JavaScript

- JavaScript is a language.
- It is an open source, cross platform, interpreted and just-in-time compiled programming language.
- Interpreter is a translated that translates line by line.
- Just-in-time is a compiler that compiles in browser when requested by client.
- V8 is JavaScript compiler.
- JavaScript is used
    - Client Side with HTML
    - Server Side with Node.js
    - Database with MongoDB
    - Animations with Flash
- JavaScript supports various types of programming like, imperative, functional programming, structural programming and object-oriented programming.
- In early 1994 **Brendan Eich** introduces a script called “Mocha” for Netscape browser.
- After that “Moca” was renamed into “Live Script”
- 1995 Sun Microsystems took responsibility of maintaining Live Script and re-named as “JavaScript”.
- JavaScript Designed by “Brendan Eich”
- JavaScript initially belong to “Netscape communications”.
- JavaScript follows the standards of “ECMA” [European Computer Manufacturers Association].
- JavaScript versions are ECMAScript 2015, ES6, ES8 ES2020
- ES5, ES6 are most commonly used versions in various web technologies.
- ES6

Issues with JavaScript

- Browser incompatibility: Every browser has its own extensions to JavaScript and every browser have its own parser [translator].
- JavaScript is not secured: It is client side. Everyone can view.
- JavaScript can be disabled by browser. [Browser can block JavaScript].
- JavaScript is not strongly typed language.
  var x = 10; // number
  x = “John”; // string – valid

JavaScript with HTML

- JavaScript is used to manipulate the HTML DOM.
- It converts the static DOM elements into dynamic DOM elements.
- JavaScript can add elements, remove elements, modify the data, handle validations, handle plugins, browser location, history etc.
- JavaScript can reduce burden on server by managing several interactions client-side.

Using JavaScript in HTML Page:

- JavaScript can be integrated and used with HTML page by using following techniques
    - JavaScript can be inline
    - JavaScript can be embedded.
    - JavaScript can be from external file.

JavaScript Inline:

- JavaScript functions are defined within the element.
- They are faster as they are local to element.
- They can't be re-used.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
  </head>
  <body>
    <h2>Click Print Button to Print Page</h2>
    <button onclick="window.print()">Print Page</button>
  </body>
```

```
</html>
```

JavaScript Embedded:

- You can write the JavaScript functions and embed into page by using <script> element.
- You can embed in head section or body section.
- You can re-use the functions across the page from any element.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline</title>
    <script>
      function PrintPage(){
        window.print();
      }
    </script>
  </head>
  <body>
    <h2>Click Print Button to Print Page</h2>
    <button onclick="PrintPage()">Print Page</button>
    <button onclick="PrintPage()">Print</button>
  </body>
</html>
```

- Embedded scripts require **MIME type** to define.
- The JavaScript MIME type is “text/javascript” or “language=JavaScript”

Syntax:

```
<script type="text/javascript" language="javascript">
    function PrintPage(){
        window.print();
    }
</script>
```

JavaScript Strict Mode:

- JavaScript is recommended to write in “Strict” mode.
- It reduces the code inconsistency.
- You can turn ON strict mode by using “use strict” in the code.

Ex:

```
<script>
    "use strict";
    function f1()
    {
        x = 10; // x is not declared as variable
        document.write("x=" + x);
    }
    f1();
</script>
```

Note: remove “use strict” from <script> element, the above code will work normally. In strict mode you have to declare a variable “var x”.

Download ESLint for Visual Studio Code

JavaScript from External File

- JavaScript functions are maintained in a separate script file with extension “.js”.
- You can link the script file to any HTML page by using <script> element.
- You can re-use the function across multiple pages.
- It will increase the page load time.

Ex:

- Create a new folder “scripts”

- Add a new file into folder by name “printing.js”

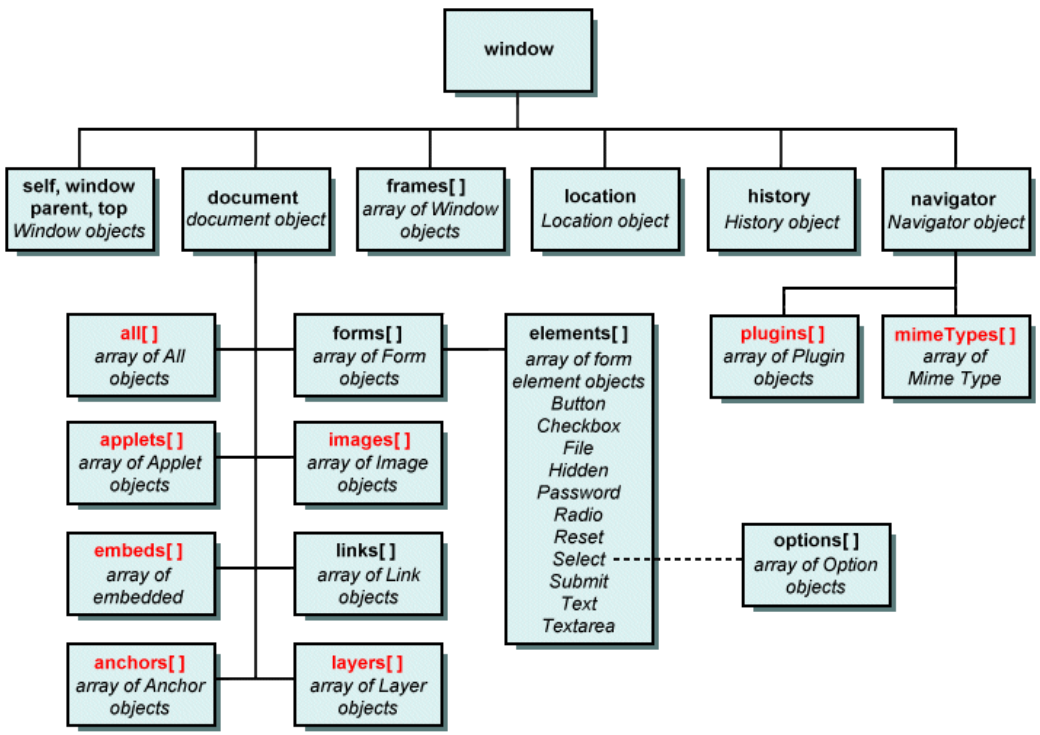
```
"use strict";
function PrintPage()
{
   window.print();
}
```

- Link to HTML page

```
<!DOCTYPE html>
<html>
   <head>
      <title>External File</title>
      <script src="../Scripts/printing.js"></script>
   </head>
   <body>
      <h2>Print Page by clicking the print button</h2>
      <button onclick="PrintPage()">Print Page</button>
   </body>
</html>
```

How JavaScript Refers HTML DOM Elements?

1. Refer by using DOM hierarchy



Ex: window.document.images[index]
window.document.forms[index].elements[index]

- Every time when you change the position of any DOM element, you have to update the index in logic.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Reference</title>
    <script type="text/javascript">
      function bodyload(){
        window.document.images[0].src="../Images/shoe.jpg";
        window.document.forms[0].elements[1].value = "Register";
      }
    </script>
  </head>
  <body onload="bodyload()">
    <div>
      <img width="100" height="100" border="1">
    </div>
    <div>
      <h2>Register</h2>
      <form>
        User Name:
        <input type="text">
        <input type="button">
      </form>
```

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

2. Refer by using “name”

- Every element can have a reference name.
- Name can be common for multiple elements.
- If you are referring a child element, it is mandatory that you have to refer the parent element.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Reference</title>
    <script type="text/javascript">
      function bodyload(){
        pic.src="../Images/shoe.jpg";
        frmRegister.btnRegister.value = "Register";
      }
    </script>
  </head>
  <body onload="bodyload()">
    <div>
      <img name="pic" width="100" height="100" border="1">
    </div>
    <div>
      <h2>Register</h2>
      <form name="frmRegister">
```

```
            User Name:
            <input name="txtName" type="text">
            <input name="btnRegister" type="button">
        </form>
    </div>
  </body>
</html>
```

3. **Refer by using ID**

- Every element can be defined with unique ID.
- You can access element by using the following document method

  document.getElementById()

- You can access any element directly with ID.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Reference</title>
    <script type="text/javascript">
      function bodyload(){
        document.getElementById("pic").src="../Images/shoe.jpg";
        document.getElementById("btnRegister").value = "Register";
      }
    </script>
  </head>
  <body onload="bodyload()">
    <div>
      <img id="pic" width="100" height="100" border="1">
```

```
    </div>
    <div>
      <h2>Register</h2>
      <form id="frmRegister">
        User Name:
        <input id="txtName" type="text">
        <input id="btnRegister" type="button">
      </form>
    </div>
  </body>
</html>
```

4. **Access elements by Tag Name**
   getElementsByTagName()

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Reference</title>
    <script type="text/javascript">
      function bodyload(){
        x = document.getElementsByTagName("img")
        alert("Total Number of Images: " + x.length);
      }
    </script>
  </head>
  <body onload="bodyload()">
```

```
        <div>
          <img id="pic" width="100" height="100" border="1">
        </div>
        <div>
          <h2>Register</h2>
          <form id="frmRegister">
            User Name:
            <input id="txtName" type="text">
            <input id="btnRegister" type="button">
          </form>
        </div>
      </body>
    </html>
```

5. **Access elements by class Name**
   Every element can have multiple classes.
   You can access element by using their class name.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .effects {
        background-color: yellow;
      }
    </style>
    <title>Reference</title>
    <script type="text/javascript">
```

```
        function bodyload(){
          x = document.getElementsByClassName("effects");
          alert("Total Count: " + x.length);
        }
      </script>
    </head>
    <body onload="bodyload()">
      <div>
        <img id="pic" width="100" height="100" border="1">
      </div>
      <div>
        <h2>Register</h2>
        <form id="frmRegister">
          User Name:
          <input class="effects" id="txtName" type="text">
          <input class="effects" id="btnRegister" type="button">
        </form>
      </div>
    </body>
</html>
```

6. **Access elements that have common name**

Ex:

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

```
  <head>
    <style>
      .effects {
        background-color: yellow;
      }
    </style>
    <title>Reference</title>
    <script type="text/javascript">
      function bodyload(){
       x = document.getElementsByName("pay");
       alert("Total Payment Methods: " + x.length);
      }
    </script>
  </head>
  <body onload="bodyload()">
    <fieldset>
      <legend>Payment</legend>
      <input type="radio" name="pay" value="Cash"> Cash
      <input type="radio" name="pay" value="UPI"> UPI
      <input type="radio" name="pay" value="Credit Card"> Credit Card

    </fieldset>
  </body>
</html>
```

JavaScript Output and Input Methods

JavaScript Output Method and Properties:

- alert()
- confirm()
- document.write()
- innerHTML
- innerText
- outerHTML
- console.log()
- console.error()
- console.warn()
- console.debug()
- console.info()

alert():

It is used to display output in a message box.

User have to confirm the message in order to continue.

Syntax:

alert(“message/ expression”);

You can’t cancel alert. “esc” key is similar to OK.

EX:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .effects {
        background-color: yellow;
      }
    </style>
```

```
        <title>Reference</title>
        <script type="text/javascript">
          function DeleteClick(){
             alert("Record Deleted");
          }
        </script>
    </head>
    <body>
      <button onclick="DeleteClick()">Delete</button>
    </body>
</html>
```

confirm():

It is similar to alert, but provides option to cancel the message.

It returns **true** on OK and **false** on Cancel.

Ex:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .effects {
                background-color: yellow;
            }
        </style>
        <title>Reference</title>
```

```
        <script type="text/javascript">
          function DeleteClick(){
             x = confirm("Are you sure want to Delete?");
             if(x==true){
                alert("Record Deleted");
             } else {
                alert("Canceled..");
             }
          }
        </script>
    </head>
    <body>
       <button onclick="DeleteClick()">Delete</button>
    </body>
</html>
```

document.write()

It is used to print output in a new screen of same page.

The output is erased when you refresh the page.

EX:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .effects {
                background-color: yellow;
```

```
        }
    </style>
    <title>Reference</title>
    <script type="text/javascript">
     function DeleteClick(){
        x = confirm("Are you sure want to Delete?");
        if(x==true){
           document.write("Record Deleted");
        } else {
           alert("Canceled..");
        }
     }
    </script>
  </head>
  <body>
    <button onclick="DeleteClick()">Delete</button>
  </body>
</html>
```

innerHTML & innerText

It is used to define output in containers like div, span, dd, dt, td, p, blockquote.

innerText is an RC data element. [Text without formats]

innerHTML supports formats.

Ex:

```
<!DOCTYPE html>
```

```
<html>
   <head>
      <style>
         .effects {
            background-color: yellow;
         }
      </style>
      <title>Reference</title>
      <script type="text/javascript">
       function DeleteClick(){
          x = confirm("Are you sure want to Delete?");
          if(x==true){
             document.getElementById("msg").innerText="<font
color='red'>Record Deleted</font>";
          } else {
             alert("Canceled..");
          }
       }
      </script>
   </head>
   <body>
     <button onclick="DeleteClick()">Delete</button>
     <p align="center" id="msg"></p>
   </body>
</html>
```

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         .effects {
            background-color: yellow;
         }
      </style>
      <title>Reference</title>
      <script type="text/javascript">
       function DeleteClick(){
          x = confirm("Are you sure want to Delete?");
          if(x==true){
             document.getElementById("msg").innerHTML="<font
color='red'>Record Deleted</font>";
          } else {
             alert("Canceled..");
          }
       }
      </script>
   </head>
   <body>
     <button onclick="DeleteClick()">Delete</button>
     <p align="center" id="msg"></p>
   </body>
</html>
```

innerHTML will add inside existing element.

outerHTML will add by replacing existing element.

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         .effects {
            background-color: yellow;
         }
      </style>
      <title>Reference</title>
      <script type="text/javascript">
        function DeleteClick(){
           x = confirm("Are you sure want to Delete?");
           if(x==true){
              document.getElementById("msg").outerHTML="<h2><font
color='red'>Record Deleted</font></h2>";
           } else {
              alert("Canceled..");
           }
        }
      </script>
   </head>
   <body>
     <button onclick="DeleteClick()">Delete</button>
     <p align="center" id="msg"></p>
```

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

Console options

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         .effects {
            background-color: yellow;
         }
      </style>
      <title>Reference</title>
      <script type="text/javascript">
        function DeleteClick(){
           x = confirm("Are you sure want to Delete?");
           console.info("Delete Initiated");
           if(x==true){
              console.error("Delete Completed");
              document.getElementById("msg").outerHTML="<h2><font
color='red'>Record Deleted</font></h2>";
           } else {
              alert("Canceled..");
              console.warn("Canceld action");
           }
        }
      </script>
```

```
  </head>
  <body>
    <button onclick="DeleteClick()">Delete</button>
    <p align="center" id="msg"></p>
  </body>
</html>
```

JavaScript Input Properties

- JavaScript can handle input from user by using
  - prompt()
  - Form Input Elements

Prompt():

- It opens a dialog in browser and allows user to input value.
- User can input any type of value like text, number, boolean etc. but not complex values like images, binary type data etc.
- Prompt is used when input is not regular.

Syntax:

prompt(“Message”, “Default Value”);

- Prompt returns following
  - null: When it is cancelled with or without value.
  - Empty: When you click OK without value.
  - Value: When you click OK with value.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Input</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
```

```
    <script>
      function CreateClick(){
        folderName = prompt("Enter Folder Name:","New_Folder");
        if(folderName==null) {
          document.write("You cancelled");
        } else if (folderName=="") {
          document.write("Name can't be empty");
        } else {
          document.getElementById("msg").innerHTML+= "Folder Created :" +
folderName + "<br>";
        }
      }
      function bodyload(){
        username = prompt("Your Name:");
        document.getElementById("heading").innerHTML="Hello ! " +
username;
      }
    </script>
  </head>
  <body class="container-fluid" onload="bodyload()">
    <h2 id="heading"></h2>
    <div class="form-group">
      <button onclick="CreateClick()" class="btn btn-primary">Create
Folder</button>
    </div>
    <div class="form-group">
      <p id="msg"></p>
```

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

Input using Form Elements:

- HTML form provide several elements that allow the user to input a value.
- The commonly used elements for input
    - Textbox
    - Checkbox
    - RadioButton
    - ListBox
    - Dropdown etc.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Form input</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script>
      function RegisterClick(){
        document.getElementById("lblName").innerHTML =
document.getElementById("txtName").value;
        document.getElementById("lblPrice").innerHTML =
document.getElementById("txtPrice").value;
        document.getElementById("lblMfd").innerHTML =
document.getElementById("txtMfd").value;
        document.getElementById("lblShippedTo").innerHTML =
document.getElementById("lstShippedTo").value;
```

```
        stock = document.getElementById("optStock");
        status = "";
        if(stock.checked) {
           status = "Available";
        } else {
           status = "Out of Stock";
        }
        document.getElementById("lblStock").innerHTML = status;
     }
  </script>
</head>
<body class="container-fluid">
  <div class="row">
     <div class="col-3">
        <h3>Register Product</h3>
        <div class="form-group">
           <label>Name</label>
           <div>
              <input type="text" id="txtName" class="form-control">
           </div>
        </div>
        <div class="form-group">
           <label>Price</label>
           <div>
              <input type="text" id="txtPrice" class="form-control">
           </div>
```

```
        </div>
        <div class="form-group">
          <label>Manufactured</label>
          <div>
            <input type="date" id="txtMfd" class="form-control">
          </div>
        </div>
        <div class="form-group">
          <label>Shipped To</label>
          <div>
            <select class="form-control" id="lstShippedTo">
              <option>Delhi</option>
              <option>Hyderabad</option>
            </select>
          </div>
        </div>
        <div class="form-group">
          <label>In Stock</label>
          <div>
            <input type="checkbox" id="optStock"> Yes
          </div>
        </div>
        <div class="form-group">
          <button onclick="RegisterClick()" class="btn btn-primary btn-
block">Register</button>
        </div>
      </div>
```

```
<div class="col-9">
    <h3>Product Details</h3>
    <table class="table table-hover">
        <colgroup span="1" style="font-weight: bold; background-color: lightpink; color:white"></colgroup>
        <tr>
            <td>Name</td>
            <td id="lblName"></td>
        </tr>
        <tr>
            <td>Price</td>
            <td id="lblPrice"></td>
        </tr>
        <tr>
            <td>Manufactured</td>
            <td id="lblMfd"></td>
        </tr>
        <tr>
            <td>Shipped To</td>
            <td id="lblShippedTo"></td>
        </tr>
        <tr>
            <td>Stock </td>
            <td id="lblStock"></td>
        </tr>
    </table>
</div>
```

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

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Input</title>
    <style>
      label {
        font-weight: bold;
      }
      .box{
        width: 300px;
        justify-content: center;
        align-items: center;
        margin: auto;
      }
    </style>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script src="../node_modules/jquery/dist/jquery.js"></script>
    <script
src="../node_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>
    <script>
      function RegisterClick(){
```

```
            document.getElementById("lblName").innerText =
document.getElementById("txtName").value;
            document.getElementById("lblPrice").innerText =
document.getElementById("txtPrice").value;
            document.getElementById("lblCity").innerText =
document.getElementById("lstCities").value;


            stock = document.getElementById("optStock");
            status = "";
            if(stock.checked){
               status = "Available";
            } else {
               status = "Out of Stock";
            }
            document.getElementById("lblStock").innerText = status;
            document.getElementById("lblMfd").innerText =
document.getElementById("txtMfd").value;
         }
      </script>
   </head>
   <body class="container-fluid">
      <div class="box">
       <form>
          <h2 class="text-primary text-center">Register Product</h2>
          <div class="form-group">
           <label>Name</label>
           <div>
```

```
    <input type="text" id="txtName" class="form-control">
  </div>
</div>
<div class="form-group">
  <label>Price</label>
  <div>
    <input type="text" id="txtPrice" class="form-control">
  </div>
</div>
<div class="form-group">
  <label>Shipped To</label>
  <div>
    <select id="lstCities" class="form-control">
      <option value="Delhi">Delhi</option>
      <option value="Hyderabad">Hyderabad</option>
    </select>
  </div>
</div>
<div class="form-group">
  <label>Manufactured</label>
  <div>
    <input type="date" id="txtMfd" class="form-control">
  </div>
</div>
<div class="form-group">
  <label>In Stock</label>
```

```
            <div>
                <input type="checkbox" id="optStock"> Yes
            </div>
          </div>
          <div class="form-group">
            <button onclick="RegisterClick()" type="button" data-target="#summary" data-toggle="modal" class="btn btn-primary btn-block">Register</button>
          </div>
        </form>
      </div>
      <div class="modal fade" id="summary">
        <div class="modal-dialog">
          <div class="modal-content">
            <div class="modal-header">
              <h2>Product Details</h2>
              <button class="close" data-dismiss="modal" >x</button>
            </div>
            <div class="modal-body">
              <table class="table table-hover">
                <tr>
                  <td>Name</td>
                  <td id="lblName"></td>
                </tr>
                <tr>
                  <td>Price</td>
                  <td id="lblPrice"></td>
```

```
                    </tr>
                    <tr>
                      <td>Shipped To</td>
                      <td id="lblCity"></td>
                    </tr>
                    <tr>
                      <td>Stock</td>
                      <td id="lblStock"></td>
                    </tr>
                    <tr>
                      <td>Manufactured</td>
                      <td id="lblMfd"></td>
                    </tr>
                  </table>
                </div>
                <div class="modal-footer">
                  <button data-dismiss="modal" class="btn btn-
primary">OK</button>
                </div>
              </div>
            </div>
          </div>
        </body>
</html>
```

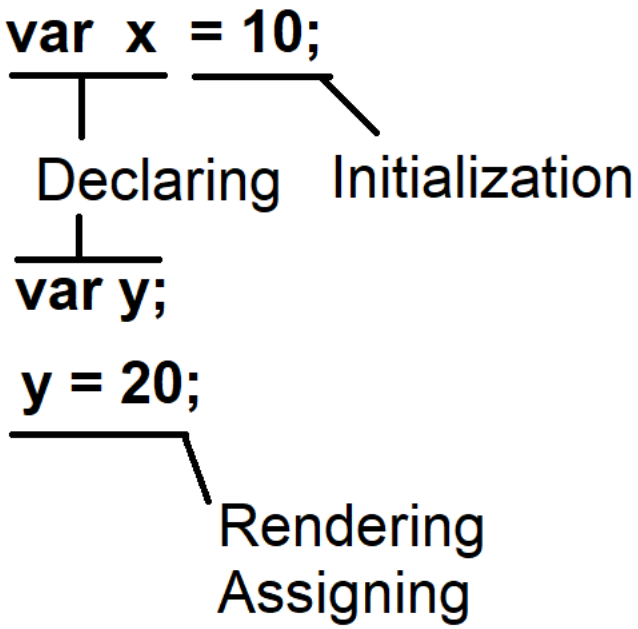
JavaScript Language Basics

- 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 give 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.
	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.

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:

```
<!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/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";
        }
      }
      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/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";
        }
      }
      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/bootstrap.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/bootstrap.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;
 }
```

Ex:

```
<!DOCTYPE html>
<html>
   <head>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script src="../node_modules/jquery/dist/jquery.js"></script>
```

```
    <script src="../node_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>
    <style>
        body {
            background-color: darkred;
            width: 90%;
        }
        .card {
            padding: 20px;
        }
    </style>
    <script>
        function OrderClick(){
            document.getElementById("lblName").innerHTML = document.getElementById("txtName").value;
            document.getElementById("lblMobile").innerHTML = document.getElementById("txtMobile").value;

            var mealName = '';
            var mealCost = 0;

            var adonName = '';
            var adonCost = 0;

            var total = 0;

            var optBurger = document.getElementById("optBurger");
```

```
var optRoller = document.getElementById("optRoller");

var optFries = document.getElementById("optFries");
var optKrusher = document.getElementById("optKrusher");

if(optBurger.checked) {
   mealCost = 130;
   mealName = optBurger.value;
}

if(optRoller.checked) {
   mealCost = 100;
   mealName = optRoller.value;
}

if(optFries.checked) {
   adonCost = 80;
   mealCost = mealCost + adonCost;
   adonName += optFries.value + "<br>";
}

if(optKrusher.checked) {
   adonCost = 40;
   mealCost = mealCost + adonCost;
   adonName += optKrusher.value + "<br>";
}
```

```
            total = mealCost;
            document.getElementById("lblMeal").innerHTML = mealName;
            document.getElementById("lblAdon").innerHTML = adonName;
            document.getElementById("lblTotal").innerHTML= "&#8377;" + total;
        }
    </script>
</head>
<body class="container-fluid">
    <header>
        <img src="../Images/kfctop.PNG" width="100%" height="150">
    </header>
    <section>
        <div class="accordion" id="orderForm">
            <div class="card">
                <div class="card-header">
                    <button data-target="#customerInfo" data-toggle="collapse" class="btn btn-danger btn-block">Customer Info</button>
                </div>
                <div class="collapse show" id="customerInfo" data-parent="#orderForm">
                    <div class="form-group">
                        <label>Customer Name</label>
                        <div>
                            <input type="text" id="txtName" class="form-control">
                        </div>
                    </div>
                    <div class="form-group">
```

```
                <label>Mobile Number</label>
                <div>
                  <input type="text" id="txtMobile" class="form-control">
                </div>
              </div>
            </div>
          </div>
          <div class="card">
            <div class="card-header">
              <button data-target="#mealInfo" data-toggle="collapse" class="btn btn-danger btn-block">Select Meal</button>
            </div>
            <div class="collapse" id="mealInfo" data-parent="#orderForm">
              <div class="row">
                <div class="col">
                  <div class="card">
                    <div class="card-header">
                      <h3>OMG Burger</h3>
                    </div>
                    <div class="card-body">
                      <img src="../Images/omg1.PNG">
                    </div>
                    <div class="card-footer">
                      <h4>
                        <input name="meal" id="optBurger" value="OMG Burger" type="radio"> &#8377; 130/-
                      </h4>
```

```
                    </div>
                  </div>
                </div>
                <div class="col">
                  <div class="card">
                    <div class="card-header">
                      <h3>OMG Roller</h3>
                    </div>
                    <div class="card-body">
                      <img src="../Images/omg2.PNG">
                    </div>
                    <div class="card-footer">
                      <h4>
                        <input name="meal" id="optRoller" value="OMG Roller" type="radio"> &#8377; 100/-
                      </h4>
                    </div>
                  </div>
                </div>
              </div>
            </div>
            <div class="card">
              <div class="card-header">
                <button data-target="#adonInfo" data-toggle="collapse" class="btn btn-danger btn-block">Select Ad-ON</button>
              </div>
```

```
<div class="collapse" id="adonInfo" data-parent="#orderForm">
  <div class="row">
    <div class="col">
      <div class="card">
        <div class="card-header">
          <h3>Large Fries</h3>
        </div>
        <div class="card-body">
          <img src="../Images/fries1.PNG">
        </div>
        <div class="card-footer">
          <h4>
            <input id="optFries" value="Large Fries"
type="checkbox"> &#8377; 80/-
          </h4>
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card">
        <div class="card-header">
          <h3>Krusher Brownie</h3>
        </div>
        <div class="card-body">
          <img src="../Images/krusher1.PNG">
        </div>
        <div class="card-footer">
```

```
                        <h4>
                            <input type="checkbox" id="optKrusher" value="Krusher Browser"> &#8377; 40/-
                        </h4>


                    </div>
                </div>


            </div>
            <button onclick="OrderClick()" data-target="#billSummary" data-toggle="modal" class="btn btn-danger btn-block">Place Order</button>
        </div>
    </div>
</div>
</section>
<div class="modal fade" id="billSummary">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h3>Bill Summary</h3>
        <button data-dismiss="modal" class="close">x</button>
      </div>
      <div class="modal-body">
        <table class="table table-hover">
          <tbody>
            <tr>
```

```
                    <td>Customer Name</td>
                    <td id="lblName"></td>
                  </tr>
                  <tr>
                    <td>Mobile</td>
                    <td id="lblMobile"></td>
                  </tr>
                  <tr>
                    <td>Meal Name</td>
                    <td id="lblMeal"></td>
                  </tr>
                  <tr>
                    <td>Ad-ONs</td>
                    <td id="lblAdon"></td>
                  </tr>
                  <tr>
                    <td>Total Amount</td>
                    <td id="lblTotal"></td>
                  </tr>
                </tbody>
              </table>
            </div>
            <div class="modal-footer">
              <button data-dismiss="modal" class="btn btn-
primary">OK</button>
            </div>
          </div>
```

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

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

Array Type

- Arrays in computer programming are used to **reduce overhead and complexity**.
- JavaScript **Array can store different types of values** in sequential order.
- It can reduce overhead by storing values in sequential order.
- It can reduce complexity by storing multiple values under one name.
- **Array size can be changed dynamically.**
- Array in JavaScript have the behaviour of collections like stack, queue, hash table.

Declaring Array:

- Array can be declared by using
    - Array Meta Character “[]”
    - Array Constructor “Array()”

Ex:

```
<script>
  function f1(){
    var categories = [];
    var products = new Array();
  }
  f1();
</script>
```

Initialize values into Array:

```
<script>
  function f1(){
    var categories = ["Electronics","Footwear"];
    var products = new Array("Speaker","Nike Causals");
  }
```

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

Assign Values by using Array Property

- Property is used to map with index number in memory.
- So you can use property to access and or send value into memory.

Ex:

```
<script>
  function f1(){
    var categories = [];
    categories["0"] = "Electronics";
    categories["1"] = "Footwear";
    for(var property in categories) {
       document.write(`${property} : [${typeof property}]<br>`);
    }
  }
  f1();
</script>
```

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Slide Show</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../Fonts/css/all.css">
    <script>
```

```
        var products = ["../Images/jblspeaker.jpg", "../Images/earpods.jpg",
"../Images/shoe.jpg", "../Images/shirt.jpg"];
        var productNames = ["JBL Speaker","Earpods", "Nike Casuals", "Shirt"];
        function loadImage(index){
            var pic = document.getElementById("pic");
            pic.src= products[index];
            var lblName = document.getElementById("lblName");
            lblName.innerHTML = productNames[index];
        }
        var index = 0;
        function NextClick(){
          index++;
          loadImage(index);
        }
        function PreviousClick(){
            index--;
            loadImage(index);
        }
        function SlideShow(){
            var txtRange = document.getElementById("txtRange").value;
            loadImage(txtRange);
        }
    </script>
  </head>
  <body class="container-fluid" onload="loadImage(0)">
    <div class="card">
      <div class="card-header text-center">
```

```
            <h3 id="lblName"></h3>
        </div>
        <div class="card-body text-center">
            <button onclick="PreviousClick()" class="btn btn-outline-danger">
                <span>&lt;</span>
            </button>
            <img id="pic" width="500" height="400">
            <button onclick="NextClick()" class="btn btn-outline-danger">
                <span>&gt;</span>
            </button>
        </div>
        <div class="card-footer text-center">
            <input onchange="SlideShow()" class="form-control-range"
id="txtRange" type="range" min="0" value="0" max="3">
        </div>
    </div>
  </body>
</html>
```

Array Manipulation

Read Array Elements:

Method	Description

toString()	Returns array elements separated with comma.
	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	document.write(products.toString());
	}
	f1();
	</script>

join()	Returns array elements separated with custom delimiter.

	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	document.write(products.join("-->"));
	}
	f1();
	</script>

slice()	Return array element between specified index.
	
	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	document.write(products.slice(1,2));
	}
	f1();
	</script>

for..of	It reads and return all array elements in sequential order.
	
	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	for(var item of products) {
	document.write(item + "
	");
	}
	}
	f1();
	</script>

for..in	It reads and return all array properties.
	
	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	for(var item in products) {
	document.write(item + "
	");
	}
	}
	f1();
	</script>

for	It uses a loop to read all elements by using initialization, condition and counter. Syntax: for(initializer, condition, iterator) {

	}
	
	Ex:
	<script>
	function f1(){
	var products = ["TV", "Mobile", "Shoe"];
	for(var i=0; i<products.length; i++) {
	document.write(products[i] + "
	");
	}
	}
	f1();
	</script>

Add Array Elements into HTML Page to present as DOM elements:

- To Add any element you have first create element by using the method document.createElement(“elementName”)
- You can add element by using the method append(), appendChild()

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Array</title>
    <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script>
      var categories = ["All", "Electronics", "Footwear", "Fashion", "Accessories"];
      function bodyload(){
        var lstCategories = document.getElementById("lstCategories");
        var optCategories = document.getElementById("optCategories");
        for(var item of categories) {
          var li = document.createElement("li");
          li.innerHTML = item;
          lstCategories.appendChild(li);
```

```
                var option = document.createElement("option");
                option.text = item;
                option.value = item;
                optCategories.appendChild(option);
            }

        }
    </script>
</head>
<body onload="bodyload()" class="container-fluid">
    <div class="form-group">
      <h3>Select a Category</h3>
      <ol id="lstCategories">

      </ol>
    </div>
    <div class="form-group">
        <h3>Select Category</h3>
        <select class="form-control" id="optCategories">

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

Adding and Removing Elements from Array:

Method	Description
push()	Add new elements as last item.
unshift()	Add new elements as first item.
pop()	Remove and return last item.
shift()	Remove and return first item.

splice()	It is used to add or remove item at any specific index. Syntax: splice(startIndex, removeCount, NewItems…)

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Array</title>
    <link rel="stylesheet" href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script>
      var categories = ["All", "Electronics", "Footwear"];
      function bodyload(){
        var lstCategories = document.getElementById("lstCategories");
        var optCategories = document.getElementById("optCategories");
        lstCategories.innerHTML="";
        optCategories.innerHTML="";
        for(var item of categories) {
          var li = document.createElement("li");
          li.innerHTML = item;
          lstCategories.appendChild(li);

          var option = document.createElement("option");
          option.text = item;
          option.value = item;
          optCategories.appendChild(option);
        }
      }
      function AddClick(){
        var txtName = document.getElementById("txtName");
```

```
            categories.splice(1,0,txtName.value);
            alert("Item Added");
            txtName.value="";
            bodyload();
        }
        function RemoveClick(){
            var item = categories.shift();
            alert(`${item} Removed`);
            bodyload();
        }
        function RemoveSelected(){
            var selectedItem = document.getElementById("optCategories").value;
            var selectedIndex = categories.indexOf(selectedItem);
            var c = confirm("Are you Sure Want to Delete?");
            if(c==true) {
                categories.splice(selectedIndex,1);
                bodyload();
            }
        }
    </script>
</head>
<body onload="bodyload()" class="container-fluid">
    <div class="form-group">
        <label>Add Category</label>
        <div>
            <input id="txtName" type="text">
            <button onclick="AddClick()">Add</button>
        </div>
    </div>
```

```
        <div class="form-group">
          <h3>Select a Category</h3>
          <ol id="lstCategories">

          </ol>
          <div>
            <button onclick="RemoveClick()">Remove</button>
          </div>
        </div>
        <div class="form-group">
          <h3>Select Category</h3>
          <select size="3" class="form-control" id="optCategories">

          </select>
          <div>
            <button onclick="RemoveSelected()">Remove Selected</button>
          </div>
        </div>
    </body>
</html>
```

Hotel Registration

Customer Info

Customer Name: text
Check in Date: date
Total No of Days: number
Total No of People: number

Room Type

Image
2000/day
○ Delux Room

Image
3000/day
○ Suite Room

Aminities

Image
500/day
☐ Locker

Image
1000/-
☐ A/C

Advance

Mandatory

Register

Bill Summary

Cusomer Name :
Check in Date :
Total No of Days:
Total No People :
Room Type:
Aminities:
Advance :
Balance :

Tdays = 2
Delux: 2000/day 2x2000
Aminities
A/C = 1000x2 4000
2000
Advance = 2000 - 2000
======
Balance

Searching for Elements in Array

indexOf()	It can search for element in array based on given string and returns the “index” number.
lastIndexOf()	It returns the last occurrence index number.

find()	It finds and returns the first occurrence element that matches the given condition.
	
	Ex:
	<script>
	function f1(){
	var sales = [34500, 20000, 45000, 12000, 30000];
	var result = sales.find(function(val){
	return val>30000;
	});
	document.write(result);
	}
	f1();
	</script>

filter()	It finds and returns all elements that matches the given condition.
	Ex:
	<script>
	function f1(){
	var sales = [34500, 20000, 45000, 12000, 30000];
	var result = sales.filter(function(val){
	return val<=30000;
	});
	document.write(result.toString());
	}
	f1();
	</script>

Ex:

```
<script>
   function f1(){
      var sales = [34500, 20000, 45000, 12000, 30000];
      function search(val){
         return val>=30000;
      }
      var result = sales.find(search);
      document.write(result);
   }
```

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

Sort Array Elements

- sort() arranges elements in ascending order.
- reverse() arranges elements in reverse order [bottom to top]

Ex:

```
<script>
  function f1(){
    var sales = [34500, 20000, 45000, 12000, 30000];
    function search(val){
      return val>=30000;
    }
    var result = sales.filter(search);
    result.sort();
    result.reverse();
    document.write(result.toString());
  }
  f1();
</script>
```

FAQ:

1. **What type of values we can store in array?**
   You can store any type of value.
2. **We store function in Array?**
   Yes.
   Ex:

```
<script>
  function f1(){
```

```
    var methods = [function(){return "Hello !"}, function(a, b){return a +
b}];
    document.write(methods[0]()  + "<br>");
    document.write(methods[1](10,20));
  }
  f1();
</script>
```

3. **What is Array Destruction?**

   It is a technique used to access array elements and store in individual memory references.

   Ex:

```
<script>
  function f1(){
    var methods = [function(){return "Hello !"}, function(a, b){return a +
b}];
    //Without Destruction
    var m1 = methods[0];
    var m2 = methods[1];
    document.write(m1() + "<br>");
    document.write(m2(10,30) + "<br>");
    // With Destruction
    var [x1, x2] = methods;
    document.write(x1() + "<br>");
    document.write(x2(10,20));
  }
  f1();
</script>
```

4. **Can we define Array inside Array [Multi Dimension]?**

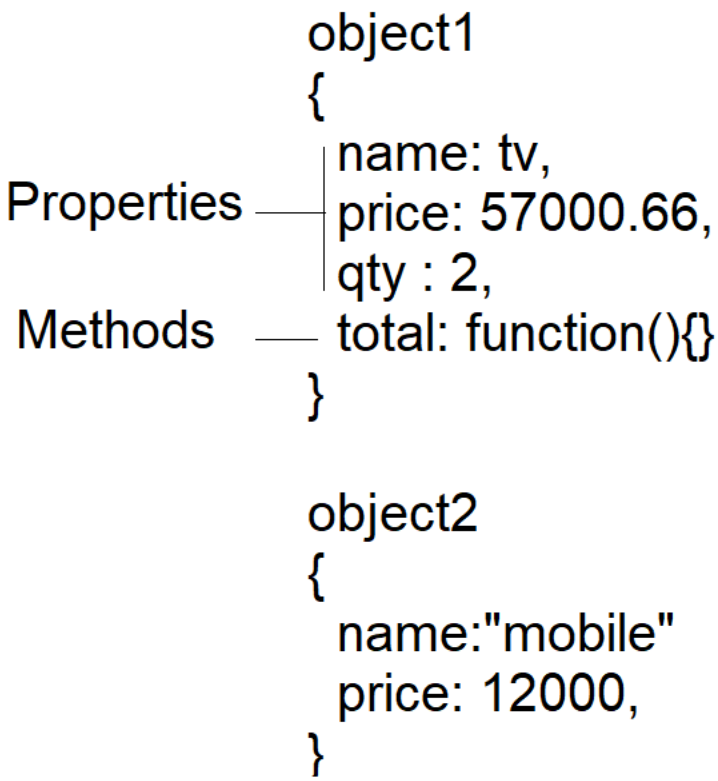
   Yes.

   Ex:

```
<script>
  function f1(){
    var values = [[10,20],["A","B"]];
    document.write(values[0][1]);
  }
  f1();
</script>
```

Object Type

- Object in computer programming was introduced in early 1960's by "**Alan Kay**".
- Object can keep all related data and functionality at one memory reference.
- Object comprises of data and functionality.
- Object is a set of properties and methods.



- Data is stored in properties.
- Functionality is defined in methods.
- In JavaScript early version object is also known as "pseudo class".

Syntax:

```
var object = {
  property: value,
  method: function(){}
```

}

- You can access object property within the object by using “this” keyword.
- You can access object property outside the object by using object name.
  Syntax:
  object
  {
  this.property
  this.method()
  }
  object.property
  object.method()
- Later in early 1967 “**Johan Olay, Kristian Nygaard**” introduced the concept of reusing object with class. [OOP]
- The first OOP language was **SIMULA 67**, Small Talk, C++, Java, .NET Languages

Ex:

```
<script>
  function f1(){
    var product = {
      Name: "",
      Price: 0,
      Qty: 0,
      Total: function(){
        return this.Qty * this.Price;
      },
      Print: function(){
        document.write(`
         Name : ${this.Name} <br>
         Price: ${this.Price}<br>
```

```
            Qty: ${this.Qty}<br>
            Total: ${this.Total()}
           <br>`);
        }
    }
    product.Name = "Samsung TV";
    product.Price = 4000.44;
    product.Qty = 2;
    product.Print();
    document.write("<hr>");
    product.Name = "Nike Casuals";
    product.Price = 2000.44;
    product.Qty = 3;
    product.Print();

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

JSON Type Data

[JavaScript Object Notation]

- It is a format for data.
- It is a collection objects.

  Ex:

```
<script>
   function f1(){
     var products = [
        {Name: "TV", Price: 45000.44, Cities:['Delhi', 'Hyd']},
        {Name: "Mobile", Price: 12000.33, Cities: ['Hyd','Chennai']}
     ];
```

```
            for(var product of products) {
               document.write(product.Name + "-" + product.Price + "-" +
        product.Cities.toString() + "<br>");
            }
          }
          f1();
        </script>
```

Ex: Filtering of data

```
<script>
  function f1(){
    var products = [
      {Name: "TV", Price: 45000.44, Cities:['Delhi', 'Hyd'], Category:
"Electronics"},
      {Name: "Mobile", Price: 12000.33, Cities: ['Hyd','Chennai'],
Category:"Electronics"},
      {Name: "Nike Casuals", Price: 4000.44, Cities: ['Chennai', 'Hyd'],
Category: "Footwear"}
    ];
    var result = products.filter(function(product){
      return product.Category=="Electronics";
    })
    for(var item of result) {
      document.write(item.Name + "-" + item.Price + "<br>");
    }
  }
  f1();
</script>
```

Ex: JSON Type Data for Products

```
<!DOCTYPE html>
<html>
  <head>
    <title>Slide Show</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../Fonts/css/all.css">
    <script>
      var products = [
        {Name: "JBL Speaker", Price: 4500.55, Photo:
"../Images/jblspeaker.jpg"},
        {Name: "Earpods", Price: 2500.55, Photo: "../Images/earpods.jpg"},
        {Name: "Nike Casuals", Price: 6500.55, Photo: "../Images/shoe.jpg"},
        {Name: "Lee Boot", Price: 1500.55, Photo: "../Images/shoe1.jpg"},
      ];
      function loadImage(index){
        var pic = document.getElementById("pic");
        pic.src= products[index].Photo;
        var lblName = document.getElementById("lblName");
        var lblPrice = document.getElementById("lblPrice");
        lblName.innerHTML = products[index].Name;
        lblPrice.innerHTML = "&#8377;" + products[index].Price;
      }
      var index = 0;
      function NextClick(){
        index++;
        loadImage(index);
```

```
        }
        function PreviousClick(){
            index--;
            loadImage(index);
        }
        function SlideShow(){
            var txtRange = document.getElementById("txtRange").value;
            loadImage(txtRange);
        }
    </script>
</head>
<body class="container-fluid" onload="loadImage(0)">
    <div class="card">
        <div class="card-header text-center">
            <h3 id="lblName"></h3>
        </div>
        <div class="card-body text-center">
            <button onclick="PreviousClick()" class="btn btn-outline-danger">
                <span>&lt;</span>
            </button>
            <img id="pic" width="500" height="400">
            <button onclick="NextClick()" class="btn btn-outline-danger">
                <span>&gt;</span>
            </button>
        </div>
        <div class="card-footer text-center">
```

```
            <h3 id="lblPrice"></h3>
            <input onchange="SlideShow()" class="form-control-range"
id="txtRange" type="range" min="0" value="0" max="3">
        </div>
      </div>
   </body>
</html>
```

Ex: Adding Rows into Table Dynamically

```
<!DOCTYPE html>
<html>
   <head>
      <title>Dynamic Table</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
          var products = [
            {Name: "JBL Speaker", Price: 4500.55, Photo:
"../Images/jblspeaker.jpg"},
            {Name: "Earpods", Price: 2500.55, Photo: "../Images/earpods.jpg"},
            {Name: "Nike Casuals", Price: 6500.55, Photo: "../Images/shoe.jpg"},
            {Name: "Lee Boot", Price: 1500.55, Photo: "../Images/shoe1.jpg"},
         ];
         function bodyload(){
            var tbody= document.getElementById("tbody");
            for(var item of products)
            {
```

```
            var tr = document.createElement("tr");
            var tdName = document.createElement("td");
            var tdPrice = document.createElement("td");
            var tdPhoto = document.createElement("td");

            tdName.innerHTML = item.Name;
            tdPrice.innerHTML = item.Price;

            var pic = new Image();
            pic.src= item.Photo;
            pic.height="50";
            pic.width="50";

            tdPhoto.appendChild(pic);
            tr.appendChild(tdName);
            tr.appendChild(tdPrice);
            tr.appendChild(tdPhoto);

            tbody.appendChild(tr);
        }
    }
  </script>
</head>
<body onload="bodyload()" class="container-fluid">
  <h2>Product Details</h2>
  <table class="table table-hover">
```

```
        <thead>
          <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Preview</th>
          </tr>
        </thead>
        <tbody id="tbody">

        </tbody>
      </table>
   </body>
</html>
```

Ex: **Dynamic List with Nested Data**

```
<!DOCTYPE html>
<html>
   <head>
      <title>Dynamic List</title>
      <script>
         var data = [
            {Category: "Electronics", Products:["JBL Speaker", "Earpods"]},
            {Category: "Footwear", Products: ["Nike Casuals", "Lee Cooper Boot"]},
            {Category: "Fashion", Products: ["Watch", "Shirt", "Jeans"]}
         ];
         function bodyload(){
```

```
            var lstCategories = document.getElementById("lstCategories");
            for(var item of data)
            {
               var parentLi = document.createElement("li");
               parentLi.innerHTML = item.Category;
               lstCategories.appendChild(parentLi);
               for(var product of item.Products)
               {
                  var ul = document.createElement("ul");
                  var childLi = document.createElement("li");
                  childLi.innerHTML = product;
                  ul.appendChild(childLi);
                  parentLi.appendChild(ul);
               }
            }
         }
      </script>
   </head>
   <body onload="bodyload()">
      <ol id="lstCategories">

      </ol>
   </body>
</html>
```

Ex: Dynamically Adding into Table

```
<!DOCTYPE html>
<html>
  <head>
    <title>Dyamic Insert</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script>
      var data = [
        {Name: "Samsung TV", Price: 45000.55},
        {Name: "Nike Casuals", Price: 4200.33}
      ];
      function LoadTable(){
        var tbody = document.getElementById("tbody");
        tbody.innerHTML="";
        for(var item of data){
          var tr = document.createElement("tr");
          var tdName = document.createElement("td");
          var tdPrice = document.createElement("td");

          tdName.innerHTML = item.Name;
          tdPrice.innerHTML = item.Price;

          tr.appendChild(tdName);
          tr.appendChild(tdPrice);

          tbody.appendChild(tr);
```

```
        }
    }
    function bodyload(){
        LoadTable();
    }
    var newObject = {
        Name: "",
        Price: 0
    };
    function AddClick(){
        var txtName = document.getElementById("txtName");
        var txtPrice = document.getElementById("txtPrice");

        newObject = {
            Name: txtName.value,
            Price: txtPrice.value
        }
        data.push(newObject);
        alert("Record Added");
        txtName.value="";
        txtPrice.value="";
        LoadTable();
    }
  </script>
</head>
<body onload="bodyload()" class="container-fluid">
```

```
<div class="row">
    <div class="col-3">
        <h3>Register Product</h3>
        <div class="form-group">
            <label>Name</label>
            <div>
                <input type="text" id="txtName" class="form-control">
            </div>
        </div>
        <div class="form-group">
            <label>Price</label>
            <div>
                <input type="text" id="txtPrice" class="form-control">
            </div>
        </div>
        <div class="form-group">
            <button onclick="AddClick()" class="btn btn-primary btn-
block">Add Product</button>
        </div>
    </div>
    <div class="col-9">
        <table class="table table-hover">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
                </tr>
```

```
            </thead>
            <tbody id="tbody">

            </tbody>
          </table>
        </div>
      </div>
    </body>
</html>
```

Regular Expression

- Regular expression is used to verify the format of input value.
- Expression is defined by using meta character and quantifiers enclosed in “/ /”
- Regular expression is verified by using “match()”.
- match() is a boolean method that return true if expression is matching with value.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Regular Expression</title>
    <script>
      function VerifyPassword(){
        var txtPwd = document.getElementById(“txtPwd”).value;
        var regExp = /(?=.*[A-Z])\w{4,10}/;
        var msg = document.getElementById(“msg”);
        var grade= document.getElementById(“grade”);
```

```
        function ShowGrade(mn, mx, val){
            grade.min = mn;
            grade.max= mx;
            grade.value = val;
        }

        if(txtPwd.match(regExp)){
            msg.innerHTML="Strong Password".fontcolor('green');
            ShowGrade(1,100,100);
        } else {
            if(txtPwd.length<4){
                msg.innerHTML="Poor Password".fontcolor('red');
                ShowGrade(1,100,20);
            } else {
                msg.innerHTML="Weak Password".fontcolor('orange');
                ShowGrade(1,100,70);
            }
        }
    }
</script>
<style>
    progress{
        height: 20px;
        width: 150px;
    }
```

```
    </style>
  </head>
  <body>
    <fieldset>
      <legend>Password</legend>
      <div>
        <input onkeyup=”VerifyPassword()” id=”txtPwd” type=”password”>
        <div>
          <progress id=”grade” min=”1” value=”0” max=”100”></progress>
        </div>
        <div id=”msg”></div>
      </div>
    </fieldset>
  </body>
</html>
```

JavaScript Date Type

- JavaScript date values are handle by using “Date()” constructor.
- It allocates memory to store date and time type value.
- In the memory date is defined as string with “yy-mm-dd” format.
  Syntax:
  var mfd = new Date(“YY-MM-DD”);
  var mfd = new Date(); // it loads current date into memory.
- To Access date values JavaScript provides several methods
  - getHours() – 0 to 23
  - getMinutes() – 0 to 59
  - getSeconds() – 0 to 59
  - getMilliseconds() – 0 to 999

- getDate() – returns the date number
- getDay() – returns the weekday number [0=Sunday]
- getMonth() – returns the month number [0=January]
- getFullYear()
- toString()
- toLocaleDateString()
- toLocaleTimeString()
- toDateString()
- toTimeString()

Ex:

```
<script>
   function f1(){
      var product = {
         Name: "Samsung TV",
         Price: 45000.55,
         InStock: true,
         Mfd: new Date("2020-03-18")
      };
      var months = ["January", "Feb", "March", "Apr"];
      var weekdays = ["Sunday", "Monday", "Tue", "Wednesday", "Thu"];

document.write(`Name=${product.Name}<br>Price=${product.Price}<br>InStock=${product.InStock}<br>
         Manufactured Month: ${months[product.Mfd.getMonth()]} <br>
         Manufactured Weekday: ${weekdays[product.Mfd.getDay()]} <br>
         Manufactured Date: ${product.Mfd.getDate()} <br>
         Manufactured Year: ${product.Mfd.getFullYear()}<br>
         Date: ${product.Mfd.toLocaleDateString()}
```

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

- You can set date dynamically by using “set” methods.
  - setMonth()
  - setDate()
  - setFullYear()
  - setHours()
  - setMinutes()
  - setSeconds()
  - setMilliseconds()

Ex:

```
<script>
  function f1(){
    var product = {
      Name: "Samsung TV",
      Price: 45000.55,
      InStock: true,
      Mfd: new Date("2020-03-18")
    };
    var months = ["January", "Feb", "March", "Apr"];
    var weekdays = ["Sunday", "Monday", "Tue", "Wednesday", "Thu"];
    product.Mfd.setFullYear(2021);
    product.Mfd.setMonth(1);
document.write(`Name=${product.Name}<br>Price=${product.Price}<br>InStock=${product.InStock}<br>
      Manufactured Month: ${months[product.Mfd.getMonth()]} <br>
      Manufactured Weekday: ${weekdays[product.Mfd.getDay()]} <br>
```

```
        Manufactured Date: ${product.Mfd.getDate()} <br>
        Manufactured Year: ${product.Mfd.getFullYear()}<br>
        Date: ${product.Mfd.toLocaleDateString()}
      `);
  }
  f1();
</script>
```

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Date</title>
    <style>
      .container{
        width: 600px;
        height: 400px;
        justify-content: center;
        align-items: center;
        margin:auto;
      }
    </style>
    <script>
      function bodyload(){
        var now = new Date();
        var hrs = now.getHours();
        var pic = document.getElementById("pic");
```

```
            if(hrs>=00 && hrs<=12) {
               pic.src="../Images/morning.gif";
            } else if(hrs>12 && hrs<=17) {
               pic.src="../Images/afternoon.gif";
            } else {
               pic.src="../Images/evening.gif";
            }
         }
      </script>
   </head>
   <body onload="bodyload()">
      <div class="container">
         <img id="pic" width="300" height="300">
      </div>
   </body>
</html>
```

Summary

- Primitive Types
    - Number
    - String
    - Boolean
    - Null
    - Undefined
- Non-Primitive Types
    - Array
    - Object
    - Regular Expression
    - Date

JavaScript Operators and Expressions

- Operator is an object that evaluates a value.
- Based on what type of value an operator evaluates the operators are classified into following types:
    - Arithmetic Operators
    - Logical Operators
    - Comparison Operators
    - Assignment Operators
    - Special Operators
- Operators are also classed into 3 type based on the number of operands they can handle.
    - Unary : One Operand [x++]
    - Binary : Two operands [x + y]
    - Ternary : Three operands [(condition)?true:false]

Arithmetic Operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus division
**	Exponent [new in ES5] [old version – Math.pow()]
++	Increment
--	Decrement

Ex:

```
<script>
   function f1(){
      var x = 2;
```

```
    var y = 3;
    document.write(`x**y=${x**y} <br>`);
    document.write(`Power=${Math.pow(x,y)}`);
  }
  f1();
</script>
```