2017 WEDNESDAY

10 20th Week • 130-235
MAY

MAY 2017

Wk	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

09.00

10.00

11.00

12.00

01.00

02.00 JS:- Coercion

03.00

04.00

05.00

06.00

07.00

✉ @ ☎

JUNE 2017

WK	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

4:15 got

THURSDAY 2017

20th Week • 131-234

MAY 11

Abstract Operations

09.00 ⊙ These are some set of algorithms, that is present in ecmascript docs,

10.00 but they are not available for usage in ecmascript.

11.00 ie we as developers cannot use these operations directly.

12.00 ⊙ They are mentioned in the docs to help the documentation only.

01.00 In the ecma docs there are a lot of things that are done by the lang. internally. To explain these internal details of how & what lang.

02.00 is doing, we have abstract operations. mentioned in the docs.

03.00 Summarizing,

04.00

05.00 Abstract Operations are some algorithms, that js internally executes.

We as a end developer cannot use those operations

06.00

07.00 -: Coercion :- → type conversion.

Whenever we do an operation, based on the I/P we actually convert the I/P for conversion.

convert → We can convert the type of I/P.

When we do this type conversion manually → Explicit type conversion

When the language based on certain rules automatically converts the type → Implicit type conversion.

Implicit type conversion → Coercion.

2017 FRIDAY

12 MAY

20th Week • 132-233

MAY 2017

WK	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

Operators

add → (+) operator.

09.00

(1) 12 + 13 (Assumption :- 12 & 13 are no.s)

10.00

∵ Both (12) & (13) are no.s ∴ (+) acts as Arithmatic Operator.

11.00

(2) 12.00 12 + "13" (12 → number)

01.00 "1213" concatenation.

02.00

(3) "1" − 1 → 0

Kinds of Abstract Operations :-

03.00

Coercion in Subtraction opr

(A) To Number :- the abstract Operation (To Number) converts argument

04.00

↓

(-) operator. 05.00 to a value of type Number. {To Number (argument)}

Undefined	→ NaN
Null	→ +0
Boolean	→ True → 1 | False → +0
Number	→ no conversion.
Symbol	→ Throw Typeerror exception.

06.00

07.00

(1) 10 − Null

10 +0 ∴ 10 − 0 = 10

(2) 10 − undefined

10 NaN = NaN

✉ @ ☎

(3) 10 − true

10 1 = 9

(4) "6384" → $6 \times 10^3 + 8 \times 10^2 + 8 \times 10^1 + 4 \times 10^0$

09.00

1 — "68 bcd 98" → NaN

10.00

∴ For Subtraction Operator :- Algorithm

11.00

Js will convert left operand to a valid Num.

12.00

" " right " " " " Num

01.00

Then, apply Subtraction operator b/w them.

02.00

03.00 (5) Console.log (1 - "0xa") → -9

04.00

(B) To Primitive :- ToPrimitive (input, [Preferred Type])

05.00

→ optional

06.00 ⊙ Convert the I/P to non-object type.

07.00 ⊙ Consider a case where the given I/P is capable of being converted to more than one type. In that case, which type to prefer we are going to determine via preferred type.

SUNDAY 14

Algorithm :- ① Given I/P is a valid Ecma script lang. val.

② If type of I/P is object → Case 1
Case 2
Case 3

Else : return I/P.

✉ @ ☎

Case-01 :— Preferred type is not present.

⊙ We create a variable (Hint) and assign a value (default) to it.

∴ hint → "default".

Case-02:— Preferred type is a String.

∴ hint → "String".

Case-03 :— Preferred type is a Number.

∴ hint → "Number".

⊙ Once, the hint has been initialized. then,

. If variable hint is still default. We will set it to Number.

∴ To Primitive converts the I/P to a non-object type.

primitive.

Return [Ordinary To Primitive (input, hint)]

Ordinary to Primitive :— Ordinary to Primitive (Input, hint)

Input → Always an object.

hint → From Case1, 2 & 3 only 2 values are possible

(i) "string" (ii) "number".

MAY 2017

W	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

Ordinary to Primitive (Input, hint)

Ordinary to Primitive (continued) :-

hint

"String"

ⓐ methodNames = ["toString", "valueOf"]

ⓑ Run a for loop on methodNames.

ⓒ Call "toString" on Input.

ⓓ If it returns a primitive type return it.

If "Not Primitive"

⊙ Call "valueOf" on I/P.

⊙ If "valueOf" returns a Primitive type return it.

If both of them are unable to return a Primitive type return a type error exception.

"Number"

ⓐ methodNames = ["valueOf", "toString"]

ⓑ Same process as before just that we call valueOf first of then toString.

If it succesfully able to convert Object to primitive type return it.

If it succesfully able to convert Object to prim. return it

else

throw type error exception.

JUNE 2017

WK	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

THURSDAY 2017

21st Week • 138-227

MAY 18

Putting things into perspective,

09.00

① To Number (Object) → To Primitive (argument, hint)

10.00

11.00

12.00 Then, return it. ← Whatever is the result, to Number is called on that result.

01.00 ② The default representation of toString in a Object returns

02.00 String ([object Object]) → default type "String"

03.00 ③ The default implementation of valueOf in a Object returns the same

04.00 Object. → default type "Object"

05.00 But, we can over-ride them.

06.00 let obj = {}; console.log (10 - obj)

07.00 10 gets converted to 10.

obj → To Primitive (obj)

∵ No preferred type is mentioned

∴ preferred type = "number"

∴ hint → "Number".

∴ To Primitive (obj, "Number")

↓

Ordinary to Primitive ([valueOf", "toString"])

⊙ obj. valueOf ()

↳ {} (Same Object)

↓ ✓

is it an Object | Non-Object

⊙ obj. toString ()

↳ [object Object]

↓ ✓

is it object | Non-Object

∴ return String ([object Object])

FRIDAY

19 MAY

21st Week • 139-226

MAY 2017

Wk	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

Now, for the final touch

ToNumber ([object Object])

this is a string

↓ dfp

NaN

∴ 10 − NaN = NaN ✓

let obj = {x:9, y:8} ; console.log (100 − obj1)

100 gets converted to 100.

{x:9, y:8} ⟶ ToNumber ({x:9, y:8}) (Object)

↓

ToPrimitive (obj)

∵ No preferred type is set.

∴ hint ⟶ "default" ⟶ hint = "Number"

↓

OrdinaryToPrimitive (obj, "Number")

↓

["valueOF", "toString"]

obj.valueOf()

↓

{x:9, y:8} Object | NonObject ✓

obj.toString()

"[object Object]" Object | NonObject ✓

return

∴ ToNumber (obj) ie ToNumber ("[object Object]")

↓ NaN

∴ 100 − NaN = NaN.

09.00 # let Obj = {x:7, valueOf(){return 99;}}.

console.log(100 - Obj)

10.00

⊙ 100 gets converted to 100.

11.00

⊙ Obj → ToNumber(Obj) → ToPrimitive(Obj)

12.00

∵ No preferred type is mentioned.

01.00

∴ hint → "default"

02.00 ↓

03.00 hint = "Number".

04.00 ↓

05.00 OrdinaryToPrimitive(Obj, "Number")

06.00 ∵ "Number" → ["valueOf", "toString"];

07.00 For each loop is run of the Array.

Obj.valueOf()

↓

99 Object / Not Object ✓

↑ return.

∴ 100 − (99)

↓

ToNumber(99) = 99.

∴ 100 − 99 = (1) ✓ ans.

SUNDA

✉ @ ☎

MONDAY

22 MAY

22nd Week • 142-223

MAY 2017

WK	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

let obj = { x:8, toString() { return "88" } };

09.00

console.log (100 - obj)

10.00

100 get converted to 100.

11.00

Object.

obj → ToNumber({ x:8, toString() { return "88" } });

12.00

↓

01.00

ToPrimitive (obj)

02.00

∵ No preferred type has been set.

03.00

∴ hint → "default" → hint = "Number"

04.00

↓

05.00

Ordinary ToPrimitive (obj, "Number");

06.00

∵ hint is a "Number" ∴ ["valueOf", "toString"].

07.00

ForEach loop is run on the array.

obj.valueOF()

↓

{x:8, toString() { return "88" }}

✓

Object | Non Object

obj.toString();

↓

"88"

✓

Object | Non-Object

↑ return "88"

✉ @ ☎

Now, 100 - ("88")

↑

ToNumber("88") → 88

∴ 100 - 88 = (12) ans.

let obj = {2:7, toString(){return {}}};

09.00 console.log(100 - obj)

10.00

100 gets converted to 100.

11.00

obj → ToNumber(obj) → ToPrimitive(obj)

12.00

∵ No preferred type is mentioned.

01.00

∴ hint → "default" → hint = "Number"

02.00

↓

03.00 OrdinaryToPrimitive(obj, "Number")

04.00 ∵ hint is "Number" → ["valueOf", "toString"]

05.00 For Each loop on the Array.

06.00 obj.valueOf();

07.00 {2:7, toString(){return {}}};

Object | Non Object

obj.toString();

{} Object | Non-Object

∵ Both returned Non Primitive / Object.

∴ Return TypeError exception.

∴ o/p is TypeError exception. (Ans)

✉ @ ☎

Coercion in Addition Operator

09.00

10.00 ① Fetches the value of both the operands.

11.00 ② Tries to convert both of them into primitive

12.00 using, toPrimitive (I/P) also, no preferrence parameter.
∴ Conversion preferrence is "Number"

01.00 ⊙ ~~if not primitive, then~~. After point ②

02.00 (A) IF any of left operand or right operand is a String.

03.00 Then, convert both of them into String.

04.00 & return the String concatenated result.

05.00 (B) IF both of them are not String.

06.00 Then, we do simple addition operator.

07.00 Note
IF we want to convert an I/P to a Number (num). Then,

+I/P or -I/P. and store it inside a variable.

Equality Operators

⊙ Abstract Equality Operator (==)
⊙ Strict Equality Operator (===)

} Both of them, checks type.

let obj = { } ; console.log ("18" + obj)

09.00

"18" gets converted to "18" ie. toPrimitive ("18") → "18"

10.00

obj → ToPrimitive (obj)

11.00

∵ No preference type hint → "default"

12.00

↓

01.00 hint = "Number"

↓

02.00 Ordinary toPrimitive (obj, "Number")

03.00

∵ hint = "Number"

04.00 → ["valueOF", "toString"] ✓

05.00 obj.valueOF() → { } object | Not-object

06.00

∴ obj.toString() → "[object Object]" Object | Not object ✓

07.00 ↑ return.

∴ "18" + "[object Object]"

→ "18 [object Object]" ans.

Concatenated ans

2017 FRIDAY

26 22nd Week • 146-219 MAY

	M	T	W	T	F	S	S
19	1	2	3	4	5	6	7
20	8	9	10	11	12	13	14
21	15	16	17	18	19	20	21
22	22	23	24	25	26	27	28
23	29	30	31				

let obj = { }. ; console.log (18 + obj)

09.00

18 ⟶ toPrimitive (18) ⟶ (18)

10.00

obj ⟶ toPrimitive (obj)

11.00

∵ No preferred type

12.00

∴ hint → "default" ⟶ hint = "Number"

01.00

↓

02.00

Ordinary toPrimitive (obj, "Number")

03.00

∵ hint is "Number"

04.00

["valueOF", "toString"].

05.00 (Foreach) loop on the array.

06.00 obj.valueOF() ⟶ { } ✓ object | Non-object

07.00 obj.toString() ⟶ "[object Object]" ✓ Object | Non-object

↑ return

18 + "[object Object]"

"18" + "[object Object]"

↓

"18 [object Object]" Ans

→ logical NOT (!)

To Boolean. { To Boolean (argument) }.

undefined :- False.	Boolean :- T | F
Null :- False	Number → +0 | -0 | NaN → false → Rest all → true
Symbol :- true	
Object :- true.	String → empty string → false → else, return true.

09.00 10.00 11.00 12.00 01.00

let x = 10; console.log (!x)

02.00

x → true ∴ !true → false ans.

03.00

let y = undefined; console.log (!y)

04.00

y → false ; ∴ !false → true ans.

05.00

06.00 if (10) { → true

07.00 console.log ("Hi"); → "Hi" as o/p

}

SUNDAY 28

Equality Operator (continued)

Abstract Equality Operator (==) :-

① If type of left operand and type of right operand is same.

Return the result performing x === y.

② If type are not same, ⊙ x → null ; y → undefined (return true)

⊙ x → undefined; y → null (return true)

③ X → number ; y → String.

09.00

converts (y) to a num & again does the comparison.

10.00

12 == "12" ⟶ 12 == (12) ∴ o/p true.

11.00

④ X → String ; y → Number

12.00

converts (X) to a num & again does the comparison.

01.00

02.00

∴ From ③ & ④ ; if any of the operand is a number and the other is a String. Then the String is being converted to a Number and then the Comparison happens.

03.00

04.00

⑤ 05.00 If x → Boolean.

06.00 Convert (X) to a Number & then the comparison happens.

⑥ 07.00 If y → Boolean.

Convert (Y) to a Number & then the comparison happens.

For ⑤ & ⑥ examples.

true == "1"

↓

1 == ("1")

↑ toNumber

∴ 1 == 1 ≃ true (Ans)

false == "0"

↓

0 == "0"

↑ toNumber ("0") = 0

0 == 0 true (Ans)

(7) X → Number, String, Symbol

09.00

Y → Object we convert (Y) to primitive. and compare again.

10.00

(8) X → Object

11.00

Y → Number, String, Symbol we convert (X) to primitive & compare again.

12.00

(9) Return false.

01.00

null == false → o/p:- false.

02.00

Strict Equality Operator :-

03.00

(1) If types are different, return false.

04.00

(2) If (X) is a Number → x is NaN return false. { NaN is a value, which is not equals to itself. }

05.00

→ y " NaN " "

06.00

imp { NaN === NaN (false) }.

07.00

→ x & y are same no.s return true.

→ x → +0 ; y → -0 return true.

→ return false.

console.log ("NaN" == NaN)

↓

To Number ("NaN") → NaN

∴ 'NaN == NaN (types are same, call for strict equality Opr)

∴ ↓

NaN === NaN ≃ (false) o/p

let obj = { x:10, valueOf() { return 100; } };

(a) console.log (99 == obj)

(b) console.log (100 == obj)

(a) 99 == obj (

Are the types same (Yes/NO) ?

∴ Coercion

99 remains as it is.

obj → toPrimitive (obj)

∵ No preferred type is mentioned.

∴ hint → "default" → hint = number.

↓

Ordinary toPrimitive (obj, "Number")

∵ Hint is "Number". Then,

["valueOF", "toString"]

Foreach loop on the array.

obj.valueOF() → 100 object | Not Object

↑ return 100.

99 == 100 (types are now same. Call for strict equality Operator)

false o/p.

(b). 100 == obj

Are the types same ? (Yes/No)

∴ Coercion.

100 remains as it is.

obj → ToPrimitive (obj)

∵ No, preferred type is mentioned.

∴ hint → "default" → hint = "Number"

↓

Ordinary to Primitive (Object, "Number")

∵ Hint is "Number"

∴ ["valueOf", "toString"]

For each loop on the array.

obj.valueOF() → 100 Object | Not Object ✓

↑ return 100.

∴ 100 == 100 (types are same. Call for Strict Equality Operator)

true o/p

⊙ Strict Equality Operator (continued) :- (Neither X nor Y is a number)

⊙ If X is not a Number then, Same Value Non Number (X, Y) is called.

✉ @ ☎

① (X) is not a num.

② type of (X) same as type of (Y).

③ x → undefined ; y → undefined

return true.

④ x → null value. ie x = null ; y = null
return true if (null === null) → return true.

⑤ x → String.
"abc" === "abc"
Same length & indx by indx comparison.
return true | else return false.

⑥ x → Boolean
x → true | y → true } → return true
x → false y → false }
else, return false.

⑦ x → Symbol
x & y are same symbols → return true
else, return false.

⑧ x & y are same object value → return true
↳ Same memory objects.
else, return false.

let obj1 = {x:10} ; let obj2 = {x:10} ; let obj3 = {y:10}.

console.log (obj1 === obj2) ; → false (diff. memory objects).

console.log (obj1 === obj3) → false (diff. memory objects)

console.log (obj1 === obj1) → true (Same memory objects)

Console.log ({x:10} === {a:10}) → false

This will be created as new object in the memory.

ToString(argument) → Concatenation (+) "" + —

SATURDAY 2017
23rd Week • 154-211
JUNE 03

09.00 undefined → "undefined"

10.00 null → "null"

11.00 Boolean → true → "true"
false → "false"

12.00 Number → NumberToString(argument)

01.00

String → return argument.

Symbol:- TypeError Exception.

Object:- ToPrimitive(arg, hint: String)
↑ toString()

02.00

Special Cases:-

03.00 console.log("" + 0); → 0 (+0 converted to "0")

04.00 console.log("" - 0); → 0 (-0 " " "0")

05.00 console.log("" + []) → [] → "" empty String

06.00 console.log("" + {}) → [object Object].

07.00 console.log("" + [1,2,3]) → 1,2,3.

console.log("" + [null, undefined]) → ,

SUNDAY

console.log("" + [1, 2, null, 4]) → 1, 2, , 4

console.log(0 - "010") → -10

console.log(0 - "010") → NaN.

console.log(0 - 010) → this will get converted to an Octal Num.
o/p :- -8

2017 MONDAY

05 24th Week • 156-209 JUNE

JUNE 2017

WK	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

09.00 console.log (1 - []) → 1

10.00 console.log (0 - []) → 0

11.00 console.log ([] - 1) → -1

12.00 console.log (["a"] - 1) → -1

01.00 If we've to escape coercion, go for ===

02.00 Explicit Conversion:-

03.00 ① Boolean(0) → false

04.00 Boolean("") → false.

05.00 ② String(123) → "123"

06.00 -: NaN :-

07.00 How to check if given I/P has value NaN inside it?

console.log (isNaN(I/P));

isNaN(I/P) :- Does toNumber Abstract Operation.

let I/P = "figo";

console.log (isNaN(I/P)); → ~~false~~. true.

NaN

isNaN(NaN) → true.

Wk	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

TUESDAY 2017

24th Week • 157-208

JUNE 06

Better alternative,

09.00

Number.isNaN('1/p')

10.00

Number.isNan("Figo") → false

11.00

Number.isNan(NaN) → true.