UNIVERSIDAD
DE MÁLAGA

Iniciación a JavaScript

Variables y funciones

Daniel Garrido

dgm@uma.es

UNIVERSIDAD
DE MÁLAGA

Variables

Usamos variables para almacenar datos

Declarar e inicializar en 2 sentencias:

```
var x;
x = 5;
console.log(x);
```

O declarar e inicializar en una sentencia:

```
var y = 2;
console.log(y);
```

Posteriormente veremos ***let*** y ***const*** como forma alternativa (y preferible) de declaración

Nombres de Variables

- Comienzan con letras, $ o _
- Solo contienen letras, números, $ y _
- Distinguen mayúsculas y minúsculas. Evitar las palabras reservadas
- Utilizar nombres claros y con significado
- Utilizar estilo “camelCase" para múltiples palabras

OK:

```
var numPeople, $mainHeader, _num, _Num;
```

Incorrecto:

```
var 2coolForSchool, soHappy!
```

Tipos de Datos Primitivos

- **String:** cadena **inmutable** de caracteres:

```
var greeting = 'Hello Kitty';
var restaurant = "Pamela's Place";
```

- **Número:** enteros (6, -102) o punto flotante (5.8737):

```
var myAge = 28;
var pi = 3.14;
```

- **Lógico:** Representa valores lógicos verdadero o falso:

```
var catsAreBest = true;
var dogsRule = false;
```

Tipos de Datos Primitivos

- **undefined:** Representa un valor que no ha sido definido.

```
var notDefinedYet;
```

- **null:** Representa un valor vacío explícito.

```
var goodPickupLines = null;
```

Expresiones

Las variables pueden también almacenar el resultado de cualquier "expresión" usando la “asignación” (=):

```
var x = 2 + 2;
var y = x * 3;
```

```
var name = 'Claire';
var greeting = 'Hello ' + name;
var title = 'Baroness';
var formalGreeting = greeting + ', ' + title;
```

Las variables pueden almacenar entradas del usuario usando la función **prompt**.

```
var name = prompt("What's your name?");
console.log('Hello ' + name);
```

Operadores

Operadores clásicos similares a otros lenguajes:

+ Suma
- Resta
* Multiplicación
** Potencia (ES7)
/ División
% Módulo
++ Incremento
-- Decremento

Operadores de asignación tipo += (también para strings), -= etc.

Concatenación de strings: +

Operadores

Más ejemplos:

```
var x = 10;
x += 5;
```

```
var text1 = "What a very ";
text1 += "nice day";
```

```
var x = 5;
var y = 2;
var z = x / y;
```

```
var x = 5;
x++;
var z = x;
```

```
var x = 5;
var z = x ** 2;
```

```
var x = 5;
var y = 2;
var z = x % y;
```

Tipado Débil

JS deduce el tipo basado en el valor, y el tipo puede cambiar:

```
var x;
console.log(typeof x)   // undefined

x = 2;
console.log(typeof x)   // number

x = 'Hi';
console.log(typeof x)   // string
```

Una variable solo puede tener un tipo:

```
var y = 2 + ' cats';
console.log(typeof y);
```

Modo Estricto

Podemos activarlo escribiendo al inicio:

```
"use strict";
```

No permite, por ejemplo, usar variables no declaradas

```
"use strict";
x=3;
console.log(x);
```

Se puede aplicar de manera global o dentro de funciones

```
function myFunction() {
  "use strict";
  y = 3.14;   // Esto produce un error
}
```

Ámbito de las Variables

Las variables en JS tienen "ámbito de función". Solo son visibles en la función donde son definidas:

```
function addNumbers(num1, num2) {
  var localResult = num1 + num2;
  console.log("The local result is: " + localResult);
}
addNumbers(5, 7);
console.log(localResult);  // ReferenceError
```

Una variable con ámbito "global":

```
var globalResult;

function addNumbers(num1, num2) {
  globalResult = num1 + num2;
  console.log("The global result is: " + globalResult);
}
addNumbers(5, 7);  console.log(globalResult);  // 12
```

La Declaración let

- Ámbito de bloque { }
- Los valores pueden ser reasignados

```
let name = 'Zeta';
{
    let name = 'Alpha';
    console.log(name);
}
console.log(name);
```

Prueba con var para ver la diferencia

La Declaración const

- Ámbito de bloque, como let
- Los valores no pueden ser modificados
  - Los contenidos de objetos complejos como objetos y arrays sí pueden ser modificados
- Deben ser inicializadas en la declaración
- ¡No hay necesidad de usar var!

```
const MAX_ITEMS = 6;
console.log(MAX_ITEMS);
MAX_ITEMS = 2; // TypeError: Assignment to constant variable
console.log(MAX_ITEMS);
```

Más sobre Strings

Un string almacena una lista ordenada de caracteres:

```
var alphabet = "abcdefghijklmnopqrstuvwxyz";
```

La propiedad length retorna el tamaño del string:

```
console.log(alphabet.length); // 26
```

Cada carácter tiene un índice donde el primero empieza en el índice 0 y el ultimo está en length-1:

```
console.log(alphabet[0]); // 'a'
console.log(alphabet[1]); // 'b'
console.log(alphabet[2]); // 'c'
console.log(alphabet[alphabet.length]); // undefined
console.log(alphabet[alphabet.length-1]); // 'z'
console.log(alphabet[alphabet.length-2]); // 'y'
```

Métodos para Strings

- indexOf retorna la primera ocurrencia de un string en otro
- lastIndexOf retorna la última ocurrencia de un string en otro
- split
  devuelve un array con las palabras de un string
- slice, substring, substr para obtener subcadenas
- trim elimina los espacios iniciales y finales
- Infinidad de métodos...

Métodos para Strings

Algunos ejemplos:

```
var str = "Apple, Banana, Kiwi";
str.slice(7, 13)     // Retorna Banana

var text = "Hola Antonio!";
var newText = text.replace(“Hola", “Hi");

var text1 = "Hello World!";       // String
var text2 = text1.toUpperCase();  // Todo a mayúsculas

var text = "       Hello World!        ";
text.trim()    // Retorna "Hello World!"

var str = "How are you doing today?";
var myArr = str.split(" "); // Separa las palabras
```

Métodos de Ayuda para Strings (ES6)

- includes() retorna true si el texto dado es incluido en otro string
- startsWith() retorna true si el texto dado es encontrado al inicio del string
- endsWith() retorna true si el texto dado es encontrado al final del string

```
const message = 'abcdef';

console.log(message.includes('b')); // true
console.log(message.includes('c', 2)); // true -- incluye c en

console.log(message.startsWith('abc')); // true
console.log(message.startsWith('A')); // false --minus/mayus

console.log('endsWith()');
console.log(message.endsWith('def')); // true
```

Template Strings

```
const message = `User ${name} scored ${score} on the exam`;
```

- Usar (` `) alrededor del string
- Usar ${ } alrededor de cualquier variable o expresión
- Permite strings multilínea

```
`<div>
  <h1>Hello ${str}!</h1>
</div>`;
```

Conversiones para Números

- isNaN indica si un valor no es un número
  - Valores especial NaN e Infinity,-Infinity
- toString retorna un número como String
- parseInt procesa el valor y retorna un entero
- parseFloat procesa el valor y retorna un flotante

```
> cad="3.14"
< '3.14'
> num=parseFloat(cad)
< 3.14
>
```

Comentarios

Como en C, C++:

```
// Comentario de una sola línea
var x = 4; // O después del código

/*
También se pueden escribir comentarios en múltiples
líneas,  para comentarios de mayor tamaño.
*/
```

¡TIEMPO DE EJERCICIOS!
Variables

Funciones

- Las funciones permiten reutilizar código.
- Primero, declaramos la función:

```
function sayMyName() {
  console.log('Hi Christina!');
}
```

Y después la llamamos (tantas veces como queramos):

```
sayMyName();
```

Funciones

Las funciones pueden aceptar cualquier número de **argumentos**:

```
function sayMyName(name) {
  console.log('Hi, ' + name);
}

sayMyName('Claire');
sayMyName('Testy McTesterFace');
```

```
function addNumbers(num1, num2) {
  var result = num1 + num2;
  console.log(result);
}

addNumbers(7, 21);
addNumbers(3, 10);
```

Funciones

También se pueden usar variables como argumentos:

```
var number = 10;
addNumbers(number, 2);
addNumbers(number, 4);
```

Y valores por defecto (ES6) (ejemplo “avanzado”)

```
function makeRequest(url, timeout = 2000, callback = () => {}) {
  callback(); }

makeRequest('/api');
makeRequest('/api', 500);
makeRequest('/api', 0, requestData => updateDatabase(requestData));
makeRequest('/api', undefined, requestData =>
  updateDatabase(requestData));
```

Funciones

Colocando ... delante de un parámetro, transforma a ese parametro y al resto en un array

```
function giveItARest(param1, ...allTheRest) {
  console.log(param1);
  console.log(allTheRest);
}

giveItARest(1, 2, 3, 4, 5, 6);
// 1
// [2, 3, 4, 5, 6]
```

Usando ... delante de un array, lo "expande" a sus elementos individuales

```
const arr1 = ['a', 'b', 'c'];
const arr2 = ['x', 'y', 'z'];
const combo = [...arr1, ...arr2];
console.log(combo); // ['a', 'b', 'c', 'x', 'y', 'z']
```

Funciones

La palabra clave **return** devuelve un valor a quien usó la función (y finaliza la función):

```
function addNumbers(num1, num2) {
  var result = num1 + num2;
  return result; // Todo lo que venga después, no será ejecuta
}

var sum = addNumbers(5, 2);
```

Se pueden usar llamadas en las expresiones:

```
var biggerSum = addNumbers(2, 5) + addNumbers(3, 2);
```

Funciones Arrow (Flecha)

```
var getMessage = function(name) {
  return 'Hello ' + name + '!';
}
```

ES6

```
const getMessage = name => `Hello ${name}!`;
```

1. Borrar la palabra function
2. Poner la flecha (=>) después de los parámetros

- Si solo hay un parámetro, quitar los paréntesis ( )
- Si solo hay una empresión en el cuerpo de la función, quitar las { } y `return` (implícito)

Funciones Arrow (Flecha)

Algunos ejemplos

```
const sum = (num1, num2) => num1 + num2;

console.log(sum(3, 4)); // 7
```

```
const fibonacci = n => {
  if (n < 3) {
    return 1;
  }
  return fibonacci(n - 1) + fibonacci(n - 2);
};

console.log(fibonacci(9)); // 34
```

Convenciones de Código

- JavaScript Standard Style (https://standardjs.com/)
- Airbnb JavaScript Style Guide (https://github.com/airbnb/javascript)
- Google JavaScript Style Guide (https://google.github.io/styleguide/jsguide.html)
- Idiomatic JavaScript Style Guide (https://github.com/rwaldron/idiomatic.js/)
- jQuery JavaScript Style Guide (https://contribute.jquery.org/style-guide/js/)

Convenciones de Código

Un ejemplo...

Mal:

```
function addNumbers(num1,num2) {return num1 + num2;}

function addNumbers(num1, num2) {
return num1 + num2;
}
```

Bien:

```
function addNumbers(num1, num2) {
  return num1 + num2;
}
```

¡TIEMPO DE EJERCICIOS!
Funciones

Ejercicio Adicional

Calculadora

Acknowledgements / Reconocimientos:

Gran parte del material del curso procede originalmente de la web Teaching Materials Posteriormente, han sido enriquecidos con material adicional de diversas fuentes públicas así como adaptados a las necesidades específicas del curso a impartir.