Web browser e programmazione client-side

Corso di *Applicazioni Telematiche*

A.A. 2006-07 – Lezione n.10 – parte I

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

Client-Side Programming

- For several reasons, it can be desirable to let the web browser execute code to generate parts of the HTML code to be rendered and/or to perform computation on input data provided by the user while interacting with the page content
- Code executed by the browsers may be:
  - Written in a scripting language, interpreted by an interpreter embedded into the browser application
  - Written in binary native format, directly executed on the client host
  - Written in Java bytecode and executed by a Java Virtual Machine activated by the brower application
- The browser executes the program as it loads the page, integrating the dynamic output of the program with the static content of HTML

Scripts vs. Programs

- Most *browsers* support a <SCRIPT> tag that is used to include executable content in an HTML document
- A number of *scripting* languages are supported by common web browsers:
  - e.g., JavaScript, JScript, VBScript
- A scripting language is a simple, interpreted programming language
  - scripts are embedded as plain text, interpreted by application
  - *simpler execution model:* don't need compiler or development environment
  - *platform-independence:* code interpreted by any script-enabled browser
  - *but:* slower than compiled code, not as powerful/full-featured
  - for security reasons, scripts are limited in what they can do
    - *e.g., can't access the client's hard drive*

Web scripting languages

- JavaScript: the first Web scripting language, developed by Netscape in 1995
  - syntactic similarities to Java/C++, but simpler & more flexible
    - loose typing, dynamic variables, simple objects

- JScript: Microsoft version of JavaScript, introduced in 1996
  - same core language, but some browser-specific differences
  - fortunately, IE & Netscape can (mostly) handle both JavaScript & JScript

- VBScript: client‐side scripting version of Microsoft Visual Basic

Common Scripting Tasks

- adding dynamic features to Web pages
  - validation of form data
  - image rollovers
  - time-sensitive or random page elements
  - handling cookies

- defining programs with Web interfaces
  - utilize buttons, text boxes, clickable images, prompts, frames

JavaScript Capabilities

- Add content to a web page dynamically.
- Alter a web page in response to user actions.
- React to user events.
- Interact with frames.
- Manipulate HTTP cookies

Typical uses of Javascript

- Browser Detection
  Detecting the browser used by a visitor at your page. Depending on the browser, another page specifically designed for that browser can then be loaded.
- Cookies
  Storing information on the visitor's computer, then retrieving this information automatically next time the user visits your page. This technique is called "cookies".
- Control Browsers
  Opening pages in customized windows, where you specify if the browser's buttons, menu line, status line or whatever should be present.
- Validate Forms
  Validating inputs to fields before submitting a form.
  An example would be validating the entered email address to see if it has an @ in it, since if not, it's not a valid address.

JavaScript is not Java

- JavaScript is a very simple scripting language.
- Syntax is similar to a subset of Java.
- Interpreted language.
- Uses objects, but doesn't really support the creation of new object types*

*It almost does, but it's cumbersome.

JavaScript and HTML

- JavaScript code is "embedded" into the HTML code of a web page
- The "script" pair of tags define where the JavaScript code starts and ends

```
<html>
<head>
<title>My Javascript Page</title>
</head>
<body>
<script type="text/javascript">
alert("Welcome to this web page!");
</script>
</body>
</html>
```

JavaScript and HTML (2)

- Javascript code can be included in both the <head> and <body> sections of the document
- In general, however, it is advisable to keep as much as possible in the <head> section
- Javascript lines end with a semicolon
- Instead of having Javascript write something in a popup box we could have it write directly into the document
- The **document.write** is a javascript command telling the browser that what follows within the parentheses is to be written into the document

JavaScript and HTML (3)

- All kinds of HTML tags can be inserted into the HTML page with the **document.write** method

```
<html>
<head>
<title>My Javascript Page</title>
</head>
<body>
Hello!!!<br>
<script>
document.write("Welcome to my world!!!<br>");
</script>
Enjoy your stay...<br>
</body>
</html>
```

Language Elements

- Variables
- Literals
- Operators
- Control Structures
- Objects

JavaScript Data Types & Variables

- JavaScript has only three primitive data types
  String : `"foo"` `'howdy do'` `"I said 'hi'."` `""`
  Number: `12` `3.14159` `1.5E6`
  Boolean : `true` `false`
- Assignments are as in C++/Java
  ```
  message = "howdy";
  pi = 3.14159;
  ```
- Variable names are sequences of letters, digits, and underscores *starting with a letter*
- Variables names are case-sensitive
- Variables don't need to be declared first
- Variables are loosely typed, can be assigned different types of values

Variables declaration

- Using **var** to declare a variable results in a *local* variable (inside a function).

- If you don't use **var** – the variable is a global variable.

JavaScript Operators & Control Statements

- standard C++/Java operators & control statements are provided in JavaScript
  - +, -, *, /, %, ++, --, …
  - ==, !=, <, >, <=, >=
  - &&, ||, !,===,!==
  - if-then, if-then-else, switch
  - while, for, do-while, …

Objects

- Objects have attributes and methods.
- Many pre-defined objects and object types.
- Using objects follows the syntax of C++/Java:
  ```
  objectname.attributename
  objectname.methodname()
  ```

JavaScript Functions

- The keyword `function` used to define a function (subroutine):

  ```
  function add(x,y) {
      return(x+y);
  }
  ```

- function definitions are similar to C++/Java, except:
  - no return type for the function (since variables are loosely typed)
  - no types for parameters (since variables are loosely typed)
  - by-value parameter passing only (parameter gets copy of argument)

User-Defined Functions

```
<HTML><HEAD><TITLE>A function with a parameter</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function openAndCenterWindow(url)
{
    var the_window =  window.open(url,'the_window,
                         'height=100,width=300');
    var screen_height = window.screen.availHeight;
    var screen_width = window.screen.availWidth;
    var left_point = parseInt(screen_width / 2) - 150;
    var top_point = parseInt(screen_height/2);
    the_window.moveTo(left_point, top_point);
    the_window.focus();
}
</SCRIPT></HEAD>
<BODY>click on me to open a small centered window
<a href="#" onClick="openAndCenterWindow('http://www.unina.it/');
      return false;">to UniNA website</a>
</BODY>
</HTML>
```

Interactive Pages Using Prompt

```
<html>
<head>
  <title>Interactive page</title>
</head>
<body>
<script type="text/javascript">
    userName = prompt("What is your name?", "");

    userAge = prompt("Your age?", "");
    userAge = parseFloat(userAge);

    document.write("Hello " + userName + ".")
    if (userAge < 18) {
      document.write("  Do your parents know " +
                     "you are online?");
    }
</script>
<p>The rest of the page...
</body>
</html>
```

JavaScript Strings

- a *class* defines a new type (formally, *Abstract Data Type*)
  - encapsulates data (properties) and operations on that data (methods)
- a String encapsulates a sequence of characters, enclosed in quotes

  properties include
  - `length`   : stores the number of characters in the string

  methods include
  - `charAt(index)`   : returns the character stored at the given index (as in C++/Java, indices start at 0)
  - `substring(start, end)`   : returns the part of the string between the start (inclusive) and end (exclusive) indices
  - `toUpperCase()`   : returns copy of string with letters uppercase
  - `toLowerCase()`   : returns copy of string with letters lowercase

  to create a string, assign using `new` or just make a direct assignment (`new` is implicit)

      `word = new String("foo");`   `word = "foo";`

  properties/methods are called exactly as in C++/Java

  - `word.length`   `word.charAt(0)`

JavaScript Arrays

- arrays store a sequence of items, accessible via an index

  since JavaScript is loosely typed, elements do not have to be the same type

  - to create an array, allocate space using `new` (or can assign directly)

    ```
    items = new Array(10);          // allocates space for 10 items

    items = new Array();   // if no size, will adjust dynamically

    items = [0,0,0,0,0,0,0,0,0]; // can assign size & values []
    ```

  - to access an array element, use `[]` (as in C++/Java)

    ```
    for (i = 0; i < 10; i++) {
        items[i] = 0;              // stores 0 at each index

    }
    ```

Array Objects

- Arrays are supported as objects.

- Attribute **length**

- Methods include:
  concat join pop push reverse sort

Date Class

- ## String & Array are the most commonly used classes in JavaScript
  - other, special purpose classes & objects also exist

- ## the Date class can be used to access the date and time
  - to create a Date object, use new & supply year/month/day/… as desired
    - `today = new Date();          // sets to current date & time`
    - `newYear = new Date(2002,0,1); //sets to Jan 1, 2002  12:00AM`

document Object

Both IE and Netscape allow you to access information about an HTML document using the `document` object *(Note: not a class!)*

```
<html>
<!-- COMP519  js13.html  09.09.2005 -->

<head>
  <title>Documentation page</title>
</head>

<body>
  <table width="100%">
    <tr>
      <td><small><i>
        <script type="text/javascript">
          document.write(document.URL);
        </script>
      </i></small></td>
      <td align="right"><small><I>
        <script type="text/javascript">
          document.write(document.lastModified);
        </script>
      </i></small></td>
    </tr>
  </table>
</body>
</html>
```

`document.write(…)`
method that displays text in the page

`document.URL`
property that gives the location of the HTML document

`document.lastModified`
property that gives the date & time the HTML document was saved

navigator Object

`navigator.appName`

property that gives the browser name

`navigator.appVersion`

property that gives the browser version

```
<!-- MSIE.css  -->

a {text-decoration:none;
   font-size:larger;
   color:red;
   font-family:Arial}
a:hover {color:blue}
```

```
<!-- Netscape.css  -->

a {font-family:Arial;
   color:white;
   background-color:red}
```

```
<html>
<!-- COMP519  js14.html  09.09.2005 -->

<head>
  <title>Dynamic Style Page</title>

  <script type="text/javascript">
    if (navigator.appName == "Netscape") {
      document.write('<link rel=stylesheet '+
        'type="text/css" href="Netscape.css">');
    }
    else {
      document.write('<link rel=stylesheet ' +
        'type="text/css" href="MSIE.css">');
    }
  </script>
</head>

<body>
Here is some text with a
<a href="javascript:alert('GO AWAY')">link</a>.
</body>
</html>
```

JavaScript Events

- JavaScript supports an event handling system.
  - You can tell the browser to execute javascript commands when some event occurs.
  - Sometimes the resulting *value of the command* determines the browser action.

Simple Event Example

```
<BODY BGCOLOR=WHITE onUnload="restore()">
<H5>Hello - I am a very small page!</H5>
<SCRIPT>
savewidth =  window.innerWidth;
saveheight = window.innerHeight;
function restore() {
  window.innerWidth=savewidth;
  window.innerHeight=saveheight;
}
// Change the window size to be small
window.innerWidth=300; window.innerHeight=50;
document.bgColor='cyan';
</SCRIPT>
```

Buttons

- You can associate buttons with JavaScript events (buttons in HTML forms)

```
<FORM>
<INPUT TYPE=BUTTON
VALUE="Don't Press Me"
onClick="alert('now you are in trouble!')" >
</FORM>
```

Java applets

- An applet is a Java program that is run by a Java enabled web browser
- Classes downloaded from network
- Applets have special security restrictions
  - Executed in applet sandbox
- Java programs are platform independent:
  - compiled to "bytecode" (class files) not machine code
  - Java "bytecode" can be run on any machine with a Java Virtual Machine installed
- Most of the browsers need a *Java Runtime Environment* (JRE) installed, that provides a *Java Virtual Machine* (JVM)
- A web browser runs an Applet by first loading an HTML document (*HyperText Markup Language*)

Java applets and HTML

```
<HTML>
<HEAD>
   <TITLE>Applet0</TITLE>
</HEAD>

<BODY>
   <HR>
     <APPLET
          CODE="HelloApplet.class"
          WIDTH=300 HEIGHT=60>
     </APPLET>
   <HR>
</BODY>
</HTML>
```

load an applet

from this file

in a box this big

La classe Applet

Una applet è implementata come sottoclasse
della classe java.applet.Applet

java.applet

Class Applet

```
java.lang.Object
    └ java.awt.Component
        └ java.awt.Container
            └ java.awt.Panel
                └ java.applet.Applet
```

Applet: ciclo di vita

4 eventi a ciascuno dei quali è associato un "metodo"

Inizializzazione	→ void init()
Avvio	→ void start()
Arresto	→ void stop()
Distruzione	→ void destroy()

Applet: ciclo di vita (2)

- The init() method is called when the applet is initially loaded. This method is used to do one time setup features such as add components to the layout manager, set screen colors, and connect to a host database.
- The start() method is called after the applet has been initialized, and also each time the applet is restarted after being stopped. Applets can be stopped when the user changes to another Web page. If the user returns at a later time to the page with the applet on it, the applet will be restarted by the browser. Therefore, the start() method can be called many times during an applet's life cycle. Common operations that occur during an applet's start() method are the initialization of threads within the applet and the updating of the screen display.

Applet: ciclo di vita (3)

- The stop() method is called whenever the user leaves the current page. Note that by default when the user leaves a page, the applet will continue to run. This can result in an enormous consumption of system resources if many applets have been loaded and these applets are performing some resource intensive task such as animation. (In fact, it is quite common to see poorly written applets loaded from the Internet that obviously did not implement this method. They never stop running!) The stop() method is used to temporarily suspend the execution of the applet until the start() method is called again.
- The destroy() method is called whenever it is time to completely finish the applet's execution. This method is generally called when the browser is exiting or the applet is being reloaded from the host server. The destroy() method is used to free up allocated resources such as threads or database connections.

Il codice dell'applet "Hello, world!"

```
import java.awt.*;
import java.applet.*;

public class HelloApplet extends Applet {

    public void paint(Graphics g) {
        g.drawString("Hello, world!", 10, 30);
    }
}
```

ereditarietà

Ereditato dalla classe **Component** del pacchetto **awt**. Serve per visualizzare qualcosa nella applet.

LifeCycle applet (1)

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class LifeCycleApplet extends java.applet.Applet
{
  Font theFont = new Font("Helvetica", Font.BOLD, 20);
  int  i;
  String String1, String2;

  public void paint(Graphics g)
  {
    g.setFont(theFont);
    g.setColor(Color.blue);
    g.drawString(String1 + String2, 5, 30);
  }

  public void init()
  {
    i = 0;
    String1 = "";
    String2 = "The applet is initializing!";
    repaint();
    showStatus("The applet is initializing!");
  }
```

LifeCycle applet (2)

```
public void start()
{ i = 1;
  String1 = String2;
  String2 = "The applet is starting!";
  repaint();
  showStatus("The applet is starting!");
}
public void stop()
{ i = 2;
  String1 = String2;
  String2 = "The applet is stopping!";
  repaint();
  showStatus("The applet is stopping!");
}
public void destroy()
{ i = 3;
  String1 = String2;
  String2 = "The applet is being destroyed!";
  repaint();
  showStatus("The applet is being destroyed!");
}
}
```
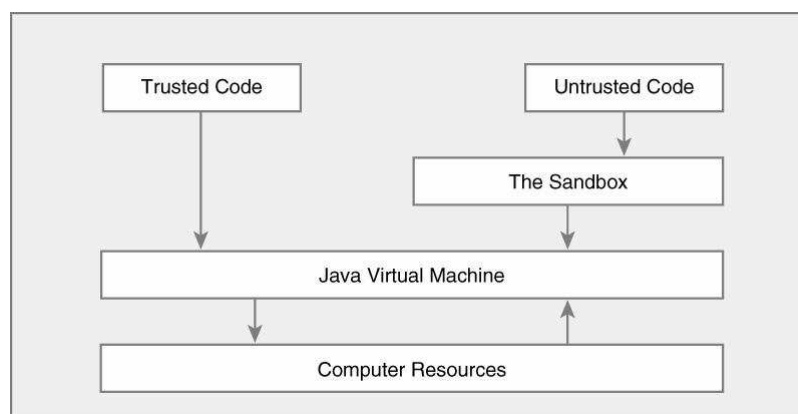
Restrictions on Applets

- Certain restrictions on what applets can do, so can be safely run on web user's machine:
  - Can't read and write files on user's machine.
  - Can't generally make network connections
  - Can't start other programs.
- But can make connections to server hosting Applet, and (e.g) access Database there.

Applet Sandbox

- Prevents
  - Loading libraries
  - Defining native methods
  - Accessing local host file system
  - Running other programs (Runtime.exec())
  - Listening for connections
  - Opening sockets to new machines
    - Except for originating host
- Restricted access to system properties

Applet Sandbox

Helpers and Plug-ins

- Have so far looked at JavaScript and Java for client-side web applications.
- JavaScript - Simple, but can only manipulate web page.
- Java - Complex, suited for serious programming.
- Both interpreted by browser.
- How can we run other sorts of applications on WWW?

Helpers and Plug-ins

- Don't rely on browser to display/run every kind of data/program.
- Some can be handled by special extra applications installed on client machine.
- Helpers - display results in separate window.
- Plug-ins - act with browser and display results within page.

Examples

- Shockwave Flash (plugin)
- ghostview (helper application for postscript)
- acroread (helper application for pdf)
- Range of applications for audio and video.

Mime Types

- How does the browser know which application to use for which type of file?
- Each file may have a MIME type (standard way of specifying file types).
- On Windows the file name suffix (e.g, thing.html, thing.ps) indicates the MIME type.

Flash Example

- A complicated way to say Hello World
- Plugins can be embedded in HTML using object and embed tags.
- The "swf" suffix indicates Shockwave/Flash

```
<object>
 <embed src="FlashHelloWorld.swf" width="550"
        height="400" </embed>
<noembed>unlucky</noembed>
</object>
```

Try it

Choosing a client-side technology

- Java, JavaScript and various plug-ins can all be used for web applications
- Which is best when?
  - Will it run on most user's machines?
    - Not everyone has plug-ins installed
    - Some users disable JavaScript
    - Old browsers may not have CSS etc.
  - Does the task need specialised tools (e.g. multimedia)?
  - What expertise is available?

Domande?