Comparison Operators

Connor Rowe

# JavaScript comparisons will result in either Boolean True or False, they are binary

# JavaScript will convert some values to numbers for compariosn

# strings are compared on a **char** by **char** basis and more specifically by their unicode value

# **==** is used for equality, **===** is used for strict equality

* Some examples between strict equality: === and regular equality: ==

console.log( 15 == 15n ); // >>> true

console.log( 15 ==== 15n ); // >>> false

console.log( 15 == “15” ); // >>> true

console.log( 15 === “15” ); // >>> true

console.log( 0 == false ); // >>> true

console.log( 0 === false ); // >>> false

console.log( undefined == false ); >>> false

console.log( NaN == NaN ); >>> false

// Almost all the time will a == statement that’s false, their === statement will also be

False

**The negation ! operator can be attached to equality and strict equality to form the ( !== ) non identity operator and the inequality operator ( != )**

* **If the two operands in the inequality or strict equality operator are different, a false will be returned**

console.log( 10 !== 5 ); // >>> true

console.log( 10 != 5 ); // >>> true

console.log ( 10 !== “10” ) // >>> true

console.log ( 10 != “10” ) // >>> false

# Smaller Than, Greater Than

console.log ( 10 > 100 ); // >>> false

console.log ( 200 > 100 ); // >>> true

console.log (50n < 100 ); // >>> true

console.log( “10” < 50 ); // >>> true

## Smaller than, greater than Strings

# single char values are derived from their position in the alphabet, and any char that is further down the alphabet than a different one will have a larger value

console.log( ‘b’ > ‘a’ ); // >>> true

console.log( ‘ab4’ > ‘ab1’ ); // >>> true

console.log( ‘ab4’ > ‘abC’ ); // >>> true

console.log( ‘abcd’ > ‘abc’ ); // >>> true

Trying to remember all these rules is tedious, it’s probably better to just figure these out

Case-by-case in a separate compiler.

# Other Operators

## typeof

# Tells you the type of the thing that’s to the right of the typeof operator. Was already

mentioned.

console.log(typeof year); // >>> this will display the type, that is number (number)

console.log(typeof false); // >>> boolean

## instanceof

# the **instanceof** operator checks whether an object is of some type

let names = [“Connor”, “Sam”];

name = names[0];

console.log(names instanceof Array); // >>> true

console.log(name instanceof Array); // >>> false

## delete

# The **delete** operator will delete the selected field.

let user = Connor {

name: “Connor”,

Age: 22

};

console.log(user.age); // >>> Connor

delete user.age;

console.log(user.age); // >>> undefined

# Operator Precedence

# precedence can be presented as a numerical value where the highest number has the highest priority in the operation. For example, OP1 has less priority than OP2. For associativity, if the JavaScript interpreter has left-associativity, the order of operations for (a OP1 b OP2 c) will be that operator1 (OP1) is performed first, and then operator2. **Brackets allow you to impose a specific order of operation.**