FRONT END DEVELOPMENT

accenture

MODULE 3: JAVASCRIPT FOR FRONT END DEVELOPMENT

Learning and Knowledge Management

LEARNING OBJECTIVES

At the end of this module, you should be able to:

- Explain the Fundamentals of JavaScript
- Explain Conditional Statements and Loops
- Explain JavaScript Functions
- Explain JavaScript Exception Handling
- Explain working with BOM and DOM
- Explain HTML/DOM events to handle Form Validations

Insights into Fundamentals of Javascript

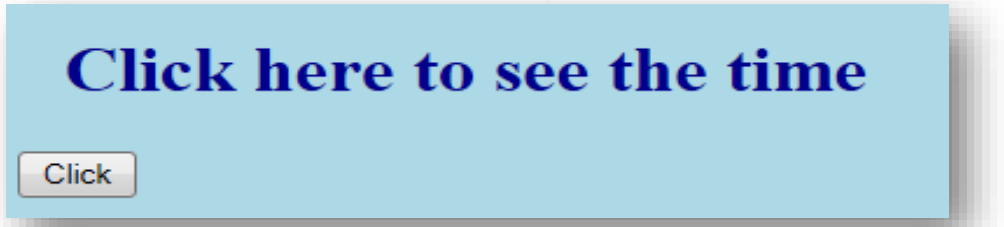
FINDING THE BEHAVIOR

- HTML specifies the content of the web page. It defines overall structure of the web page
- Adding CSS to the HTML pages allows to specify the layout of the page
- But there is still something missing! Can you notice what?

```
<html>
<head>
<meta
<title
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
    <h1>Click here to see the time</h1>
    <button type="button">Click</button>

    <p i
</body>
</html>
```

```
l";

r: lightblue;
}

h1 {
    color: darkblue;
    margin-left: 20px;
}
```

Correct! The web page does not respond! Nothing happens on click of the button



FINDING THE BEHAVIOR(CONTD.)

Response on a web page can be sent from:

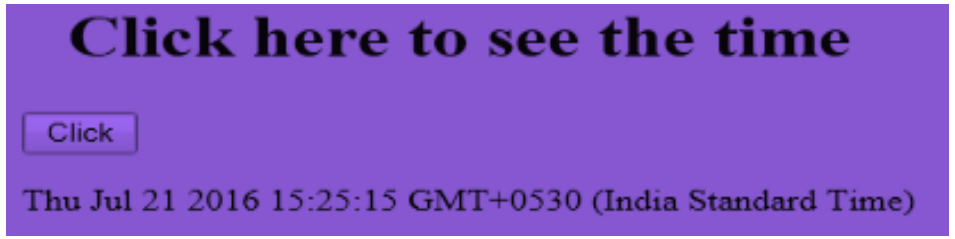
- Server side
- Client side

Scripting language like JavaScript is used to create response on the client side

FINDING THE BEHAVIOR(CONTD.)

```
function getTime(){
    document.getElementById('time').innerHTML = Date();
}
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>My Page</title>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<script src="my.js" type="text/javascript">
</script>
</head>
<body>
    <h1>Click here to see the time</h1>
    <button type="button" onclick="getTime()">Click</button>

    <p id="time"></p>
</body>
</html>
```



LEARNING TO ADD BEHAVIOR FOR AN ACTION USING JAVASCRIPT.

INTRODUCTION TO JAVASCRIPT

JavaScript is a loosely-typed client scripting language that executes in the user's browser.

It is the default scripting language for HTML.

Used in web pages for the following reasons:

- Client-side validations
- To improve design and add functionality to web pages
- Interact with HTML elements(DOM) to make interactive user interface.

Scripts are executed when,

- Web page loads into the browser (immediate scripts)
- An event is triggered (deferred scripts)

JavaScript follows and implements ECMAScript's specification.

INTRODUCTION TO JAVASCRIPT

Features of JavaScript

- Programming tool for creating interactive HTML pages.
- Interpreted Scripting Language
- Dynamically typed language
- Event driven programming
- Platform independent

- Note:
  - JavaScript can also used for non web-based platforms like PDF, desktop widgets, Game development etc.
  - JavaScript is traditionally used for client-side validations but can also be used for Server-side web applications

INTRODUCTION TO JAVASCRIPT

Class Discussion

Choose the odd one out:

1. **JavaScript is a client-side scripting language.**
2. **JavaScript is adding interactivity to HTML Pages**
3. **JavaScript is to style HTML pages.**
4. **JavaScript is an interpreted Scripting Language.**

INTRODUCTION TO JAVASCRIPT

JavaScript in HTML Pages

- JavaScript code is mentioned:
  - Within the <script> tag
    - In the <body> section
    - In the <head> section
  - In an external JavaScript file (*.js)

INTRODUCTION TO JAVASCRIPT

Including the Script in the <head>

- <head> section of the HTML is used to store meta information about the page
  - Charset, Keywords , link to resources etc.
  - It is not purposed to contribute to the HTML view

- The script mentioned in the <head> section is interpreted before loading the HTML view
- Generally the library scripts (functions) are mentioned in the <head> section

INTRODUCTION TO JAVASCRIPT

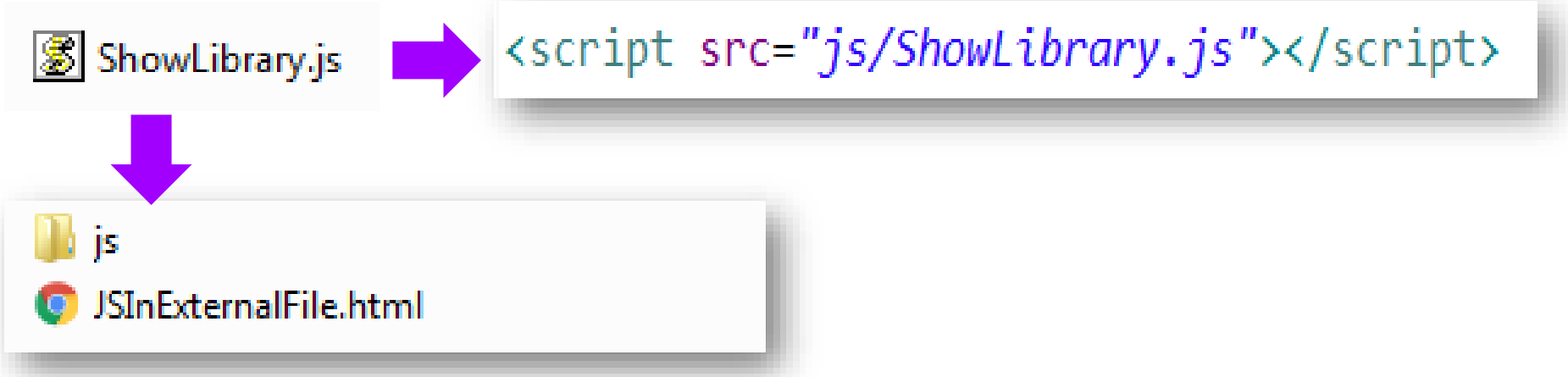
Including the Script in the <head>

- The content of the <body> section generates the view.
- Script mentioned in the <body> section participates in generating the view
- It is good practice to keep the script at the bottom of the <body>
    - Improves page load and renders the page faster without losing any time for script interpretation

INTRODUCTION TO JAVASCRIPT

Using External JavaScript File

- The library scripts or functions can be separated in an external document
- The external file should have .js extension



- Allows code reusability
  - Same library can be referenced from multiple HTML files
  - Manageable and maintainable

INTRODUCTION TO JAVASCRIPT

Where to put JavaScript?

There is no restriction regarding where to put the JavaScript, it depends on the requirement and scope

- The script can be kept ‘anywhere for any number of times’

```
<html>
    <head>
        <script>
            document.write('This is from head section');
        </script>
    </head>
<body>
        <h2>JavaScript at multiple location</h2>
        <script>
            document.write('This is from body section');
        </script>
</body>
</html>
```

LEARNING TO EMBED JAVASCRIPT IN A DOCUMENT.

HEAD SECTION
BODY SECTION
EXTERNAL JS

JAVASCRIPT BASIC

JavaScript Comments

- The JavaScript comments are additional lines of information
  - Ignored by the JavaScript interpreter
- Comments can be of two types:
  - Single line comments

```
// This line is a single line comment
document.write('This line is interpreted by the JS Engine');
```

  - Multi line comments

```
/*
This is a multi line comment
We can add our comments in multiple lines
*/
document.write('This line is interpreted by the JS Engine');
```

- Advantage
  - It makes the code readable and understandable
  - It avoids execution of a portion of code

JAVASCRIPT BASIC

JavaScript Variables

- Variables are containers to store data.
- It represents a storage location that holds the data referenced.
- It can be referenced by an identifier to use or manipulate its value.

JAVASCRIPT BASIC

Variable Naming Convention

- The variable identifier should maintain the following:
  - The size should not exceed 255 characters
  - Should start with a character (a to z or A to Z), underscore (_) or dollar symbol ($)
  - After first letter the following letters can be alphanumeric
  - Reserved words (for e.g. JavaScript keywords) cannot be used
  - case sensitive in nature

- It is always advised to keep the variable names meaningful and readable

JAVASCRIPT BASIC

Variable Declaration

- There are two ways of declaring a variable
  - Implicit declaration

```
companyName='Accenture';
```

  - Explicit declaration

```
var companyName='Accenture';
```

- If no value is assigned, then the default value will be set as undefined

But the type of the variable is not mentioned anywhere

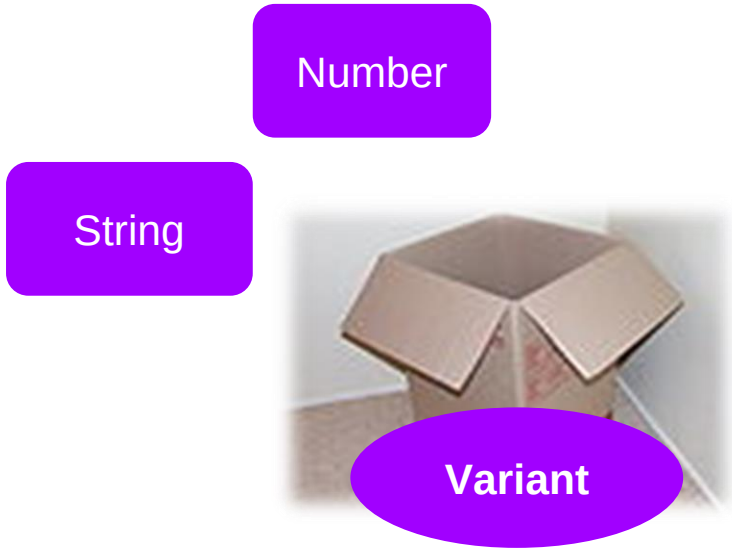
JAVASCRIPT BASIC

JavaScript Data types

JavaScript is a dynamic type language

Supported data types can be categorized into:

- Primitive data types
  - String
  - Number
  - Boolean
  - Undefined
  - Null
- Non primitive data types
  - Array
  - Object
  - Date



In JavaScript the type of variable can be verified using typeof().

JAVASCRIPT BASIC

JavaScript Primitive Data types

String

```
var str="Hello World";
```

Number

```
var int = 100;          var float = 100.5;
var hex = 0xfff;        var exponential = 2.56e3;
var octal = 030;
```

Boolean

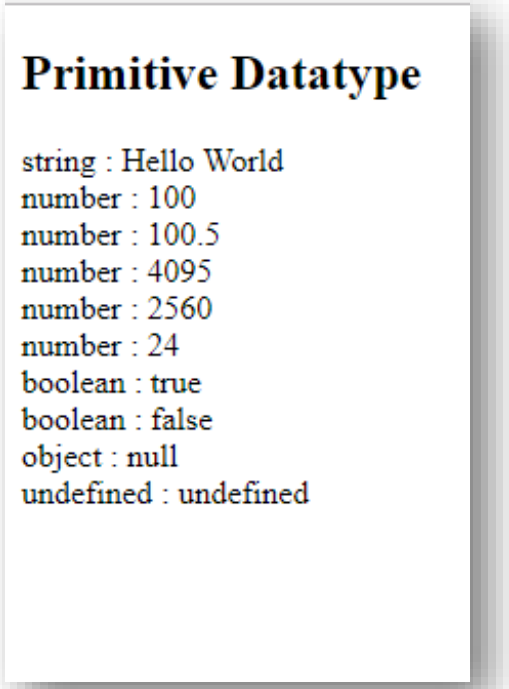
```
var  agree= true;
var  disagree = false;
```

null

```
var myVar = null;
```

undefined

```
var myVar;
alert(myVar); // undefined
```



LEARNING VARIABLE AND DATATYPE (PRIMITIVE)

JAVASCRIPT BASIC

PREDICT THE OUTPUT

Can you predict the output?

```
var aVal=10+1+'JavaScript';
document.write(aVal);
```

11JavaScript

```
var aVal='JavaScript'+10+1;
document.write(aVal);
```

JavaScript101

JAVASCRIPT BASIC

JavaScript Non-Primitive Data types

Array:

```
var <array-name> = [element0, element1, element2,... elementN];
```

Example:

```
var strArr = ["one", "two", "three"];
var numArr = [1, 2, 3, 4];
var mixedArray = [1, "two", "three", 4];
```

Array Constructor:

var arrayName = new Array();	var strArr = new Array(); strArr[0] = "one"; strArr[1] = "two";

var arrayName = new Array(Number length);	var numArr = new Array(3);
	numArr[0] = 1;
	numArr[1] = 2;

var arrayName = new Array(element1, element2, element3,... elementN);	var mixedArray = new Array(1, "two", 3, "four");

JAVASCRIPT BASIC

JavaScript Non-Primitive Data types (Date)

```
var currentDate = new Date(); // current Date
```

```
var dt = new Date('date string');

var date1 = new Date("2015 3 February");

var date2 = new Date("3 2015 February ");
```

JAVASCRIPT BASIC

JavaScript Non-Primitive Data types

Object is like other variables in JavaScript, only difference is it can hold multiple values unlike variable.

There are two ways to create an object :

- Object Literal

```
var <object-name> = { key1: value1, key2: value2,... keyN: valueN};
Example :
var product={prodName:”Eraser”,prodPrice:15.56,prodType:”stationary”};
console.log(product.prodName); //OR product[“prodName”];
```

- Object Constructor

```
var <object-name> = new Object();
Example :
var product=new Object();
product.prodName=“Eraser”;
prod.prodPrice=15.56;
console.log(product.prodName);
console.log(product.prodPrice);
```

JAVASCRIPT BASIC

Class **Discussion**

Choose the keyword to declare all types of variable in JavaScript:

1. variable
2. obj
3. jvar
4. var

JAVASCRIPT BASIC

Class Discussion

State true or false:

1. Variable can hold only one value at a time.
2. Object can hold multiple values.

JAVASCRIPT BASIC

Class Discussion

Choose output for below code snippet:

```
var str1=“123”;
var str2=123;
console.log(str1+str2);
```

1. 246
2. 123
3. 123123
4. Error

JAVASCRIPT BASIC

Operator	Description
Arithmetic Operators	+ , - , * , /
Relational Operators	>=,<=,>,<, == ,=== etc.
Logical Operators	AND (&&), OR(||), NOT(!)
Assignment Operators	= , +=, *=, /= etc.
Bitwise Operators	&, | , ^ , ~, << ,>> ,>>>
Special Operators	delete, new, instanceof , typeof etc.

JAVASCRIPT BASIC

JavaScript Conditional Statements

- In JavaScript we have following constructs to apply decision control logic
  - If statement
  - If…else statement
  - If…else…if statement
  - Switch statement

JAVASCRIPT BASIC

If..Else Statements

If statements are simplest method of decision control structure

It verifies only one Boolean condition

```
if(count==5){
    document.write('Five');
}
```

Example: If the value of a variable ‘count’ is equal to 5. Print ‘Five ’

The else statement is applicable where the if condition is false

We can multiply or nest the if…else condition if required

```
if(count==5){
    document.write('Five');
}else{
    document.write('Not Five');
}
```

Example: If the value of a variable ‘count’ is equal to 5. Print ‘Five’; Else print ‘Not Five’

JAVASCRIPT BASIC

Switch Statement

- Switch statement is used when there are too many conditions based on the value of a variable
- Example: the value of the variable ‘result’ depends on the value of variable ‘location’

Key	Value	result
location	BDC1	Cunningham road
	BDC6	Whitefield
	<No match found>	External Location

- Switch case block is fall-through
  - All the cases will be passed if break is not used

```
var office = 'BDC1';
var result;
switch (office) {
case 'BDC1':
    result = 'Cunningham road';
    break;
case 'BDC6':
    result = 'Whitefield';
    break;
default:
    result = 'External office';
    break;
}
document.write(result);
```

JAVASCRIPT BASIC

Iterative Statements

Iteration is a programming construct to execute instructions repeatedly.

For loop

- Iterates for fixed number of times
- Used when the number of iteration is known

While loop

- Iterates until the loop condition fails
- Used when number of iteration is not known
- While loop has two types
  - Simple While loop: Entry Control Loop
  - do-while loop: Exit Control Loop

It is important to use a feasible terminating condition to prevent infinite loops.

```
for (var i = 0; i < 5; i++)
{
    console.log(i);
}
```

```
var i =0;
while(i < 5)
{
    console.log(i);
    i++;
```

```
var i = 0;
do{
    alert(i);
    i++;
} while(i < 5)
```

LEARNING CONDITIONAL STATEMENTS AND LOOPS

JAVASCRIPT BASIC

Choose the correct output for below code snippet:

```
var str1=“123”;
var str2=123;
if(str1===str2)
{
        console.log(“Match found”);
}
else
{
        console.log(“Match not found”)
}
```

1.Match Found

2.Match not Found

3.Exception

Insights into JavaScript Functions

JAVASCRIPT FUNCTIONS

- Function is a block of code, which performs a specific task
- Functions are often treated as black boxes, which encapsulates any repetitive operation and executes only when invoked

```
function functionName(argument1,argument2,...,argumentN){
    // Code to perform some action
    return retValue; // optional
}
```

- Advantage
  - Increases code reusability by reducing repetitive lines
  - Enhances the readability and maintainability of the code

JAVASCRIPT FUNCTION

JavaScript Function Expression

- JavaScript allows a function to be assigned to a variable, and later use that variable as function, this concept is called Function Expression.

```
var <object-name> = function <function-name(parameter list){
        //function body;
};
```

- **Example:**

```
var greeting=function showMsg(msg){
        return “Welcome”+msg;
};

var result=greeting(“Rose”);
console.log(result);
```

JAVASCRIPT FUNCTION

JavaScript Anonymous Function

JavaScript allows a function without any name. This unnamed function is called anonymous function.

Anonymous function must be assigned to a variable.

```
var <object-name> = function(parameter list){
        //function body;
};
```

Example:

```
var greeting=function(msg){
        return “Welcome”+msg;
};

var result=greeting(“Rose”);
console.log(result);
```

LEARNING FUNCTIONS, FUNCTION EXPRESSION AND ANONYMOUS FUNCTIONS

JAVASCRIPT FUNCTIONS

Local and Global Variables

Based on the scope, variables can be categorized in two types:

- Local variables
  - Variable is explicitly declared within a block or function
  - The scope of the variable is limited within the block or function

- Global variables
  - The variable is declared
    - Outside the function
    - Associated with the window object
    - Declared implicitly
  - Accessible throughout the code i.e. any block or function

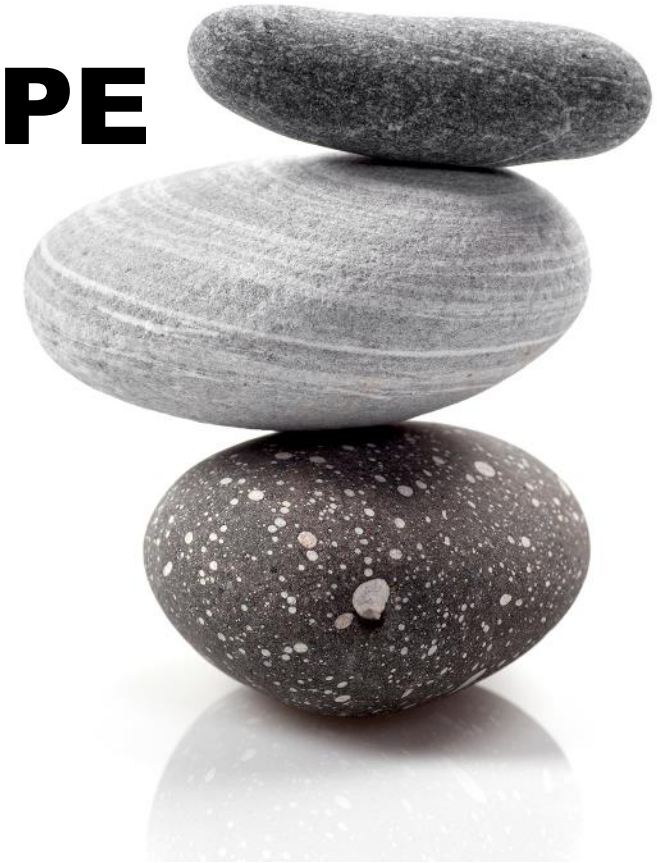
JAVASCRIPT FUNCTIONS

Based on the scope, variables can be categorized in two types:

- Local variables
  - Variable is explicitly declared within a block or function
  - The scope of the variable is limited within the block or function

- Global variables
  - The variable is declared
    - Outside the function
    - Associated with the window object
    - Declared implicitly
  - Accessible throughout the code i.e. any block or function

LEARNING VARIABLE SCOPE (LOCAL AND GLOBAL)



JAVASCRIPT FUNCTIONS

Class Discussion

Consider below code snippet , choose the most appropriate output:

```
<script>
                        var gnum=10;

                        function increment()
                        {
                                counter=10;
                                gnum=gnum+counter;
                        }

                        function getValues()
                        {
                                increment();
                                console.log("gnum:"+gnum);

        console.log("counter:"+counter);

                        }

        </script>
<body>
        <input type="button" value="Get Values"
onclick="getValues()"/>
</body>
```

1. gnum:20
   undefined
2. gnum:20
   null
3. gnum:20
   counter:10
4. Exception

Insights into Exception Handling

EXCEPTION HANDLING

Exception Handling in JavaScript

JavaScript provides Exception Handling mechanism to handle runtime errors using try..catch..finally block.

try-catch statement

- Marks a block of statements to try, and specifies one or more responses should an exception be thrown.
- If an exception is thrown, the catch statement catches it.

```
try
{
    // code that may throw an error
}
catch(ex)
{
    // code to be executed if an error occurs
}
finally{
    // code to be executed regardless of an error occurs or not
}
```

EXCEPTION HANDLING

Try-catch statement

```
 function getMonthName(mo)
{
mo=mo-1;
var months=new
Array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","D
ec");
            if (months[mo] != null) {
                        return months[mo];
            }
            else {
                        throw "InvalidMonthNo"; //throw keyword is
used here
            }
}
try {// statements to try
            monthName=getMonthName(-1) // function could throw
exception
            console.log("Month:"+monthName);
}
catch (e) {
            monthName="unknown";
            document.getElementById("errorMessage").value=e;
}
```

EXCEPTION HANDLING

Finally block :

- Contains statements to execute after the try and catch blocks execute but before the statements following the try...catch statement.
- The finally block executes whether or not an exception is thrown.

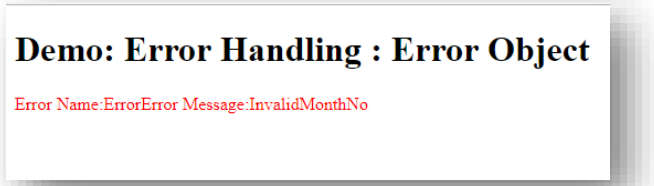
```
openMyFile();
try {

writeMyFile(theData); //This may throw a error
}
catch(e){
handleError(e); // If we got a error we handle it
}
finally {
        closeMyFile(); // always close the
resource
 }
```

EXCEPTION HANDLING

Utilizing error object

```
function getMonthName(mo) {
mo=mo-1;
var months=new
Array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct"
,"Nov","Dec");
        if (months[mo] != null) {
                return months[mo];
        }
        else {
                throw(new Error("InvalidMonthNo"));
        }
}
try {// statements to try
        monthName=getMonthName(-1) // function could
throw exception
        console.log("Month:"+monthName);
}
catch (e) {
document.getElementById("errorMessage").value="Error
```

New error object creation



Learning exception handling

Exception Handling with try..catch
Exception Handling with new Error object

EXCEPTION HANDLING

Class Discussion

Choose the most appropriate output:

```
<script>
        try
        {
                throw(new Error("Its an
error"));

        }
        catch(e)
        {
                console.log("Error handled");
        }
        finally
        {
                console.log("Resource
released");
        }
</script>
```

1.Error Handled

2.Resource released

3.Error Handled

  Resource released

4.Undefined

Insights into BOM

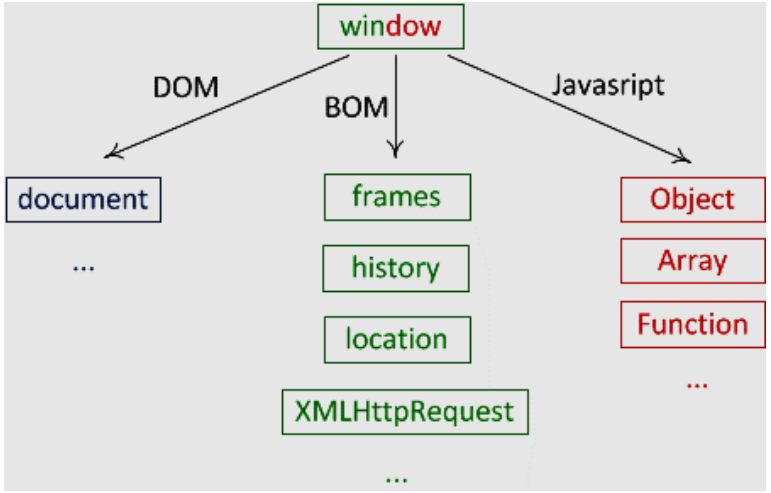
BROWSER OBJECT MODEL

Browser Object Model

Browser Object Model allows the JavaScript to respond to an event by means of the browser controls.

Browser is represented by an implicit object, named window.

window object provides the utilities (functions and the references) to generate response from the browser side.



BROWSER OBJECT MODEL

Window Object

- Represent a open window in the browser

window Object properties:

Property	Description
innerHeight	the inner height of the browser window
innerWidth	the inner width of the browser window
applicationCache	provides access to the offline resources for the window
history	Returns reference to history object
screen	a reference to the screen object associated with the window

BROWSER OBJECT MODEL

Window Object methods:

Method	Description
open()	open a new window
close()	close the current window
moveTo()	move the current window
resizeTo()	resize the current window
alert()	Displays an alert dialog
confirm()	To receive confirmation from the user
prompt()	To receive a value from user

LEARNING BOM METHODS

Insights into DOM

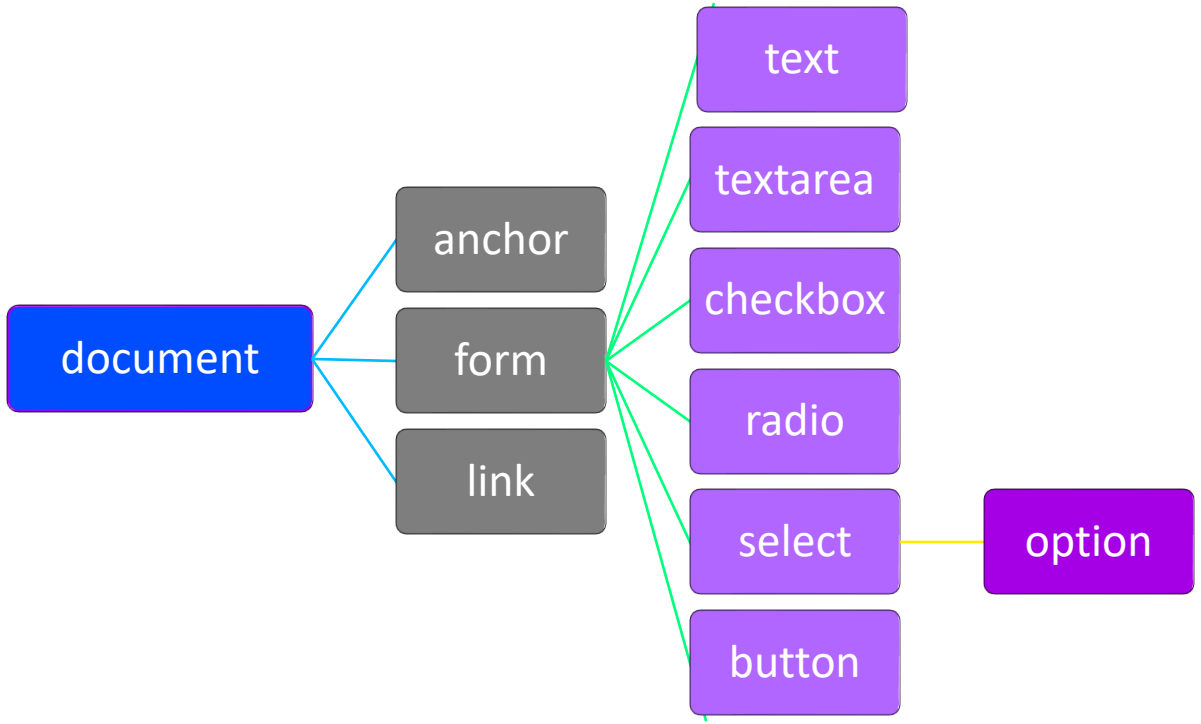
DOM

DOM Hierarchy

- Document is a container for all the tags in a HTML page
- HTML elements can be accessed and modified

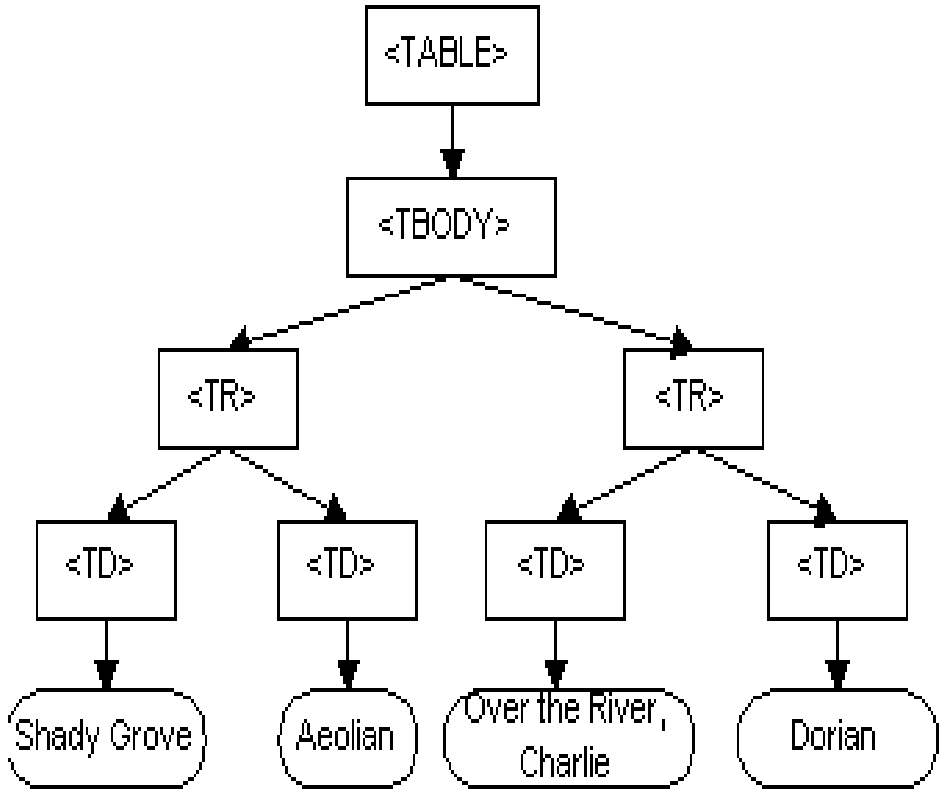
by following DOM as shown below:

window. document
or
document



DOM

```
<TABLE>
    <TBODY>
      <TR>
        <TD>Shady Grove</TD>
        <TD>Aeolian</TD>
      </TR>
      <TR>
        <TD>Over the
River,Charlie</TD>
        <TD>Dorian</TD>
      </TR>
    </TBODY>
</TABLE>
```



DOM PROPERTIES

DOM PROPERTIES

Following are some useful document object properties:

- innerHTML (used to write the dynamic html on the html document)
- innerText (used to write the dynamic text on the html document. This is normal text, and not HTML text)
- bgcolor (used to set background color for a page)
- fgcolor (used to set text color for a page)
- linkColor (used to specify the color of a hyperlink on a page)
- title (used to set the title of the page, which is usually done by the <title> tag)

DOM

DOM METHODS

- Following are few important methods of document object:

Method Name	Description
write(“String value”)	Writes the given string in the HTML document
getElementById(“Id value”)	Retrieves the element with given Id value
getElementByTagName(“name”)	retrieve an array of all elements having the given tag name
getElementsByName(“element name”)	Returns an array of all elements with the specified name.

Learning DOM methods

getElementById()
getElementsByName()
getElemenstByTagName()

Insights into HTML/DOM Events for Form Validation

HTML/DOM Events for Form Validation

HTML/DOM Events

Events are mode of interaction between the user and the web page

JavaScript is used to react to the events

Example of an event can be:

- An HTML Page is loaded
- A field value is changed
- An HTML button is clicked

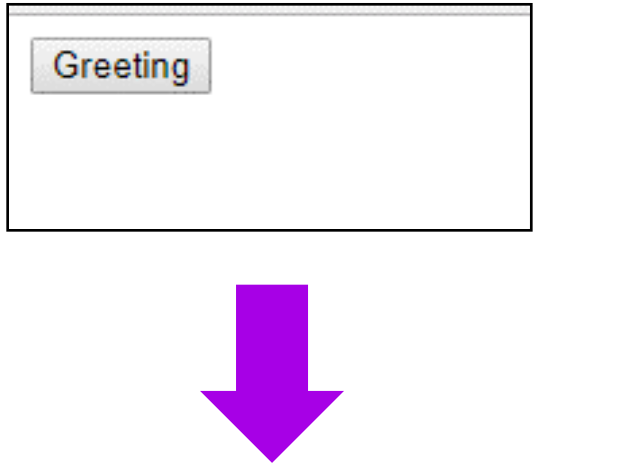
Specific HTML elements have event handler attributes for supported events

When the event is triggered corresponding script is executed

HTML/DOM Events for Form Validation

HTML/DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Learning DOM</title>
<script>
function greeting(){
document.write("Welcome to Learning DOM");
}
</script>
   </head>
  <body>
<button onclick="greeting()">Greeting</button>
  </body>
 </html>
```



HTML/DOM Events for Form Validation

HTML/DOM Events

Based user interaction, events can be of many types. For example, Mouse events, Keyboard events etc.

Following are list of few common HTML events:

Event	Event Type	Description
onblur	Form Event	This event is triggered only when the concerned element loses cursor focus
onfocus	Form Event	This event is triggered only when the concerned element gets cursor focus
onclick	Mouse Event	This event is triggered when a button or hyperlinked is clicked

Learning DOM Events

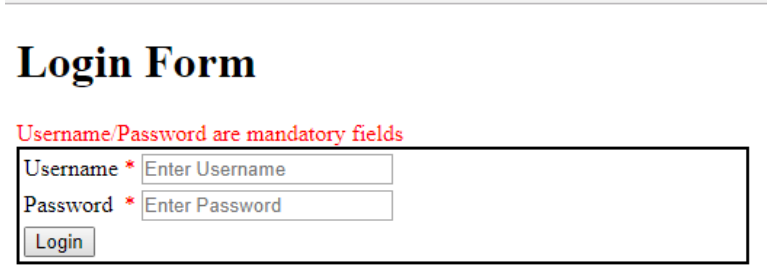
1. Blur
2. Focus
3. Click

HTML/DOM Events for Form Validation

Form Validation

JavaScript with DOM and Events provides a way to validate form data.

Types of Form Validation that can be performed using JavaScript are:



- Validating for mandatory fields
- Validating for correct value

JavaScript Form Validation is also called as Client-Side Validation enhancing user experience by

- Faster validation on the client machine
- Avoiding sending data to sever for validation

Learning Form Validation

HTML/DOM Events for Form Validation

JavaScript innerHTML

innerHTML property can be used to write the dynamic html on the html document.

```
<div id="mylocation"></div>
<input type=“button” onclick=“showLocation()”/>

<script>

function showLocation()
{
document.getElementById(‘<b>mylocation</b>').innerHTML=data;

}

</script>
```

MODULE SUMMARY

Now, you should be able to:

- Create script using JavaScript Syntax
- Create Functions
- Handle runtime errors
- Manipulate DOM
- Handle DOM events
- Form Validations

THANK YOU