

Assignment No-4b

AIM: Write a program on inheritance , iterators , generators

LO MAPPED:- LO4

THEORY:- The ES6 JavaScript supports Object-Oriented programming components such as Object, Class and Methods. Further in Classes we can implement inheritance to make child inherits all methods of Parent Class. This can be done using the extends and super keywords. We use the extends keyword to implement the inheritance in ES6. The class to be extended is called a base class or parent class. The class that extends the base class or parent class is called the derived class or child class. The super() method in the constructor is used to access all parent's properties and methods that are used by the derived class.

ES6 introduces a new mechanism for traversing data: iteration. Two concepts are central to iteration:

An iterable is a data structure that wants to make its elements accessible to the public. It does so by implementing a method whose key is Symbol.iterator. That method is a factory for iterators. An iterator is a pointer for traversing the elements of a data structure (think cursors in databases) Iterable values in Javascript

The following values are iterable:

Arrays

Strings

Maps

Sets DOM data structures (work in progress)

Plain objects are not iterable.

Symbols offer names that are unique and cannot clash with other property names. Also, Symbol.iterator will return an object called an iterator. This iterator will have a method called next which will return an object with keys value and done.

Prior to ES6, functions in JavaScript followed a run-to completion model.

- ES6 introduces functions known as Generator which can stop midway and then continue from where it stopped.

- A generator prefixes the function name with an asterisk * character and contains one or more yield statements.

- The yield keyword returns an iterator object

1)Code for GENERATOR :-

```
<html>
<head>
  <title>generator</title>
</head>
<body>
<script>
  //generator
  console.log('generator function used')
  function* fun() {
    yield 10;
    yield 20;
```

```

}
let gen = fun();
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
</script>
</body>
</html>

```

Output:-

The screenshot shows a Replit environment with a file named `index.js` and a console window. The code in `index.js` is as follows:

```

1 //generator
2 console.log('generator function used')
3 function* fun() {
4   yield 10;
5   yield 20;
6 }
7 let gen = fun();
8 console.log(gen.next().value);
9 console.log(gen.next().value);
10 console.log(gen.next().value);
11

```

The console output shows the following:

```

generator function used
10
20
undefined
Hint: hit control+c anytime to enter REPL.
>

```

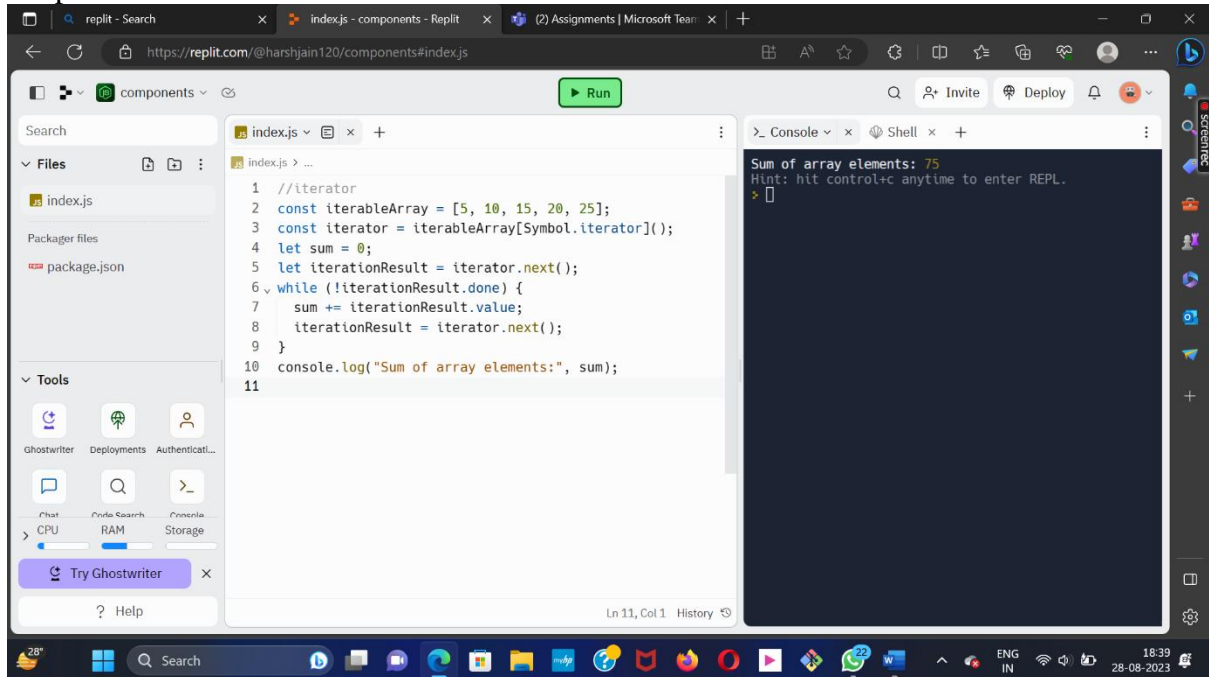
2)Code for ITERATOR :-

```

<html>
<head>
  <title>iterator</title>
</head>
<body>
<script>
//iterator
const iterableArray = [5, 10, 15, 20, 25];
const iterator = iterableArray[Symbol.iterator]();
let sum = 0;
let iterationResult = iterator.next();
while (!iterationResult.done) {
  sum += iterationResult.value;
  iterationResult = iterator.next();
}
console.log("Sum of array elements:", sum);
</script>
</body>
</html>

```

Output:-



The screenshot shows a Replit IDE interface. On the left, there's a file explorer with 'index.js' and 'package.json'. The main editor displays the following JavaScript code:

```
1 //iterator
2 const iterableArray = [5, 10, 15, 20, 25];
3 const iterator = iterableArray[Symbol.iterator]();
4 let sum = 0;
5 let iterationResult = iterator.next();
6 while (!iterationResult.done) {
7   sum += iterationResult.value;
8   iterationResult = iterator.next();
9 }
10 console.log("Sum of array elements:", sum);
11
```

The console on the right shows the output: 'Sum of array elements: 75'. Below the output, it says 'Hint: hit control+c anytime to enter REPL.' and there's a cursor.

3)Code for INHERITANCE:-

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Class Inheritance</h2>
<p id="demo"></p>

<script>
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  atWork() {
    return this.name + " is at work, ";
  }
  atHome() {
    return this.name + " is at home";
  }
  sleeping() {
    return this.name + " is sleeping";
  }
}
class SoftwareEngineer extends Person {
  constructor(name, age) {
    super(name, age);
  }
  profession() {
    return this.name +
    " is a Software Engineer";
  }
}
```

```

doTasks() {
return super.atWork() + this.profession();
}
}
function display(content) {
console.log(content);
}
const character =
new SoftwareEngineer("Piyush", 30);
display(character.profession());
display(character.atHome());
display(character.doTasks());

```

</script>

</body>

</html>

Output:-

The screenshot shows a Replit IDE window with the following content:

- File Explorer:** Shows a file named `index.js` and a `package.json` file.
- Code Editor:** Contains the following JavaScript code:


```

17 constructor(name, age) {
18   super(name, age);
19 }
20 profession() {
21   return this.name +
22     " is a Software Engineer";
23 }
24 doTasks() {
25   return super.atWork() + this.profession();
26 }
27 }
28 function display(content) {
29   console.log(content);
30 }
31 const character =
32   new SoftwareEngineer("Harsh", 30);
33 display(character.profession());
34 display(character.atHome());
35 display(character.doTasks());
36
37

```
- Console:** Shows the output of the code:


```

Harsh is a Software Engineer
Harsh is at home
Harsh is at work, Harsh is a Software Engineer
Hint: hit control+c anytime to enter REPL.
>

```

REFERENCES :- 1) <https://www.geeksforgeeks.org/how-to-implement-inheritance-in-es6/> 2) <https://www.qed42.com/insights/coe/javascript/implementing-iterators-andgenerators-javascript>

CONCLUSION :- Learnt about Iterators , Generators their basic syntax and functioning in JavaScript , also learnt about inheritance in JavaScript , use of constructor , super keyword , extends in JavaScript