Type Casting / Type Conversion / Type Coercion in JavaScript

- Type coercion is the process of converting value from one type to another.
- JavaScript is a "loosely typed" language, which means that whenever an operator or statement is expecting a particular data-type, JavaScript will automatically convert the data to that type.
- There are various operator and functions in JavaScript which automatically converts a value to the right type like alert(), log(), write() etc. function in JavaScript accepts any value and convert it into a string.

Type coercion in JavaScript

There are two types of coercion in JavaScript :-

a) Implicit Coercion

- Type conversion is done implicitly(automatically) by JavaScript.
- Implicit coercion is also done by the if() condition and == operator.

Example

```
console.log( "6" / "2" );        // returns 3
console.log( 1 + '2' );          // returns 12
console.log(12 + "");            // returns 12
console.log("15" * 2);           // returns 30
console.log("15" - "11")         // returns 4
console.log(undefined + 6)       // returns NaN
console.log("Hello " + null);    // returns Hello null
console.log(null + 25);          // returns 25
console.log(true + true);        // returns 2
console.log(false + 10);         // returns 10
console.log(10 * [6]);           // returns 60
console.log(10 * [10, 20]);      // return NaN
console.log([1] + [1,2]);        // Output is "11,2" as [1] is converted to "1" and [1,2]
                                 is converted "1,2". Finally, the two are concatenated to
                                 give the result "11,2"
```

b) Explicit Coercion

- Type conversion is done explicitly(manually by user) in code using the inbuilt functions like Number(), String(), Boolean(), etc.

Example

```
console.log(String(2));          // returns "2"
console.log(Boolean(0));         // returns false
console.log(Number("2"));        // returns 2
```

b.a) Converting To Number

- The Number() global method is used to convert any other data type value to numeric values.
- The parseInt() and parseFloat() methods can also be used to convert numbers stored as a string to a number but for all other data types, it will return NaN.

Example With Number()

```
console.log(Number("25")) //Output is 25 as "25" string is converted to number 25
console.log(Number("")) //Output is 0 as "" string is converted to 0
console.log(Number([])) //Output is 0 as [] is converted to 0
console.log(Number(null)) //Output is 0 as null is converted to 0
console.log(Number(true)) //Output is 1 as true is converted to 1
console.log(Number(false)) //Output is 0 as false is converted to 0
console.log(Number("Test")) //Output is NaN as "Test" could not be converted to number
```

Example With parseInt()

```
console.log(parseInt("25")) //Output is 25 as "25" string is converted to number 25
console.log(parseInt("")) //Output is NaN
console.log(parseInt([])) //Output is NaN
console.log(parseInt(null)) //Output is NaN
console.log(parseInt(true)) //Output is NaN
console.log(parseInt(false)) //Output is NaN
console.log(parseInt("Test")) //Output is NaN
```

Example With parseFloat()

```
console.log(parseFloat("25")) //Output is 25 as "25" string is converted to number 25
console.log(parseFloat("")) //Output is NaN
console.log(parseFloat([])) //Output is NaN
console.log(parseFloat(null)) //Output is NaN
console.log(parseFloat(true)) //Output is NaN
console.log(parseFloat(false)) //Output is NaN
console.log(parseFloat("Test")) //Output is NaN
```

b.b) Converting To String

- The String() & toString() global method is used to convert any other data type value to string values.

Example

```
console.log(String(25)) //Output is "25" as 25 is converted to string "25"
console.log(String([])) //Output is "" as [] is converted to empty string ""
console.log(String(null)) //Output is "null" as null is converted to string "null"
console.log(String(true)) //Output is "true" as true is converted to string "true"
console.log(String(false)) //Output is "false" as false is converted to string "false"
console.log(String({})) //Output is "[object Object]" as {} is converted to string(similar to
calling toString() on a object)
```

b.c) Converting To Boolean

- The Boolean() global method is used to convert any other data type value to Boolean values.

Example

```
console.log(Boolean(25)) //Output is true
console.log(Boolean([])) //Output is true
console.log(Boolean(null)) //Output is false
console.log(Boolean({})) //Output is true
console.log(Boolean("Hello Students")) //Output is true
```

Type Checking

- The 'typeof' is a JavaScript keyword / operator that will return the data type of a variable.
- It is a unary operator.
- The 'typeof' operator takes the variable as a parameter and returns the type of a variable or an expression.
- In the case of a complex data type, it returns an object (objects, arrays, and null) or function (functions).
- The return type of 'typeof' operator is 'string'.

Syntax

```
typeof operand
Or,
typeof(operand)
```

Example

```
var lang = {markup : "HTML", styling : "CSS", scripting : "JavaScript"}
console.log(typeof lang) // Object
```

Example

```
var lang = {markup : "HTML", styling : "CSS", scripting : "JavaScript"}
console.log(typeof(lang)) // Object
```

Example

```
var a = 23;                    // number
var b = 2n ** 53n;             // bigint
var c = "Hello";               // string
var d = true;                  // boolean
var e = null;                  // null
var f = undefined              // undefined
var g = {};                    // object
var h = [];                    // object
function hello(){              // function
    console.log("Hello Function")
}
console.log(typeof a);
console.log(typeof b);
console.log(typeof c);
console.log(typeof d);
console.log(typeof e);
console.log(typeof f);
console.log(typeof g);
console.log(typeof h);
console.log(typeof hello)
```

Example

```
console.log(typeof "Hello"); // returns String
console.log(typeof 21); // returns Number
console.log(typeof true); // returns Boolean
console.log(typeof {}); // returns Object
console.log(typeof []); // returns Object
console.log(typeof null); // returns Object
```

JavaScript Operators

- JavaScript operators are the symbols that are used to perform operations on operands.
- JavaScript Operators don't have any data type.
- JavaScript Operators have their return values.
- An operator performs some operation on single or multiple operands (data value) and produces a result.
- JavaScript contains both unary (single operand) and binary operators (two operands), and a ternary operator (conditional operator).

Syntax

    <Left operand> operator <right operand>

Example

```
var sum=10+20;
```

note :-
Here , + is arithmetic operator , while = is the assignment operator , whereas 10 & 20 are operands.

Types of JavaScript operators

There are following types of operators in JavaScript.

1) Arithmetic Operators
2) Comparison (Relational) Operators
3) Bitwise Operators
4) Logical Operators
5) Assignment Operators
6) Special Operators

Arithmetic Operators

- Arithmetic operators are used to perform mathematical operations between numeric operands.
- Return Type is Number.

a) Addition Operator(+)
It Adds two numeric operands.

Example
```
var x = 5, y = 10;
console.log(x + y) //returns 15
```

b) Substraction Operator(-)
Subtract right operand from left operand & Vice-Versa.

Example
```
var x = 15, y = 10;
console.log(x - y) //returns 5
```

c) Multiplication Operator(*)
Multiply two numeric operands.

Example
```
var x = 10, y = 10;
console.log(x * y) //returns 100
```

d) Division Operator(/)
Divide left operand by right operand.

Example
```
var x = 10, y = 5;
console.log(x / y) //returns 2
```

e) Modulus Operator(%)
Modulus operator. Returns remainder of two operands.

Example
```
var x = 10, y = 5;
console.log(x % y) //returns 1
```

f) Increment Operator(++)
Increment operator. Increase operand value by one.

Example
```
var x = 5;
console.log(x++) //returns 6
```

g) Decrement Operator(--)
Increment operator. Increase operand value by one.

Example
```
var x = 5;
console.log(x--) //returns 4
```

note :- + operator performs concatenation operation when one of the operands is of string data type.

Example

```
var a = 5, b = "Hello ", c = "World!", d = 10;

a + b; // "5Hello "
b + c; // "Hello World!"
a + d; // 15
```

Comparison Operators

JavaScript language includes operators that compare two operands and return Boolean value true or false.

a) Equals To (==)
Compares the equality of two operands values without considering their data type.

Example
```
var a = 5, b = 10;
console.log(a == b); // returns false
```

b) Strict Equals To (===)
Compares equality of two operands values along with their data types.

Example
```
var a = 5, b = 10, c = "5";
a === c; // returns false
```

c) Not Equals To (!=)
Compares inequality of two operands.

Example
```
var a = 5, b = 10, c = "5";
a != b; // returns true
```

d) Not Identical To (!==)
Checks whether the two operands are not identical.

Example
```
var x = 25;
var y = 35;
var z = "25";
alert(x !== z); // returns true
```

e) Greater Than (>)

Checks whether left side value is greater than right side value. If yes then returns true otherwise false.

Example

```
var x = 25;
var y = 35;
var z = "25";

alert(x > y);   // returns false
```

f) Greater Than or Equal To (>=)

Checks whether left operand is greater than or equal to right operand. If yes then returns true otherwise false.

Example

```
var x = 25;
var y = 35;
var z = "25";

alert(x >= y);  // returns: false
```

g) Less Than (<)

Checks whether left operand is less than right operand. If yes then returns true otherwise false.

Example

```
var x = 25;
var y = 35;
var z = "25";

alert(x < y);   // returns true
```

h) Less than or equal to (<=)

Checks whether left operand is less than or equal to right operand. If yes then returns true otherwise false.

Example

```
var x = 25;
var y = 35;
var z = "25";

alert(x <= y);  // returns true
```

Logical Operators

Logical operators are used to combine two or more conditions.

a) AND (&&) Operator

- && is known as AND operator.
- True if both conditions are true.

Example

```
var a = 5, b = 10;
(a != b) && (a < b); // returns true
```

b) OR (||) Operator

- || is known as OR operator.
- True if either first or either second condition is true.

Example 1

```
var a = 5, b = 10;
(a > b) || (a == b); // returns false
```

Example 2

```
var a = 5, b = 10;
(a < b) || (a == b); // returns true
```

c) NOT (!) Operator

- ! is known as NOT operator.
- It reverses the boolean result of the condition.

Example 1

```
var a = 5, b = 10;
(a != b) && (a < b); // returns true
```

Example 2

```
var a = 5, b = 10;
!(a < b); // returns false
```

Assignment Operator

The assignment operators are used to assign values to variables.

a) Assign (=) Operator
Assigns right operand value to left operand.

Example
```
var x = 5, y = 10;
console.log(x = y) // returns 10
```

b) Add & Assign (+=) Operator
Sums up left and right operand values and assign the result to the left operand.

Example
```
var x = 5;
console.log(x += 1) // returns 6
```

c) Subtract and Assign(-=) Operator
Subtract right operand value from left operand value and assign the result to the left operand.

Example
```
var x = 5;
console.log(x -= 1) // returns 4
```

d) Multiply and Assign(*=) Operator
Multiply left and right operand values and assign the result to the left operand.

Example
```
var x = 5;
console.log(x *= 5) // returns 25
```

e) Divide and Assign Quotient (/=) Operator
Divide left operand value by right operand value and assign the result to the left operand.

Example
```
var x = 5;
console.log(x /= 5) // returns 5
```

f) Divide and Assign Modulus (%=) Operator
Get the modulus of left operand divide by right operand and assign resulted modulus to the left operand.

Example
```
var x = 5;
console.log(x %= 2) // returns 1
```

Special Operators

- JavaScript's special operators are a hodge-podge of miscellaneous other symbols and words that perform other and important functions.
- These operators are used for some special purpose.

There are following types of special operators in JavaScript that includes :-

a) Ternary Operator
b) Comma Operator
c) Delete Operator
d) In Operator
e) Instanceof Operator
f) Typeof Operator
g) Void Operator
h) Exponential Operator
i) Unary Operator
j) Postfix & Prefix Operator
k) String Operator

a) Ternary Operator

- Ternary Operator returns value based on the condition.
- It is like if-else.
- It is also known as conditional operator.
- It is represented by (?:).

Syntax

```
(condition) ? code block of if statement : code block of else statement;
```

Example

```
var value = 5;
var check = (value > 10) ? console.log("Greater Than 10") : console.log("Less Than 10");
```

b) Comma Operator

- The comma operator evaluates two operands and returns the value of the second one.
- The comma operator (,) is one of the rarest and most unusual operators.

Example

```
var a = (1 + 2, 3 + 4);
console.log( a ); // 7 (the result of 3 + 4)
```

c) Delete Operator

- The delete operator removes a property from an object or an element from an array.
- The delete operator to remove an element from an array, the length of the array stays the same.
- The removed element will have a value of undefined.

Syntax

```
delete abc[indexArray];
```

Example

```
var abc = [10,20,"hello","world"];
console.log(abc[0]); // returns 10
delete abc[0];
console.log(abc[0]); // returns undefined
```

d) in Operator

- The in operator returns true if the specified value exists in an array or object.
- It checks the index value in array & key in objects.

Syntax

```
arrayIndex in array
```

Example

```
var animals = ['dog','cat','bird','octopus'];
if (3 in animals) {
 console.log ('it's in there');
}
```

e) instanceof Operator

- The instanceof operator returns true if the given object is same as the type of specified object.

Example 1

```
var myString = new String();
if (myString instanceof String) {
 console.log('it's a string!');
}
```

Example 2

```
var obj = new Boolean();
console.log(obj instanceof Boolean);
```

f) typeof Operator

- The 'typeof' operator is used to check the data type.

Syntax

```
typeof variableName
```

Example

```
var abc = 23;
var bac = "hey";

console.log(typeof abc); // returns number
console.log(typeof bac); // returns string
```

g) void Operator

- It discards the expression's return value.
- This operator returns 'undefined'.

Example

```
console.log(void(0)); // returns undefined
```

h) Exponential Operator

- This operator is used to powers an integer.

Example

```
console.log( 2 ** 2 );      // returns 4  (2 * 2)
console.log( 2 ** 3 );      // returns 8  (2 * 2 * 2)
console.log( 2 ** 4 );      // returns 16 (2 * 2 * 2 * 2)
console.log( 4 ** (1/2) );  // 2 (power of 1/2 is the same as a square root, that's
                            maths)
console.log( 8 ** (1/3) );  // 2 (power of 1/3 is the same as a cubic root)
```

i) Unary Operator

- A unary operator is one that takes a single operand/argument and performs an operation.
- A unary operation is an operation with only one operand.
- This operand comes either before or after the operator.

Example

```
console.log(+3); // returns 3
console.log(+true); // returns 1
console.log(+null); // returns 0
console.log(+undefined); // returns NaN
console.log(+"Infinity"); // returns Infinity
console.log(+Infinity); // returns NaN
console.log(-3); // returns -3
console.log(-true); // returns -1
console.log(!false); // returns true
console.log(!{}); // returns false
console.log(![]); // returns false
console.log(!undefined); // returns false
```

j) Postfix & Prefix Operator

1) postfix operator

1.1) post-increment (x++)

Returns x, then increments x by one.

Example

```
var x = 10;
console.log(x++); // Outputs: 10
console.log(x);   // Outputs: 11
```

1.2) post-decrement (x--)
Returns x, then decrements x by one.

Example
```
var x = 10;
console.log(x--); // Outputs: 10
console.log(x);   // Outputs: 9
```

2) prefix operator

2.1) pre-increment (++x)
Increments x by one, then returns x.

Example
```
var x = 10;
console.log(++x); // Outputs: 11
console.log(x);   // Outputs: 11
```

2.2) pre-decrement (--x)
Decrements x by one, then returns x.

Example
```
var x = 10;
console.log(--x); // Outputs: 9
console.log(x);   // Outputs: 9
```

k) String Operator

- There are two operators which can also used be for strings.
- String Operators are used to join two strings.

Example 1 (Concatenation Operator)

```
var str1 = "Hello";
var str2 = " World!";
console.log(str1 + str2); // Outputs: Hello World!
```

Example 2 (Concatenation Assignment Operator)

```
var str1 = "Hello";
var str2 = " World!";
str1 += str2;
console.log(str1); // Outputs: Hello World
```