Basic Operators and Comparsion

Math Operators

```
+, addition
-, subtraction
*, multiplication
/, division
%, remainder
**, exponentiation
```

String Operators

The plus symbol if used with Strings will be for concatenation, you join two String together. If any of the operand is a String, the other one is also converted to a String too

Assignments

Assignment statement will do the assignment and return the value that was assigned, just like in Ruby

`console.log(x = 5);` This will print out 5 because after 5 is being assigned into the variable x, it will return 5 for the `console.log()` function to print.

Bitwise Operators

- AND = &
- OR = |
- XOR = ^
- NOT = ~
- LEFT SHIFT = <<
- RIGHT SHIFT = >>
- ZERO-FILL RIGHT SHIFT = >>>

== vs ===

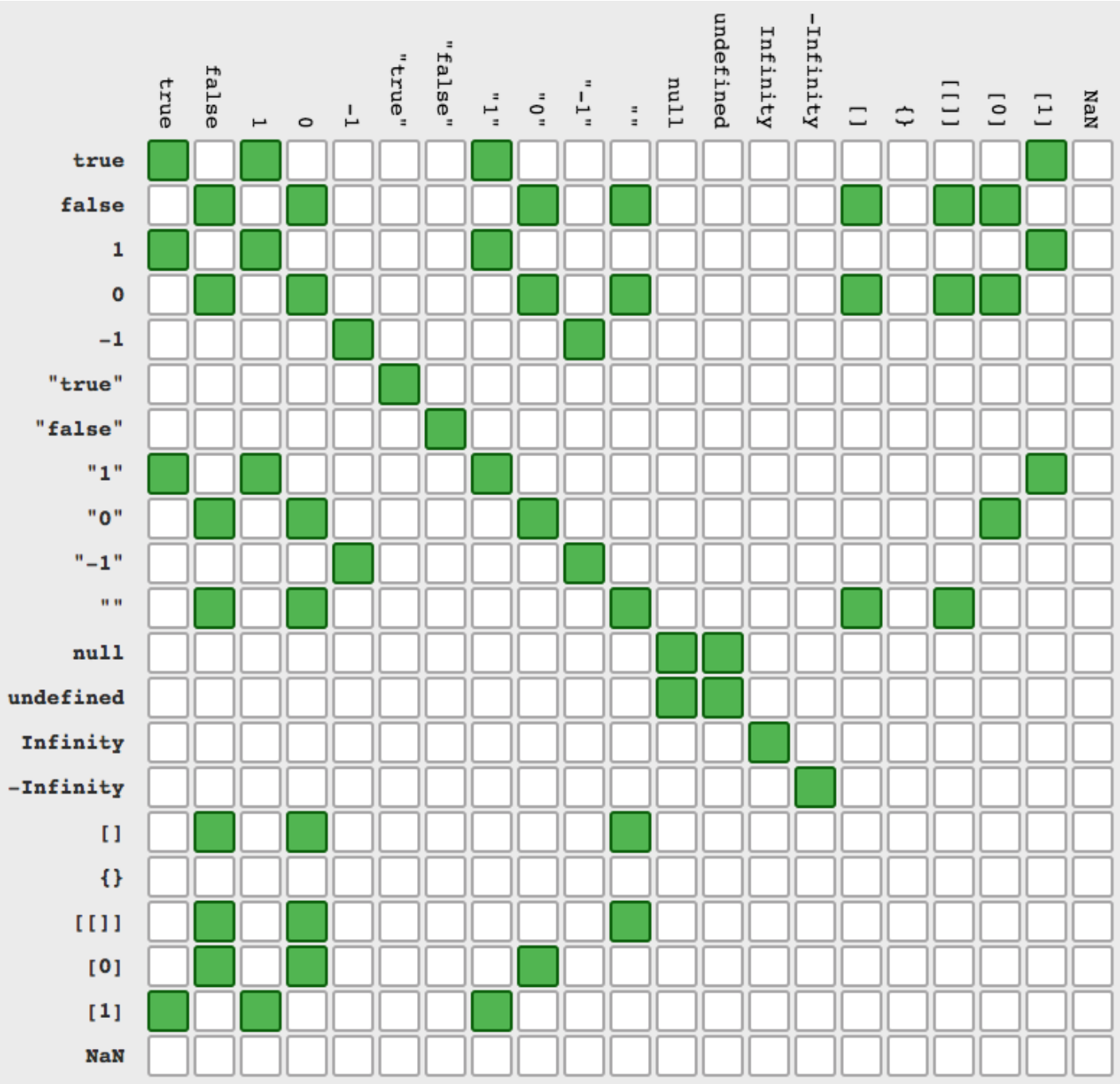
The problem with regular equality check is that it does type conversion by default. So you cannot distinguish between say 0 and false since `0 == false` result in true. This is because anything besides 0 are considered to be false.

In order to do equality check without type conversion you would use the triple equality operator, which does checks without type conversion.

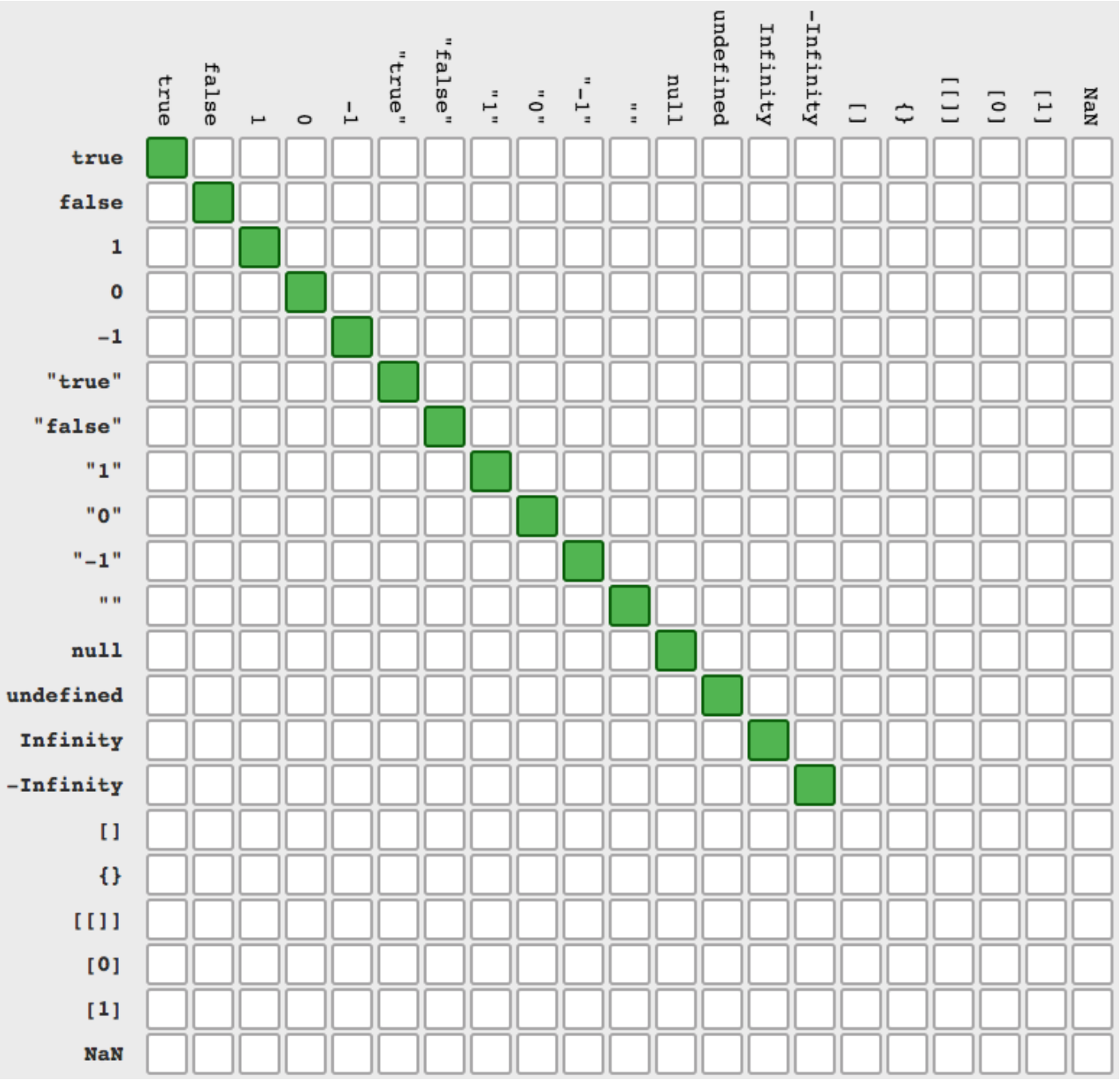
`a === b`, will result in `false` if `a` and `b` are different type.

There is also the `!==` variant compare to `!=`

Comparison using ==



Comparison using ===



null and undefined

When comparing `null` and `undefined` using non-strict check you will see them to be equal to equal other

```
null == undefined // true
null === undefined // false, to be expected
```

When `null` is used together with numbers, `null` is type converted to 0. However, this is only for comparison operator, not equality check!

```
null > 0 // false
null == 0 // false
null >= 0 // true
```

> Hence, you see that `null == 0` is false, because the type conversion from `null` to 0 isn't carried out. The equality check for `undefined` and `null` is defined such that without any conversion, they are equal to each other only and equal to nothing else!

When `undefined` is used together with numbers for comparison operator, it will always be false. This is because `undefined` gets converted to `NaN`. And equality check don't work because like mentioned previously `undefined` only equal to `null` and nothing else.

Takeaway

- If you are going to compare a variable that might be `undefined/null` treat it with care
- Don't use `>=, >, <, <=` if the variable might be `undefined/null` have a separate check to deal with those values.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness A nice read up on how the comparison is actually done, if needed for further clarification.

Revision #7
Created 2022-12-17 17:21:12 UTC by Tamarine
Updated 2023-07-27 02:35:11 UTC by Tamarine