JavaScript

Instructor: Wei Ding

Brief History

- JavaScript, Jscript, ECMAScript
- **JavaScript** was invented by Brendan Eich at Netscape and first appeared in Navigator 2.0 browser. It has appeared in all subsequent browsers from Netscape and in all browsers from Microsoft starting with Internet Explorer 2.0.
- Microsoft's version of JavaScript is called **JScript**.
- Netscape, Microsoft and other companies cooperated with the ECMA (European Computer Manufacturer's Association) to produce a universal, client-side scripting language, which is referred to as **ECMA-262**. JavaScript and Jscript each conform to this standard.

Introduction

- JavaScript is a *scripting* language.
- A scripting language is a *lightweight* programming language.
- A JavaScript is lines of executable computer code that can be inserted into an HTML file.
- JavaScript is an open scripting language that anyone can use without purchasing a license.

Introduction

- Many people think that JavaScript is related to or is a simplified version of the Java programming language. However, the languages are entirely different. Java is a compiled, object-oriented programming language that was created by Sun Microsystems and is considerably more difficult to master than JavaScript.
- JavaScript is an interpreted scripting language that was created by Netscape.

How does a JavaScript work?

- When a JavaScript is inserted into an HTML document, the Internet browser will read the HTML and interpret the JavaScript. The JavaScript can be executed immediately or at a later event.
- The Web browser contains a JavaScript interpreter, which processes the commands written in JavaScript. The interpreter translates programming code into an executable format each time the program runs, one line at a time.

Example : Print a Line of Text in a Web Page

- A simple script that displays "*Welcome to JavaScript Programming!*" in the body of an HTML document.
- The <**Script**> tag indicates the browser that the text which follows is part of a script.
- The **type** attribute specifies the type of a file as well as the scripting language used in the script – in this case, a **text** file written in **javascript**.

The language attribute

- You can also use **language** attribute: <script language="JavaScript">
- Both Microsoft Internet Explorer and Netscape Communicator use JavaScript as the default scripting language. If you omit the **language** or **type** attribute from the <script> tag, your JavaScript program should still run.
- If you anticipate that your Web page will be displayed on older versions of Web browsers, then you should use the **language** attribute to specify the JavaScript version number for the version of JavaScript that is supported by that browser.

The document object

- The browser's **document** object, which represents the HTML document the browser is currently displaying.
- The JavaScript calls the **document** object's **write/writeln** method to write a line of HTML markup in the HTML document.
- JavaScript is case-sensitive. The statement Document.write("Hello World"); will cause an error message since the JavaScript interpreter does not recognize an object names Document with an uppercase D.

write vs. writeln

- document.writeln or document.write, its argument in the parentheses and the semicolon (;), together are called a **statement**. Although semicolon is optional, it is a good programming practice to always include the semicolon at the end of a statement to terminate the statement. This notation clarifies where one statement ends and the next statements begins.
- The write() and writeln() methods perform essentially the same function as adding text directly to the body of a standard HTML document. The only difference between the two methods is that the writeln() method adds a carriage return after the line.

Inside <head> or <body>?

- The browser interprets the contents of the <head> section first, so the JavaScript programs we write there will execute before the <body> of the HTML document displays. JavaScript can also be written in the <body> of an HTML file.
- It is a good idea to place as much of your JavaScript code as possible in the <head> section, since the <head> section of an HTML document is rendered before the <body> section.

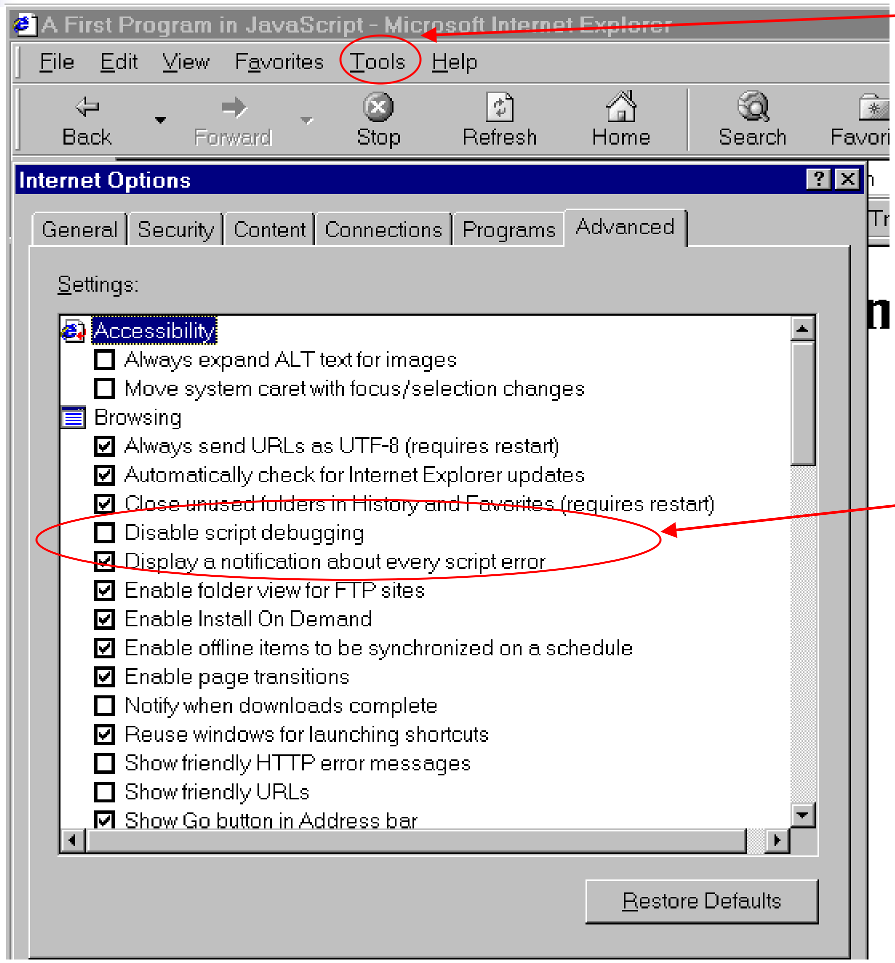
Older Web browsers

- For older Web browsers that do not support scripting, enclose the script code in an HTML comment <!-- -->. The two forward slashes in front of the end of comment line (//) are a JavaScript comment symbol, and prevent the JavaScript from trying to compile the line.
- Most Web browsers do not display lines that appear between HTML comment tags <!-- -->. However, browsers compatible with JavaScript ignore the HTML comment tags and execute the JavaScript code normally.

Escape characters

- JavaScript allows large statements to be split over many lines. However, you cannot split a statement in the middle of a string.
- The backslash (\) in a string is an **escape character**. It indicates that a "special" character is to be used in the string. When a \ is encountered in a string, the next character is combined with the \ to form an **escape sequence**.
- Many people confuses the writing of HTML text with the rendering of HTML text. Writing HTML text creates the HTML that will be rendered by the browser for presentation to the user.

Enable script debugging with IE



A First Program in JavaScript - Microsoft Internet Explorer

Internet Options

Settings:

- Accessibility
  - ☐ Always expand ALT text for images
  - ☐ Move system caret with focus/selection changes
- Browsing
  - ☑ Always send URLs as UTF-8 (requires restart)
  - ☑ Automatically check for Internet Explorer updates
  - ☑ Close unused folders in History and Favorites (requires restart)
  - ☐ Disable script debugging
  - ☑ Display a notification about every script error
  - ☑ Enable folder view for FTP sites
  - ☑ Enable Install On Demand
  - ☑ Enable offline items to be synchronized on a schedule
  - ☑ Enable page transitions
  - ☐ Notify when downloads complete
  - ☑ Reuse windows for launching shortcuts
  - ☐ Show friendly HTTP error messages
  - ☐ Show friendly URLs
  - ☑ Show Go button in Address bar

1. Click Tools and select Internet Options

2. Enable script debugging

3. Click OK to accept the selection.

Testing and Debugging Tips

- JavaScript is case sensitive. Not using the proper uppercase and lowercase letters is a syntax error.
- A syntax error occurs when the script interpreter cannot recognize a statement. The interpreter normally issues an error message to help the programmer locate and fix the incorrect statement.
- When the interpreter reports a syntax error, the error may not be on the line indicated by the error message. First, check the line for which the error was reported. If that line does not contain errors. Check the preceding several lines in the script.

Example: Display multiple lines in a dialog (Contd.)

- The example displays information in a dialogs that pop up on the screen to grab user's attention.
- window object: The IE browser build-in object. The argument to the window object's alert method is the string to display.

Example: Display multiple lines in a dialog (Contd.)

- Dialogs display plain text; they do not render HTML. Therefore, specifying HTML elements as part of a string to be displayed in a dialog results in the actual characters of the tags being displayed.
- The example also demonstrates using escape sequence in dialogs.

Basic debugging techniques

- Fully use the error messages and line number provided by the browser.
- When you are unable to locate a bug in your program by using error messages, or if the bug is a logic error that does not generate error messages, then you must trace your code.
- The alert() method provides one of the most useful ways to trace JavaScript code. You place an alert() method at different points in your program and use it to display the contents of a variable, an array, or the value returned from a function.

Variables

- You can create a variable with the **var** statement
- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Variable names are case sensitive. They must begin with a letter or the underscore character

Lifetime of Variables

- When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

JavaScript Operators

- Arithmetic operators: + - * / % ++ --
- Assignment operators: = += -= *= /= %=
- Comparison Operators: == != > < >= <=
- Logical Operators: && || !
- String Operator: +

We are different!

JavaScript operators (continue)

- **When performing arithmetic operations on string values, the JavaScript interpreter will attempt to convert the string values to numbers.** For an operator like division (/) or multiplication (*), the JavaScript interpreter will correctly perform the associated arithmetic operations.
- **But the JavaScript interpreter will not convert strings to numbers when you use the addition operator (+).** When you use the addition operator, the strings are combined instead of being added together.

Array Object

- An **array** object is used to store a set of values in a single variable name.

Capital A

```
var family_names=new Array(5);
family_names[0]="Paul";
family_names[1]= 26;
document.write(family_names.length);
```

Specifying number of array elements is optional

Example

- The script inputs two integers typed by a user at the keyboard, computes the sum of the values and displays the result.
- window.prompt displays a dialog to allow the user to input a value for use in the script.

JavaScript Functions

- A **function** is a reusable code-block that will be executed by an event, or when the function is called.
- By placing functions in the head section of the document, you make sure that all the code in the function has been loaded before the function is called.

JavaScript Conditional Statements

- Conditional statements in JavaScript are used to perform different actions based on different conditions.
- In JavaScript we have three conditional statements:
  - **if statement** - use this statement if you want to execute a set of code when a condition is true
  - **if...else statement** - use this statement if you want to select one of two sets of lines to execute
  - **switch statement** - use this statement if you want to select one of many sets of lines to execute

JavaScript Looping

- Looping statements in JavaScript are used to execute the same block of code a specified number of times.
- In JavaScript we have the following looping statements:
  - **while** - loops through a block of code while a condition is true
  - **do...while** - loops through a block of code once, and then repeats the loop while a condition is true
  - **for** - run statements a specified number of times

Using events

- Event-driven programming - The user interacts with a GUI component, the script is notified of the event and the script processes the event.
- We can use JavaScript events to add interactivity between web pages and users.
- An **event** is a specific circumstance that is monitored by JavaScript.

Events

- For example, The clicking of the button is knows as the **event**.
- You can think of an event as a trigger that fires specific JavaScript code in response to a given situation.
- The function that is called when an event occurs is known as an **event-handling function** or **event handler**.

Examples of JavaScript events

blur	an element, such as a radio button, becomes inactive
click	an element is clicked once
change	the value of an element changes
focus	an element becomes active
mouseOver	the mouse moves over an element
select	a user selects a field in a form
submit	a user submits a form

Java Script Regular Expressions

- A **regular expression** is really just a sequence or a pattern of characters that is matched against a string of text when performing searches and replacements.
- Pattern Matching and Regular Expressions were introduced in Netscape Navigator 4.0x and Microsoft Internet Explorer 4.0.

Defining a Regular Expression

- One way to define a regular expression is to simply assign it to a variable:

  var varName = /PATTERN/[g|i|gi];

  - i means performing case-insensitive matching.
  - g means performing a global match. That is, find a matches rather than stopping after the first match.

  Note that the modifier (/g, /i, or /gi) is not required.

String Methods for Pattern Matching

- **search()** method takes a regular expression argument and returns either the character position of the start of the first matching substring, or –1 if there is no match.

“JavaScript”.search(/script/i);

The above statement returns 4.

String Methods for Pattern Matching

- **match()** method takes a regular expression argument as its only argument and returns an array that contains the results of the match.

“1 plus 2 equals 3”.match(/[a-z]+/g); //returns [“1”, “2”, “3”]

- If the regular expression does not have the g flag set, match() does not do a global search; it simply searches for the *first* match. In this case, the first element of the array is the matching string, and any remaining elements are the **parenthesized** subexpressions of the regular expression.

String Methods for Pattern Matching

- **split()** method breaks the string on which it is called into an array of substrings, using the argument as a separator.

```
“123,456,789”.split(“,”);
// returns [“123”, “456”, “789”]
“1,2, 3 ,    4,5”.split(/\s*,\s*/);
// returns [“1”,”2”,”3”,”4”,”5”]
```

RegExp Methods for Pattern Matching

- **test()** method takes a string and returns true if the string matches the regular expression.

Var pattern = /java/i;

pattern.test("JavaScript");// Returns true

DHTML

- DHTML is an abbreviation for the term “Dynamic HTML”.
- DHTML is not a standard defined by the World Wide Web Consortium (W3C). DHTML is just a buzzword.
- Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated.

The Elements of DHTML (To be continued)

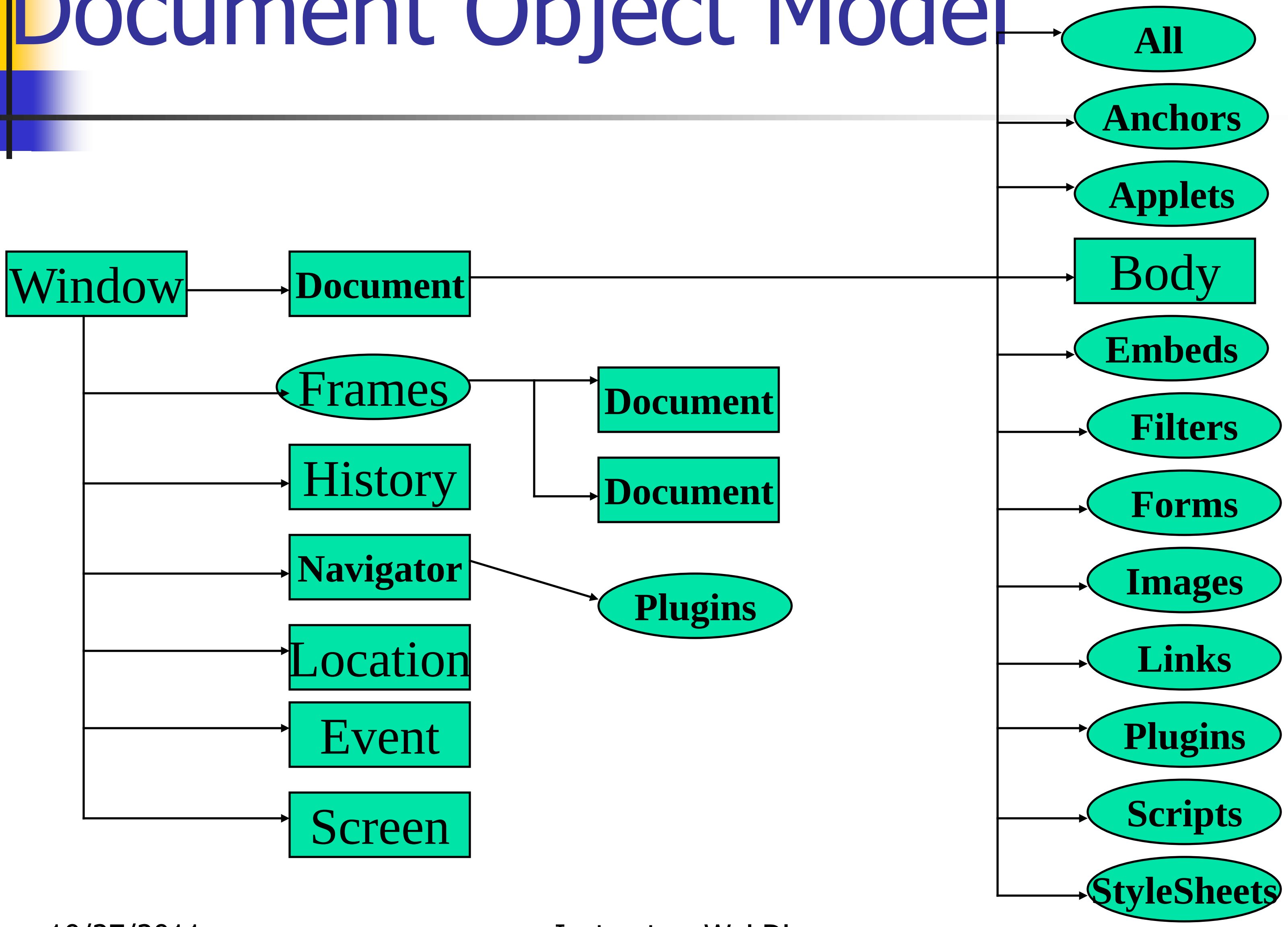
- HTML 4.0: Introduced two important things: Cascading Style Sheets(CSS) and the Document Object Model (DOM).
  - With **CSS** we got a style and layout model for HTML documents.
  - With the **DOM** we got a document content model for HTML documents
  - **Client-side scripting language** (JavaScript and VBScript) allow you to write scripting code to control HTML elements.

  To most people Dynamic HTML means a combination of DOM, Style Sheets and JavaScript.

DHTML Document Object Model

- **DOM**: HTML 4.0 introduced the **Document Object Model**, which gives us access opportunity to every element in the document. It contains no functions or fancy layout tricks, it is more like a map of all the elements, and we use this map to access the elements.
- Literally every element in a document is represented by a separate object.

Document Object Model



DHTML Objects and Collections (To be continued)

- window object: Represents the browser window and provides access to the **document** object contained in the **window**. If the **window** contains frames, a separate **window** object is created automatically for each frame, to provide access to the **document** rendered in that frame. Frames are considered to be subwindows in the browser.
- document Object: Represents the HTML documents rendered in a **window**; Provides access to every element in the HTML document and allows dynamic modification of the HTML document.

How to access the elements

- We use JavaScript to access the elements. To make an element available for the DOM, the element must define the id attribute.

DHTML Event Model

- DHTML with event model exists so that scripts can respond to user interactions and change the page accordingly.
- It makes Web applications more responsive and user-friendly and can reduce server load.
- With the event model, scripts can respond to a user who is moving the mouse, scrolling up or down the screen or entering keystrokes. Content becomes more dynamic while interfaces become more intuitive.

DHTML Event Examples

- **event** object: Can be used in an event handler to obtain information about the event that occurred. (e.g., the mouse coordinates during a mouse event)
- Event **onmousemove**: Whenever the user moves the mouse over the Web page, the event fires repeatedly.
- Event **onfocus** and **onblur**: The **onfocus** event fires when an element gains focus (I.e., when the user clicks a form field or when the user uses the Tab key to move between form elements) and **onblur** fires when an element loses focus, which occurs when another control gains the focus.

DHTML Cascading Style Sheets

- HTML 4.0 introduced CSS, which gave us a style and layout model for HTML. With CSS, HTML elements can be styled with colors, backgrounds, borders, position, and visibility. The style can be changed dynamically in DHTML.

Acknowledgements

- www.w3schools.com
- David Flanangan, "JavaScript, The definitive Guide", O'Reilly
- Don Gosselin, "JavaScript", Course Technology, Thomson Learning