

Agenda

- Language Fundamentals
- Functions
- Object
- Class
- ~~Array~~

Variable Scope

1. global

- a variable declared outside any function
- can be declared with or without var keyword
- can be accessed outside or inside any function
- E.g.

```
num = 100;  
var salary = 4.5;
```

- can be declared inside a function without using a var keyword
- E.g.

```
function function1() {  
  // global  
  firstName = "test";  
}
```

2. Local

- Must be declared inside a function with keyword var
- Can NOT be accessed outside the function in which it is declared
- E.g.

```
function function1() {  
  // local  
  var firstName = "test";  
}
```

Function Alias

- Another way/name to call a function
- Syntax: var = ;
- E.g.

```
function function1() {  
  console.log("inside function1");  
}  
  
// function alias  
var myFunction1 = function1;  
myFunction1();
```

Anonymous function

- Function without a name is called as anonymous function
- Syntax: var = function() { // body }
- E.g.

```
var multiply = function(p1, p2) {  
  console.log("p1 * p2 = " + (p1 * p2));  
}
```

Properties of function

- A function CAN NOT decide the data type of parameters
- Only caller decides the data type of parameters
- E.g.

```
function function1 (p1) {  
  console.log("p1 = " + p1 + " type = " + typeof(p1));  
}  
  
function1(10); // number  
function1("test"); // string  
function1(true); // boolean
```

- If a program contains multiple functions with same name then only bottom-most function's definition will be used
- E.g.

```
function function1 () {  
  console.log("function1 - 1");  
}  
  
function function1 () {  
  console.log("function1 - 2");  
}  
  
function function1 () {
```

```
console.log("function1 - 3");  
}  
  
// output: function1 - 3  
function1();
```

- A function can be called with excess number of parameters
- Excess parameters can be used by using a hidden parameter arguments
- E.g.

```
function function1 (p1, p2) {  
  // body  
}  
function1(10, 20, 30, 40, 50); // p1 = 10, p2 = 20
```

- A function can be called with less number of parameters
- E.g.

```
function function1 (p1, p2) {  
  // body  
}  
function1(); // p1 = undefined, p2 = undefined  
function1(10); // p1 = 10, p2 = undefined  
function1(10, 20); // p1 = 10, p2 = 20
```

- A function can be called before its declaration
- E.g.

```
function1();  
function function1() {  
  // body  
}
```

Object

- collection of properties (data members or fields) and methods
- Everything in JS is an Object, even functions are also objects
- To create an object (instance)
 - Use Object Literal ({})
 - Use new keyword
 - Use constructor function

```
// using Object
var c1 = new Object();
c1.model = "i10";
c1.company = "Hyundai";

// using construction function
function Car(model, company) {
    this.model = model;
    this.company = company;
}
var c2 = new Car("Fabia", "Skoda");

// using Object Literal
// also called as JSON
var c3 = {
    model: "X5",
    company: "BMW"
};

var cars = [c1, c2, c3];
for (var index = 0; index < cars.length; index++) {
    var car = cars[index];
    console.log("Model: " + car.model);
    console.log("Company: " + car.company);
}
```

Function alias

- We can create an alias for our function.
- declare a variable and initialize it with the existing function which will create an alias for that function

```
function function1() {
    console.log("Inside function1")
}
function1()

// function alias
var myfunction1 = function1
myfunction1()
```