# Overview

* The goal was to deliver a functional solution which was to be cosmetically neat but was to be kept concise and simple.
* The tools used are HTML, CSS, JavaScript, jQuery, PHP & MYSQL.
* It has been designed to ensure as much as possible a clear demarcation of the function of each tool used.

# jQuery

* This library doesn’t offer anything that couldn’t be hand-coded in plain JavaScript & CSS. What it does offer is short-cut syntax to what would be convoluted JavaScript. This has the benefit of saving time when coding and also some more readable code when used properly.
* Syntax is pre-fixed with a dollar sign to easily identify it.
* Supports themes, which can be used to colour code features such as buttons and tabs. There are many themes available and it’s also possible to create a bespoke theme via the jQuery website. The theme I have utilised is called ‘start’ and I’ve uploaded a few others to take a look at *(‘blitzer’, ’smoothness’ ,’south-street’, ‘ui-lightness’*).
* The header section of ‘index.php’ has the links to the library, including the theme name.

# PHP Sessions

* These can be considered as global variables which are created at runtime and specific to the PC calling the program.
* Generally these values are set once and then accessed throughout a project. In this instance currently just the currently logged in User ID is kept.

# Folder Layout

**Root folder** - The hand coded programs that make up the bespoke part of the project.

**images** - Various picture files that are used for display purposes (eg coin).

**docs -** Documentation relevant to the project, (such as this one).

# Database

**users** -Valid users of the application.

ID: Unique ID (primary key)

Nam: Full Name (not currently used)

IP: IP address (set on login)

FirstActivity: Date & time of first visit to game

LastScreen: Dump of screen relevant data.

showTCP: Boolean flag indicates whether to display Total Coins Possible.

showTCC: Boolean flag indicates whether to display Total Coins Collected

showTCL: Boolean flag indicates whether to display Total Coins Lost

showCS: Booloean flag indiacted whether to display Coins Spent

**config** – General game configuration

AdminID: Administrator; triggers special menu if used to login

Blocks: Number of blocks in game

Rounds: Number of rounds in a block

BankStart: Starting number of coins in bank

Sizes: Comma separated list of collector sizes

Prices: Comma separated list of coin prices

**sequence** – Pre-defined coin drop sequences .

UserID: Valid ID as set in users table

Round: Game round number

Coins: Coins dropped.

**log\_block** – User decisions taken at each block stage

UserID: Valid ID as set in users table

Block: Block number

Size: Collector size chosen by user

Submitted: Date & time stamp of the log

**log\_round**– User decisions taken at each round stage

UserID: Valid ID as set in users table

Block: Block number

Round: Round number

CoinsAvail: Total coins available on drop

CoinsColl: Total coins collected on drop

Submitted: Date & time stamp of the log

**log\_survey** – User decisions taken at each survey stage

UserID: Valid ID as set in users table

Question: Question number (1&2 = post block, 3&4 = post game)

Answer: Numeric value of the user’s selection from the slider.

Submitted: Date & time stamp of the log

# Program Files

**index.php –** Layouts for screen segments

* Essentially an HTML file and the entry point for the application. It has an extension of PHP so that the server knows to execute PHP code on the form before loading it into a browser. Sometimes it’s necessary to code imbedded PHP into an HTML form, say for dynamic data, although this technique hasn’t been implemented here, it’s still good practice to permit it. Marking with an extension of html would prevent PHP execution.
* There are definitions and layouts here for all the elements required for user interaction. There are a number of encapsulations in <div> tags, which are switched on and off by JavaScript functions depending on the situation.

**main.js** – Entry point & central control of the whole application

* Enables & disables visible elements on the UI forms
* Requests & processes database functionality

**Classes.js** – Definitions of the object classes used.

**main.css** - Contains typical CSS values for fonts, colours etc of the various UI elements.

**data.php** - Server side hub for processing of database requests. Requests will be made by the JavaScript program, the relevant SQL is fired to the DB and the results formatted & returned.

# Client/Server

* It should be emphasised that HTML & JavaScript code is executed by the user’s browser and PHP & SQL code is executed on the server. Generally speaking PHP embedded in HTML gets performed on the server first, irrelevant of location in the file and then the resultant file is loaded into the user’s browser.
* There are two techniques for loading server side data into a browser and its good practice to decide early on which to use based on a critical mass basis, as there are pros and cons to both techniques.
  1. Building the webpage entirely on the server with a tool such as PHP.
     + PROS:
       1. Performed once.
       2. Faster processing & display on client browser.
       3. Security; code is invisible to anyone examining source code of the website.
     + CONS: HTML& JavaScript code embedded in PHP code can be difficult to read and debug.
     + Example: A website which displays daily horse race meetings has a large amount of dynamic data which changes daily (race meeting locations, race times, horse entries, jockeys etc). Due to the volume of this type of data it makes sense for the server to build the webpage with the relevant text in tabs, buttons, lists etc than multiple client browsers continually polling the server.
  2. Making AJAX/JSON requests from JavaScript to a server side process.
     + PROS: Tidy code, clear demarcation of the function of each tool.
     + CONS: Can cause screen update delays if the server or the connectivity (eg internet) has performance issues.
     + Example: The market odds for each horse in a race fluctuate on a minute by minute basis. It is therefore necessary to access a remote server for updated data on a regular basis.
* For the purposes of this project, the second technique is applied in order to access user and configuration data. This decision was made for the following reasons :-
  1. The data volumes are small.
  2. Network traffic will be light.
  3. Easier to code, debug & read.
  4. Allows administrator to alter configuration data on the fly if needs be.