

# Database Systems and Web

## JAVA Script

---

### Lecture 5

# 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

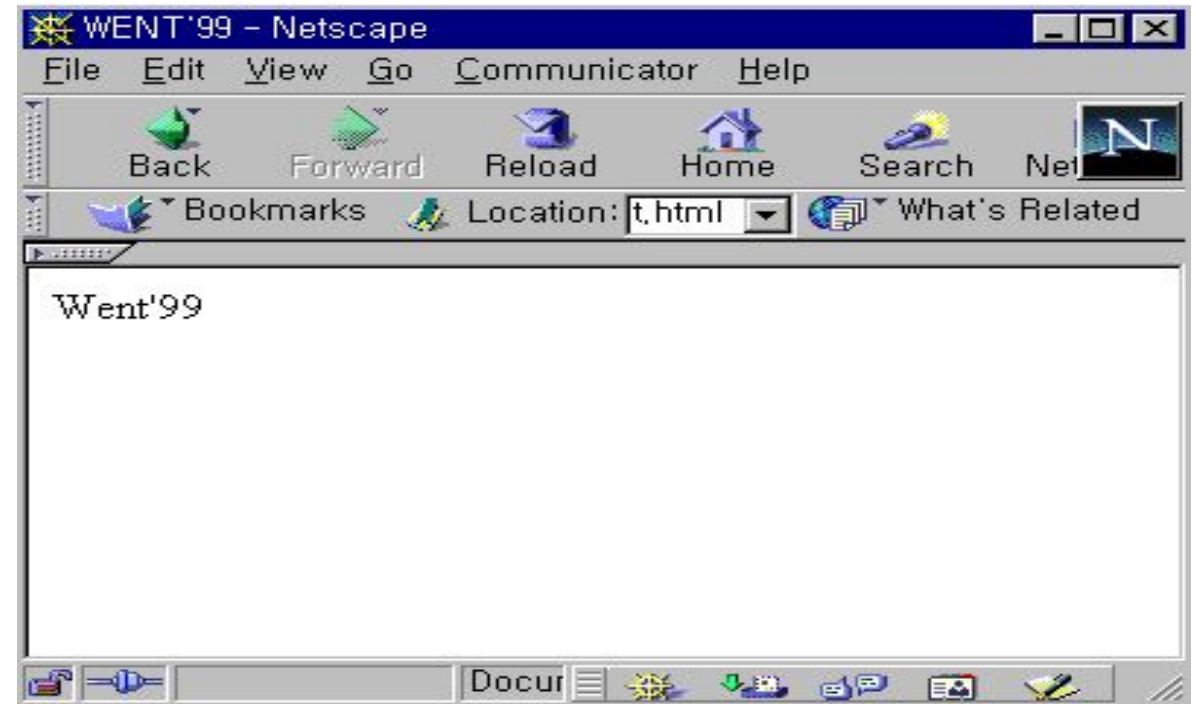
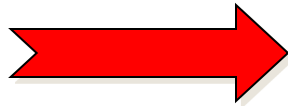
# How Does It Work?

- 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)

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 +
    6;
document.write("hello world");
</script>
</body>
</html>
```



## My First Web Page

My First Paragraph

11

hello world

# 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 Operators

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

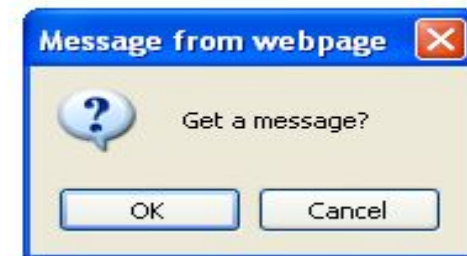
- Confirm Box

- 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!";  
}
```

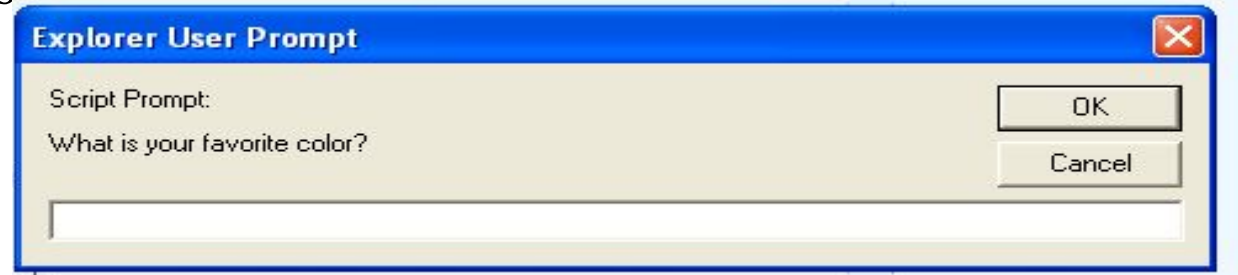
## JavaScript Popup Boxes – 2 (Cont..)

```
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>
```

# 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 Statements

- Executes one action if the condition is true and a different action if the condition is false
- Syntax for an `if . . . else` statement

```
if (conditional expression)  
    statement;  
else  
    statement;
```

## if...else Statements (cont)

- Example:

```
var today = "Tuesday"
if (today == "Monday")
    console.log("Today is Monday");
else
    console.log("Today is not Monday");
```

# Nested `if` and `if...else` Statements (cont)

- Example:

```
var salesTotal = prompt("What is the sales  
total?");  
  
if (salesTotal > 50) {  
    if (salesTotal < 100) {  
        alert("The sales total is between  
50 and 100.");  
    }  
}
```



# switch Statements (cont)

- Example:

```
var age = prompt("Please enter age: ");
switch ( age ) {
    case 25:
        alert("25 is a good age");
        alert("lots of fun");
    case "thirty":
        alert("Thirty is a good age");
    case 40:
        alert("40 is a great age");
    default:
        alert("that is a good age");
}
```

# While Loop and Do ... While

## Syntax

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

## Syntax

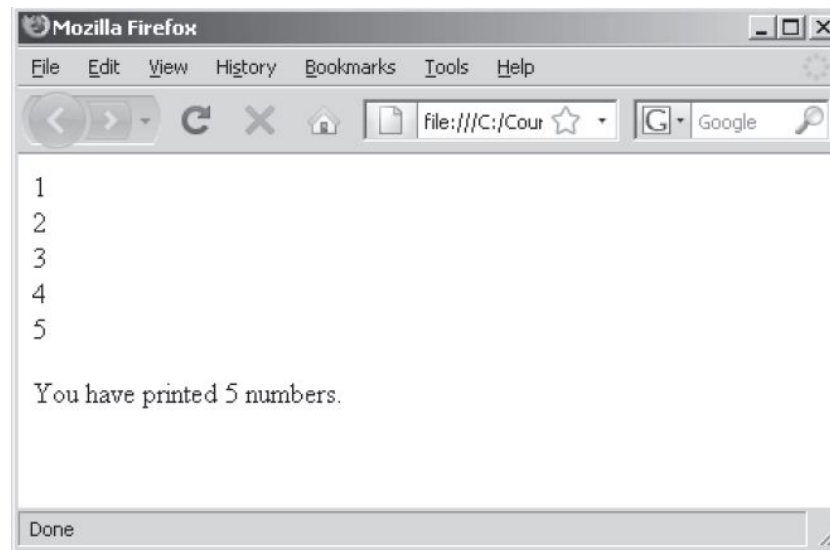
```
do{  
    Statement(s) to be executed;  
} while (expression);
```

# while Statements (cont)

```
var x = 1;
while ( x <= 5) {
    console.log( x );
    ++x;
}
console.log("<p>You have printed 5
numbers.</p>");
```

**Note the special ++  
increment operator**

**++ increment operator  
adds 1 to the variable**



# 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...



# do...while Statements (cont)

- Example:

```
var count = 0;  
do {  
    console.log("The count is: " + count );  
    ++count;  
} while (count < 5);
```

# For Loop

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

# for Statements (cont'd.)

- Example:

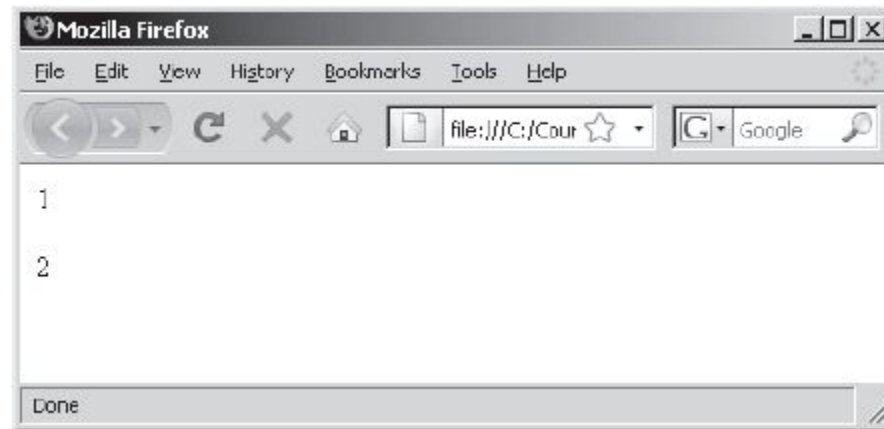
```
var daysOfWeek = new Array();
daysOfWeek[0] = "Monday";
daysOfWeek[1] = "Tuesday";
daysOfWeek[2] = "Wednesday";
daysOfWeek[3] = "Thursday";
daysOfWeek[4] = "Friday";
daysOfWeek[5] = "Saturday";
daysOfWeek[6] = "Sunday";
for (var count = 0; count < daysOfWeek.length;
++count) {
    console.log(daysOfWeek[count] );
}
```

# 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>
```

# Using Break Statements Execution

```
for (var count = 1; count <= 5; ++count) {  
    if (count == 3)  
        break;  
    document.write("<p>" + count + "</p>");  
}
```



# 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

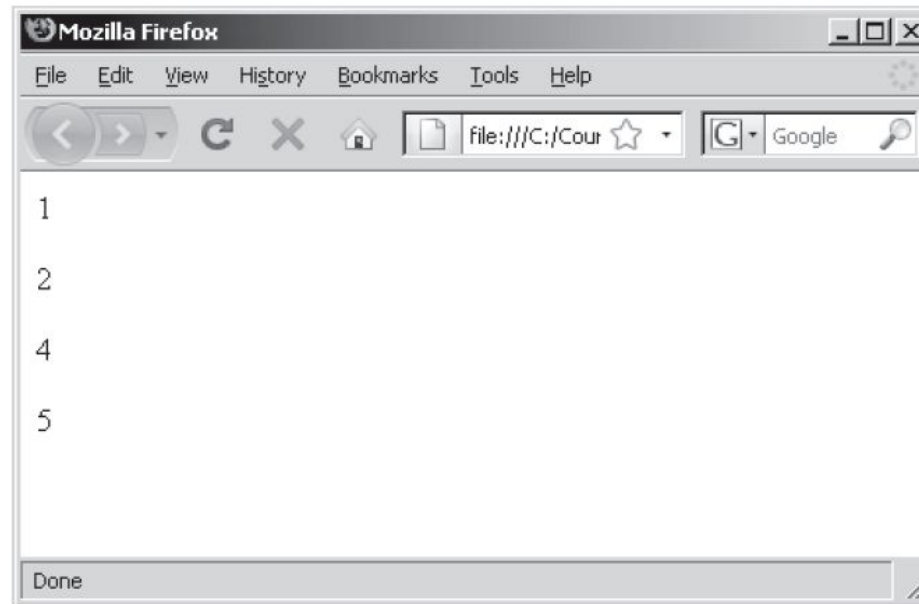
5

Exiting the loop!

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

# Using CONTINUE Statements to Restart Execution (cont'd.)

```
for (var count = 1; count <= 5; ++count) {  
    if (count == 3)  
        continue;  
    console.log(count );  
}
```



# 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>
```

# Placement of user-defined functions

- User defined functions should be declared in a SCRIPT element in the HEAD section of the web page

```
<html>
<head>
<script type = "text/javascript">
function sum( a , b ) {
    return a + b;
}
function avg( a , b ) {
    return ( a + b ) / 2;
}
</script>
</head>
...
```

```
<body>

...
<script type = "text/javascript">
var x = sum( 15 , 12 );
var t = avg( 105 , 65 );
</script>
...

</body>
</html>
```

define functions

call functions

# Placement of user-defined functions (2)

- Many times user-defined functions will be defined in an external JavaScript file

```
function sum( a , b ) {  
    return a + b;  
}  
  
function avg( a , b ) {  
    return ( a + b ) / 2;  
}
```

functions.js

```
<html>  
<head>  
<script type = "text/javascript"  
    src = "functions.js" >  
</script>  
</head>  
...
```

page.html

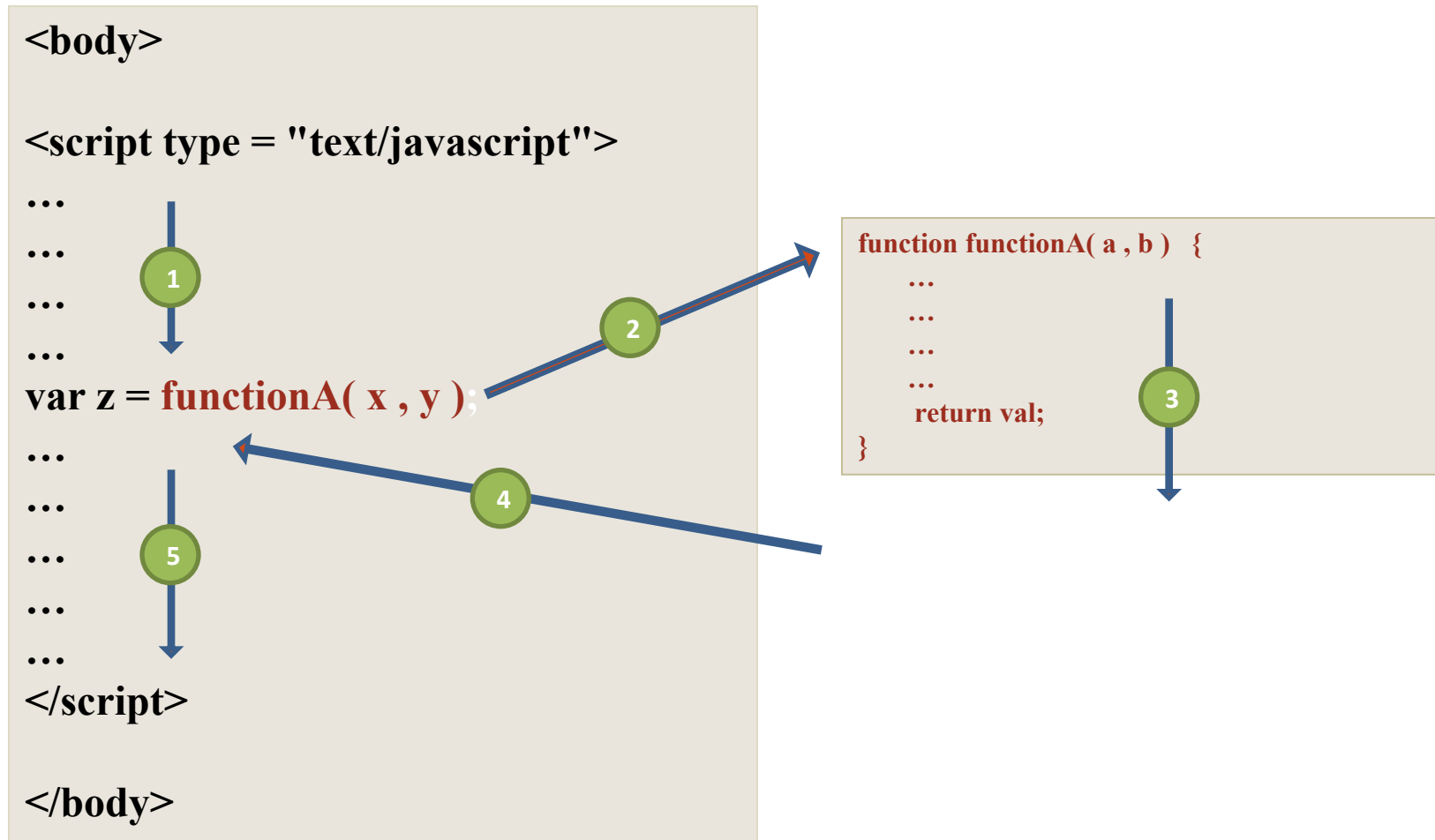
```
<body>
```

page.html

```
...  
<script type = "text/javascript">  
var x = 10;  
var y = 25;  
var z = sum( x , y );  
</script>  
...  
  
</body>  
</html>
```

# Functions – Control Flow

- When a function is called, the control flow jumps to the function code



## Example #1 helloAlert()

A function named **helloAlert( )** in the **HEAD** section of the webpage.

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
  function helloAlert( )
  {
    alert("Hello World!");
  }
</script>
</head>

<body>
<script type="text/javascript">
  helloAlert();
</script>
</body>
</html>
```

define function **helloAlert( )**

call function **helloAlert( )**

## Example #2 welcome()

**A function named `welcome( )` that takes in one parameter, named *user*.  
*When the function is called, the argument "John Smith" is passed to the function.***

```
<script>
function welcome( user )
{
    alert( "Welcome " + user + " to my website!" );
}
</script>
<script>
    welcome( "John Smith" );
</script>
```

In the function declaration,  
we specify the **parameter** `user`

When the function is called,  
"John Smith" is the **argument**  
passed to the function

**This function will display the message box "Welcome John Smith to my website!" when the function is called in the webpage.**

## Functions With a Return Value

### Example #3 average()

```
1 <html><head>
2 <script>
3   function average( num1, num2, num3 ) {
4     var avg = ( num1 + num2 + num3 ) / 3;
5     return avg;
6   }
7 </script>
8 </head>
9 <body>
10 <script>
11   var x = average( 10 , 20 , 60 );
12 </script>
13 </body></html>
```

The `average( )` function above takes three numbers as parameters, and returns the average of the three numbers. On Line 11, when the `average( )` function is called, the value 30 is returned and then stored in the variable `x`.



# Functions – Local Variables

- Some functions declare variables within the function, called *local variables*
- These variables only exist within the function and cannot be used outside the function

```
function calcShipping( state ) {  
    var shippingCost;  
  
    if ( state == "CT" )  
        shippingCost = 8.25;  
    else if (state == "MA" )  
        shippingCost = 12.60;  
    else  
        shippingCost = 18.20;  
  
    return shippingCost;  
}
```

Local variable

```
<body>  
  
<script type = "text/javascript">  
...  
var state = prompt("Enter State:");  
var cost = calcShipping( state );  
...  
</script>  
  
</body>
```

# Functions – Local Variables (cont)

- If we tried to access a local variable outside the function, we would get an error

```
function calcShipping( state ) {  
    var shippingCost;  
  
    if ( state == "CT" )  
        shippingCost = 8.25;  
    else if (state == "MA" )  
        shippingCost = 12.60;  
    else  
        shippingCost = 18.20;  
  
    return shippingCost;  
}
```

```
<body>  
  
<script type = "text/javascript">  
...  
var state = prompt("Enter State:");  
var cost = calcShipping( state );  
...  
document.write( shippingCost ); ✖  
</script>  
  
</body>
```

**Error: the variable shippingCost  
only exists within the calcShipping( )  
function**

# 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
- Syntax

```
<html>
  <element onmouseover="function_name()">
</html>

<script>
  // defining the function function_name
  function function_name()
  {
    // function script
  }
</script>
```

Example :

```
<html>
  <head>
    <script>
      // this function will execute when onmouseover is triggered
      function func()
      {
        // an alert will pop-up when onmouseover gets triggered
        alert("onmouseover alert.");
      }
    </script>
  </head>

  <body>
    <!-- creating an onmouseover event for a <b> tag-->
    <b onmouseover = "func()">
      Bring the cursor here to get an alert.
    </b>
  </body>
</html>
```

```
<html>
  <head>
    <script>
      // this function will execute when onmouseover is triggered
      function func()
      {
        // an alert will pop-up when onmouseover gets triggered
        alert("onmouseover alert.");
      }
    </script>
  </head>

  <body>
    <!-- creating an onmouseover event for a <b> tag-->
    <b onmouseover = "func()">
      Bring the cursor here to get an alert.
    </b>
  </body>
</html>
```

← → ↻ 🏠 ⓘ 127.0.0.1:5500/mouseover.html

**Bring the cursor here to get an alert.**

**127.0.0.1:5500 says**

onmouseover alert.

OK

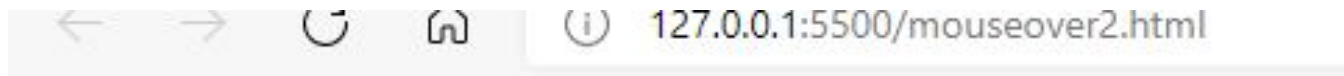


```
<html>
  <script>
    // Defining the function func()
    function func()
    {
      // changing the color
      document.getElementById("para1").style.color = "red";

      // changing the text
      document.getElementById("para1").innerHTML = "I am red in color";
    }
  </script>

  <body>
    <h3> Hover over the text below to change the color and text. </h3>

    <!-- Using onmouseover-->
    <p id="para1" onmouseover="func()"> I am black in color </p>
  </body>
</html>
```



**Hover over the text below to change the color and text.**

I am black in color

After the mouseover event



**Hover over the text below to change the color and text.**

I am red in color

# 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

# JavaScript Events

- Examples of HTML events:
  - When a user clicks the mouse
  - When a web page has loaded
  - When an image has been loaded
  - When the mouse moves over an element
  - When an input field is changed
  - When an HTML form is submitted
  - When a user strokes a key

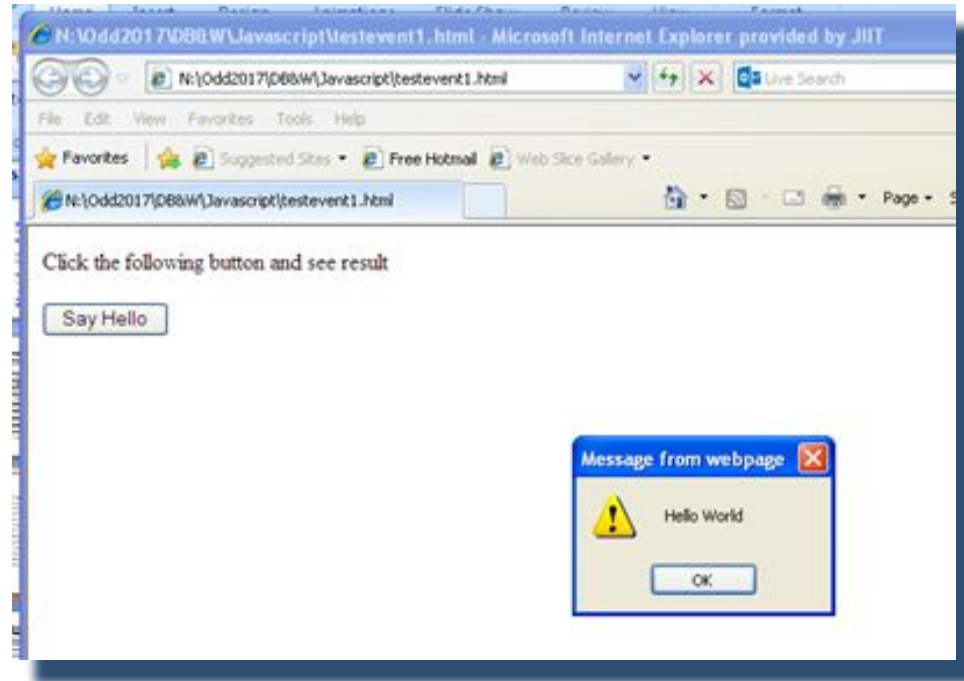
# onclick/onmouseover Events

- The **onclick** event is triggered when the user *clicks the mouse* on a specified HTML element
- The **onmouseover** event is triggered when the user *mouses over* a specified HTML element.

```
<html>
<body>
<h1>Test Events Page</h1>
<h2 onclick="alert('you clicked me')" > Click on me!</h2>
<h2 onmouseover="alert('you hovered over me')" >
Hover over me!</h2>
</body>
</html>
```

# onClick Event Handler Example

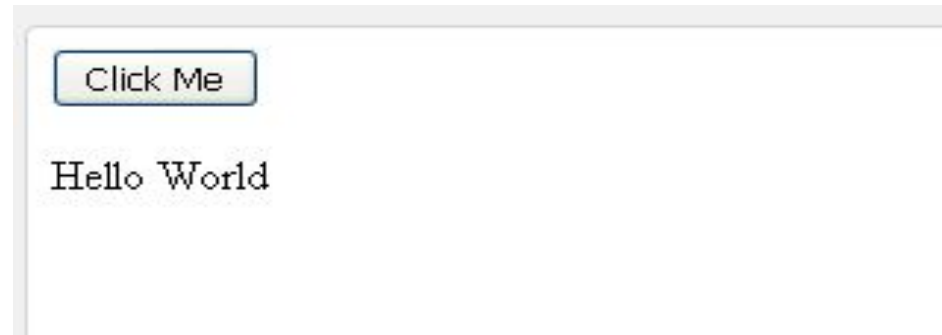
```
<html>
<head>
  <script type="text/javascript">
    <!--
      function sayHello() {
        alert("Hello World")
      }
    //-->
  </script>
</head>
<body>
```



```
<p>Click the following button and see result</p>
<form>
  <input type="button" onclick="sayHello()" value="Say
  Hello" />
</form>
</body>
</html>
```

# Example 2

```
<html>
<body>
<button onclick="myFunction()">Click Me</button>
<p id="demo"></p>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Hello World";
}
</script>
</body>
</html>
```



## The onChange event handler

□ executes JavaScript code when input focus exits the field after the user modifies its text.

Example :

```
<HTML> <TITLE>Example of onChange Event Handler</TITLE> <HEAD>
<SCRIPT LANGUAGE="JavaScript">
function valid(form){
    var input=0;
    input=document.myform.data.value;
    alert("You have changed the value from 10 to " + input );
}
</SCRIPT>
</HEAD> <BODY>
<H3>Example of onChange Event Handler</H3>
Try changing the value from 10 to something else:<br>
<form name="myform">
    <input type="text" name="data" value="10" size=10 onChange="valid(this.form)">
</form>
</BODY>
</HTML>
```

### Example of onChange Event Handler

Try changing the value from 10 to something else:

10



## onFocus: Event Handler

- ❑ **executes JavaScript code when input focus enters the field either by tabbing in or by clicking but not selecting input from the field.**

```
<HTML>
<TITLE>Example of onFocus Event
Handler</TITLE>
<HEAD></HEAD>
<BODY>
<H3>Example of onFocus Event Handler</H3>
Click your mouse in the text box:<br>
<form name="myform">
  <input type="text" name="data" value="" size=10
onFocus='alert("You focused the textbox!!")'>
</form>
</BODY>
</HTML>
```

### Example of onFocus Event Handler

Click your mouse in the text box:

**onSubmit:**

An onSubmit event handler calls JavaScript code when the form is submitted.

```
<HTML>
<TITLE> Example of onSubmit Event Handler </TITLE>
<HEAD>
</HEAD>
<BODY>
<H3>Example of onSubmit Event Handler </H3>
Type your name and press the button<br>
<form name="myform" onSubmit="alert('Thank you ' + myform.data.value
+'!')">
<input type="text" name="data">
<input type="submit" name ="submit" value="Submit this form">
</form>
</BODY>
```

## Example of onSubmit Event Handler

Type your name and press the button:

## **onSelect:**

**A onSelect event handler executes JavaScript code when the user selects some of the text within a text or textarea field.**

```
<HTML>  
<TITLE>Example of onSelect Event Handler</TITLE>  
<HEAD></HEAD>  
<BODY>  
<H3>Example of onSelect Event Handler</H3>  
Select the text from the text box:<br>  
<form name="myform">  
<input type="text" name="data" value="Select This" size=20  
onSelect="alert('This is an example of onSelect!!')">  
</form>  
</BODY>  
</HTML>
```

## **Example of onSelect Event Handler**

Select the text from the text box:

**Write a JavaScript program to add items in an blank array and display the items.**

---

Element 0 = 23

Element 1 = 12

Element 2 = 25

## Solution

```
<!DOCTYPE html>
<html>
<head>
<script>
var x = 0;
var array = Array();

function add_element_to_array()
{
    array[x] = document.getElementById("text1").value;
    alert("Element: " + array[x] + " Added at index " + x);
    x++;
    document.getElementById("text1").value = "";
}

function display_array()
{
    var e = "<hr/>";

    for (var y=0; y<array.length; y++)
    {
        e += "Element " + y + " = " + array[y] + "<br/>";
    }
    document.getElementById("Result").innerHTML = e;
}
</script>
<title>JS Bin</title>
</head>
```

```
<body>
<input type="text" id="text1"></input>
<input type="button" id="button1"
value="Add"
onclick="add_element_to_array();"></input>
<input type="button" id="button2"
value="Display"
onclick="display_array();"></input>
<div id="Result"></div>
</body>
</html>
```

**Write a JavaScript program to calculate the volume of a sphere.**


Input radius value and get the volume of a sphere.

Radius

Volume

Calculate

Write a JavaScript program to illustrate a simple calculator



The image shows a screenshot of a web browser window with the title bar "Firefox". The address bar displays three tabs: "T 200 - Ja...", "East Hartfor...", and "file:....html". The main content area of the browser displays a web form titled "Calculator Form". The form includes two input fields labeled "number 1" and "number 2". Below these fields are four buttons labeled "Add", "Subtract", "Multiply", and "Divide". At the bottom of the form is a label "Result" followed by an empty input field.

**Calculator Form**

number 1

number 2

Result

# 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

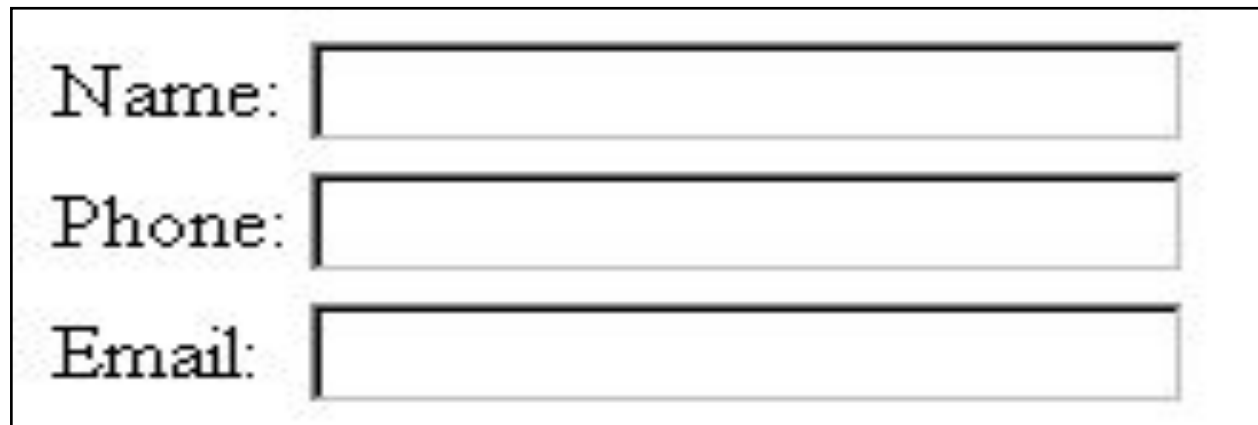
*document.**formname.elementname**.value*

Thus:

document.**addressform.yourname**.value

document.addressform.phone.value

document.addressform.email.value



A diagram of a form with three input fields. The first field is labeled "Name:" and is empty. The second field is labeled "Phone:" and is empty. The third field is labeled "Email:" and is empty. The form is enclosed in a black border.

# 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>
```

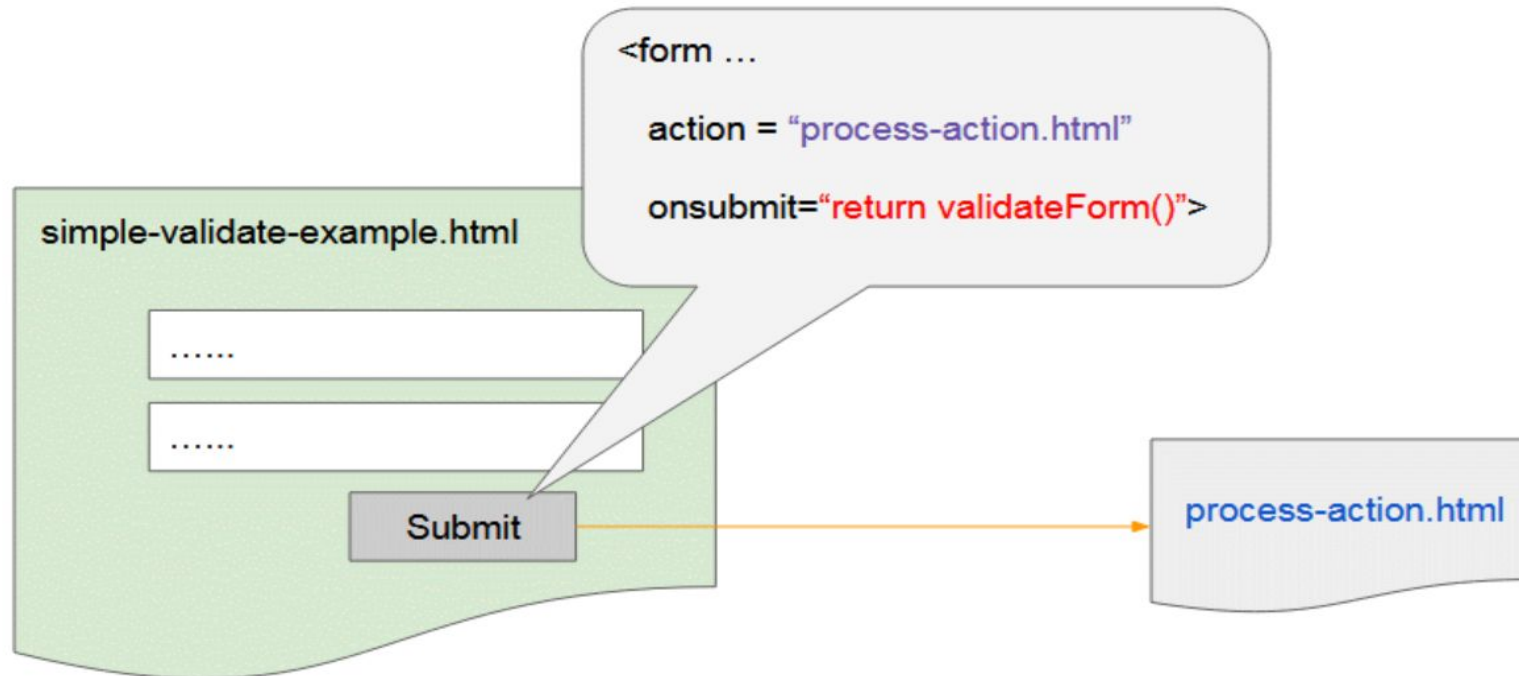
# Form validation

# Form Validation

- Form validation normally used to occur at the **server**.
- If the data entered by a client was **incorrect** or was simply **missing**, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a **lengthy** process which used to put a lot of burden on the server.
- 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



The action attribute of `<form>` is used to specify the page to which data will be given or in other words, this is the page that will process the data sent from the `<form>` of the current page.

```
<html>
  <head>
    <title>Hello Javascript</title>
    <script type = "text/javascript">
      function validateForm() {
        var u = document.getElementById("username").value;
        var p = document.getElementById("password").value;

        if(u== "") {
          alert("Please enter your Username");
          return false;
        }
        if(p == "") {
          alert("Please enter you Password");
          return false;
        }

        alert("All datas are valid!, send it to the server!")

        return true;
      }
    </script>
  </head>
```



```
<body>

  <h2>Enter your Username and Password</h2>

  <div style="border:1px solid #ddd; padding: 5px;">
    <form method="GET" action="process-action.html"
      onsubmit = "return validateForm()">
      Username: <input type="text" name="username" id="username"/>
      <br><br>
      Password: <input type="password" name = "password" id="password"/>
      <br><br>
      <button type="submit">Submit</button>
    </form>
  </div>

</body>
```

## Enter your Username and Password

Username:

Password:

## Enter your Username and Password

Username:

Password:

**127.0.0.1:5500 says**

Please enter you Password

## Enter your Username and Password

Username:

Password:

**127.0.0.1:5500 says**

All datas are valid!, send it to the server!

# Process\_action.html

```
<html>
  <head>
    <title>Process Action</title>

  </head>
  <body>

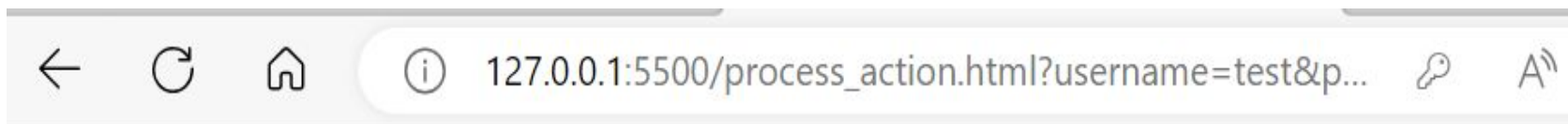
    <h3>Process Action Page</h3>

    OK, I got data!

    <br/><br/>

    <a href="javascript:history.back();">[Go Back]</a>

  </body>
</html>
```



## Process Action Page

OK, I got data!

[\[Go Back\]](#)

# Data field validation

# Access a field data through the field ID

```
// Access field via ID:  
var field = document.getElementById("fieldId");  
  
var value = field.value;
```

## Access Form fields through the name attribute:

```
<form name="myForm" ...>  
  <input type="text" name="username"/>  
  <input type="password" name="password"/>  
  <button type="submit">Submit</button>  
</form>
```

```
// Get form via form name:  
var myForm = document.forms["myForm"];  
  
var u = myForm["username"].value;  
var p = myForm["password"].value;
```

Example: Ask an user to enter a number between 0 and 10.

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

    function validateForm() {

        var myField = document.getElementById("myNumber");
        var value = myField.value;

        if( value == "" || isNaN(value) || value < 0 || value > 10) {
            alert("Invalid input!");
            myField.focus();
            return false;
        }

        return true;
    }
</script>
```



```
<body>

  <h2>Enter a Number between 0 and 10</h2>

  <div style="border:1px solid ■ #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html"
    onsubmit = "return validateForm()">
      Number: <input type="text" id= "myNumber"/>
      <br/><br/>
      <button type="submit">Submit</button>
    </form>

  </div>

</body>
```

## Enter a Number between 0 and 10

Number:

Submit

## Enter a Number between

Number:

Submit

127.0.0.1:5500 says

Invalid input!

OK

## Enter a Number between 0 and 10

Number:

Submit

← ↻ 🏠 ⓘ 127.0.0.1:5500/process\_action.html?

### Process Action Page

OK, I got data!

[\[Go Back\]](#)

# Validate automatically

The browser can automatically validate several types of data on the form, such as adding a required attribute to a form field to tell the browser that this field is mandatory. The browser will automatically check and notify a user if a user does not enter that field.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Required</title>

  </head>
  <body>

    <h2>Required attribute</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process_action.html"
        onsubmit = "return validateForm()">
        Your Name: <input type="text" name = "fullName" value = "" required/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
</html>
```



Required attribute

Your Name:

Submit

! Please fill out this field.

# References for form validation

- <https://o7planning.org/12273/javascript-form-validation>