

**NOTES**

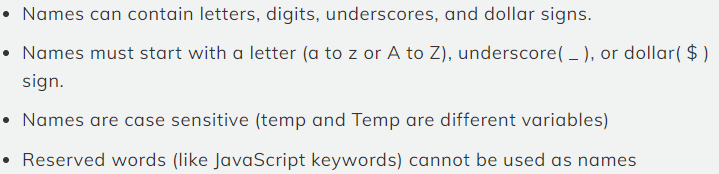
@ashish003

Introduction

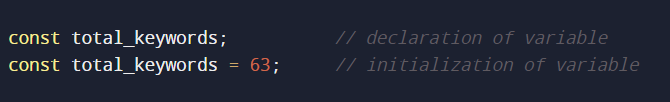
* JavaScript is an interpreted programming language and the programs in this language called scripting language used on the client side and server side.
* It was developed by Brendan Eich in 1995 and became an ECMA standard in 1997.
* Pros: Interpreted language, All browser support, Minimal syntax, Platform Independence
* Cons: weakly typed language, no support of multithreading and multiprocessing (i.e., single-threaded language), read and write of files not allowed.

Variable and Identifiers

* Variables are the name of the memory space where we store our data. which can be any type due to JavaScript's weakly typed language.
* There are four ways to declare variables in JavaScript, automatically, var, let, and const.
* Identifier is the name of the variable or function that we can identify uniquely to access in their scope.

The rules to give a unique name are: 

* Example:



Automatically, var, let, and const

* In JavaScript we can declare variables without using any reserved keyword like var, let, or const this concept is called an automatic declaration.

example: my\_var = 100;

* var is a reserved keyword in JavaScript used for declaring the variables that denote data can vary it same as the automatic declaration of the variable.

example: var my\_var = 100;

Note: The above two ways of declaring variables are not recommended due to of behavior of scope and security issues which we discuss later.

* let is another reserved keyword in JavaScript used for declaring the variable that also denotes data can vary. It is recommended instead of var.

example: let my\_var = 100;

* const is also a reserved keyword in JavaScript used for initializing the variable at the time of declaration and const denotes data (like name, date of birth) cannot vary at any point.

example: const date\_of\_birth = “07/04/1998”;

Note: The above two ways of declaring a variable are always recommended but always try to declare a variable using const if possible, which makes data secure.

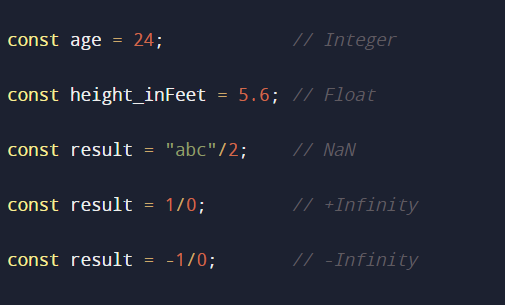
Datatypes

* Datatype means the type of data that we going to store in our variable and due to the dynamic nature of JavaScript it allows us to store any datatype in the variable.
* There are 7 primitive datatypes and 2 non-primitive datatypes.
* Primitive: Number, BigInt, String, Boolean, Undefined, Null, Symbol
* Non-Primitive: Object, function

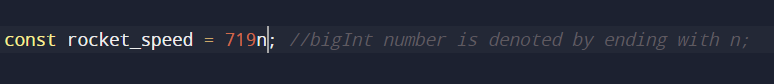
Note: primitive datatypes are immutable and take memory in the stack whereas non-primitive datatypes are mutable and take memory in the heap and reference in the stack.

Primitive datatypes

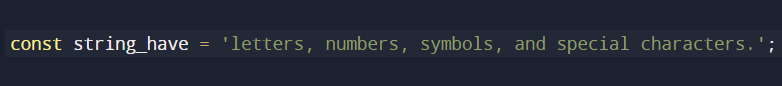
* Number datatype represents a variable whose value is either an integer, float, or JavaScript special numeric value (NaN, +Infinity, -Infinity).



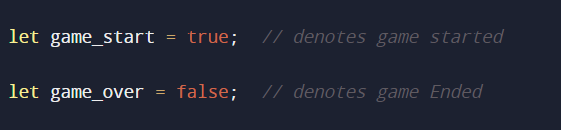
* BigInt datatype is introduced in ECMAScript 2020 and is designed to represent arbitrary precision integers means very large numbers without losing precision.



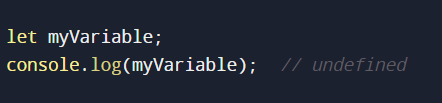
* String datatype represents the sequence of characters inside single quotes, double quotes, or in backticks.



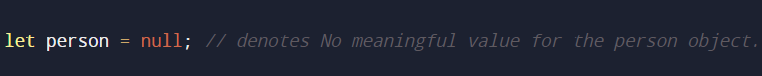
* Boolean datatype represents the two values i.e., true or false. It is used for conditional testing.



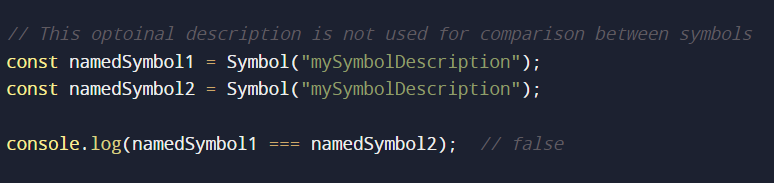
* Undefined datatype that represents the absence of value, and variables that have been declared but not assigned a value automatically have the value ‘undefined’.



* Null is a special value that represents the intentional absence of any object value. Unlike `undefined` which is often the default value of an uninitialized variable, `null` is explicitly assigned to indicate the absence of value.

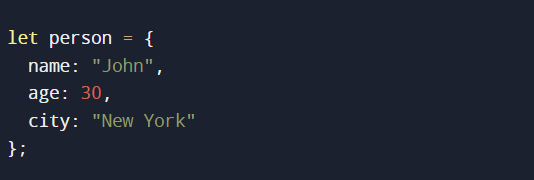


* Symbol datatype is used to provide a way to create unique and immutable identifiers. Symbols are often used to create private or hidden properties on objects and avoid naming collisions in scenarios where string-based property names might clash.

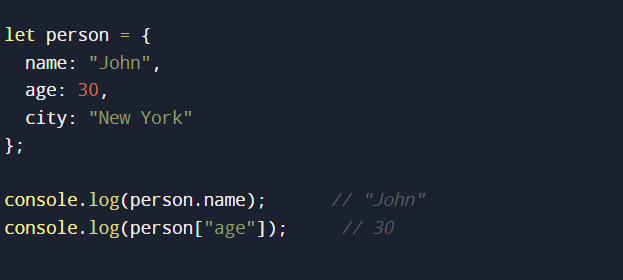


Non-primitive datatypes

* Object datatype is a complex datatype that allows you to group related data and code into a single unit. It represents the data in key-value pairs, where keys are strings or symbols, and the value can be a valid JavaScript datatype, including other objects.



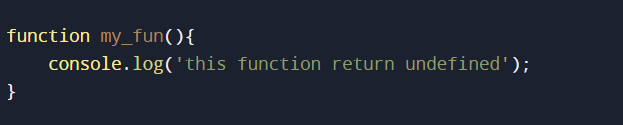
object value can be accessed using dot notation or square bracket notation example:



object datatype is further divided in JavaScript i.e., Array object, Date object, RegExp object, which we discuss later.

* Function is also the type of object and has properties and methods like other objects. functions are used to define reusable blocks of code that can be invoked to solve specific tasks.

when we see the type of `function` using the `typeof` operator shows the string function that is why I consider this function object as a second non-primitive datatype.



Note: the above data types can be verified by using the `typeOf` operator

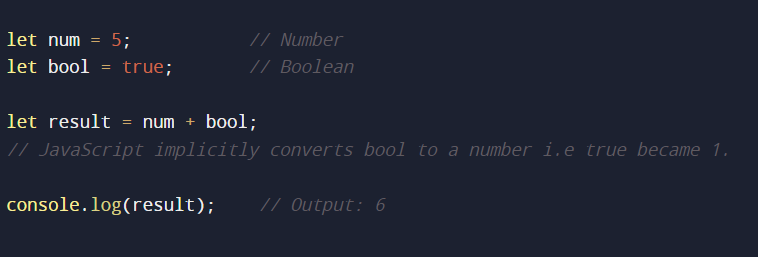
Operators

* Operators are some reserve fundamental symbols used to perform specific mathematical and logical operations on operands.
* There are many operators in JavaScript, and some important ones are on [geeksForGeeks](https://www.geeksforgeeks.org/javascript-operators-reference/)
* Note: Operator precedance

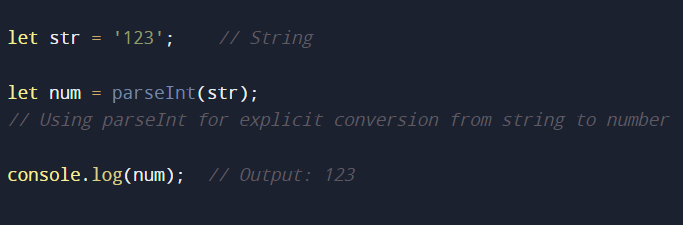
Implicit and Explicit conversion

* Implicit conversion, also known as coercion, happens automatically when JavaScript converts one data type to another without the programmer explicitly requesting the conversion.

This often occurs in situations where different data types are involved in an operation.



* Explicit conversion, also known as type casting, occurs when the programmer explicitly converts a value from one type to another. This is done using built-in functions or methods.

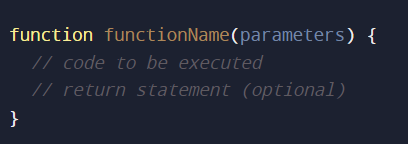


Example

Important datatypes and Operations on them

Functions

* Functions are blocks of reusable code that perform a specific task, here is a basic syntax for creating a function:



function: keyword used to declare a function.

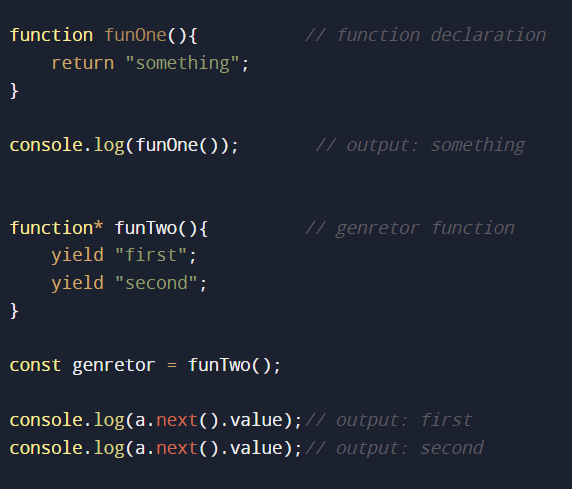
functionName: is the identifier used to call the function later.

parameters: input values that the function can accept.

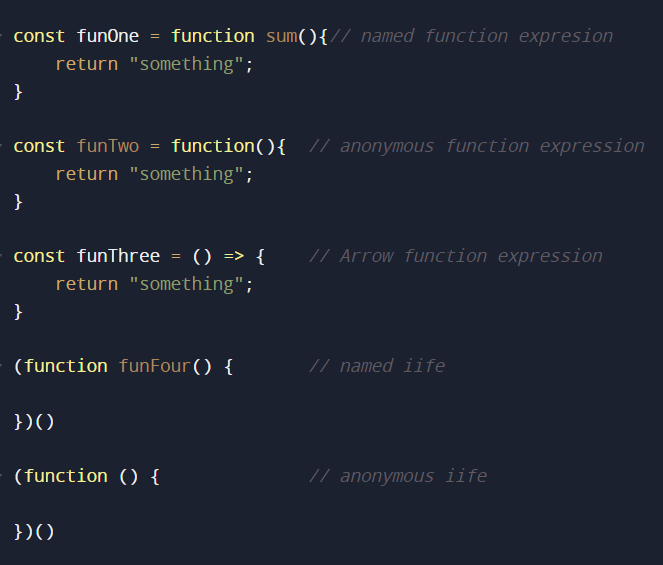
return: keyword used to return the value explicitly by default function return undefined.

Default parameter and arguments array rest operator

* Functions are of two types Function declaration and Function expression.
* Function declaration: if the first word or token of a written function is the function keyword then it is called function declaration. example:



* Function expression: if the first word or token of a written function is not a function keyword then it is called the function expression, it is of three types name function expression, anonymous function expression, and iife (immediately invoked function expression). example:



Higher order functions and call-back function

iife

Scope with variable

Hoisting and Lexical Scoping

Control flow and Boolean

Loops with respect

Closure

Opps

DOM

API and Promises