JAVASCRIPT BASICS

**Compilers**:

* Translate the entire source code into machine code before the program runs. This results in faster execution since no translation is needed during runtime.

**Interpreters**:

* Translate code written in a high-level programming language into machine code line-by-line as the code runs.
* Interpreters translate one statement at a time, create an exe of the programming language, and execute source statements line by line.
* This makes its overall speed and execution slower than a compiler.

Don’t assign a variable as “name” outside of functions i.e. globally. In browser, global variable “name” has a special meaning. However, if you assign it inside a function, no issues. Go ahead!

**console.table:**

* console.table([var1,var2,var3])
* Just like console.log, it displays content in a table.
* var1 comes at 0th index of table, var2 at 1st and so on.  
   A black and yellow text

  Description automatically generated with medium confidence

**const variable:**

* value once provided cannot be changed
* value to be provided at the time of initialization only.
* const id = “1234”

**let variable:**

* values and data type can be changed later
* only declaration is sufficient, value can be provided later.
* let username;  
   username = “Ankit”  
   username = [1,2,3]

**null:**

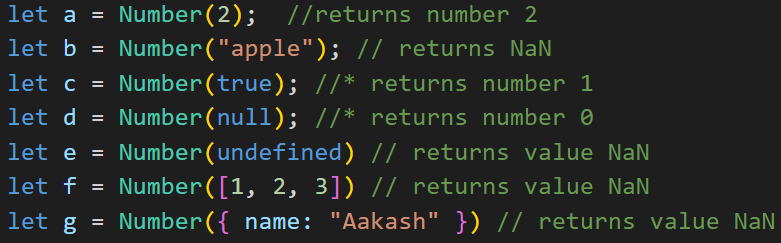
* It means the variable is empty
* Doesn’t mean 0, doesn’t mean empty string, it just means the variable is empty, there’s nothing there.

**typeof operator:**

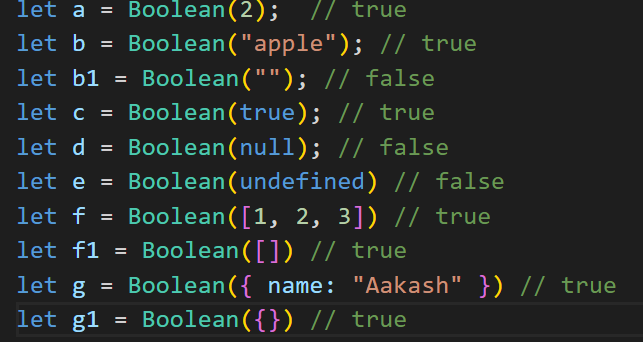
* used to check data-type of a variable
* JavaScript simplifies the work of typeof by returning 'object' whenever the given expression is a reference to some data in memory.
* typeof 2 *//Number*  
   typeof “apple” *//string*  
   typeof true *//boolean*  
   typeof null *// object\* //and not null, that’s how js is written*  
   typeof undefined *//undefined*

**Data type conversions:**

* In JavaScript, many times a need arises where we need to convert one data type to another. Sometimes it is feasible, while many a times it is not.

**Number() type conversion:****

* In above example we can see type conversion from a [string/boolean/null/undefined,array,jsonobject] to number
* Many times it’s not possible to convert a given data type to a number, hence it returns NaN, meaning Not a Number.
* Surprising thing about NaN is when you check its type:  
   - console.log(typeof NaN) //returns a number (LOL)

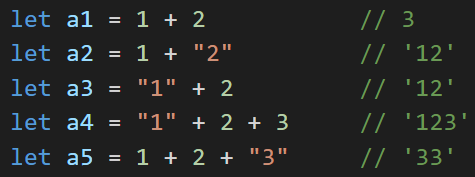
**Boolean type conversion:**  


* In above example, conversion of [string/number/null/undefined,array,jsonobject] to boolean

**String conversion:**  


* Convert any type to a string, it will add quotes around it
* [number/boolean/null/undefined,array,jsonobject] to string

**Auto conversion:**

* When performing certain operations, JavaScript converts one data type to another based on certain rules setup by ECMAScript directives.  
  
* In the above snippet, we can see numbers being converted to string. Observe the difference b/w a4 and a5

**Increment ++ operator (Prefix and Postfix):**

* it means increment the value by 1
* all concepts similar for -- decrement operator (means decrease value by 1)  
  ***Postfix***:
* first returns the “immediate” value, then increments the value
* let i=0;  
   console.log(i++) // *prints 0*  
   console.log(i) // *prints 1*
* let y = 0;  
   let n = y++ // first assign the immediate value of y to n, then increment value of y  
   console.log(n) //*prints 0*  
   console.log(y) //*prints 1*  
  ***Prefix***:
* first increment the value, then return the incremented value
* let i=0;  
   console.log(++i) // *prints 1*
* let y = 0;  
   let n = ++y // first increment value of y, then assign the incremented value of y to n  
   console.log(n) //*prints 1*  
   console.log(y) //*prints 1*

**Comparison operators:** == *equal to (data check)*

!= *not equal to (data check)*

=== *strictly equal to (data and data-type check)*

!== *strictly not equal to (data and data-type check)*

< *less than (data check)*

> *greater than (data check)*

<= *less than or equal to (data check)*

>= *greater than or equal to (data check)*

**Comparison operators – auto conversion:**

* when using above operators, they cause data type conversion as in Index 13  
  A screenshot of a computer

  Description automatically generated
* for checks related to null and undefined, see code