|  |
| --- |
| 5/10/2013 |

|  |
| --- |
| **Higher Diploma in Science in Web Technologies** |
| Rich Internet Application |
| **Technical Report on Final Project** |

**Lecturer: Philippe [Vinchon](http://www.lasvegasnevada.gov/)**

DivZap on Animal Farm



**A Rich Internet Application Game** ([**online version**](http://ria-ajax.josefzacek.com/))

Created by

**Josef Zacek** ( x11105712)

**Thomas Dowling** ( x11107049)

Table of Contents

[1 Introduction 2](#_Toc355984879)

[1.1 Rich Internet Applications 2](#_Toc355984880)

[1.2 Brief Description of the Game 2](#_Toc355984881)

[1.3 Brief Description of Technical Strategy 4](#_Toc355984882)

[1.4 Online Version of DivZap on Animal Farm 6](#_Toc355984883)

[2 Methods and Software 6](#_Toc355984884)

[2.1 Languages 6](#_Toc355984885)

[2.2 Software 6](#_Toc355984886)

[2.1 Site Architecture (Folders) 8](#_Toc355984887)

[3 Detailed Description of Functionality 9](#_Toc355984888)

[3.1 jQuery - $(document).ready() 9](#_Toc355984889)

[3.2 jQuery – Ajax 9](#_Toc355984890)

[3.3 jQuery – Play Functionality 10](#_Toc355984891)

[3.4 jQuery – slideToggle() 10](#_Toc355984892)

[3.5 jQuery - Click on Animal Image Functions 11](#_Toc355984893)

[3.6 JavaScript – playSound() 13](#_Toc355984894)

[3.7 JavaScript – Animal Sound Functions 14](#_Toc355984895)

[3.8 jQuery – makeNewPosition() 15](#_Toc355984896)

[3.9 JavaScript – countdown () 16](#_Toc355984897)

[3.10 JavaScript – popup\_window () 17](#_Toc355984898)

[3.11 jQuery – ranappear() 19](#_Toc355984899)

[3.12 jQuery – Randomly Calling ranapear() 20](#_Toc355984900)

[3.13 jQuery – animate functions 21](#_Toc355984901)

[3.14 jQuery – fadeoutin() 21](#_Toc355984902)

[3.15 Passing a Variable between Web Pages (sessvars) 22](#_Toc355984903)

[4 Detailed Description of the Game Strategy 23](#_Toc355984904)

[5 Discussion 24](#_Toc355984905)

[6 Source of Functions 27](#_Toc355984906)

[7 References 27](#_Toc355984907)

[8 Source of Images 27](#_Toc355984908)

# Introduction

## Rich Internet Applications

A rich internet application (RIA) is one that ‘approximates the look, feel and usability of a desktop application’ with two key attributes: performance and an attractive Graphical User Interface (GUI) [Deitel *et al*., 2012].

We decided to adopt these principles and build a game using only the latest Web technologies that can be played on-line with any major Web browser, and that would complete with Macintosh and PC desktop applications in terms of performance , user interface and user interactivity.

To this end, we developed a game ([DivZap](http://ria-ajax.josefzacek.com/) on Animal Farm) using only HTML5, CSS3, JavaScript, JQuery and PHP. Our game requires no plug-ins and no Adobe Flash.

## Brief Description of the Game

Briefly, the game strategy may be summarized as follows.

As pictures of sheep, pigs and dogs randomly appear on the screen (game-board), the player gains **5 points** for clicking on a sheep image, **10 points** for clicking on a pig image but incurs a penalty **of ‑ 20 points** for clicking on a dog image. All animals are equal, but some animals are more equal than others ([Orwell, 1948](http://gutenberg.net.au/ebooks01/0100011h.html)).

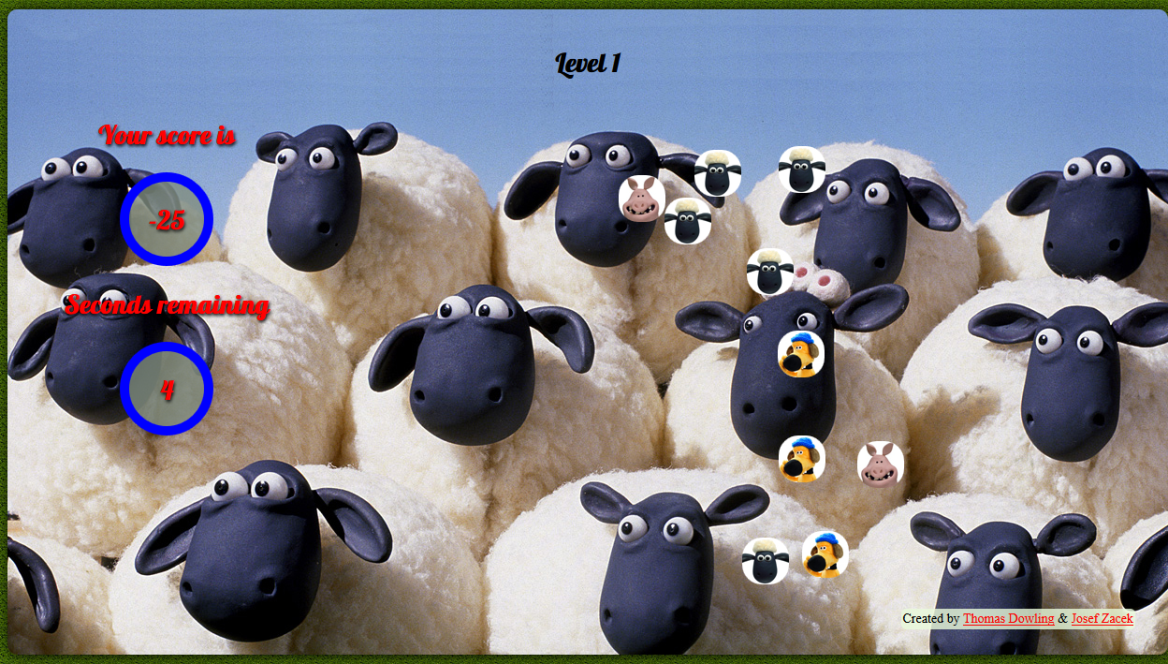
Images of the animals randomly appear on the game-board, fade in and out, and sometimes appear to move around the screen (HTML element animation).

When a player successfully clicks on an image, an appropriate sound is heard, and the score is updated. If the player obtains sufficient points, he or she may proceed to the next level.

There are five levels, each with increasing difficulty, and the player is given 15 seconds to complete each round. Thus, the game takes just 75 seconds to complete.

The cumulative score together with the time remaining is displayed on-screen at all times and the score is ‘carried over’ between levels. A screenshot of a game of [DivZap](http://ria-ajax.josefzacek.com/) in progress (level 1) is given in [Fig. 1](#FigOne_LevelOnePict).

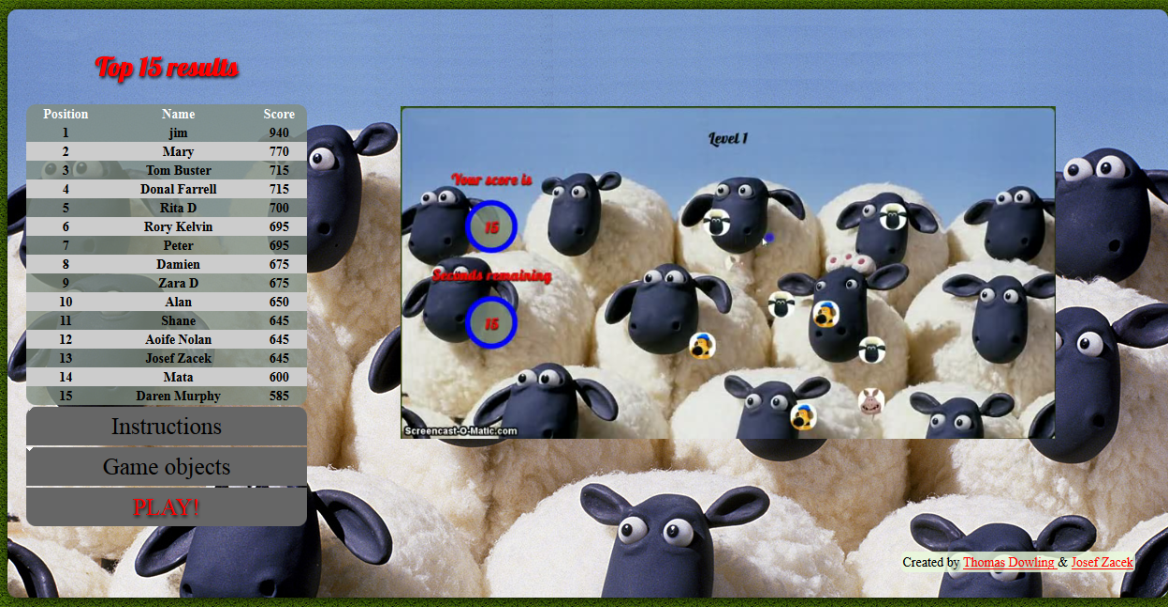
**Fig. 1. The Level One Page of** **DivZap (with Game in Progress)**



The user-interface has full instructions on how to play the game, accessible through ‘toggle panels’, and we also provide an optional video (viewable on-line from within the GUI) showing how the game should be played.

A screenshot of the index page of [DivZap](http://ria-ajax.josefzacek.com/) is given [in Fig. 2](#FigTwo_IndexPageVideoPlaying), where the introductory video is playing.

**Fig. 2. The Index Page of DivZap (with Introductory Video Playing)**



## Brief Description of Technical Strategy

A brief description of the overall strategy is as follows (A more detailed description is given in [Section 4](#DetailedDescriptionTechStrategy_4) below).

Each clickable image is contained in a HTML element, and JavaScript/jQuery is used to make images (elements) randomly appear around the game-board, to fade images in and out, to make the images appear to move around the screen (animation), and to periodically renew the game-board. In total, there are only ten such elements (five sheep, two pig and three dog elements).

In addition, JavaScript/jQuery is used to keep account of the score and to time each level. In short, the ‘engine’ of the game is provided by JavaScript/jQuery code.

HTML5 is used to create each page (level) and to create the image-containing elements upon which the JavaScript/jQuery acts.

CSS3 is used to style each page and present an attractive Graphical User Interface to the player.

Ajax is used to enable the whole game to be played on a single Web page.

PHP is used to maintain a MySQL database of the top 15 ‘all-time’ scores, to display this information to the user at the end of the game, and to allow a player to add his or her score to the database if the score is sufficiently high.

The MySQL code is as follows:

SELECT \* FROM `users` order by score DESC Limit 15

**Fig. 2. Database of Highest Scores (PHP Functionality)**



## Online Version of [DivZap](http://ria-ajax.josefzacek.com/) on Animal Farm

Our game may be played on-line at the following site

<http://ria-ajax.josefzacek.com/>

# Methods and Software

## Languages

The project was constructed using the following languages

|  |  |
| --- | --- |
| * JavaScript | *See, for example,* <http://www.w3schools.com/js/> |
| * jQuery | [jquery.com/](http://www.jquery.com/)  *(Specifically, the jQuery library available at the following site was used)*:  code.jquery.com/jquery-1.9.1.min.js |
| * HTML5 | [www.w3.org/html/wg/drafts/html/master/](http://www.w3.org/html/wg/drafts/html/master/) |
| * CSS3 | <http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/> |
| * MySQL | http://www.mysql.com/ |
| * PHP | *See, for example,* <http://www.w3schools.com/php/> |

## Software

The project was constructed on PC computers running Windows 7 using primarily the following software.

1. *Primary (Client-side) Software*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *(Version)* |  |  |
| Notepad++ | 6.3 | <http://notepad-plus-plus.org/> | Free |
| Adobe Photoshop CS5 | 12.1 | <http://www.adobe.com/ie/products/photoshopfamily.html> | Requires license |
| Adobe Dreamweaver CS5 | 11.5 | <http://www.adobe.com/ie/products/dreamweaver.html> | Requires license |

1. *Browsers*

|  |  |  |
| --- | --- | --- |
|  | *(Version)* |  |
| Firefox | 19.0 | <http://www.mozilla.org/en-US/firefox/new/> |
| (with Firebug) | 1.11.1 | [*https://getfirebug.com/*](https://getfirebug.com/) |
| (with SQLite Manager) | 0.7.7 | <https://addons.mozilla.org/en-us/firefox/addon/sqlite-manager/> |
| Chrome | 24.0.1312.57 | <https://www.google.com/intl/en/chrome/browser/> |
| Internet Explorer | 9.0.8112 | <http://windows.microsoft.com/en-US/internet-explorer/download-ie> |

1. *Servers*

|  |  |  |
| --- | --- | --- |
| [Wamp](http://www.wampserver.com/en/) |  | <http://www.wampserver.com/en/> |
| [XAMPP](http://sourceforge.net/projects/xampp/) |  | <http://sourceforge.net/projects/xampp/> |

1. *Video*

The video was produced using the following on-line functionality

|  |  |  |
| --- | --- | --- |
| [Screencast-o-matic](http://www.screencast-o-matic.com/) |  | <http://www.screencast-o-matic.com/> |

1. *Sound Format Inter-conversion*

|  |  |  |
| --- | --- | --- |
| Convert video to the OGG format |  | http://video.online-convert.com/convert-to-ogg |

We found that .mp4 files were compatible with Google Chrome, but Firefox and Opera required video files to be in .ogv format.

## Site Architecture (Folders)

The code is organized into the following folders

|  |  |  |
| --- | --- | --- |
| Root directory | * index.php | |
| Pages | * index.php * level1.php * level2.php * level3.php | * level4.php * level5.php * score.php |
| Sounds | * dog.wav * pig.wav | * sheep.wav |
| Images | * dog.jpg * pig.jpg * sheep.jpg * body\_ background.jpg * menu\_arrow.gif | * score\_logo.png * popup\_fail.jpg * popup\_pass.jpg * wrapper-background.jpg |
| JavaScript | * ajax.js * Index.js * level1.js * level2.js | * level3.js * level4.js * level5.js |
| Stylesheets | * styles.css | |
| Video | * movie.mp4 | * movie.ogv |
| MySQL | *ria\_project*  (This MySQL database is stored on the server and contains one table, holding name and score values). | |

# Detailed Description of Functionality

## jQuery - $(document).ready()

$(document).ready(function(){

// JavaScript/JQuery Code

});

The jQuery document ready event, with the above general syntax, prevents jQuery code from running until the Web page is finished loading. We have made extensive use of this event.

## jQuery – Ajax

// JavaScript Document

$(document).ready(function(){

// load index page

$('#wrapper').load('pages/index.php');

});

This jQuery function uses Ajax to ‘inject’ the contents of *pages*/*index.php* into a div with an id of *wrapper* in the root directory *index.php*.

The advantage of using Ajax in this manner is that the URL does not change when the player progresses to higher levels, and it makes it easier to hold the cumulative score between levels (see also [Section 3.15](#sessvars_315)), and the pages load faster.

This implementation of Ajax was done with the help the phpacademy video [Ajax Load Content No Page Refresh](http://www.youtube.com/watch?v=ytKc0QsVRY4), available at <http://www.youtube.com/watch?v=ytKc0QsVRY4>

## jQuery – Play Functionality

$('.play').on('click', navigate);

function navigate(e) {

e.preventDefault();

//var page = $(this).attr('href');

//alert(page);

$('#wrapper').load('pages/level1.php');

};

When a player clicks on the play button (which has a HTML class of *play*) on the index page, the *navigate()* function is called. This function loads the contents of *level1.php* into the div with an id of *wrapper* in the **root directory** *index.php****.***

In other words, this functionality starts the game.

## jQuery – slideToggle()

$(".instructions").click(function(){

$("#instructions").slideToggle();

});

The general syntax for jQuery code is *$(selector).action()*.

In the above code, clicking (action) on the HTML element with a class of *instructions* (selector) causes the *slideToggle ()* function (action) to be called on the HTML element with an id of *instructions* (selector).

The result is shown in [Fig. 4.](#Fig4_slide_toggle_functionality)

**Fig. 4.** **jQuery SlideToggle Functionality**



A similar function is used to toggle the *GameObjects* element.

$(".game\_objects").click(function(){

$("#game\_objects").slideToggle();

});

## jQuery - Click on Animal Image Functions

1. *Images of Pigs*

// pig elements

$('.**a1, .a2'**).click(function(e) {

score = score + 10;

$('#points').html(score);

$(this).hide();

$(this).show(900);

pig\_sound();

});

Within the HTML code there is a *section* element with an id of *game-board.* This contains **two child section elements** with class attributes of either *a1* or *a2*. For example:

<section class="a1">&nbsp;</section>

Elements with these class attributes (*a1* or *a2*) have a background image (*pig.jpg*) inserted using CSS (see *style.css*).

The jQuery code shown above acts on elements with class attributes of either *a1* or *a2*. That is, it acts on pig images.

When an image of a pig is clicked the following may be considered to occur:

* +10 is added to the variable score
* The score is displayed in the HTML element with an id of *points*
* The element is hidden (pig image is hidden)
* The element reappears after 900 milliseconds (pig image reappears)
* The *pig\_sound()* function is called.
* Invoking the jQuery selector with *this* ensures that *hide()* and *show()*  are only called on the pig image that is clicked.

1. *Images of Sheep*

// sheep elements

$('.a3, .a4, .a5, .a6, .a7').click(function(e) {

score = score +5;

$('#points').html(score);

$(this).hide();

$(this).show(700);

sheep\_sound();

});

A similar strategy to that described above is used to manipulate sheep elements. In this case there are six elements with class attributes ranging from *a2* - *a7*.

Clicking on an image of a sheep results in a score of +5, and images reappear after 700 milliseconds.

1. *Images of Dogs*

// dog elements

$('.a9, .a10').click(function(e) {

score = score - 20;

$('#points').html(score);

dog\_sound();

});

Clicking on an image of a dog incurs a penalty of – 20.

## JavaScript – playSound()

function playSound(str, vol) {

var snd = new Audio();

snd.src = str;

snd.volume = vol;

snd.play();

}

The function takes two arguments: the first corresponds to the sound source, and the second defines the volume.

The *playSound( )* function is a modification of that given at <http://stackoverflow.com/questions/15125528/play-audio-through-js-with-overlapping>

## JavaScript – Animal Sound Functions

The following functions call *playSound()* [[Section 3.6](#javaScript_playSound_36)] and define the sounds for each animal.

1. *Pig Sound*

var pig = "sounds/pig.wav";

/// sound functions

function pig\_sound() {

playSound(pig, 0.5);

}

1. *Sheep Sound*

var sheep = "sounds/sheep.wav";

function sheep\_sound() {

playSound(sheep, 0.5);

}

1. *Dog Sound*

var dog = "sounds/dog.wav";

function dog\_sound() {

playSound(dog, 0.5);

}

## jQuery – makeNewPosition()

// random values for position

function makeNewPosition(){

// Get viewport dimensions (remove the dimension of the div)

var h = $('#game\_board').height() - 50;

var w = $('#game\_board').width() - 50;

var nh = Math.floor(Math.random() \* h);

var nw = Math.floor(Math.random() \* w);

return [nh,nw];

}

This key function defines a new position within the HTML element *section* with an id of *game\_board.*

When this function is called the following may be considered to occur:

* The height and width of the HTML element with an id equal to *game\_board* are individually selected, 50px is subtracted from each value (so that an image of 50px will always remain within the game board boundary) and the results are independently assigned to the variables *h* (height) and *w* (width).
* The JavaScript function *Math.random()* will return a pseudo-random floating-point number between 0 and 1. Multiplying this by the variable *h* gives a floating point number between 0 and *h*. The *Math.Floor()* function converts this floating-point value to an integer and assigns the value to the variable *nh* (new height). The variable *w* is similarly transformed, and the result assigned to the variable *nw.*
* The function returns an array [*nh*, *nw*], where *nh* and *nw* are integers.

The dimensions of the HTML element with an id of *game\_board* were set to a width and height of 520px using CSS. Thus, every time *makeNewPosition()* is called, **an array of two pseudo-random integers**, each between 0 and 470, is returned.

The code is a modification of that given at <http://stackoverflow.com/questions/13774659/random-movement-in-a-fixed-container>

## JavaScript – countdown ()

function countdown(){

var count = 15;

countdown = setInterval(function(){

$("#counter").html(count);

if (count == 0) {

popup\_window();

clearInterval(countdown);

clearInterval(abcd);

}

count--;

}, 1000);

This JavaScript function provides the basic countdown facility.

The countdown begins at 15 seconds (var *count* = 15) and counts downwards to zero in steps of 1 second (1000 milliseconds).

* The result after each iteration is displayed in the HTML element with an id equal to counter (using the jQuery function $("#counter").html(count);)
* When the countdown reaches zero, three events occur: (*i*) The *popup\_window()* function is called. (*ii*) The time interval is reset to zero. (iii) the variable *abcd*, which holds the value of the time interval for random appearance of animal image elements, is set to zero (See [Section 3.12](#jquery_randomTimingFunction_312)).

## JavaScript – popup\_window ()

function popup\_window(){

var div = document.getElementById('popup\_window').style.display = 'block';

if (score > progressToNextLevelScore) {

var pop\_score = document.getElementById('pass').innerHTML = "Congratulation your score is " + score + " you have passed to the next level";

var myId = document.getElementById("startOver").style.visibility = 'hidden';

var holdScore;

sessvars.holdScore = score;

}

else

{

var pop\_score = document.getElementById('pass').innerHTML = "Sorry your score is " + score + " you need 100 to pass to the next level!";

document.getElementById("nextLevel").style.visibility = 'hidden';

};

};

The function *popup\_window( )*  may be thought of as ‘vanilla’ JavaScript which works by manipulating the DOM (see Ullmann, 2012).

*getElementById* is used to set the style of the HTML element with an id of *popup\_window* to block.

If the value of the variable *score* is greater than the value of the variable *progressToNextLevelScore*, two things happen.

Firstly, *document.getElementById('pass').innerHTML* sets the text of the HTML element with id of *pass* to the following:

"Congratulation your score is " + score + " you have passed to the next level";

Secondly, the style of the HTML anchor element (id = *startOver*) is set to hidden so that the player cannot see it.

In contrast, if the value of *score* is less than that of *progressToNextLevelScore,* the text of the HTML element with id of *pass* is set to the following;

“Sorry your score is " + score + " you need 100 to pass to the next level!”

In addition, the style of the HTML anchor element (id = *nextLevel*) is set to hidden so that the player cannot see it.

To clarify the last point. The HTML code for the popup window contains two anchor elements. The first one (*start over*) points to the index page and the second (*next level*) points to the next level page. Depending on the score the player achieves, only one of these anchor elements is visible each time the pop-up window appears.

## jQuery – ranappear()

function ranappear(){

var newp1 = makeNewPosition();

var newp2 = makeNewPosition();

var newp3 = makeNewPosition();

var newp4 = makeNewPosition();

var newp5 = makeNewPosition();

var newp6 = makeNewPosition();

var newp7 = makeNewPosition();

var newp8 = makeNewPosition();

var newp9 = makeNewPosition();

var newp10 = makeNewPosition();

$('.a1').css({top: newp1[0], left: newp1[1]});

$('.a2').css({top: newp2[0], left: newp2[1]});

$('.a3').css({top: newp3[0], left: newp3[1]});

$('.a4').css({top: newp4[0], left: newp4[1]});

$('.a5').css({top: newp5[0], left: newp5[1]});

$('.a6').css({top: newp6[0], left: newp6[1]});

$('.a7').css({top: newp7[0], left: newp7[1]});

$('.a8').css({top: newp8[0], left: newp8[1]});

$('.a9').css({top: newp9[0], left: newp9[1]});

$('.a10').css({top: newp10[0], left: newp10[1]});

};

*ranappear ()* is a key function responsible for the random appearance of the animal images (HTML elements) on the game board (strictly speaking for the random appearance of animal images within the HTML element with an id equal to *game\_board*).

The function calls *makeNewPosition ()* [[Section 3.8](#jquery_makeNewPosition_38)] ten times, once for each animal-containing HTML element and the results are assigned to separate variables. Each variable will hold an array of two random integers corresponding to positions within the HTML element with an id of *game\_board*.

jQuery is then used (*i*) to select an element of the appropriate class (*a1*, say) and (*ii*) to change the top and left positions of the selected element with the (jQuery) .css method, using the values in the appropriate array returned by *makeNewPosition()* as the basis for the change.

The net result is that every time *makeNewPosition ()* is called, *every* animal-containing HTML element is assigned a new random position within the game-board. (As explained in [Section 3.8](#jquery_makeNewPosition_38), the new positions will always be 50px less than the (original) CSS settings for height and width of the game\_board HTML element).

See also the function described in [Section 3.12](#jquery_randomTimingFunction_312) below

## jQuery – Randomly Calling ranapear()

// random function timing

abcd = setInterval(function(){

ranappear();

},3000);

};

This function calls *ranappear()*  [[Section 3.11](#jquery_ranAppear_311)] every three seconds and is responsible for the timed random appearance of the animal images on the board. (In other words, the game board is randomly renewed ever three seconds

## jQuery – animate functions

// animate div red

function a1(){

var newqa = makeNewPosition();

$('.a1').animate({ top: newqa[0], left: newqa[1] }, 3000, function(){

a1();

});

};

The function *a1* is shown above, and is called on the HTML element with a class of *a1* (*ie* it is called on the first pig image).

Similar functions are defined for all HTML elements containing animal elements (*a2* *– a10*). For clarity and brevity, just one is shown here.

The function *a1()* calls *animate()* on the HTML element with a class of *a1,* causing it to move to a new position is 3 seconds*.*  The function *animate()* itself calls *makeNewPosition ()* [[Section 3.8](#jquery_makeNewPosition_38)]to randomly determine the new position. Finally, *a1()* is a recursive function – *a1()* is called from within *a1(),* so that the HTML element appears to continuously move to a new random position every three seconds.

## jQuery – fadeoutin()

function fadeoutin(){

$('.a1').fadeOut(2000).fadeIn(800);

$('.a2').fadeOut(1000).fadeIn(4000);

$('.a3').fadeOut(1500).fadeIn(1300);

$('.a4').fadeOut(4000).fadeIn(1400);

$('.a5').fadeOut(3500).fadeIn(1200);

$('.a6').fadeOut(2000).fadeIn(1500);

$('.a7').fadeOut(4500).fadeIn(2600);

};

The function *fadeoutin* () is responsible for the fading in and fading out of the sheep and pig images independent of clicking on the image (dog images do not fade). Each animal image (governed by a different class) has different time setting for fading in and fading out.

## Passing a Variable between Web Pages (sessvars)

One problem encountered (at a very early stage) is the following: how do you go about storing the score when the player goes to the next level, and in so doing goes to a new Web page? In short, how do you pass a variable between Web pages? This problem is more difficult than it would seem at first sight.

One very elegant solution is to use [sessvars](http://www.thomasfrank.se/sessionvars.html) (session variables) [Frank, 2008]. Briefly:

* A small JavaScript file ([sessvars.js](http://www.thomasfrank.se/sessvars.js)) is embedded into the head section of every webpage that you want to pass variables between.
* This allows access to a new object called **sessvars** which works just like any other JavaScript variable except that it is passed between webpages.

For example, the JavaScript files for Webpage one and Webpage two could contain the following code:

*pageOneJavaScript.js*

var holdScore;

sessvars.holdScore = score;

*pageTwoJavaScript.js*

var newScore = sessvars.holdScore;

var score = newScore;

The principle of the method is that the DOM property windows.name survives reloading of a Web page (and can hold up to 2MB of data), and sessvars exploits this ([Frank, 2008](http://www.thomasfrank.se/sessionvars.html)).

This method was initially used to hold variables between levels (Web pages) in the development phase of [DivZap](http://ria-ajax.josefzacek.com/). However, when we introduced Ajax (see [Section 3.2](#jQuery_ajax_32)) this obviated the difficulties described above, as essentially the whole game is played on single page (a single URL), and *sessvars* was not necessary in the final version of [DivZap](http://ria-ajax.josefzacek.com/).

# Detailed Description of the Game Strategy

Having described the functions in detail, it is now possible to give a more detailed technical account of the game.

* Ajax is used to load the relevant level into *index*.*php* (root directory) [See [Section 3.2](#jQuery_ajax_32)]. Only a single URL is needed in order to play the game.
* The play functionality described in [Section 3.3](#jquery_Play_Functionality) is used to start the game.
* The JavaScript function *countdown ()* is used to time each level

(15 seconds per level)

* When the game starts and a user clicks on an image, the *click on animal image functions* described in [Section 3.5](#jquery_clickOnAnimalImagesFunctions_35) ‘kick in’, and are responsible for incrementing/ decrementing the score, for making the clicked image fade out and reappear, and for calling the appropriate sound function for each animal.
* As the game progresses, the game board is renewed every three seconds, and the animal images appear randomly over the game board. This functionality is controlled by *makeNewPosition ()* [[Section 3.8](#jquery_makeNewPosition_38)], *ranappear ()* [[Section 3.11](#jquery_ranAppear_311)] and the random timing function described in [Section 3.12](#jquery_randomTimingFunction_312).
* The random fading in and fading out of the pig and sheep images **independent of clicking** is controlled by *fadeoutin(),* described in [Section 3.14](#jquery_fadeInOut_314).
* The movement of the animal images is controlled by the animate functions described in [Section 3.13](#jquery_animate_functions_313).
* The cumulative score is held in the variable *score.* As the game is essentially played on one page (Ajax is used to ‘inject’ the relevant level into *index.php*), the difficulties of holding the score between HTML pages is obviated, and *sessvars* ([Section 3.15](#sessvars_315)) is not needed).
* The difficulty of each of the five levels is controlled by calling different functions, and different setting for the functions, on each of the five level pages.
* User options at the end of each level are controlled by *popup\_window()* [[Section 3.10](#popupWindow_310)], which allows the player, depending on his or her score, to either proceed to the next level or to start over. In addition, at the end of the game, the player may be given the option to add his or her name to the ‘all-time’ MySQL database.
* When the game ends, PHP is used to store the player’s score in a MySql database, provided that this score is in the top 15 ‘all-time’ scores.
* The *slideToggle()* functions are responsible for toggling the *Instructions* and *Game Objects* panels on the index screen

# Discussion

The World Wide Web has become a major platform for application development, especially for those that allow user interactivity. Often, however, the experience of using applications on the Web has been frustrating due to the limited capacity of Web browsers to deal with increasing complexity, the delays encountered on Web page reloading every time a request is sent to the server, and the need for dedicated plug-ins such as Adobe Flash.

Specifically, a simple interaction between Client and Server involves the following steps. The Browser generates a request to the Server, the request is received and processed by the Server and a response is sent to the Browser. The response is then rendered, causing the Browser to load a new page. With such a **synchronous request**, the client must wait for the Server to respond and cannot interact with the Browser until the entire page is reloaded [see, for example, Deitel *et al*. (2012, p 605)].

These difficulties have given rise to the concept of the World-Wide Wait (Deitel *et al.*, 2012, p 605).

As we alluded to in the Introduction, the two key features which made desktop applications more attractive than RIAs are much better performance and a more attractive GUI.

The difficulties outlined above have proved to be major barriers in Rich Internet Application evolution, and in such applications having the ‘look and feel’ of desktop applications. However, the driving force of the Internet, particularly of Web 2.0 with emphasis on user-interactivity, has led to major advances in technologies designed to overcome these obstacles.

Three such technologies come to mind.

*(i).* The introduction of CSS3, with its enhanced video, sound and animation functionality, has very often obviated the need for plug-ins such as Adobe Flash, and has made it much easier to provide an attractive GUI.

*(ii)* JavaScript libraries such as [jQuery](http://jquery.com/) has made it much easier to incorporate dynamic functionality into a Web application. Furthermore, the functionality offered by such libraries is often very sophisticated and professional-looking.

*(iii)*. The introduction of Ajax (Asynchronous JavaScript and XML) has allowed part of a Webpage to be reloaded asynchronously, allowing the user to continue to interact with the Web-page while a request is being processed by the Server (and send another request if desired).

This last technology (Ajax) has led to vastly increased RIA performance. Furthermore the difficulties of cross-browser compatibility when using ‘raw’ Ajax requests have been overcome by the introduction of Ajax toolkits (such as ASP.net Ajax Control Toolkit) [see Deitel *et al*., 2012, p 604].

Our own experience was as follows. We initially built a prototype application HTML5/CSS3, using mainly JavaScript to provide dynamic functionality. We then discovered that jQuery was much more versatile and simpler to use than JavaScript. Furthermore, the resulting code was easier to understand and manipulate. The result was that almost all dynamic functionality was coded in jQuery in the final project. In fact, all dynamic functionality is obtained using less than 150 lines of code. (The two languages, of course, are not mutually exclusive. Both may be used in conjunction with one another, as was done here).

Finally, the introduction of Ajax ([Section 3.2](#jQuery_ajax_32)) greatly improved performance and made coding easier. The difficulties encountered in holding a variable between web pages disappeared, for example ([Section 3.15](#sessvars_315)). The whole game could then be played on a single HTML page (single URL), where AJAX was used to ‘inject’ the code for the relevant level into a single Web page (*index.php).*

The final game has been posed on line at http://ria-ajax.josefzacek.com/, and may be played in any browser*.*

# Source of Functions

The *playSound( )* function ([Section 3.6](#javaScript_playSound_36)) is a modification of [that](http://stackoverflow.com/questions/15125528/play-audio-through-js-with-overlapping) given at <http://stackoverflow.com/questions/15125528/play-audio-through-js-with-overlapping>

The *makeNewPosition()* function ([Section 3.8](#jquery_makeNewPosition_38)) is a modification of that given at <http://stackoverflow.com/questions/13774659/random-movement-in-a-fixed-container>

# References

Frank, Thomas (2008) Session variables without cookies. [Internet] Available at: <http://www.thomasfrank.se/sessionvars.html> [Accessed 10th May 2013].

Deitel, P., Deitel, H. & Deitel, A. (2012) *Internet & World Wide Web. How to Program.* 5th Edn. Pearson Education Limited, Essex, UK.

Orwell (1945) *Animal Farm*. *A Fairy Story*. Secker and Warburg, London.

jQuery (2013) *Write Less. Do More* [Internet] Available at <http://jquery.com/>. [Accessed 10th May 2013].

phpacademy.org (2013) *jQuery Tutorials: AJAX Load Content No Page Refresh* [Internet] Available at <http://www.youtube.com/watch?v=ytKc0QsVRY4> [Accessed 10th May 2013].

Ullman, U (2012) Modern JavaScript. Develop and Design. Peachpit Press, Berkeley, CA.

# Source of Images

The images of the various animals were downloaded from the following sites (and were subsequently edited in Adobe Photoshop)

<http://images2.fanpop.com/image/photos/8900000/shaun-n-friends-shaun-the-sheep-8961124-1024-768.jpg>

<http://4.bp.blogspot.com/_oGKcJMc0whQ/TTCqx9CkrFI/AAAAAAAAAZ0/1rfanXT1BRo/s1600/shaun-the-sheep-relaxed.jpg>

<http://www.gamershell.com/static/screenshots/14196/315834_full.jpg>

<http://www.pulsarmedia.eu/data/media/1053/Shaun%20The%20Sheep%206.jpg>

<http://www.freewebs.com/raiyanalister/pig-bouncing.jpg>

[https://www.google.ie/search?q=shaun+sheep&client=firefox-a&hs=DSZ&rls=org.mozilla:en-GB:official&source=lnms&tbm=isch&sa=X&ei=oUWNUd6mKpCwhAe7t4CIAQ&ved=0CAoQ\_AUoAQ&biw=1241&bih=486#client=firefox-a&rls=org.mozilla:en-GB%3Aofficial&tbm=isch&sa=1&q=shaun+sheep+dog&oq=shaun+sheep+dog&gs\_l=img.3...2373.4393.0.5455.4.4.0.0.0.0.221.247.1j0j1.2.0...0.0...1c.1.12.img.I4ecHT8GqvA&bav=on.2,or.r\_cp.r\_qf.&bvm=bv.46340616,d.ZG4&fp=184f7acc92efd853&biw=1241&bih=486&imgrc=N3dwdiGZNW9S9M%3A%3BIYVlJu-Klj08FM%3Bhttp%253A%252F%252Fmedia.mightyape.net.nz%252Fimages%252Fproducts%252F20960418%252FShaun-the-Sheep-Bitzer-Dog-Small-14425556-5.jpeg%3Bhttp%253A%252F%252Fwww.mightyape.co.nz%252Fproduct%252FShaun-the-Sheep-Bitzer-Dog-Small%252F20960418%252F%3B620%3B1104](https://www.google.ie/search?q=shaun+sheep&client=firefox-a&hs=DSZ&rls=org.mozilla:en-GB:official&source=lnms&tbm=isch&sa=X&ei=oUWNUd6mKpCwhAe7t4CIAQ&ved=0CAoQ_AUoAQ&biw=1241&bih=486#client=firefox-a&rls=org.mozilla:en-GB%3Aofficial&tbm=isch&sa=1&q=shaun+sheep+dog&oq=shaun+sheep+dog&gs_l=img.3...2373.4393.0.5455.4.4.0.0.0.0.221.247.1j0j1.2.0...0.0...1c.1.12.img.I4ecHT8GqvA&bav=on.2,or.r_cp.r_qf.&bvm=bv.46340616,d.ZG4&fp=184f7acc92efd8)

<http://antokoe.com/wp-content/uploads/2012/04/shaun-the-sheep.jpg>