Table of Content

1. Comments
2. Identifiers
3. Variables (var)
4. let
5. const
6. Operators
7. Data types
8. Arrays
9. Objects
10. Strings
11. Conditional Statements
12. Looping Statements

# Comments

1. **Desc:**

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

1. **Usage:**

Used to prevent execution, when testing alternative code.

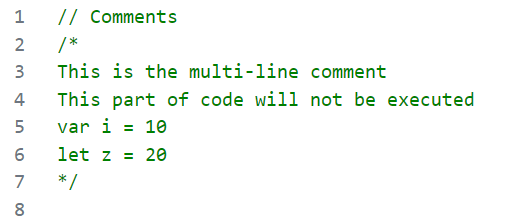
1. **Advantages:**

Makes code more readable.

1. **Disadvantages:**

Disadvantages of comments include potential clutter, outdated information, and redundancy

1. **Code Snippets:**

****

# Identifiers

1. **Desc:**
   * All JavaScript variables must be identified with unique names.
   * These unique names are called identifiers.
   * The general rules for constructing names for variables (unique identifiers) are:
     + Names can contain letters, digits, underscores, and dollar signs.
     + Names must begin with a letter.
     + Names can also begin with $ and \_ (but we will not use it in this tutorial).
     + Names are case sensitive (y and Y are different variables).
     + Reserved words (like JavaScript keywords) cannot be used as names.

Variables(var)

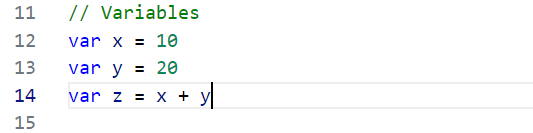
1. **Desc:**

Containers for Storing Data globally

1. **Usage:**

Storing Data

1. **Code Snippets:**



# let

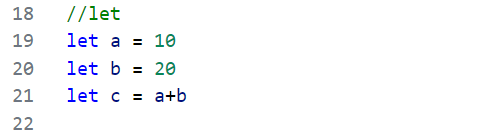
1. **Desc:**

Containers for Storing Data within the block of code.

1. **Usage:**

Used to store data which needs to be accessed within a specific block.

1. **Code Snippets:**



# const

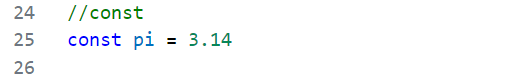
1. **Desc:**

Containers for Storing Data with **no redeclaration** and with **no reassigned** and **has local scope**.

1. **Usage:**

Can be used where Data should not be redeclared at any part of code.

1. **Code Snippets:**



**Note: Difference between let, const and var**



# Operators

1. **Desc:**

Special symbols that perform specific operations on values and variables, allowing you to manipulate data and create expressions.

1. **Usage:**

For performing calculations, comparisons, and logical operations.

1. **Types:**
   1. **Arithmetic Operators**
      1. + Addition
      2. - Subtraction
      3. \* Multiplication
      4. \*\* Exponentiation (ES2016)
      5. / Division
      6. % Modulus (Division Remainder)
      7. ++ Increment
      8. -- Decrement
   2. **Assignment Operators**
      1. =
      2. +=
      3. -=
      4. \*=
      5. /=
      6. %=
      7. \*\*=
   3. **Comparison Operators**
      1. == equal to
      2. === equal value and equal type
      3. != not equal
      4. !== not equal value or not equal type
   4. **Logical Operators**
      1. && logical and
      2. || logical or
      3. ! logical not
   5. **Bitwise Operators**
      1. & AND
      2. | OR
      3. ~ NOT
      4. ^ XOR
      5. << left shift
      6. >> right shift
      7. >>> unsigned right shift
   6. **Ternary Operators**
      1. ? :

# Data types

1. **Desc**:

A classification that determines the kind of data a variable or object can hold.

1. **Usage**:

Tells the compiler or interpreter how the programmer intends to use the data.

1. **Types:**
   1. String
   2. Number
   3. Boolean
   4. Undefined
   5. Null
   6. Array
   7. Object

# Arrays

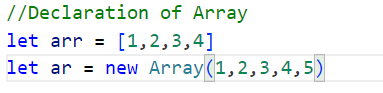
1. **Desc:**

Collection of homogenous and heterogenous data.

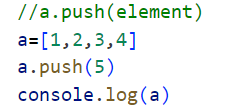
1. **Usage:**

An array can hold many values under a single name, and you can access the values by referring to an index number.

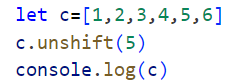
1. **Declaration of Array:**

****

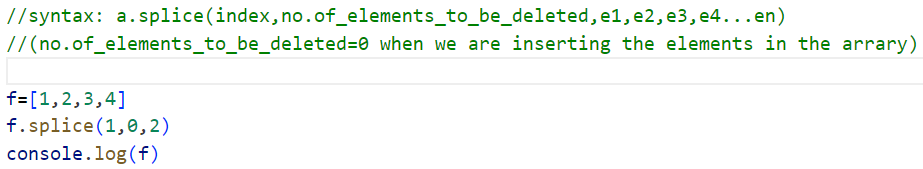
1. **Operations:**
   1. **Adding element at end position**



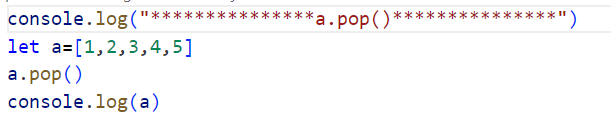
* 1. **Adding element at beginning position**

****

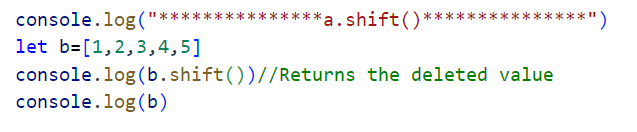
* 1. **Adding element at specific position**

****

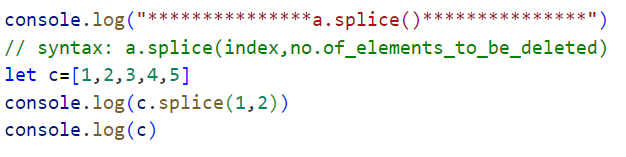
* 1. **Deleting element at end position**

****

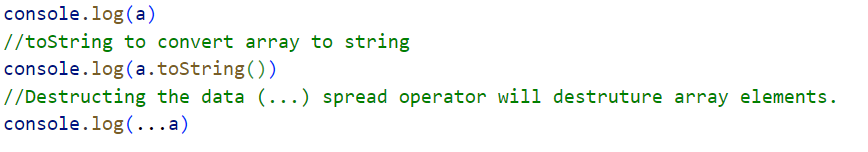
* 1. **Deleting element at beginning position**

****

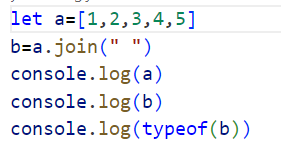
* 1. **Deleting element at specific position**

****

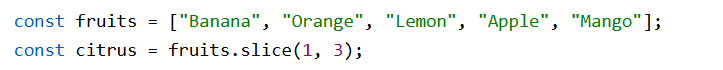
1. **Methods:**
   1. **toString method:**

****

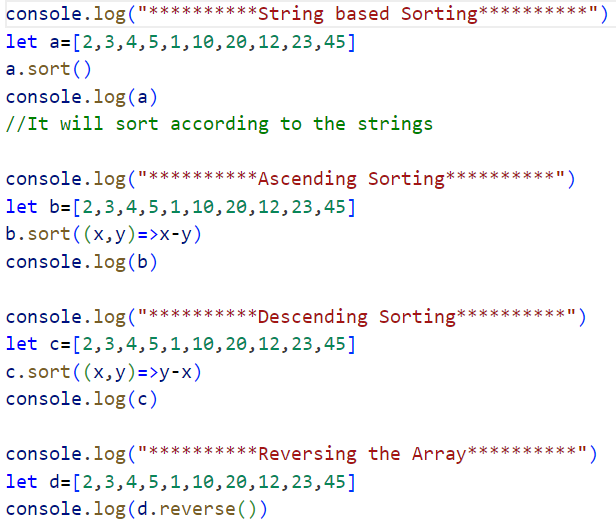
* 1. **join method:**

****

* 1. **slice() method:**

****

* 1. **sort method:**

****

1. **Advantages:**
   1. Arrays offer fast and direct access to elements using their index, which enables efficient data storage and retrieval.
   2. Arrays provide a convenient way to store and access multiple elements of data under a single variable name.
   3. Arrays offer flexibility in managing collections of data.

# Objects

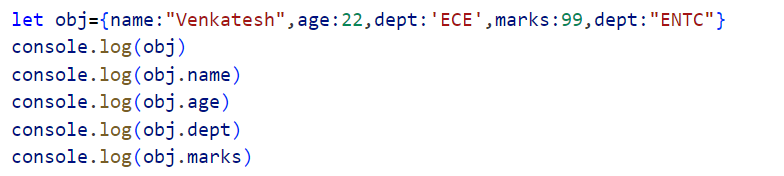
1. **Desc:**

A data type that stores key-value pairs, allowing you to group related data and functions together, acting as a container for properties and methods.

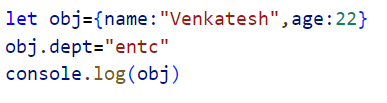
1. **Usage:**

Used to store and organize data as key-value pairs.

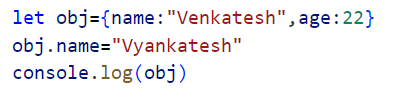
1. **Declaration of Object and accessing the elements of Objects:**

****

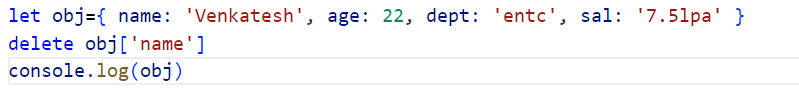
1. **Operations:**
   1. **Adding a property to the Objects:**



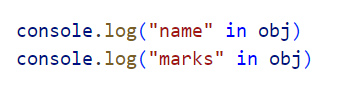
* 1. **Updating the value of property:**

****

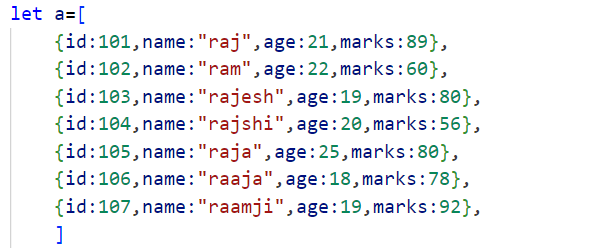
* 1. **Deleting the property:**



* 1. **To check whether the property exists or not**



1. **Array of objects:**
   1. **Declaration of Array of objects:**



1. **Advantages:**
   1. Allow data to be organized in an efficient and manageable way.
   2. Objects Are Efficient.
   3. Objects Can Represent Part-Whole Relationships

# Strings

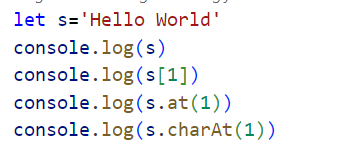
1. **Desc:**

Containers to stores a series of characters.

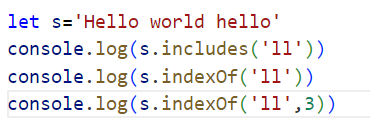
1. **Usage:**

Used to stores a series of characters

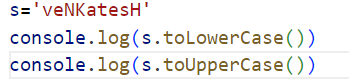
1. **Creating and accessing String:**



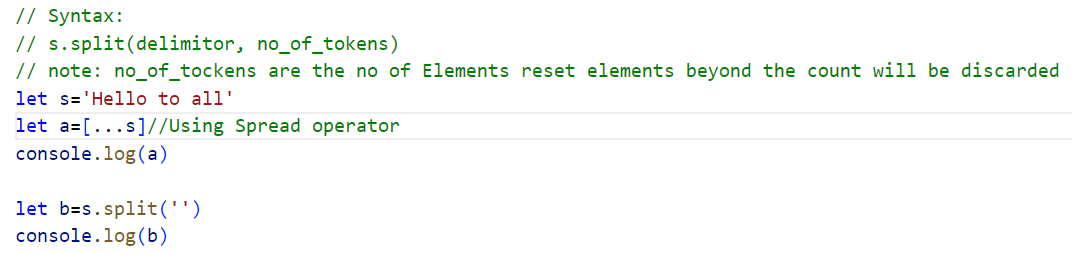
1. **Methods:**
   1. **Searching the index position of the substring:**



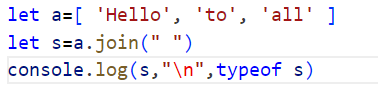
* 1. **toUpperCase & toLowerCase**



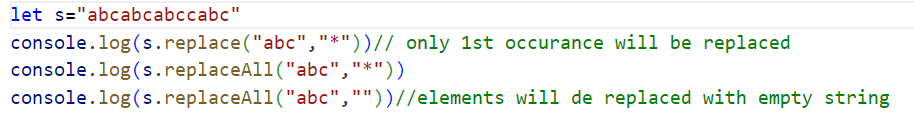
* 1. **Strings to Array**



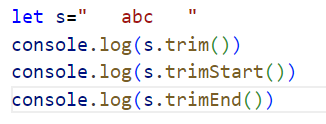
* 1. **Array to String**



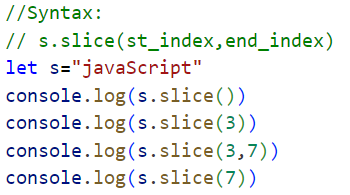
* 1. **Replace**



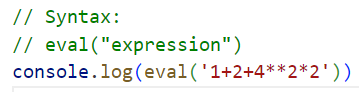
* 1. **Trim**



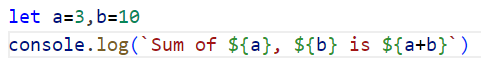
* 1. **Slice**



* 1. **Eval**



* 1. **Template String**



# Conditional Statements

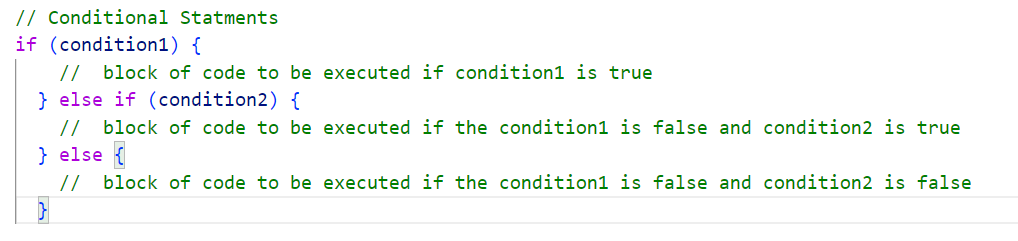
1. **Desc:**

**if, else, and else if,** control the flow of execution by allowing you to execute different blocks of code based on whether a condition is true or false

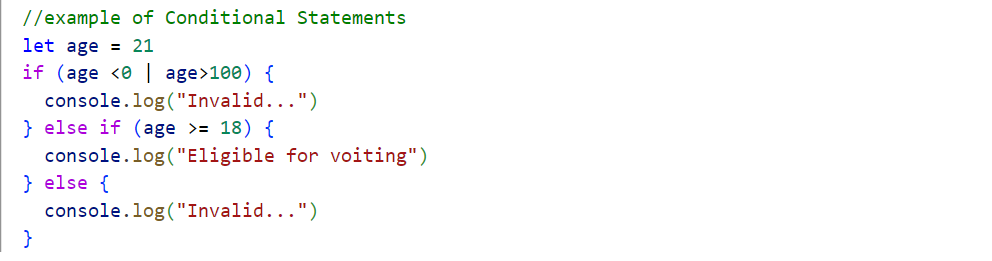
1. **Usage:**

Used to perform different actions based on different conditions.

1. **Syntax:**
2. **Code Snippet:**
   1. **Syntax:**



* 1. **Example:**



# Looping Statements

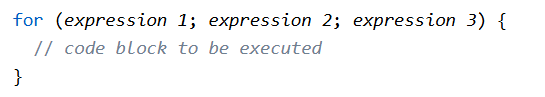
1. **Desc:**

Loops are handy, if you want to run the **same code over and over again**, each time with a different value.

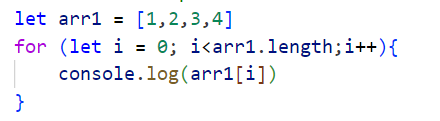
1. **Usage:**

Loops can **execute a block of code a number of times**.

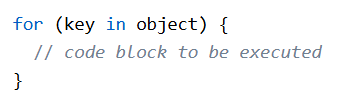
1. **Types:**
   1. **For Loop**
      1. **Normal For Loop**
         1. **Syntax:**

****

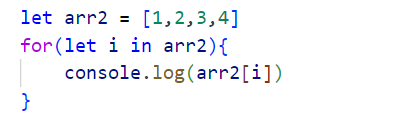
* + - 1. **Example:**

****

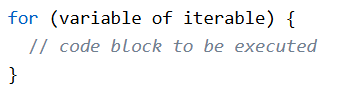
* + 1. **For In**
       1. **Syntax:**

****

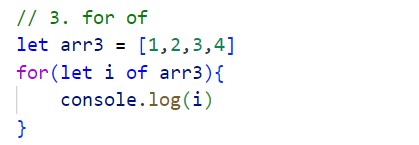
* + - 1. **Example:**

****

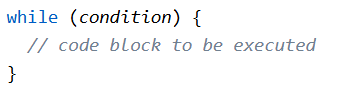
* + 1. **For Of**
       1. **Syntax:**

****

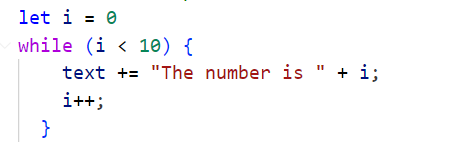
* + - 1. **Example:**

****

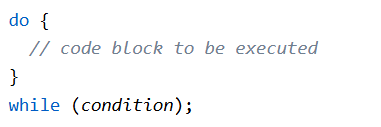
* 1. **While**
     1. **while loop**
        1. **Syntax:**

****

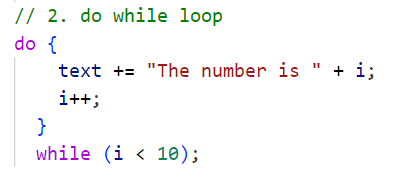
* + - 1. **Example:**

****

* + 1. **do while loop**
       1. **Syntax:**

****

* + - 1. **Example:**

****