CS604, Spring 2005
Client-Side JavaScript

Lule Ahmedi

South East European University

Communication Sciences and Technologies

Administrativia

- The midterm exam will take place on Wednesday May 11th
- Today the announcement of the new assignemt; the due date is on Tuesday at 12:00 (noon)

The Syllabus' Current State

- Introduction to the client/server computing paradigm
- The HTML standard
- Cascading Stylesheets (CSS)
- The scripting language JavaScript
- The docuement object model (DOM)
- An in-depth coverage of PHP programming techniques
- The MySQL backend databases and PHP
- Security aspects in client/server systems
- Distributed objects, RMI, CORBA

The Syllabus' Current State

- Introduction to the client/server computing paradigm
- The HTML standard
- Cascading Stylesheets (CSS)
- The scripting language JavaScript
- The docuement object model (DOM)
- An in-depth coverage of PHP programming techniques
- The MySQL backend databases and PHP
- Security aspects in client/server systems
- Distributed objects, RMI, CORBA

A Context-Aware JavaScript

Last week in lectures:

- The core JavaScript language that may be used in both client- and server-side scripts
- No particular context–they were just JavaScript "fragments"

Today:

- The legal JavaScript code "with context" given in HTML files

Client-Side Program Structure

- Embedded JavaScript between <SCRIPT> tags
  - The most common method
- A reference to an external JavaScript file using

```
<script src="someJavaScriptCode.js"></script>
```

- Defining event handlers
  - Functions in terms of JavaScript statements are assigned as the value of appropriate attributes within HTML tags

```
<body onLoad="Some JavaScript code here">
```

- Using *javascript:* as a special URL protocol identifier
  - You can also use these URLs as `href` targets within any anchor element of your document
  - The result is the execution of the JavaScript code following the *javascript:* pseudo-protocol

Embedded JavaScript: The <SCRIPT> tags

Client-side JavaScript scripts usually as part of an HTML file

- Specification between <script> and </script> tags in either the <head> or <body>
  - Any number of JavaScript statements - execution in the order they appear as part of the document loading process
  - Multiple (<script>,</script>) pairs within a single HTML document - execution in the order they appear within the document
    - Functions and variables defined in one script block will be available to all scripts that follow in the same file
      `<script>var x = 1;</script>`
      later on in the same HTML page
      `<script>document.write(x);</script>`

Embedded JavaScript (cont.)

- The `src` and `language` atributes
  - To indicate the language or the language version of the actual script code
- Nested <script> tags to dynamically generate HTML and JavaScript content for display in other browser windows and frames

```
<script>
f1.document.write("<script>");
f1.document.write("document.write
     ('<h2>This is the quoted script</h2>')");
f1.document.write("</" + "script>");
</script>
```

Embedded JavaScript Example

```
<html><head><title>Today’s Date</title>
 <script language="JavaScript">
  // Define a function for use later on.
  function print_todays_date()
  {
    var d = new Date(); // today’s date and time
    document.write(d.toLocaleString());
  }
 </script>
</head>
<body>
 <hr>The date and time are:<br><b>
  <script language="JavaScript">
   // Now call the function we defined above
   print_todays_date();
  </script>
</b><hr></body></html>
```

Including External JavaScript Files

```
<script language="JavaScript1.1"
src="../../javascriptsmyCode.js"></script>
```

Why use it?

- Keep your HTML files clear of blocks of JavaScript code
- Share the same functions, variables, or other JavaScript fragmetns by several different HTML files (reduces disk usage, easier code maintenance)
- Allows the common functions to be cached by the same browser, making them load much more quickly
- A JavaScript program or web page from one web server can employ libraries exported by other web servers

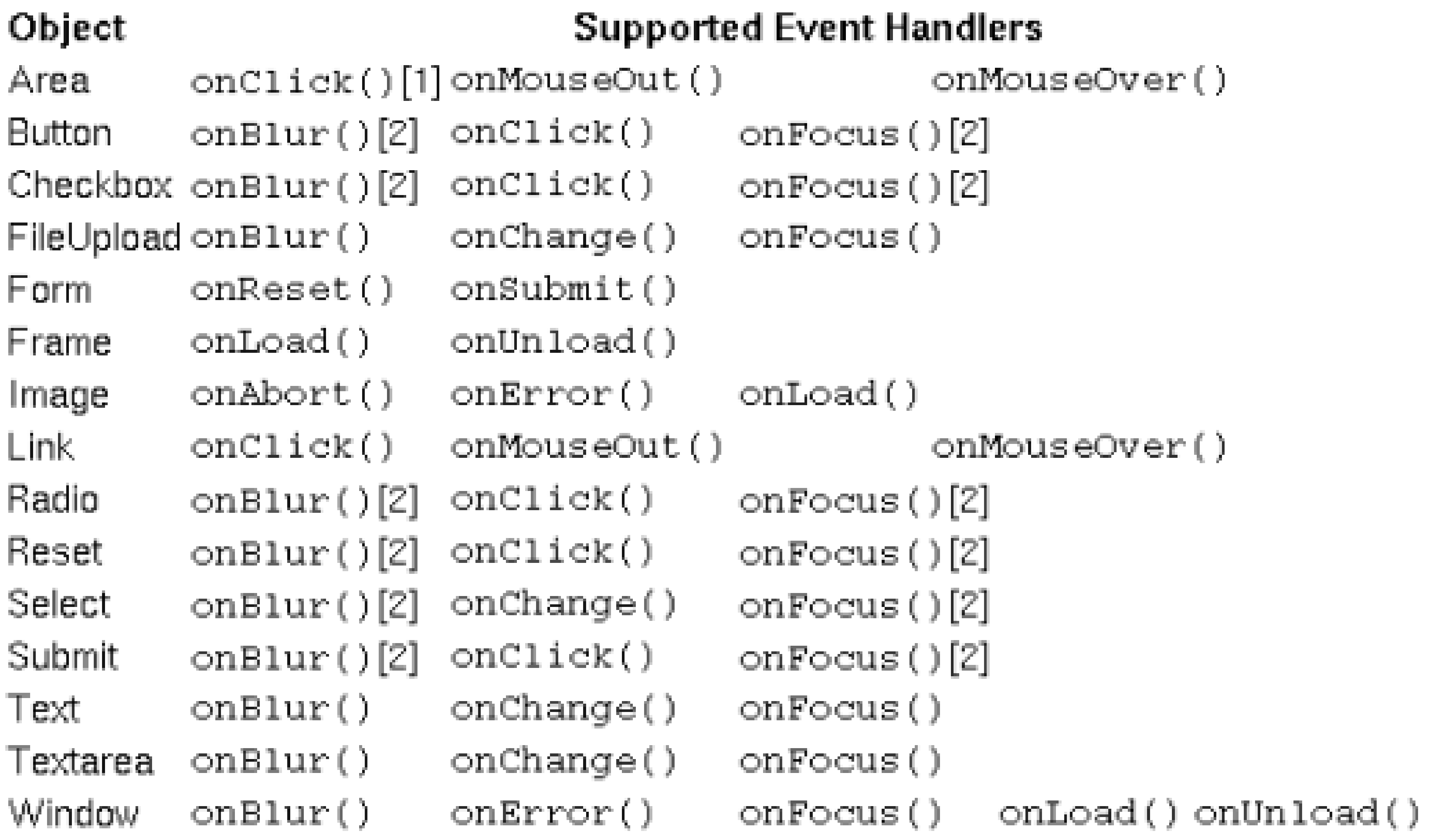
Event Hanlders in JavaScript

Defined as part of the HTML object to which it responds

```
<input type="checkbox" name="opts"
value="ignore-case" onClick="...">
```

- Event handlers as functions
  ```
  document.forms[0].opts[2].onclick
  ```

Event Hanlders in JavaScript (cont.)



Object	Supported Event Handlers			
Area	onClick()[1]	onMouseOut()	onMouseOver()	
Button	onBlur()[2]	onClick()	onFocus()[2]	
Checkbox	onBlur()[2]	onClick()	onFocus()[2]	
FileUpload	onBlur()	onChange()	onFocus()	
Form	onReset()	onSubmit()		
Frame	onLoad()	onUnload()		
Image	onAbort()	onError()	onLoad()	
Link	onClick()	onMouseOut()	onMouseOver()	
Radio	onBlur()[2]	onClick()	onFocus()[2]	
Reset	onBlur()[2]	onClick()	onFocus()[2]	
Select	onBlur()[2]	onChange()	onFocus()[2]	
Submit	onBlur()[2]	onClick()	onFocus()[2]	
Text	onBlur()	onChange()	onFocus()	
Textarea	onBlur()	onChange()	onFocus()	
Window	onBlur()	onError()	onFocus()	onLoad() onUnload()

JavaScript in URLs

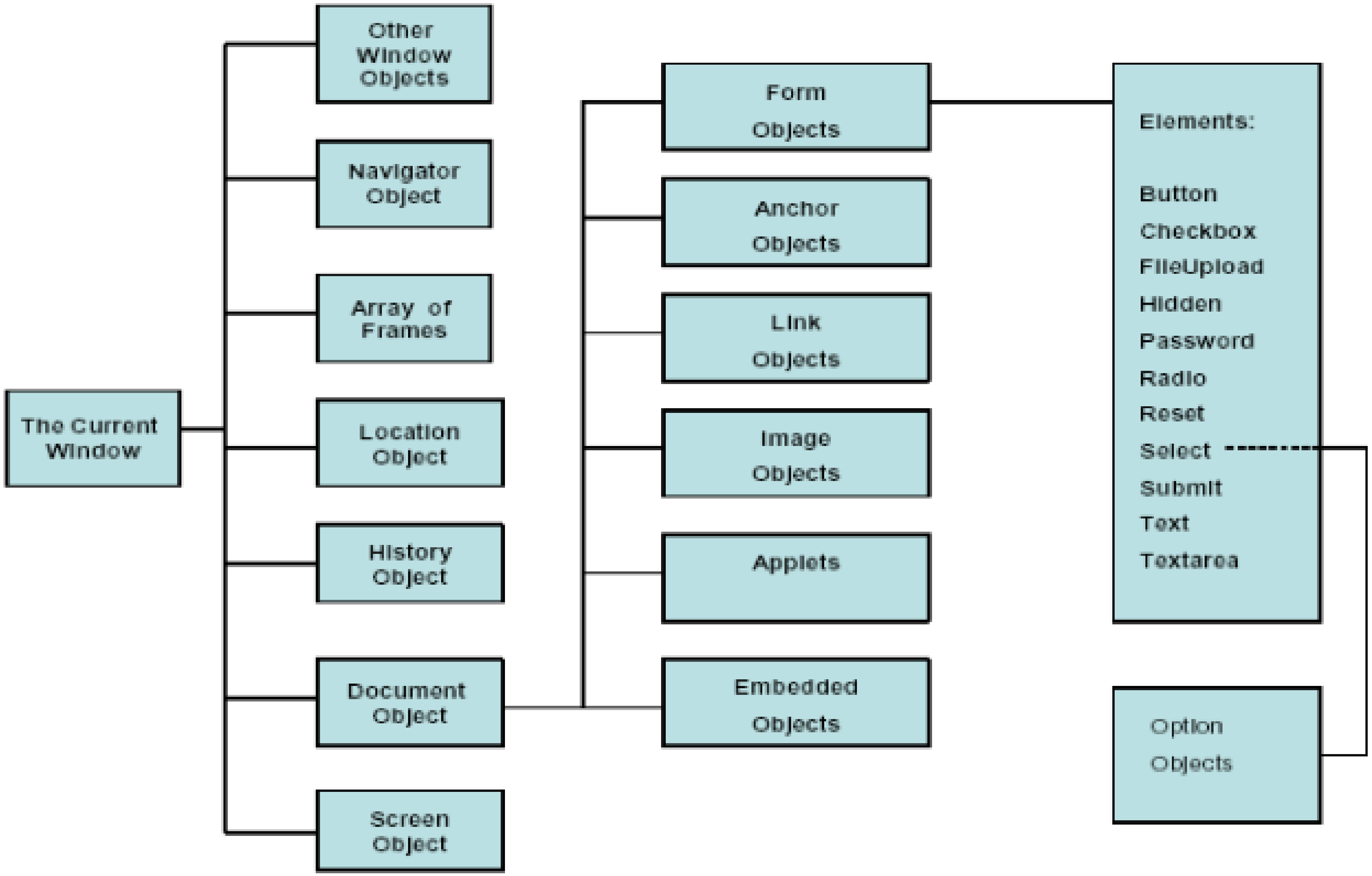
```
javascript:var now = new Date(); "<h1>The time
is:</h1>" + now;
```

- The referenced "document" is the string value of the last JavaScript statement in the URL

Window Object

- A global object of all client-side JavaScript objects that integrate the programming language with the functionality of the browser
- Incorporates the ability of a web browser to display HTML text in a window
- Also the root of the "object hierarchy": all other HTML objects in JavaScript are accessed as properties of the Window object, or as properties of those properties
- If there are more then one windows/frames in the application:
  - Each window/frame represents a global execution environment of its own
  - Global windows/frames can reference to each other

The JavaScript Object Model



Programming with Windows/Frames: Properties

- `status` and `defaultStatus`: text in a status line (URL, context help, ..)
- `parent`, `top`: reference to the parent, resp. root object in a hierarchy
- `self`, and `window`: self-reference
- `name`: specifies the name of a window or frame
- `opener`: refers to the window that opened this window
- `closed`: asks whether a window has already been closed
- `document`, `location` and `history`: refer to other HTML objects
- `frames`

Get All *Properties of a Window*

```
<script>
  for {var Allprop in window {
    document.write("Property: " + Allprop +
       "; value: " + window[Allprop], "<br>");
  }
</script>
```

Programming with Windows/Frames: Methods

- `alert()`, `confirm()`, and `prompt()`: pop up simple *dialog boxes*
  - `alert()` displays a message to the user
  - `confirm()` asks the user to click an Ok or Cancel button to confirm or abort an operation
  - `prompt()` asks the user to enter a string
- `open()` and `close()`

  ```
  window.open(url, name, [features])
  window.open("", "small",
  "location,status,width=400,height=300");
  ```
- `setTimeout()`
- `focus()` and `blur()`: transfer keyboard focus to, and away from, the window
- `scroll()`: scrolls the contents of a window (or frame)

Execution of JavaScript Programs

Part of the parsing process of the HTML page through the browser

The objects to be scripted should be parsed before the execution of the script itself

Put the definition of variables and functions in the HEAD section

- They can then be referred by your code in the BODY with the asssurance that they are **already** completely **loaded** into the browser’s memory

Ensure that all required objects are defined:

```
<body onLoad="window.loaded = true;">
 <form>
  <input type="button" value="Press Me"
    onClick="if (window.loaded != true) return;
    doit();"
  >
 </form>
</body>
```

The Navigator, Location, and History Objects

- **Navigator** provides version and configuration information about the browser
  - `appName`, `appVersion`, mimeTypes[], plugins[]
- **Location** specifies the URL currently being displayed, and allows JavaScript code to load new URLs
  - `href`: the complete text of the URL
  - `protocol`, `host`, `pathname`, and `search`: the individual parts of the URL
  - `reload()`, `replace()` methods
- **History** an array that contains information about the URLs that have been previously displayed in the window
  - The information is private, so heavy restrictions on accessing `window.history`
  - `back()`, `forward()` to move along the array

The Document Object

- Properties that give information about the document:
  - URL, `lastModified`, `referrer`–the URL of the document that linked to it, the display colors (bgColor, fgColor, ..)
- `write()` method that allow to dynamically output text into a document

```
<script>
parent.frames[0].document.open();
parent.frames[0].document.write("<h1>hi!<h1>");
parent.frames[0].document.close(); </script>
```

- Arrays that contain objects like links, anchors, HTML forms, and other embedded data in the document

Any Questions?

- See you next week