Equals Operator

#### **Learning Outcome Addressed**

  6. Assign primitive types and object types to variables

### Object Assignment vs. Primitive Assignment

We're about to learn how primitive assignment works and how it's different from object assignment.

#### Primitives vs. objects

Let's recall the different primitive types and objects in JavaScript.

**Primitive types**: number, string, Boolean, Null, Undefined, Symbol

**Object types**: Object, array, date

#### How we assign values to primitive types

Assigning a primitive value to a variable is pretty direct. The value is assigned to the variable.

Let's look at an example, running these commands in the console will show that a and b are both equal.

let a = 2  
let b = a  
console.log(a) // 2  
console.log(b) // 2

In the example above, a is set to the value 2. After b is assigned to a, b is set to the value 2, as well. This means if b is set to a new value, a will remain unchanged.

Let's change b to confirm this:

//now set b to equal 5  
b = 5  
console.log(b) // 5  
console.log(a) // 2

After running this in the console, you'll verify that b is changed but a keeps its original value.

#### Object assignment

Object assignment works differently. This is what happens when we assign an object to a variable:

* It creates an object in memory (heap)
* It assigns a reference to that object and stores that reference in a variable

//let's create one object  
let obj1 = {name: 'Demi'}  
//then assign obj1 to obj2  
const obj2 = obj1  
console.log(obj1) // {name: 'Demi'}  
console.log(obj2) // {name: 'Demi'}

The first line creates the object {name: 'Demi'} in memory and then assigns a reference to it in the variable obj1.

The second line creates a variable named obj2 and assigns it a reference to the object obj1.

Let's change a property of object assigned to obj2:

obj2.name = 'Alicia'  
console.log(obj1) // {name: 'Alicia'}  
console.log(obj2) // {name: 'Alicia'}

Since obj1 and obj2 are assigned a reference to the same object in memory, changing a property in obj2 would mean changing a property in the object in memory that both obj1 and obj2 are pointing to.

To be thorough, we can see this in action with arrays as well. Both shopping lists point to the same object in memory.

const summerShoppingList = ["figs", "apples", "cantaloupes"];  
const fallShoppingList = summerShoppingList;  
   
//if I change the first item in the list to "grapes"  
fallShoppingList[0] = 'grapes';  
   
//The first item in memory will change for both array lists, as we see in the console  
console.log(fallShoppingList); // ["grapes", "apples", "cantaloupes"]  
console.log(summerShoppingList); // ["grapes", "apples", "cantaloupes"]

Note: Be careful when copying objects; this can be unintended, and you may get unexpected results.

#### Task instructions

* Open the equals-operator.js file and create the array scottishCity and the variable americanCity. Re-assign the array to the americanCity variable. Find out the results.

Check the task when complete.

Task

Open the equals-operator-01 folder, follow the instructions to work in the task.

/\*

Instructions

- Create an array variable, scottishCity, set it equal to ["Glasgow"]

- Create another variable, americanCity, and set it equal to the array scottishCity

- Now, reassign the first element of the americanCity and set it to the value:

'New ' + americanCity

- Do you think both arrays have the same value in memory?

\*/

//write your code here

//////////////////////////

//uncomment the next lines one at a time to check your results in the browser console

//console.log(scottishCity)

//console.log(americanCity)

/////////////////////////

//don't change the code under this line

if (typeof module !== 'undefined') {

    module.exports = {scottishCity, americanCity};

}

**Equal Operator - Soccer manager**

### Task instructions

Open the equals-operator-01 folder, select the soccer-manager.js file and work on the arrays assignment.

* A soccer coach is trying to come up with the best "playing style" strategy to make his team play in 2 soccer league cups.
* He has 2 available sets of players: firstTeam and secondTeam.
* Let's help the manager set the best playing style. He has to choose between two possible positions for his teams:
  + 3, 3, 1, 3 formation
  + 4, 1, 4, 1 formation
* Declare a function called getPlayingStyle with no arguments
* Inside the function define 2 array variables, firstTeam and secondTeam
* Set firstTeam equal to the following array elements [3,3,1,3]
* Set secondTeam equal to firstTeam
* However, if the game becomes more challenging, the manager will opt for a more aggressive style of play, the [4,1,4,1] style.
* Change the playingstyle by setting firstTeam to the following new values:
* first element to 4
* second element to 1
* third element to 4
* fourth element to 1

Finally, return secondTeam array.

Hint: Remember that arrays are zero indexed.

* Do you think secondTeam has kept the same playing style or introduced a different one?
* Run the code in the console to find out what secondTeam's final playing style is.

Check your task when complete.

Task

Create the function getPlayingStyle, return the team you think it has the best playing style.

// Soccer manager activity

//Create the function getPlayingStyle, return the team you think it has the best playing style.

//Run the code in the console to find out what's the secondTeam final playing style.

/////////////////////////

//your code here

/////////////////////////

//Uncomment this line to see the results in the browser console

console.log("results = " + getPlayingStyle())

/////////////////////////

//don't change the code under this line

if (typeof module !== 'undefined') {

  module.exports = getPlayingStyle;

}