JavaScript

Dynamic Behavior in a Web Page

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

JavaScript Advantages

- JavaScript allows interactivity such as:
  - Implementing form validation
  - React to user actions, e.g. handle keys
  - Changing an image on moving mouse over it
  - Sections of a page appearing and disappearing
  - Content loading and changing dynamically
  - Performing complex calculations
  - Custom HTML controls, e.g. scrollable table
  - Implementing AJAX functionality

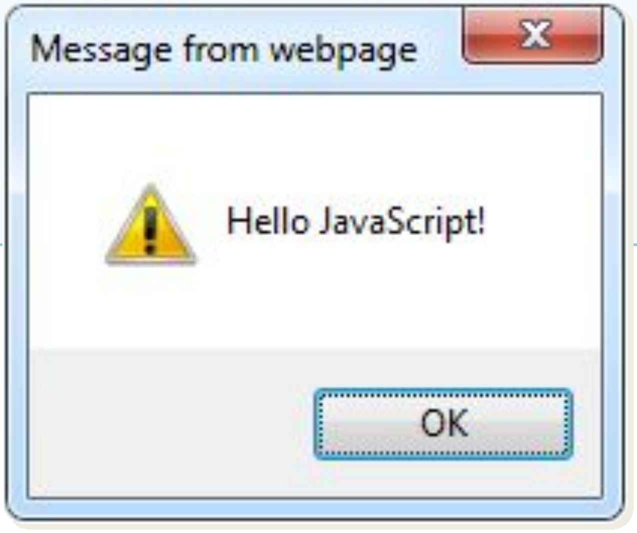
What Can JavaScript Do?

- 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

first-script.html

```
<html>

<body>
  <script type="text/javascript">
     alert('Hello JavaScript!');
  </script>
</body>

</html>
```

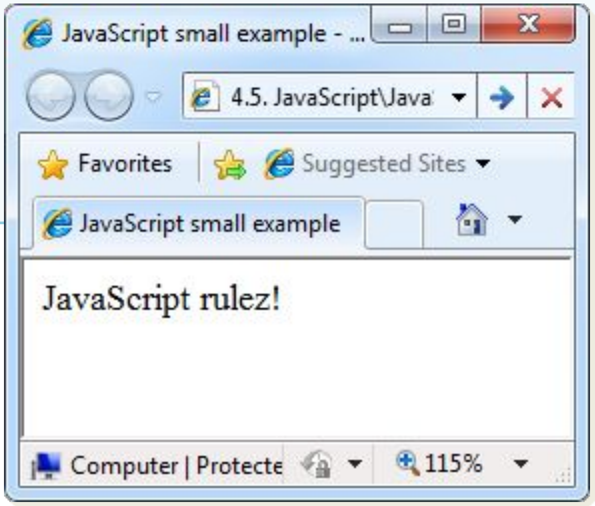

Message from webpage

Hello JavaScript!

Another Small Example

small-example.html

```
<html>

<body>
  <script type="text/javascript">
     document.write('JavaScript rulez!');
  </script>
</body>

</html>
```



JavaScript rulez!

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" type="text/javscript">
<!– code placed here will not be executed! -->
</script>
```

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

JavaScript – When is Executed?

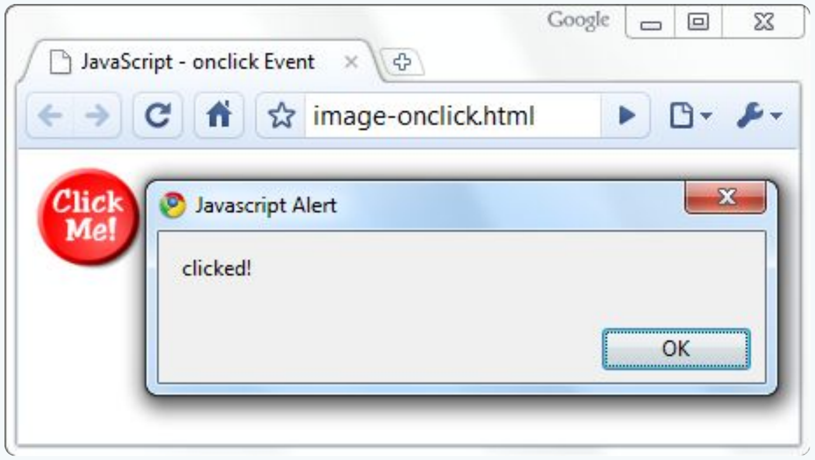
- JavaScript code is executed during the page loading or when the browser fires an event
  - All statements are executed at page loading
  - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
  - Executed when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

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

Calling a JavaScript Function from Event Handler – Example

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>
```



clicked!

Using External Script Files

- Using external script files:

external-JavaScript.html

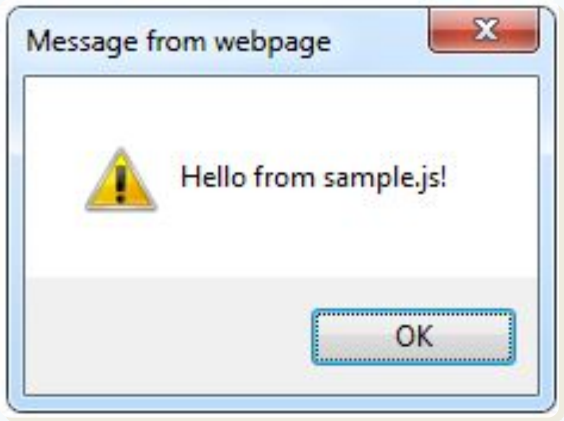
```
<html>
<head>
  <script src="sample.js" type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

The <script> tag is always empty.

- External JavaScript file:

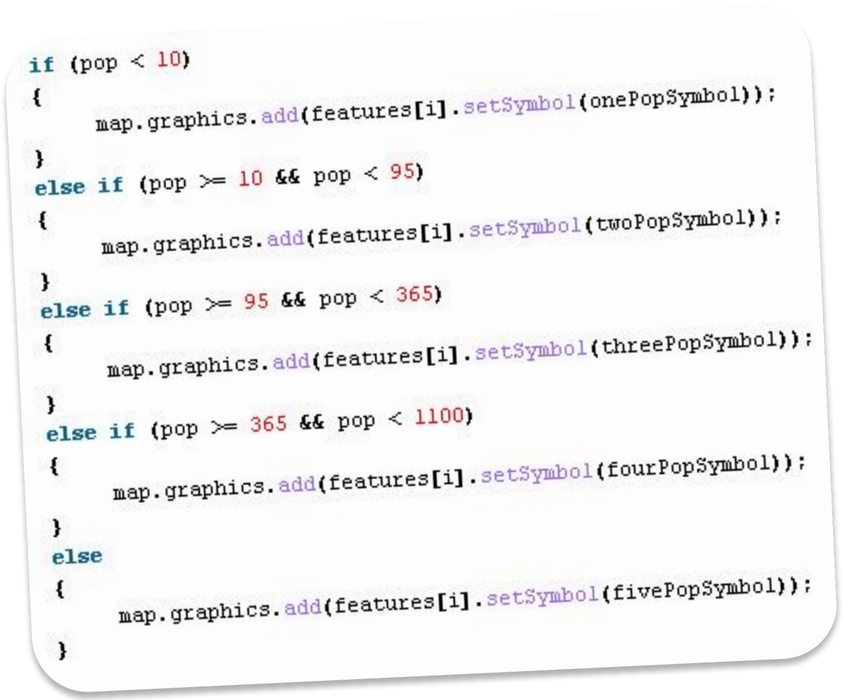
sample.js

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

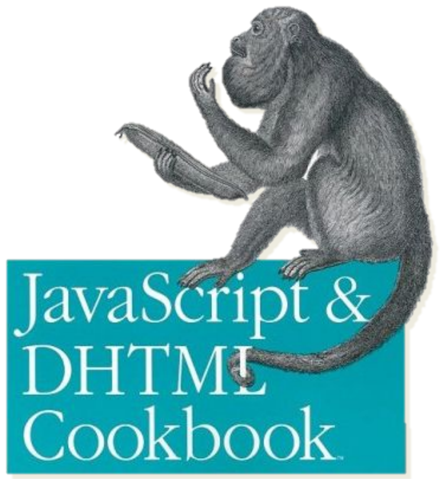


Hello from sample.js!

The JavaScript Syntax



```
if (pop < 10)
{
     map.graphics.add(features[i].setSymbol(onePopSymbol));
}
else if (pop >= 10 && pop < 95)
{
     map.graphics.add(features[i].setSymbol(twoPopSymbol));
}
else if (pop >= 95 && pop < 365)
{
     map.graphics.add(features[i].setSymbol(threePopSymbol));
}
else if (pop >= 365 && pop < 1100)
{
     map.graphics.add(features[i].setSymbol(fourPopSymbol));
}
else
{
     map.graphics.add(features[i].setSymbol(fivePopSymbol));
}
```



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)
  - Function variables (like the C# delegates)

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

```
var my_array = [1, 5.3, "aaa"];
```

- Associative arrays (hash tables)

```
var my_hash = {a:2, b:3, c:"text"};
```

Everything is Object

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

objects.html

```
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 7
```

String Operations

- The + operator joins strings

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

- What is "9" + 9?

```
alert("9" + 9);  // 99
```

- Converting string to number:

```
alert(parseInt("9") + 9);  // 18
```

Arrays Operations and Properties

- Declaring new empty array:

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

- Declaring an array holding few elements:

```
var arr = [1, 2, 3, 4, 5];
```

- Appending an element / getting the last element:

```
arr.push(3);
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

- Finding element's index in the array:

```
arr.indexOf(1);
```

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 – Example

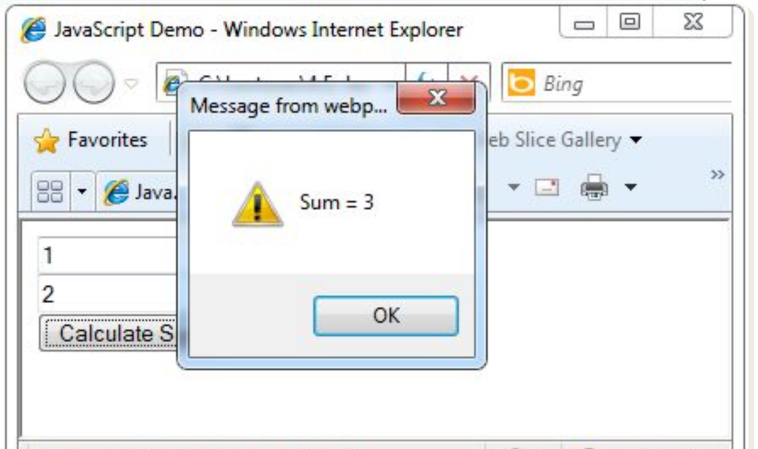
sum-of-numbers.html

```
<html>

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

Sum of Numbers – Example (2)

sum-of-numbers.html (cont.)

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

</html>
```


JavaScript Demo - Windows Internet Explorer

Message from webp...

Sum = 3

OK

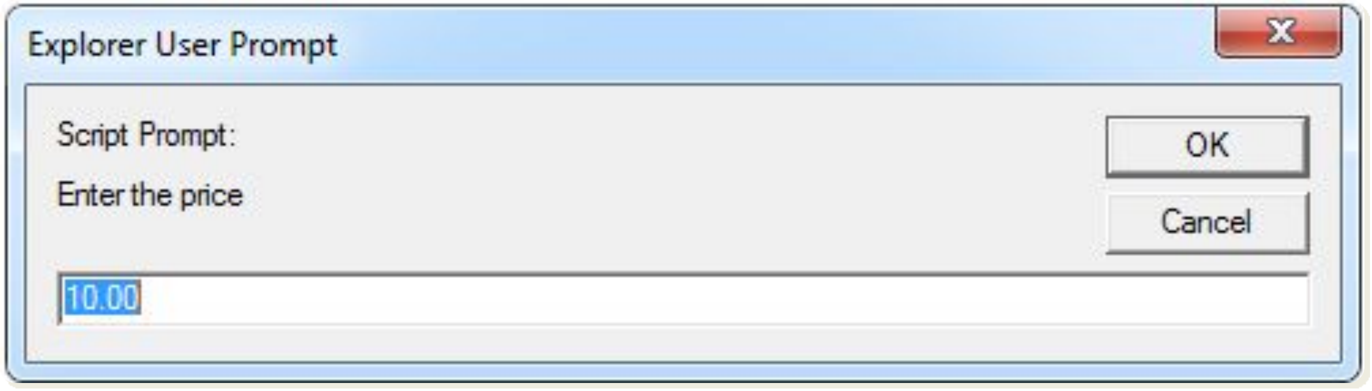
1

2

JavaScript Prompt – Example

prompt.html

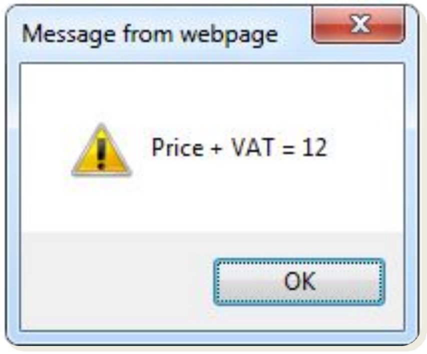
```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.2);
```



Script Prompt:

Enter the price

10.00



Price + VAT = 12

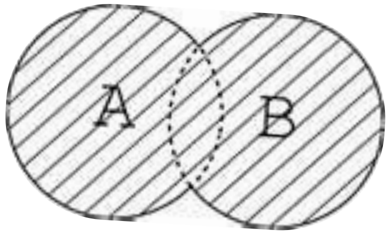
Conditional Statement (`if`)

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

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

Conditional Statement (`if`) (2)

- The condition may be of Boolean or integer type:



conditional-statements.html

```
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

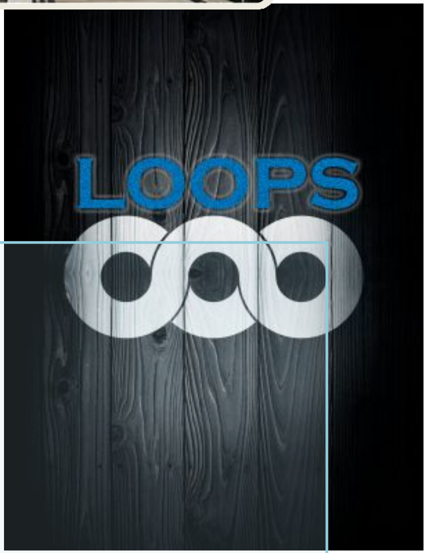
- The `switch` statement works like in C#:

switch-statements.html

```
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;
for (counter=0; counter<4; counter++) {
  alert(counter);
}
while (counter < 5) {
  alert(++counter);
}
```

loops.html

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.

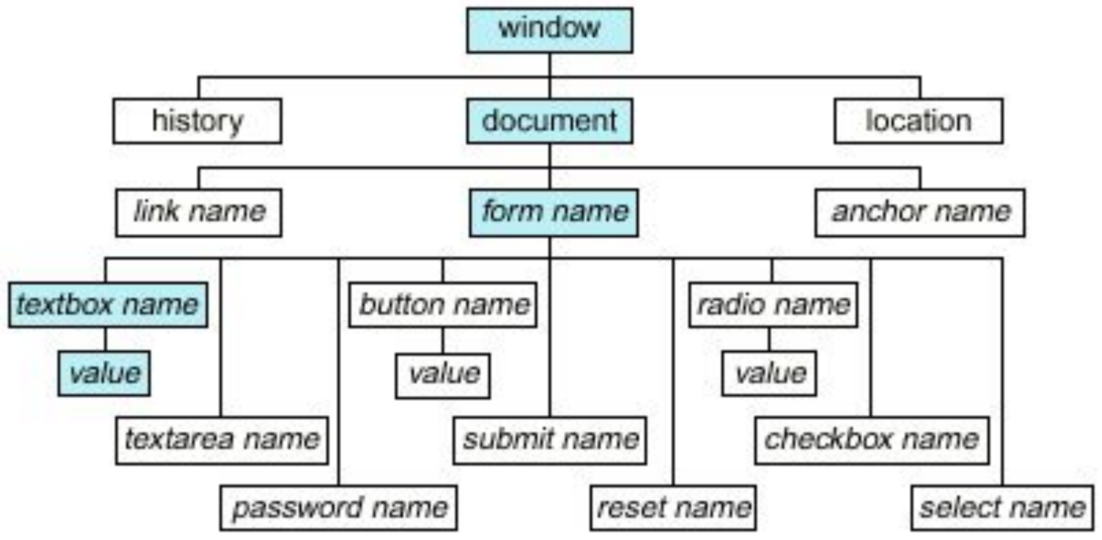
Value returned here.

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));
```

functions-demo.html



The JavaScript Object Model

Document Object Model (DOM)

Document Object Model (DOM)

- 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

Accessing Elements

- Access elements via their ID attribute

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

- Via the name attribute

```
var arr = document.getElementsByName("some_name")
```

- Via tag name

```
var imgTags = el.getElementsByTagName("img")
```

  – Returns array of descendant `<img>` elements of the element "`el`"

DOM Manipulation

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

DOM-manipulation.html

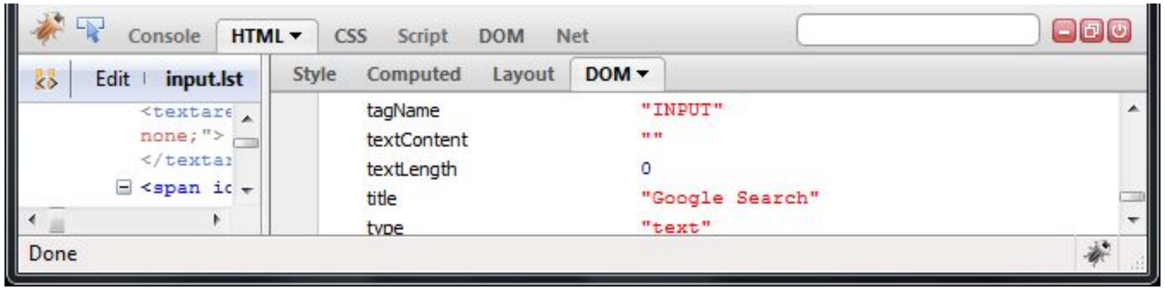
```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".png";
  var statusDiv =
    document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state";
}
…
<img src="test_on.gif" onmouseover="change('off')"
  onmouseout="change('on')"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

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`

Common Element Properties (2)

- `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…



```
<textare
none;">
</textar
<span ic
```

```
tagName       "INPUT"
textContent   ""
textLength    0
title         "Google Search"
type          "text"
```

Accessing Elements through the DOM Tree Structure

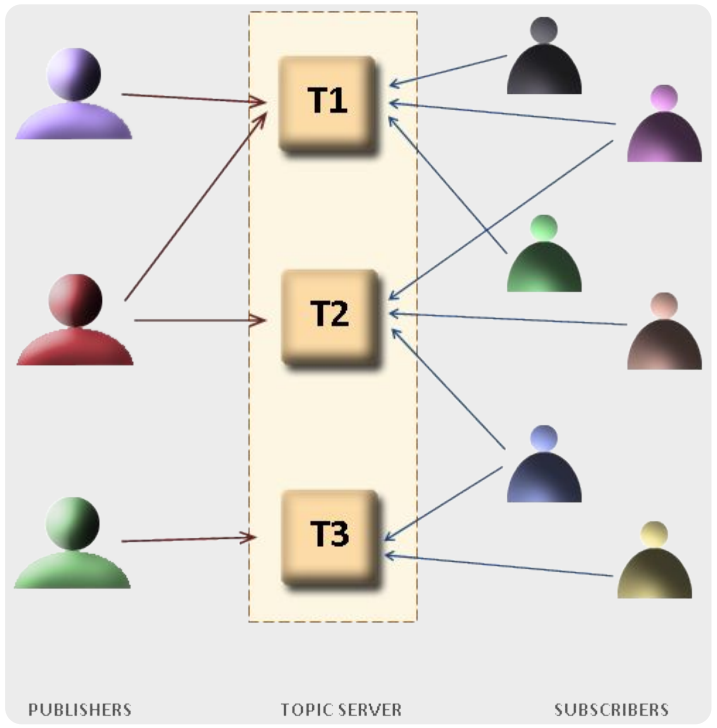
- We can access elements in the DOM through some tree manipulation properties:
  - `element.childNodes`
  - `element.parentNode`
  - `element.nextSibling`
  - `element.previousSibling`
  - `element.firstChild`
  - `element.lastChild`

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>
```

accessing-elements-demo.html

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



The HTML DOM Event Model

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;
```

The HTML DOM Event Model (2)

- 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

The HTML DOM Event Model (3)

– Holds information about the state of [Alt], [Ctrl] and [Shift] keys
– Some browsers do not send this object, but place it in the `document.event`
– Some of the names of the event's object properties are browser-specific

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 (2)

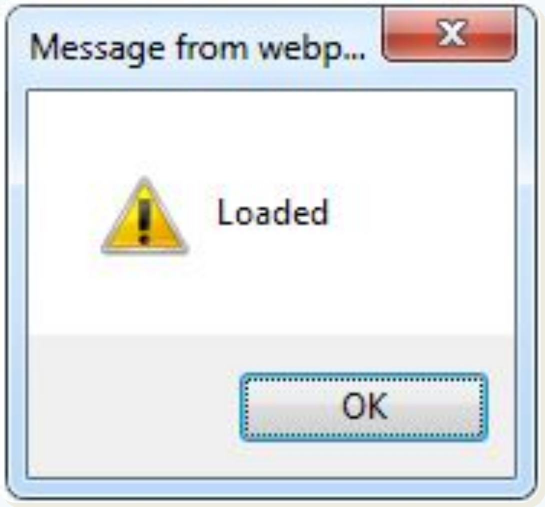
- Form events
  - `onchange` – for input fields
  - `onsubmit`
    - Allows you to cancel a form submission
    - Useful for form validation
- Miscellaneous events
  - `onload`, `onunload`
    - Allowed only for the `<body>` element
    - Fires when all content on the page was loaded / unloaded

onload Event – Example

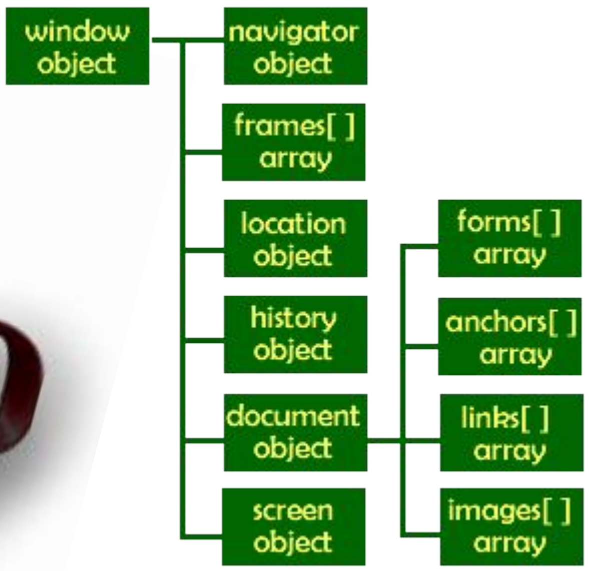
- onload event

onload.html

```
<html>

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

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

</html>
```


Message from webp...

Loaded

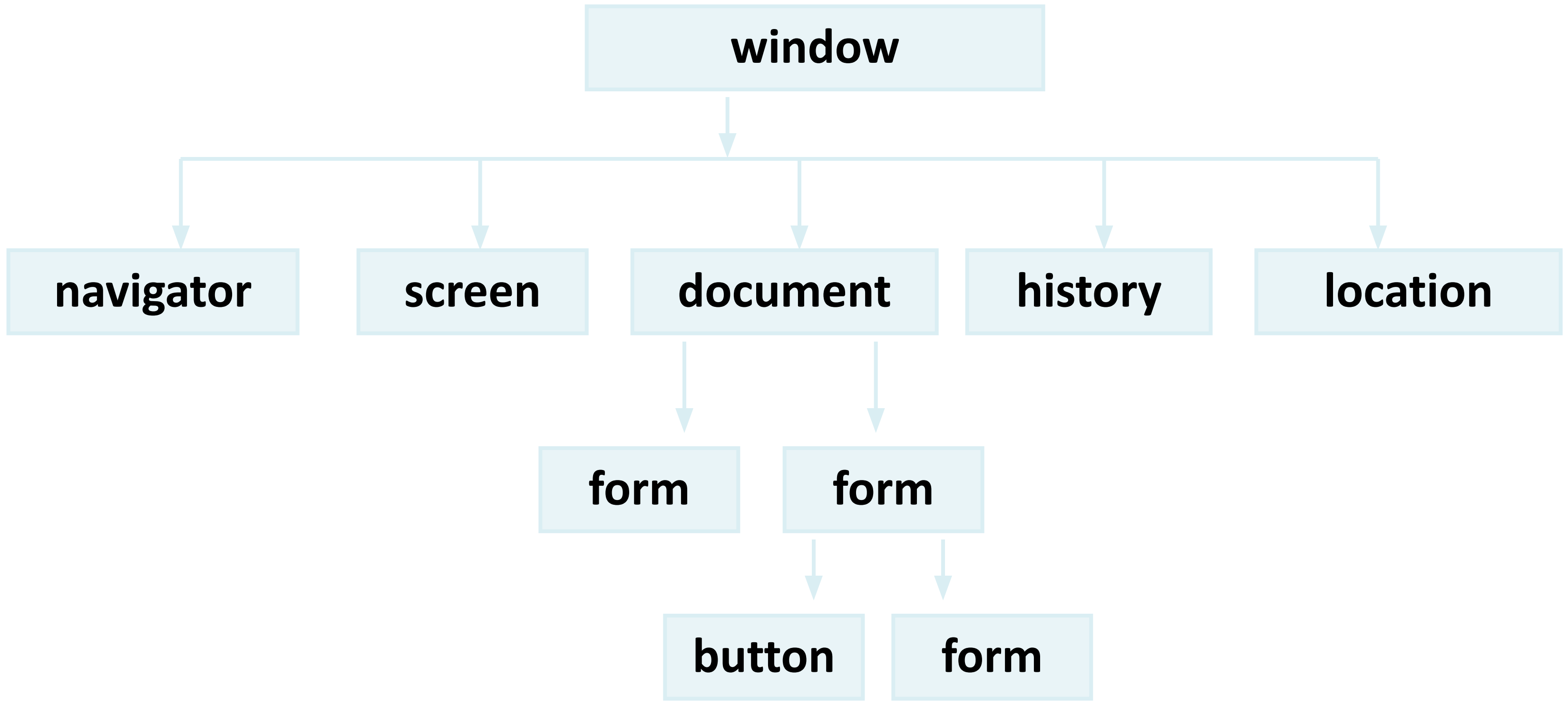


The Built-In Browser Objects

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

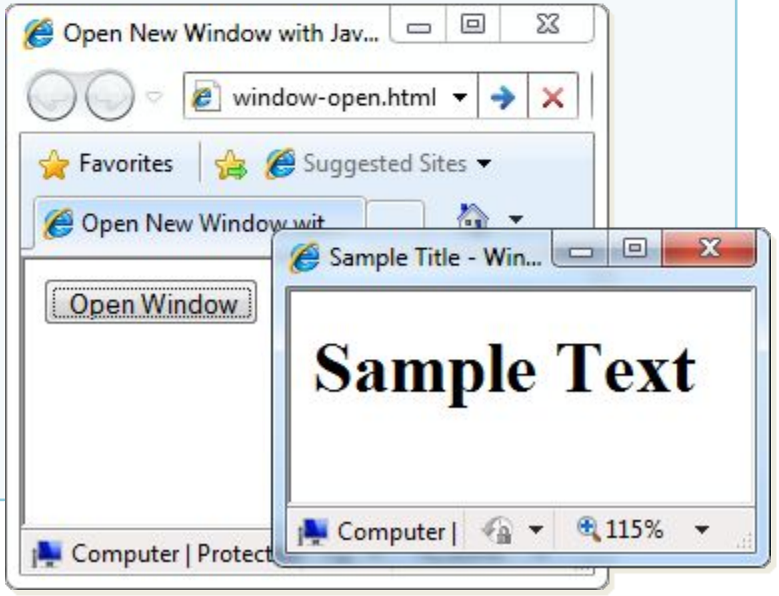


Opening New Window – Example

- `window.open()`

window-open.html

```
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";
```



Sample Text

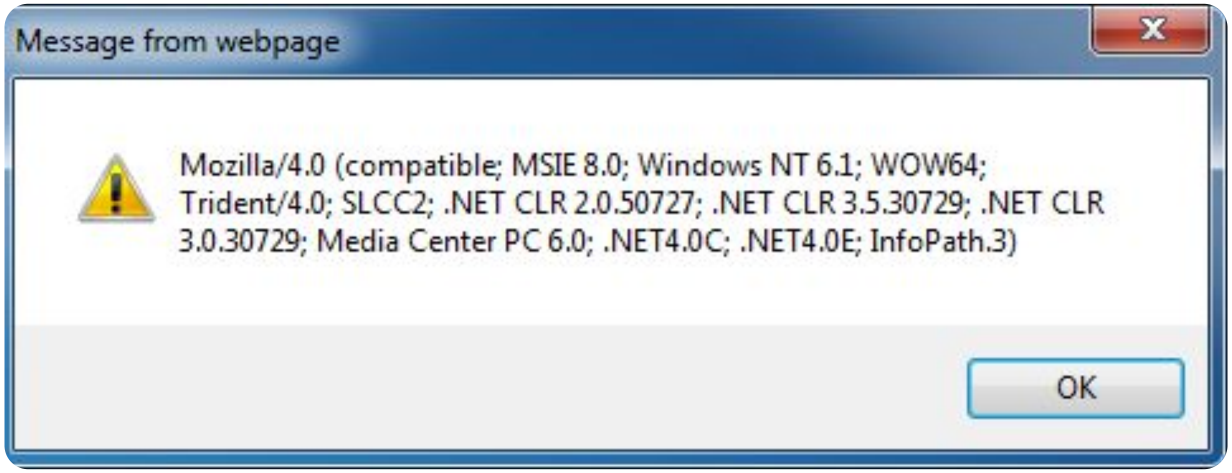
The Navigator Object

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

The browser window

The navigator in the browser window

The userAgent (browser ID)



```
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3)
```

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

form-validation.html

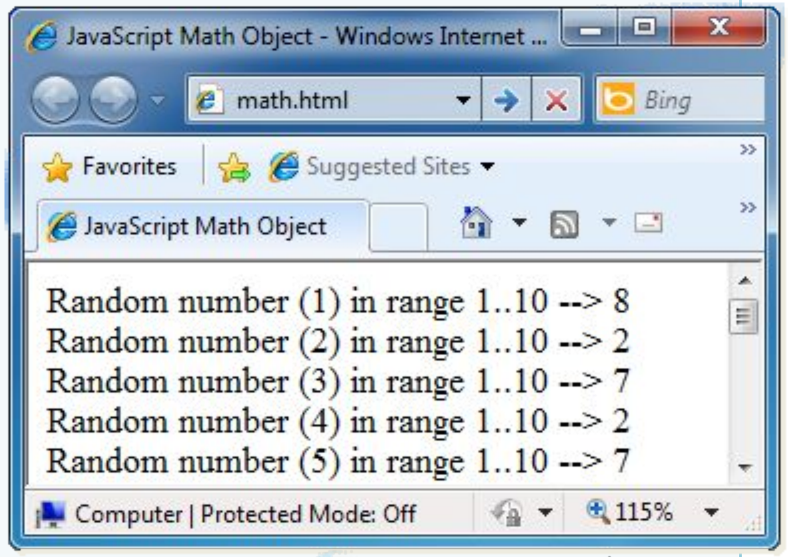
```
function checkForm()
{
  var valid = true;
  if (document.mainForm.firstName.value == "") {
    alert("Please type in your first name!");
    document.getElementById("firstNameError").
      style.display = "inline";
    valid = false;
  }
  return valid;
}
…
<form name="mainForm" onsubmit="return checkForm()">
  <input type="text" name="firstName" />
  …
</form>
```

The Math Object

- The Math object provides some mathematical functions

math.html

```
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/>");
}
```


JavaScript Math Object - Windows Internet ...
math.html
Bing
Favorites
Suggested Sites
JavaScript Math Object
Computer | Protected Mode: Off
115%

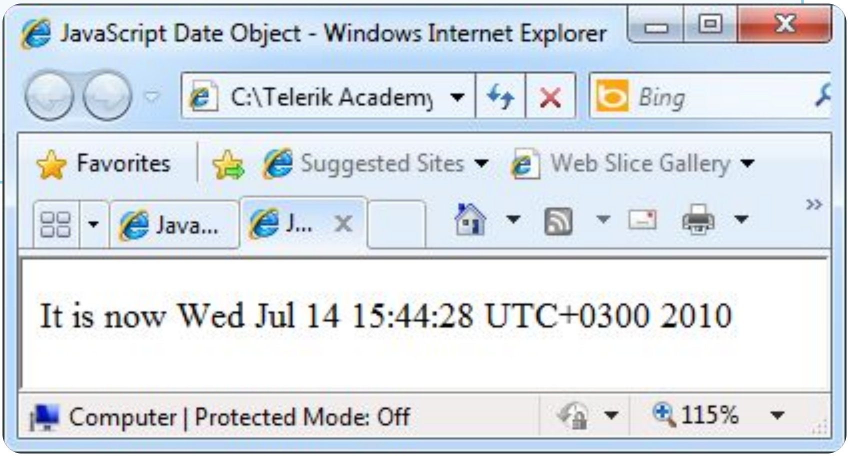
```
Random number (1) in range 1..10 --> 8
Random number (2) in range 1..10 --> 2
Random number (3) in range 1..10 --> 7
Random number (4) in range 1..10 --> 2
Random number (5) in range 1..10 --> 7
```

The Date Object

- The Date object provides date / calendar functions

dates.html

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


JavaScript Date Object - Windows Internet Explorer
C:\Telerik Academy
Bing
Favorites
Suggested Sites
Web Slice Gallery
Java...
J...
Computer | Protected Mode: Off
115%

It is now Wed Jul 14 15:44:28 UTC+0300 2010

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 the timer

Timers: `setInterval()`

- Make something happen repeatedly at fixed intervals

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

This function is called continuously per 1 second.

```
clearInterval(timer);
```

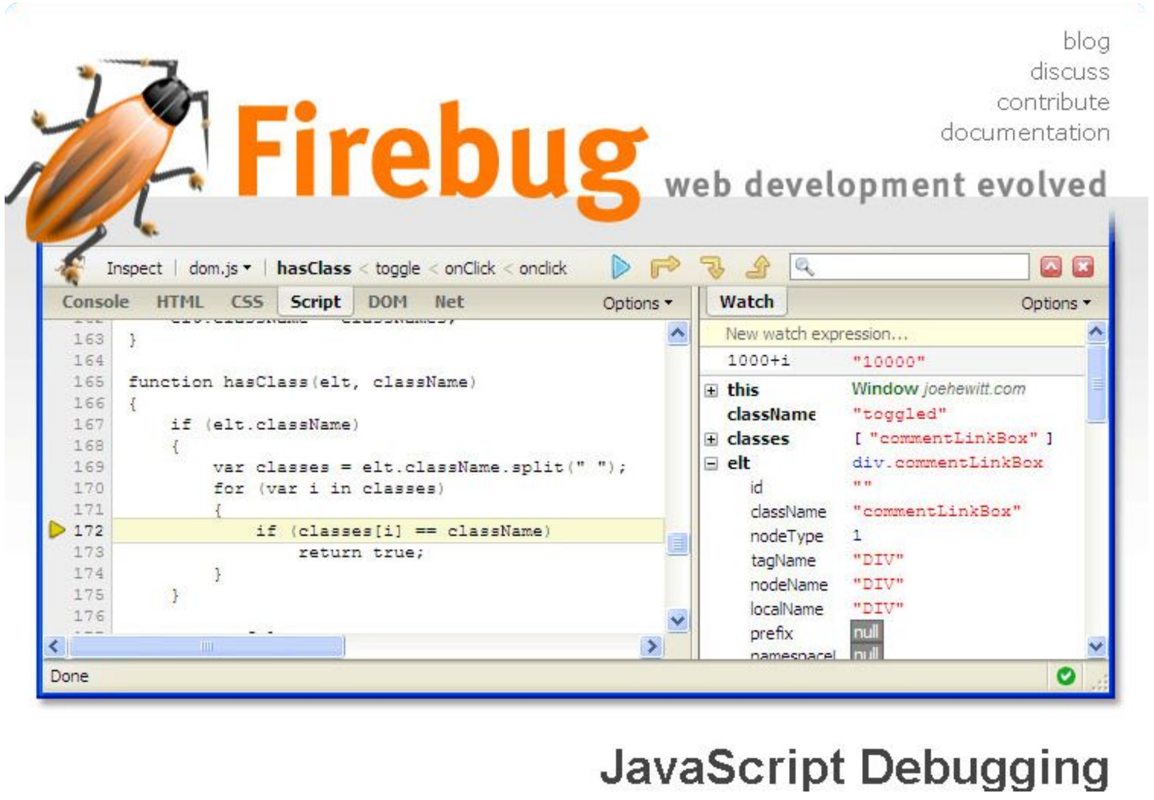
Stop the timer.

Timer – Example

timer-demo.html

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



```
}

function hasClass(elt, className)
{
    if (elt.className)
    {
        var classes = elt.className.split(" ");
        for (var i in classes)
        {
            if (classes[i] == className)
                return true;
        }
    }
```

```
1000+i          "10000"
+ this          Window joehewitt.com
  className     "toggled"
+ classes       [ "commentLinkBox" ]
- elt           div.commentLinkBox
    id          ""
    className   "commentLinkBox"
    nodeType    1
    tagName     "DIV"
    nodeName    "DIV"
    localName   "DIV"
    prefix      null
```

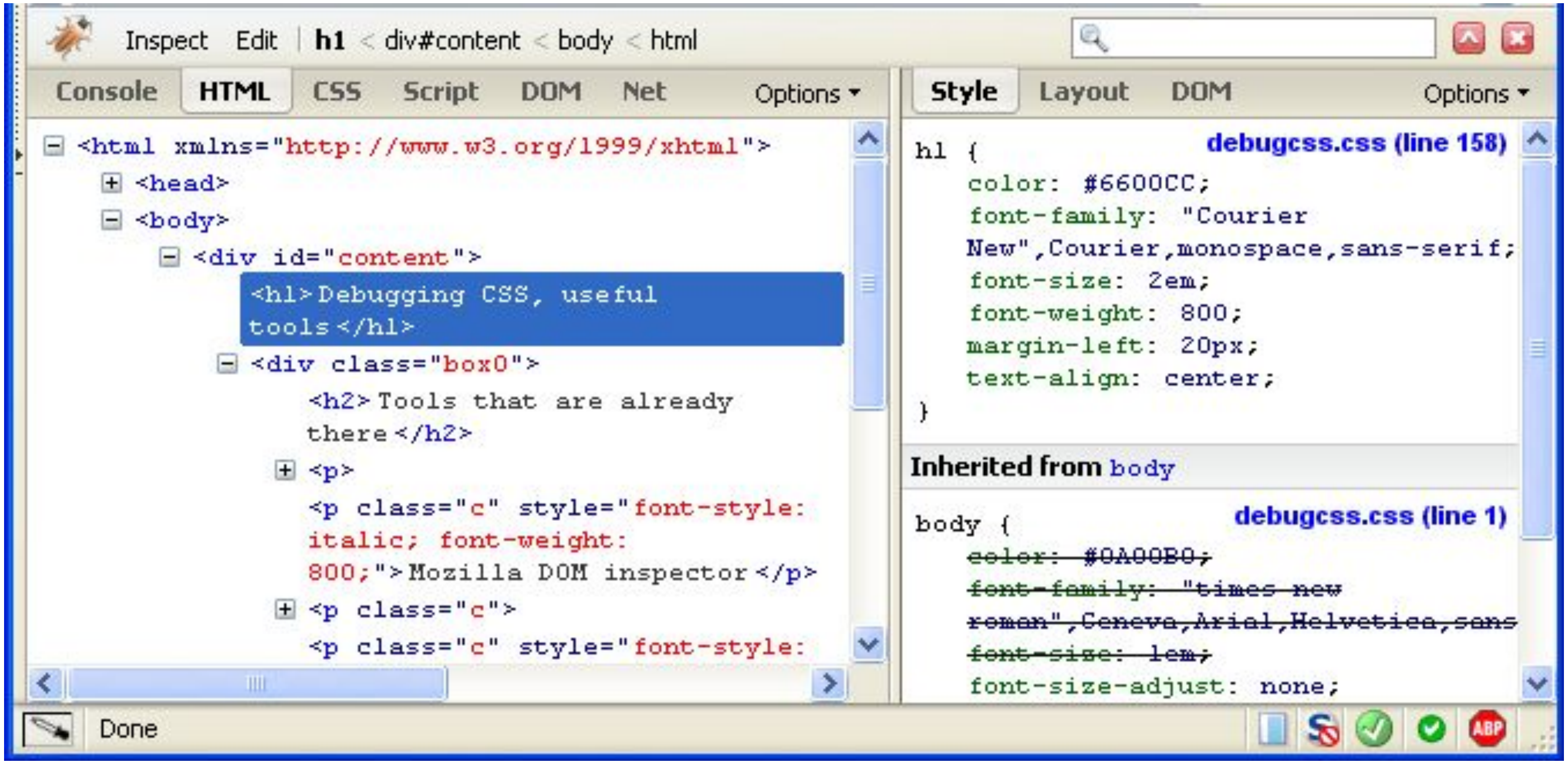
Debugging JavaScript

- Modern browsers have JavaScript console where errors in scripts are reported
  - Errors may differ across browsers
- Several tools to debug JavaScript
  - Microsoft Script Editor
    - Add-on for Internet Explorer
    - Supports breakpoints, watches
    - JavaScript statement `debugger`; opens the script editor

Firebug

- Firebug – Firefox add-on for debugging JavaScript, CSS, HTML
  - Supports breakpoints, watches, JavaScript console editor
  - Very useful for CSS and HTML too
    - You can edit all the document real-time: CSS, HTML, etc
    - Shows how CSS rules apply to element
  - Shows Ajax requests and responses
  - Firebug is written mostly in JavaScript

Firebug (2)



```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <body>
    <div id="content">
        <h1>Debugging CSS, useful
        tools</h1>
      <div class="box0">
          <h2>Tools that are already
          there</h2>
        <p>
          <p class="c" style="font-style:
          italic; font-weight:
          800;">Mozilla DOM inspector</p>
        <p class="c">
          <p class="c" style="font-style:
```

```
h1 {                                  debugcss.css (line 158)
    color: #6600CC;
    font-family: "Courier
    New",Courier,monospace,sans-serif;
    font-size: 2em;
    font-weight: 800;
    margin-left: 20px;
    text-align: center;
}
```

Inherited from `body`

body { debugcss.css (line 1)
~~color: #0A00B0;~~
~~font-family: "times new roman",Geneva,Arial,Helvetica,sans~~
~~font-size: 1em;~~
font-size-adjust: none;

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(message)`
  - `info(message)`
  - `log(message)`
  - `warn(message)`
  - `error(message)`

HTML, CSS and JavaScript Basics

Questions?