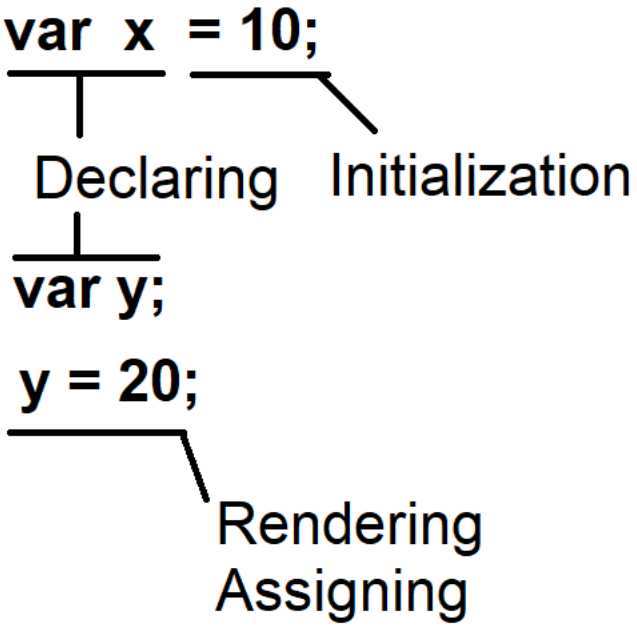
JavaScript Language Basics

- Variables
- Data Types
- Operators
- Statements
- Functions

Variables in JavaScript

- Variables are storage locations in memory where you can store a value and use it as a part of any expression.
- Variables configuration contains 3 stages
    - Declaration
    - Assigning or Rendering
    - Initialization



- Declaring variables in JavaScript is not mandatory if JavaScript is not in strict mode.
- You can directly use rendering without declaring.

  Ex:

```
<script>
   function f1(){
      x = 10;                    // valid
      document.write(`X=${x}`);
   }
   f1();
</script>
```

- Declaring variables mandatory if JavaScript is in strict mode.

Ex:

```
<script>
   "use strict";
   function f1(){
      x = 10;              // invalid
      document.write(`X=${x}`);
   }
   f1();
</script>
```

- JavaScript variables can be declared by using following keywords
  - var
  - let
  - const

Var:

- Var is used to defined a variable with function scope.
- You can declare variable in any block of a function and access anywhere in the function.
- It allows declaring, rendering and initialization.

  Ex:

  <script>

```
"use strict";
 function f1(){
    var x = 10;         // Initialization
    if(x==10) {
       var y;           // Declaration
       y = 20;          // Rendering
    }
    document.write("X=" + x + "<br>" + "Y="
+ y);
 }
 f1();
</script>
```

- **Var supports shadowing.**
- Shadowing is the process of re-declaring same name identifier in the block.

Ex:

```
<script>
  "use strict";
   function f1(){
      var x = 10;
      if(x==10) {
         var x;  // shadowing
         x = 30;
         var y;
```

```
            y = 20;
        }
        document.write("X=" + x + "<br>" + "Y="
+ y);
    }
    f1();
</script>
```

- **Var also supports “Hoisting”**
- Hoisting is a mechanism allowed for compiler or interpreter, so that you use a variable before declaring.

  Ex:

```
<script>
  "use strict";
   function f1(){
     x = 10;
     document.write("X=" + x);
     var x;  //Hoisting
   }
   f1();
</script>
```

LET:

- It used to define block scoped variable.

- You can access only within the declared block and child block.
- You can’t access outside the block.

Ex:

```
<script>
   "use strict";
    function f1(){
      let x = 10;
      if(x==10) {
         let y;
         y = 20;
      }
      document.write("X=" + x + "<br>" + "Y="
+ y); // Error: y not defined.
    }
    f1();
</script>
```

Ex:

```
<script>
   "use strict";
    function f1(){
      let x = 10;
      if(x==10) {
         let y;
```

```
        y = 20;
        document.write("X=" + x + "<br>" +
"Y=" + y);  // valid
    }

  }
  f1();
</script>
```

- **Let allows declaration, rendering, initialization.**

Ex:

```
<script>
  "use strict";
   function f1(){
     let x = 10;  // intialization
     if(x==10) {
        let y;   // declaration
        y = 20;  // rendering
        document.write("X=" + x + "<br>" +
"Y=" + y);
     }

  }
  f1();
```

```
</script>
```

- Let will not allow shadowing.
- There can’t be same name identifier with in the scope.

```
<script>
   "use strict";
    function f1(){
      let x = 10;  // intialization
      if(x==10) {
         let x = 20;
         let x = 30;                  // invalid –
shadowing not allowed
         let y;   // declaration
         y = 20;  // rendering
         document.write("X=" + x + "<br>" +
"Y=" + y);
      }

   }
   f1();
</script>
```

- **Let will not allow hoisting.**

Ex:

```
<script>
```

```
"use strict";
 function f1(){
  x  = 10;
  document.write("X=" + x);
  let x;              // Not Allowed
 }
 f1();
</script>
```

const:

- It is also a block scope variable.
- It will allow only initialization, no declaration, no rendering.
- It will not allow shadowing.
- It will not allow hoisting.

Ex:

```
<script>
 "use strict";
  function f1(){
   const x;              // not allowed
   x= 10;
```

```
    document.write("x=" + x);
  }
  f1();
</script>
```

Global Scope for Variables:

- You can declare any variable outside the function to make it global in access.
- You can use var, let or const.
- If you declare variables outside the function the it is referred as module scope.
- You can import a module into another module and access its variables.

Ex:

```
<script>
  "use strict";
  var x = 10;
  let y = 20;
  const z = 30;
  function f1(){
```

```
    document.write(`f1 values: <br>
x=${x}<br>y=${y}<br>z=${z}<br>`);
  }
  function f2(){
    document.write(`f2 values: <br>
x=${x}<br>y=${y}<br>z=${z}`);
  }
  f1();
  f2();
</script>
```

FAQ: Can we declare a variable inside function and make it global in access?

A. Yes. You can declare variable by using "window" object to make it global.

Ex:

```
<script>
  "use strict";
  var x = 10;
  let y = 20;
```

```
    const z = 30;
    function f1(){
      window.a = 40;  //window refers to browser
      document.write(`f1 values: <br>
x=${x}<br>y=${y}<br>z=${z}<br>a=${a}<br>`);
    }
    function f2(){
      document.write(`f2 values: <br>
x=${x}<br>y=${y}<br>z=${z}<br>a=${a}`);
    }
    f1();
    f2();
</script>
```

Variables Naming

- Variable name must start with an alphabet or underscore _
- It can be alpha numeric, but can't start with number.

  Syntax:

  var jan2020 = 12000;

var 2020jan = 23000; invalid

var _2020jan = 34000; valid

var jan_2020 = 34000; valid

FAQ: Why underscore is supported? What underscore means?

A. Underscore is special symbol that doesn't have any compile time issues.

underscore is used by developers to indicate that the variable requires further implementation.

"Marked for Implementation"

- ECMA standards prefer variable name length maximum 255 chars.
- Variable name must speak what it is.
- Always use camel case for naming references.

```
class EmployeeSalary

{

}

var employeeSalary = new Employee();

var txtName;

var txtPassword;
```

var btnSubmit;

Data Types

- Data type determines the data structure.
- Data structure specifies the size and behaviour of value in memory.
- Data Type uses a data structure to define the type of value that can be stored in memory.
- JavaScript is implicitly typed; the data type is determined according to the value assigned. There is no specific built-in type.
- JavaScript is not strongly typed, you can store any contradictory values.

  Syntax:

  var x = “John”; //string

  x = 10; // valid and x changes to number

What type of values JavaScript can handle?

The JavaScript data types are classified into 2 groups

- Primitive Types
- Non-Primitive Types

Primitive Types

- Primitive types are stored in memory stack. [LIFO]
- They have a fixed range for values.
- They are Immutable types.
- Their values can’t change according to state and situation.
- JavaScript primitive types are
    - Number
    - String
    - Boolean
    - Null
    - Undefined

Number Type:

- Number type is used to handle a numeric value.
- JavaScript number type can allow
    - Signed Integer

      var x = -10;

      var x = +10;
    - Unsigned Integer

      var x = 10;

- Floating point
  var x = 4.5;
  var x = 33.55;
- Double
  var x = 553.558
- Decimal
  var x = 45600.6669594; [29 decimal places]
- Hexa
  0xf00d
- Binary
  0b1010
- Octa
  0o744
- Exponent
  Var x = 2e3; [2 x 10^3] = 2000

BigInt = 100n; [Binary Data – images - complex]

Ex:

```
<script>
  "use strict";
```

```
    function f1(){
      document.write(`Min Integer:
${Number.MIN_SAFE_INTEGER} <br> Max
Integer: ${Number.MAX_SAFE_INTEGER}`);
    }
    f1();
</script>
```

Validating Numbers

- The operator "typeof" is used to verify and return the data type of variable.
- IsNaN() is a JavaScript that verifies whether the given value is a number or any other type.
- Every value you entered in form element will be "string" type.
- You have to convert the string into number by using the functions
    - parseInt()
    - parseFloat()

Ex:

```
<script>
var x = 10;
var y = "4";
```

```
if(isNaN(y)){
   document.write("Invalid Number");
} else {
var z =  x * y;
document.write("z=" + z);
}
</script>
```

Ex:

```
<script>
var x = prompt("Enter X Value");
var y = prompt("Enter Y Value");
if(isNaN(y)){
   document.write("Invalid Number");
} else {
var z =  x * y;
document.write("z=" + z);
}

</script>
```

String Type

- String is a literal with group of characters enclosed in Quotes.
- JavaScript string can be enclosed in
  - Single Quote ‘ ‘
  - Double Quote “ “
  - Back Tick ` `
- Single and double quotes are used to swap between inner and outer string.

Ex:

```
<script>
"use strict";
 function f1(){
   var link = "<a href='home.html'>Home</a>";
   document.write(link);
 }
 f1();
</script>
```

Ex:

```
<script>
  "use strict";
   function f1(){
     var link = '<a href="home.html">Home</a>';
     document.write(link);
   }
   f1();
</script>
```

Back Tick:

- Back Tick [` `] is available from ES5
- It is used to define a string with embedded expression.
- Expression can be embedded with “${}”
- Expression can’t be embedded into string with single or double quote.

Ex:

```
<script>
  "use strict";
   function f1(){
```

```
    var age = 20;
    var year = 2020;
    document.write("You will be" + " " + (age+1) +
" " + "Next Year" + " " + (year+1) + "<br>");
    document.write(`You will be ${age+1} Next
Year ${year+1}`);
  }
  f1();
</script>
```

Ex:

```
<script>
  "use strict";
  function f1(){
    var title = "Admin Login";
    var login = `
      <h2>${title}</h2>
      <dl>
      <dt>Name</dt>
```

```
    <dd><input type="text"></dd>
    <dt>Price</dt>
    <dd><input type="text"></dd>
    </dl>
    <button>Login</button>
  `;
  document.write(login);
}
f1();
</script>
```

- Several special characters defined in a string will escape printing.
- To print the non-printable characters, we have to use “\”.

Ex:

```
<script>
  "use strict";
   function f1(){
     var path =
"\"D:\\Images\\Pics\\mobile.jpg\"";
```

```
        document.write(path);
    }
    f1();
</script>
```

Ex:

```
<script>
  "use strict";
    function f1(){
        alert("Hello \n Welcome \n to \n JavaScript");
        document.write("Hello ! <br> Welcome");
    }
    f1();
</script>
```

Note: The numbers or the values that you access from any element are string type. You have to convert the string into number to handle

expressions. JavaScript functions to convert string into number.

- parseInt()
- parseFloat()

Ex:

```
<!DOCTYPE html>
<html>
 <head>
  <title>String</title>
  <script>
   function Calculate(){
      var txt1 =
document.getElementById("txt1").value;
      var txt2 =
document.getElementById("txt2").value;

document.getElementById("result").innerHTML =
parseFloat(txt1) + parseFloat(txt2);
   }
  </script>
```

```
  </head>
  <body>
    <dl>
      <dt>Number-1</dt>
      <dd><input id="txt1" type="text"></dd>
      <dt>Number-2</dt>
      <dd><input id="txt2" type="text"></dd>
    </dl>
    <button
onclick="Calculate()">Calculate</button>
    <h2 id="result"></h2>
  </body>
</html>
```

String Manipulation Functions

- JavaScript string object provides a set of properties and methods that are used to manipulate and format string.
- Manipulation methods

Method	Description

charAt()	It returns the character as specified index. Syntax: string.charAt(); Ex: <script> function f1(){ var str = "Welcome to JavaScript"; var char1 = str.charAt(0); var char2 = str[1]; //New in ES5 document.write(`Char1=${char1} Char2=${char2}`) } f1(); </script>

charCodeAt()	It returns the character code of character at specified index. ASCII code of characters are accessed. A=65, Z=90
indexOf()	Returns the first occurrence index number of specified characters.
lastIndexOf()	Returns the last occurrence index number of specified char. If the character not found then both methods return “-1”.
trim()	It is used to remove the leading spaces in a string.
substring()	It can extract a portion of string based on specified index. It is similar to slice but will not allow negative values. You can access right to left by using positive value. string.substring(start position, end position) Ex:

	<script>
	function f1(){
	var str = "Welcome to JavaScript";
	document.write(str.substring(7,0));
	}
	f1();
	</script>

substr()	It is a legacy method, will not allow the values right to left. string.substr(start, length) start Required. The start position. First character is at index 0. If start is greater than the length, substr() returns "". If start is negative, substr() counts from the end of the string. Length is Optional.

	The number of characters to extract. If omitted, it extracts the rest of the string
slice()	It is used to extract a part of string and return a new string. Syntax: slice(startIndex, endIndex) slice(startIndex); slice upto end. Slice(-1); It returns the last character. Slice(-4); It returns the last 4 chars.
split()	It splits string at specific delimiter and returns an array of substrings. You can also restrict the number of items to split. Syntax: String.split('delimiter', count) Ex: <script> function f1(){

	var mobiles = "9876543210,9988776655,9008833113";
	var numbers = mobiles.split(',', 2);
	for(var number of numbers) {
	document.write(number + "
	");
	}
	}
	f1();
	</script>

startsWith()	It returns true if string starts with specified chars.
endsWith()	It returns true if string ends with specified chars. Ex: <!DOCTYPE html> <html>

```
<head>
    <title>String</title>
    <script>
        function Verify(){
            var txtEmail =
document.getElementById("txtEmail")
.value;

if(txtEmail.endsWith("gmail.com")) {
                document.write("Your
Gmail Verified..");
            } else {
                document.write("Only
Gmail allowed");
            }
        }
    </script>
</head>
<body>
```

	<fieldset> <legend>Your Email</legend> <input type="text" id="txtEmail" placeholder="Only Gmail Allowed"> <button onclick="Verify()">Submit</button> </fieldset> </body> </html>
search()	You can search for any char using a regular expression. It returns the index number of searched string. If character not found then it returns -1 Ex: <script> function f1(){

	var str = "Welcome to JavaScript";
	
	document.write(str.search(/javascript/i));
	}
	f1();
	</script>

Ex: Credit/Debit Card number

```
<!DOCTYPE html>
<html>
   <head>
      <title>String Demo</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         function VerifyName(){
```

```
            var txtName =
document.getElementById("txtName").value;
            var msg =
document.getElementById("msg");
            var firstCharCode =
txtName.charCodeAt(0);
            if(firstCharCode>=65 &&
firstCharCode<=90) {
                msg.innerHTML = "";
            } else {
                msg.innerHTML = "Name must start
with Uppercase Letter";
            }
        }
        function VerifyCard(){
            var txtCard =
document.getElementById("txtCard").value;
            var firstChar = txtCard.charAt(0);
            var cardLogo =
document.getElementById("cardLogo");
```

```
            if(firstChar=="4") {
                cardLogo.src="../Images/visa.png";
            } else if(firstChar=="5"){
                cardLogo.src="../Images/master.png";
            } else {
                cardLogo.src="../Images/invalid.png";
            }
        }
    </script>
</head>
<body class="container-fluid">
    <div class="form-group">
        <label>User Name</label>
        <div>
            <input onblur="VerifyName()"
class="form-control" placeholder="Name must
start with Uppercase Letter" type="text"
id="txtName">
            <div id="msg" class="text-
danger"></div>
```

```
            </div>
        </div>
        <div class="form-group">
            <label>Card Number</label>
            <div class="input-group">
                <input onkeyup="VerifyCard()"
type="text" id="txtCard" class="form-control">
                <div class="input-group-append">
                    <span class="input-group-text">
                        <img id="cardLogo" width="50"
height="20">
                    </span>
                </div>
            </div>

        </div>
    </body>
</html>
```

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <title>String Demo</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootst
rap.css">
      <script>
         function VerifyName(){
            var txtName =
document.getElementById("txtName").value;
            var msg =
document.getElementById("msg");
            var firstCharCode =
txtName.charCodeAt(0);
            if(firstCharCode>=65 &&
firstCharCode<=90) {
               msg.innerHTML = "";
            } else {
```

```
            msg.innerHTML = "Name must start
with Uppercase Letter";
        }
    }
    function VerifyCard(){
        var txtCard =
document.getElementById("txtCard").value;
        var firstChar = txtCard.charAt(0);
        var cardLogo =
document.getElementById("cardLogo");
        if(firstChar=="4") {
            cardLogo.src="../Images/visa.png";
        } else if(firstChar=="5"){
            cardLogo.src="../Images/master.png";
        } else {
            cardLogo.src="../Images/invalid.png";
        }
    }
    function VerifyEmail(){
```

```
            var txtEmail =
document.getElementById("txtEmail").value;
            var emailError =
document.getElementById("emailError");
            var atPos = txtEmail.indexOf("@");
            var dotPos = txtEmail.lastIndexOf(".");
            if(atPos<=2 && (dotPos-atPos)<=2) {
               emailError.innerHTML = "Error: @
missing or not at valid position in email";
            } else {
               emailError.innerHTML = "Email
Verified";
            }


         }
      </script>
   </head>
   <body class="container-fluid">
      <div class="form-group">
         <label>User Name</label>
```

```
        <div>
            <input onblur="VerifyName()"
class="form-control" placeholder="Name must
start with Uppercase Letter" type="text"
id="txtName">
            <div id="msg" class="text-
danger"></div>
        </div>
    </div>
    <div class="form-group">
        <label>Card Number</label>
        <div class="input-group">
            <input onkeyup="VerifyCard()"
type="text" id="txtCard" class="form-control">
            <div class="input-group-append">
                <span class="input-group-text">
                    <img id="cardLogo" width="50"
height="20">
                </span>
            </div>
```

```
            </div>
        </div>
        <div class="form-group">
            <label>Email</label>
            <div>
                <input onblur="VerifyEmail()"
id="txtEmail" type="text" class="form-control">
                <div id="emailError">

                </div>
            </div>
        </div>
    </body>
</html>
```

Ex:

```
<!DOCTYPE html>
<html>
    <head>
```

```
    <title>String Demo</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootst
rap.css">
    <script>
        function VerifyName(){
            var txtName =
document.getElementById("txtName").value;
            var msg =
document.getElementById("msg");
            var firstCharCode =
txtName.charCodeAt(0);
            if(firstCharCode>=65 &&
firstCharCode<=90) {
                msg.innerHTML = "";
            } else {
                msg.innerHTML = "Name must start
with Uppercase Letter";
            }
        }
```

```
        function VerifyCard(){
            var txtCard =
document.getElementById("txtCard").value;
            var firstChar = txtCard.charAt(0);
            var cardLogo =
document.getElementById("cardLogo");
            if(firstChar=="4") {
                cardLogo.src="../Images/visa.png";
            } else if(firstChar=="5"){
                cardLogo.src="../Images/master.png";
            } else {
                cardLogo.src="../Images/invalid.png";
            }
        }
        function VerifyEmail(){
            var txtEmail =
document.getElementById("txtEmail").value;
            var emailError =
document.getElementById("emailError");
            var atPos = txtEmail.indexOf("@");
```

```
            var dotPos = txtEmail.lastIndexOf(".");
            if(atPos<=2 && (dotPos-atPos)<=2) {
               emailError.innerHTML = "Error: @
missing or not at valid position in email";
            } else {
               emailError.innerHTML = "Email
Verified";
            }
         }
         function VerifyPassword(){
            var txtPwd =
document.getElementById("txtPwd").value;
            var pwdError =
document.getElementById("pwdError");
            if(txtPwd.trim()=="john") {
               pwdError.innerHTML = "Verified";
            } else {
               pwdError.innerHTML = "Invalid
Password";
            }
```

```
        }
    </script>
</head>
<body class="container-fluid">
    <div class="form-group">
        <label>User Name</label>
        <div>
            <input onblur="VerifyName()"
class="form-control" placeholder="Name must
start with Uppercase Letter" type="text"
id="txtName">
            <div id="msg" class="text-
danger"></div>
        </div>
    </div>
    <div class="form-group">
        <label>Password</label>
        <div>
```

```
            <input onblur="VerifyPassword()"
type="password" id="txtPwd" class="form-
control">
            <div id="pwdError">

            </div>
          </div>
        </div>
        <div class="form-group">
          <label>Card Number</label>
          <div class="input-group">
            <input onkeyup="VerifyCard()"
type="text" id="txtCard" class="form-control">
            <div class="input-group-append">
              <span class="input-group-text">
                <img id="cardLogo" width="50"
height="20">
              </span>
            </div>
          </div>
```

```
        </div>
        <div class="form-group">
            <label>Email</label>
            <div>
                <input onblur="VerifyEmail()"
id="txtEmail" type="text" class="form-control">
                <div id="emailError">

                </div>
            </div>
        </div>
    </body>
</html>
```

Compare String and Verify Equality:

- To compare string in JavaScript you can use operators like
    - Greater Than >, >=
    - Less than <, <=
    - Equal =

- Not-Equal !=

- JavaScript can also compare the string by using “localeCompare()” method.
- “localeCompare()” method returns 0 when matching and 1 on mismatch.

  Syntax:

  sourceString.localeCompare(targetString, ‘lang’, {options})

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Compare</title>
      <script>
         function VerifyPassword(){
            var pwd = document.getElementById("txtPwd").value;
            var cPwd = document.getElementById("txtConfirm").value;
            var result = pwd.localeCompare(cPwd);
```

```
            var lblError =
document.getElementById("lblError");
            lblError.innerHTML = (result==0)?"<font
color='green'>Password Verified</font>":"<font
color='red'>Password Mismatch</font>";
        }
    </script>
  </head>
  <body>
    <div>
      <label>Password</label>
      <div>
        <input type="password" id="txtPwd">
      </div>
    </div>
    <div>
      <label>Confirm Password</label>
      <div>
        <input onblur="VerifyPassword()"
type="password" id="txtConfirm">
```

```
            <div id="lblError">


            </div>
          </div>
        </div>
    </body>
</html>
```

Compare with Regular Expression:

- The function “match()” is used to verify the value with regular expression.
- It returns true if value is matching with regular expression.

Syntax:

```
String.match(regularExpression);   // true-false
```

Ex:

```
<!DOCTYPE html>
<html>
    <head>
```

```
    <title>Verify Password</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootst
rap.css">
    <style>
        progress {
            height: 50px;
            width: 100%;
        }
    </style>
    <script>
        function VerifyPassword(){
            var regExp = /(?=.*[A-Z])\w{4,10}/;
            var txtPwd =
document.getElementById("txtPwd").value;
            var lblMsg =
document.getElementById("lblMsg");
            var grade =
document.getElementById("grade");
            grade.style.display = "inline";
```

```
            function GradeDisplay(min, max, value){
                grade.min = min;
                grade.max = max;
                grade.value = value;

            }

            if(txtPwd.match(regExp)){
                lblMsg.innerHTML="Strong Password";
                GradeDisplay(1,100,100);
            } else {
                if(txtPwd.length<4) {
                    lblMsg.innerHTML="Poor Password";
                    GradeDisplay(1,100,20)
                } else {
                    lblMsg.innerHTML="Weak
Password";
                    GradeDisplay(1,100,60);
```

```
            }
        }
    }
    </script>
</head>
<body class="container-fluid">
    <h2>Regular Expression</h2>
    <div class="form-group">
        <label>Password</label>
        <div>
            <input onkeyup="VerifyPassword()" id="txtPwd" type="password" class="form-control">
            <div>
                <progress min="1" max="100" style="display: none;" id="grade"></progress>
                <span id="lblMsg"></span>
            </div>
        </div>
```

```
    </div>
  </body>
</html>
```

String Formatting Functions

- You can dynamically format a string by using following methods
  - bold()
  - italic()
  - fontsize()
  - fontcolor()
  - sup()
  - sub()
  - toUpperCase()
  - toLowerCase()

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Verify Password</title>
```

```
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootst
rap.css">
    <style>
        progress {
            height: 50px;
            width: 100%;
        }
    </style>
    <script>
        function VerifyPassword(){
            var regExp = /(?=.*[A-Z])\w{4,10}/;
            var txtPwd =
document.getElementById("txtPwd").value;
            var lblMsg =
document.getElementById("lblMsg");
            var grade =
document.getElementById("grade");
            grade.style.display = "inline";
```

```
            function GradeDisplay(min, max, value){
                grade.min = min;
                grade.max = max;
                grade.value = value;

            }

            if(txtPwd.match(regExp)){
                lblMsg.innerHTML="Strong
Password".fontcolor('green').bold();
                GradeDisplay(1,100,100);
            } else {
                if(txtPwd.length<4) {
                    lblMsg.innerHTML="Poor
Password".fontcolor('red').italics();
                    GradeDisplay(1,100,20)
                } else {
                    lblMsg.innerHTML="Weak
Password".fontcolor('yellow').bold().italics();
```

```
                GradeDisplay(1,100,60);
            }
        }
    }
    </script>
</head>
<body class="container-fluid">
    <h2>Regular Expression</h2>
    <div class="form-group">
        <label>Password</label>
        <div>
            <input onkeyup="VerifyPassword()" id="txtPwd" type="password" class="form-control">
            <div>
                <progress min="1" max="100" style="display: none;" id="grade"></progress>
                <span id="lblMsg"></span>
            </div>
```

```
        </div>
      </div>
   </body>
</html>
```

Boolean Types

- Boolean types are defined by using “true or false”.
- Boolean type is used to handle decision making in programming.
- JavaScript boolean types refer to numeric value
    - 0 : false
    - 1 : true
- The boolean conditions in JavaScript can be configure with 0 or 1.
- JavaScript can control several HTML properties by using boolean type, which includes checked, selected, disabled etc.

Syntax:

```
var x = true;
```

```
if(x==1) {
    statement on true;
} else {
    statement on false;
}
```

Undefined Type

- Undefined type is configured for variables that are not defined with value.
- Variable is defined but value is not assigned or initialized then the compile will configure as “undefined”.
- You can verify whether value defined or not by using undefined.

Ex:

```
<script>
  function f1(){
    var x;
    if(x==undefined) {
      document.write("there is No value in x");
```

```
        } else {
        document.write(`x=${x}`);
        }
    }
    f1();
</script>
```

Null Type

- Value is not defined into a reference dynamically during run time.
- Null is reference type, which indicates that value is not supplied to variable during run time.

Ex:

```
<script>
    function f1(){
        var uname = prompt("Enter Name");
        if(uname==null) {
            document.write("You canceled");
        } else {
```

```
    document.write(`Hello ! ${uname}`);
    }
  }
  f1();
</script>
```

Summary

- Number
- String
- Boolean
- Null
- Undefined

Non-Primitive Types

- The non-primitive types are “Mutable” types.
- Their reference can be changed according to state and situation.
- They don’t have fixed range of values.
- The value range varies according to the memory available.
- JavaScript non-primitive types are

- Array
- Object
- Regular Expression