Generators

→ A generator is a special kind of function that was introduced in ES6. In Javascript, once we execute a function, it has to be executed entirely. But, generator functions enable us to create functions that another code can enter multiple times. Nothing from outside of the generator function can make it pause. Generator function

Date

pauses itself when it runs into a yield expression on its own. Something from outside has to continue its execution. Another important difference b/w generators and normal functions is that generator functions can produce multiple values during its execution. Hence, they can generate a sequence of values, not all at once, but on a per request basis. At every request, the generator function gives us a value until it reaches the end of its execution. Once that happens, the done flag will be set to true. ✓

Syntax:

The syntax of declaring the generator function is quite similar to traditional functions. We declare a generator function by using the * (asterisk) operator after the function keyword:

```
FUNCTION* MYGENERATOR() {
    //CODE
}
```

YIELD:

The yield keyword pauses the generator

function execution, and the value of the expression following the yield keyword is returned to the generator's caller. It acts as a generator-based version of the return keyword. In the following example, to pause the generator's execution, and we use the statement yield:

```
FUNCTION* AWESOMEGENERATOR() {
    YIELD 'HELLO WORLD' // He pause the
                           execution here.
    CONSOLE.LOG('WE ARE BACK AGAIN') // When
                    We resume, we are here
}
```

NEXT() method:

A generator gives us the next() method, which is used to resume the execution. This method returns an object with two properties. These are value & done:

```
{
  VALUE : [NEXT VALUE],
  DONE : [TRUE IF WE REACH THE END, ELSE
          FALSE]
}
```