INTRODUCTION TO JAVASCRIPT (JS)

INTRODUCTION

- To JavaScript is a scripting language most often used for client-side web development.
- JavaScript is an implementation of the ECMAScript standard.
  - The ECMAScript only defines the syntax/characteristics of the language and a basic set of commonly used objects such as Number, Date, Regular Expression, etc.
- JScript is the Microsoft version of JavaScript.
- JavaScript and Java are completely different languages.

JAVASCRIPT IN WEB

- To create interactive user interface in a web page (e.g., menu, pop-up alert, windows, etc.).
- Manipulating web content dynamically
  - The Change the content and style of an element
  - Replace images on a page without page reload
  - Hide/Show contents
- Generate HTML contents on the fly
- Form validation
- AJAX (e.g. Google complete)

JAVASCRIPT IN WEB

Used to enhance the functionality and appearance of web pages.

EMBEDDING JAVASCRIPT

- JavaScript is embedded in HTML using <script> element.
- Any number of scripts can be placed in a web page.
- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.
- The scripts inside an HTML document is interpreted in the order they appear in the document.

EMBEDDING JAVASCRIPT

```
<html>
    <head><title>First JavaScript Page</title></head>
    <body>
            <h1>First JavaScript WebPage</h1>
            <script type="text/javascript">
              document.write("<hr>");
              document.write(“Welcome to JS");
              document.write("<hr>");
            </script>
    </body>
</html>
```

EMBEDDING JAVASCRIPT

```
<html>
    <head><title>First JavaScript Page</title>
    <script>
        function myFunction() {


                                        }
    </script>
    </head>
    <body>
            <h1>First JavaScript WebPage</h1>
    </body>
</html>
```

EXTERNAL JAVASCRIPT

- JavaScript can be written in separate files from HTML.
- External scripts are used to utilize the same code in many different web pages.
- External JavaScript files have .js file extension.
- The src attribute of a <script> is used to include JavaScript codes from an external file.
- External scripts cannot contain <script> tags.

EXTERNAL JAVASCRIPT

```
<html>
    <head><title>First JavaScript Page</title></head>
    <body>
            <h1>First JavaScript WebPage</h1>
            <script type="text/javascript" src="Welcome.js" >


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

EXTERNAL JAVASCRIPT

```
<html>
    <head><title>First JavaScript Page</title></head>
    <body>
            <h1>First JavaScript WebPage</h1>
            <script type="text/javascript" src="Welcome.js" >


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

Welcome.js

```
document.write("<hr>");
document.write("Welcome to JS");
document.write("<hr>");
```

JAVASCRIPT (JS)
VARIABLES & DATA TYPES

JAVASCRIPT STATEMENTS

- A JavaScript program is collection of JavaScript Statements.
- Statement is composed of
  - Values
  - Operators
  - Expressions
  - Keywords
  - Comments

JAVASCRIPT STATEMENTS

- Statements are executed in the same order as written.
- Semicolon (;) separates statements, but not required.
- curly brackets {...} defines a group of statements (code block) in code.

JAVASCRIPT VALUES

- JavaScript defines two type values:
  - Fixed : Literals
    - Numbers : It can be a decimal value.

      1, 123, 11.45, 45.7
    - Strings : It is a text, written using double or single quotes.

      “JavaScript” , ‘JavaScript’
  - Variable : Variables
    - Locations in memory to store data values.

JAVASCRIPT VARIABLES

- It is a container to store data values.
- Every variable has a name, a type and a value.
- JavaScript does not require variables to have a declared type before they can be used in a script.
- A variable in JavaScript can contain a value of any data type.
- JavaScript is referred to as a loosely typed language.

```
VarName = Value;
```

VARIABLE SCOPE

- There are two types of scopes:
  - Global Scope
    - A variable declared without any prefix have by default global scope.
  - Local Scope
    - Variables are declared using let keyword.
    - var is also used to declare function scoped variables

JS DATA TYPES

- Primitive data types
  - Number: integer & floating-point numbers
  - Boolean: true or false
  - String: a sequence of alphanumeric characters
- Composite data types (or Complex data types)
  - Object: a named collection of data
  - Array: a sequence of values
- Special data types
  - Null: the only value is "null" – to represent nothing.
  - Undefined: the only value is "undefined" – to represent the value of an uninitialized variable

JS COMMENTS

- Two type of comments are used in JavaScript
  - Single Line:
    - Single line comments start with //

```
// This line will not be executed
```

  - Multi Line:
    - Multi-line comments start with /* and end with */.

```
/* Line 1
   Line 2  */
```

JAVASCRIPT (JS)
OPERATORS
&
CONDITIONAL STATEMENTS

JS OPERATORS

- JavaScript supports
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - Logical Operators
  - Conditional Operator
  - Typeof Operator

JS OPERATORS

- JavaScript supports
  - Arithmetic Operators

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	+	a + b	a + b
Subtraction	-	x - y	x - y
Multiplication	*	ab	a * b
Division	/	x/y or $\frac{x}{y}$ or x÷y	x / y
Remainder	%	r mod s	r % s

Addition operator (+) works for Numeric as well as Strings.

JS OPERATORS

- JavaScript supports
  - Assignment Operators

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Simple Assignment	=	x = y	x = y
Add and Assignment	+=	a = a + b	a += b
Subtract and Assignment	-=	x = x - y	x -= y
Multiply and Assignment	*=	a = ab	a *= b
Divide and Assignment	/=	x = x/y or $\frac{x}{y}$ or x÷y	x /= y
Remainder and Assignment	%=	r = r mod s	r % = s

JS OPERATORS

- JavaScript supports
  - Comparison Operators

JavaScript operation	Arithmetic operator	Description	JavaScript expression
Equal	==	Checks if the value of two operands are equal or not	x == y
Not Equal	!=		a != b
Greater than	>	Checks if the value of the left operand is greater than or less than the value of the right operand	x > y
Less than	<		a < b
Greater than or Equal to	>=	Checks if the value of the left operand is greater than or less than or equal the value of the right operand	x >= y
Less than or Equal to	<=		a <= b

JS OPERATORS

- JavaScript supports
  - Comparison Operators
    - ===, !== (Strictly equals and strictly not equals)
    - Type and value of operand must match / must not match

JS OPERATORS

- JavaScript supports
  - Logical Operators

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Logical AND	&&	Returns true, If both the operands are non-zero	x && y
Logical OR	||	Returns true, If any one of the two operands is non-zero	a || b
Logical NOT	!	Reverses the logical state of its operand.	!x

JS OPERATORS

- JavaScript supports
  - Conditional Operator (? :)

```
(Condition) ? Exp-T : Exp-F
```

  - typeof Operator

```
typeof  Exp
```

    - Its value is a string indicating the data type of the operand.

JS OPERATORS

- Operator Precedence & Associativity

Operators	Associativity
* / %	Left to Right
+ -	Left to Right
< <= > >=	Left to Right
== != === !===	Left to Right
=	Right to Left

JS CONDITIONAL STATEMENTS

- JavaScript supports
  - if

```
if (condition) {
                    // if block
                    }
```

JS CONDITIONAL STATEMENTS

- JavaScript supports
  - if
  - if …..else

```
if (condition) {
                    // if block
                    }
else                {
                    // else block
                    }
```

JS CONDITIONAL STATEMENTS

- JavaScript supports
  - if
  - if …..else
  - else if

```
if (condition1)        {
                              // if c1 block
                       }
else if (condition2)   {
                              // if c2 block
                       }
else                   {
                       // else block
                       }
```

JS CONDITIONAL STATEMENTS

- JavaScript supports
  - if
  - if …..else
  - else if
  - switch

```
switch (expression) {
                case x:
                        // code block
                        break;
                case y:
                        // code block
                        break;
                default:
                        // code block
                                }
```

JS ITERATIVE STATEMENTS

- JavaScript supports
  - for

```
for ( init ; condition ; inc\dec )
        {
          // code block to be repeated
        }
```

JS ITERATIVE STATEMENTS

- JavaScript supports
  - for
  - for / in
  - for / of
  - while

```
while ( condition )
        {
          // code block to be repeated
        }
```

JS ITERATIVE STATEMENTS

- JavaScript supports
  - for
  - for / in
  - for / of
  - while
  - do / while

```
do {
        // code block to be repeated
    } while ( condition ) ;
```

JAVASCRIPT (JS) FUNCTIONS

JS FUNCTIONS

- A function is set of statements that takes do some computation on given inputs and produce output to fulfill a task.
- JavaScript also supports functions.
- It supports two type of functions:
  - User-defined functions
  - In-built functions

JS FUNCTIONS

- User-defined functions
  - A user defined function is defined by **function** keyword.
  - Syntax:

```
function functionName(Param1, Param2, …ParamN)
                                {
                                        // function body
                                }
```

  - A function can be called by function name with list of parameter values.

```
functionName(V1, V2, …Vn);
```

JS FUNCTIONS

- User-defined functions
  - **return** statement is used to get a return value from function.

```
return value;
```

  - A JavaScript function can also be defined with an expression that will be stored in a variable.

```
var x = function ( a , b) { return a * b };
```

  - Variable can be used as a function.
  - This is also called anonymous function.

JS FUNCTIONS

- User-defined functions
  - Functions can also be defined using a built-in JavaScript function constructor called **Function()**.

```
var functionName = new Function( “a” , “b”, “return exp” )
```

  - It is possible to call JavaScript functions before declaration, but not anonymous functions.
  - An anonymous self-invoking functions are invoked automatically, without being called.

```
( function ( ) { //Body of function } ) ( );
```

JS FUNCTIONS

- In-built functions
  - JS also have In-built functions that can be called in any script without declaring.

Function	Description
encodeURI()	encode a URI
decodeURI()	decode a URI
eval()	evaluates or executes an argument.
isFinite()	determines whether a number is a finite, legal number.
isNaN()	determines whether a value is an illegal number.

JS FUNCTIONS

- In-built functions
  - JS also have In-built functions that can be called in any script without declaring.

Function	Description
Number()	converts the object argument to a number that represents the object's value.
parseFloat()	parses a string and returns a floating point number.
parseInt()	parses a string and returns an integer.
String()	converts the value of an object to a string.

JAVASCRIPT
OBJECTS & ARRAYS

JS OBJECTS

- JS object is a collection of properties (named values).
- A property is a variable or JS function.
- It is written as:

```
name : value
```

- A property value can be primitive, function or other JS object.
- A property with function definition as a value is known as method.
- Methods are actions that can be performed on object.

JS OBJECTS

- JS object can be created by different ways:
  - Using object literal
    - Object literal is a set of named value pair enclosed in { }.

```
var objName = {
                    p1  : val1 ,
                    p2  : val2 ,
                    m1 : function( ) {   }
                              }
```

JS OBJECTS

- JS object can be created by different ways:
  - Using new keyword

```
var objName = new Object( );
    objName.p1  =  val1 ;
    objName.p2  =  val2 ;
    objName.m1 =  function( ) {   } ;
```

JS OBJECTS

- JS object can created by different ways:
  - Using object constructor
    - Similar type of multiple objects can be created by object constructor function.
    - Constructor will be called with the new keyword.

JS OBJECTS

- JS object can created by different ways:
  - Using object constructor

```
function ConstName(Param1, Param2, …ParamN)
                {
                        this.p1  = Param1 ;
                        this.p2  = Param2 ;
                        this.m1  = function( ) {    } ;
                                }
```

```
var obj1 = new ConstName(val1,val2);
var obj2 = new ConstName(val11,val22);
```

JS BUILT-IN OBJECTS

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Error	Returns run-time error information
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

JS ARRAY

- Array can store more than one values by a single name.
- Values can be accessed by referring to an index.
- An array index starts with 0.
- An array can be created by different ways:
  - Using array literal:

```
var array_name = [ item1, item2, ... ];
```

  - Using Array() constructor:

```
var array_name = new Array(item1, item2, ... );
```

JS ARRAY

- Arrays are accessed by index numbers.

```
array_name[ index ];
```

- Array value can be changed by array index.

```
array_name[ index ] = newValue;
```

- Arrays in JS are objects.
- Array have properties and methods:
  - length : returns the length (number of elements)
  - push() : add new element in array.

JS FOR IN LOOP

- for/in statement are used to loop through the properties of an JS object.

- Syntax:

```
for ( key in object )
        {
          // code block to be repeated
        }
```

JS FOR OF LOOP

- for/of statement are used to loop through the values of an iterable object (array).

- Syntax:

```
for ( variable of iterableObj )
        {
          // code block to be repeated
        }
```

JAVASCRIPT (JS) EVENTS

JS EVENT HANDLING

- Events allow scripts to respond to a user interactions and modify webpage accordingly.
- An HTML event can be something the browser does, or something a user does.
- HTML allows event handler attributes that takes JavaScript code as a value.

```
<element event='JavaScript Code'>
```

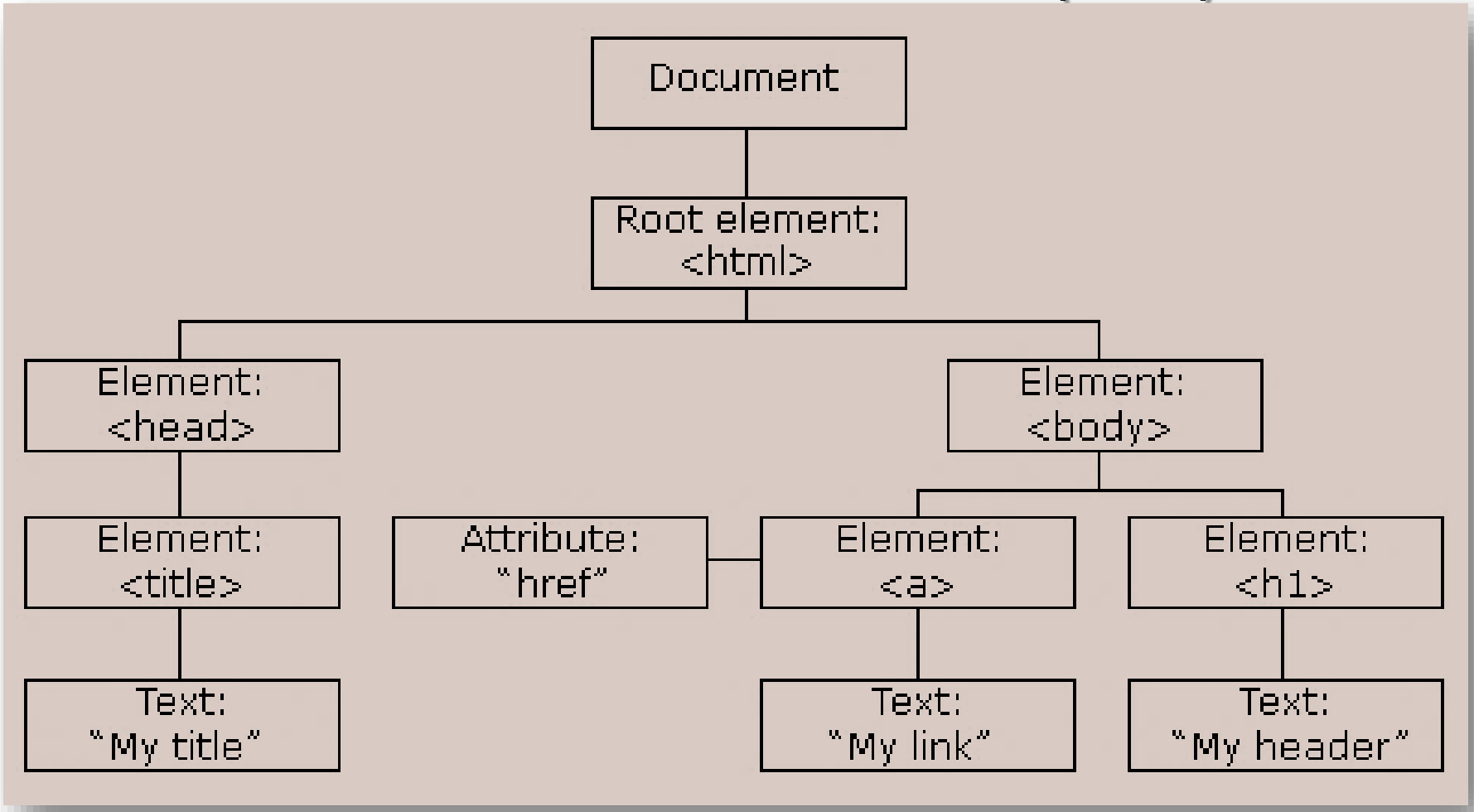
JS EVENT HANDLING

Event	Description
onclick	User clicks an HTML element
onmouseover	User moves the mouse over an HTML element
onmouseout	User moves the mouse away from an HTML element
onload	Browser has finished loading the page
onfocus	When an input field gets focus
onsubmit	When a form is submitted

DOCUMENT OBJECT MODEL (DOM)

- Inside the browser, the whole web page—paragraphs, forms, tables, etc.—is represented in an object hierarchy.
- Every piece of an HTML page (elements, attributes, text, etc.) is modeled in the web browser by a DOM node.
- All the nodes in a document make up the page's DOM tree.
- Nodes are related to each other through child-parent relationships.
- An HTML element inside another element is said to be its child—the containing element is known as the parent.

DOCUMENT OBJECT MODEL (DOM)



REGULAR EXPRESSION IN JS

REGULAR EXPRESSION

- Regular expression is an object that used to define pattern of characters.
- It is used to perform pattern-matching in JavaScript.
- The JavaScript **RegExp** class represents regular expressions.
- Syntax

```
var patt = new RegExp(pattern, attributes)
```

```
var patt = /pattern/attributes
```

REGULAR EXPRESSION

- Attributes\Modifiers
  - Modifiers are used to perform case-insensitive and global searches.
    - **g** : find out all matches
    - **i** : perform case-insensitive matching
    - **m** : perform multiline matching

REGULAR EXPRESSION

- Pattern
  - A string that specifies the pattern of the regular expression.
  - A regular expression pattern is composed of simple characters or combination of simple and special characters.
  - Simple pattern string are used to find out exact match of a string.
  - Special characters are used to define variable pattern matching.

REGULAR EXPRESSION

- Categories of Special Characters
  - **Assertions**
    - ✓ Assertions defines boundaries, that indicate the beginnings and endings of lines and words.
      - **^** : Matches the beginning of input
      - **$** : Matches the end of input

REGULAR EXPRESSION

Character classes

- Character classes distinguish between different kinds of characters.
  - **.** : Matches any single character
  - **\d** : Matches any digit
  - **\D** : Matches any character that is not a digit
  - **\w** : Matches any alphanumeric character including underscore
  - **\W** : Matches any character that is not a word character

REGULAR EXPRESSION

- **Character classes**
  - ✓ Character classes distinguish between different kinds of characters.
    - **\s** : Matches a single white space character
    - **\S** : Matches a single character other than white space
    - **** : Used for escape character

REGULAR EXPRESSION

Groups and ranges

- Groups and ranges indicate groups and ranges of expression characters.
  - **x|y** : Matches either "x" or "y".
  - **[xyz]** : Matches any one of the enclosed character.
  - **[a-z]** : Matches a character from a to z
  - **[A-Z]** : Matches a character from A to Z
  - **[0-9]** : Matches a character from 0 to 9

REGULAR EXPRESSION

Quantifiers

- Quantifiers indicate numbers of characters or expressions to match.
  - **x*** : Matches "x" 0 or more times.
  - **x+** : Matches "x" 1 or more times.
  - **x{n}** : Matches "x" exactly n times.
  - **x{n,}** : Matches "x" at least n times.
  - **x{n, m}** : Matches "x" at least n times and atmost m times.

REGULAR EXPRESSION

- **Example**
  - ^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$

    It defines pattern for email.
  - ^.*[a-zA-Z]$

    It defines a string ends with an alphabet.
  - ^[1-9][0-9]{9}$

    It defines a pattern 10 digit number

REGULAR EXPRESSION

- RegExp Methods
  - It also have some methods for pattern matching.
    - test(‘string’)
      - If it finds a match, it returns true; otherwise, it retu
    - exec(‘string’)
      - If it finds a match, it returns array of results; otherwise, it returns null.