

# Database Systems and Web

## **JAVA Script**

---

### Lecture 4

# JAVA Script

# What is JavaScript?

- Browsers have limited functionality
  - Text, images, tables, etc.
- JavaScript allows for interactivity
  - Perform calculations
  - Validation of input
- Browser/page manipulation
  - Reacting to user actions
- A type of programming language (Object Based and Interpreted)
  - Easy to learn
  - Developed by Netscape
  - Now works in all major browsers
  - is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more

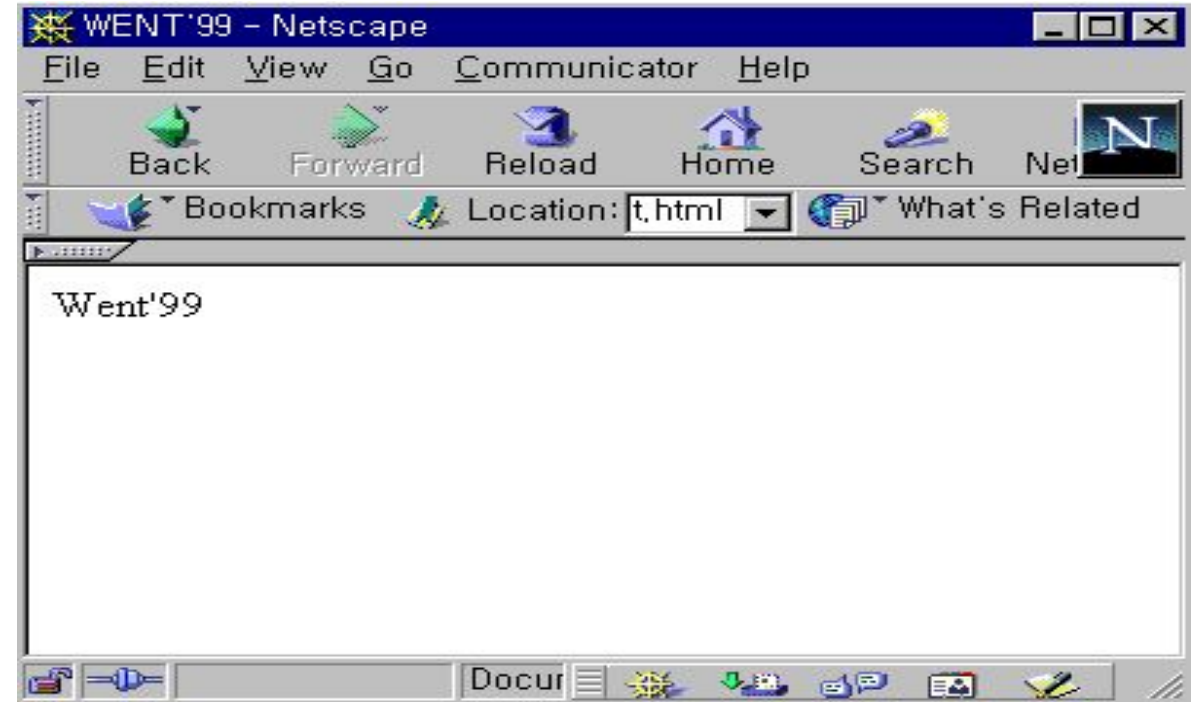
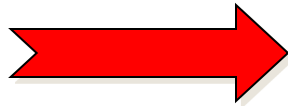
# How Does It Work?

- Embedded within HTML page
  - View source
- Executes on client
  - Fast, no connection needed once loaded
- Simple programming statements combined with HTML tags
- Interpreted (not compiled)
  - No special tools required
- Ending Statements With a Semicolon?
  - Semicolons are **optional**! Required if you want to put more than one statement on a single line.

# Basic HTML Document Format

```
<HTML>
<HEAD>
  <TITLE>WENT'99</TITLE>
</HEAD>
<BODY>
  Went'99
</BODY>
</HTML>
```

See what it  
looks like:



Javascript code can be added to a web page in three ways:

1. In the page's body (runs when page loads)
2. In the page's head (used for events)
3. In a link to an external .js script file

## Example

**In the page's body (runs when page loads)**

```
<html>
```

```
<body>
```

```
<script type="text/javascript">
```

```
    document.write ("Hello World!") ;
```

```
    document.write (“<h1>Hello World! </h1>”);
```

```
document.write (“<p>Hello World! </p>”);
```

```
</script>
```

```
</body>
```

```
</html>
```

# Data Types

- JavaScript allows the same variable to contain different types of data values.
- **Primitive data types**
  - **Number**: integer & floating-point numbers
  - **Boolean**: logical values “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**: an initial value is assigned
  - **Undefined**: the variable has been created but not yet assigned a value



# Numeric Data Types

- It is an important part of any programming language for doing arithmetic calculations.
- JavaScript supports:
  - **Integers:** A positive or negative number with no decimal places.
    - Ranged from  $-2^{53}$  to  $2^{53}$
  - **Floating-point numbers:** usually written in exponential notation.
    - 3.1415..., 2.0e11

- `var length = 16; // Number`  
`var lastName = "Johnson"; // String`  
`var cars = ["Saab", "Volvo", "BMW"]; // Array`  
`var x = {firstName:"John", lastName:"Doe"}; // Object`
- Javascript has dynamic types
  - `var x; // Now x is undefined`  
`var x = 5; // Now x is a Number`  
`var x = "John"; // Now x is a String`
- You can use the JavaScript **typeof** operator to find the type of a JavaScript variable:
- `typeof "John" // Returns string`  
`typeof 3.14 // Returns number`  
`typeof false // Returns boolean`  
`typeof [1,2,3,4] // Returns object`  
`typeof {name:'John', age:34} // Returns object`

# Array

- An Array contains a set of data represented by a single variable name.
- Arrays in JavaScript are represented by the Array Object, we need to “new Array()” to construct this object.
- The first element of the array is “Array[0]” until the last one Array[i-1].
- E.g. myArray = new Array(5)
  - We have myArray[0,1,2,3,4].

# Array Example

```
<script type="text/JavaScript">  
    Car = new Array(3);  
    Car[0] = "Ford";  
    Car[1] = "Toyota";  
    Car[2] = "Honda";  
    document.write(Car[0] + "<br>");  
    document.write(Car[1] + "<br>");  
    document.write(Car[2] + "<br>");  
</script>
```

- You can also declare arrays with variable length.
  - `arrayName = new Array();`
  - `Length = 0`, allows automatic extension of the length.
  - `Car[9] = "Ford"; Car[99] = "Honda";`

# JavaScript Operators

## Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

# JavaScript Operators – 2

## Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

# JavaScript Operators - 3

## Comparison Operator

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5"  x==y returns true  x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

# JavaScript Operators - 4

## Logical Operators

Operator	Description	Example
&&	and	x=6 y=3  (x < 10 && y > 1) returns true
	or	x=6 y=3  (x==5    y==5) returns false
!	not	x=6 y=3  !(x==y) returns true



## Example -2

```
<script>  
s1=12  
s2=28  
total=s1+s2  
document.write("Total is: ",total)  
</script>
```

# JavaScript Popup Boxes

- Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.

```
<script>
```

```
alert("Hello World!")
```

```
</script>
```



# JS Examples -1

$Y=20x+12$  and  $x=3$

```
<script>
```

```
x=3
```

```
y=20*x+12
```

```
alert(y)
```

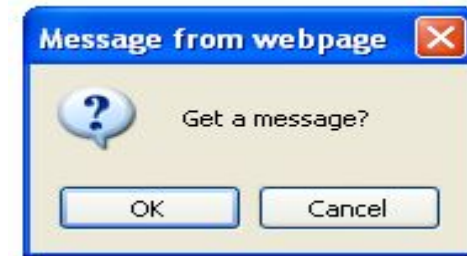
```
</script>
```

# JavaScript Popup Boxes - 2

- Confirm Box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

```
<script>  
confirm("Get a message?");  
</script>
```



## JavaScript Popup Boxes – 2 (Cont..)

- ```
var r = confirm("Press a button");  
if (r == true) {  
    x = "You pressed OK!";  
} else {  
    x = "You pressed Cancel!";  
}
```

# JavaScript Popup Boxes – 2

(Cont.)

- `if (confirm("Do you agree")) {`
- `alert("You agree")`
- `}`
- `else{`
- `alert ("You do not agree")`
- `};`

# JavaScript Popup Boxes - 3



- Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK", the box returns the input value. If the user clicks "Cancel", the box returns null.

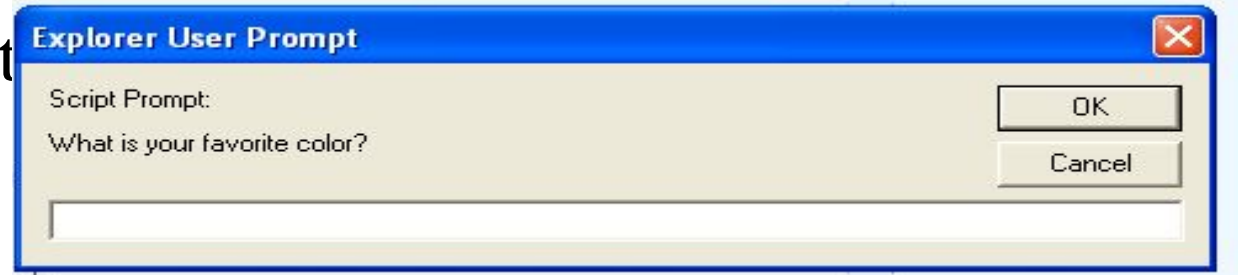
```
<script>
```

```
prompt("What is your favorite  
color?", "");
```

```
</script>
```

# JavaScript Popup Boxes - 3

- Prompt Box
  - A prompt box is often used if you want the user to input a value before entering a page.
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - If the user clicks "OK", the box returns the input value. If the user clicks "Cancel", the box returns null.



```
<script>  
prompt("What is your favorite  
color?", "");  
</script>
```



# Conditional and Repetition Statements

- **Conditional statements:**
- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed
- **Repetition structure:** four in JavaScript
  - while
  - do...while
  - for
  - for...in

# If ..else..if

```
<html>
<body>

<script type="text/javascript">
<!--
var book = "maths";
if( book == "history" ){
    document.write("<b>History Book</b>");
}else if( book == "maths" ){
    document.write("<b>Maths Book</b>");
}else if( book == "economics" ){
    document.write("<b>Economics Book</b>");
}else{
    document.write("<b>Unknown Book</b>");
}
//-->
</script>

<p>Set the variable to different value and then try...</p>
</body>
</html>
```



# While Loop and Do ...While

## Syntax

```
while (expression){  
    Statement(s) to be executed if expression is true  
}
```

## Syntax

```
do{  
    Statement(s) to be executed;  
} while (expression);
```

# Example

```
<html>
<body>

<script type="text/javascript">
<!--
var count = 0;
document.write("Starting Loop ");
while (count < 10){
    document.write("Current Count : " + count + "<br />");
    count++;
}
document.write("Loop stopped!");
//-->
</script>

<p>Set the variable to different value and then try...</p>
</body>
</html>
```

## Output

Starting Loop Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

Set the variable to different value and then try...

# For Loop

```
for (initialization; test condition; iteration statement){  
    Statement(s) to be executed if test condition is true  
}
```

# Example

```
<html>
  <body>
    <script type="text/javascript">
      for (var i = 0; i < 10; i++) {
        document.write("<p>" + i + " squared = " +
          (i * i) + "</p>");
      }
    </script>
  </body>
</html>
```



# Break Statement

```
<html>
<body>

<script type="text/javascript">
<!--
var x = 1;
document.write("Entering the loop<br /> ");
while (x < 20)
{
    if (x == 5){
        break; // breaks out of loop completely
    }
    x = x + 1;
    document.write( x + "<br />");
}
document.write("Exiting the loop!<br /> ");
//-->
</script>

<p>Set the variable to different value and then try...</p>
</body>
</html>
```

## Output

Entering the loop

2

3

4

5

Exiting the loop!

Set the variable to different value and then try...



# Continue Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var x = 1;
document.write("Entering the loop<br /> ");
while (x < 10)
{
    x = x + 1;
    if (x == 5){
        continue; // skip rest of the loop body
    }
    document.write( x + "<br />");
}
document.write("Exiting the loop!<br /> ");
//-->
</script>

<p>Set the variable to different value and then try...</p>
</body>
</html>
```

## Output

```
Entering the loop
2
3
4
6
7
8
9
10
Exiting the loop!
```

# Functions

- A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes.

# Function Syntax

```
<script type="text/javascript">  
<!--  
function functionname(parameter-list)  
{  
    statements  
}  
//-->  
</script>
```

# Example

## Output

Click the following button to call the function

Say Hello

Use different parameters inside the function and then try...

```
<html>
<head>
<script type="text/javascript">
function sayHello(name, age)
{
    document.write (name + " is " + age + " years old.");
}
</script>
</head>

<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
</form>

<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

# Return Statement Is Optional In Java Script

```
<html>
<head>
<script type="text/javascript">
function concatenate(first, last)
{
    var full;

    full = first + last;
    return full;
}
function secondFunction()
{
    var result;
    result = concatenate('Zara', 'Ali');
    document.write (result );
}
</script>
</head>
```

Continued on next slide

# Return Statement Is Optional In Java Script

## Output

Click the following button to call the function

Say Hello

Use different parameters inside the function and then try...

```
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>

<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

# Event

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- click , pressing any key, closing a window, resizing a window, etc.

# onmouseover

- The onmouseover event triggers when you bring your mouse over any element



# Example

- `<html>`
- `<head>`
- `<script type="text/javascript">`
- `<!--`
- `function over() {`
- `document.write ("Mouse Over");`
- `}`
- `//-->`
- `</script>`
- `</head>`
- `<body>`
- `<p>Bring your mouse inside the division to see the result:</p>`
- `<div onmouseover="over()" >`
- `<h2> This is inside the division </h2>`
- `</div>`
- `</body>`
- `</html>`

- `<html>`
- `<body>`
- `<p onclick="myFunction()">Double-click this paragraph to trigger a function.</p>`
- `<p id="demo"></p>`
- `<script>`
- `function myFunction() {`
- `document.getElementById("demo").innerHTML = "Hello World";`
- `}`
- `</script>`
- `</body>`
- `</html>`

# Other Events

onclick	script	Triggers on a mouse click
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element

# HTML Forms and JavaScript

- JavaScript is very good at processing user input in the web browser
- HTML `<form>` elements receive input
- Forms and form elements have unique names
  - Each unique element can be identified

# Naming Form Elements in HTML

Name:	<input type="text"/>
Phone:	<input type="text"/>
Email:	<input type="text"/>

```
<form name="addressform">  
Name:  <input name="yourname"><br />  
Phone: <input name="phone"><br />  
Email: <input name="email"><br />  
</form>
```

# Forms and JavaScript

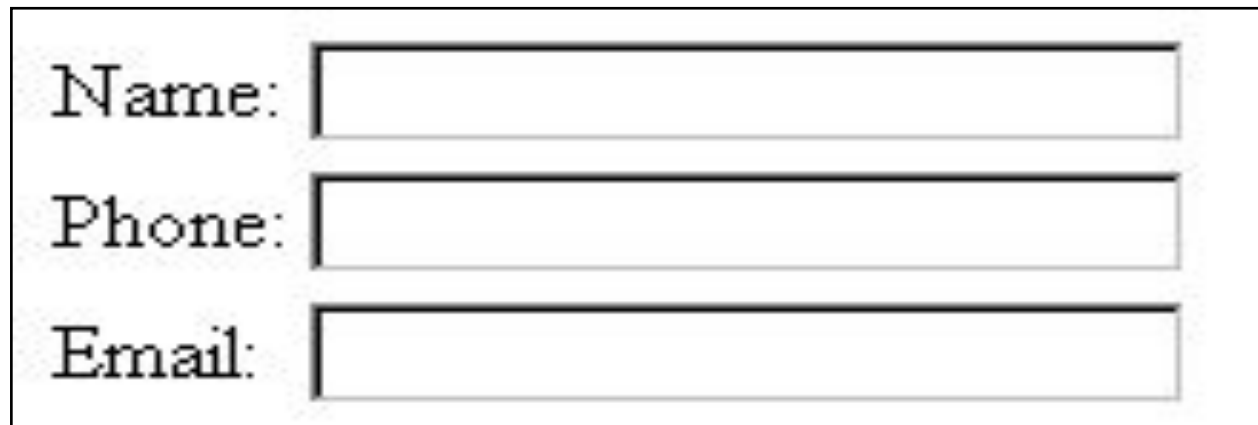
*document.**formname.elementname**.value*

Thus:

document.**addressform.yourname**.value

document.addressform.phone.value

document.addressform.email.value



A rectangular box containing three vertically stacked input fields. Each field is preceded by a label: 'Name:', 'Phone:', and 'Email:'. The input fields are empty text boxes.

Name:	<input type="text"/>
Phone:	<input type="text"/>
Email:	<input type="text"/>

# Using Form Data

## Personalising an alert box

Enter your name:



```
<form name="alertform">
```

Enter your name:

```
<input type="text" name="yourname">
```

```
<input type="button" value= "Go"  
  onClick="window.alert('Hello ' + →  
    document.alertform.yourname.value) ; ">
```

```
</form>
```

# Form Validation

- JavaScript provides a way to validate form's data on the **client's computer** before sending it to the web server



# Form Validation

- Basic Validation :
  - Make sure all the mandatory fields are filled in
- Data Format Validation
  - The data that is entered must be checked for correct form and value

# Form Validation

Output

Name	<input type="text"/>
EMail	<input type="text"/>
	<input type="submit" value="Submit"/>

Use this complete path for testing the submitted data  
ACTION="http://www.rebol.com/cgi-bin/test-cgi.cgi"

A

```
<td align="right"></td>
<td><input type="submit" value="Submit" /></td>
</tr>
</table>
</form>
</body>
</html>
```

```
<html>
<head>
<title>Form Validation</title>
<script type="text/javascript">
<!--
// Form validation code will come here.
//-->
</script>
</head>
<body>
  <form action="/cgi-bin/test.cgi" name="myForm"
        onsubmit="return(validate());">
    <table cellspacing="2" cellpadding="2" border="1">
      <tr>
        <td align="right">Name</td>
        <td><input type="text" name="Name" /></td>
      </tr>
      <tr>
        <td align="right">EMail</td>
        <td><input type="text" name="EMail" /></td>
      </tr>
```

Continues At A

# Form Validation

```
<script type="text/javascript">
<!--
// Form validation code will come here.
function validate()
{

    if( document.myForm.Name.value == "" )
    {
        alert( "Please provide your name!" );
        document.myForm.Name.focus() ;
        return false;
    }
    if( document.myForm.EMail.value == "" )
    {
        alert( "Please provide your Email!" );
        document.myForm.EMail.focus() ;
        return false;
    }
    return( true );
}
//-->
</script>
```

# Data Format Validation

```
<script type="text/javascript">
<!--
function validateEmail()
{
    var emailID = document.myForm.EMail.value;
    atpos = emailID.indexOf("@");
    dotpos = emailID.lastIndexOf(".");
    if (atpos < 1 || ( dotpos - atpos < 2 ))
    {
        alert("Please enter correct email ID")
        document.myForm.EMail.focus() ;
        return false;
    }
    return( true );
}
//-->
</script>
```

# Get and Post Method of Form

**GET is used to request data from a specified resource.**

**GET is one of the most common HTTP methods.**

**Note that the query string (name/value pairs) is sent in the URL of a GET request:**

`/test/demo_form.php?name1=value1&name2=value2`

**Some other notes on GET requests:**

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests is only used to request data (not modify)

# Get and Post Method of Form

**POST is used to send data to a server to create/update a resource.**

The data sent to the server with POST is stored in the request body of the HTTP request:

POST /test/demo\_form.php HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

**POST is one of the most common HTTP methods.**

**Some other notes on POST requests:**

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

# Get and Post Method of Form

Both GET and POST method is used to transfer data from client to server in HTTP protocol but main difference between POST and GET method is that GET carries request parameter appended in URL string while POST carries request parameter in message body which makes it more secure way of transferring data from client to server.

# Practice Example 1

- Write a program to change text



# Practice Example 1- Solution

- `<html>`
- `<head>`
- `<script type="text/javascript">`
- `function change_header()`
- `{`
- `document.getElementById("myHeader").innerHTML="Nice day!";`
- `}`
- `</script>`
- `</head>`
- `<body>`
- `<h1 id="myHeader">Hello World!</h1>`
- `<button onclick="change_header()">Change text</button>`
- `</body>`
- `</html>`

## Practice Example 2

- Create a form with input type number. Validate the form that number entered is between 1 to 10

# Practice Example 2- Solution

```
<html>
<body>
<h1>JavaScript Can Validate Input</h1>
<p>Please input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>
```

```
function myFunction() {
    var x, text;
    // Get the value of the input field with id="numb"
    x = document.getElementById("numb").value;
    // If x is Not a Number or less than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}</script>
</body>
</html>
```