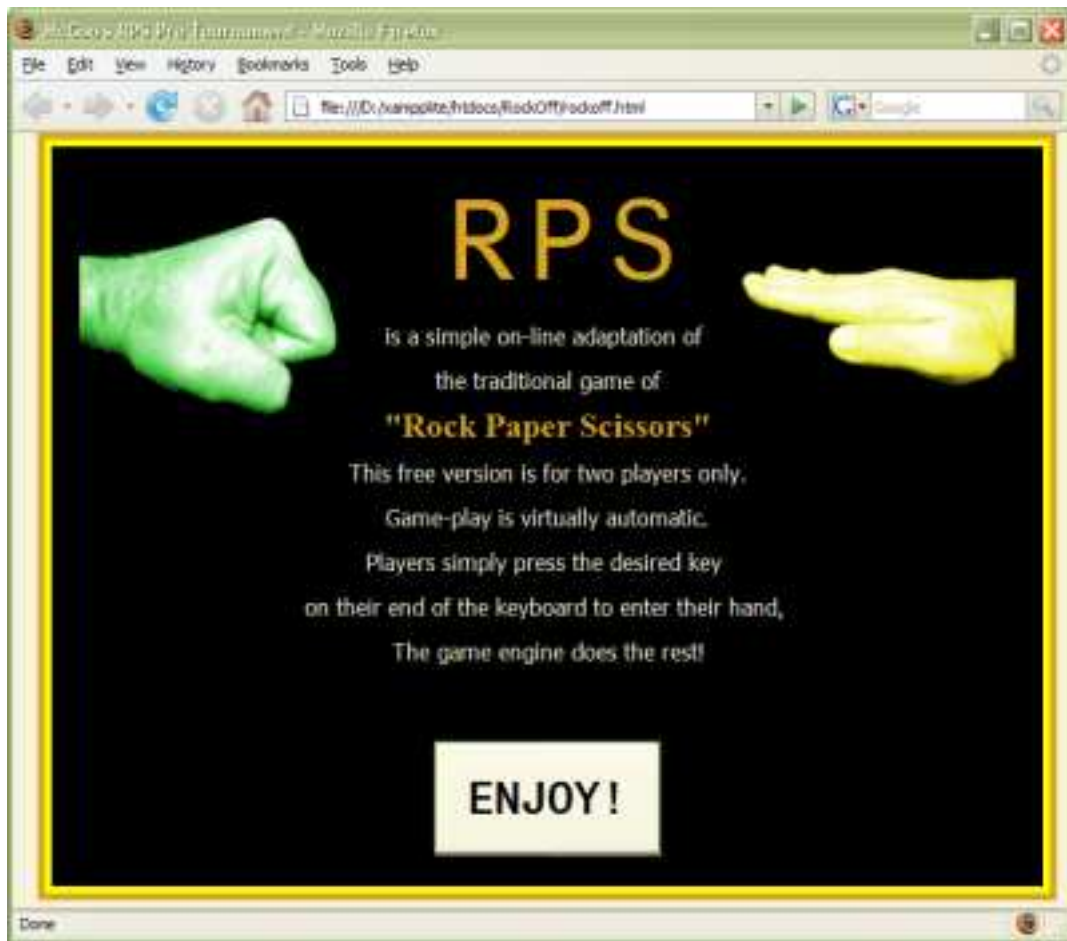


Software Engineering for IPT using Single Page Applications developed with DHTML



By
Dave McGuinness
Urangan State High School

QSITE Conference 2007
IPT Strand

An Egyptian Ant's Eye View

“Web-based Programming is a bit like an ant building its hill on top of the great pyramid. It took a lot of guys a very long time to build the pyramid, but you don't have to know much about that to build a great ant-hill. A reasonable idea about the layout of your local block, a bit of organisational ability, and some hard work should be enough...”

McGoo, 2007

SPA/DHTML – What is it?

- **SPA**

(Single Page Application) is a term used to describe browser-based programming that contains the complete user interface within one page, without the need for a page refresh. This approach provides the user with a more *desktop-centric* experience which is more immediate. While Firefox and IE7 refresh much faster than IE6, there is still an annoying delay at virtually every step of the traditional form-based approach to web-programming. The SPA application should allow the user to forget that they are remote from the experience and just get on with their business. This is particular true of game applications.

- **DHTML**

Dynamic HTML most commonly uses a range of reasonably mature technologies to provide a useful client-side programmable interface within an internet browser. Careful selection of the relevant technologies can provide a rich multimedia experience on a 2-D level.

DHTML is best used with the **XHTML** 1.0 standard which extends **HTML** 4.01 to standardise on an **XML**(e**X**tensible **M**arkup **L**anguage) compliant document model.

- **XHTML** is used to outline the basic structure of the document
- **CSS** (**C**ascading **S**tyle **S**heets) standard is used to outline the appearance of the page
- **Javascript** programming language(standardised by W3C as **ECMAScript**) is used to manipulate the page via the **DOM** (**D**ocument **O**bject **M**odel).

DHTML can be easily extended to include server-side interactivity and database server access via XML or PHP. A popular combination is known as **AJAX** (**A**ynchronous **J**avascript and **X**ML). I am working on my own variant based on my familiarity with PHP, which I call **AJUP** (**A**ynchronous **J**avascript **U**sing **P**HP). However, the term AJAX has generally been broadened to include any type of asynchronous programming technique. **SPA** is probably a better term than AJAX only because it is less language-specific.

AJAX relies on an un-planned use of the XMLHttpRequest object of the browser available in IE, but part of all browsers. This allows a browser page to request XML data from the server and be notified when it becomes available. Then the application parses the data to provide output to page via Javascript. This allows the user to go on using the application while the program updates as necessary in the background. To use AJAX then, one must develop a further knowledge of XML, etc.

AJUP relies on another un-planned use of a little-known feature of browsers to produce a similar effect. Javascript can defines in-line within the HTML code, or using the appropriate code, through an external script. Because Javascript is dynamic, it can create new elements on the page including new script references. New Javascript script references are automatically executed by the browser. However, it turns out that any script will be automatically executed including PHP, which is of course server-side and thus immediately gives the developer background access to the server and associated databases. If one, is already familiar with, and indeed has already developed a substantial code-base in, PHP then progress to useful SPA development is very smooth. The real power in this approach occurs when the called PHP script includes Javascript commands which are also executed automatically. This allows real-time client-side reaction to changes in the database on the server.

SPA/DHTML – Why use it?

- TCO – Total Cost of Ownership = zero dollars. Mozilla Firefox is free. Many suitable text editors, starting with Notepad.exe are free. No more software is required.
- Clear and easily accessible standards; all readily available from websites. The HTML, CSS, XHTML, XML, DOM and ECMA standards are all maintained by CERN at www.w3c.org. Downloadable references are available. The Mozilla Project also provides very useful references for Javascript and DOM.
- Narrow frame of reference and limited environment make DHTML more accessible for students. It does not overwhelm with gadgets like VB or Delphi. Students switching from the Delphi IDE immediately were more comfortable and "felt" that the work was easier. This is only a perception, but a very important one when considering the benefits of scaffolding student work. Everything has to be done within the bounds of a normal web page. Surprisingly, very useful applications and entertaining games can be created in this way. The environment is definitely more than sufficient for a secondary school IT program.
- Clearer separation of interface design from program code. Eg, concepts such as syntax and layout of code can be learned in the simpler and more forgiving environment of HTML/CSS, then enhanced by upgrading to XHTML, before venturing into the more daunting area of coding where students tend to focus on functionality and the important issues of meaningful identifiers and readable code tends to get lost in the confusion.

DHTML – What do you need?

1. A text editor, most preferably with syntax highlighting and a multiple document interface. ConTEXT (<http://www.context.cx>) is the best one I have seen so far. It is freeware and readily available.
2. A standards-compliant browser, like **Mozilla Firefox**, definitely not IE, not even version 7, which still does not implement the CSS standard correctly and drives students mad with paranoid defence mechanisms. **Firefox** is very compliant and includes some very useful features, including the **DOM Inspector** and a very helpful **Javascript Error Console**.
3. An extension of the **Firefox** interface called the **Web Developers Toolbar** adds some brilliant functionality to **Firefox** that makes it the browser of choice for web development.
4. Quicktime plug-ins for **Firefox** give good multimedia access while allowing Javascript control to Play, Stop, Rewind, etc.

DHTML = DOM + XHTML + CSS +, Javascript

DHTML can be organised logically by separating code into relevant files. Because the interface is built on the Document Object Model within the browser window, the DOM is the foundation code.

The layout of the application in the browser window is built-in **HTML**. Students will need a working knowledge of HTML structures before attempting DHTML. Latest versions of Javascript require dynamically created code to meet the XHTML standard or errors may be thrown by the interpreter. It follows that it is a good idea to insist that students follow the more stringent syntax of XHTML from the beginning to avoid such problems later. Understanding of the DIV and SPAN elements is important as these give complete control over location and movement on the page.

Once students have mastered HTML, **CSS** presentation should be added in a separate .css file. While in-line stylesheets work fine, this clear organisational separation is a useful concept the flows on to further work.

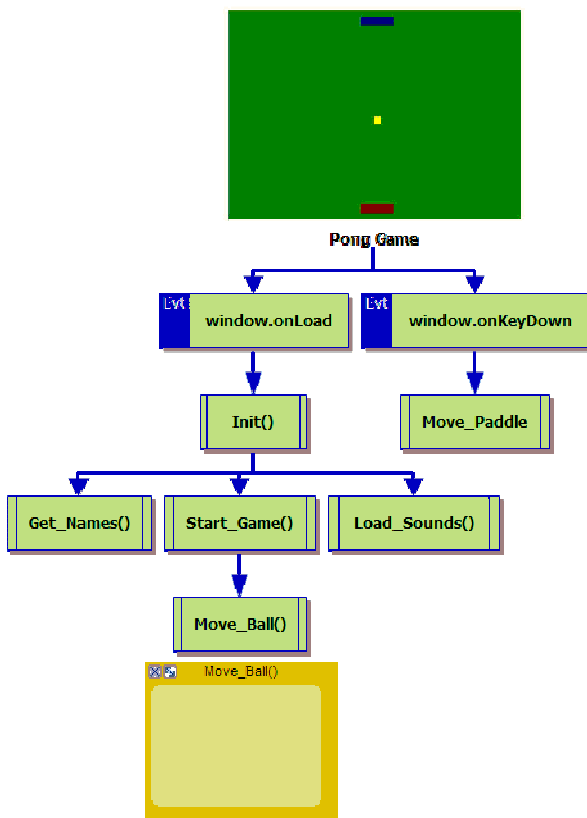
Javascript code should also be added mainly in a separate file for clear separation of function from layout. Where necessary it is useful to place some code in-line, but a much more useful construct is to use a top-down design approach to link Javascript functions in the .js file to events triggered by the interface.

Once students have mastered the basis of the three coding elements, you can then move on to developing basic applications to demonstrate the use of sequence, selection, and iteration.

Beyond that, application development should be taught via **top-down design** and **step-wise refinement** approach. I recommend using a design tool like **Inspiration** to develop a block diagram for the top-down design and **Pseudocode** the step-wise refinement. Examples of these are available from the workshop resources at <http://www.mcgoo.com.au>.

Pong: An Example

Top-Down Design



This top-down structure diagram has been developed in **Inspiration** (www.inspiration.com) which is an excellent application for diagramming structural ideas. Students are taught to develop a mock-up of the interface in **HTML** and **CSS** before attempting algorithm design. This requires them to think about **HCI** (Human Computer Interface). What will the game look like to the user? How will the user interact with the game? Events that represent all possible user interactions are linked to the interface, then appropriate functions to handle these events are linked to the events.

Pseudocode

The next step is to start stepwise refinement of the algorithm by using pseudocode. Inspiration allows the use to add notes to each node in the design tree. We use this note window to enter the pseudocode steps for the program. Later this pseudocode is copied as comments in to the **Javascript** code editor from which to develop the actual program code. In Outline View, **Inspiration** shows all of the notes as an outline of the design.

HTML

```
<!--
  DHTML Pong Game
  Based on traditional Pong
  Copyright Dave McGuinness 2007
  HTML Layout
-->

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <title>McGoo's Pong</title>
    <link rel="stylesheet" href="style.css" type="text/css" />
    <script type="text/javascript" src="pong.js"></script>

  </head>

<body onload="Init();" onkeydown="Key_Down(event);">
<div id="theScreen">

  <div id="theBall"></div>

  <div id="theBat"></div>

</div>
<div id="qt"></div>
</body>
</html>
```

CSS

```
/*
    DHTML Pong Game
    Based on traditional Pong
    Copyright Dave McGuinness 2007
    Style Sheet Layout
*/
body {
    margin: 4px;
    background-color : #EEEEFF;
    font-family:tahoma;
}
div{
    color:white;
    text-align:center;
}
#theScreen{
    position:absolute;
    margin:10px;
    top:10px;
    left:10px;
    height:500px;
    width:700px;
    background-color:green;
}

#theBat{
    position:absolute;
    top:465px;
    left:315px;
    height:20px;
    width:80px;
    background-color:maroon;
/*    background-image:url(bat.png);
*/
}
#theBall{
    z-index:2;
    position:absolute;
    top:255px;
    left:345px;
    height:20px;
    width:20px;
    background-color:yellow;
/*    background-image:url(ball.png);
*/
}
```

Javascript

```
/* *****  
  
    DHTML Pong Game  
    Copyright Dave McGuinness 2007  
  
    Javascript code  
  
/* *****/  
  
/* global variables */  
var sPlayer_Name = '';  
var iVertical = -2;  
var iHorizontal = -2;  
var iSpeed = 2;  
  
/* function to initialise game */  
function Init()  
{  
  
    // define bat object  
    objBat = document.getElementById("theBat");  
    // define ball object  
    objBall = document.getElementById('theBall');  
    // run function to Load quicktime sounds  
    Load_Sounds();  
    // run function to get user names  
    Get_Names();  
    // set global iSpeed = prompt for speed  
    iSpeed = prompt("Select a speed for the game.", iSpeed);  
    // set global iVertical = - speed  
    iVertical = -iSpeed;  
    // set global iHorizontal = - speed  
    iHorizontal = -iSpeed;  
    // run function to start game  
    Start_Game();  
  
}    /* end of Init function */  
  
//function to start new game  
function Start_Game()  
{  
    // run function to move ball  
    Move_Ball();  
}    /* end of Start_Game function */  
  
// function to move ball  
function Move_Ball()  
{  
    // if ball is on bottom  
    if(objBall.offsetTop > 480)  
    {  
        // show game over message  
        alert("Game Over!");  
        // set bGameOver to true  
        var bGameOver = true;  
    }  
    // else if ball is at left or right of screen  
    else if(objBall.offsetLeft > 680 || objBall.offsetLeft < 0)  
    {
```

```

        // toggle iHorizontal
        iHorizontal = -iHorizontal;
        document.getElementById("qt_player2").Play();
    }
    // else if ball is at top of screen
    else if(objBall.offsetTop < iSpeed)
    {
        // toggle iVertical
        iVertical = -iVertical;
        document.getElementById("qt_player2").Play();
    }
    // else if ball is touching bat
    else if(objBall.offsetTop > objBat.offsetTop - 20 &&
        objBall.offsetLeft > objBat.offsetLeft &&
        objBall.offsetLeft < objBat.offsetLeft + 80
        )
    {
        // toggle iVertical
        iVertical = -iVertical;
        document.getElementById("qt_player1").Play();
    }
    // if not game over
    if(!bGameOver)
    {
        // run Move_Ball function
        window.setTimeout(Move_Ball, 10);
    }
    // set new left position of ball object
    objBall.style.left = (objBall.offsetLeft + iHorizontal) + "px";
    // set new top position of ball object
    objBall.style.top = (objBall.offsetTop + iVertical) + "px";
}

/* end of Move_Ball function */

/* function to handle keydown event */
function Key_Down(evt)
{
    // alert(evt.keyCode);
    // set batLeft == left position of Lo Bat
    var batLeft = document.getElementById("theBat").offsetLeft;
    // depending on key pressed move a bat left or right
    switch(evt.keyCode)
    {
        // case ">" key
        case 190:
            // if
            if(batLeft < 600)
            {
                // move bat right
                objBat.style.left = (batLeft + 5 * iSpeed) + "px";
            }
            break;
        // case "<" key
        case 188:
            if(batLeft > 20)
            {
                // move bat left
                objBat.style.left = (batLeft - 5 * iSpeed) + "px";
            }
            break;
    }
}

/* end of Key_Down function */

// function to get player's names

```

```

function Get_Names()
{
    // set sPlayer_Name = prompt for name
    sPlayer_Name = prompt("Please enter a name for Top Player", "Bat");
    // set bat inner HTML = sPlayer_Name
    objBat.innerHTML = sPlayer_Name;
}

/* end of Get_Names function */

/* function to load media player objects */
function Load_Sounds()
{
    // declare and set local sHTML = empty string
    var sHTML = "";
    // set sHTML = sHTML concat "media objects, etc"
    sHTML = sHTML.concat('<object classid="clsid:02BF25D5-8C17-4B23-BC80-
D3488ABDDC6B" ',
        'codebase="http://www.apple.com/qtactivex/qtplugin.cab"> ',
        '<embed id="qt_player1" src="pong.mp3" width="0" height="0" ',
        'autoplay="false" controller="false"></embed></object> ',
        '<embed id="qt_player2" src="blip.mp3" width="0" height="0" ',
        'autoplay="false" controller="false"></embed></object> ');
    // set element id "qt" innerHTML = sHTML
    document.getElementById("qt").innerHTML = sHTML;
}

/* end function Load_Sounds */

/* end javascript pong.js */

```

Useful Texts

Javascript: A Programmer's Companion from the Basics through DHTML, CSS, and DOM, Stefan Koch, Wiley.

Javascript and AJAX, Tom Negrino and Dori Smith, Peachpit Press.

spring into HTML and CSS, Molly E. Holzschlag, Addison-Wesley

DHTML Utopia: Modern Web Design Using Javascript and DOM, Stuart Langridge, Sitepoint.

HTML Utopia: Designing Without Tables Using CSS, Dan Shafer, Sitepoint.

Simple Program Design, Lesley Anne Robertson , Nelson

Information and Intelligent Systems, Kevin Savage, www.edit.net.au