Școala
informală
de IT

Javascript

Agenda

- HTML, CSS, JavaScript
- Introduction to JavaScript
  - What is JavaScript ?
  - Implementing JavaScript into Web pages
- JavaScript Syntax
- Document Object Model
- Debugging in JavaScript

HTML, CSS and Javascript?

HTML, CSS, JavaScript

- HTML defines Web sites content through semantic tags (headings, paragraphs, lists, …)
- CSS defines 'rules' or 'styles' for presenting every aspect of an HTML document
  - Font (family, size, color, weight, etc.)
  - Background (color, image, position, repeat)
  - Position and layout (of any object on the page)
- JavaScript defines dynamic behavior
  - Programming logic for interaction with the user, to handle events, etc.

Introduction to JavaScript

Dynamic Behavior in a Web Page

What is JavaScript ?

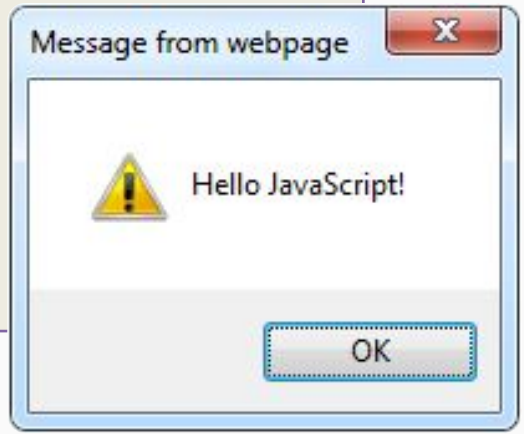
- JavaScript is a front-end scripting language developed by Netscape for dynamic content
  - Lightweight, but with limited capabilities
  - Can be used as object-oriented language
- Client-side technology
  - Embedded in your HTML page
  - Interpreted by the Web browser
- Simple and flexible
- Powerful to manipulate the DOM

What can JavaScript do?

- Can handle events
- Can read and write HTML elements and modify the DOM tree
- Can validate form data
- Can access / modify browser cookies
- Can detect the user's browser and OS
- Can be used as object-oriented language
- Can handle exceptions
- Can perform asynchronous server calls (AJAX)

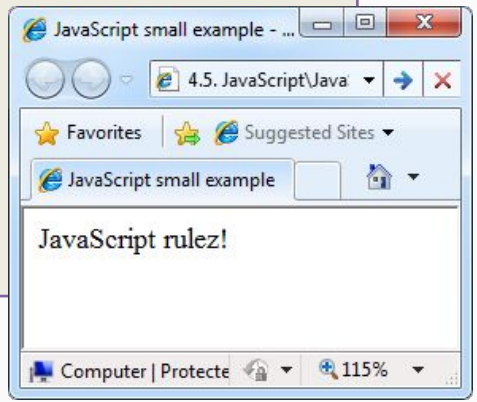
The First Script

```
<html>

<body>
  <script>
     alert('Hello JavaScript!');
  </script>
</body>

</html>
```



Another Small Example

```
<html>

<body>
  <script>
     document.write('JavaScript rulez!');
  </script>
</body>

</html>
```



Using JavaScript Code

- The JavaScript code can be placed in:
  - `<script>` tag in the head
  - `<script>` tag in the body – not recommended
  - External files, linked via `<script>` tag the head
    - Files usually have `.js` extension

```
<script src="scripts.js">
<!- code placed here will not be executed! -->
</script>
```

    - Highly recommended
    - The `.js` files get cached by the browser

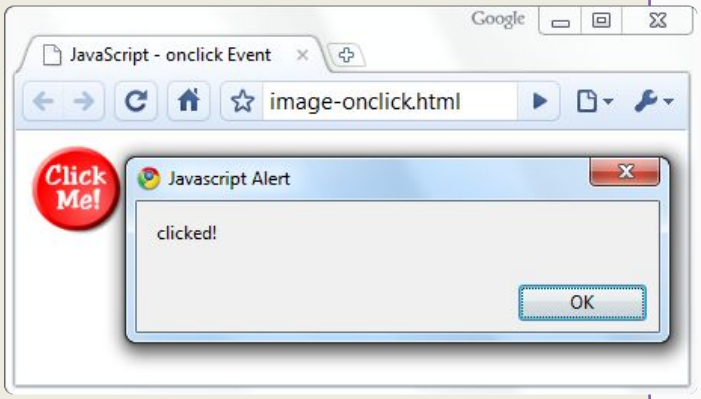
JavaScript - When is Executed ?

- JavaScript code is executed during the page loading or when the browser fires an event
  - All statements are executed at page loading
  - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
  - Executed when the event is fired by the browser

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

Calling a JavaScript Function from Event Handler - Example

```
<html>
<head>
<script>
  function test(message) {
    alert(message);
  }
</script>
</head>

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



Using External Script Files

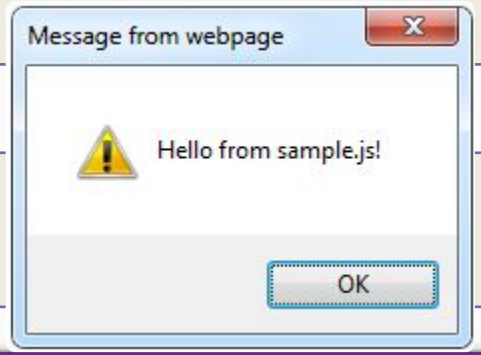
- **Using external script files:**

```
<img <html>
<head>
  <script src="sample.js">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

The <script> tag is always empty

- **External JavaScript file:**

```
function sample() {
  alert('Hello from sample.js!')
}
```



JavaScript Syntax

JavaScript Syntax

- The JavaScript syntax is similar to C# and Java
  - Operators (`+`, `*`, `=`, `!=`, `&&`, `++`, …)
  - Variables (typeless)
  - Conditional statements (`if`, `else`)
  - Loops (`for`, `while`)
  - Arrays (`my_array[]`) and associative arrays (`my_array['abc']`)
  - Functions (can return value)

Data Types

- JavaScript data types:
  - Numbers (integer, floating-point)
  - Boolean (true / false)
- String type – string of characters

```
var myName = "You can use both single or double quotes for strings";
```

- Arrays and associative arrays (hash tables)

```
var my_array = [1, 5.3, "aaa"];
var my_hash = {a:2, b:3, c:"text"};
```

- Undefined and null types

```
var someData;              // undefined until initialized
someData = null;           // data has no type nor value
```

Everything is Object

- Every variable can be considered as object
  - For example strings and arrays have member functions:

```
var test = "some string";
alert(test[7]); // shows letter 'r'
alert(test.charAt(5)); // shows letter 's'
alert("test".charAt(1)); //shows letter 'e'
alert("test".substring(1, 3)); //shows 'es'
```

```
var arr = [1, 3, 4];
alert(arr.length); // shows 3
arr.push(7); // appends 7 to end of array
alert(arr[3]); // shows 4th element which is 7
```

String Operations

- The + operator joins strings

```
string1 = "fat";
string2 = "cats";
alert(string1 + " " + string2);  // fat cats
```

- What is "9" + 9?

```
alert("9" + 9); // 99, + acts as concatenation
```

- Converting string to number (integer or decimal):

```
alert(parseInt("9.2") + 9); // 18, + acts as addition
alert(parseFloat("9.2") + 9); // 18.2, + acts as addition
```

Arrays Operations and Properties

- Declaring new empty array:

```
var arr = new Array();
```

- Declaring an array holding few elements:

```
var arr = ["Ford", "Volvo", "BMW", "Peugeot"];
```

- Appending an element / getting the last element:

```
arr.push("Audi");
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

- Finding element's index in the array:

```
arr.indexOf("Volvo");
```

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);
```

Sum of Numbers using prompt - Example

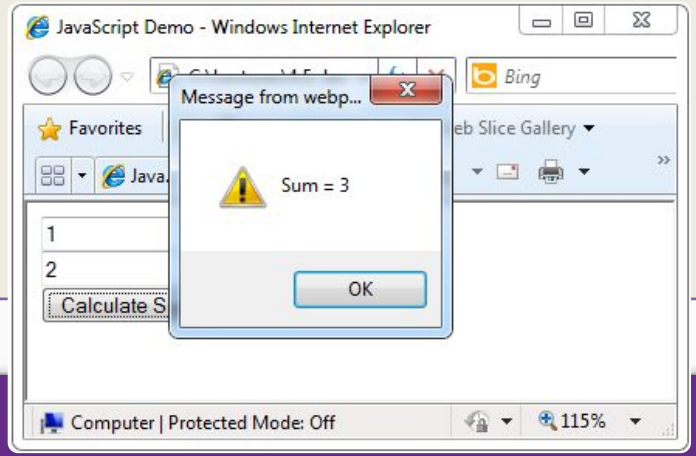
```
<html>

<head>
  <title>JavaScript Demo</title>
  <script>
    function calcSum() {
      value1 = parseInt(prompt("Give first value"));
      value2 = parseInt(prompt("Give second value"));
      sum = value1 + value2;
      alert(sum);
    }
  </script>
</head>
<body>
<input type="button" value="Process" onclick="calcSum()" />
</body>
```

Sum of Numbers using textbox value - Example

```
<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>
```

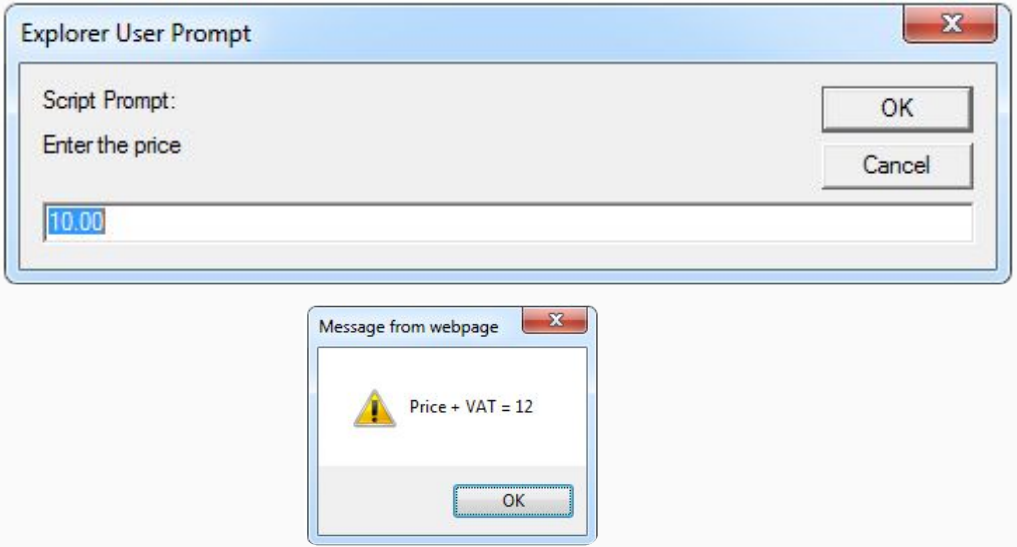
Sum of Numbers - Example (2)

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

</html>
```



JavaScript Prompt - Example

```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.24);
```



Conditional Statement (if)

```
unitPrice = 1.30;
if (quantity > 100) {
  unitPrice = 1.20;
}
else {
  unitPrice = 1.40;
}
```

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal
!=	Not equal

Conditional Statement (if)

- The condition may be of boolean or integer type:

```
var a = 0;
var b = true;
if (typeof(a) == "undefined" || typeof(b) == "undefined") {
  document.write("Variable a or b is undefined.");
}
else if (!a && b) {
  document.write("a == 0; b == true;");
} else {
  document.write("a == " + a + "; b == " + b + ";");
}
```

Switch Statement

- The `switch` statement works like in C#:

```
switch (variable) {
  case 1:
    // do something
    break;
  case 'a':
    // do something else
    break;
  case 3.14:
    // another code
    break;
  default:
    // something completely different
}
```

Loops

- Like in C#
  - `for` loop
  - `while` loop
  - `do ... while` loop

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

```
var counter = 0;
do
  alert(++counter);
while (counter < 5);
```

```
for (var counter = 0; counter < 4; counter++) {
  alert(counter);
}
```

Functions

- Code structure – splitting code into parts
- Data comes in, processed, result returned

```
function average(a, b, c)
{
    var total;
    total = a + b + c;
    return total / 3;
}
```

Parameters come in here

Declaring variables is optional. Type is never declared

Value returned here

Function Arguments and Return Value

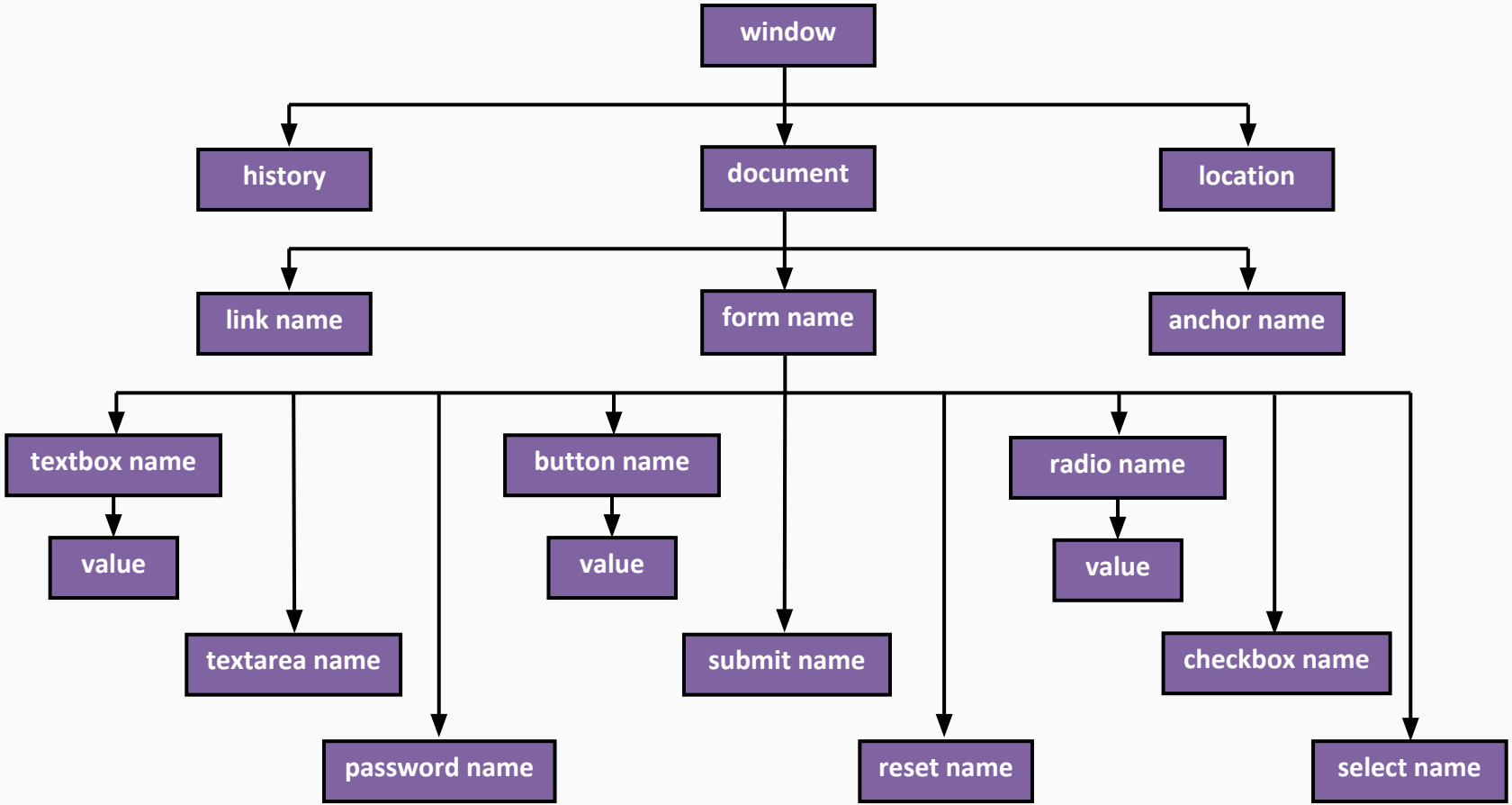
- Functions are not required to return a value
- When calling function it is not obligatory to specify all of its arguments
  - The function has access to all the arguments passed via `arguments` array

```
function sum() {
  var sum = 0;
  for (var i = 0; i < arguments.length; i ++)
    sum += parseInt(arguments[i]);
  return sum;
}

alert(sum(1, 2, 4));
```

Document Object Model
(DOM)

Document Object Model

- Every HTML element is accessible via the JavaScript DOM API
- Most DOM objects can be manipulated by the programmer
- The event model lets a document to react when the user does something on the page
- Advantages
  - Create interactive pages
  - Updates the objects of a page without reloading it


window
history
document
location
link name
form name
anchor name
textbox name
button name
radio name
value
value
value
textarea name
submit name
checkbox name
password name
reset name
select name

Accessing Elements

- Access elements via their ID attribute

```
var elem = document.getElementById("some_id")
```

- Via the class name attribute

```
var cls = document.getElementsByClassName("some_name")
```

- Via tag name

```
var images = document.getElementsByTagName("img")
```

  - Returns array of descendant `<img>` elements of the element "el"
  - Instead of document, any element can be used starting access point

DOM Manipulation

- Once we access an element, we can read and write its attributes

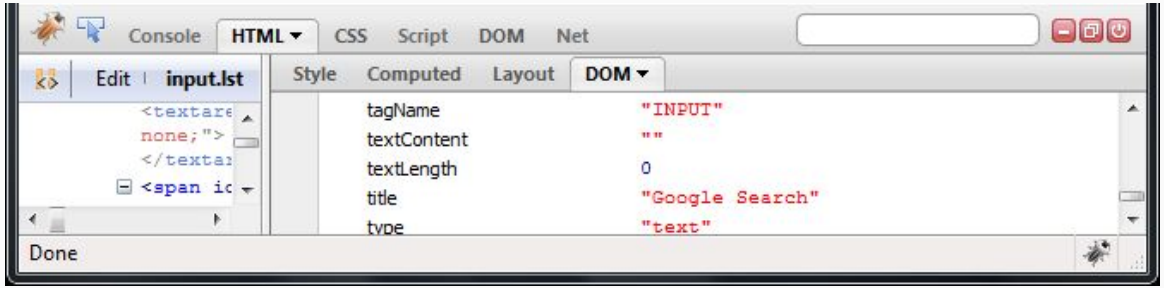
```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".png";
  var statusDiv = document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state";
}
…
<img src="lamp_off.png" onmouseover="change('on')"
  onmouseout="change('off')" />
<div id="statusDiv">The lamp is off</div>
```

Common Element Properties

- Most of the properties are derived from the HTML attributes of the tag
  - E.g. `id`, `name`, `value`, `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`

Common Element Properties

- `className` – the `class` attribute of the tag
- `innerHTML` – holds all the entire HTML code inside the element
- Read-only properties with information for the current element and its state
  - `tagName`, `offsetWidth`, `offsetHeight`, `scrollHeight`, `scrollTop`, `nodeType`, etc...



Accessing Elements through the DOM Tree Structure

- **We can access elements in the DOM through some tree manipulation properties:**
  - `element.childNodes`
  - `element.parentNode`
  - `element.nextSibling`
  - `element.previousSibling`
  - `element.firstChild`
  - `element.lastChild`

Accessing Elements through the DOM Tree - Example

```
var el = document.getElementById('div_tag');
alert(el.childNodes[0].value);
alert(el.childNodes[1].getElementsByTagName('span').id);
…
<div id="div_tag">
  <input type="text" value="test text" />
  <div>
    <span id="test">test span</span>
  </div>
</div>
```

- **Warning: may not return what you expected due to Browser differences**

The HTML DOM Event Model

- JavaScript can register event handlers
  - Events are fired by the Browser and are sent to the specified JavaScript event handler function
  - Can be set with HTML attributes:

```
<img src="test.gif" onclick="imageClicked()" />
```

  - Can be accessed through the DOM:

```
var img = document.getElementById("myImage");
img.onclick = imageClicked;
…
<img src="test.gif" />
```

The HTML DOM Event Model

- All event handlers receive one parameter
  - It brings information about the event
  - Contains the type of the event (mouse click, key press, etc.)
  - Data about the location where the event has been fired (e.g. mouse coordinates)
  - Holds a reference to the event sender
    - E.g. the button that was clicked

Common DOM Events

- **Mouse events:**
  - `onclick`, `onmousedown`, `onmouseup`
  - `onmouseover`, `onmouseout`, `onmousemove`
- **Key events:**
  - `onkeypress`, `onkeydown`, `onkeyup`
  - **Only for input fields**
- **Interface events:**
  - `onblur`, `onfocus`
  - `onscroll`

Common DOM Events

- Form events
  - `onchange` – for input fields
  - `onsubmit`
    - Allows you to cancel a form submission
    - Useful for form validation
- Miscellaneous events
  - `onload`, `onunload`
    - Allowed only for the `<body>` element
    - Fires when all content on the page was loaded / unloaded

onload Event - Example

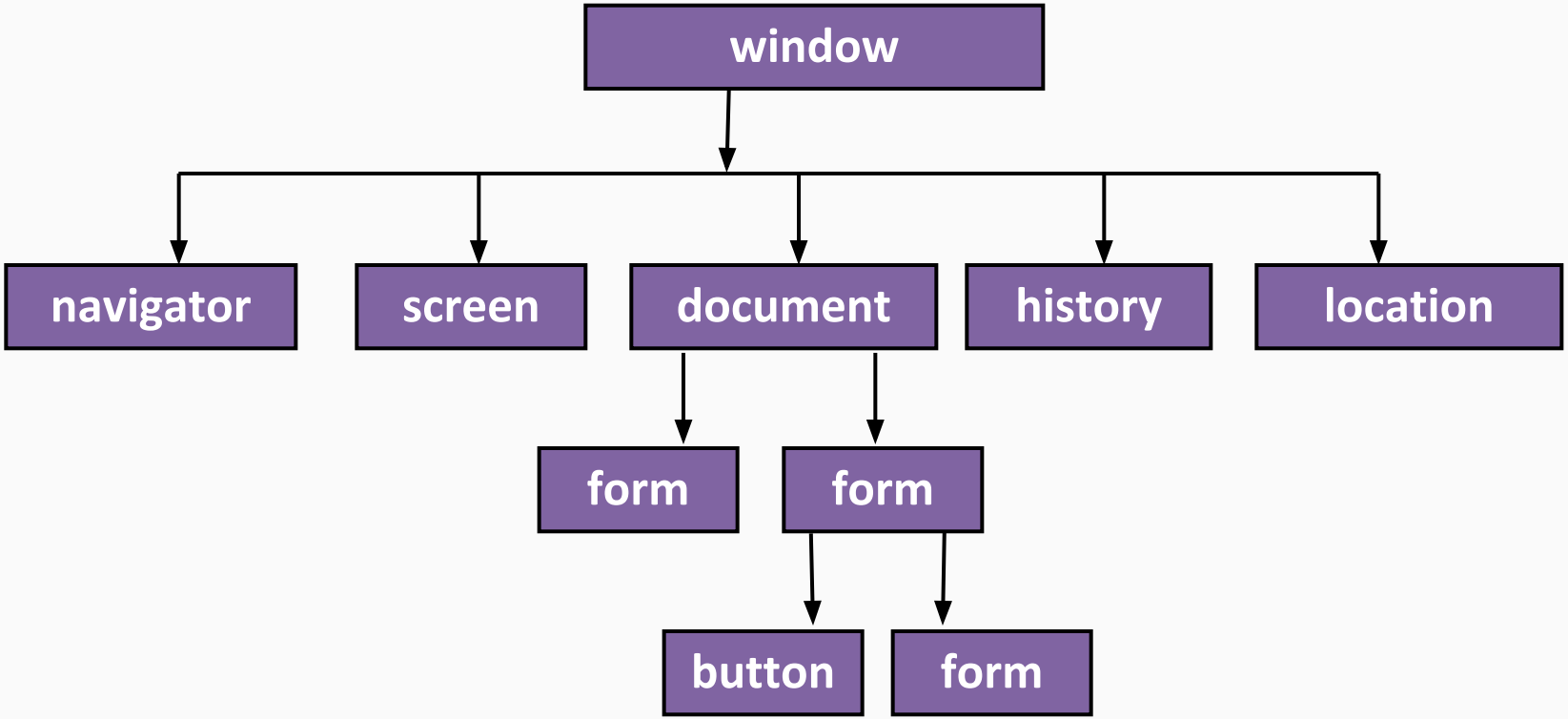
- `onload` event

```
<html>

<head>
  <script>
    function greet() {
      alert("Loaded.");
    }
  </script>
</head>

<body onload="greet()" >
</body>

</html>
```

Built-in Browser Objects

- The browser provides some read-only data via:
  - `window`
    - The top node of the DOM tree
    - Represents the browser's window
  - `document`
    - holds information the current loaded document
  - `screen`
    - Holds the user’s display properties
  - `browser`
    - Holds information about the browser

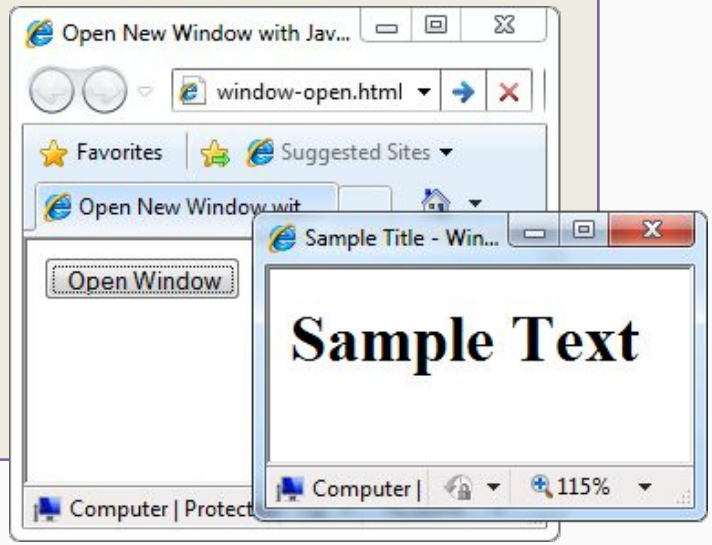
DOM Hierarchy - Example



Opening New Window - Example

- window.open()

```
var newWindow = window.open("", "sampleWindow",
  "width=300, height=100, menubar=yes,
  status=yes, resizable=yes");

newWindow.document.write(
  "<html><head><title>
  Sample Title</title>
  </head><body><h1>Sample
  Text</h1></body>");
newWindow.status = "Hello folks";
```



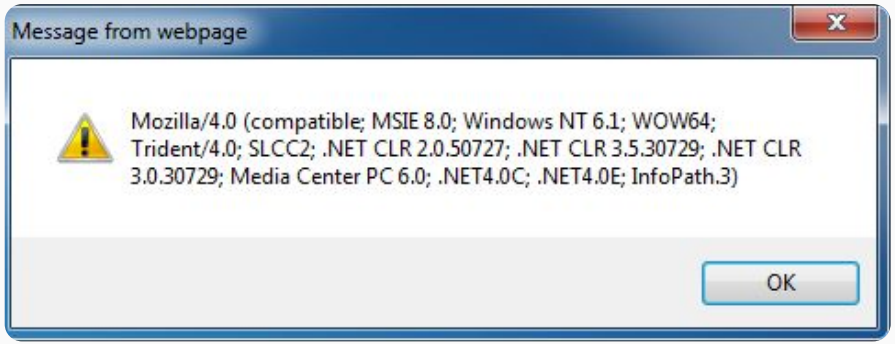
The Navigator Object

```
alert(window.navigator.userAgent);
```

The browser window

The navigator in the browser window

The userAgent (browser ID)



The Screen Object

- The `screen` object contains information about the display

```
window.moveTo(0, 0);
x = screen.availWidth;
y = screen.availHeight;
window.resizeTo(x, y);
```

Document and Location

- `document` object
  - Provides some built-in arrays of specific objects on the currently loaded Web page

```
document.links[0].href = "yahoo.com";
document.write("This is some <b>bold text</b>");
```

- `document.location`
  - Used to access the currently open URL or redirect the browser

```
document.location = "http://www.yahoo.com/";
```

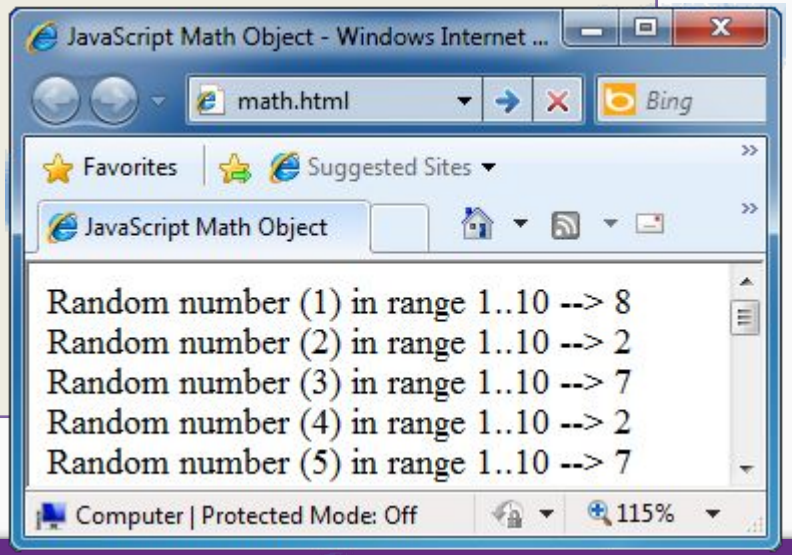
Form Validation - Example

```
function checkForm()
{
  var valid = true;
  if (document.getElementById("firstName").value == "") {
    alert("Please type in your first name!");
    document.getElementById("firstNameError").style.display = "inline";
    valid = false;
  }
  return valid;
}
...
<form name="mainForm" onsubmit="return checkForm()">
  <input type="text" id="firstName" />
  <div id="firstNameError" style="display: none"></div>
</form>
```

The Math Object

- The `Math` object provides some mathematical functions

```
for (i = 1; i <= 20; i++) {
  var x = Math.random();
  x = 10 * x + 1;
  x = Math.floor(x);
  document.write(
    "Random number (" + i + ")
    in range " + "1..10 --> "
    + x + "<br/>");
}
```



The Date Object

- The `Date` object provides date / calendar functions

```
var now = new Date();
var result = "It is now " + now;
document.getElementById("timeField").innerHTML =
  result;
...
<p id="timeField"></p>
```

Timers: setTimeout()

- Make something happen (once) after a fixed delay

```
var timer = setTimeout('bang()', 5000);
```

5 seconds after this statement executes, this function is called

```
clearTimeout(timer);
```

Cancels this timer

Timers: setInterval()

- Make something happen repeatedly at fixed intervals

```
var timer = setInterval('clock()', 1000);
```

This function is called continuosly per 1 second

```
clearInterval(timer);
```

Stop the timer

Timer- Example

```
<script>
  function timerFunc() {
    var now = new Date();
    var hour = now.getHours();
    var min = now.getMinutes();
    var sec = now.getSeconds();
    document.getElementById("clock").value =
      "" + hour + ":" + min + ":" + sec;
  }

  setInterval('timerFunc()', 1000);
</script>

<input type="text" id="clock" />
```

Debugging JavaScript

Debugging JavaScript

- Modern browsers have JavaScript console where errors in scripts are reported
  - Errors may differ across browsers
- Several tools to debug JavaScript
  - Firebug
  - Chrome Inspector
  - Microsoft Developer Tool
- Supports breakpoints, watches
- Console editor, CSS Editor, show AJAX requests and responses

JavaScript Console Object

- The `console` object exists only if there is a debugging tool that supports it
  - Used to write log messages at runtime
- Methods of the `console` object:
  - `debug`, `info`, `log`, `warn`, `error`

```
<script>
  var x = prompt("Get x value: ");
  if (x >= 0) {
    console.log("x is positive!");
  }
  else {
    console.log("x is negative!");
  }
</script>
```