Table of Content

[Data Types 3](#_Toc115695818)

[Example 3](#_Toc115695819)

[Null vs Undefined 3](#_Toc115695820)

[Variable 4](#_Toc115695821)

[Example 4](#_Toc115695822)

[Variable – (var vs let vs const) 4](#_Toc115695823)

[⚡ Initialization 4](#_Toc115695824)

[⚡ Re-Assignment 4](#_Toc115695825)

[⚡ Re-Declaration 5](#_Toc115695826)

[⚡ Scope 5](#_Toc115695827)

[Loop 6](#_Toc115695828)

[⭐ For loop 6](#_Toc115695829)

[Example 7](#_Toc115695830)

[⭐ For in loop 7](#_Toc115695831)

[Example 7](#_Toc115695832)

[⭐ forEach loop 8](#_Toc115695833)

[Example 8](#_Toc115695834)

[⭐ For In loop 9](#_Toc115695835)

[Example 9](#_Toc115695836)

[⭐ While In loop 9](#_Toc115695837)

[Example 9](#_Toc115695838)

[⭐ doWhile In loop 9](#_Toc115695839)

[Example 10](#_Toc115695840)

[Events 10](#_Toc115695841)

[Example 10](#_Toc115695842)

[Operators(comp arth assig logical) 11](#_Toc115695843)

[Aritmetic Example 11](#_Toc115695844)

[Assignmet Example 12](#_Toc115695845)

[Comparison Example 13](#_Toc115695846)

[Logical Example 14](#_Toc115695847)

[Bitwise Example 15](#_Toc115695848)

# Data Types

In computer science and computer programming, a data type or simply type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data.

## Example

* Number (Integer/ Float)
* String (string/ Character)
* Object
* Array
* Boolean
* undefined
* null

1. var age = 18;                           // number
2. var name = "Jane";                      // string
3. var name = "J";                      // string
4. var name = {first:"Jane", last:"Doe"};  // object
5. var sheets = ["HTML","CSS","JS"];       // array
6. var truth = true;                      // boolean
7. var truth = false;                      // boolean
8. var a;                        // undefined
9. var a = undefined;                  // undefined
10. var a = null;                           // value null

Type of will return the type of value in its parameters.

1. typeof(variable\_name)

## Null vs Undefined

1. typeof null          // "object" (not "null" for legacy reasons)
2. typeof undefined     // "undefined"
3. null === undefined   // false
4. null  == undefined   // true
5. null === null        // true
6. null  == null        // true
7. !null                // true
8. isNaN(1 + null)      // false
9. isNaN(1 + undefined) // true

# Variable

Variables are containers for storing data (storing data values).

There are Four Ways to Declare a JavaScript Variable:

* Using var
* Using let
* Using const
* Using nothing

## Example

* var x = 5;
* let y = 6;
* const pi = 3.14;
* z=20;

## Variable – (var vs let vs const)

There is a slight difference between var let and const which is discussed following

## ⚡ Initialization

var a; //✅

let b; //✅

const c; //❌ const must have an initial value

## ⚡ Re-Assignment

var d = 1;

d = 10; //✅

let e = 2;

e = 20; //✅

const f = 3;

f = 30 //❌ const value is always constant and cannot be reassigned

## ⚡ Re-Declaration

var g = 1;

var g = 10; //✅

let h = 2;

h = 2; //✅

let h = 20; //❌ must use variable name only to reassign, cannot be reassigned

const i = 3;

const i = 30 //❌ cannot be reassigned

## ⚡ Scope

Global Scope ==> Any variable that is declared outside any of the function

Function Scope(var) ==> Can be accessed inside the function it was declared and its child

Block Scope (let and const) ==> Can be accessed inside curly brackets where it was declared and its child

function myFunction(){

  var fname = 'Syed';

  let mName = 'Muhammad';

  const lname = 'Ahmad';

  console.log(fname);       // Syed

  console.log(mName);       // Muhammad

  console.log(lname);       // Ahmad

  if(true){

    console.log(fname);     // Syed

    console.log(mName);     // Muhammad

    console.log(lname);     // Ahmad

      var myAge1 = 13;

      let myAge2 = 15;

      const myAge3 = 14;

      console.log(myAge1);  // 13

      console.log(myAge2);  // 15

      console.log(myAge3);  // 14

  }

  console.log(fname);       // Syed

  console.log(mName);       // Muhammad

  console.log(lname);       // Ahmad

  console.log(myAge1);      //13

  console.log(myAge2);      //not defined

  console.log(myAge3);      //not defined

};

myFunction();

# Loop

A loop is a sequence of instruction s that is continually repeated until a certain condition is reached. JavaScript supports different kinds of loops:

for - loops through a block of code a number of times

for/in - loops through the properties of an object and arrays

forEach - loops through the properties of an arrays

for/of - loops through the values of an iterable object

while - loops through a block of code while a specified condition is true

do/while - also loops through a block of code while a specified condition is true

## ⭐ For loop

Loops through a block of code a number of times

## Example

for (let i = 0; i < 5; i++) {

  text += "The number is " + i + "<br>";

}

for (let i = 0, len = cars.length, text = ""; i < len; i++) {

  text += cars[i] + "<br>";

}

let i = 2;

let len = cars.length;

let text = "";

for (; i < len; i++) {

  text += cars[i] + "<br>";

}

let i = 0;

let len = cars.length;

let text = "";

for (; i < len; ) {

  text += cars[i] + "<br>";

  i++;

}

## ⭐ For in loop

Loops through the properties of an object and arrays.(Don’t use for in on array if order is important)

## Example

for (key in object) {

  // code block to be executed

}

const person = {fname:"John", lname:"Doe", age:25};

let text = "";

for (let x in person) {

  text += person[x];

}

for (variable in array) {

  code

}

const numbers = [45, 4, 9, 16, 25];

let txt = "";

for (let x in numbers) {

  txt += numbers[x];

}

## ⭐ forEach loop

Loops through the properties of an arrays.

## Example

const fruits = ['mango', 'papaya', 'pineapple', 'apple'];

// Iterate over fruits below

// Normal way

fruits.forEach(function(fruit){

  console.log('I want to eat a ' + fruit)

});

const numbers = [45, 4, 9, 16, 25];

let txt = "";

numbers.forEach(myFunction);

function myFunction(value, index, array) {

  txt += value;

}

## ⭐ For In loop

The JavaScript for of statement loops through the values of an iterable object. It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

## Example

const cars = ["BMW", "Volvo", "Mini"];

let text = "";

for (let x of cars) {

  text += x;

}

let language = "JavaScript";

let text = "";

for (let x of language) {

text += x;

}

## ⭐ While In loop

while - loops through a block of code while a specified condition is true

## Example

let i = 0;

while (i < 10) {

  text += "<br>The number is " + i;

  i++;

}

## ⭐ doWhile In loop

do/while - also loops through a block of code while a specified condition is true

## Example

do {

  text += "The number is " + i;

  i++;

}

while (i < 10);

# Events

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

## Example

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

# Operators (comp arth assig logical)

There are different types of JavaScript operators:

* Aritmetic Operators
* Assignment Operators
* Comparison Operators
* Logical Operators
* Conditional Operators
* Type Operators

## Aritmetic Example

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| \*\* | Exponentiation ([ES2016](https://www.w3schools.com/js/js_2016.asp)) |
| / | Division |
| % | Modulus (Division Remainder) |
| ++ | Increment |
| -- | Decrement |

## Assignmet Example

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |

## Comparison Example

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| ? | ternary operator |

## Logical Example

|  |  |
| --- | --- |
| && | logical and |
| || | logical or |
| ! | logical not |

## Bitwise Example

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Same as** | **Result** | **Decimal** |
| & | AND | 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | left shift | 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | right shift | 5 >> 1 | 0101 >> 1 | 0010 | 2 |
| >>> | unsigned right shift | 5 >>> 1 | 0101 >>> 1 | 0010 | 2 |