Mistakes

Overview

This module covers some basic and common mistakes that most programmers make in their early stages of programming.

Using Assignment Operator

It is a common mistake of using an **assignment operator(=)** in place of the **comparison operator(== / ===)** that gives unexpected results

Example :

```
let temp = 5 ;
if( temp = 6 ) {
    console.log("true") ;
}else{
    console.log("false") ;
}
```

Expected output: false
Current output: true

0 - > false
Else all values except 0 - > true

Confusing Addition & Concatenation

Addition: Adding numbers
Concatenation: Adding strings

Both operations use the same **+** operator.

Because of this, adding a number as a number will produce a different result from adding a number as a string.

Example :
```
let x = 2 ;
let y = 3 ;
let z = x + y ;     // z = 5

let x = 2 ;
let y = "3" ;
let z = x + y ;    // z = “23”(String)
```

Mistake with Floats

Number with decimals are called floating values or floats
All numbers in JavaScript are stored as **64-bits Floating point numbers** (Floats).
All programming languages, including JavaScript, have difficulties with precise floating-point values:

Example :
```
let x = 0.1 ;
let y = 0.2 ;
console.log( x + y ) ;
```
Output : 0.30000000000000004

Solution :
```
let z = (x * 10 + y * 10) / 10 ;
console.log( z ) ;
```
Output : 0.3

Misplacing Semicolon

Because of a misplaced semicolon, this code block will execute regardless of the value of x

```
let x = 10 ;
if (x == 20) ;
{
   console.log(“Hello”) ;
}
```
Output: Hello (Not expected)

Misplaced Comma in Definitions

Trailing commas in object and array definition are legal after ECMAScript 5.

Object :

Example : person = { name : "Sam ", institute : "Coding Ninjas" , age : 20 , }

Array :

Example : var arr = { 1 , 2 , 3 , 4 , } ;