FSD Assignment – 3

Watch the following YouTube lectures on JavaScript – Values, Types and Operators. Please feel free to watch more videos of your choice.

1. History of JS

2. Data Types in JavaScript - 1

3. Data Types in JavaScript - 2

4. Type Conversion and Coercion

5. Arithmetic Operators

6. Relational Operators

7. Logical Operators

8. Ternary Operators

9. Template Literals

Once you complete watching those videos, document each topic you learnt with a proper definition(in your own words, don’t copy paste what was told in the video) and at least 3 examples for each topic.

**1. History of JS:**

**->**  in 1990, the internet was new. And mostly was used by companies to see information.

-> HTML authors who create websites using HTML, CSS.

-> To view a webpage we need a browser, there we have Mosaic in 1993.

-> Netscape created their own browser called Netscape navigator

-> giant companies like Microsoft thought of creating their own browser with free internet explorer.

-> Due to this competition, netscape thought of creating a few components like Sun microsystems to view so that they stay in the market and give a good user experience.

-> Brendan Eich created JavaScript in May 1995 while working at Netscape.

-> The language was initially called Mocha, then LiveScript, and finally JavaScript.

-> In 1996, Microsoft introduced JScript, a version of JavaScript for Internet Explorer.

-> JavaScript was standardized by ECMA in 1997 as ECMAScript.

-> JavaScript has evolved with new features and capabilities in various ECMAScript versions.

-> It is now widely used for web, server-side, and mobile applications, including technologies like Node.js and Electron.

**2. Data Types in JS- 1**

**->** Data is very important because any operation depends on data to store, process and make something meaningful out of it. So it is important to know about its type as well.

-> if we normally assign a number to a variable, its type is number.

-> if we assign any value in double quotes to a variable, its type is string.

-> Data types are of two types: Primitive and Object

-> Primitive consists of Number, String, Boolean, Null, Undefined, Symbol.

-> maximum value that a number type can hold is 1.7976931348623257e+308 but it might be inconvenient. So the safe maximum number is 900719925470991

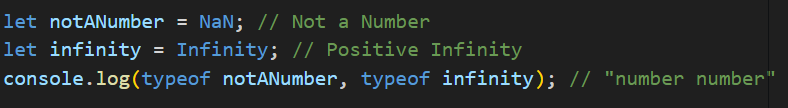
-> if we have a bigger number than that, we use BigInt which came recently and not there earlier. Eg: any big number + 2n gives a big integer value.

-> all floating, exponential numbers, not a number NaN comes under Number type. If we want to know the type of any variable we use “typeof”. Even octal, binary, hexadecimal numbers, Infinity, -Infinity are Number type.

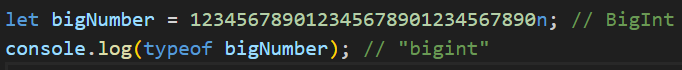
Example 1:



Example 2:



Example 3:



**3. Datatypes in JS- 2**

-> String carries text (eg: “text”) or string literal.

-> several functions can be performed on strings like searching for a character, fetching a character etc..

-> if we want to add space between two words we can concatenate an empty string ( “ “ ).

-> if we want a double quote inside a text we can either use backticks or single quotes for the whole sentence. And inside that we can use double quotes for the required word. or else, we can use open a back slash and close the backslash for the word which we want in double quotes. This is called an escape character.

-> \n is used for next line,\t is for horizontal tab, \v is for vertical tab, \b for deleting extra character

-> boolean data type is to check for something as true or false. That works in comparison.

-> Null type is a type which doesn’t have any value but doesn’t occupy any memory.

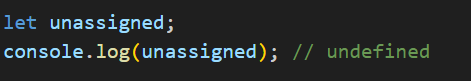
-> if we are trying to print any variable that is not assigned with any value. It is undefined.

-> Not a Number (NaN): if we try to perform any operation other than + with a string it gives NaN. But the typeof NaN is a number.

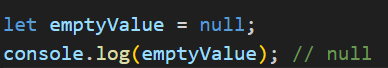
Example 1:



Example 2:



Example 3:



**4. Type conversion and coercion:**

-> If we want to convert one type of data to another type.

Ex: if we want to convert a number type to a string, we must give the string before the number. Such that it results as string type.

-> Similarly, we can do for booleans.

-> This is type conversions.

-> Coming to type coercion

-> In this if we add a number and an empty string, the resultant will be in string type but not always because that works only for “ + ". For other operations it performs that particular operation.

-> In case of boolean, if the data is 0 it is considered as false, rest numbers from - infinity to +infinity are true. Which is also known as truthy and falsy values.

-> Null is also treated as 0 because it doesn’t carry any number.

-> If we add a number and empty string, the resultant type will be string. If we want to convert that to a number, we must put the + or - symbol in front of the variable.

Parseint helps in extraction of the numbers in case we have to print a combination of number and string.

**5. Arithmetic Operators:**

-> Basic Operations: To perform basic mathematical calculations, we use arithmetic operators like + (addition), - (subtraction), % (modulus), / (division), and \* (multiplication).

-> Boolean Conversion: If two variables have boolean values, JavaScript converts true to 1 and false to 0 when performing arithmetic operations.

-> Adding to Itself: When you want to add a variable to itself, you can use the += operator. For example, x += 5 is the same as x = x + 5.

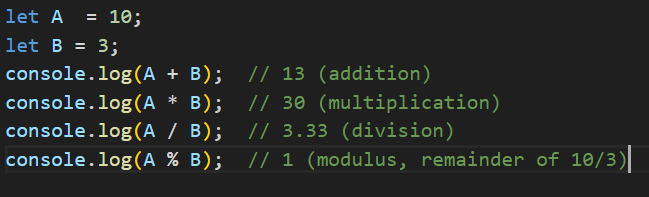
-> Increment and Decrement:

* Increment (++): Increases a variable's value by 1.
  + Pre-Increment (++variable): Increments the value first, then assigns it.
  + Post-Increment (variable++): Assigns the value first, then increments it.
* Decrement (--): Decreases a variable's value by 1.
  + Pre-Decrement (--variable): Decrements the value first, then assigns it.
  + Post-Decrement (variable--): Assigns the value first, then decrements it.

-> Calculating Power:

* To find the cube of a number, you can multiply the number by itself three times, like n \* n \* n.
* Alternatively, you can use Math.pow(n, 3) or the exponentiation operator n \*\* 3.

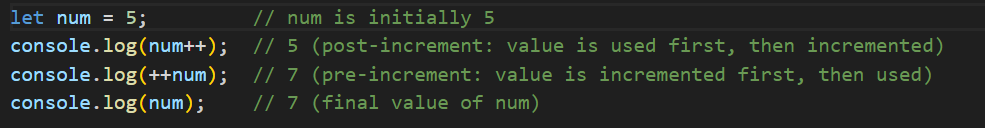
Example !:



Example 2:



Example 3:



**6. Relational Operators:**

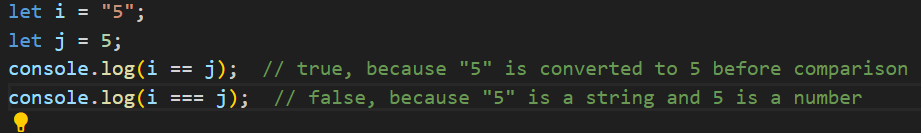
->Relational operators are used to compare numbers, strings, or other values.

-> Strings are compared based on ASCII values.

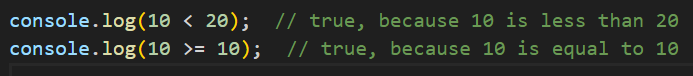
->If one variable is a string and the other is a number type, if we use “ + “, it’ll take it as concatenation. ->Other than that if we use “ >, <, <=, >=, ==(compares only value), ===( compares value and type). So, it’s better to use === than ==.

->JavaScript automatically converts values to a common type when comparing them, which can lead to unexpected results. Because of this, it's loosely typed.

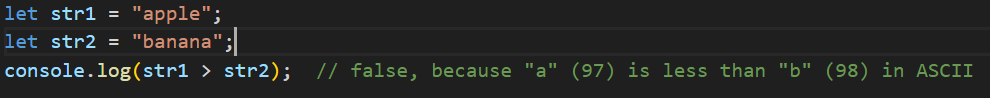
Example 1:



Example 2:



Example 3:



**7. Logical Operator:**

-> Whenever we want to compare two or more expressions, we use logical operators. The result will be in Boolean format.

AND (&), OR ( | ), NOT ( ! ). This is based on the truth table

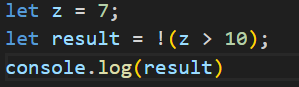
| A | B | A AND B | A OR B | NOT A |
| --- | --- | --- | --- | --- |
| T | T | T | T | F |
| T | F | F | T | F |
| F | T | F | T | T |
| F | F | F | F | T |

-> AND (&&): Returns true only if both conditions are true. If either one is false, the result is false.

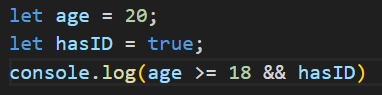
-> OR (||): Returns true if at least one condition is true. It is false only when both conditions are false.

-> NOT (!): Reverses the result. If something is true, NOT make it false, and if something is false, NOT make it true.

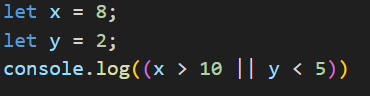
Example 1: output is True



Example 2: output: true



Example 3: output is true

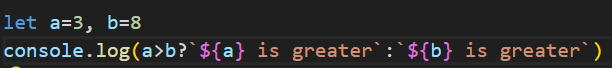


**8. Ternary Operator:**

-> Ternary operator provides us a short way to check conditional cases instead of using “if” “else” conditions.

-> Syntax is [ condition? ”Prints this statement if expression1 is true”: “Else prints expression 2” ]

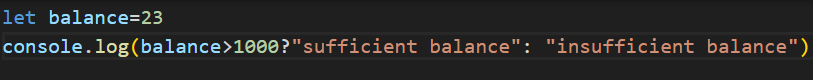
Example 1:



-> Here, the above can be written using if else conditions but, it takes many lines of code. Rather, we can use a ternary operator which has the power to write it shortly.

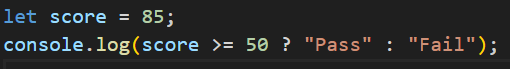
Therefore, in the above example, b is greater. Hence returns **8 is greater.**

Example 2:



-> Since the balance initialized is not greater than 1000, it returns **insufficient balance**.

Example 3:

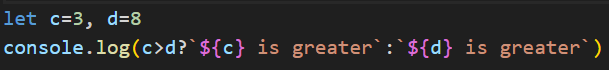


9. Template Literal:

-> Template literal is a special way to create strings that allows you to easily include variables or expressions within the string. Instead of using regular quotes (" or '), you use backticks (`). It also lets us write the string on multiple lines without any extra operators.

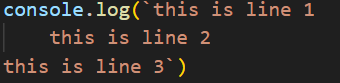
->With template literals, we can insert variables directly into the string by wrapping them in ${}.

Example 1

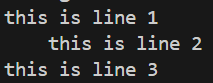


Output : 8 is greater

Example 2



Output:



Example 3



Output: hey,"hello"