8/5/21 **Hoisting** → Supported for 'var'...

Ex →
① `var marks = 100`
`Console.log(marks)`

`Console.log(marks)` → o/p → undefined
→ 100

② `Console.log(marks)`
`let marks = 100`
`Console.log(marks)`

o/p ↓
→ cannot access 'marks' before initialization.

[same in case of 'const', throws the same error]

→ So, what happens in hoisting is that whenever it will try to run the code line by line, since the var marks would be stored in the memory and since it will not be initialized for the first line it will show us 'undefined'.

* Variables Naming Convention ...

(1) → Variable names can consist only of letters (a-z), (A-Z) numbers (0-9), dollar ($) sign symbol, and underscore (_).

(2) → Variable names can't contain any white space characters (tabs or spaces)

(3) → A Js identifier must start with a letter, (_) or ($).

(4) → There are several reserved Keywords which can't be used as the name of a variable.

(5) → Variable names are case sensitive.

Ex → (1) var abc , var ab2 , var ab$, var ab_

(2) {var cost price → } X

[var cost_price or costPrice] ✓

(3) [var _abc , var $ab] → ✓

{var 2abc → ✗ }

(4) We have 33 Keywords and if try to use them as a name for a variable

it implies that we are trying to change the meaning of those keywords.

Ex → break, finally, class, catch, void, var etc...

* Keywords used in strict mode →

- implements
- interface
- protected
- Package
- private
- yield
- public
- let
- static
- ~~void~~

* Keywords reserved only if it is not used in current module →

→ await.

* Keywords reserved for future.

→ enum.

— × —

* Operators, Overview & Arithmetic Operators →

→ Operators are the one that gives us the power of doing some mathematical calculations, to establish some relationships, or to do some comparisons.

* Kind of Operators →

(1) Arithmetic Operators (+, -, /, %, *, **)
(2) Assignment operator (=)
(3) Comparison operator
(4) Bitwise operator
(5) String operator
(6) Logical operator
(7) Ternary operator
(8) Comma operator
(9) Unary operator.

* Comparison Operators →

→ Equal to → (==)
→ Strict Equal to → (===)
→ Greater than → (>)
→ Less than → (<)
→ Greater than Equal to → (>=)
→ Less than Equal to → (<=)
→ Not Equal to → (!=)
→ Strict not Equal to → (!==)

* Diff. b/w (==) & (===)

⇒ (==) just checks the value, whereas (===) checks the value as well as the data type.

Some in case of (!=) and (!==)