ArgentinaPrograma
YoProgramo

JS parte 2

por Leonardo Blautzik, Federico Gasior y Lucas Videla

Julio / Diciembre 2021

Ministerio de
Desarrollo Productivo
Argentina

Variables

Las variables son contenedores de valores

En JS estos valores puede ser desde un simple número, hasta cosas complejas como un objeto o incluso una función

Las variables son declaradas con **let** o **var**. Su diferencia es el contexto con el cual trabajan. Las declaradas con **let** tienen contextos muy similares a los de Java

Sintaxis

```
let profesor = "Fede";

variableNoDeclarada = 10;

console.log(profesor); // "Fede"
console.log(variableNoDeclarada); // 10??
console.log(variableNoExistente); // undefined
```

Contexto de var

```
let cosas = [1, 2, 3, 4, 5];
for (var i = 0; i < cosas.length; i++) {
  // ...
}

console.log(i); // 5
```

Tipos de datos

- **number:** 100, 5.5, NaN
- **string:** “”,”hola”, ‘hola’,
- **boolean:** true, false
- **object:** {} (new Object), new Date(), [] (new Array), {key: value, key2: value2}
- **function:** function () {/* algo */}
- **undefined:** undefined
- **symbol (2015):** Symbol(‘simbolo’)
- **bigint (2020):** 81n

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof

Objetos

Los objetos son estructuras del estilo clave-valor que pueden contener como valor cualquier tipo de dato, incluso otro objeto. Cada clave es única. Similar a la estructura Map de Java

```
let objeto = {
  clave1: 'texto',
  clave2: {
    clave1objeto: 10
  }
}
```

Objetos globales

En JS siempre vamos a contar con un objeto global, el cual no solo contiene todas las funciones y valores por defecto sino que también va a contener todos los valores y funciones que nosotros creemos.

Depende del contexto, la misma puede ser “window” normalmente en los navegadores, “global” en Node.js, entre otros

Un valor global se define agregandoselo al objeto global

Tipado dinámico

JS es un lenguaje de tipado dinámico, por lo tanto no es necesario especificar el tipo de dato del contenedor, ni tampoco mantenerlo en el tiempo.

```
let numero = "123.45";
typeof numero === 'string';
numero = +numero;
typeof numero === 'number';
```

Constantes

En JS podemos declarar un valor al cual no se le permite ser cambiado

```
const GRAVITATIONAL_CONSTANT = 6.674e-11;
const title = document.querySelector("h1");
```

El contexto de las constantes es igual que el de las variables declaradas con **let**

Operadores de comparación

Además de contar con los comparadores clásicos “<”, “>”, “<=” y “>=”, JS cuenta con la manera estricta y no estricta de comparar por igualdad o no-igualdad. La no estricta es con los símbolos “==” y “!=”, y su uso es desaconsejado, ya que pueden ocurrir cosas como:

```
2 == "2"
1 == true
2 == false
!!2 == true
```

Comparación estricta

Siempre debemos tener cuidado con los tipos de datos, y además siempre utilizar las comparaciones de tipo estrictas, las cuales primero comprueban el tipo de dato de ambas partes.

- Para igualdad: “===”
- Para no igualdad: “!==”

Operador +

El operador + puede utilizarse tanto para suma como para concatenación de cadenas. El problema se da cuando los tipos de datos de la operación no son los adecuados (es decir, o números o cadenas)

La mejor recomendación, es siempre hacer casteos explícitos antes de operar

```
// Para convertir una cadena a número
let numero = Number("123.45")

// Para convertir un número a cadena
let cadena = (123.45).toString();
```

Ejemplos básicos con cadenas

```
let c = "¡Utilizando cadenas en JS!";

c.length; // Largo de la cadena
c[4]; // Obtener el 5to caracter como string
c.indexOf("JS"); // Obtiene la posición de comienzo de "JS"
                 //-1 si no la encuentra
c.slice(1, c.length - 1); // Extraemos los signos de exclamación
c.replace("cadena", "string"); // Devolvemos una nueva cadena
```

Más información: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

Ejemplos básicos con arrays

```
let a = [1, "hola", ["1a", "2a"], null, 5];

a.length; // Largo del array -> 5
a[1]; // Obtener el segundo elemento del array
a.slice(0, 2); // Obtener los elementos del primero al tercero
a.indexOf("hola"); // Obtiene la posición de "hola" o -1
a.pop(); // Extrae el último elemento del array, modificandolo
a.push("5"); // Agrega el elemento al final del array
```

Más información: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Bloques en JS

No vamos a entrar en detalles en esta parte ya que el armado de los bloques condicionales y ciclos es muy similar al de Java

Funciones

Las funciones son lo que en Java hemos llamado métodos. Es aquello que nos permite guardar código para invocarlo cuantas veces queramos con un simple comando.

Aunque en JS también tenemos métodos además de funciones. O no. O.. quizás

Para simplificar, nosotros vamos a llamar a todo función, salvo que esté específicamente dentro de una clase en JS

Funciones

Ya hemos utilizado funciones, pero veremos como declararlas y que significa utilizarlas

```
function funcion() { // Creación
  // ...
}
funcion(); // Invocación

(function() { // Creación de función anónima
  //...
})(); // Ejecución de la misma
```

Funciones

Las funciones pueden tener parámetros, de los cuales no se conoce su tipo, ni tampoco siquiera si están definidos

```
function mayor(a, b) {
  return a > b;
}

mayor(1, 4);
mayor(4);

function mayor(a) {
  return a;
}
```

Funciones

Parámetros por defecto y validaciones de tipo

```
function potenciaValor(x = 0) {
  return typeof x !== "number" ? NaN : x ** x;
}
```

Eventos

Acciones que ocurren en el entorno del sistema, el cual el sistema te indica para que puedas ejecutar acciones si así lo deseas

Hay una cantidad gigantesca de eventos que pueden ocurrir y a los cuales se les puede agregar acciones

Los eventos no forman parte del núcleo de JS, sino que son su elemento aparte, como la manipulación del DOM

Ejemplo

Prueben el siguiente ejemplo con distintos tipos de eventos como focus, blur, dblclick, mouseover, mouseout, keydown, keyup, input

```
<input id="campoTexto" />

// document.createElement
function funcion(e) { console.log(e); }
const campoTexto = document.querySelector("campoTexto");
// campoTexto.onclick = funcion; // Solo 1 a la vez
campoTexto.addEventListener("click", funcion); // O anónima
campoTexto.removeEventListener("click", funcion);
```

El objeto evento y sus métodos más útiles

En el ejemplo anterior, la variable “e” era un objeto de evento, el cual nos brinda muchísima información acerca del evento al cual estamos escuchando. Investiguen al mismo, y sobre todo, investiguen para que sirven los siguientes métodos, son muy útiles en ciertas circunstancias

```
event.preventDefault();
event.stopPropagation();
```

Objetos

Son contenedores de información relacionada

```
const estudiante1 = {
  nombre: "Franco",
  apellido: "Vazquez",
  cursosEnProgreso: ["programación 1", "ingles 1"],
  saludar: function() {
    let cursando = this.cursosEnProgreso.length ?
      "estoy cursando " + this.cursosEnProgreso.join(", ") :
      "no estoy cursando nada";
    alert("¡Buenas! Mi nombre es " + this.apellido + ", " +
      this.nombre + " y actualmente " + cursando);
  }
}
```

Objetos

Prueben modificar las distintas propiedades en el ejemplo anterior para ver que resultados se pueden lograr. Todo es dinámico. Se puede agregar una nueva función al objeto si es deseado también

```
estudiante1.cursosEnProgreso = [];
estudiante1.saludar();
```

Notación

Podemos acceder a las propiedades y métodos usando dos notaciones distintas. La del punto “.” y la de los corchetes “[]”

```
document.querySelector("query")
document["querySelector"]("query");

document[funcion]();
```

Orientado a objetos y prototipos

JS puede escribirse a través de clases y prototipos. No va a formar parte del curso, pero con lo aprendido en Java tienen una buena base para investigar por su cuenta lo que les falta propio de JS si les interesa

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

¡Muchas Gracias!

continuará...

Ministerio de
Desarrollo Productivo
Argentina