EXPRESSIONS AND STATEMENTS

PRIMARIAS

OBJETOS Y ARRAYS

FUNCIONES

ACCESO CONDICIONAL

CREACIÓN DE CLASES

OPERADORES

PRIMARIAS

Valores “a pincho”

- 1.23
- “hola peter”
- /1-9/ (expresiones regulares)
- true
- false
- null
- undefined

OBJETOS Y ARRAYS

ARRAYS

Lista de valores

let vector = [1, 2, 3]

let two_per_two = [[1, 2], [3, 4]]

let undefined_array = [,]

OBJECTS

Lista de propiedades

Parecido a los JSON

let object = { property1: “hola”, property2: 2.3 }

let nested_object = { property1: { stringie: “hola” }, property2: 2.3 }

FUNCIONES

Se definen en una sentencia

Sirven para agrupar código por funcionalidad mínima

```
let suma = function(a, b) {return a + b}
```

ACCESO CONDICIONAL

Sirve para acceder a una propiedad que quizá no exist

Evita un error

let a = {b: null}

a.b?.c // undefined

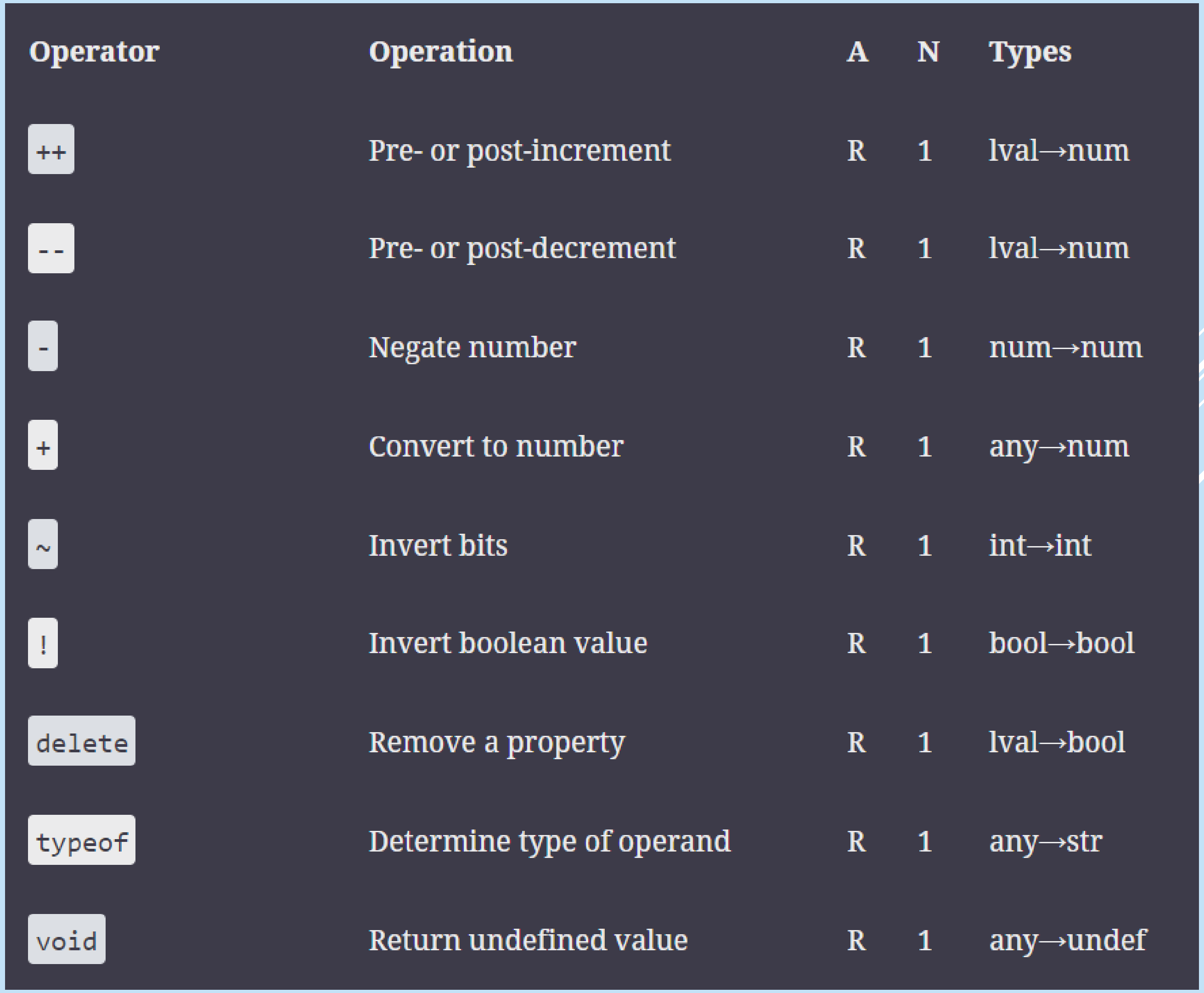
CREACIÓN DE CLASES

Constructor (más adelante)

let object1 = new Object()

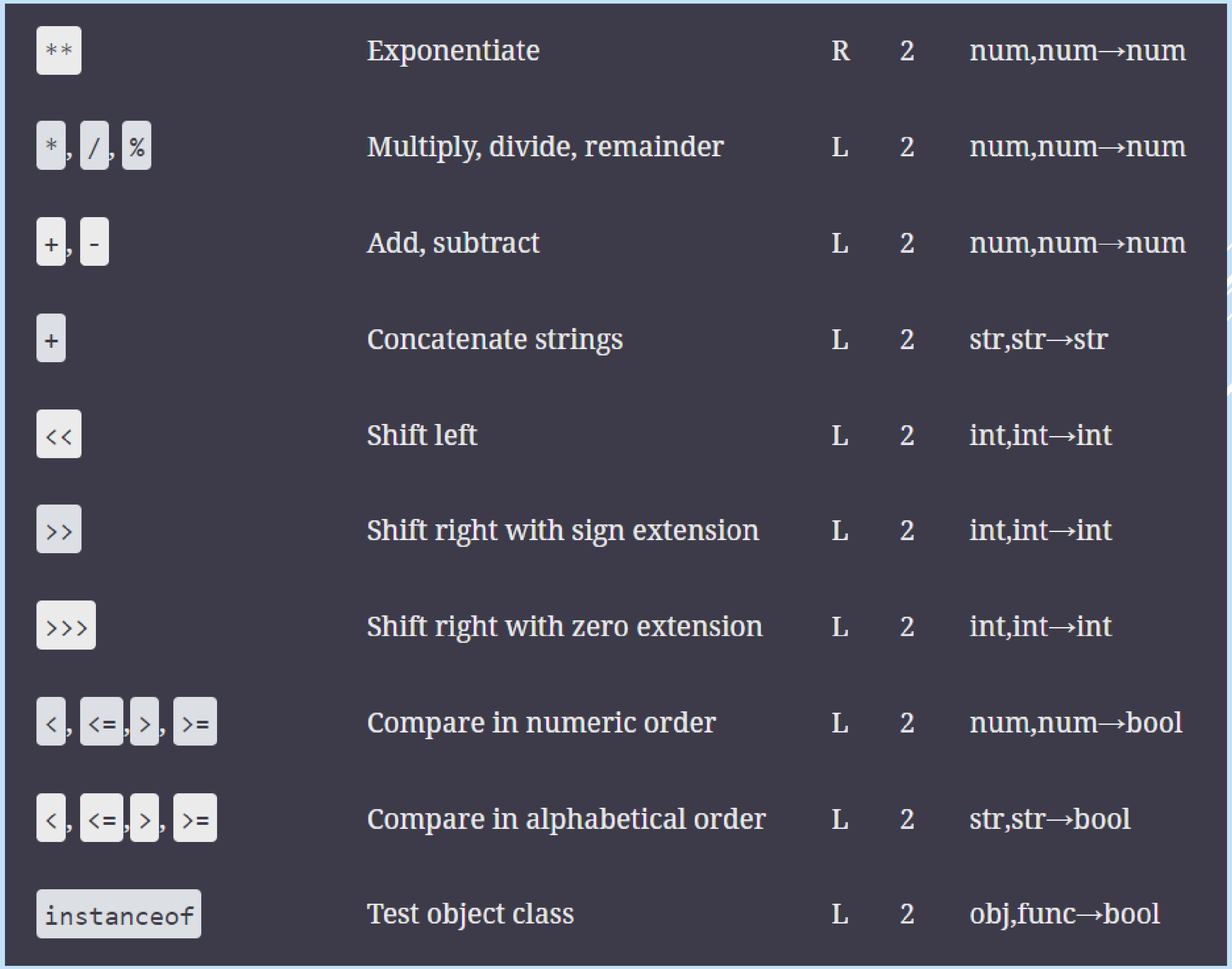
let object2 = new Alumnado([“Martin”, “Alvaro”])

OPERADORES



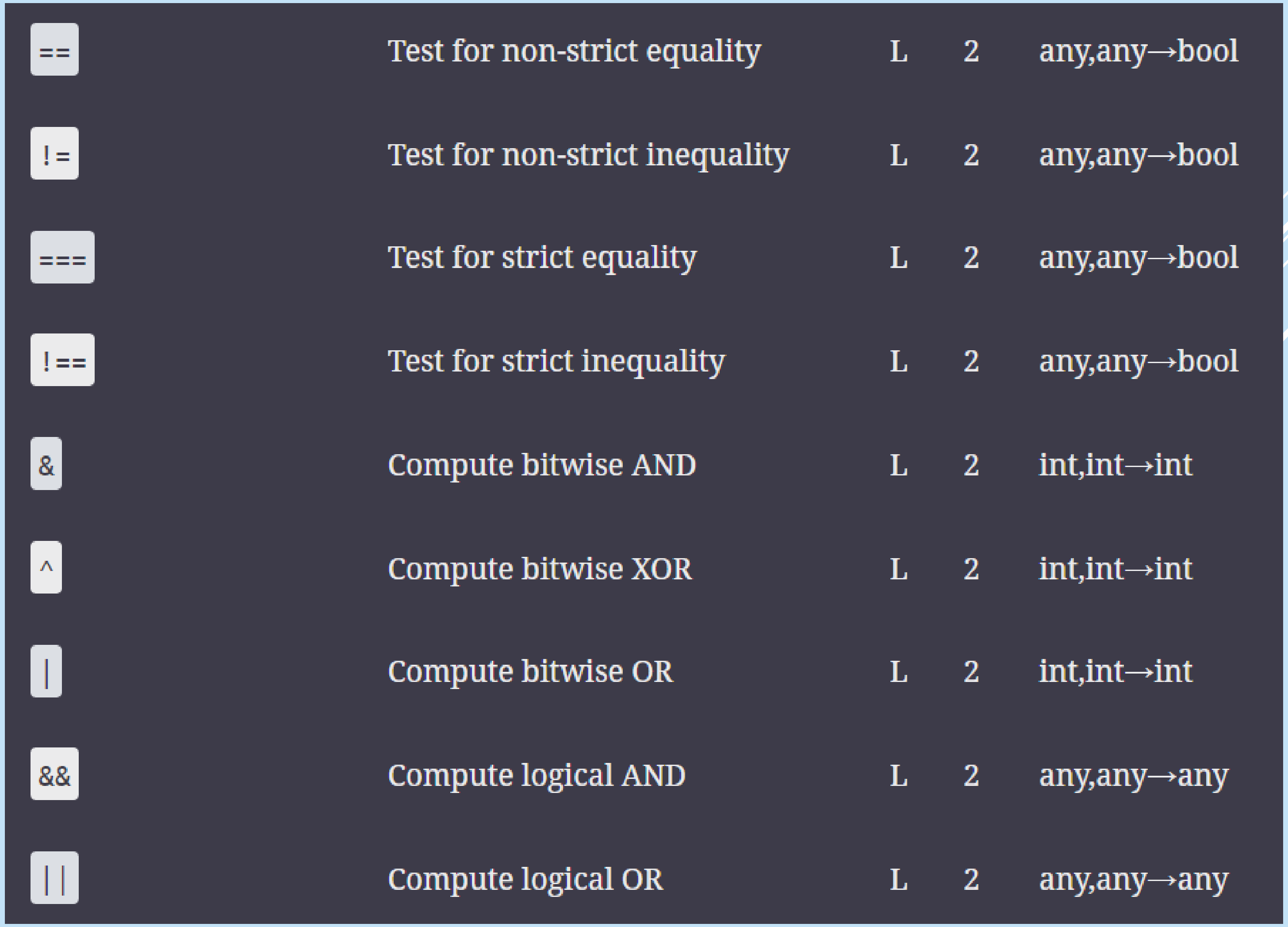
Operator	Operation	A	N	Types
`++`	Pre- or post-increment	R	1	lval→num
`--`	Pre- or post-decrement	R	1	lval→num
`-`	Negate number	R	1	num→num
`+`	Convert to number	R	1	any→num
`~`	Invert bits	R	1	int→int
`!`	Invert boolean value	R	1	bool→bool
`delete`	Remove a property	R	1	lval→bool
`typeof`	Determine type of operand	R	1	any→str
`void`	Return undefined value	R	1	any→undef

OPERADORES



`**`	Exponentiate	R	2	num,num→num
`*`, `/`, `%`	Multiply, divide, remainder	L	2	num,num→num
`+`, `-`	Add, subtract	L	2	num,num→num
`+`	Concatenate strings	L	2	str,str→str
`<<`	Shift left	L	2	int,int→int
`>>`	Shift right with sign extension	L	2	int,int→int
`>>>`	Shift right with zero extension	L	2	int,int→int
`<`, `<=`, `>`, `>=`	Compare in numeric order	L	2	num,num→bool
`<`, `<=`, `>`, `>=`	Compare in alphabetical order	L	2	str,str→bool
`instanceof`	Test object class	L	2	obj,func→bool

OPERADORES



==	Test for non-strict equality	L	2	any,any→bool
!=	Test for non-strict inequality	L	2	any,any→bool
===	Test for strict equality	L	2	any,any→bool
!==	Test for strict inequality	L	2	any,any→bool
&	Compute bitwise AND	L	2	int,int→int
^	Compute bitwise XOR	L	2	int,int→int
|	Compute bitwise OR	L	2	int,int→int
&&	Compute logical AND	L	2	any,any→any
||	Compute logical OR	L	2	any,any→any

OPERADORES

Ternario

3 == 3 ? 1 : 2

Answer: 1

BIBLIOGRAFÍA

1. JavaScript: The Definitive Guide, 7th Edition
2. Eloquent JavaScript, 3th edition, Marijn Haverbeke
3. Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
4. https://www.guru99.com/v-model-software-testing.html, 20/03/2022
5. Modern c++ Programming with Test-Driven Development, Jeff Langr
6. Refactoring: Improve the design of existing Code, Martin Fowler
7. Game programming patterns, Robert Nystrom