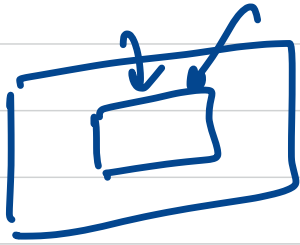
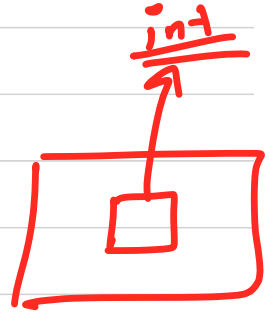


Coercion

↳ Coercion stands for type inter conversion.



implicit explicit



C++, Java, C → types exist for variables.

↳ `int x = 10;`

JavaScript → types exist for values.
`let x 10;`
`x = "str";`

"Everything in JS is an object" → False

let x = true;

How JS handles coercion??

Abstract Operations →

→ first abstract operation that we need to learn
is ToPrimitive

* The ToPrimitive abstract operation, takes an input argument and an optional Preferred Type argument.

* This operation converts the input to a non-object type value. If an argument is capable of getting converted into more than one primitive type, then the funcⁿ uses Preferred Type argument to resolve it.

⇒ As we said, this is also an abstract operation.
that means it is conceptual, we cannot invoke it,
but JS internally can.

verify

1. Assert: input is an ECMAScript language value.

2. If Type(input) is Object, then

a. If PreferredType is not present, let hint be "default".

b. Else if PreferredType is hint String, let hint be "string".

c. Else PreferredType is hint Number, let hint be "number".

d. Let exoticToPrim be ? GetMethod(input, @@toPrimitive).

e. If exoticToPrim is not **undefined**, then

i. Let result be ? Call(exoticToPrim, input, « hint »).

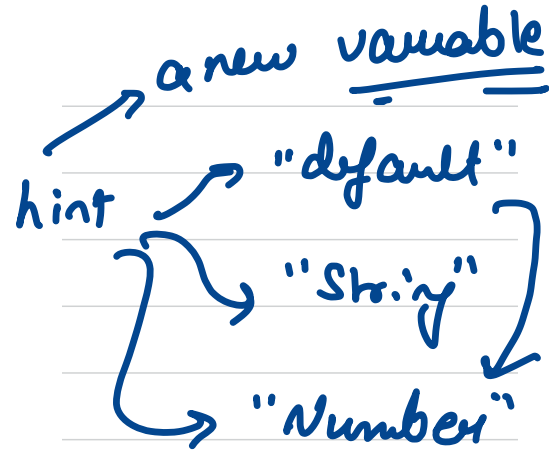
ii. If Type(result) is not Object, return result.

iii. Throw a **TypeError** exception.

f. If hint is "default", set hint to "number".

g. Return ? OrdinaryToPrimitive(input, hint).

3. Return input.



↳ the ToPrimitive function prefers string &
number conversion.

hint



"number"



method _{name} → ["valueOf", toString]

String
↓

methodName → ["toString", valueOf]



result ← toString (input)

→ if hint/preferred type is number then on our
input argument we call valueOf() function
to get a number & if it doesn't give a primitive
we call toString() function.

Otherwise

if hint argument is "String" then we first call
toString() & then valueOf()

hint → "Number"



valueOf()

toString()

hint → "String"

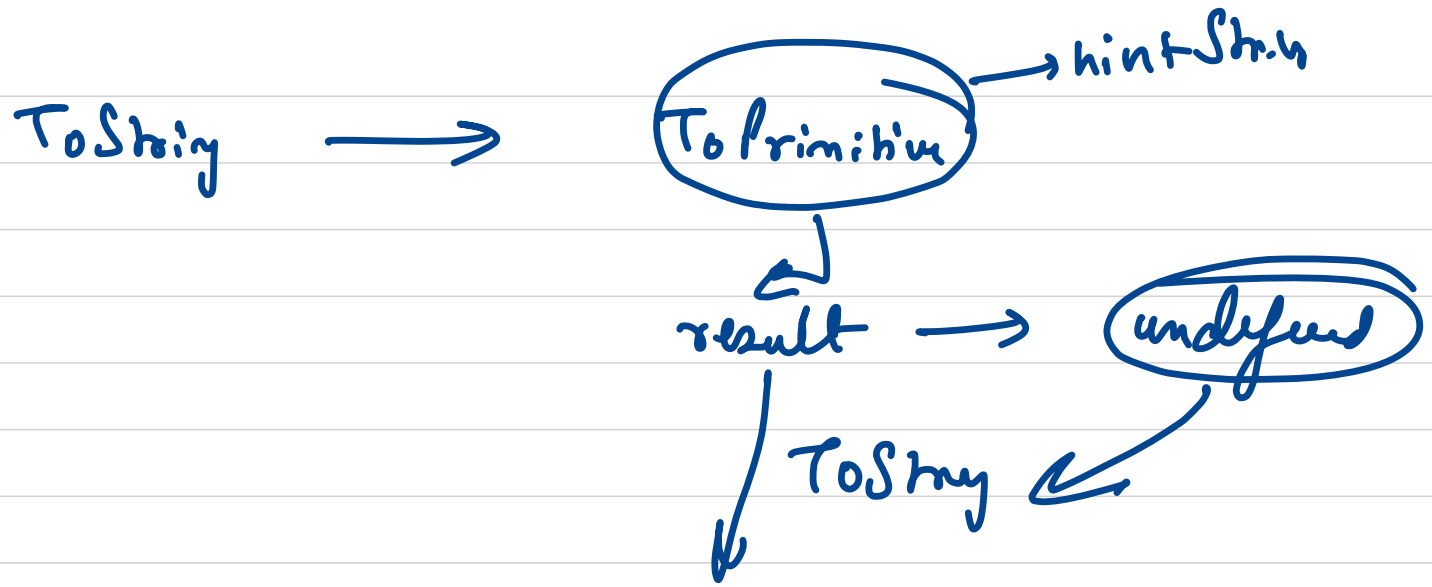
toString()

valueOf()

{ "a": 5

valueOf: fun()

} →



To Number

" " $\rightarrow 0$

"0" $\rightarrow 0$

"-0" $\rightarrow -0$

"3.145" $\rightarrow 3.145$

"0." $\rightarrow 0$

"." $\rightarrow \text{NaN}$

⋮

"123" $\rightarrow 123$

"12+3" $\rightarrow \underline{\underline{\text{NaN}}}$

[null] $\rightarrow 0$

[undefined] $\rightarrow 0$

["0"] $\rightarrow 0$

[""] $\rightarrow 0$

[1,2,3] $\rightarrow \text{NaN}$

$$\text{true} - 8 = -7$$

↓

$$1 - 8 \Rightarrow \underline{\underline{-7}}$$

$$\text{"123"} \rightarrow \underline{\underline{123}}$$

$$\text{"687340"} \rightarrow 687340$$

$$\text{"68+347"} \rightarrow \text{NaN}$$

$$\text{"abc"} \rightarrow \text{NaN}$$

Whenever is an numeric operation we don't have a number, `ToNumber()` is called. for example in

Subtraction operation.

$x = \{ \widetilde{\text{key}} : \widetilde{\text{value}} \}$

$x = \{ \text{valueOf}() \text{ } \}$



$\{ \text{valueOf} : () \}$

$x = \{ \}$

object to primitive
hint → number

→ value of
to String

