== ? === ???

...#@^%

TYPE COERCION IN JAVASCRIPT

TYPE CONVERSION IN GENERAL

- aka typecasting or type coercion
- explicitly or implicitly changing an entity of one data type to another
  - ex: converting an int to a string

TYPE CONVERSION IN JAVASCRIPT

- JavaScript is dynamically typed
- type conversion can happen in two ways:
  - explicitly specified in the code
    - ex:

```
parseInt("123");
value.toString();
```

  - implicitly with type coercion

WHY LEARN THIS?

- it's built into the language
  - ex:

```
if (value.length) {}
```

- more flexibility, convenience
  - dealing with user input
- LVL UP

CONVERTING TYPES

TYPES IN JAVASCRIPT

- simple types
  - numbers, strings, booleans (true and false), null, undefined
- everything else is an object
  - arrays, functions, regular expressions, objects are objects

SIMPLE TYPES TO NUMBERS

- strings → try to parse to a number otherwise NaN
- booleans
  - true → 1
  - false → +0
- null → +0
- undefined → NaN

SIMPLE TYPES TO BOOLEANS

- numbers → true
  - unless the value is +0, -0, or NaN
- strings → true
  - unless the value is the empty string
- null → false
- undefined → false

CONVERTING OBJECTS

- objects to numbers or strings
  - implicit valueOf() and toString()
- objects to booleans → true

CONDITIONAL STATEMENTS

TRUTHY AND FALSY VALUES

- truthy
  - 123, "smalltalk", true, objects
- falsy
  - 0, NaN, "", false, null, undefined

IF, FOR, WHILE, DO-WHILE

- expect to evaluate boolean conditions
- if the expression does not return a boolean, operands are implicitly converted to booleans
- ex:

```
var value = document.getElementById("smalltalk");
if (value) {}
```

IF VS. SWITCH

- when evaluating case statements, no type coercion is performed
- ex:

```
var value = document.getElementById("smalltalk");
switch (value) {
  case true:
    // does not enter
    // will only enter here if var value = true;
    break;
  case default:
    // enters here
    break;
}
```

LOGICAL OPERATORS

&& ||

- operands are implicitly converted to booleans
- operand itself is returned, not the boolean value
- ex:

```
var value = 0 || "apple";
// 0 → false
// “apple” → true
// value is set to "apple"
```

MATHEMATICAL OPERATORS

+ - / * %

- operands are implicitly converted to numbers (except for dates)
- ex:

```
var value = true + null + 5;
// true → 1
// null → 0
// value is set to 6
```

OVERLOADED +

- addition between two numbers
- string concatenation
  - when one operand is a string and the other isn't, the other is converted to a string
- ex:

```
var value = "smalltalk is " + 2;
// value is set to "smalltalk is 2"
var value = 1 + 2 + " smalltalk";
// value is set to "3 smalltalk"
```

RELATIONAL OPERATORS

> < ≥ ≤

- operands are implicitly converted to numbers (even dates)
  - except for strings which are compared via lexicographical string comparison
- any comparison involving NaN evaluates to false
- ex:

```
var value = false < "123";
// false → 0
// "123" → 123
// value is set to true
```

EQUALITY OPERATORS

EQUALITY

- values are considered equal if they are:
  - identical boolean values
  - identical strings
  - numerically equivalent numbers
    - +0 equals -0
  - the same object
- NaN is not equal to anything, including itself

== !==

- compares for equality after doing any necessary type coercions if the types are different
- uses the abstract equality comparison algorithm, a series of 22 steps
- a surprise:

```
var value = null >= 0;
// value is set to true
var value = null == 0;
// value is set to false
```

== !== ALGORITHM (#14-22)

- null equals undefined
- if one operand is a string and the other operand is a number, implicitly convert the string to a number
- if one of the operands is a boolean, implicitly convert the boolean to a number
- if one the operands is a string or a number and the other operand is an object, implicitly convert the object to a primitive

=== !===

- compares for equality but no type coercion is done
- uses the strict equality comparison algorithm
- more faith in the result, much safer

SOME TIPS

- favor === over ==
- perform explicit type conversion
- use the JavaScript console and the ECMAScript Language Specification as resources
- http://dorey.github.io/JavaScript-Equality-Table/

THANK YOU