JS Assignment and Comparison Operators

Lesson Time: 30 Minutes

Assigning a value and comparing two values use different code symbols.

Assignment and comparison can be tricky for new students, and it's very easy to make a typo by using an assignment where you should have used a comparison and vice-versa.

Assignment operators assign values. We give a value to a variable by using the assignment operators.

```
var x = 5
//declare variable x and assign it the value of 5

var y = 5
y += 3
//declare y and assign it the value of 5
//y is equal to the current value of y + 3 (y will be equal to
8)
```

Assignment operators include

Symbol	**Meaning**
=	Equals. Set the value to this.
+=	Plus equals (the current value plus something else)
-=	Minus equals (the current value minus something else)

+	add
-	substract
%	divide
*	multiple

Comparison

Comparison operators ***compare two values***. In JS , the comparison operator is ==

```
var a = 10 //set a equal to 10
var b = 10 // set b equal to 10
if (a==b){
     console.log("a is equal to b")
}
```

Symbol	Meaning
==	Is equal to
===	Is equal to value AND Datatype
!=	Is not equal to
>	Is greater than
<	Is less than
>=	Is greater than or equal to
<=	Is less than or equal to

A quick note on **loose comparison**. Js gives us two ways to check if things are equal to each other, the double equal and the triple equal. It's best practice to use the triple equal, because it compares the value and the data type.

```
var x = 1 //number
var y = "1" //string
if (x == y)
```

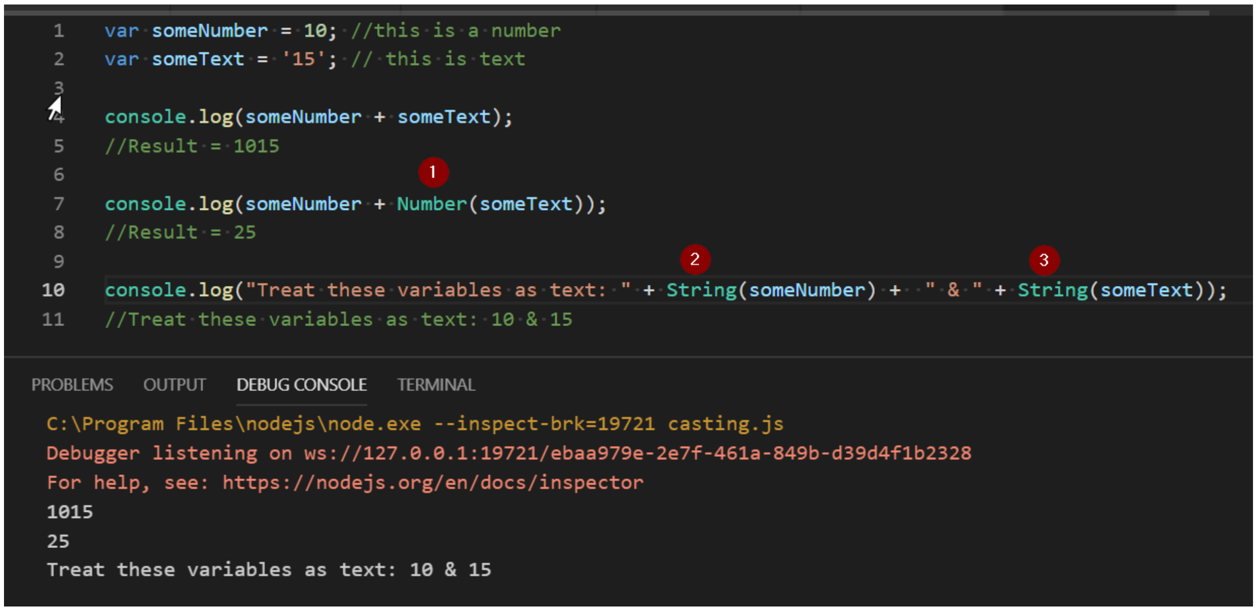
The statement above will evaluate as true, because the values are the same. However, x is a number and y is a string.

```
var x = 1
var y = "1"
if (x === y)
```

The statement above will evaluate as false, because they are different data types.

Casting / Data Type Conversion

Often we will have a text variable that needs to be treated like a number, or a number that needs to be treated like text. We can convert our variable by casting it. Look at the code example below.



```
var someNumber = 10; //this is a number
var someText = '15'; // this is text

console.log(someNumber + someText);
//Result = 1015

console.log(someNumber + Number(someText));
//Result = 25

console.log("Treat these variables as text: " + String(someNumber) +  " & " + String(someText));
//Treat these variables as text: 10 & 15
```

```
C:\Program Files\nodejs\node.exe --inspect-brk=19721 casting.js
Debugger listening on ws://127.0.0.1:19721/ebaa979e-2e7f-461a-849b-d39d4f1b2328
For help, see: https://nodejs.org/en/docs/inspector
1015
25
Treat these variables as text: 10 & 15
```

We have a text variable and a number variable and in our first example, we try to add them together with no casting. JS returns 1015, which is obviously wrong, because it can't add the text like it's a number.

In the second example, we cast someText as a number using Number(). JS correctly adds them like we expect.

In the final example, we want to print a string of text, so we use a String() cast to make the number be treated like text.

Key Terms	Comparison, assignment

Lesson Files	
Additional Resources	https://www.w3schools.com/js/js_operators.asp https://www.w3schools.com/js/js_type_conversion.asp
Further Learning	