JavaScript

Lesson 11: JavaScript Functions

Objective

- At the end of this session participants will be able to
  –
    - Create and use anonymous function
    - Create and use Closures
    - Serialize and deserialize JavaScript Object using JSON

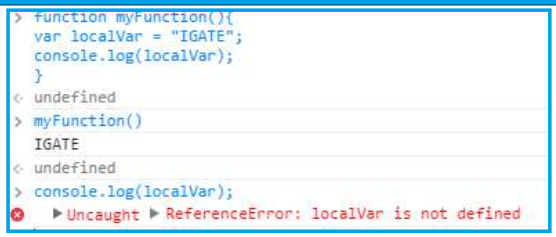
Agenda

- JavaScript Variable Scope
- JavaScript Functions
- Working with JavaScript Functions
- JSON Object
- JSON.stringify and JSON.parse

JavaScript Variable Scope

- Local Variables –
  - Variables declared within a JavaScript function, become **Local** to the function.
  - Local variables have local scope i.e. They can only be accessed within the function.
  - Local variables are created at the beginning of function , and deleted at the end of function .

```
function myFunction() {
    var  localVar = "iGate";
    console.log (localVar);  // code here can use localVar
}
    console.log (localVar);  // code here can not use localVar
```



JavaScript Variable Scope (Contd.)

- Global Variables -
  - A variable declared outside a function, is Global
  - A global variable has global scope i.e. all scripts and functions on a page can access it.

```
var  globalVar = "IGATE";
function myFunction() {
      console.log (globalVar );  // code here can use globalVar
}
    console.log (globalVar ); // code here can use globalVar
```

```
> var globalVar = "IGATE";
< undefined
> function myFunction(){
  console.log(globalVar);
  }
< undefined
> myFunction()
  IGATE
< undefined
> console.log(globalVar);
  IGATE
```

JavaScript Variable Scope (Contd.)

- Auto Global Variables -
  - If you assign a value to a variable that has not been declared, it will automatically become a Global variable.

```
function myFunction() {
    autoGlobalVar = "IGATE";
    console.log (autoGlobalVar);  // code here can access Variable
}
  console.log (autoGlobalVar ); // code here can access Variable
```

```
> function myFunction(){
  autoGlobalVar = "IGATE";
  console.log(autoGlobalVar);
  }
< undefined
> myFunction();
  IGATE
< undefined
> console.log(globalVar);
  IGATE
```

Add instructor notes here.

JavaScript Functions

- JavaScript treats functions as objects(first-class functions).
- In JavaScript functions can be instantiated, returned by other functions, stored as elements of arrays and assigned to variables.
- A function with no name is called an anonymous function.
- Closure is a function to which the variables of the surrounding context are bound by reference.
- JavaScript function acts as a constructor when we use it together with the new operator

Add instructor notes here.

Working with JavaScript Functions

- Declaring the function anonymously

```
function(){
    console.log(`IGATE');
}
```

- Invoking the anonymous function. Function executes immediately after declaration.

```
(function(){
    console.log(`IGATE');
})();
```

Add instructor notes here.

Working with JavaScript Functions

- Declaring a named function. function doSomething will be available inside the scope in which it's declared.

```
function doSomething(){
      console.log('IGATE');
}

/* Inner Scope */
(function(){
      doSomething();
})();
```

- Assigning function to a variable.

```
var doSomething = function(){
      console.log('IGATE');
}
```

Add instructor notes here.

Working with JavaScript Functions

```
/*Anonymous Closures*/
(function(){
    var data = "Closing the variables inside the function
from the rest of the world"
    console.log('Closure Invoked');
})();

var employee = function(){
    this.employeeId = 0;
    this.name = "";
};
/* JavaScript function acts as a constructor */
var emp = new  employee();
```

JSON Object

- JSONobject that provides functions to convert JavaScript values to and from the JavaScript Object Notation (JSON) format.
- The JSON.stringify function serializes a JavaScript value to JSON text.
- The JSON.parse function deserializes JSON text to produce a JavaScript value.

JSON.stringify

- Converts a JavaScript value to a JavaScript Object Notation (JSON) string.

```
var contact = new Object();
contact.fnmame = "Donald";
contact.lname = "Duck";
var jsonText = JSON.stringify(contact);
console.log(jsonText);

{"fnmame":"Donald","lname":"Duck"}
```

JSON.parse

- Converts a JavaScript Object Notation (JSON) string into an object.

```
var jsontext = '{"fnmame":"Donald","lname":"Duck"}';
var contact = JSON.parse(jsontext); console.log(contact.surname + ", "
+ contact.firstname);
```

```
> var str = '{"fname" : "Donald" , "lname" : "Duck" }';
< undefined
> var c = JSON.parse(str);
< undefined
> console.log(c);
  Object {fname: "Donald", lname: "Duck"}
< undefined
```

Summary

- In this lesson we have learned about –
  - JavaScript Functions
  - Working with JavaScript Functions
  - JSON Object
  - JSON.stringify and JSON.parse