telerik

Web Engineering

Muhammad Kamran

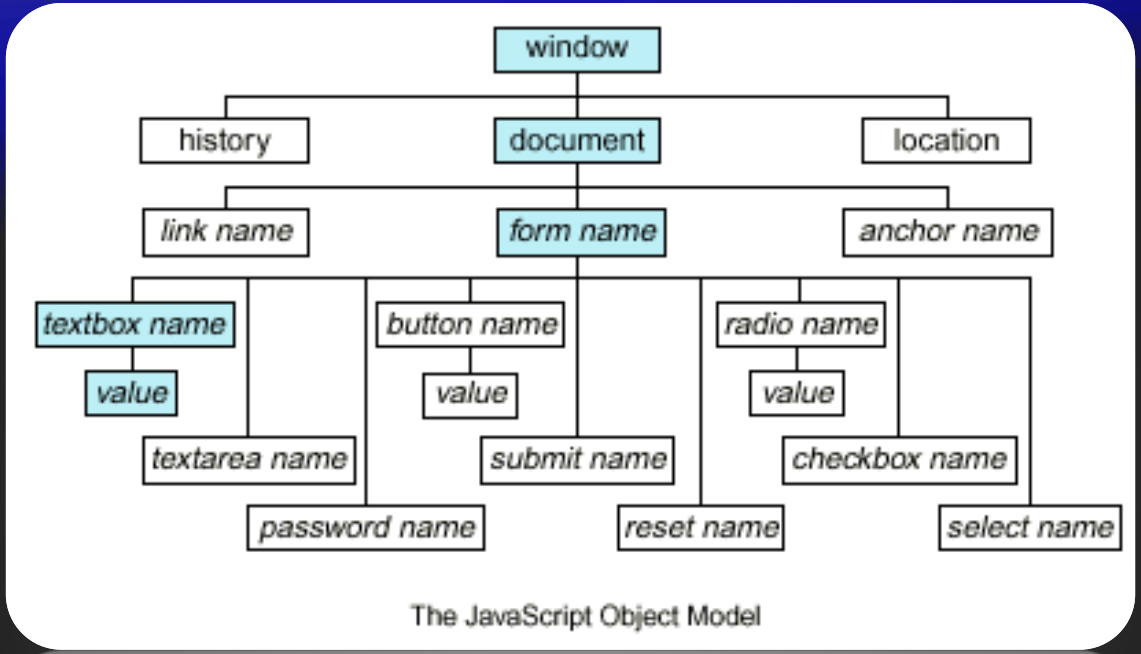
Lecture 16

telerik

Outline

- DOM (Document Object Model)
  - DOM Introduction
  - DOM Methods
  - DOM Document
  - DOM Events
  - DOM Nodes
- BOM (Browser Object Model)
  - BOM Display
  - BOM Location
  - BOM History
  - BOM Navigator
  - BOM Popup
  - BOM Timing
  - BOM Cookies

telerik



The JavaScript Object Model

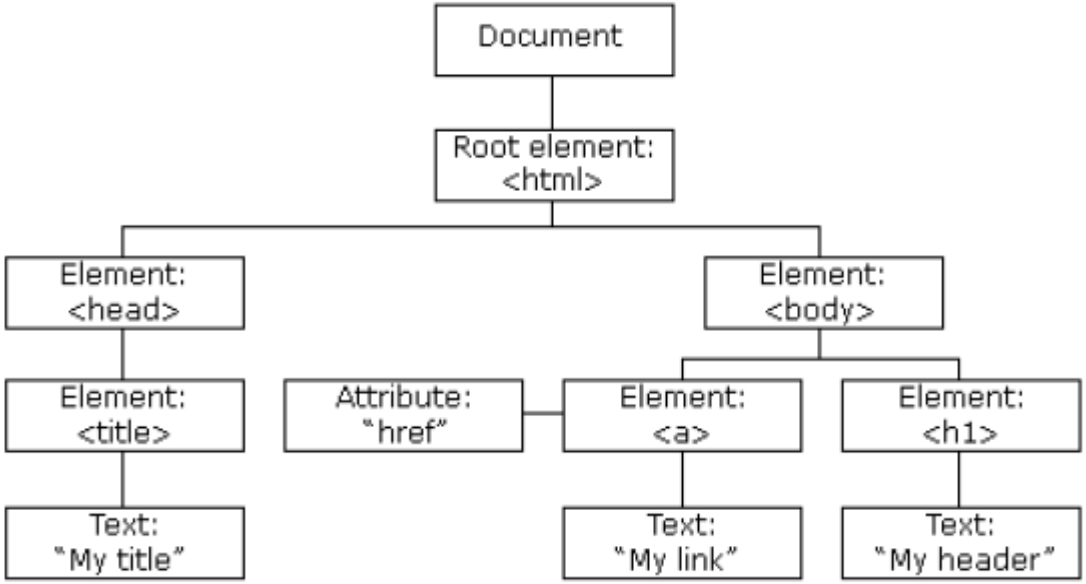
Document Object Model (DOM)

telerik

JavaScript HTML DOM

- **The HTML DOM (Document Object Model)**
  - When a web page is loaded, the browser creates a Document Object Model of the page.
  - The HTML DOM Tree of Objects

```
<html>
  <head>
    <title> My title</title>
  </head>
  <body>
     <a  href="" > My Link</a>
     <h1> My header</h1>
  </body>
</html>
```



Document Object Model (DOM)

JavaScript HTML DOM (cont'd)

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
  - JavaScript can change all the HTML elements in the page.
  - JavaScript can change all the HTML attributes in the page.
  - JavaScript can change all the CSS styles in the page.
  - JavaScript can remove existing HTML elements and attributes.
  - JavaScript can add new HTML elements and attributes
  - JavaScript can react to all existing HTML events in the page.
  - JavaScript can create new HTML events in the page.

telerik

What is the DOM?

- A W3C (World Wide Web Consortium) standard
- Defines a standard for accessing documents
  - *"The W3C Document Object Model(DOM) is platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- Separated into 3 different parts
  - Core DOM – standard model for all document types
  - XML DOM – standard model for XML documents
  - HTML DOM – standard model for HTML documents

telerik

What is the HTML DOM?

- A standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

telerik

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
  - A **property** is a value that you can get or set (like changing the content of an HTML element). (e.g., innerHTML)
  - A **method** is an action you can do (like add or deleting an HTML element). (e.g., getElementById)

telerik

Elements JavaScript HTML DOM Document

- The HTML DOM Document Object
  - The document object represents your web page.
  - The document object is the owner of all other objects in your web page.
  - If you want to access objects in an HTML page, you always start with accessing the document object.
- Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

telerik

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"

telerik

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')" />
```

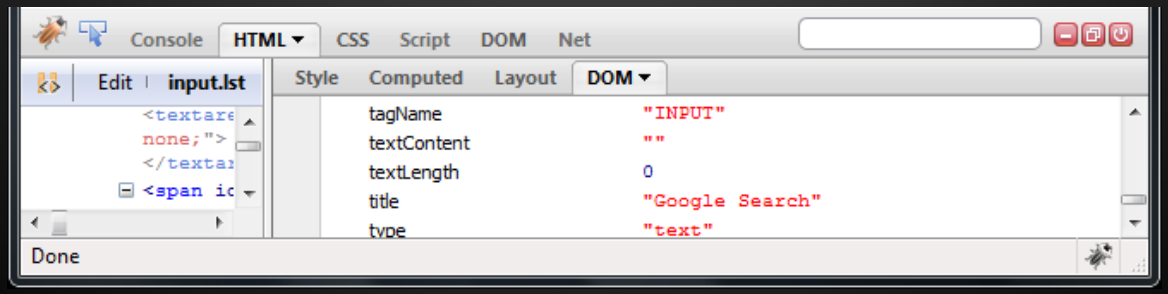
telerik

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…



telerik

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`

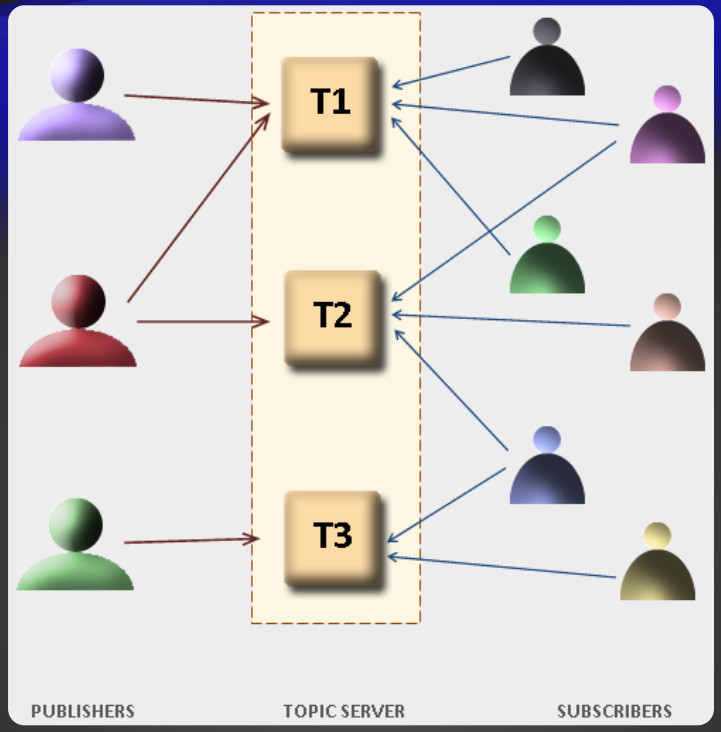
Accessing Elements through the DOM Tree – Example

```
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

telerik



The HTML DOM Event Model

telerik

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

telerik

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

telerik

Common DOM Events (2)

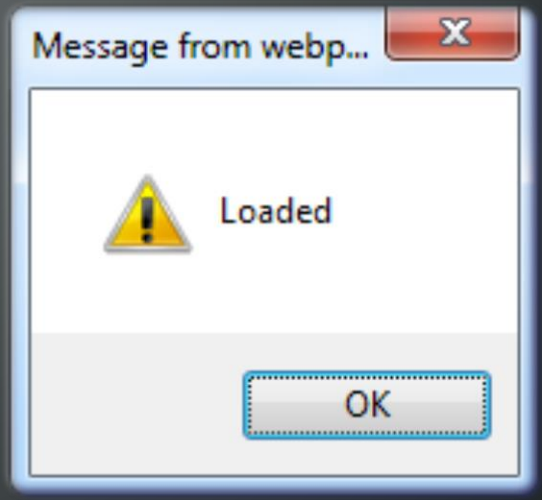
- Form events
  - onchange – for input fields
  - onsubmit
    - Allows you to cancel a form submission
    - Useful for form validation
- Miscellaneous events
  - onload, onunload
    - Allowed only for the <body> element
    - Fires when all content on the page was loaded / unloaded

onload Event – Example

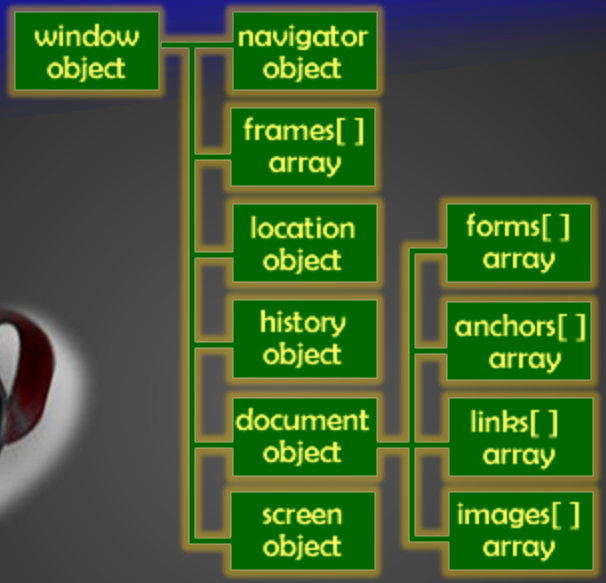
- onload event

onload.html

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



telerik



The Built-In Browser Objects

telerik

The Browser Object Model

- **Brower Object Model (BOM) allows JavaScript to "talk to" the browser.**
- **The Browser Object Model**
  - There are no official standards for the Browser Object Model.
  - Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

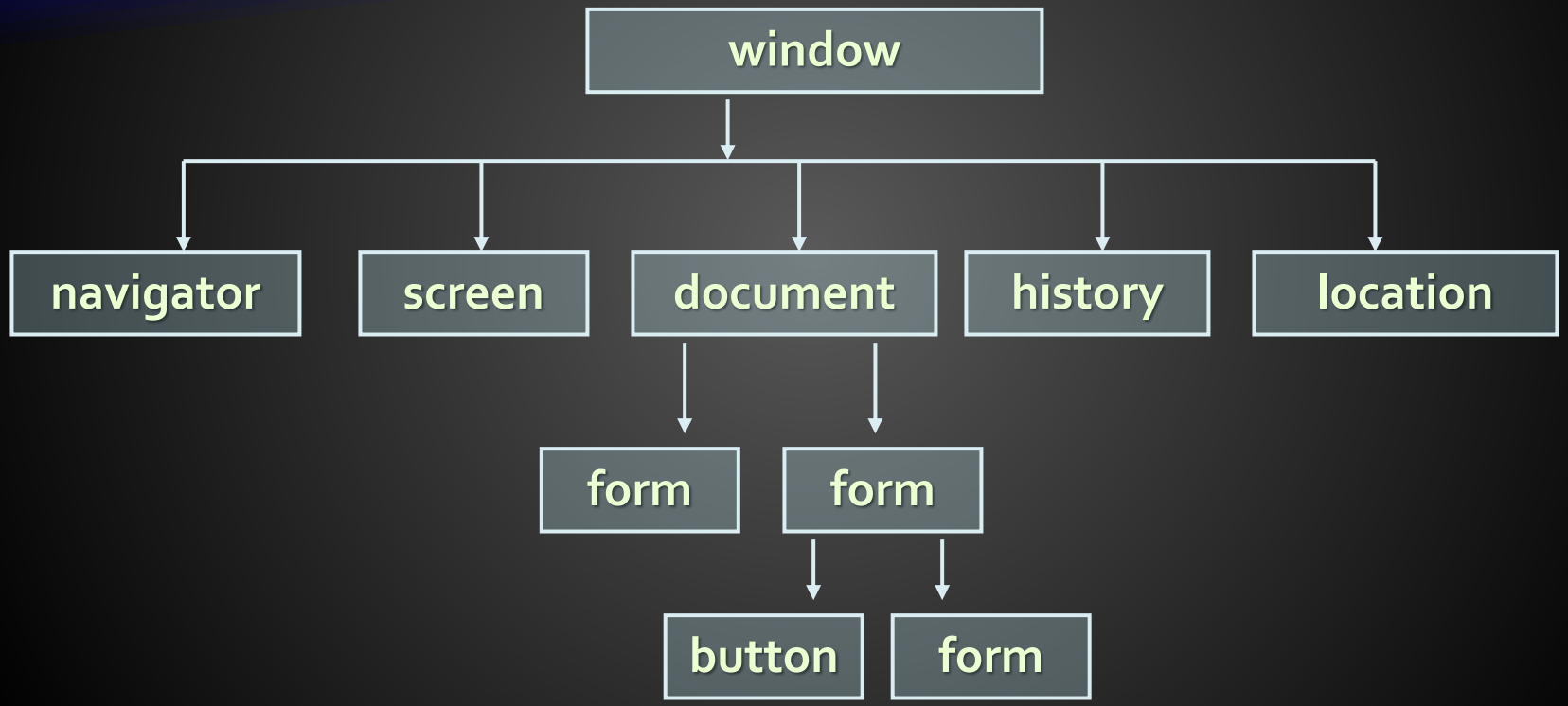
telerik

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

telerik

DOM Hierarchy – Example



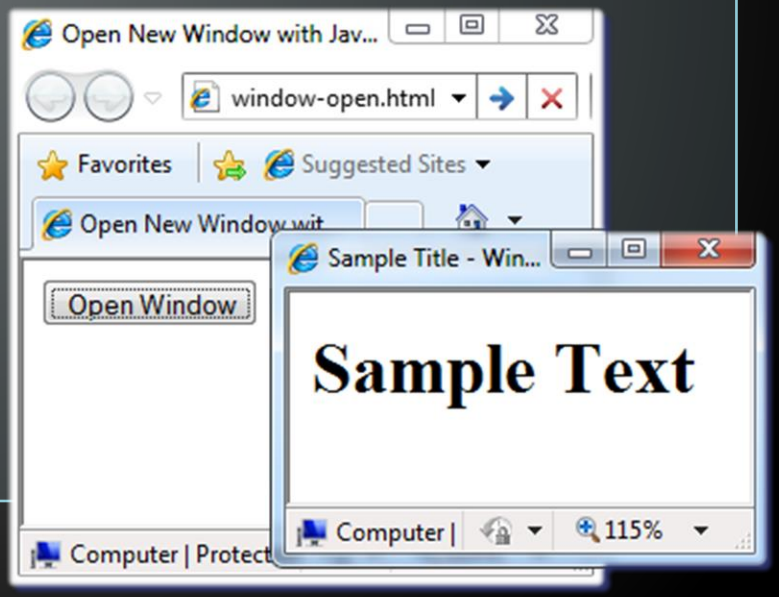
Opening New Window – Example

- 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";
```



telerik

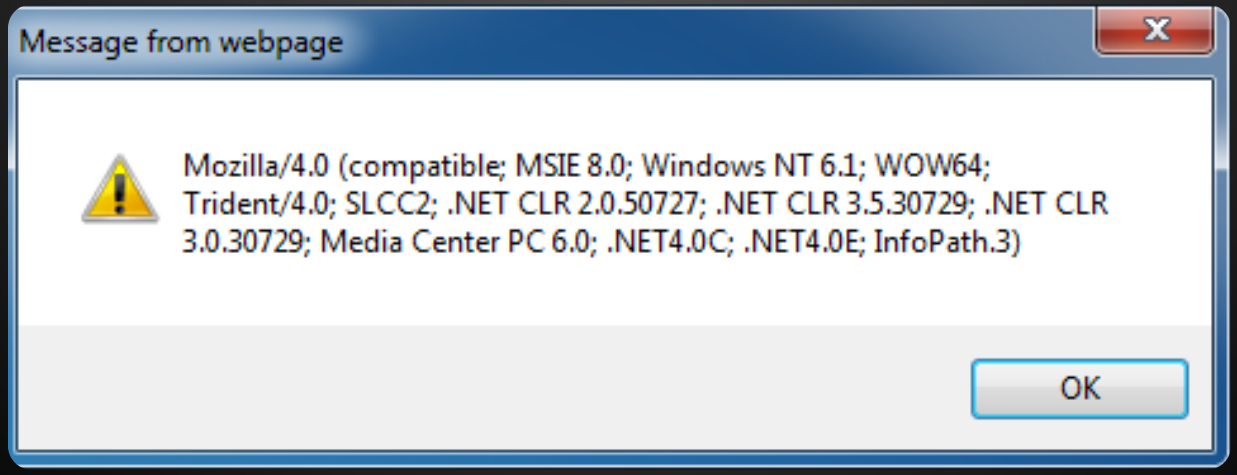
The Navigator Object

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

The browser window

The navigator in the browser window

The userAgent (browser ID)



telerik

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);
```

telerik

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/";
```

telerik

Form Validation – Example

form-validation.html

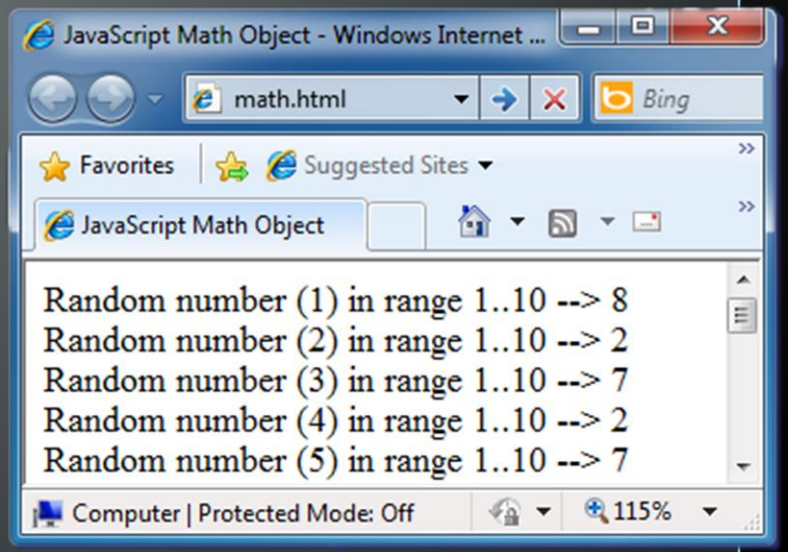
```
function checkForm()
{
  var valid = true;
  if (document.mainForm.firstName.value == "") {
    alert("Please type in your first name!");
    document.getElementById("firstNameError").
      style.display = "inline";
    valid = false;
  }
  return valid;
}
…
<form name="mainForm" onsubmit="return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

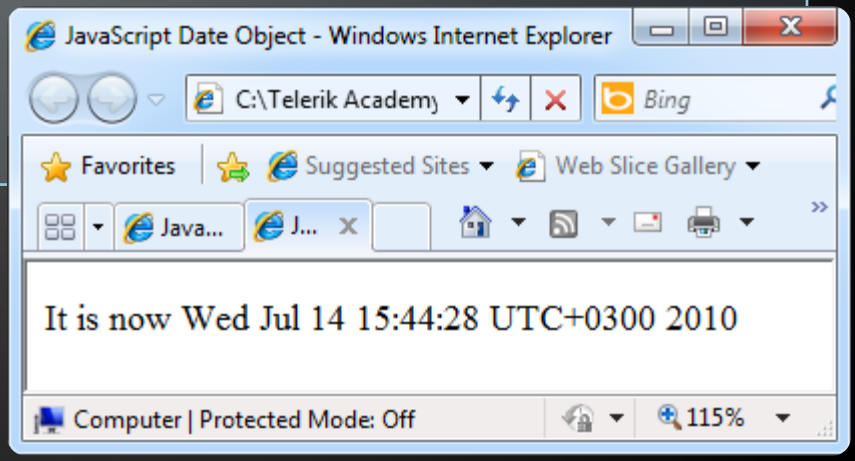


The Date Object

- 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>
```



telerik

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

telerik

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.

telerik

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" />
```

telerik

Debugging JavaScript

telerik

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

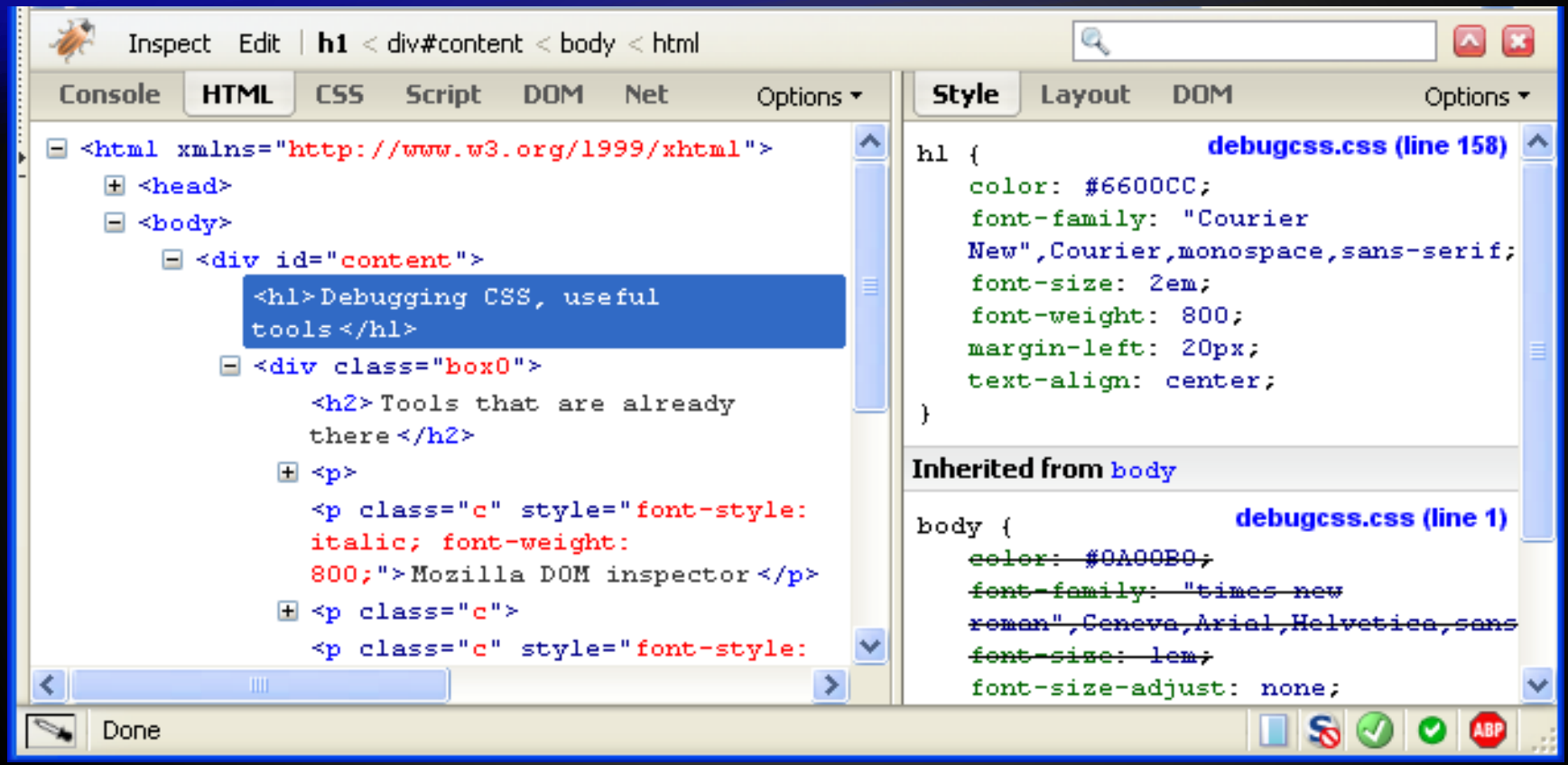
telerik

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

telerik

Firebug (2)



telerik

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)`

telerik

Introduction to JavaScript

telerik

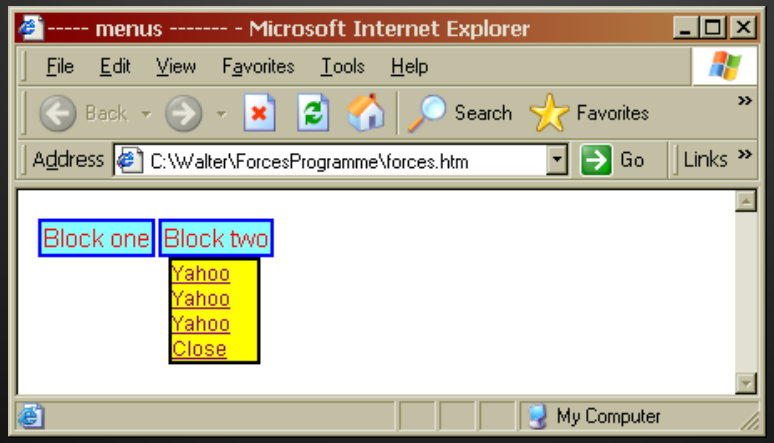
Assgnement#2 Due Date: 17-10-2017

1. Create an HTML page that has two text fields (first name and last name) and a button. When the user clicks the button, a message should show the text in the text fields followed by the current time.
2. Create a Web page that asks the user about his name and says goodbye to him when leaving the page.
3. Modify the previous HTML page to have a text field for email address and on clicking the button check if the email is valid (it should follow the format `<something>@<host>.<domain>`).
4. Create a Web page that shows 20 elements with random location, size and color.

telerik

Assgnement#2 Due Date: 17-10-2017

5. Create a drop-down menu
   - Use table for the main menu blocks
   - Use hidden <DIV> elements (`display: none; position:absolute; top:30px`)
   - Use `JavaScript` and `onmouseover` and `onmouseout` event to change display: none/block



telerik

Assgnement#2 Due Date: 17-10-2017

6. Create a DTHML page that has `<div>` containing a text that scrolls from right to left automatically
   - Use `setInterval()` function to move the text at an interval of 500 ms
   - Use `overflow:hidden` for the `<div>`
   - Use `scrollLeft` and `scrollWidth` properties of the `<div>` element