



# /JS 101

Introduction to JS





# /Table of contents (I)



**/01** /INTRODUCTION

**/02** /VALUES AND TYPES

**/03** /OPERATORS

**/04** /CONDITIONALS AND  
LOOPS





# /Table of contents (II)



**/05** /FUNCTIONS

**/06** /DATA STRUCTURE



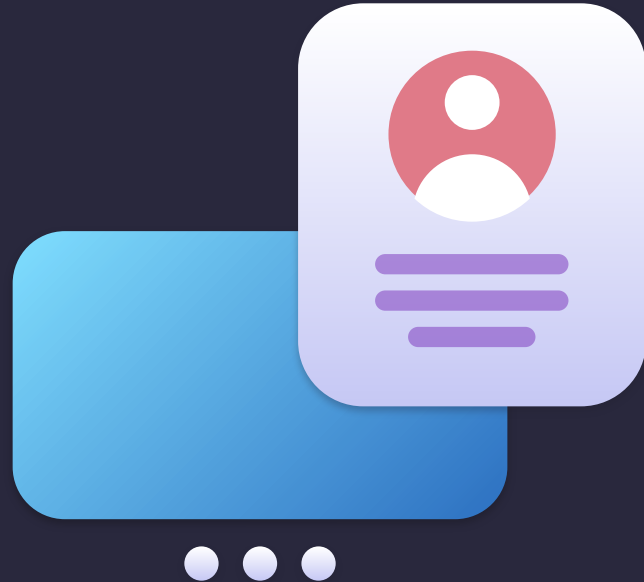
# /Who are we?



**Adal Díaz Fariña**  
**alu0101112251@ull.edu.es**



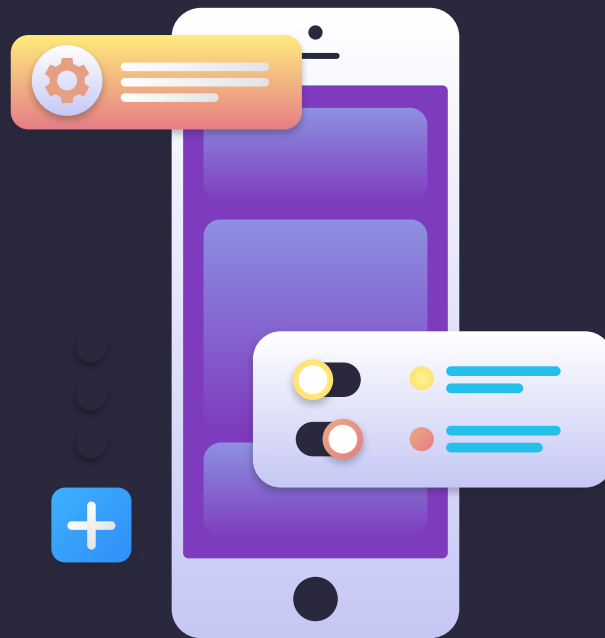
**Marcos Rodríguez Vega**  
**alu0101266329@ull.edu.es**



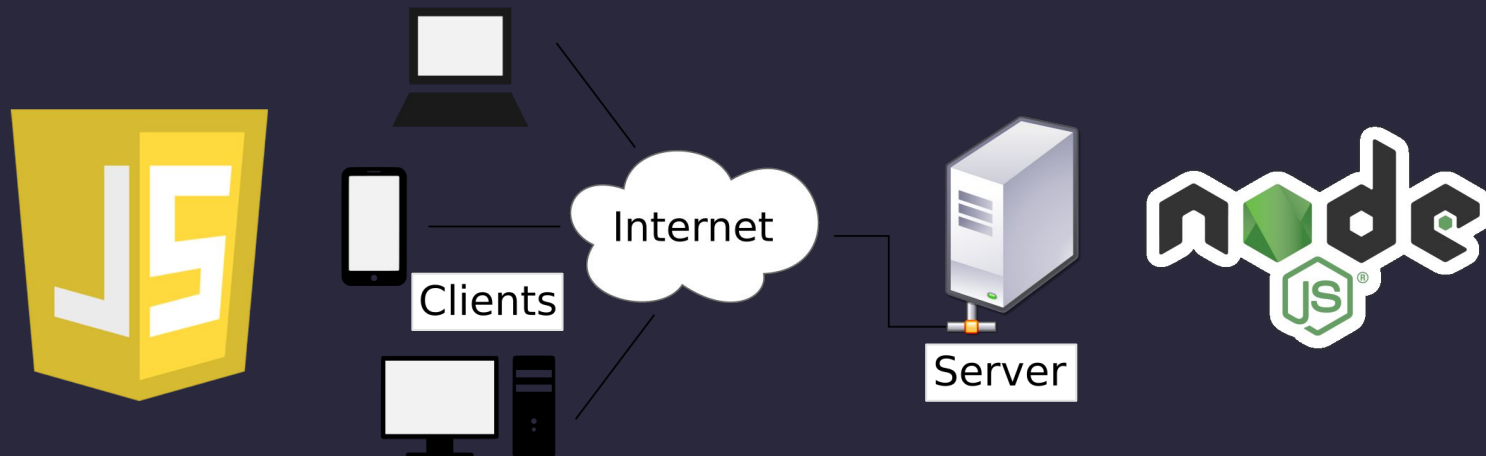


/01

# /INTRODUCTION



# /What is Javascript?



# JS-Javascript

- Multi paradigm - Object Oriented, Functional, etc.
- Dynamically typed
- Weakly typed
- Most implementations are interpreted rather than compiled
- C-like syntax - so you should be pretty familiar with all of the curly braces

# Code Structure: Basics



```
console.log("Hello");  
console.log("World");
```



```
// Schema  
Statement1;  
Statement2;  
.  
.  
.  
StatementN;
```



# Code Structure: Semicolons Problem



```
console.log("Hello")  
[1, 2].forEach(console.log);
```



```
console.log("Hello")  
[1, 2].forEach(console.log); // This is an error
```



```
// This is how the engine sees it  
console.log("Hello")[1, 2].forEach(console.log);
```

# Code Structure: Comments



```
// This is a line comment in JS
```

```
/*  
 * This is a block comment  
 * in JS  
*/
```



```
/* This is fine too */
```

# Code Structure: Nested comments



```
/*  
  /* My comment */  
*/  
console.log("Example");
```

# Code Structure: 'Use Strict';



```
'use strict';
```

```
// this code works the modern way
```



```
alert('some code');
```

```
// "use strict" below is ignored--it must be at the top
```

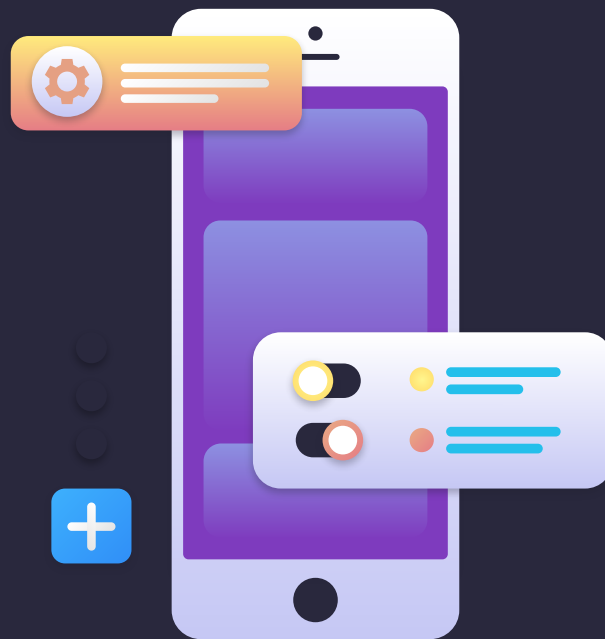
```
'use strict';
```

```
// strict mode is not activated
```



# /02

## /VALUES AND TYPES



# Values



```
317.0; // TYPE: Number
"Oh, hello"; // TYPE: String
[1, 2, 3, 4]; // TYPE: Array
true; // TYPE: Boolean
{'name': 'George'}; // TYPE: Object
function() {
  return 5;
} // TYPE: Function
undefined; // TYPE: Undefined
null; // TYPE: null
```



# Types of values

Type	Description
Number	Numeric value (integers, decimals, etc...)
String	Text value (strings, characters, etc...)
Boolean	Boolean value (true or false values)
Symbol	Unique and immutable values
Null	Null or empty value
Undefined	Undefined values (uninitialized variable)
Function	Function (Function saved in a variable)
Object	Object (More complex structure, for example: Array, Date, String, etc...)



# Typeof



```
let str = "Hola, mundo!";  
let num = 42;  
let boolean = true;  
let undefined1;  
let array = [1, 2, 3];  
let object = {};  
  
console.log(typeof str); // "string"  
console.log(typeof num); // "number"  
console.log(typeof boolean); // "boolean"  
console.log(typeof undefined1); // "undefined"  
console.log(typeof array); // "object"  
console.log(typeof object); // "object"
```



# Variable: var, let and const



```
var myString = "This is a global string"
```



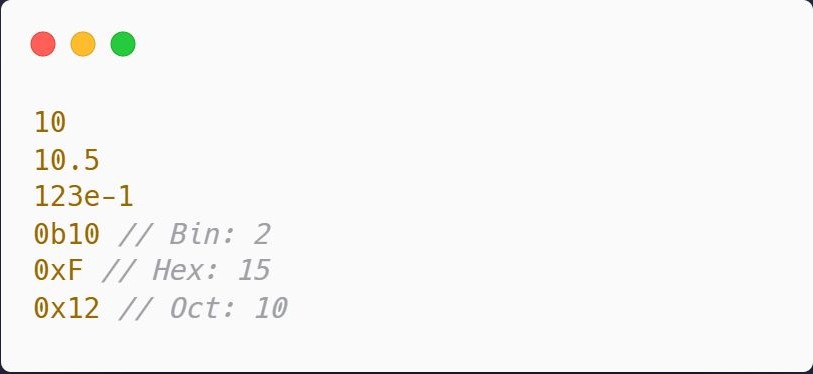
```
let myNumber = 100;  
let myString = "myString";
```



```
const MYNUMBER = 3.14;  
MYNUMBER = 16 // ERROR
```

# Number (I)

- Number
  - Number is a primitive wrapper object used to represent and manipulate numbers
  - JavaScript does not distinguish between ints, floats, longs, doubles, etc.
  - The JavaScript Number type is a double-precision 64-bit binary format IEEE 754 value, like double in Java or C++
- Number literals



```
10
10.5
123e-1
0b10 // Bin: 2
0xF // Hex: 15
0x12 // Oct: 10
```

# Number (II)

Number is subject to rounding and when we have a very large number, higher than what Number can encompass, this value is replaced by Infinity.



```
const x = Number.MAX_SAFE_INTEGER + 1;
const y = Number.MAX_SAFE_INTEGER + 2;

console.log(Number.MAX_SAFE_INTEGER);
// expected output: 9007199254740991

console.log(x);
// expected output: 9007199254740992

console.log(x === y);
// expected output: true
```



```
const biggestNumber = Number.MAX_VALUE;

console.log('\nNumber.MAX_VALUE demo');
function multiply(x, y) {
  return (x * y);
}

console.log(multiply(biggestNumber, 1));
// expected output: 1.7976931348623157e+308

console.log(multiply(biggestNumber, 2));
// expected output: "Process as Infinity"

console.log(biggestNumber === (biggestNumber -
1));
// expected output: true
```

# Number (III)

To represent exact floating point numbers or very large integers we need to make use of the BigInt, on the other hand we can also make use of rounding with different methods:

1. toFixed
2. toPrecision
3. toExponential

```
// Return a string representing the number in a exponential notation
const exponential = (num, decimals) => Number.parseFloat(num).toExponential(decimals);
console.log(`(toExponential) Number 123456 in exponential notation: ${exponential(123456, 2)}`); // expected output: 1.23e +5
// Returns a string representing the number in fixed-point notation.
const fixed = (num, decimals) => Number.parseFloat(num).toFixed(decimals);
console.log(`(toFixed) Number 12.12345 in fixed-point notation: ${fixed(12.12345, 4)}`); // expected output: 12.1235
// Returns a string representing the number to a specified precision in fixed-point or exponential notation
const precision = (num, value) => num.toPrecision(value); // expected output: 12.12
console.log(`(toPrecision) Number 12.12345 in a specified precision in fixed-point or exponential notation
${precision(12.12345, 4)}`);
```



# Strings Literals



```
let str = "Hello"; // Normal double quotes
let str2 = 'Single quotes are ok too'; // Single quotes too
let phrase = `can embed another ${str}`; // You can embed variable with backticks
let num1 = 120;
let phrase2 = `60 * 2 is equal to: ${num1}`; // You can also embed numbers!
```

# String methods and properties

Live Example

# Type Conversion



# Type Conversion: Strings Conversion



```
let value = true;  
console.log(typeof value); // boolean  
  
value = String(value); // now value is a string "true"  
console.log(typeof value); // string
```



```
let number = 180;  
console.log(number.toString()); // '180'
```

# Type Conversion: Numeric Conversion



```
console.log(+true) // return 1
console.log(+"") // return 0
console.log(+" ") // return 0

// case 1 conversion string to number

let num1 = "6";
let num2 = "7";

console.log(+num2 + -num1); // 1

// case 2 conversion number literal to string

console.log("2" + 1 + 1) // 211, the eval order is (("2" + 1) + 1) => ("21" + 1) => "211"
console.log(1 + 1 + "2") // 22, the eval order is ((1 + 1) + "2") => (2 + "2") => "22"
```

# Type Conversion: Numeric Conversion



```
console.log(Number(" 123  ")); // 123
console.log(Number("123z"));    // NaN (error reading a number at "z")
console.log(Number(true));      // 1
console.log(Number(false));     // 0
```

# Type Conversion: Boolean

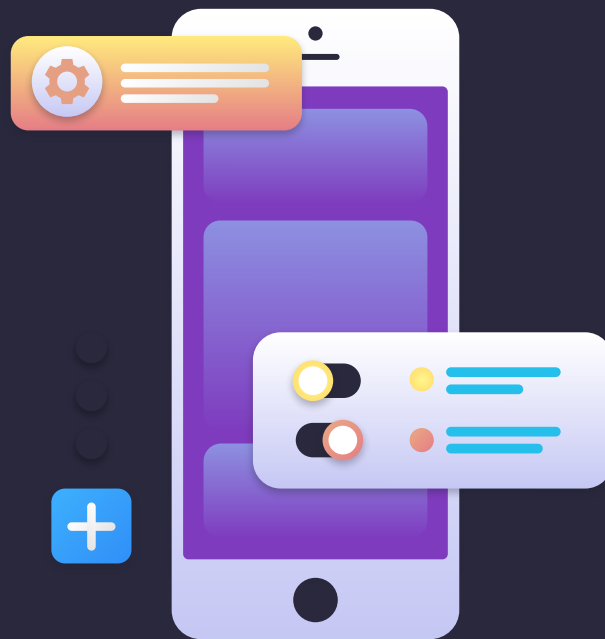


```
console.log(Boolean(1)); // true  
console.log(Boolean(0)); // false  
console.log( Boolean("hello") ); // true  
console.log( Boolean("") ); // false
```



# /03

# /OPERATORS



# Basic Math operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (sometimes called modulo)
**	Exponent

Operator	Name
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment

# Comparators

Operator	Name
===	Strict equality
!==	Strict-non-equality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

# Logical Operators

Symbol	Description
&&	Logic AND
	Logic OR
!	Logic NOT
??	Nullish coalescing operator (NEW)



# Ternary Operator



```
let howitwork1 = (true) ? true: false; // true  
let howitwork2 = (!true) ? true: false; // false
```

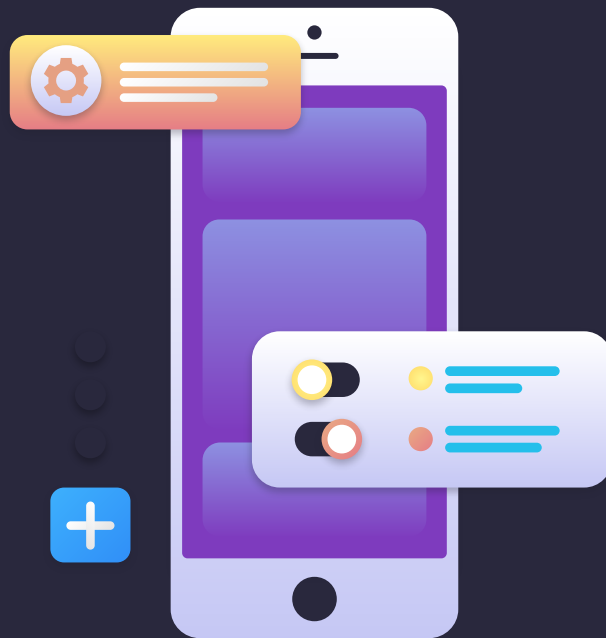


```
let howItWorkWithFunction = (true) ? functionA(): functionB(); // Function A  
let howItWorkWithFunction = (!true) ? functionA(): functionB(); // Function B
```



# /04

## /CONDITIONALS AND LOOPS



# If Statement



*// if statement examples*

```
if (5 == "5" && 8 == "8") {  
    console.log("C++ programmer: 'How in the world a string number is equal to it number literal!!!'");  
} else {  
    console.log("Everything is okey :)");  
}
```

# Switch Statement




```
switch(parseInt((Math.random() * 100) % 4)) {  
  case 0:  
    console.log("The random number is multiple of 4");  
    break;  
  case 1:  
    console.log("The reminder is 1");  
    break;  
  case 2:  
    console.log("The reminder is 2");  
    break;  
  case 3:  
    console.log("The reminder is 3");  
    break;  
}
```

# Basic loops

The loops are very similar to C++, java, etc...

Types of loops:

1. for
2. while
3. do..while
4. for..in
5. for..of



```
//      initialization
//      |          condition
//      |          |          afterthought/increment
//      |          |          |
for (let i = 0; i < 5; i++) {
  console.log(`Value of i = ${i}`);
}
```



```
while (boolean_expression) {  
    // repeat this stuff as long as boolean  
    // expression is true  
}
```

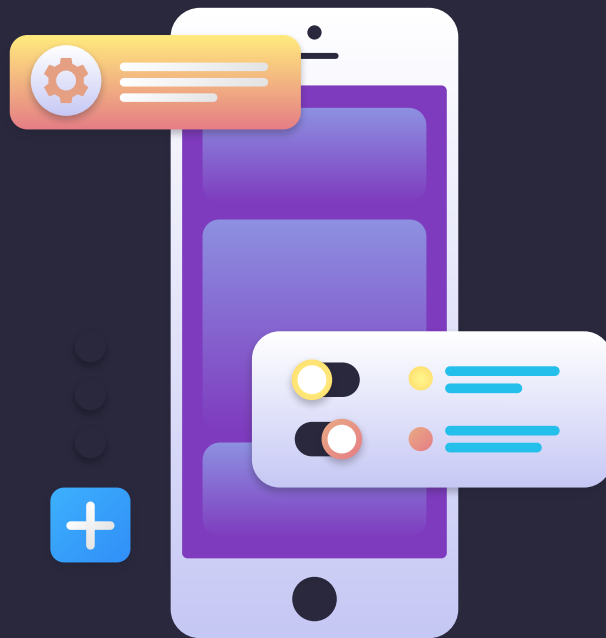


```
do {  
    // repeat this stuff at least once  
} while (boolean_expression)
```



/05

/FUNCTIONS



# Function Declaration



```
function name(parameter1, parameter2, ... parameterN) {  
  ...body...  
}
```



```
name();
```

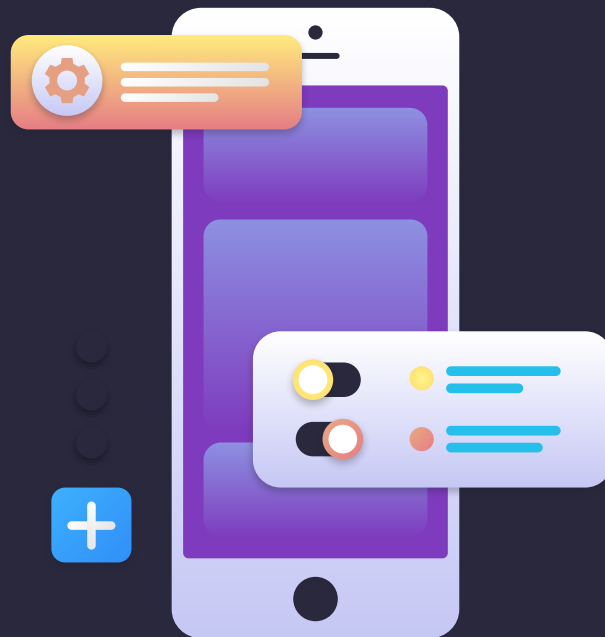


# How to write a Function

Function
By Declaration
By Expression
As Object
Autoexecutable
Closure
Callback
Anonymous
Arrow



# /06 /DATA STRUCTURE



# Arrays

Array is a collection or grouping of elements in a single variable, each of them located by the position it occupies in the array.

Definition



```
let array = new Array(e1, e2, ...);  
let array = [e1, e2, ...];
```

Arrays in js can be mixed (Contain different types of data)

# Objects

The object class represents one of the javascript data types. It is used to store a collection of defined data and more complex entities.

## Definition



```
const cat = {  
  name: 'Gara',  
  age: 4,  
  color: 'black and white',  
  sex: 'female'  
};  
  
const cat = new Object({ name: 'Gara',  
  age: 4,  
  color: 'black and white',  
  sex: 'female'});
```



**THANK FOR  
YOUR  
ATTENTION!**

