**What are variable ?**

n JavaScript, variables act like named storage containers. They allow you to store data values and refer to them using the variable's name throughout your code. This makes your code more readable, maintainable, and reusable

**Declaring Variables:**

* To create a variable, you use keywords like let, const, or var (although var is generally discouraged due to potential scoping issues).
* The keyword is followed by a chosen name for your variable, which must follow naming conventions (starting with a letter, underscore, or dollar sign, and containing letters, numbers, or underscores).
* JavaScript
* let name = "Alice";
* const age = 30;

 **Assigning Values:**

* Once you declare a variable, you can assign a value to it using the assignment operator (=).
* The value can be a number, string, boolean (true/false), array, object, or even another variable.

JavaScript

let message = "Hello, world!";

let isMember = true;

Use code [with caution.](/faq#coding)

 **Using Variables:**

* After assigning a value, you can reference the variable by its name to use the stored data in your code.
* This allows you to perform operations, display information, or manipulate data based on the variable's value.

JavaScript

console.log(name); // Outputs "Alice"

if (isMember) {

console.log("Welcome back!");

}

**Data types**

JavaScript uses data types to categorize the different kinds of data it can work with. These data types define how data is stored and what operations can be performed on it. There are two main categories of data types in JavaScript:

1. **Primitive Data Types:**
   * These are fundamental data types that represent basic values.
   * They are stored directly in memory and are immutable (their values cannot be changed after assignment).
   * Here are the common primitive data types:
     + **Number:** Represents numeric values, including integers (whole numbers) and decimals. (e.g., 42, 3.14)
     + **String:** Represents textual data enclosed in quotes (single or double). (e.g., "Hello", 'World')
     + **Boolean:** Represents logical values, either true or false. Used for conditional statements.
     + **Undefined:** Represents a variable that has been declared but not yet assigned a value.
     + **Null:** Represents the intentional absence of any object value.
     + **Symbol (ES6+):** A unique and immutable identifier (not commonly used).
2. **Non-primitive Data Types:**
   * These are more complex data structures used to organize collections of data.
   * They are stored as references in memory, meaning the variable holds a location pointer to the actual data.
   * Unlike primitives, non-primitive data can be modified after creation.
   * Common non-primitive data types include:
     + **Object:** A collection of key-value pairs used to store related data. (e.g., { name: "Alice", age: 30 })
     + **Array:** An ordered list that can hold multiple values of any data type. (e.g., ["apple", "banana", 10])
     + **Function:** A block of code designed to perform a specific task.

**What are operator in javacript ?**

In JavaScript, operators are special symbols that perform operations on values (operands). They allow you to manipulate data, perform calculations, compare values, and control the flow of your program. JavaScript offers a variety of operators categorized by their function:

1. **Arithmetic Operators:**
   * Perform mathematical computations on numbers:
     + + (Addition)
     + - (Subtraction)
     + \* (Multiplication)
     + / (Division)
     + % (Modulus - remainder after division)
     + \*\* (Exponentiation)
2. **Comparison Operators:**
   * Compare values and return a boolean (true/false) result:
     + == (Loose equality - checks for equal value, can perform type coercion)
     + === (Strict equality - checks for equal value and type)
     + != (Inequality - opposite of loose equality)
     + !== (Strict inequality - opposite of strict equality)
     + < (Less than)
     + > (Greater than)
     + <= (Less than or equal to)
     + >= (Greater than or equal to)
3. **Logical Operators:**
   * Combine boolean expressions to make complex conditional statements:
     + && (Logical AND - both operands must be true for true result)
     + || (Logical OR - at least one operand must be true for true result)
     + ! (Logical NOT - inverts the value of a boolean expression)
4. **Assignment Operators:**
   * Assign values to variables, with some offering shorthand assignment:
     + = (Simple assignment)
     + +=, -=, \*=, /=, %= (Shorthand assignment - performs operation and assigns result)
5. **Bitwise Operators:**
   * Perform bit-level operations on numbers (less common):
     + & (Bitwise AND)
     + | (Bitwise OR)
     + ^ (Bitwise XOR)
     + ~ (Bitwise NOT)
     + << (Left shift)
     + >> (Right shift)
6. **Special Operators:**
   * Perform various tasks like type checking or conditional expressions:
     + typeof (Returns the data type of a value)
     + delete (Deletes a property from an object)
     + , (Comma operator - evaluates multiple expressions, returns the value of the last one)
     + ?: (Ternary conditional operator - shorthand if-else statement)

**Control structure**

Control structures in JavaScript are fundamental building blocks that dictate the flow of execution in your code. They allow you to make decisions, repeat code blocks, and handle errors, making your programs more dynamic and interactive. Here's a breakdown of the main control structures:

1. **Conditional Statements:**
   * These structures control the flow of code based on whether a certain condition is true or false.
     + **if statement:** Executes a block of code if a specified condition is true.

JavaScript

let age = 25;

if (age >= 18) {

console.log("You are eligible to vote.");

}

Use code [with caution.](/faq#coding)

* + - **if-else statement:** Provides an alternative block of code to execute if the condition in the if statement is false.

JavaScript

let time = 20;

if (time < 18) {

console.log("Good evening!");

} else {

console.log("Good night!");

}

Use code [with caution.](/faq#coding)

* + - **switch statement:** Evaluates an expression and executes a block of code corresponding to the matching case label. Useful for handling multiple conditions.

JavaScript

let day = "Monday";

switch (day) {

case "Monday":

console.log("Start of the week!");

break;

case "Friday":

console.log("TGIF!");

break;

default:

console.log("Just another day.");

}

Use code [with caution.](/faq#coding)

1. **Looping Statements:**
   * These structures allow you to repeat a block of code multiple times, based on a certain condition.
     + **for loop:** Executes a code block repeatedly for a predetermined number of iterations.

JavaScript

for (let i = 1; i <= 5; i++) {

console.log("Iteration", i);

}

Use code [with caution.](/faq#coding)

* + - **while loop:** Keeps executing a code block as long as a specified condition is true.

JavaScript

let count = 0;

while (count < 3) {

console.log("Count:", count);

count++;

}

Use code [with caution.](/faq#coding)

* + - **do-while loop:** Similar to a while loop, but guarantees at least one execution of the code block even if the condition is initially false.

JavaScript

let count = 0;

do {

console.log("Count:", count);

count++;

} while (count < 0);