

Upskill Report

2nd Week of JavaScripts

1st session

Topic:

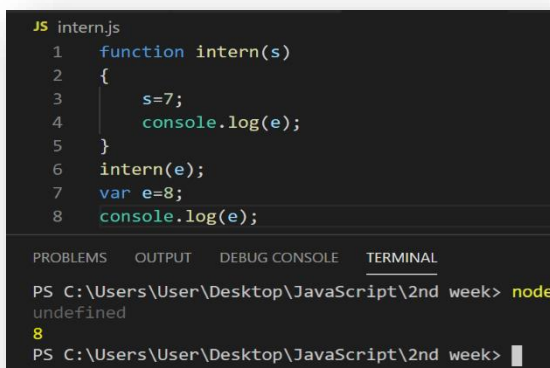
1. Function explanation and definition
2. Recursive function
3. First class citizen
4. Function assignment(expression)
5. Difference between declaration and expression.

1. Function explanation and definition

Function explanation and definition can be declare before define. Scope of the function is functional scope.

Function declaration gets hoisted (function gets called before it is defined) and function declaration has a name to function.

In JavaScript only ES5 and function declaration get hoisted.



```
JS intern.js
1  function intern(s)
2  {
3      s=7;
4      console.log(e);
5  }
6  intern(e);
7  var e=8;
8  console.log(e);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

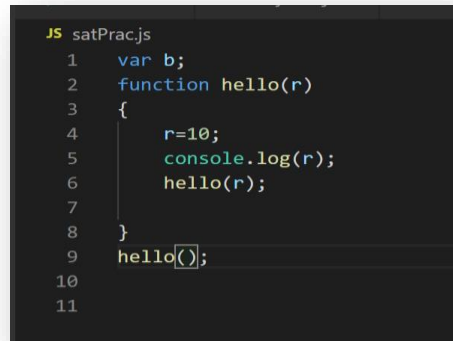
```
PS C:\Users\User\Desktop\JavaScript\2nd week> node
undefined
8
PS C:\Users\User\Desktop\JavaScript\2nd week>
```

In the image function intern is a function explanation and definition.

2. Recursive function:

The function whoes calling itself is known as recursive function.

Example:



```
JS satPrac.js
1  var b;
2  function hello(r)
3  {
4      r=10;
5      console.log(r);
6      hello(r);
7  }
8
9  hello();
10
11
```

3. First class citizen:

In JavaScript functions are called **first class citizen** it means that we can do anything with function.

- We can pass value to function
- Can be acts as a class.
- Function has **key : value** pair.
- We can pass function as an argument
- It is a part of variable
- Function can acts as an object

First class citizen can go anywhere and can do anything.

It can be assign to variable, pass as a parameter to function and return from function.

Function is non-primitive and a kind of variable which holding reference to function.

4. Function assignment(expression):

- When we create function which assign to variable called as function expression. Function expression works like function declaration only difference is that it can not has a name.
- It is anonymous function which has no name or it is also known as unnamed function.
- Function expression does not get hoisted

Example :

```
JS satPrac.js
1  var express = function(x,y)
2    {
3      return x * y;
4    };
5  var ans = express(4,6);
6  console.log(ans);
7
```

5. Difference between declaration and expression:

Function declaration	Function expression
Function can call before definition	Function expression can not call before definition.
Function declaration gets hoisted	Function expression does not get hoisted.
Function declaration has a name to a function	Function expression doesn't have name to function
	Also known as anonymous function or unnamed function.
Example : Function declr() //fun declaration { Console.log("something"); } declr(); //calling function	Example : var multi = function (x , y) //function expression { return x * y; } Console.log(multi(2,7)); //function called

*Note: Nowadays function declaration is rarely used. Function expression are mostly use in industry.
Only ES5 and function declaration gets hoisted in JavaScript.*

2nd Session

Topic:

1. Learn more about functions
2. Passing reference to function
3. Return statement

1. Learn more about function:

Learn about function parameter calling functions declaring function

```
JS satPrac.js
1  const first = function (params)
2  {
3      return params ;
4  }
5  first(10);
6
```

Above example we are passing one parameter to function and function return value of that parameter but above function will not display any output because return value is not caught in any variable so it will not display output.

```
13
14
15  const first = function (params) {
16      return params ;
17  }
18  var ans = first(10);
19  console.log(ans);
20
```

The above code display the output because here the return value is caught in ans variable so it prints 10.

```

14
15  const first = function (params) {
16      return params + 9;           // 10 + 9 = 19
17      return params + null;        // 10 + 0 = 10
18      return params + "SOMENAME"; // it make concatenation "10SOMENAME"
19      return params + {} ;         // 10 [object Object]
20  }
21  var ans= first(10);
22  console.log(ans);
23

```

we can use like this also, in above code we can make addition with return value, concatenate some string with value etc.

2. Passing reference to function:

Referring one function to another function by passing it as a parameter.

Example :

```

JS satPrac.js
1  const first = function(params) //ref to second function
2  {
3      return params;
4  }
5
6  const second = function(name)
7  {
8      console.log("ROHEAT");
9  }
10 }
11
12 const ans = first(second); //calling first function
13 //and passing second function as a param
14 console.log(ans);          // ans contain second function and it is type of function
15

```

3. Return statement:

In JavaScript when return statement used in function body the execution of function is stopped.

Example :

```
JS satPrac.js
1  const ret = function () {
2      var inside=function()
3      {
4          console.log("Before"); //it will gets executed
5
6          return 1;
7          console.log("after"); // it will not get executed because
8      }; // the statement is written after return
9          // statement and return stopped the execution of function
10     return inside;
11 }
12
13 const value = ret();
14 value();
15 console.log(value);
16
```

Note: in javascript you can return from anywhere and it can return primitive as well as non primitive.

3rd Session

Topic:

1. Higher Order Function
2. IIFE (Immediate invoke Function Expression)
3. Inline function
4. Arrow function

1. Higher Order Function:

- A higher order function is a function that take another function as an argument or that return a function as a result.
- Function is a First class citizen and function is an object.
- Used to wrapping different function and return a new function.

Example:

```
1 function one(params) {  
2   var name="Roheat";  
3  
4   function infun(params) {  
5     console.log("Name",name); //Name Roheat  
6   }  
7  
8   return infun(params);  
9 }  
10 one();
```

2. IIFE (Immediate invoke Function Expression):

- Immediate invoked function expression (IIFE) is a self executing function is an anonymous function expression that invoked immediately.
- IIFE function defining and calling function itself.
- IIFE function that runs as soon as it is defined.
- It executes only once and it is used to avoid collision.

Syntax: (function () {}) ();

Example:

```
1 const output = (function(){  
2   var name = "ROHEAT";  
3   return name;  
4 }());  
5 console.log("Name : ", output); //immediately create the output  
6 //Name : ROHEAT  
7
```

```

1  const IIFE = function()
2  {
3      console.log("inside");
4  }
5  }
6  ;(function (params) {
7      console.log("IIFE function");
8      params();
9  })
10 })(IIFE);

```

Note: it is a feature of IIFE function that use ; (semicolon) in front of function.

3. Inline function

Function within define within a function call and it is not accessible outside the function call.

```

1  const car = function (params) {
2      const inside = function()
3      {
4          console.log("Car Name: ", params ); //prints Car name: BMW
5      }
6      return inside;
7  }
8
9
10 const value = car("BMW"); //calling function with passing parameter as string
11
12 value();

```

In the above code we cannot call inside function directly because it is a inline function. Without calling car function we cannot achieve inside function.


```

1  ;(function (params) {
2      console.log("param");
3      let neww = 30;
4      params[neww];
5  })(function(data){
6      console.log("called IIFE",data);
7  });
8
9
10

```

Note: Here we can achieve security to inside function.

4. Arrow function:

- It is introduced in ES6.
- Arrow function is a shorter way of writing function in javascript.
- Need to “=>” arrow to use arrow functionality.
- It is always anonymous.

Example :

Small code Without using arrow function:

```

10  const arrow = function (x,y)
11  {
12      return x + y;
13  }
14
15  var add = arrow(8,9); //calling arrow fun and passing 2 param to function
16  console.log(add); // 17
17

```

Small code with using arrow function:

```
1  const arrow = (x,y) => //using arrow function
2  {
3      return x + y;
4  }
5
6  var add = arrow(8,9); //calling arrow function passing to parameter
7  console.log(add);
8
```

4th Session

Topic:

1. Javascript object
2. Prototypes in javascript
3. This keyword.

1. Javascript Object:

- In javascript everything is an object so function is also object.
- Object is also hold function.
- In object when we write function it is always expression (key:value pair) key should be always string and value can be anything
- To define object inside function instead of = operator : (colon) will get use.
- Object can hold function because it is first class citizen it means it can take function and return function.

Example:

```
1  const person={
2    name: "ROHEAT",
3    age: 29,
4    hobbies: ["Riding Bike","Playing volleyball","travelling"],
5
6    getdata : function () {
7      console.log(person.name,"likes",person.hobbies);
8    }
9  };
10
11  person.getdata();
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\JavaScript\2nd week> node .\satPrac.js
OHEAT likes [ 'Riding Bike', 'Playing volleyball', 'travelling' ]
PS C:\Users\User\Desktop\JavaScript\2nd week>
```

In the above code getdata is a object which points to anonymous function which prints information of person like name and hobbies.

```
1  const person={
2    name: "ROHEAT",
3    age: 29,
4    hobbies: ["Riding Bike","Playing volleyball","travelling"]
5
6    getdata : function () {
7      return function()
8      {
9
10     }
11   }
12 };
13
14 const data = person.getdata();
15 //person.getdata();
16 console.log(data);
17
```

As I mentioned some points above, the code shows object getdata contains function which return a new function it is possible in javascript because function is first class citizen.

2. Javascript Prototypes:

- Javascript is a prototype based language so, whenever we create a function using javascript, javascript engine adds a prototype property inside function, prototype is basically an object.
- It is also known as a object, to add method in function(object variable) it become object prototype.
- In javascript object prototype only valid in a method it is not valid in function.

Example:

```
var arr=[3,7,6,8,23,76];  
console.log( arr.reverse()); // [ 76, 23, 8, 6, 7, 3 ]
```

3. This keyword:

- This is a reference variable that refers to the current invoking object.
- In javascript this keyword refers to the object.
- This keyword always refers to global scope only.
- If the function used within method then this keyword always refers to global object.
- This is a object reference variable.
- We can read object by this keyword to print any value we use this keyword inside object.

Example:

```
name="Roheat";  
function a1()  
{  
    console.log(this.name); //access a global variable  
                           // prints Roheat  
}  
a1();  
}
```

5th Session

Topic:

1. In this session we learn more about Inline functions, IIFE function and overview and understanding with more example of function.
2. IIFE calling shortcut (short circuit)
3. Function chaining

In this session we only understand about function and done lots of examples on function with all the possibilities like noticing local scope , global scope, lexical scope, passing parameters to function, passing reference of another function as a parameter, returning function, inline function etc.

Function chaining:

```
22  const car = function()
23  {
24      return {
25          name:"BMW",
26          getname: function()
27          {
28              return {
29                  name:"audi",
30                  getname:function()
31                  {
32                      return "TATA";
33                  }
34              }
35          }
36      }
37  }
38
39  const cardetails = car(); // { name: 'BMW', getname: [Function: getname] }
40  const cardetails = car().getname(); // { name: 'audi', getname: [Function: getname] }
41  const cardetails = car().getname().getname(); // TATA
42  console.log(cardetails);
```

6th Session

Topic:

1. How to read a code
2. How to write pseudo code from any given expression
3. Brackets notation
4. Json
5. Built in prototype in javascript

1. How to read code:

Example:

```
const car =  
  {  
    model:  
    {  
      type:  
      {  
        fourwheeler:  
          function(tyre)  
          {  
            Return  
            {  
              dashboard:  
                function(dash)  
                { }  
            }  
          }  
        }  
      }  
    }  
  }
```

Car is an object, which has a key model, which is also an object type, which is an object again and have a key fourwheeler which is a method and takes tyre as a parameter, and return another object dashboard which has a key dashboard which is again a method and takes a dash as a parameter.

2. How to write pseudo code from any given expression

Const split = name.spilt(“ ”).join(“ “).toUpperCase();

```
Const name = {  
  Split: function(dm)  
  {  
    Return  
      {  
        Join: function(dml)  
        {  
          Return  
            {  
              toUpperCase: function()  
              {  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

3. Brackets notation:

Brackets notation value is always string to read value if we have space in between two words and that is key then in javascript it will not work with .(dot) notation that's why we use bracket notation.

Example:

```
{  
  Likes:  
    Meta data:  
    {  
    }s  
}
```

4. Json :

- Json stands for javascript object notation.
- Json is used to transfer data from one application to another.
- Json is purely data which hold json object.
- Json start with { } , key should always string and in “ “
- Json cant have method.

Javascript object:

```
Var r = {  
    Name: "ROHEAT",  
    Age : 29,  
}
```

Json object:

```
{  
    "name" : "ROHEAT"  
    "age"   : "29"  
}
```

- JavaScript engine work inside and json work outside.
- In JavaScript, for every last line separated with (,) comma and In json we wont use comma(,) , if comma used it will gives error.

It has a two predefined methods json.stringify and json.parse

- Json.stringify method JavaScript object used to convert JavaScript object to json object.

```
Var js = {  
    Name: "rohit"  
    City : "nagpur"  
}  
Const exe = json.stringify(js);
```

- Json.parse method is used to convert json object to javascript object

```
Var js = {  
    Name: "rohit"  
    City : "nagpur"  
}  
Const exe = json.parse(js);
```


5. Builtin prototype in javascript:

- Object prototype
- Array prototype
- Function prototype
- String prototype
- Boolean prototype
- Number prototype
- Math prototype
- regEx prototype