

PRACTICAL NO 5

Date of Performance: 15-09-2022

Software Requirement: Visual Studio Code, Notepad, Notepad++, NodeJS

Aim: To use JavaScript to write codes based on normal, arrow, anonymous and generator functions .

Objectives: The aim of this experiment is that the students will be able to:

- ★ Implement JavaScript codes based on functions.
- ★ Understand different features of JavaScript functions & its usage in code.

Outcomes: After study of this experiment, the students will be able to:

- ★ Write and run functions in JavaScript.
- ★ Implement the usage of different functions in JavaScript.

Prerequisite: Basic knowledge of JavaScript syntax required.

Theory:

JavaScript is the world most popular lightweight, interpreted compiled programming

language. It is also known as scripting language for web pages.

Function in any programming language is the basic building block to create and combine the related bits of code. Every programming language provides certain kinds of practices to write any function.

Syntax of regular functions:-

```
let x = function function_name(parameters){  
  // body of the function  
};
```

ES6 introduced a new and shorter way of declaring an anonymous function, which is known as Arrow Functions. In an Arrow function, everything remains the same, except here we don't need the function keyword also. Here, we define the function by a single parenthesis and then '=>' followed by the function body. The arrow function syntax is one of the most used and efficient ones to create a function in JavaScript. To write the arrow function, simply create any variable it can be const, let, or var but always do prefer with const to avoid unnecessary problems.

Syntax:

```
let myFunction = (arg1, arg2, argN) => {  
  statement(s)
```

```
}
```

Anonymous Function is a function that does not have any name associated with it. Normally we use the function keyword before the function name to define a function in JavaScript, however, in anonymous functions in JavaScript, we use only the function keyword without the function name. An anonymous function is not accessible after its initial creation, it can only be accessed by a variable it is stored in as a function as a value. An anonymous function can also have multiple arguments, but only one expression.

Syntax:

```
let x = function () {  
  //statements  
};  
x();
```

A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. The yield statement suspends function's execution and sends a value back to caller, but retains enough state to enable function to resume where it is left off. When resumed, the function continues execution immediately after the last yield run.

Syntax :

```
function* gen(){  
  yield 1;  
  yield 2;  
  ...  
}
```

Problem Statement:

Implement functions in JS (Arrow, Normal & Anonymous) & generator function.

Code:

```
function add() {  
  let a = 8,b = 2;  
  console.log(` The addition using normal function without parameter is ${a +  
  b}` );  
}  
add();  
function add2(a, b) {  
  console.log(` The addition using normal function with parameter is ${a + b}` );  
}  
add2(45, 67);  
const arrowAdd = () => {  
  let a = 100,b = 25;  
  console.log(` The addition using arrow function without parameter is ${a +  
  b}` );  
};  
arrowAdd();
```

```
const arrowAdd2 = (a, b) => {  
  console.log(` The addition using arrow function with parameter is ${a + b}` );  
};  
arrowAdd2(90, 35);  
const anonymousAdd = function () {  
  let a = 52, b = 78 ;  
  return ` The addition using anonymous function without parameter is ${a +  
b}` ;  
};  
let x = anonymousAdd();  
console.log(x);  
const anonymousAdd2 = function (a, b) {  
  return ` The addition using anonymous function with parameter is ${a + b}` ;  
};  
let y = anonymousAdd2(42, 8);  
console.log(y);
```

Output:

```
PS G:\html> node --version
```

```
v16.17.0
```

```
PS G:\html> node exp5.js
```

```
The addition using normal function without parameter is 10
```

```
The addition using normal function with parameter is 112
```

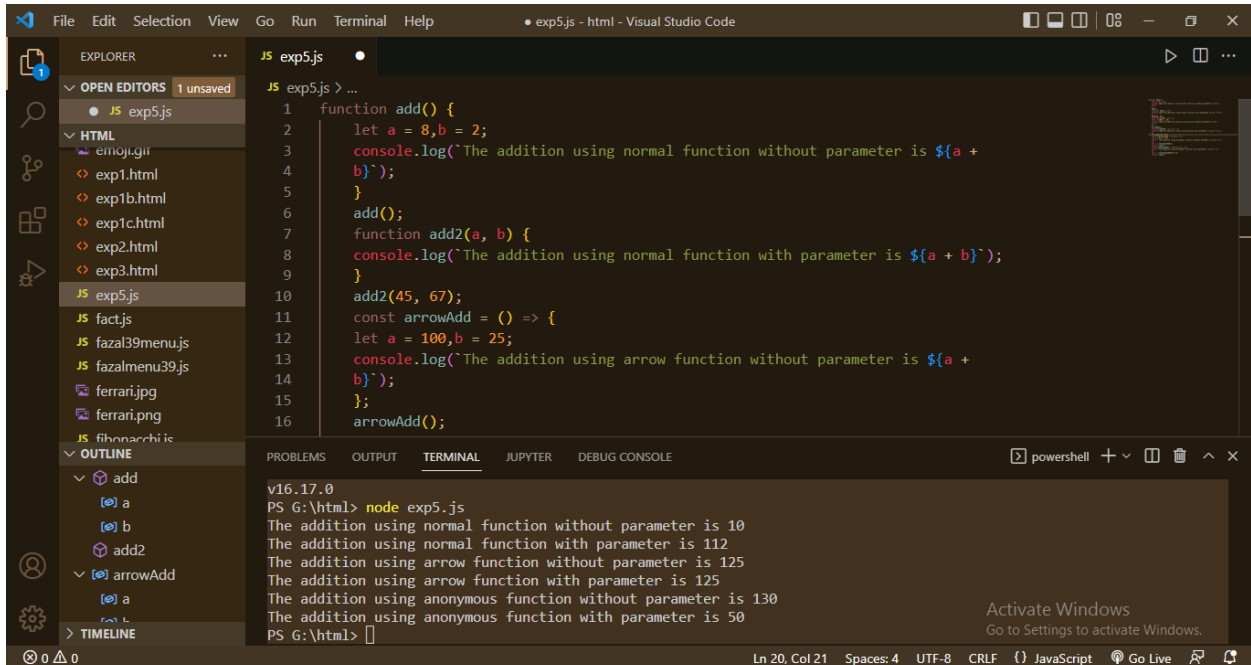
```
The addition using arrow function without parameter is 125
```

```
The addition using arrow function with parameter is 125
```

```
The addition using anonymous function without parameter is 130
```

```
The addition using anonymous function with parameter is 50
```

```
PS G:\html>
```



```
1 function add() {
2   let a = 8, b = 2;
3   console.log(`The addition using normal function without parameter is ${a +
4   b}`);
5 }
6 add();
7 function add2(a, b) {
8   console.log(`The addition using normal function with parameter is ${a + b}`);
9 }
10 add2(45, 67);
11 const arrowAdd = () => {
12   let a = 100, b = 25;
13   console.log(`The addition using arrow function without parameter is ${a +
14   b}`);
15 };
16 arrowAdd();
```

Terminal Output:

```
v16.17.0
PS G:\html> node exp5.js
The addition using normal function without parameter is 10
The addition using normal function with parameter is 112
The addition using arrow function without parameter is 125
The addition using arrow function with parameter is 125
The addition using anonymous function without parameter is 130
The addition using anonymous function with parameter is 50
PS G:\html>
```

Source Code: Generator function:

```
function* generator(x) {
  yield x;
  yield x * 5;
  yield x * 3;
  yield x + 2;
  yield x - 8;
}

const gen = generator(39);
console.log("Output for each yield is:")
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);
```

Output:

```
PS G:\html> node --version
```

```
v16.17.0
```

```
PS G:\html> node exp52.js
```

Output for each yield is:

```
39
```

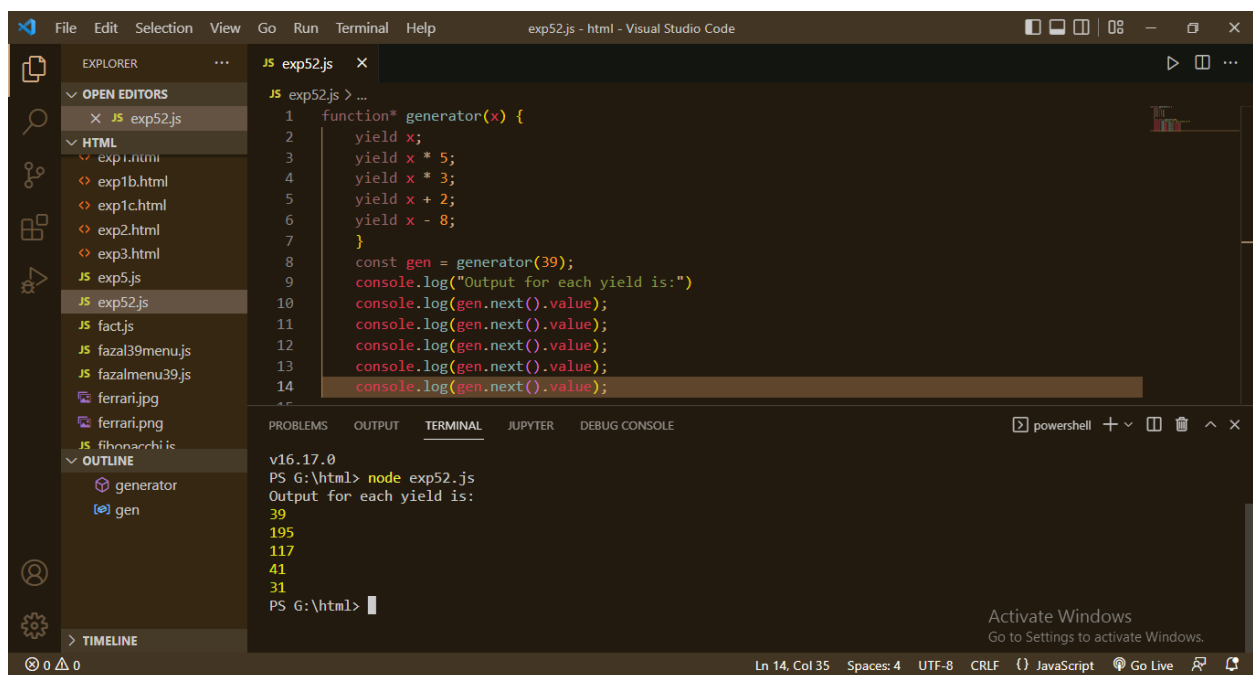
```
195
```

```
117
```

```
41
```

```
31
```

```
PS G:\html>
```



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a JavaScript file named `exp52.js` containing a generator function `generator(x)` with five `yield` statements. The terminal shows the command `node exp52.js` being executed, and the output displays the values of the yields: 39, 195, 117, 41, and 31.

```
JS exp52.js
1 function* generator(x) {
2   yield x;
3   yield x * 5;
4   yield x * 3;
5   yield x + 2;
6   yield x - 8;
7 }
8 const gen = generator(39);
9 console.log("Output for each yield is:");
10 console.log(gen.next().value);
11 console.log(gen.next().value);
12 console.log(gen.next().value);
13 console.log(gen.next().value);
14 console.log(gen.next().value);
```

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
v16.17.0
PS G:\html> node exp52.js
Output for each yield is:
39
195
117
41
31
PS G:\html>
```

Conclusion:

From this experiment we understand that JavaScript is a very powerful web language and is used at a large scale. We have learned how to implement JavaScript normal, arrow and anonymous

functions/generator functions in programs and understood it's syntax.

Performance: 7M	Journal: 3M	Lab Ethics: 2M	Attendance: 3M	Total: 15M	Faculty Sign