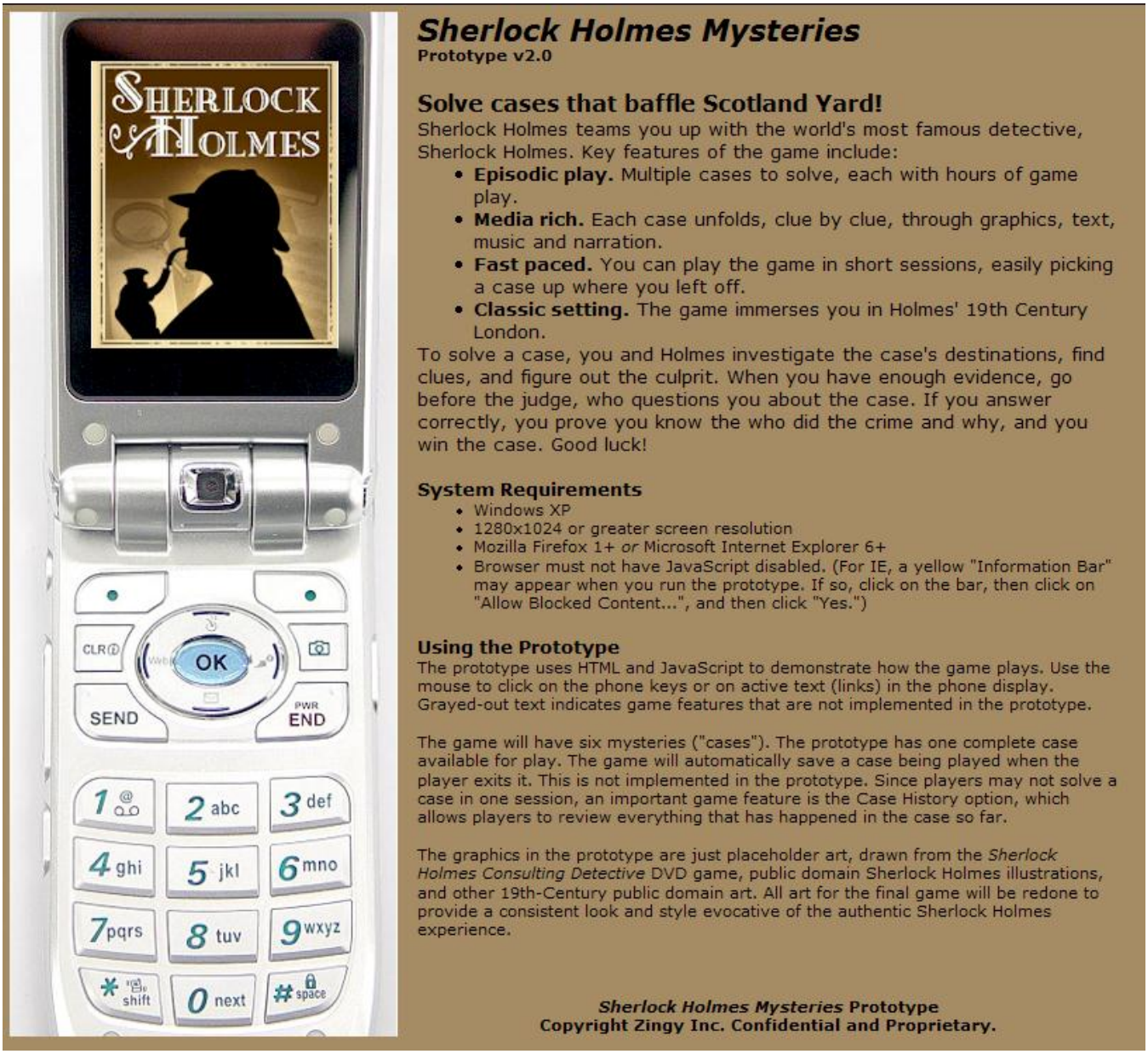
Sherlock Holmes Mysteries

v2.1 Prototype Documentation



1. Overview

This document covers how to use and modify version 2.1 of the *Sherlock Holmes Mysteries* prototype.

The document also outlines how to use reuse the underlying elements of the prototype to prototype other text-and-graphics mystery or storytelling mobile games. Indeed, elements of the prototype can be used to mock up or prototype a wide range of mobile games and applications.

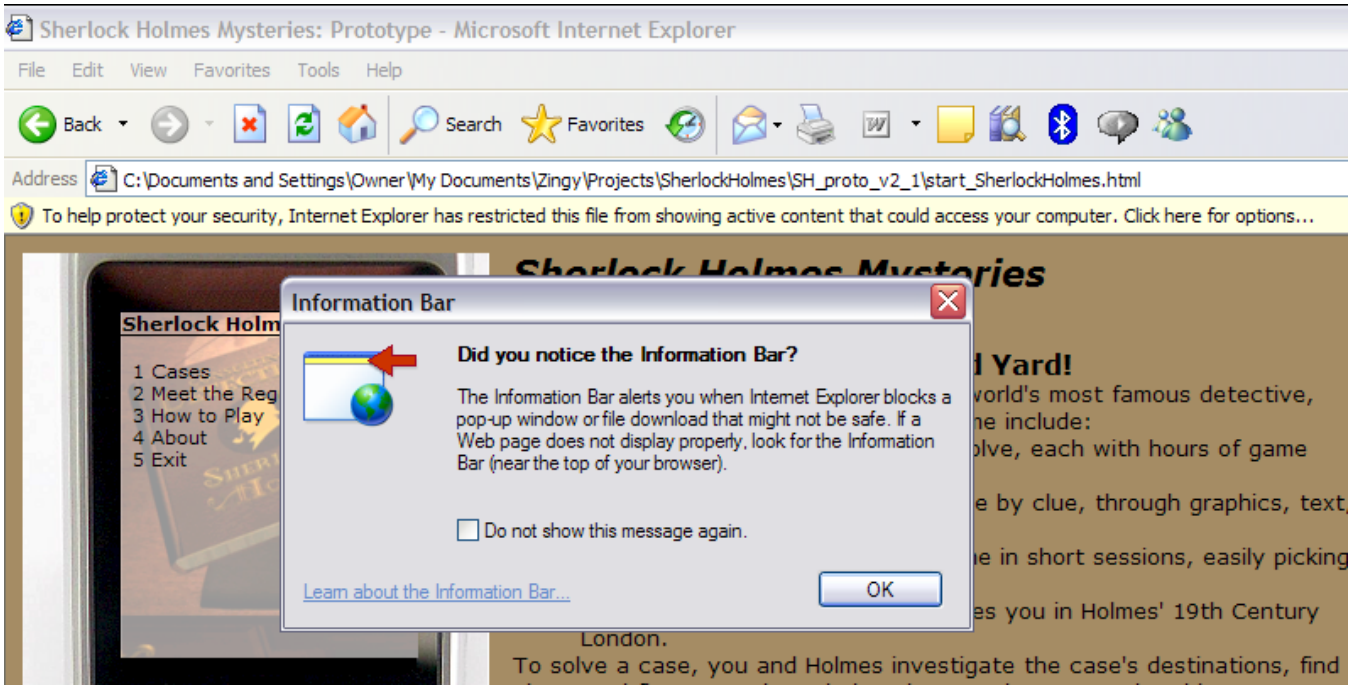
The prototype is designed so that it can be used as a demo for the game, showing prospective carriers the main features of the game and including marketing text about the game.

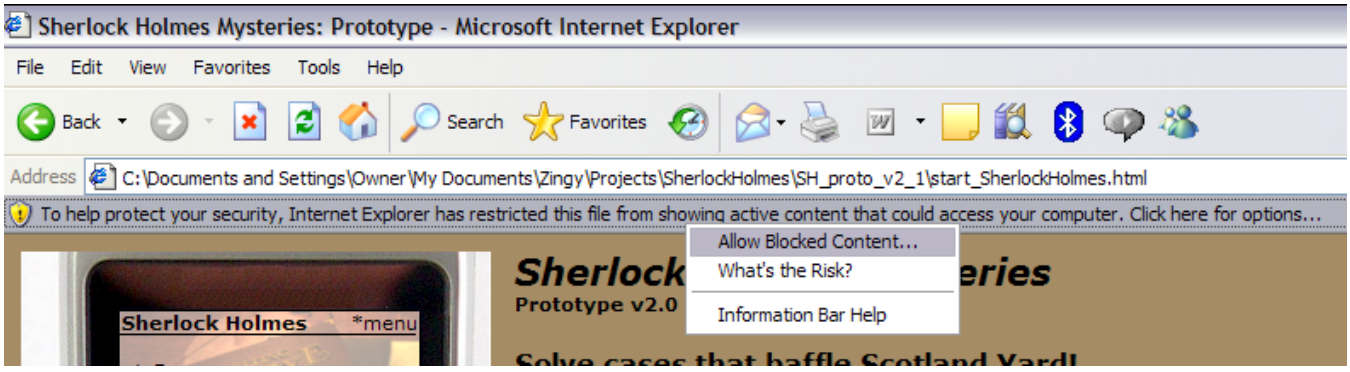
2. About the 2.1 Version Prototype

2.1 System Requirements

The prototype uses current Web technologies: HTML 4, CSS 2, and JavaScript 1.5 and must be run on a current-technology browser. The prototype has been tested on the following:

- Windows XP.
- 1280x1024 or greater screen resolution.
- Mozilla Firefox 1+ or Microsoft Internet Explorer 6+ browser.
- Browser must not have JavaScript disabled.
- Depending on Internet Explorer's security settings, a yellow "Information Bar" may appear when you run the prototype. If so, click on the bar, then click on "Allow Blocked Content...", and then click "Yes.")





Note: The prototype may run on other operating systems and on other modern standards-compliant browsers, but it has not been tested on these systems.

2.2 Staring the Prototype

Double click the start_SherlockHolmes.html file, or open start_SherlockHolmes.html from the browser.

2.3 Using the Prototype

The prototype uses HTML and JavaScript to demonstrate how the game plays. Use the mouse to click on the phone keys or on active text (links) in the phone display. Grayed-out text indicates game features that are not implemented in the prototype.

The game will have six mysteries ("cases"). The prototype has one complete case available for play. The game will automatically save a case being played when the player exits it. This is not implemented in the prototype. Since players may not solve a case in one session, an important game feature is the Case History option, which allows players to review everything that has happened in the case so far.

The graphics in the prototype are just placeholder art, drawn from the *Sherlock Holmes Consulting Detective* DVD game, public domain Sherlock Holmes illustrations, and other 19th-Century public domain art. All art for the final game will be redone to provide a consistent look and style evocative of the authentic Sherlock Holmes experience.

Players navigate through the game by making selections from menus, such as the Main Menu, and the Cases Menu.

On most screens in the game, pressing the * key calls up a menu of basic actions and options the players can select, such as navigation choices, getting help, toggling the sound on or off, and resetting the game.

When the player holds the mouse over an active area (a link) on the simulated phone's screen, the link is highlighted. This does not work exactly like highlights do on most mobile phones, but it is sufficient.

2.4 Features not Implemented in the Prototype

The following features are not implemented in the prototype:

- Automatic Save Game Ability. The game will automatically save a case being played when the player exits it. This feature could be added to the prototype by the use of cookies, but the amount of work required to do this seems excessive for the needs of prototyping or demonstrating the game. (See Adding a Save Game Feature for details.)
- The game will have narration introducing the scenes, but the prototype does not implement this. One sample of narration in included in the game. To hear it, run the prototype and go to: Main Menu -> Cases Menu -> The Mummy's Curse -> Next -> Next -> Next. Click the narration link on the right side of the screen.
- The game will have music, but this is not implemented in the prototype.
- Various in-game help screens, such as context-sensitive help and the game instructions.
- Game options relating to music, narration, and resetting the game.
- Slow Page Loads. The prototype runs quite quickly and smoothly in the browser. In actuality, various mobile phones would load the content a bit slower (or much slower), especially when data is being downloaded from the network. This could be simulated by a JavaScript function that could delay page transitions by a few seconds, but this does not seem necessary.

2.5 Cheats

There are two cheats enabled in the prototype, both on the 221B Baker Street page in The Mummy's Curse case. The cheats let you see parts of the game without playing through the entire case:

- Cheat: Start Trial lets you go before the judge even if you have not gathered sufficient evidence yet.
- Cheat: End Trial lets you win the case without having to prove that you figured out the culprits and their motives.

3. How the Prototype Works

3.1 Overall Structure of the Prototype

The prototype works by using a mix of HTML pages, JavaScript scripts and data files, and CSS formatting. The key concept is that the same HTML page can show differed content by feeding it JavaScript files that contain the actual HTML content. For example, case_dest_scene_content.html contains the template for displaying any destination scene in a case. The page is called from the Destination Menu, in such a manner so that the page loads the correct data file. For example, when the player chooses to visit the British Museum, case_dest_scene_content.html is called with the mum_data_brmuseum1a.js data file, which contains the actual HMTL code for the first part of the British Museum scene.

While this structure may sound a bit complicated in abstract, it is very powerful in that it allows the same key HTML pages to be used over and over again. Rather than having dozens of destination pages, one for each part of each scene, there is just one page. If you decided to change the layout of the destination scene, you only have to change one file, rather than dozens of different ones.

Another advantage to this approach is that the JavaScript data files can be used in different parts of the game without modification. For example, after you visit a destination, that scene is removed from the Destination Menu and is added to the Case History Menu. A case history scene page (case_hist_scene_content.html) uses the same data files as the destination scene page. Again, with separate pages for everything, you would have to two pages for a scene, one for destination scenes and one for case history scenes. If you wanted to change the content of a scene, you would have to change it in two different files. In the prototype, you only edit one file to change a scene's content.

3.2 Standards

The prototype extensively uses current Internet standards, specifically HTML 4.01, CSS 2, and JavaScript 1.5. This ensures that the prototype should work in modern, standards-compliant browsers. It has been tested and found to work in Firefox 1+ (which had very good standards compliance) and Internet Explorer 6+ (which has moderately good standards compliance).

The prototype extensively uses HTML division ( ) tags with CSS formatting to precisely position content. This greatly helps the content of the prototype to look like content on a mobile phone.

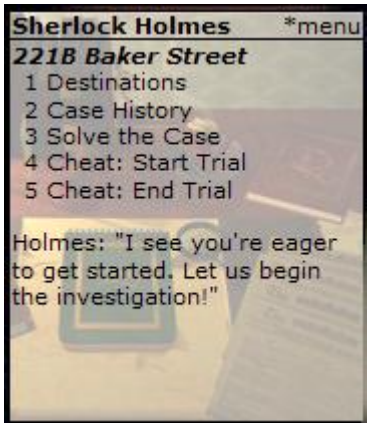
Some exceptions to the standards are used:

- The sample narrated scenes use <embed> tags to include the audio. <embed> is not an HTML 4 standard. <object> is, but Internet Explorer interprets <object> as an ActiveX control rather than as a standard HTML 4 tag. Given that many users turn off ActiveX for security reason, <embed>, which is supported in both Firefox and Internet Explorer, is used.

- The keyboard overlays use the CSS `opacity` feature to work correctly. `opacity` is a draft CSS 3 recommendation rather than a CSS 2 standard, but Firefox already supports it. Internet Explorer also supports it, with a proprietary `filter: alpha(opacity: #)` feature. The CSS file for the keypad is structured to support both features.

3.3 File Structure

Directory	File Types	Notes
`\SH_proto_v2_1`		
	`start_SherlockHolmes.html`	This file starts the prototype and contains the core HMTL and JavaScript that is used throughout the game. This file is extensively commented, explaining how things are done. Review this file before modifying the underlying structure of the prototype!
	`index.html`	This is a renamed copy of `start_SherlockHolmes.html`. It is not used unless you place the prototype on a Web server.
`\SH_proto_v2_1\files`		

	`case_XXX_content.html` `case_XXX_keypad.html`	Case-specific HTML files always start with "case". The XXX is replaced by the actual file function, such as dest_menu for the Destination Menu. There are two files for each function: a "content" file in which the game content in the simulated phone's screen is displayed, and a "keypad" file, in which the correct links for the phone keypad are stored. For example, case_bakerst_content.html contains the menu at 221B Baker Street:  while case_bakerst_keypad.html sets the keypad to the proper links:  Clicking the 1 key, for example, navigates to the Destinations Menu.
	`game_data_XXX.js`	These files contain general HTML data for the game, such as help text.

	main_XXX_content.html main_XXX_keypad.html	Top level game HTML files always start with "main". The XXX is replaced by the actual file function, such as star_menu for the top-level * menu. There are two files for each function: a "content" file in which the game content in the simulated phone's screen is displayed, and a "keypad" file, in which the correct links for the phone keypad are stored (as explained above).
	mum_data_XXX.js	YYY_data_XXX.js are JavaScript data files containing the data for the scenes in a case. The YYY is replace by an ID for the specific case ("mum" for The Mummy's Curse in the prototype) and the XXX is replaced by the ID of the specific scene. If a scene has multiple parts, each part has the same number followed by a letter: mum_data_brmuseum1a.js is the first part of the British Museum scene, while mum_data_brmuseum1b.js is the second part.
	proto_help_XXX.html	These files contain the content that appears on the right side of the phone in the prototype. Whenever a case or main content file is loaded, JavaScript in that file loads its associated proto_help_XXX.html file. Note that this mean each HTML could have its own proto_help file if so desired. Various pages in the actual prototype use the same proto_help file (and a nice JavaScript function skips loading a proto_help file if it is already loaded, as this makes page transitions smoother).
	reg_data_XXX.js	These files contain the HTML data for the regulars, with XXX IDing the specific regular.
	sample_narrated1a.html sample_narrated1b.html sample_narrated1c.html	These files contain the sample narrated scene.
	sample_unnarrated1a.html sample_unnarrated1b.html sample_unnarrated1c.html	These files display how the narrated scene would appear if the user turned off narration in the game. (The first paragraph appears as text rather than being spoken.)
	SH_style.css SH_content_style.css SH_keypad_style.css SH_proto_help_style.css	These are the CSS stylesheets for the game. SH_style.css contains the overall styles for every page in the game. The content, keypad, and proto_help stylesheets modify the overall styles as necessary for their sections of the prototype.

	TEMPLATE_BASIC_HTML_FILE.html TEMPLATE_GAME_CONTENT_HTML_FILE.html TEMPLATE_JS_DATA_FILE.js TEMPLATE_KEYPAD_HTML_FILE.html TEMPLATE_PROTO_HELP_HTML_FILE.html	These are read-only template files for the basic HTML pages and JavaScript data files in the game. Use them when creating new pages or data for the prototype. The comments in the files will guide you in using them.
\SH_proto_v2_1\files\media		
	XXX.gif XXX.jpg XXX.png	These are the graphic files for the most images in the game.
	XXX.wav	These are audio files contain the narration of the sample narrated scenes.
\SH_proto_v2_1\files\media\regulars		
	XXX.png	These are the graphic files for the images of the regulars.

By understanding the file structure, the templates, and the JavaScript (next section), you can easily add content to the prototype or change its functionality. You can use the basic structure to create new prototypes for other mobile games and applications.

3.4 JavaScript

3.3.1 Prototype-wide JavaScript

start_SherlockHolmes.html contains game-wide JavaScript. All JavaScript in this page is retained across page transitions within the subsidiary IFRAME frames, thus making this the ideal place for variables and functions used across many pages. For ease of reference, all important or often used JavaScript features are located in this page. Some minor, page-specific scripts are located in individual pages' HTML files.

Variables

```
var htmlDataFile = "";
```

This variable stores the name of the HTML data file to use when generating an intro, destination, or case history scene.

```
var caseDestTitles    = new Array();
var caseDestData      = new Array();
var caseHistoryTitles = new Array();
var caseHistoryData   = new Array();
var caseHistoryTemp   = new Array();
```

These arrays store the destination and case history data.

```
var nextDataFile = "";
```

This variable stores the next data file for multipart scenes; scripts in destinations and case history content and keypad files use this.

```
var commentHolmes = '';
```

This variable stores the Sherlock Holmes' comments displayed on 221B Baker St. These change dynamically based on the destinations the players visit. The comments are also displayed in the case history.

```
var sceneLength = "";
```

This variable tracks length of scene; allowed values are:

- `single`, a single scene
- `multi_start`, the start of a multipart scene (another part follows)
- `multi_more`, the middle of a multipart scene (another part follows)
- `multi_end`, the end of a multipart scene

The UpdateGameState function uses these values to handle multipart scenes correctly. Navigation scripts in destination and case history content and keypad files also use these values to write links correctly.

```
var triggerSolve = 0;
var triggerState = 0;
```

The `triggerSolve` value is initialized in each case, in the `InitializeCase()` function, to a value appropriate for that case. When `triggerState` reaches `triggerSolve` , the players have visited all "must-see" scenes and now can attempt to solve the case.

```
var caseSolve = 0;
```

This variable tracks whether or not the player has investigated the case enough to try to solve it. This prevents players from guessing their way through the judge's questions without playing the game. It is set to:

- `0` to prevent premature solutions
- `1` when the player may attempt to solve the case
- `2` when the player has actually solved the case (this prevents Solve the Case from running again)

```
var solveLevel = 0;
```

This variable tracks solve-scene level for navigation purposes.

```
var protoHelpFile = "proto_help_start.html";
```

This variable stores the name of the file for the proto_help_content frame. This is used in the `CheckDestTitles()` function to avoid needless page loads. "proto_help_start.html" is the initial file.

Functions

Note: Although the prototype only contains one case, the functions are written so that they can handle multiple cases. Typically, the necessary functions contain a `switch` statement, with one case block for The Mummy's Curse. Simply add case blocks for additional cases, using The Mummy's Curse code as a guide.

```
function InitializeCase( caseName )
```

This function initializes a case when the player selects one from the Cases Menu. It sets the game's variables to their appropriate values for the case.

```
function UpdateGameState()
```

This function updates the game state:

1. It manages the Case History and Destination Menus.
2. It unlocks the Solve the Case link when the player has viewed all the key scenes.
3. It is set up to allow for special functionality in multi-part scenes, although none is currently needed.

function WriteHtmlData()

This function is used by many game content pages (e.g., main_regulars_content.html or case_hist_menu_content.html) to write HTML data contained in JavaScript data files (e.g., mum_data_afahmi.js) as part of the page. It only works with content pages (the game_content pages in the IFRAMES), which the "parent.game_content" code in the function refers to.

function WriteDestMenu()

This function creates the Destination Menu for case destination menu pages, with everything linked up correctly. It is called in case_dest_menu_content.html. It contains some tricky coding to work correctly, as explained in comments inside the function.

function WriteHistoryMenu()

This function creates the Case History Menu for case history menu pages, with everything linked up correctly. It is called in case_hist_menu_content.html. It contains some tricky coding to work correctly, as explained in comments inside the function.

function CheckFooterNav(caseName, pageType, linkState)

This function makes sure the links work correctly for the foot2 link on intro scene and summary scene content pages.

function SetSolveScene(caseName, level)

This function controls how the solve-the-case sequence is run, on a case-by-case basis.

function CheckDestTitles(title)

Every destination page must have a unique title. This function ensures that a destination is only added to the destinations arrays once. (Otherwise, viewing Case History pages can cause the same destination to be added multiple times.)

function CheckHelpText(page)

This checks that the proto_help_content frame contains the correct help text.

Other Code

```
<body onLoad="if (self != top) top.location = self.location;">
```

This code in the body tag ensures that the starting page loads as the top page in the browser. If it is not the top page, then some JavaScript in the IFRAME pages may not work right.

3.3.2 JavaScript for Pages Using JavaScript Data Files

<script language="JavaScript" type="text/javascript">

```
document.write( '<script src="' + top.htmlDataFile + '" language="JavaScript"
type="text/javascript"><\/script>\n' );
</script>
```

Key HTML pages, like case_dest_scene_content.html, use JavaScript data files to read in the actual HTML content of the page. Each page contains the above script, which does the magic. The `top.htmlDataFile` variable in the script uses the value of `htmlDataFile`, which other code (such as the Destination Menu page) had already set to the correct data file name (e.g., mum_data_brmuseum1a.js).

3.5 Adding a Save Game Feature

This feature could be added to the prototype by the use of cookies to save state information between browser sessions. See http://www.quirksmode.org/js/cookies.html for a good overview on how to implement cookies.

For the prototype, a save game feature can be implemented by saving to the cookie the values of the overall JavaScript variables found in the start_SherlockHolmes.html file:

- htmlDataFile
- caseDestTitles
- caseDestData
- caseHistoryTitles
- caseHistoryData
- caseHistoryTemp
- sceneLength
- nextDataFile
- commentHolmes
- triggerSolve
- triggerState
- caseSolve
- solveLevel

(The protoHelpFile variable does not need to be saved.)

These variables change as the player plays a case, so they should be saved to the cookie each time the player navigates to a new HTML page in a case. (That way, if the play simply quits the browser without formally exiting the game, the state information is still saved.)

If Save Game is implemented, then the game should check if there is a saved game in the cookie upon initialization. If there is a saved game, the game should add a "Resume Saved Game" option to the Main Menu. Further, if the player chooses not to resume a saved game but goes to start a case, the player should be warned that doing so will erase the currently saved game.

A cookie can be no larger than 4K. Some of the JavaScript variables are arrays that store a fair amount of information and may be larger than 2K. If so, the a coding system to just store small tokens representing data like page names and files will be need to keep the cookie small.

Browsers do not always retain cookies! If the number of cookies on a browser exceeds a given number (300 for Internet Explorer), some cookies will be automatically deleted. So, unfortunately, a Sherlock Holmes cookie could be deleted without the user being aware of it.

Finally, the JavaScript should check to see if the browser has disabled cookies (trying to write a sample cookie and then reading it back is a good test). If so, then the game should advise the players to turn cookies on if they want to use the Save Game feature.

3.6 Disabling the Cheats

To produce a version of the prototype without the two cheats on the 221B Baker Street page, modify the case_bakerst_content.html file as follows:

- To completely delete the cheats, remove the following lines from the file:

```
<p class="ver08"><a onClick="top.caseSolve=1; top.SetSolveScene('mummy',1)"
href="case_solve_scene_content.html">4 Cheat: Start Trial</a></p>
<p class="ver08"><a onClick="top.caseSolve=2; top.SetSolveScene('mummy',7)"
href="case_solve_scene_content.html">5 Cheat: End Trial</a></p>
```

- To just make the cheats not appear on the page, enclose the following lines in the file with HTML comment marks, as follows:

```
<!--
<p class="ver08"><a onClick="top.caseSolve=1; top.SetSolveScene('mummy',1)"
href="case_solve_scene_content.html">4 Cheat: Start Trial</a></p>
<p class="ver08"><a onClick="top.caseSolve=2; top.SetSolveScene('mummy',7)"
href="case_solve_scene_content.html">5 Cheat: End Trial</a></p>
-->
```