**JavaScript**

Complete Programming Guide

**From Basics to Advanced Concepts**

***A comprehensive guide covering data types, functions, OOP, and modern JavaScript features***

**Author:** Mohamed Ebrahim Awod

### June 8, 2025

**Contents**

1. [Data Types in JavaScript 3](#_TOC_250037)
2. [Variables and Declarations 3](#_TOC_250036)
3. [String Operations 3](#_TOC_250035)
   1. [String Concatenation 3](#_TOC_250034)
   2. [Template Literals (ES6) 4](#_TOC_250033)
4. [Operators and Arithmetic 4](#_TOC_250032)
   1. [Remainder and Comparison Operators 4](#_TOC_250031)
5. [Number and String Methods 5](#_TOC_250030)
   1. [Number Methods 5](#_TOC_250029)
   2. [String Methods 5](#_TOC_250028)
   3. [Method Chaining 6](#_TOC_250027)
6. [Control Flow 6](#_TOC_250026)
   1. [IF Statements and Nested Conditions 6](#_TOC_250025)
   2. [Logical Operators 6](#_TOC_250024)
   3. [Switch Statements 7](#_TOC_250023)
7. [Loops and Iteration 7](#_TOC_250022)
   1. [For Loops 7](#_TOC_250021)
   2. [While Loops 7](#_TOC_250020)
   3. [Loop Control 8](#_TOC_250019)
8. [Arrays and Collections 8](#_TOC_250018)
   1. [Array Basics 8](#_TOC_250017)
   2. [Array Methods 9](#_TOC_250016)
9. [Objects and Data Structures 9](#_TOC_250015)
   1. [Object Creation and Access 9](#_TOC_250014)
   2. The this Keyword 10
10. [Functions 11](#_TOC_250013)
    1. [Basic Functions 11](#_TOC_250012)
    2. [Rest Parameters and Default Values 11](#_TOC_250011)
    3. [Arrow Functions (ES6) 12](#_TOC_250010)
11. [Higher Order Functions 12](#_TOC_250009)
12. [Modern JavaScript Collections 13](#_TOC_250008)
    1. [Set Collections 13](#_TOC_250007)
    2. [Map Collections 13](#_TOC_250006)
13. [Object-Oriented Programming 13](#_TOC_250005)
    1. [Class Definition 14](#_TOC_250004)
    2. [Inheritance 14](#_TOC_250003)
    3. [Private Properties 15](#_TOC_250002)
14. [Advanced Object Features 15](#_TOC_250001)
15. [Conclusion 16](#_TOC_250000)

# Data Types in JavaScript



**Understanding JavaScript Data Types**

JavaScript has several built-in data types that are essential for programming.

#### JavaScript provides several primitive data types and one complex data type (objects). Un- derstanding these is crucial for effective programming.

1

// Basic data types

console . log( typeof (" Hello Mohamed ")); // string console . log( typeof (1234 )); //

console . log( typeof ("1234 ")); //

console . log( typeof( true )); // console . log( typeof ({ name : " Mohamed "})); //

console . log( typeof ([])); //

number

string boolean object object

// Array checking

console . log ([] instanceof Array ); // true console . log (" Ebrahim " instanceof Array ); // false

2

3

4

5

6

7

8

9

10

11

#### Listing 1: Data Type Examples

**Important Note:** Arrays in JavaScript are actually objects, which is why typeof [] returns "object". Use instanceof Array or Array.isArray() to check for arrays specifi- cally.

# Variables and Declarations



**Variable Declaration Patterns**

Learn the different ways to declare and initialize variables in JavaScript.

1

// Multiple variable declaration in one line var name = " Mohamed Ebrahim Awod ",

age = 20 ,

job = " Student ";

console . log( name ); // Mohamed Ebrahim Awod console . log( age); // 20

console . log( job ); // Student

2

3

4

5

6

7

8

#### Listing 2: Variable Declarations

# String Operations

## String Concatenation

1

var firstName = " Mohamed "; var second Name = " Ebrahim ";

console . log( firstName + second Name ); // Mohamed Ebrahim

2

3

4

console . log (" My Name Is \ n Ahmed "); // With line break

#### Listing 3: Traditional String Concatenation

## Template Literals (ES6)

1

let nickname = " Ebrahim "; console . log(‘ My name Is Mohamed ${ nickname }‘);

// Multi - line string with variable interpolation

2

3

4

#### Listing 4: Modern Template Strings

**ES6 Feature:** Template literals use backticks (‘) instead of quotes and allow embedded expressions with ${expression}.

# Operators and Arithmetic



**Mathematical and Assignment Operators**

Master JavaScript operators for calculations and assignments.

1

// Arithmetic operators

console . log (10 + 20);

console . log (10 + "20");

// 30

// "1020 " ( string concatenation )

console . log( typeof (10 + "20"));// string

console . log (20 - 10); // 10

console . log (10 - "10"); // 0 ( automatic type conversion )

// Increment/ Decrement

let a = 10; a++;

console . log( a); a--;

console . log( a);

// a = a + 1

// 11

// a = a - 1

// 10

// Assignment operators

let num = 50; num += 50;

console . log( num ); num -= 20; console . log( num );

// num = num + 50

// 100

// num = num - 20

// 80

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

#### Listing 5: Operator Examples

## Remainder and Comparison Operators

1

// Remainder ( Modulo ) operator console . log (8 / 2); // 4

console . log (8 % 2); // 0

console . log (9 / 2); // 4.5

console . log (9 % 2); // 1

2

3

4

5

6

// Comparison operators

console . log (8 == "8");

console . log (8 === "8");

// true ( loose equality )

// false ( strict equality )

7

8

9

#### Listing 6: Advanced Operators

# Number and String Methods



**Built-in Methods for Data Manipulation**

Explore powerful built-in methods for numbers and strings.

## Number Methods

1

let check = 100;

console . log( check ); console . log( check . to String ());

// 100 ( number)

//

"100" ( string )

let number = 11 .1111234 ;

console . log( number. to Fixed (2)); console . log( number. to Fixed ());

// "11 .11 "

// "11"

let copy Number = 11.5;

console . log( copy Number. to Fixed ());

// "12" ( rounds up)

let string = "100 ";

console . log( parse Int( string ));

// 100

let id = 100;

console . log( Number. isNaN ( id));

// false

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

#### Listing 7: Number Manipulation Methods

## String Methods

1

const hello = " Hello World ";

console . log( hello . charAt (0)); // " H"

console . log( hello . length ); // 11

console . log( hello . indexOf (" l")); // 2

console . log( hello . lastIndex Of (" l")); // 9 console . log( hello . repeat (2)); // " Hello World Hello World "

let pop = " look out Ị ";

console . log( pop . trim ()); // " look out Ị"

// Advanced string methods let change1 = " Hello world ";

console . log( change1 . slice (0 , 4)); // " Hell" console . log( change1 . slice (0 , 5)); // " Hello " console . log( change1 . split (" ")); // [" Hello ", " world "] console . log( change1 . substring (0 , 5)); // " Hello "

let change2 = " Hello Hello Hello World ";

console . log( change2 . replace (" Hello ", " Hi"));

// " Hi Hello Hello World "

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

console . log( change2 . replace All (" Hello ", " Hi")); // " Hi Hi Hi World "

#### Listing 8: String Manipulation Methods

## Method Chaining

1

let chaining = " Hello World "; console . log( chaining . slice (0 , 5)

. replace (" Hello ", " Hi")

. startsWith (" H")); // true

2

3

4

#### Listing 9: Chaining Multiple Methods

# Control Flow



**Conditional Statements and Decision Making**

Learn how to control program flow with conditions and loops.

## IF Statements and Nested Conditions

1

let userName = " Mohamed "; var role = " Admin ";

if ( userName == " Mohamed ") { if ( role == " Admin ") {

console . log (" Hello " + userName ); console . log (" You ’ re Admin ");

}

} else {

console . log (" Thank you ");

}

// Ternary operator ( shorthand )

console . log( userName == " Mohamed " ? " Hello " + userName : " Thank You ");

2

3

4

5

6

7

8

9

10

11

12

13

14

#### Listing 10: Conditional Logic

## Logical Operators

1

var checks = " Mohamed ";

console . log( checks == " Mohamed " || 0); console . log( checks == " Ahmed " || 0);

// true

// 0

var nullish 1 = " Mohamed ";

console . log( nullish 1 == " Mohamed " && 0); // 0

var nullish 2 = 0;

console . log( nullish 2 == " Mohamed " && 100); // false

2

3

4

5

6

7

8

9

#### Listing 11: Logical Operations

## Switch Statements

1

var country Code = " Eg ";

switch ( country Code ) {

case " Eg ":

console . log (" Welcome break ;

case " Br":

console . log (" Welcome break ;

case " USA ":

console . log (" Welcome break ;

case " Ca ":

console . log (" Welcome break ;

default:

To

Egypt ");

To

Brazil ");

To

America ");

To

Canada ");

console . log (" Thank you for waiting ");

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

#### Listing 12: Switch Case Example

# Loops and Iteration



**Repetitive Execution Patterns**

Master different types of loops for efficient code repetition.

## For Loops

1

// Basic for loop (0 to 10)

for ( var i = 0; i <= 10; i++) { console . log( i);

}

// Reverse for loop (10 to 1) for ( var i = 10; i > 0; i--) {

console . log( i);

}

// Nested for loop

for ( var i = 0; i <= 5; i++) { console . log (" i = ", i);

for ( var k = 0; k < 3; k ++) { console . log(’ k = ’, k);

}

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

#### Listing 13: For Loop Variations

## While Loops

1

// While loop var n = 0;

while ( n <= 5) { console . log( n); n ++;

}

// Do - while loop var j = 5;

do { console . log( j); j++;

} while ( j <= 6);

2

3

4

5

6

7

8

9

10

11

12

13

#### Listing 14: While Loop Examples

## Loop Control

1

// Break statement

for ( var i = 0; i < 10; i++) { console . log( i);

if ( i == 5) break ;

}

// Continue statement

for ( var i = 0; i < 10; i++) { if ( i == 5) continue ; console . log( i);

}

// Labeled statements

main For: for ( var i = 0; i < 5; i++) { console . log (" I = ", i);

nested For: for ( var n = 0; n < 3; n ++) { console . log (" N = ", n);

if ( n == 1) break main For;

}

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

#### Listing 15: Break

# Arrays and Collections



**Working with Arrays and Data Collections**

Learn comprehensive array manipulation and methods.

## Array Basics

1

let array = [" Mohamed ", " Ahmed ", 1 , false , [1 , 5 , 6], " Basma "]; console . log( array );

console . log( array [3]); // false

array [3] = " Mohamed "; // change the value console . log( array [4 ][0 ]); // 1 ( nested array access)

2

3

4

5

#### Listing 16: Array Creation and Access

## Array Methods

1

let arr = [1 , 2 , 3];

console . log( Array . isArray ( arr));

// true

// Adding / removing elements

arr. push (4); console . log( arr); arr. pop (); console . log( arr);

// Add to end

// [1 , 2 , 3 , 4]

// Remove from end

// [1 , 2 , 3]

let arrays = [" string ", false , {}];

arrays. unshift (0 , 1 , true ); console . log( arrays); arrays. shift (); console . log( arrays);

// Add to beginning

// [0 , 1 , true , " string ", false , {}]

// Remove from beginning

// [1 , true , " string ", false , {}]

// Search methods

let tests = [" Ahmed ", " Omar", " Wael", " Ahmed ", " Ahmed "]; console . log( tests. includes (" Omar ")); // true console . log( tests. indexOf (" Ahmed ")); // 0

console . log( tests. lastIndex Of (" Ahmed "));// 4

// Splice method

tests. splice (2 , 1 , " Mohamed "); Mohamed "

console . log( tests); " Ahmed "]

// Remove 1 element at index 2 , add "

// [" Ahmed ", " Omar", " Mohamed ", " Ahmed ",

// Joining arrays

let array1 = [1 , 2 , 3];

let array2 = [4 , 5 , 6]; console . log( array1 . concat( array2 )); console . log( array1 . join (" "));

// [1 , 2 , 3 , 4 , 5 , 6]

// "1 2 3"

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

#### Listing 17: Essential Array Methods

# Objects and Data Structures



**Object-Oriented Programming Fundamentals**

Understand objects, properties, methods, and the this keyword.

## Object Creation and Access

1

var user = {

// Properties name : " Mohamed ", age: 30 ,

password : 123456 ,

2

3

4

5

6

7

// Methods

getPassword : function () { console . log( this. password );

}

};

console . log( user. name ); user. getPassword ();

// Dynamic property access var variable = " country "; var object = {

name : " Omar", age: 30 , country : " Egypt"

};

console . log( object[ variable ]);

// " Egypt"

// Nested objects

var nested Object = { name : " Omar", age: 30 ,

children : [" Ahmed ", " Mohamed ", " Sayed "], nested : {

first: 1 ,

second : 2 ,

third : 3

}

};

console . log( nested Object. nested . first); // 1

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

#### Listing 18: Object Fundamentals

* 1. **The** this **Keyword**

1

var globalAge = 50;

var

objectWith This = {

age: 100 ,

getAge : function () { console . log( this. age);

},

getGlobalAge : function () { console . log( globalAge );

}

// refers to object ’ s age

// refers to global age

};

objectWith This . getAge (); // 100

objectWith This . getGlobalAge (); // 50

2

3

4

5

6

7

8

9

10

11

12

13

14

#### Listing 19: Understanding ’this’ Context

# Functions



**Function Declarations and Advanced Concepts**

Master function creation, parameters, and advanced features.

## Basic Functions

1

function isTall( name ) {

if ( name . length > 6) { console . log (" String is tall ");

} else {

console . log (" String is short ");

}

}

isTall (" Mohamed ");

isTall (" Ahmed ");

// String is tall

// String is short

function say Hello ( name ) {

return " Hello " + name ;

}

// Using function return values

if ( say Hello (" Mohamed ") == " Hello Mohamed ") { console . log (" Yes");

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

#### Listing 20: Function Fundamentals

## Rest Parameters and Default Values

1

// Rest parameters

function welcome ( name , age , ... values) { console . log (" Name : " + name + ", Age: " + age); console . log (" Other values :", ... values);

}

welcome (" Mohamed ", 21 , " Ahmed ", false , {}, [], 5);

// Default parameters and rest

function sum ( a = 0 , b = 0 , c = 0 , ... rest) {

let result = Number( a) + Number( b) + Number( c); for ( var i = 0; i < rest. length ; i++) {

result = result + rest[ i];

}

return result;

}

console . log( sum ("1", 2 , 3 , 4 , 5)); // 15

// Nested functions function parent () {

function nested () {

return " Hi From Nested ";

}

console . log( nested ()); return " Hi From Parent ";

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

}

console . log( parent ());

26

#### Listing 21: Advanced Function Features

## Arrow Functions (ES6)

1

// Arrow function with block body let say Welcome = ( name ) => {

return " Welcome " + name ;

};

console . log( say Welcome (" Mohamed "));

// Arrow function with expression body let sums = (a, b) => a + b;

console . log( sums(1 , 2)); // 3

2

3

4

5

6

7

8

9

#### Listing 22: Modern Arrow Function Syntax

# Higher Order Functions



**Functional Programming with Arrays**

Learn powerful array methods for functional programming.

1

let numbers = [1 , 2 , 3 , 4 , 5];

// Map - transform each element

let map Result = numbers. map ( function ( value , index ) { return value \* 2;

});

console . log (" Map result :", map Result); // [2 , 4 , 6 , 8 , 10]

// Filter - select elements based on condition

let names = [" Mohamed ", " Ahmed ", " Tarak ", " Basma ", " Israa ", " Malak "]; let filterResult = names. filter (( value ) => {

return value . length > 6;

});

console . log (" Filter result :", filterResult); // [" Mohamed "]

// Reduce - accumulate values var arr3 = [1 , 2 , 3 , 4 , 5];

var reduce Result = arr3 . reduce ( function ( prevValue , currentValue , index ) { console . log (" Prev :", prevValue , " Current :", currentValue , " Index :", index ); return prev Value + currentValue ;

});

console . log (" Reduce result :", reduce Result); // 15

// Reduce with initial value

var reduce Result2 = arr3 . reduce ( function ( prevValue , currentValue ) { return prev Value + currentValue ;

}, 15);

console . log (" Reduce with initial value :", reduce Result2 ); // 30

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

#### Listing 23: Map

# Modern JavaScript Collections



**Set and Map Data Structures**

Explore ES6 Set and Map collections for advanced data handling.

## Set Collections

1

var array With Duplicates = [1 , 2 , 3 , 4 , 4 , 5 , 5]; console . log (" Array :", array With Duplicates );

var sets = new Set([1 , 2 , 3 , 4 , 4 , 5 , 5]);

console . log (" Set:", sets); // Set removes duplicates console . log (" Set [0]:" , sets [0]); // undefined ( can ’ t access by index )

// Set methods sets. add (6);

console . log (" After add (6):", sets); console . log (" Has 3:", sets. has (3)); // true console . log (" Size :", sets. size ); // 6 sets. delete (6);

console . log (" After delete (6):", sets);

2

3

4

5

6

7

8

9

10

11

12

13

14

#### Listing 24: Working with Sets

## Map Collections

1

// Traditional object var person = {

name : " Mohamed ", age: 30 , country : " Egypt"

};

// Map can use any data type as keys var person Map = new Map (); person Map . set(" name ", " Mohamed "); person Map . set( true , 30); person Map . set(" country ", " Egypt "); person Map . set(1 , 2);

console . log (" Map :"); person Map . forEach (( value , key ) => {

console . log( key , ":", value );

});

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

#### Listing 25: Working with Maps

# Object-Oriented Programming



**Classes and Inheritance in JavaScript**

Master modern JavaScript classes and object-oriented programming.

## Class Definition

1

class User {

constructor( name , password , email) { this. name = name ;

this. password = password ; this. email = email;

}

getPassword () {

return this. password ;

}

getInfo () {

return ‘ Name : ${ this. name }, Email: ${ this. email}‘;

}

}

let user1 = new User (" Mohamed ", 123 , " mohamed@ example . com "); let user2 = new User (" Ahmed ", 456 , " ahmed@ example . com ");

let user3 = new User (" Hany ", 789 , " hany@ example . com ");

console . log( user1 . getInfo ()); console . log( user2 . getInfo ()); console . log( user3 . getInfo ());

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

#### Listing 26: Creating Classes

## Inheritance

1

class Parent {

constructor( name , id , password ) { this. name = name ;

this. id = id; this. password = password ;

}

getID () {

return ‘ Your ID is: ${ this. id }‘;

}

getInfo () {

return ‘ Name : ${ this. name }, ID: ${ this. id }‘;

}

}

class Child extends Parent { constructor( name , id , password , age) {

super( name , id , password ); // Call parent constructor this. age = age;

}

getAge () {

return ‘ Your age is: ${ this. age }‘;

}

getFullInfo () {

return ‘${ this. getInfo ()}, Age: ${ this. age }‘;

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

}

var parent = new Parent (" Youssef", 1 , 15432 ); var child = new Child (" Fady ", 2 , 43156 , 20);

console . log (" Parent info :", parent. getInfo ()); console . log (" Child info :", child . getFullInfo ()); console . log (" Child ID :", child . getID ()); // Inherited method

31

32

33

34

35

36

37

#### Listing 27: Class Inheritance

## Private Properties

1

class UserWith Private {

// Private property ( using # syntax ) # password ;

constructor( id , name , password , age) { this. id = id;

this. name = name ; this .# password = password ; this. age = age;

}

// Method to change password with validation change Password ( oldPass , new Pass) {

if ( old Pass === this .# password ) { this .# password = new Pass;

console . log (" Password changed successfully Ị"); return true ;

} else {

console . log (" Incorrect old password Ị"); return false ;

}

}

// Method to verify password verify Password ( password ) {

return password === this .# password ;

}

}

var user5 = new UserWith Private (1 , " Mohamed ", 123 , 19); console . log (" Direct password access :", user5 . password ); // undefined

console . log (" Password verification :", user5 . verify Password (123)); // true user5 . change Password (123 , 456); // Password changed successfully Ị

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

#### Listing 28: Private Class Members

# Advanced Object Features



**Object Property Descriptors and Meta Data**

Learn advanced object property control and configuration.

1

var object = {

name : " Mohamed ", age: 30 ,

};

Object. define Property ( object , " country ", {

writable : true ,

enumerable : true , configurable : true , value : " Egypt",

// Controls if the property

// Controls if the property

// Controls if the property

value can be changed

shows up in loops can be deleted

// The actual value of the property

});

console . log (" Can delete :", delete object. country ); // true console . log( object);

2

3

4

5

6

7

8

9

10

11

12

13

14

#### Listing 29: Object Property Descriptors

**Property Descriptors:**

* **writable**: Controls if the property value can be changed
* **enumerable**: Controls if the property appears in for...in loops
* **configurable**: Controls if the property can be deleted or its attributes changed
* **value**: The actual value of the property

# Conclusion



**Congratulations!**

You’ve completed the comprehensive JavaScript guide covering all fundamental and ad- vanced concepts!

#### This guide has covered:

#### Data types and variables Operators and expressions Control flow and loops

#### Functions and arrow functions

#### Arrays and higher-order functions Objects and the this keyword

#### Modern ES6+ features (Sets, Maps, Classes) Object-oriented programming

#### Inheritance and private properties Advanced object features

 **Happy Coding with JavaScript! **