Introducción a JS

Taller parte 1 de 3

Rodrigo Francisco
Armando Rivera

Agenda

JavaScript

JavaScript (JS) es un lenguaje de programación multiplataforma

- Se considera un lenguaje orientado a objetos.
- Es un lenguaje basado en prototipos.
- Es un lenguaje interpretado (o just-in-time compiled).

JavaScript

JavaScript (JS) es un lenguaje de programación multiplataforma

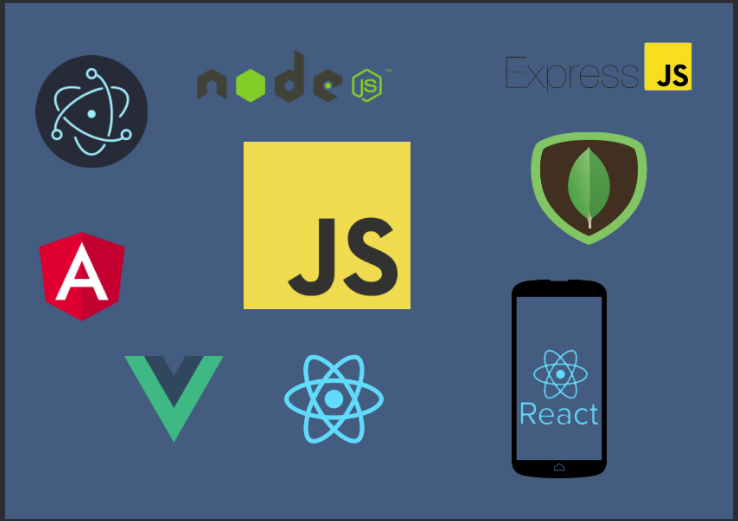
- Java y Javascript no tienen relación como lenguajes de programación.
- Las implementación de JS se rigen por el estándar ECMA-262.
- El nombre oficial del lenguaje es ECMAScript.

JavaScript

JavaScript (JS) es un lenguaje de programación multiplataforma

- Fue inventado en 1995 por Brendan Eich
- JS originalmente se creo para *darle vida* a los navegadores.
- JS nació cómo un lenguaje del lado del cliente.

¿Por qué aprender JS?



¿Por qué aprender JS?

Existen muchas razones para aprender JS

- Se pueden crear aplicaciones para casi todas las plataformas
- MongoDB y CouchDB usan javacript como su lenguaje de programación por defecto.
- Existen librerías/paquetes desarrolladas por la comunidad para casi todo tipo de requerimientos

¿Por qué aprender JS?

Ventajas y desventajas

Ventajas

- Simplicidad. La sintaxis de javascript es muy sencilla.
- Popularidad. Javascript es un lenguaje muy popular lo cual le garantiza tener soluciones implementadas por la comunidad para casi todas las necesidades.
- Interoperable. Javascript puede ser usado en junto con otros lenguajes de programación como Pearl y PHP.
- Funcionalidad extendida. Js permite extender la funcionalidad de algunos sitios web al incorporar scripts de js a estos.
- Rápido. Js tiende a ser un lenguaje rápido siempre que no tenga que obtener recursos externos.

¿Por qué aprender JS?

Ventajas y desventajas

Desventajas

- Vulnerabilidades del lado del cliente. Al ser un lenguaje del lado del cliente en ocasiones se suelen encontrar un bug que puede ser explotado para propósitos malicisos.
- Soporte de navegadores. En pleno 2020 esto ya no es una desventaja, actualmente todos los navegadores soportan JS en sus últimas versiones.
- Tricky language. A veces un puede tener dolores de cabeza al debugear un programa escrito en JS por que a veces hace varias asumpciones implicitas.

Evolución del lenguaje

ES5, ES6

En las primeras versión del lenguaje hasta la versión ES5, no se incorporaban clases ni exponenciación en la versión ES6.
ES6 tiene las siguientes características:

- JavaScript let
- JavaScript const
- JavaScript Arrow Functions
- JavaScript Classes
- Parámetros con valores por defecto
- Array.find()
- Array.findIndex()
- Exponentiation (**) (EcmaScript 2016)

Motores de JS

Spider Monkey, V8, JScript

Google, Mozilla y Microsoft tiene sus propios motores de JS

- Spider Monkey es el nombre del motor de JS de Mozilla
- V8 es el motor de JS de Google Chrome
- Jscript es desarrollado por Microsoft.

Herramientas para ejecutar JS

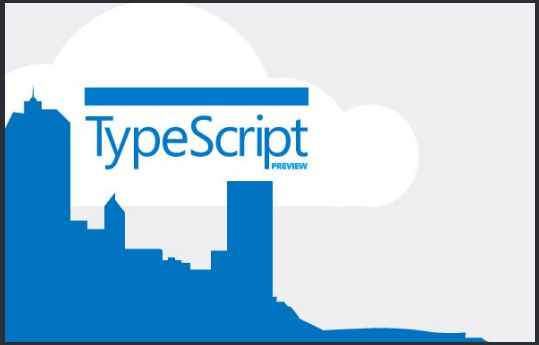
PlayCode,JSFiddle

- JavaScript.com (`https://www.javascript.com/try`)
- PlayCode (`https://playcode.io/`)
- ES6 console (`https://es6console.com/`)
- jsconsole (`https://jsconsole.com/`)
- jsfiddle (`https://jsfiddle.net/`)
- Plunker (`https://plnkr.co/`)
- JSbin (`https://jsbin.com/?html,output`)
- CodePen (`https://codepen.io/`)
- Stackblitz (`https://stackblitz.com/`)

Typescript

El hermano perdido de JS

TypeScript es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft.



Referencias

Docs, MDN

- La mejor documentación que he tenido oportunidad de leer está en la página de desarolladores de Mozilla.
  - MDN web doc (`https://developer.mozilla.org/en-US/docs/Web/JavaScript`)
  - JavaScript reference (`https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference`)
- w3schools.com (`https://www.w3schools.com/Js/`)
- javascript.info(`https://javascript.info/`)
- What the f*ck is JavaScript? (`https://github.com/denysdovhan/wtfjs`)
- jsher (`https://www.jshero.net/en/success.html`)

Agenda

Sintaxis básica

Las reglas

- JS es case-sensitive, sensible a mayúsculas y minúsculas
- Usa el conjunto de caracteres Unicode
- El punto y coma (;) son es necesario para separar sentencias (instrucciones).

Un programa de computadora es una lista de *instrucciones* para ser *ejecutadas* por una computadora.
En un lenguaje de programación, estas instrucciones de programación se denominan alertsentencias.
Un programa de JavaScript es una lista de sentencias/instrucciones de programación.

Sintaxis básica

Palabrar reservadas

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

Sintaxis básica

Comentarios

Los comentarios son sentencias o comentarios (valga la redundancia) que no serán tomados en cuenta por el intérprete.
Para poner comentarios en JS tenemos las siguientes opciones.

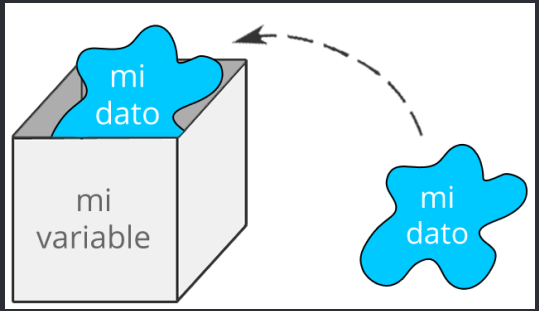
- Comentario de una línea, se utiliza `//`
- Comentario de multiples líneas: Tiene un inicio y un fin
  - El inicio se denota con `/*` y el fin com `*/`

¿Qué es un variable?

Declaración de variables, var, const, let

Una variable es donde se guarda (y se recupera) datos que se utilizan en un programa.
Para declarar una variable en JS se puede hacer de 3 formas

- Utilizando la palabra reservada var
- Utilizando la palabra reservada const
- Utilizando la palabra reservada let



Variables

Indentificadores

Los identificadores se usan para nombrar variables (y palabras clave, funciones y etiquetas).
Todas las variables de JavaScript deben identificarse con nombres únicos (identificadores).

Variables

Reglas para los indentificadores

Las reglas generales para construir nombres para variables (identificadores únicos) son:

- Los nombres pueden contener letras, dígitos, guiones bajos y signos de dólar.
- Los nombres deben comenzar con una letra
- Los nombres también pueden comenzar con $ y _
- Los nombres distinguen entre mayúsculas y minúsculas (y e Y son variables diferentes)
- Las palabras reservadas (como las palabras clave de JavaScript) no se pueden usar como nombres

Tipos de datos

Según ECMAScript

En total tenemos 8 tipos de datos.

- Siete tipos de datos que son primitivos:
  - Boolean. Verdadero y falso.
  - null. Una palabra clave especial que denota un valor nulo.
  - undefined. Una propiedad de nivel superior cuyo valor no está definido.
  - Number. Un número entero o de coma flotante. Por ejemplo: 42 o 3.14159.
  - BigInt. Un entero con precisión arbitraria. Por ejemplo: 9007199254740992n.
  - String. Una secuencia de caracteres que representan un valor de texto. Por ejemplo: "Hola"
  - Símbolo. (nuevo en ECMAScript 2015). Un tipo de datos cuyas instancias son únicas e inmutables.
- y Object

Tipos de datos

Numbers

Los tipos Number y BigInt se pueden escribir en decimal (base 10), hexadecimal (base 16), octal (base 8) y binario (base 2).

- Un literal numérico decimal es una secuencia de dígitos sin un 0 inicial (cero).
- Un 0 inicial (cero) en un literal numérico, o un 0o inicial (ó 0O) indica que está en octal.
- Un 0x inicial (o 0X) indica un tipo numérico hexadecimal.
- Si al número se le agrega una n al final se convierte en un BigInt
- Un 0b inicial (o 0B) indica un literal numérico binario.

Tipos de datos

Strings

Caracteres especiales

Table: JavaScript special characters

Character	Meaning
`\0`	Null Byte
`\b`	Backspace
`\f`	Form feed
`\n`	New line
`\r`	Carriage return
`\t`	Tab
`\v`	Vertical tab
`\'`	Apostrophe or single quote
`\"`	Double quote
`\\`	Backslash character

Operadores

Operadores de comparación

Comparison operators

Operator	Description	Examples returning true

Equal (==)	Returns `true` if the operands are equal.	`3 == var1`
		`"3" == var1`
		`3 == '3'`

Not equal (!=)	Returns `true` if the operands are not equal.	`var1 != 4` `var2 != "3"`
Strict equal (===)	Returns `true` if the operands are equal and of the same type. See also `Object.is` and sameness in JS.	`3 === var1`
Strict not equal (!==)	Returns `true` if the operands are of the same type but not equal, or are of different type.	`var1 !== "3"` `3 !== '3'`
Greater than (>)	Returns `true` if the left operand is greater than the right operand.	`var2 > var1` `"12" > 2`
Greater than or equal (>=)	Returns `true` if the left operand is greater than or equal to the right operand.	`var2 >= var1` `var1 >= 3`
Less than (<)	Returns `true` if the left operand is less than the right operand.	`var1 < var2` `"2" < 12`
Less than or equal (<=)	Returns `true` if the left operand is less than or equal to the right operand.	`var1 <= var2` `var2 <= 5`

Operadores

Operadores aritméticos

Arithmetic operators

Operator	Description	Example
Remainder (`%`)	Binary operator. Returns the integer remainder of dividing the two operands.	12 % 5 returns 2.
Increment (`++`)	Unary operator. Adds one to its operand. If used as a prefix operator (`++x`), returns the value of its operand after adding one; if used as a postfix operator (`x++`), returns the value of its operand before adding one.	If `x` is 3, then `++x` sets `x` to 4 and returns 4, whereas `x++` returns 3 and, only then, sets `x` to 4.
Decrement (`--`)	Unary operator. Subtracts one from its operand. The return value is analogous to that for the increment operator.	If `x` is 3, then `--x` sets `x` to 2 and returns 2, whereas `x--` returns 3 and, only then, sets `x` to 2.
Unary negation (`-`)	Unary operator. Returns the negation of its operand.	If `x` is 3, then `-x` returns -3.
Unary plus (`+`)	Unary operator. Attempts to convert the operand to a number, if it is not already.	`+"3"` returns `3`. `+true` returns `1.`
Exponentiation operator (`**`)	Calculates the `base` to the `exponent` power, that is, $base^{exponent}$	`2 ** 3` returns `8`. `10 ** -1` returns `0.1`.

Operadores

Operadores bit a bit

Bitwise operators

Operator	Usage	Description
Bitwise AND	`a & b`	Returns a one in each bit position for which the corresponding bits of both operands are ones.
Bitwise OR	`a | b`	Returns a zero in each bit position for which the corresponding bits of both operands are zeros.
Bitwise XOR	`a ^ b`	Returns a zero in each bit position for which the corresponding bits are the same. [Returns a one in each bit position for which the corresponding bits are different.]
Bitwise NOT	`~ a`	Inverts the bits of its operand.
Left shift	`a << b`	Shifts `a` in binary representation `b` bits to the left, shifting in zeros from the right.
Sign-propagating right shift	`a >> b`	Shifts `a` in binary representation `b` bits to the right, discarding bits shifted off.
Zero-fill right shift	`a >>> b`	Shifts `a` in binary representation `b` bits to the right, discarding bits shifted off, and shifting in zeros from the left.

Operadores

Operadores lógicos

Logical operators

Operator	Usage	Description
Logical AND (`&&`)	`expr1 && expr2`	Returns `expr1` if it can be converted to `false`; otherwise, returns `expr2`. Thus, when used with Boolean values, `&&` returns `true` if both operands are true; otherwise, returns `false`.
Logical OR (`||`)	`expr1 || expr2`	Returns `expr1` if it can be converted to `true`; otherwise, returns `expr2`. Thus, when used with Boolean values, `||` returns `true` if either operand is true; if both are false, returns `false`.
Logical NOT (`!`)	`!expr`	Returns `false` if its single operand that can be converted to `true`; otherwise, returns `true`.

Estructuras de control

if, else, else if

La instrucción if se usa para especificar un bloque de código Jque se ejecutará si una condición es verdadera.

Sentencia if

```
if (condition_1) {
  statement_1;
} else if (condition_2) {
  statement_2;
} else if (condition_n) {
  statement_n;
} else {
  statement_last;
}
```

Estructuras de control

Valores falsy

Los siguientes valores se evalúan como falsos (también conocidos como valores Falsy):

- false
- undefined
- null
- 0
- NaN
- the empty string (“")

Estructuras de control

Operador ternario

Si la condición es verdadera, el operador tiene el valor que esta después del signo de interrogación. De lo contrario, tiene el valor que está después de los 2 puntos.

Operador ternario u Operador condicional

```
condition ? val1 : val2

var status = (age >= 18) ? 'adult' : 'minor';
```

Estructuras de control

switch

La instrucción `switch` permite que un programa evalúe una expresión e intente hacer coincidir el valor de la expresión con una etiqueta de caso. Si se encuentra una coincidencia, el programa ejecuta la declaración asociada.

Sentencia switch

```
switch (expression) {
  case label_1:
    statements_1
    [break;]
  case label_2:
    statements_2
    [break;]
    . . .
  default:
    statements_def
    [break;]
}
```

Estructuras de iteración

for

Un ciclo for se repite hasta que una condición especificada se evalúe como falsa.

Sentencia for

```
for ([initialExpression]; [condition]; [incrementExpression])
  statement
```

Estructuras de iteración

while

Una sentencia while ejecuta sus sentencias siempre que una condición especificada se evalúe como verdadera.

Sentencia while

```
while (condition)
  statement
```

Estructuras de iteración

do while

La instrucción do … while se repite hasta que una condición específica se evalúa como falsa.

Sentencia do while

```
do
  statement
while (condition);
```

Arreglos

Las características importantes son ...

Un arreglo es una variable especial, que puede contener más de un valor a la vez.

Un arreglo puede contener muchos valores con un solo nombre, y puede acceder a los valores haciendo referencia a un número de índice.

Los arreglos son un tipo especial de objetos.

Funciones

Definición

Una función de JavaScript es un bloque de código diseñado para realizar una tarea en particular.
Una definición de función (también llamada declaración de función o declaración de función) consiste en la palabra clave de función, seguida de:

- El nombre de la función.
- Una lista de parámetros para la función, entre paréntesis y separados por comas.
- Las declaraciones de JavaScript que definen la función, encerradas entre llaves, {...}

Funciones

Tipos de funciones

- Funciones anónimas
- Funciones anidadas
- Funciones con parametro por defecto
- Funciones que reciben funciones
- Paso por valor y paso por referencia
- Arrow Function (introducción)

Referencias y Bibliografía

MDN,w3schools

[1] Mozilla Developer Network. *JavaScript*. Disponible en `https://developer.mozilla.org/en-US/docs/Web/javascript`

[2] w3schools. *JavaScript Tutorial*. Disponible en `https://www.w3schools.com/Js/`

[3] Free Code Camp, *The Best JavaScript Examples*. Disponible en `https://www.freecodecamp.org/news/javascript-example/`

[4] Denys Dovhan. *What the f*ck JavaScript?*. Disponible en `https://github.com/denysdovhan/wtfjs`