CMSC 100: Web Programming

Client-Side Scripting

Scripting Language

- a domain-specific language
  - e.g. text-processing languages in Unix-environment
- in Web Programming:
  - *Client-side Scripting*
  - *Server-side Scripting*

Client-Side Scripting

- Client-side scripting is done on the HTML page sent by server to browser

- Script is interpreted by browser and action occurs on browser

Applications of Client-Side Scripting

- Web page responds to or reacts directly with user interaction through HTML *Form elements*, eg, input fields, text areas, buttons, radio buttons, etc.
- Distributing a small amount of information from the server directly on the Web page
- You need to control the Web page appearance based on user selections
- You want to preprocess data before submission to the server

Client-side Script can't ...

- Set or retrieve browser preferences
- Launch an application on client computer
- Read or write files on client computer
- Do much of anything on server computer including accessing a database

Some Client-Side Scripting Languages

- ActionScript
  - used to create animated interactive web applications for Adobe Flash Player
- VBScript
  - Modeled on Visual Basic, developed by Microsoft
- JavaScript
  - Originally developed on Netscape; became the most popular client-side scripting language
- Dart
  - Developed by Google as an alternative to JavaScript

JavaScript

- Is the most popular scripting language in all major browsers e.g.
  - Chrome
  - Internet Explorer
  - Firefox
  - Netscape
  - Opera
- JavaScript is used in millions of web pages

JavaScript and HTML page

```
<html>
<body>
<script type=“text/javascript”>
document.write(“Hello World!”);
</script>
</body>
</html>
```

Tells where the JavaScript starts

Commands for writing output to a page

Tells where the JavaScript ends

This code produce the output on an HTML page:
Hello World!

JavaScript and HTML page

```
<html>
   <head>
   <script src="xyz.js"> </script>
   </head>
<body>
</body>
</html>
```

A separate file

Statements and Comments

- JavaScript statements
  - are codes to be executed by the browser
  - tells the browser what to do
  - commands to the browser
  - add semicolons at the end
  - can be grouped together into blocks using curly brackets
  - try…catch statement allows to test a block of code for errors
- JavaScript comments make the code more readable
  - Single line comments start with //
  - Multi line comments start with /* and end with */

JavaScript Variables

- JavaScript Variables
  - are containers for storing information e.g. x=15;
  - hold values or expressions
  - can hold a text value like in theName="some name"
  - **var statement** can declare JavaScript variables: **var x**; **var y**;
- Variable names
  - are case sensitive i.e. "myVar" is not the same as "myvar"
  - must begin with a letter or the underscore character

JavaScript Operators

- **Arithmetic Operators**:
  - perform arithmetic operations between the values of the variables
    - *addition (+) , subtraction (-), multiplication (*), division (/), modulus (%), increment (++), decrement (- -)*
- **Assignment Operators**:
  - assign values to variables

    =, + =, - =, * =, / =, % =

JavaScript Operators, *cont'd*

- **Comparison Operators**:
  - determines equality or difference between variables or values
    - equal to (= =), exactly equal to (= = =),
    - not equal (!=), greater than (>), less than (<),
    - greater than or equal to (>=), less than or equal to (<=)

- **Logical Operators**:
  - impose the logic between variables or values
    - AND (&&), OR (| |), NOT (!)

- **Conditional Operator**:
  - assign value to a variable based on some conditions
    - ?:

JavaScript Conditional Statements

- **if statement** - to execute some code only if a specified condition is true
- **if...else statement** - to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - to select one of many blocks of code to be executed
- **switch statement** - to select one of many blocks of code to be executed

JavaScript Looping

- JavaScript looping
  - Executes the same block of codes
  - Executes a specified number of times
  - Execution can be controlled by some control logic
    - uses **for**, **while**, **do….while** statements

JavaScript Functions and Events

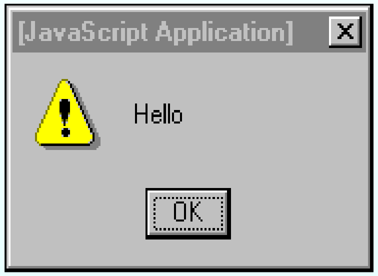
- **JavaScript Functions**
  - Can be called with the function name
  - Can also be executed by an event
  - Can have parameters and return statement
- **Events**
  - are actions that can be detected e.g. onMouseOver, onMouseOut etc.
  - are normally associated with functions
  - <input type="text" size="30" id="email" onChange="checkEmail()">

JavaScript Event Example

```
<html>
<head><title>My Page</title></head>
<body>
<p>
<a href="myfile.html">My Page</a>
<br />
<a href="myfile.html"
onMouseover="window.alert('Hello');">
My Page</A>
</p>
</body>
</html>
```

An Event

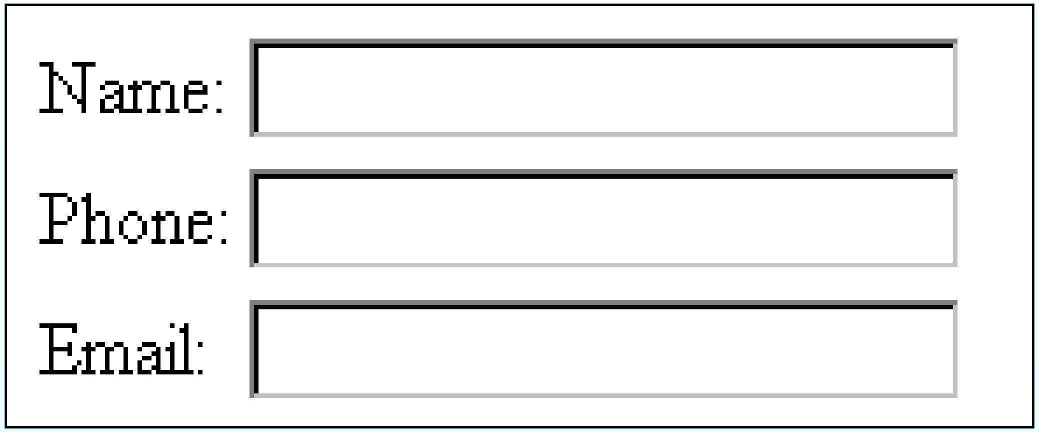
JavaScript written inside HTML



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
  - Uses JavaScript Document Object Model (DOM)

Naming Form Elements in HTML


Name:
Phone:
Email:

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

Forms and JavaScript

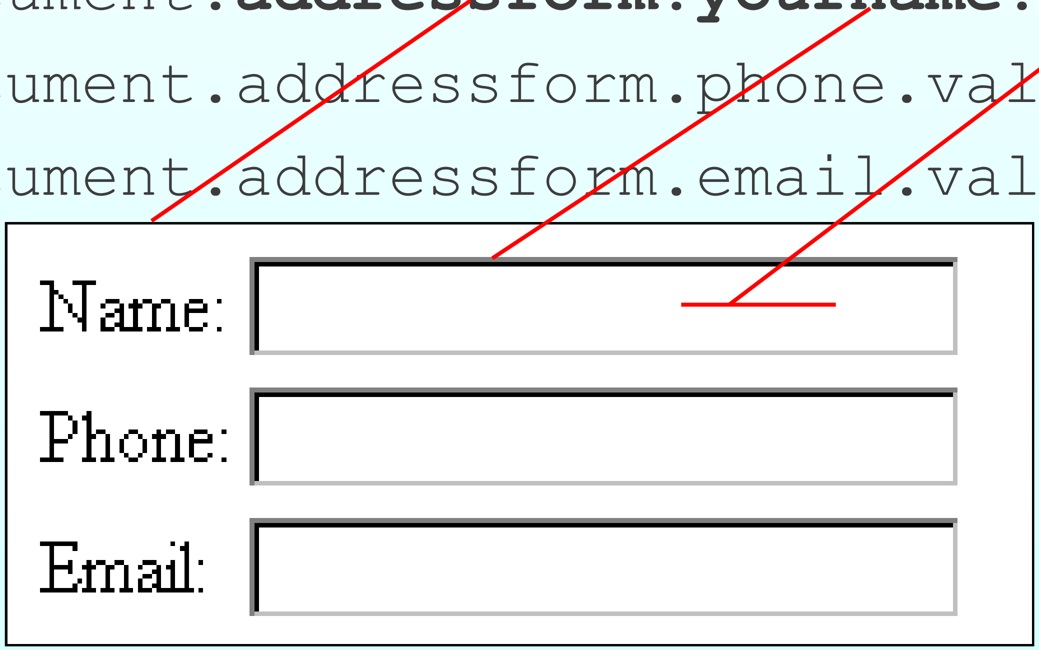
*document.**formname.elementname**.value*

Thus:

document.**addressform.yourname**.value

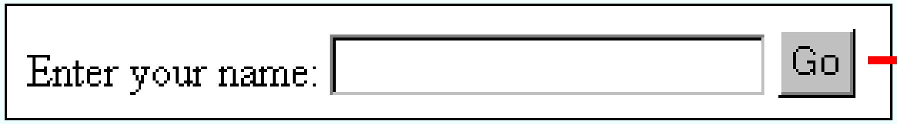
document.addressform.phone.value

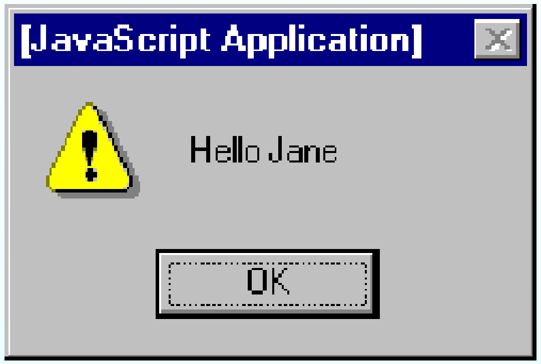
document.addressform.email.value



Using Form Data

Personalising an alert box





```
<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>
```

JavaScript: Events

- Javascript actions may be triggered from events, e.g. changes on form fields or a submit button being clicked:

– onfocus =	Form field gets focus (validation)
– onblur=	Form field looses focus (validation)
– onchange=	Content of a field changes (validation)
– onselect=	Text is selected
– onmouseover=	Mouse moves over a link (animated buttons)
– onmouseout=	Mouse moves out of a link (animated …)
– onclick=	Mouse clicks an object
– onload=	Page is finished loading (initial actions, info,)
– onSubmit=	Submit button is clicked (validation etc.)

JavaScript: DOM

- DOM is a representation of the document in an object form, accessible from JavaScript programs

JavaScript and OOP

- **JavaScript**
  - is an Object Oriented Programming language
  - contains built-in JavaScript objects, e.g. window, history, array, date, etc.
  - objects contain Properties and Methods

You can read more about JS Objects on:

http://www.w3schools.com/js/js_objects.asp

Additional Note

- Many JavaScript libraries are now available, e.g.
  - JQuery
  - Prototype
  - MooTools
  - YUI
  - Google API

Read more on http://www.w3schools.com/js/js_libraries.asp