**LAB # 06**

**Advance Java Script**

1. **Function**

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**. Function names can contain letters, digits, underscores, and dollar signs (same rules as variables). The parentheses may include parameter names separated by commas:  
**(parameter1, parameter2, ...)**

let x=myFunction(4, 3);      
function myFunction(a,b)

{  
  return a\*b;                
}

1. **Methods**

The method is a function associated with an object. The following is an example of a method in JavaScript –

var employee = {

empname: "David",

department : "Finance",

id : 002,

details : function() {

return this.empname + " with Department " + this.department;

}

};

employee.details();

1. **Ternary Operator**

let age = 18;

let message;

message = age >= 16 ? 'You can drive.' : 'You cannot drive.';

console.log(message);

1. **Switch**

The switch statement is used to perform different actions based on different conditions.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript switch</h2>

<p id="demo"></p>

<script>

let day;

switch (new Date().getDay()) {  
  case 0:  
    day = "Sunday";  
    break;  
  case 1:  
    day = "Monday";  
    break;  
  case 2:  
     day = "Tuesday";  
    break;  
  case 3:  
    day = "Wednesday";  
    break;  
  case 4:  
    day = "Thursday";  
    break;  
  case 5:  
    day = "Friday";  
    break;  
  case 6:  
    day = "Saturday";  
}

document.getElementById("demo").innerHTML = "Today is " + day;

</script>

</body>

</html>

1. **Arrow Function**

An **arrow function expression** is a compact alternative to a traditional [function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function), but is limited and can't be used in all situations.

hello = () => {  
  return "Hello World!";  
}

// ES5 Regular function  
let add = function(a, b) {   
 return a + b;  
};

// ES6 Arrow function  
let add = (a, b) => { return a + b };

## One argument

// ES5 Regular function  
let double = function(n) { return n \* 2 }   
alert( double(3) ); // 6

// ES6 Arrow function  
let double = n => n \* 2;   
alert( double(3) ); // 6

## Multiline arrow functions

Graphical user interface, text, application

Description automatically generated

# “this” binding

For example, look at the setTimeout function below:

// ES5  
let obj = {  
 id: 42,  
 counter: function counter() {  
 setTimeout(function() {  
 console.log(this.id);  
 }.bind(this), 1000);  
 }  
};

In the ES5 example, .bind(this) is required to help pass the this context into the function. Otherwise, by default this would be undefined.

// ES6  
let obj = {  
 id: 42,  
 counter: function counter() {  
 setTimeout(() => {  
 console.log(this.id);  
 }, 1000);  
 }  
};

# **Argument binding**

The arguments is a reference of the name in the enclosing scope, for example:

function show() {  
 return x => x + arguments[0];  
}let display = show(10, 20);  
let result = display(5);  
console.log(result); *// 15*

# Using “new” keyword

For example, the Car() function creates instances of a car:

function Car(color) {  
 this.color = color;  
}const blueCar = new Car('blue');  
blueCar instanceof Car; // => true

Car is a regular function, and when invoked with new keyword, it creates new instances of Car type.

JavaScript Form Validation Example

<html>

<body>

<script>

function validateform(){

var name=document.myform.name.value;

var password=document.myform.password.value;

if (name==null || name==""){

alert("Name can't be blank");

return false;

}else if(password.length<6){

alert("Password must be at least 6 characters long.");

return false;

}

}

</script>

<body>

<form name="myform" method="post" action="#" onsubmit="return validateform()" >

Name: <input type="text" name="name"><br/>

Password: <input type="password" name="password"><br/>

<input type="submit" value="register">

</form>

</body>

</html>

## JavaScript Retype Password Validation

<script type="text/javascript">

function matchpass(){

var firstpassword=document.f1.password.value;

var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){

return true;

}

else{

alert("password must be same!");

return false;

}

}

</script>

## JavaScript Number Validation

function validate(){

var num=document.myform.num.value;

if (isNaN(num)){

  document.getElementById("numloc").innerHTML="Enter Numeric value only";

  return false;

}

Else

{

  return true;

  }

## JavaScript email validation

function validateemail()

{

var x=document.myform.email.value;

var atposition=x.indexOf("@");

var dotposition=x.lastIndexOf(".");

if (atposition**<1** || dotposition**<atposition**+2 || dotposition+2**>**=x.length){

  alert("Please enter a valid email address \n atpostion:"+atposition+"\n dotposition:"+dotposition);

  return false;

  }

**Lab Task**

1. Create a function that calulates the power of 100 of a number entered as a parameter.
2. Write a javascript function that takes an array of numbers and a target number. The function should find two different numbers in the array that, when added together, give the target number. For example: answer([1,2,3], 4) should return [1,3].
3. Write a javascript function and ask user to enter the age, this function aim to show a message that a person can drive, if the age is greater than or equal to 18 otherwise can’t drive, by using a ternary operator instead of the [if-else](https://www.javascripttutorial.net/javascript-if-else/) statement.
4. Using switch in javascript, write a program to create any four browsers cases of your choice and generate alert (“I am using xyz browser) if user select any one of them.
5. Write a javascript arrow function named calculateSupply that:
   1. takes 2 arguments: age, amount per day.
   2. calculates the amount consumed for rest of the life (based on a constant max age).
   3. outputs the result to the screen like so: "You will need NN to last you until the ripe old age of X"
6. Create an object to hold information on your favorite recipe. It should have properties for title (a string), servings (a number), and ingredients (an array of strings). Write a method “Recipe” that print the separate lines (one console.log statement for each), log the recipe information so it looks like:

Tea

Serves: 2

Ingredients:

Sugar

Tea

Water

Milk

1. Write a javascript program to get the below object to go from:

let obj = {

my: 'name',

is: 'Rudolf',

the: 'raindeer'

}

// to this:

'my name is Rudolf the raindeer'

1. Design a Registration form using Bootstrap and validate it using Javascript

Graphical user interface, application

Description automatically generated with medium confidence