entre claves y valores.

STRING

JavaScript automáticamente convierte las primitivas en objetos String, por lo que es posible utilizar métodos del objeto String en cadenas primitivas.

```
let string1 = "Una cadena primitiva";
let string2 = 'También una cadena primitiva';
let string3 = `Otra cadena primitiva más`;
let string4 = new String("Un objeto String");


let s_prim = 'foo'
let s_obj = new String(s_prim)

console.log(typeof s_prim) // Registra "string"
console.log(typeof s_obj)  // Registra "object"

let longString = "Esta es una cadena muy larga que necesita " +
                 "que dividimos en varias líneas porque " +
                 "de lo contrario, mi código es ilegible."

String('4');
```

NUMBER

```
new Number(value);
var a = new Number('123'); // a === 123 es false
var b = Number('123'); // b === 123 es true
a instanceof Number; // es true
b instanceof Number; // es false

Number('123')  // retorna el número 123
Number('123') === 123  // retorna true
Number("unicorn")  // NaN
Number(undefined)  // NaN

const MásgrandeNum    = Number.MAX_VALUE ;
const MáspequeNum     = Number.MIN_VALUE ;
const infinitoNum     = Number.POSITIVE_INFINITY ;
const notInfinitoNum  = Number.NEGATIVE_INFINITY ;
const noEsNum         = Number.NaN ;

// Convierte cadenas numéricas a números
Number('123')     // 123
Number('12.3')    // 12.3
Number('123e-1')  // 12.3
Number('')        // 0
Number('0x11')    // 17
Number('0b11')    // 3
Number('0o11')    // 9
Number('foo')     // NaN
Number('100a')    // NaN
```

BOOLEANO

```
var x = false;
if (x) {
 // este código no se ejecuta
}

var x = new Boolean(false);
if (x) {
 // este código se ejecuta
}

// Creación de objetos Boolean con un valor inicial de false
var bNoParam = new Boolean();
var bZero = new Boolean(0);
var bNull = new Boolean(null);
var bEmptyString = new Boolean('');
var bfalse = new Boolean(false);

// Creación de objetos Boolean con un valor inicial de true
var btrue = new Boolean(true);
var btrueString = new Boolean('true');
var bfalseString = new Boolean('false');
var bArrayProto = new Boolean([]);
var bObjProto = new Boolean({});
```

Cualquier objeto cuyo valor no sea undefined o null, incluido un objeto Boolean cuyo valor es false, se evalúa como true cuando se pasa a una declaración condicional.

ARREGLOS

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];
shopping;

let sequence = [1, 1, 2, 3, 5, 8, 13];
let random = ['tree', 795, [0, 1, 2]];

shopping[0];
// returns "bread"

shopping[0] = 'tahini';
shopping;
// shopping => [ "tahini", "milk", "cheese", "hummus", "noodles" ]

// Tamaño de un arreglo
sequence.length;  // retorna 7

let sequence = [1, 1, 2, 3, 5, 8, 13];
for (var i = 0; i < sequence.length; i++) {
 console.log(sequence[i]) ;
} // imprime cada uno de los elementos
```

ARREGLOS

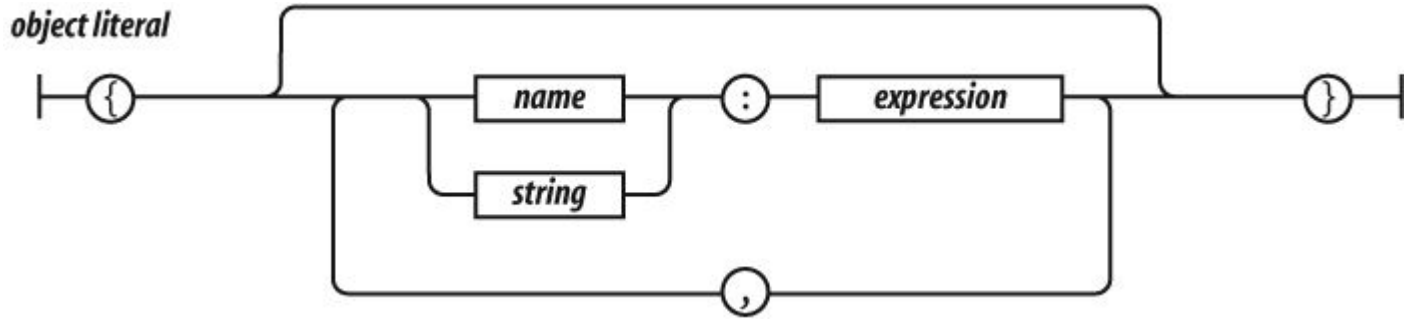
```
//De cadenas a arreglos
let myData = 'Manchester/London,Liverpool/Birmingham,Leeds/Carlisle';
let myArray = myData.split('/');
myArray;

myArray.length;
myArray[0]; // the first item in the array
myArray[1]; // the second item in the array
myArray[myArray.length-1]; // the last item in the array

// De arreglos a cadenas
let dogNames = ['Rocket','Flash','Bella','Slugger'];
dogNames.toString(); //Rocket,Flash,Bella,Slugger

// Con diferentes separadores
let myNewString = myArray.join(',');
myNewString;
```

OBJETOS




```
let dog = { name : 'Spot', breed : 'Dalmatian' };
dog.name

let obj = new Object();
obj.foo = "bar";
obj.hello = function() { console.log("Hello world!"); }

let obj = {
   "foo"   : "bar",
   "hello" : function() { console.log("Hello world!"); }
};
```

OBJETOS

```
let flight = {
    airline: "Oceanic",
    number: 815,
    departure: {
        IATA: "SYD",
        time: "2004-09-22 14:55",
        city: "Sydney"
    },
    arrival: {
        IATA: "LAX",
        time: "2004-09-23 10:42",
        city: "Los Angeles"
    }
};

// modificando un objeto
flight["airline"] = "LATAM";
flight.number = 3;
```

OPERADORES DE ASIGNACIÓN - PT1

Nombre	Operador abreviado	Significado
Asignación	`x = y`	`x = y`
Asignación de adición	`x += y`	`x = x + y`
Asignación de resta	`x -= y`	`x = x - y`
Asignación de multiplicación	`x *= y`	`x = x * y`
Asignación de división	`x /= y`	`x = x / y`
Asignación de residuo	`x %= y`	`x = x % y`
Asignación de exponenciación	`x **= y`	`x = x ** y`
Asignación de desplazamiento a la izquierda	`x <<= y`	`x = x << y`

OPERADORES DE ASIGNACIÓN - PT2

Asignación de desplazamiento a la derecha	`x >>= y`	`x = x >> y`
Asignación de desplazamiento a la derecha sin signo	`x >>>= y`	`x = x >>> y`
Asignación AND bit a bit	`x &= y`	`x = x & y`
Asignación XOR bit a bit	`x ^= y`	`x = x ^ y`
Asignación OR bit a bit	`x |= y`	`x = x | y`
Asignación AND lógico	`x &&= y`	`x && (x = y)`
Asignación OR lógico	`x ||= y`	`x || (x = y)`
Asignación de anulación lógica	`x ??= y`	`x ?? (x = y)`

OPERADORES DE ASIGNACIÓN

```
var numero1 = 5;
var numero2 = 2;
numero3 = numero1++ + numero2;
// numero3 = 7, numero1 = 6

var numero1 = 5;
var numero2 = 2;
numero3 = ++numero1 + numero2;
// numero3 = 8, numero1 = 6

var valor1 = true;
var valor2 = false;
resultado = valor1 && valor2; // resultado = false

valor1 = true;
valor2 = true;
resultado = valor1 && valor2; // resultado = true
```

OPERADORES DE COMPARACIÓN - PT1

Operador	Descripción	Ejemplos que devuelven `true`

`Igual` (==)	Devuelve `true` si los operandos son iguales.	`3 == var1`
		`"3" == var1`
		
		`3 == '3'`

`No es igual` (!=)	Devuelve `true` si los operandos *no* son iguales.	`var1 != 4` `var2 != "3"`
`Estrictamente igual` (===)	Devuelve `true` si los operandos son iguales y del mismo tipo. Consulta también `Object.is` y `similitud en JS`.	`3 === var1`
`Desigualdad estricta` (!==)	Devuelve `true` si los operandos son del mismo tipo pero no iguales, o son de diferente tipo.	`var1 !== "3"` `3 !== '3'`

OPERADORES DE COMPARACIÓN - PT1

Mayor que (>)	Devuelve true si el operando izquierdo es mayor que el operando derecho.	var2 > var1 "12" > 2
Mayor o igual que (>=)	Devuelve true si el operando izquierdo es mayor o igual que el operando derecho.	var2 >= var1 var1 >= 3
Menor que (<)	Devuelve true si el operando izquierdo es menor que el operando derecho.	var1 < var2 "2" < 12
Menor o igual (<=)	Devuelve true si el operando izquierdo es menor o igual que el operando derecho.	var1 <= var2 var2 <= 5

OPERADORES DE COMPARACIÓN

```
3 === 3   // true
3 === '3' // false
var object1 = {'key': 'value'}, object2 = {'key': 'value'};
object1 === object2 // false

1    ==  1          // true
'1'  ==  1          // true
1    == '1'         // true
0    == false       // true
0    == null        // false
var object1 = {'key': 'value'}, object2 = {'key': 'value'};
object1 == object2 // false
0    == undefined  // false
null == undefined  // true

4 >= 3 // true
3 >= 3 // true
```

ParqueSoft.TI®
Talent Training & Improvement

FORMACIÓN EFECTIVA PARA LA DEMANDA LABORAL ACTUAL!

www.parquesoft.com/ti

ParqueSoft.TI®
Talent Training & Improvement