Basic Constraint Validation of a form

- <input type=”email” /> The field value must be an email
- **required**: A required attribute indicates that a value must be specified for the input element.
- **maxlength**: This is an integer value that specifies the maximum number of characters allowed for a particular input field.

```
<input type=”text” maxlength=”20” />
```

pattern: The pattern attribute is used to specify a regular expression and the field value must match this pattern. This attribute can be used with input types like text, password, email, url, tel and search.

```
<input type=”text” pattern=”[.4,]”
                    title=“four or  more characters”/>
```

NB:

- The browser displays an error message if any validation errors occur.

Automatic HTML Form Validation

- If a form field is empty, the **required** attribute prevents this form from being submitted

```
<body>
<form method="post">
 FirstName: <input type="text" name="fname" required>
    <br>
LastName: <input type="text" name="Lname" required>
 Country <input type="text" pattern=".{2}"
                        title="two characters"/>
<input type="email" />
<br>
<input type="submit" value="Submit">
   </form>
</body>
```

JavaScript

Language Fundamentals

About JavaScript

- JavaScript *is not Java,* or even *related to* Java
  - The original name for JavaScript was “LiveScript”
  - The name was changed when Java became popular
  - Now that Microsoft no longer likes Java, its name for their JavaScript dialect is “Active Script”
- Statements in JavaScript resemble statements in Java, because both languages borrowed heavily from the C language
  - JavaScript should be fairly easy for Java programmers to learn
  - However, JavaScript *is* a complete, full-featured, complex language
- JavaScript is seldom used to write complete “programs”
  - Instead, small bits of JavaScript are used to add functionality to HTML pages
  - JavaScript is often used in conjunction with HTML “forms”
- JavaScript is *reasonably* platform-independent

Primitive data types

- JavaScript has three "primitive" types: number, string, and boolean
- Strings may be enclosed in single quotes or double quotes
  - Strings can contains \n (newline), \" (double quote), etc.
- Booleans are either true or false
  - 0, "0", empty strings, undefined, null, and NaN are false , other values are true

Variables

- Variables are declared with a var statement:
  - var pi = 3.1416, x, y, name = "Dr. Dave" ;
  - Variables names must begin with a letter or underscore
  - Variable names are case-sensitive
  - Variables are *untyped* (they can hold values of any type)
  - The word var is optional (but it's good style to use it)
- Variables declared within a function are local to that function (accessible only within that function)
- Variables declared outside a function are global (accessible from anywhere on the page)

Operators, I

- Because most JavaScript syntax is borrowed from C (and is therefore just like Java)
- Arithmetic operators (all numbers are floating-point):
  `+` `-` `*` `/` `%` `++` `--`
- Comparison operators:
  `<` `<=` `==` `!=` `>=` `>`
- Logical operators:
  `&&` `||` `!` (`&&` and `||` are *short-circuit* operators)
- Assignment operators:
  `+=` `-=` `*=` `/=` `%=`

Operators, II

- String operator:

  +

```
text1 = "John";
text2 = “Adams";
text3 = text1 + " " + text2;
```

- The conditional operator:

  condition ? value_if_true : value_if_false

Comments

- Comments are as in C or Java:
  - Between /* and */

Statements, I

- Most JavaScript statements are also borrowed from C
  - Assignment: `greeting = "Hello, " + name;`
  - If statements:

    `if (condition) statement;`

    `if (condition) statement; else statement;`
  - Familiar loop statements:

    `while (condition) statement;`

    `do statement while (condition);`

    `for (initialization; condition; increment) statement;`

Statements, II

- The switch statement:

```
switch (expression) {
   case label :
      statement;
      break;
   case label :
      statement;
      break;
   ...
   default : statement;
}
```

- Other familiar statements:
  - break;
  - continue;

Functions

- Functions should be defined in the <head> of an HTML page, to ensure that they are loaded first
- The syntax for defining a function is:
  function ***name***(***arg1***, …, ***argN***) { ***statements*** }
  - The function may contain return ***value***; statements
  - Any variables declared within the function are local to it
- The syntax for calling a function is just
  name(***arg1***, …, ***argN***)
- Simple parameters are passed *by value,* objects are passed *by reference*

JavaScript is not Java

- JavaScript has some features that *resemble* features in Java:
  - JavaScript has Objects and primitive data types
  - JavaScript has qualified names; for example, `document.write("Hello World");`
  - JavaScript has Events and event handlers
  - Exception handling in JavaScript is *almost* the same as in Java
- JavaScript has some features *unlike* anything in Java:
  - *Variable names* are untyped: the type of a variable depends on the value it is currently holding
  - Objects and arrays are defined in quite a different way

Warnings

- JavaScript is a big, complex language
- JavaScript is not totally platform independent
  - Expect different browsers to behave differently
- Browsers aren't designed to report errors

Using JavaScript in HTML document

- JavaScript code is included within <script> tags:
  - <script type="text/javascript">
    //some javascript code here
    </script>
- Notes:
  - **The semicolon** at the end of the JavaScript statement is optional
    - You need semicolons if you put two or more statements on the same line

JavaScript isn't always available

- Some old browsers do not recognize script tags
  - These browsers will ignore the script tags but will *display* the included JavaScript
- Some users turn off JavaScript
  - Use the <noscript>***message***</noscript> to display a message in place of whatever the JavaScript would put there

```
<script type="text/javascript">
        //some javascript code here
    <noscript>message</noscript>
</script>
```

Where to put JavaScript

- JavaScript can be put in the <head> or in the <body> of an HTML document
  - JavaScript *functions* should be defined in the <head>
    - This ensures that the function is loaded before it is needed
  - JavaScript in the <body> will be executed as the page loads
- JavaScript can be put in a separate .js file
  - <script src="myJavaScriptFile.js"></script>
  - Put this HTML wherever you would put the actual JavaScript code
  - An external .js file lets you use the same JavaScript on multiple HTML pages
  - The external .js file cannot itself contain a <script> tag
- JavaScript can be put in an HTML *form object,* such as a button
  - This JavaScript will be executed when the form object is used

```
<!doctype html>
<html>
<head>
    <script type=“text/javascript”>
        //some javascript code here
    </script>
</head>
<body>
</body>
</html>
```

```
<!doctype html>
<html>
<head>
</head>
<body>
  <script type=“text/javascript”>
        //some javascript code here
    </script>
</body>
</html>
```

```
<!doctype html>
<html>
<head>
    <script src=“filename.js”>
    </script>
</head>
<body>
</body>
</html>
```

```
<!doctype html>
<html>
<head>
</head>
<body>
<button type="button" onclick=“some
javascript code here”> text <button>
</body>
</html>
```

JavaScript Events

- Events are things that happen to HTML elements when JavaScript is used in HTML pages.
- Event can be something the browser does, or something a user does.

HTML events:

  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked

- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.
- With single quotes:

  *<some-HTML-element some-event=***'*some JavaScript*'***>*

Common HTML Events

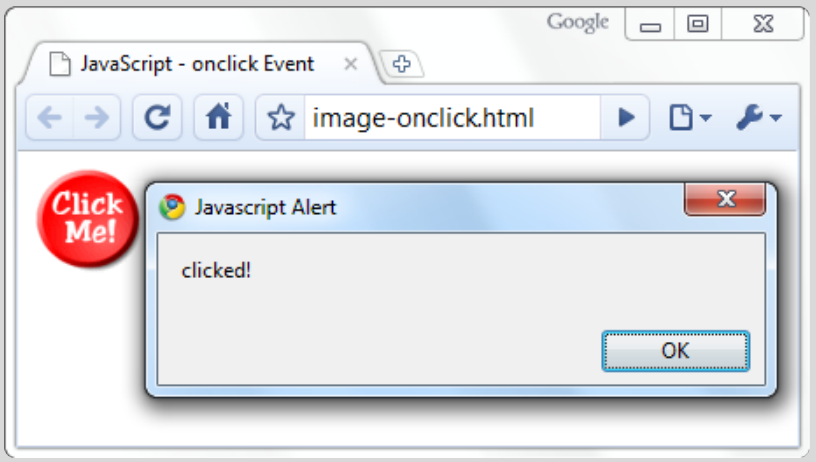
- List of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
Onload	The browser has finished loading the page

Calling a JavaScript Function from Event Handler – Example

image-onclick.html

```
<html>
<head>
<script type="text/javascript">
  function test (message) {
    alert(message);
  }
</script>
</head>

<body>
  <img src="logo.gif"
    onclick="test('clicked!')" />
</body>
</html>
```



JavaScript can "display" data in different ways:

- Writing into an alert box, using **window.alert()**.
- Writing into the HTML output using **document.write()**.
- Writing into an HTML element, using **innerHTML**.

Using window.alert()

```
<!DOCTYPE html>
 <html>
  <head></head>
 <body>

 <h1>Window  Alert</h1>
 <p>Welcome!</p>
  <p> This is a demonstration Using the window  alert output.</p>
 <script>
 window.alert(“welcome to my page”);
 </script>

 </body>
 </html>
```

Standard Popup Boxes

- Alert box with text and [OK] button
  - Just a message shown in a dialog box:

```
alert("Some text here");
```

- Confirmation box
  - Contains text, [OK] button and [Cancel] button:

```
confirm("Are you sure?");
```

- Prompt box
  - Contains text, input field with default value:

```
prompt ("enter amount", 10);
```

Using document.write()

```
<!DOCTYPE html>
  <html>
   <head></head>
  <body>

  <h1>Document Write</h1>
  <p>Welcome!</p>
   <p>This is a demonstration Using the document  write output.</p>
  <script>
  document.write (“Thank you for visiting this page”);
  </script>

  </body>
  </html>
```

```
<!DOCTYPE html>
  <html>
   <head></head>
  <body>

  <h1>Document Write</h1>
  <p>Welcome!</p>
   <p>This is a demonstration Using the document  write output.</p>

   <button onclick="document.write('Thank you for visiting this page’);">
        Click here
    </button>

  </body>
  </html>
```

Using innerHTML

- To access an HTML element, JavaScript can use the **document.getElementById(id)** method.
- The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

```
<!DOCTYPE html>
  <html>
   <head></head>
  <body>

  <h1>Using innerHTML</h1>
  <p>Welcome!</p>
   <p>This is a demonstration Using the innerHTML output.</p>
  <p id=“par"></p>

    <script>
   document.getElementById(‘par’).innerHTML =“Hello User!”;
   </script>

  </body>
 </html>
```

Sum of Numbers – Example

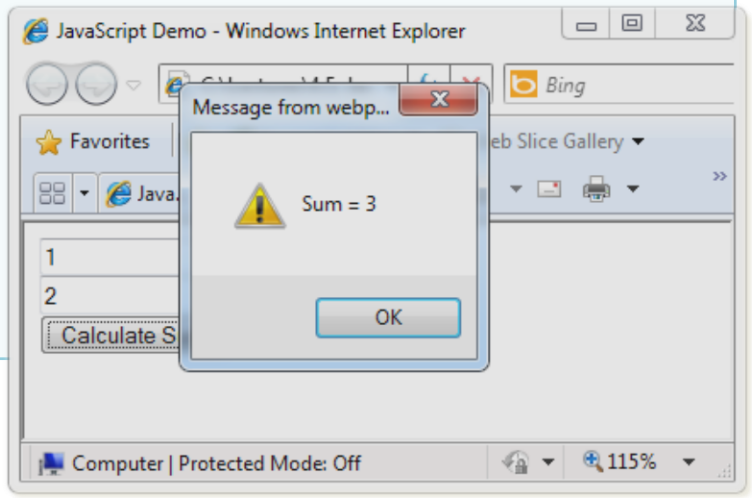
sum-of-numbers.html

```
<html>

<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calcSum() {
      value1 =
        parseInt(document.mainForm.textBox1.value);
      value2 =
        parseInt(document.mainForm.textBox2.value);
      sum = value1 + value2;
      document.mainForm.textBoxSum.value = sum;
    }
  </script>
</head>
```

Sum of Numbers – Example (2)

sum-of-numbers.html (cont.)

```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript: calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>

</html>
```



Common Element Properties

- Most of the properties are derived from the HTML attributes of the tag
  - E.g. `id`, `name`, `href`, `alt`, `title`, `src`, etc…
- `style` property – allows modifying the CSS styles of the element
  - Corresponds to the inline style of the element
    - Not the properties derived from embedded or external CSS rules
  - Example: `style.width`, `style.marginTop`, `style.backgroundImage`

Other Javascript Functions Examples

- To make sliding images
- To show and hide password characters
- Dropdown links
- Handling the ok and cancel buttons of the confirm and prompt pop up boxes
- Validating the HTML Form
- Making the online multiple choice test