**Equality**

So far, you’ve seen how you can use == and != to compare numbers and strings for equality. However, if you use == and != in situations where the values that you're comparing have different data-types, it can lead to some interesting results. For example,

"1" == 1

***Returns:****true*

and

0 == false

***Returns:****true. The == operator is unable to differentiate 0 from false.*

' ' == false

***Returns:****true. Both the operands on either side of the == operator are first converted to zero, before comparison.*

All of the above three evaluate to true. The reason for such interesting outcomes is ***Type Conversion***. In the case of regular comparison, the operands on either side of the == operator are first converted to numbers, before comparison. Therefore, a ' ', false, and 0 are all considered equal. Similarly, a '1' and 1 are also considered equal. If we don't want to convert the operands, before comparison, we have to use a **strict comparison** ===, that is explained below.

**Implicit type coercion**

JavaScript is known as a *loosely typed language*.

Basically, this means that when you’re writing JavaScript code, you do not need to specify data types. Instead, when your code is interpreted by the JavaScript engine it will automatically be converted into the "appropriate" data type. This is called *implicit type coercion* and you’ve already seen examples like this before when you tried to concatenate strings with numbers.

"julia" + 1

***Returns:****"julia1"*

In this example, JavaScript takes the string "julia" and adds the number 1 to it resulting in the string "julia1". In other programming languages, this code probably would have returned an error, but in JavaScript the number 1 is converted into the string "1" and then is concatenated to the string "julia".

It’s behavior like this which makes JavaScript unique from other programming languages, but it can lead to some quirky behavior when doing operations and comparisons on mixed data types.

**QUESTION 1 OF 2**

What value do you think the result of "Hello" % 10 will be?

* 

0

* 

"Hello10"

* 

10

* 

SyntaxError

* NaN

SUBMIT

***DEFINITION:****A****strongly typed language****is a programming language that is more likely to generate errors if data does not closely match an expected type. Because JavaScript is loosely typed, you don’t need to specify data types; however, this can lead to errors that are hard to diagnose due to implicit type coercion.*

**Example of strongly typed programming language code**

**int** count = 1;

string name = "Julia";

**double** num = 1.2932;

**float** price = 2.99;

**Equivalent code in JavaScript**

*// equivalent code in JavaScript*

**var** count = 1;

**var** name = "Julia";

**var** num = 1.2932;

**var** price = 2.99;

In the example below, JavaScript takes the string "1", converts it to true, and compares it to the boolean true.

"1" == true

***Returns:****true*

When you use the == or != operators, JavaScript first converts each value to the same type (if they’re not already the same type); this is why it's called "type coercion"! This is often not the behavior you want, and **it’s actually considered bad practice to use the == and != operators when comparing values for equality**.

**Strict equality**

Instead, in JavaScript it’s better to use **strict equality** to see if numbers, strings, or booleans, etc. are identical in *type* and *value* without doing the type conversion first. To perform a strict comparison, simply add an additional equals sign = to the end of the == and != operators.

"1" === 1

***Returns:****false*

This returns false because the string "1" is not the same type *and* value as the number 1.

0 === false

***Returns:****false*

This returns false because the number 0 is not the same type *and* value as the boolean false. Just like strict equality operator, there is also a ***strict non-equality*** operator !== that can be used instead of != if you don't want a type-conversion, before comparison. For example,

0 !== true

***Returns:****true*

and

'1' !== 1

***Returns:****true*

**QUESTION 2 OF 2**

Check the expressions that evaluate to true.

* 

"3" > 1

* 

3 != "3"

* 

true >= 0

* 

1 !== false

* 

"false" === 0

* 

3 === 3

SUBMIT

NEXT