const = use it if the value can’t be change.

let = it can be use to change value multiple times.

Object:

const objectVariable={prop1:20,prop2:50};

undefined

objectVariable

▸Object{prop1:20, prop2:50}

objectVariable[0]

[error]

objectVariable.prop1

20

objectVariable['prop1'];

20

NestedObject:

const nestedObject={

layer1:{

layer2:{

layer3:{

targetValue: 20

}

}

}

}

undefined

nestedobject.layer1.layer2.layer3.targetValue;

20

function

const function ContainerVariable=function(){

return 20;

}

undefined

functionContainerVariable();

20

Operators

== doesn’t care about data type.

=== does care about data type.

Callback

function myCallback(someNumber) {

return someNumber \* 2;

function mainFunction(randomNumber, shouldCall, callback) I {

let result = random/umber; // in this example result === 20

// In this example, shouldCall is `true`, so we do reach the callback

if (shouldCall) {

// In this example, `callback` represents `myCallback` from above

result = callback(randomNumber);

// Since `result` was re-assigned by the callback function, returns 40

return result;

mainFunction (20, true, myCallback); // returns 40

Simple Solution

: mainFunction(20, true, function (num) {

return num \* 2;

})

: mainFunction (20, true, (num) => {

return num \* 2;

})

: mainFunction (20, true, num => num \* 2)