Type Coercion:

একটা ডাটাকে অন্য ডাটাতে Convert করার প্রসেসকে জাভাস্ক্রিপ্টের ভাষায় Type Coercion বলা হয়ে থাকে। এটা দুই ভাবে হতে পারে। আমরা আমাদের প্রয়োজন অনুসারে নিজের ইচ্ছেই একটা ডাটাকে Convert করতে পারি, এটাকে বলে Explicit Coercion, আবার বিভিন্ন সময় জাভাস্ক্রিপ্ট নিজেই ডাটা গুলোকে এক টাইপ থেকে অন্য টাইপে Convert করে, এটাকে বলে Implicit Coercion। যেমন — string এর ভিতরে (+) ব্যবহার করে কোন নাম্বার যুক্ত করলে সেটি string এ Convert হয়ে যায়।

Implicit and Explicit Coercion:

Explicit Coercion আমরা আগে থেকেই দেখে আসছি। যেমন Number('123') বা Number(true)। দুইটা উদাহরণ এই string এবং বুলিয়ান নাম্বার এ Convert হয়ে যাবে।

যেহেতু জাভাস্ক্রিপ্ট weakly typed language তাই যে কোন প্রিমিটিভ ভ্যালু অথবা অবজেক্ট বিভিন্ন সময় অটোমেটিক্যালিও Convert হতে পারে। এটা সাধারণত ঘটে থাকে যখন আমরা ভিন্ন ভিন্ন অপারেটর ভিন্ন ভিন্ন ভ্যালুর ওপরে ব্যবহার করে থাকি। যেমন — 1 == null, 2 / '5', null + new Date() শুধুমাত্র

Three Types of Conversion:

Javascript এ শুধুমাত্র তিন ধরনের কনভার্শনই হয় -

to string
to boolean
to number

String Conversion:

String conversion happens when we need the string form of a value.

যদি আমরা যেকোনো ভ্যালুকে explicit ভাবে string এ Convert করতে চাই সেক্ষেত্রে আমরা String() ফাংশন ব্যবহার করতে পারি। আর যখন string এর সাথে কোথাও binary + অপারেটর ব্যবহার করা হয় তখনই এটা Implicit Coercion কে ট্রিগার করে।

```
String(123);              // explicit
1234 + '';                // implicit
```

সমস্ত প্রিমিটিভ ভ্যালু নিচের মত করে স্ট্রিং এ কনভার্ট হয়ে যাবে -

```
String(123);                // '123'
String(-12.1);         // '-12.1'
String(null);         // 'null'
String(undefined);      // 'undefined'
String(true);         // 'true'
String(false);         // 'false'
```

Boolean Conversion:

Explicit ভাবে একটি ভ্যালুকে বুলিয়ান এ Convert করতে আমরা Boolean() ফাংশনকে কল করতে পারি।

The conversion rule:

- Values that are intuitively “empty”, like `0`, an empty string, `null`, `undefined`, and `NaN`, become `false`.
- Other values become `true`.

যে কোন ভ্যালু যা ওপরের লিস্টে নেই সেটি আটোমেটিক্যালি true রিটার্ন করবে, এমনকি অবজেক্ট, অ্যারে এবং ফাংশন এর ক্ষেত্রেও সেটি true রিটার্ন করবে। Symbol, Empty Array, Empty Object ও true রিটার্ন করবে।

```
Boolean({});             // true
Boolean([]);             // true
Boolean(Symbol());       // true
!!Symbol();              // true
Boolean(function() {});  // true
```

```
Boolean(" "); //true// spaces, also true (any non-empty string is true)
```

Numeric Conversion:

Numeric conversion happens in mathematical functions and expressions automatically.

Explicit কনভার্শন এর জন্য শুধুমাত্র Number() ফাংশন কল করলেই সেটি নাম্বার এ রূপান্তরিত হয়ে যাবে।

comparison operators (>, <, <=,>=)

bitwise operators (| & ^ ~)

arithmetic operators (- + * / %). Note, that binary+ does not trigger numeric conversion, when any operand is a string.
unary + operator
loose equality operator == (incl. !=) Note that == does not trigger numeric conversion when both operands are strings

```
Number('123');   // explicit
+'123';          // implicit
123 != '456';    // implicit
4 > '5';         // implicit
5/null;          // implicit
true | 0;        // implicit
```

প্রিমিটিভ ভ্যালু গুলো যেভাবে নাম্বার এ কনভার্ট হয়ে থাকে -

```
Number(null);                   // 0
Number(undefined);              // NaN
Number(true);                   // 1
Number(false);                  // 0
Number(" 12 ");                 // 12
Number("-12.34");               // -12.34
Number("\\n");                  // 0
Number(" 12s ");                // NaN
Number(123);                    // 123
```

null এবং undefined একটু ভিন্ন ভাবে কাজ করবে। null হয়ে যাবে 0 এবং undefined হয়ে যাবে NaN।

আরও দুইটা স্পেসিয়াল রুলস আমাদের কে মনে রাখতে হবে -

১। যখন null অথবা undefined এর সাথে == অপারেটর ব্যবহার করা হবে তখন কোন রকম নিউমেরিক কনভার্শন ঘটবে না। null শুধুমাত্র null অথবা undefined এর সাথেই কম্পেয়ার করা যায়, অন্য কোন কিছুর সাথেই এর কম্পারিসন সম্ভব নয়।

```
null == 0;               // false, null is not converted to 0
null == null;            // true
undefined == undefined;  // true
null == undefined;       // true
```

If the string is not a valid number, the result of such a conversion is `NaN`. For instance:

```
let age = Number("conversion");

console.log(age); // NaN, conversion failed
```

২। NaN কখনই কারোর সমান হতে পারে না, এমনকি নিজের ও না।

Type Coercion for Objects:

এবার আমরা জাভাস্ক্রিপ্টের weird behaviour গুলোর সাথে পরিচিত হতে যাচ্ছি। যখন জাভাস্ক্রিপ্ট ইঞ্জিন [1] + [2, 3] এই রকম কোন এক্সপ্রেশন এর সম্মুখীন হয় তখন এটি প্রথমেই অবজেক্টটিকে প্রিমিটিভ ভ্যালুতে রুপান্তর করে ফেলে। তারপর এই রুপান্তরিত ভ্যালুটি আবার কনভার্ট হয়ে ফাইনাল রেসাল্ট দিয়ে থাকে। যেকোনো অবজেক্ট কনভার্ট হয়ে number, string বা boolean ই হতে পারে।

সাধারণ ভাবে অ্যালগোরিদমটি নিচের মত কাজ করে থাকে -

- যদি ইনপুটটি আগে থেকেই প্রিমিটিভ হয়ে থাকে, কিছুই করতে হবে না, শুধুমাত্র এটাকেই রিটার্ন করবে
- call input.toString(), যদি রেসাল্ট প্রিমিটিভ হয়, রিটার্ন করবে
- call input.valueOf(), যদি রেসাল্ট প্রিমিটিভ হয়, রিটার্ন করবে
- যদি input.toString() অথবা input.valueOf() কোনটাই প্রিমিটিভ রিটার্ন করতে না পারে তাহলে error থ্রো করবে

নিউমেরিক কনভার্শনের ক্ষেত্রে এটি প্রথমে valueOf() ফাংশনকে কল করবে এবং ফলব্যাক হিসেবে toString() ফাংশনকে কল করবে। স্ট্রিং কনভার্শনের ক্ষেত্রে ঠিক উল্টা ব্যাপারটা ঘটবে, প্রথমে toString() কল হবে, আর ফলব্যাক হিসেবে valueOf() কল হবে।

বেশিরভাগ বিউল্টইন টাইপের valueOf ফাংশনটি নেই, আর থাকলেও এটা নিজ অবজেক্টকেই রিটার্ন করে থাকে। এবং যেহেতু এটা প্রিমিটিভ না তাই কম্পাইলার এটাকে ইগ্নোর করে থাকে। সেই হিসেবে নিউমেরিক কনভার্শন এবং স্ট্রিং কনভার্শন মূলত একই ভাবে কাজ করে থাকে।

Examples:

Example - 1: Binary + অপারেটর নিউমেরিক কনভার্শন ট্রিগার করছে -

```
true + false
==> 1 + 0
==> 1
```

Example -2: অ্যারিথমেটিক / অপারেটর নিউমেরিক কনভার্শন ট্রিগার করছে -

```
12 / '6'
==> 12 / 6
==> 2
```

Example-3:

```
!+[]+[]+![]
==> (!+[]) + [] + (![])
==> !0 + [] + false
==> true + [] + false
==> true + '' + false
==> 'truefalse'
```

Example - 4:

```
[]+[]+'foo'.split('')
==> []+[]+['f', 'o', 'o']
==> '' + '' + 'f,o,o'
==> 'f,o,o'
```