Iterators and Generators in Javascript

Mokshad Ketan Sankhe

Iterators:

Iterators are objects that allow you to traverse through a collection, like arrays or objects, one element at a time. They provide a simple way to iterate over any collection that implements the iterator protocol. The iterator protocol requires the collection to have a next() method that returns the next element in the sequence.
This next() method returns an object with two properties:

- value - the next value in the iteration
- done - a boolean indicating whether the iteration is complete

To create an iterator object, you call the Symbol.iterator method on the collection:

```
const arr = [1, 2, 3];
const iterator = arr[Symbol.iterator]();
```

Now you can call next() on the iterator to get the next value:

```
iterator.next(); // { value: 1, done: false }
iterator.next(); // { value: 2, done: false }
iterator.next(); // { value: 3, done: false }
iterator.next(); // { value: undefined, done: true }
```

When done is true, it means the iteration is complete.

Generators:

Generators are a special type of function that works as an iterator. They allow you to define an iterative algorithm by writing a single function whose execution is not continuous.
Generator functions use the function* syntax and yield keywords:

```
function* generatorFn() {
  yield 1;
  yield 2;
  yield 3;
}
```

When you call a generator function, it returns an iterator object. You can use next() to iterate over the yields:

```
const generator = generatorFn();
generator.next(); // { value: 1, done: false }
generator.next(); // { value: 2, done: false }
generator.next(); // { value: 3, done: false }
generator.next(); // { value: undefined, done: true }
```

This allows you to pause execution and resume later, which is useful for iterative processes like waiting on promises.

In summary, iterators provide a simple way to iterate over collections while generators allow you to define iterative algorithms through functions. Both are useful tools in JavaScript for working with sequences and collections.