-JavaScript interprets the line break as an “implicit” semicolon. This is called an automatic semicolon insertion.

-Please make sure that "use strict" is at the top of your scripts, otherwise strict mode may not be enabled.

-There is no directive like "no use strict" that reverts the engine to old behavior.

-When you use a developer console to run code, please note that it doesn’t use strict by default.

Sometimes, when use strict makes a difference, you’ll get incorrect results.

So, how to actually use strict in the console?

First, you can try to press Shift+Enter to input multiple lines, and put use strict on top,

- variable should be declared only once.

- A repeated declaration of the same variable is an error:

-The name must contain only letters, digits, or the symbols $ and _.

-case sensitive (apple and APPLE both are different)

-Normally, we need to define a variable before using it. But in the old times, it was technically possible to create a variable by a

mere assignment of the value without using let. This still works now if we don’t put use strict in our scripts

to maintain compatibility with old scripts.

-Variables declared using const are called “constants”. They cannot be reassigned. An attempt to do so would cause an error:

-capital-named constants are only used as aliases for “hard-coded” values.

-JavaScript : “dynamically typed lang"

-The 'number'- type represents both integer and floating point numbers.

-NaN represents a computational error. It is a result of an incorrect or an undefined mathematical operation

-NaN is sticky. Any further mathematical operation on NaN returns NaN

-In JavaScript, the “number” type cannot safely represent integer values larger than (253-1) (that’s 9007199254740991), or less than -(253-1) for negatives.

-Backticks(`---`) are “extended functionality” quotes.

-null->represents “nothing”, “empty” or “value unknown”.

ex. let A = null;

- The meaning of 'undefined' is “value is not assigned.

-If a variable is declared, but not assigned, then its value is undefined

ex) let a;

-undefined is reserved as a default initial value for unassigned things.

-typeof(x), although the typeof x syntax is much more common.(both are same)

-typeof() : Returns a 'string type' with the name of the type, like "string".

-typeof(NaN) -> number

-The …args parameter allows us to collect all remaining arguments into an array, and in Javascript typeof an array is an object.

-The Math.max() method returns -Infinity by default and the Math.min() method returns Infinity value by default when passed without any parameters.

---------- alert, prompt, confirm ----------\

1)alert :

-The mini-window with the message is called a 'modal window'.

-alert returns 'undefine'

-The word “modal” means that the visitor can’t interact with the rest of the page, press other buttons, etc, until they have dealt with the window. In this case – until they press “OK”.

2) prompt :

-The function prompt accepts two arguments:

- syntax: prompt("msg",[default]); (here [] shows that parameter is optional, not required.)

-default is optional

-if user click on cancel button then we get 'null' as the result

3)confirm

- returns true for OK and false for Cancel/Esc.

---------- Type conversion ----------

-If the string is not a valid number, the result of such a conversion is NaN

ex . let age = Number("an arbitrary string instead of a number");

alert(age); // NaN, conversion failed

-any non-empty string is true

- while converting into Number :

'null' becomes zero while 'undefined' becomes NaN

-The unary plus or, in other words, the plus operator + applied to a single value, doesn’t do anything to numbers. But if the operand is not a number, the unary plus converts it into a number.

-ex) : // Converts non-numbers

alert(+true); // 1

lert(+""); // 0

-Bitwise operators treat arguments as 32-bit integer numbers

-alert(2 + 2 + '1'); // "41" and not "221"

-alert('1' + 2 + 2); // "122" and not "14"

-----string comparison

-If both strings end at the same length, then they are equal. Otherwise, the longer string is greater.

->'a'>'A' (lowercase character has a greater index in the internal encoding table JavaScript uses (Unicode))

- An equality check converts values using the numeric conversion (hence "0" becomes 0),

ex) let a = 0;

alert(Boolean(a)); // false

let b = "0";

alert(Boolean(b)); // true

alert(a == b); // true!

-alert(null === undefined); // false

-alert(null == undefined); // true

```
alert( null > 0 );  // (1) false
alert( null == 0 ); // (2) false
alert( null >= 0 ); // (3) true
```

-The value undefined shouldn’t be compared to other values:

---------- Logical Operators ----------

- four logical operators in JavaScript: || (OR), && (AND), ! (NOT), ?? (Nullish Coalescing)
- OR : OR || returns the first truthy value or the last one if no truthy value is found.
- OR || operator is the so-called “short-circuit” evaluation.

--AND &&

-When all values are truthy, the last value is returned:

```
alert( 1 && 2 && 3 ); // 3, the last one
```

--

A double NOT !! is sometimes used for converting a value to boolean type:

```
ex) alert( Boolean("non-empty string") ); // true
alert( Boolean(null) ); // false
```

Nullish coalescing operator '??'

- it treats null and undefined similarly
- select the first value from a list that isn’t null/undefined.

ex)

```
let firstName = null;
let lastName = null;
let nickName = "Supercoder";
// shows the first defined value:
```

alert(firstName ?? lastName ?? nickName ?? "Anonymous"); // Supercoder

---

IMP :

|| : returns the first truthy value.

?? : returns the first defined value.

Inline variable declaration :

the “counter” variable i is declared right in the loop. This is called an “inline” variable declaration. Such variables are visible only inside the loop.

ex)

```
for (let i = 0; i < 3; i++) {
  alert(i); // 0, 1, 2
}
alert(i); // error, no such variable
```

FUNCTIONS

- function always gets a copy of the value, so changes made in the arguments are not visible out of the function block

-parameter : function declaration

- argument : function call

-if argument is not provided, then the corresponding value becomes 'undefined'

-If a function does not return a value, it is the same as if it returns undefined:

-An empty return is also the same as return undefined:

```
function doNothing() { /* empty */ }
alert( doNothing() === undefined ); // true
```

-In JavaScript, a function is a value,

-function has global scope(you can declare function any where in the script)

- function expression has no global scope, we need to declare the function first,and then can be called

---

Multi line arrow function :

-curly braces require a return within them to return a value

ex) let sum = (a,b) =>{

let result = a+b;

return result;

}

NUMBERS

-If we want to call a method directly on a number, like toString in the example above, then we need to place two dots .. after it.

ex) alert(123456..toString(36)); // 2n9c

- Math.floor : Rounds down: 3.1 becomes 3, and -1.1 becomes -2.

-Math.ceil : Rounds up: 3.1 becomes 4, and -1.1 becomes -1.

- The method toFixed(n) rounds the number to n digits after the point and returns a string representation of the result.

ex)let num = 12.34;

alert(num.toFixed(1)); // "12.3"

-If a number is really huge, it may overflow the 64-bit storage and become a special numeric value Infinity:

ex) alert(1e500); // Infinity

-alert(NaN === NaN); // false

isFinite(value) converts its argument to a number and returns true if it’s a regular number, not NaN/Infinity/-Infinity:

alert(isFinite("15")); // true

alert(isFinite("str")); // false, because a special value: NaN

alert(isFinite(Infinity)); // false, because a special value: Infinity

--object.is

-Values 0 and -0 are different: Object.is(0, -0) === false

Object.is(NaN, NaN) === true

-alert(parseInt('100px')); // 100 : return interger value

alert(parseFloat('12.5em')); // 12.5 : return floting value

-Math.random() : Returns a random number from 0 to 1 (not including 1).

---------- strings ----------

let str = `Hello`;

alert(str[-2]); // undefined :

The square brackets always return undefined for negative indexes, for instance:

alert(str.at(-2)); // l

-str.indexOf('char') : returns the position where the match was found or -1 if nothing can be found.

-there are 3 methods in JavaScript to get a substring: substring(start,end), substr(start,length) and slice(start,end)

.

-str.substr(start [, length])

Returns the part of the string from start, with the given length.

ARRAY

- declaration :

let arr = new Array();

let arr = [];

-shift

Extracts the first element of the array and returns it:

ex)

let fruits = ["Apple", "Orange", "Pear"];

alert(fruits.shift()); // remove Apple and alert it

alert(fruits); // Orange, Pear

-unshift

Add the element to the beginning of the array:

```
let fruits = ["Orange", "Pear"];


fruits.unshift('Apple');


alert( fruits ); // Apple, Orange, Pear
```

-Methods push and unshift can add multiple elements at once:

-Methods push/pop run fast, while shift/unshift are slow.

- simplest way to clear the array is: arr.length = 0;

-If new Array is called with a single argument which is a number, then it creates an array with given length

ex)let arr = new Array(2);

alert(arr.length); // length 2

```
alert( [] == [] ); // false

alert( [0] == [0] ); // false


alert( 0 == [] ); // true

alert('0' == [] ); // false
```

-arr.splice(start,no_of_values_to_be_removed,values to be inseert at start) : returns the array of removed elements:

- arr.concat() : creates a new array that includes values from other arrays

---

str.split(delimeter) method does exactly that. It splits the string into an array by the given delimiter delim

The split method has an optional second numeric argument – a limit on the array length. If it is provided, then the extra elements are ignored.

MAP AND WEAKMAP

the first difference between Map and WeakMap is that keys must be objects, not primitive values:

-WeakMap does not support iteration and methods keys(), values(), entries(), so there's no way to get all keys or values from it.

-new Date(year, month, date, hours, minutes, seconds, ms)

GARBAGE COLLECTION :

-The main concept of memory management in JavaScript is reachability.

The basic garbage collection algorithm is called "mark-and-sweep".

Or get symbol.description property to show the description only:

---

symbol

// read from the global registry

let id = Symbol.for("id"); // if the symbol did not exist, it is created

// read it again (maybe from another part of the code)

```
let idAgain = Symbol.for("id");

// the same symbol
alert( id === idAgain ); // true
```

-Symbol.for(key) : create symbol

Symbol.keyFor(sym): gives the name of symbol