**Technical Features of “Interactive Story”**

Product Documentation

System Documentation

*Compatibility*

Currently only suitable and tested on Chrome.

Files

This project consists of three interlinked files, using the languages HTML, CSS and JS (JavaScript). With the “index.html” page positioned as the central file and the other two files as a complement. Namely, the code used is sourced from the code library jQuery. In addition, the “interactiveStory.js” page consists of majority of the code work, which also includes alterations to both the HTML and the CSS page managed through JavaScript. The “interactiveStory.css” page mainly refers to any specifications with the aesthetic and user interface design.

Another three folders, referencing to the “images”, “transitions” and “sounds” folders, were created to categorise the images and audio tracks used within the game. The “images” folder contains all the backgrounds and characters, the “transitions” folder only includes sequential images of one particular transition and the “sounds” folder is for all audio tracks that is either BGM (background music) or sound effects.

Functions

There are several functions throughout the project, of which some were “global” and some were “local”. Global functions refer to functions that passes through multiple parameters and often were called many times. An example of this is the function “nextLine” which acts as the basis of all code to do with storyline progression. It determines whether the user ‘tells’ to progress to the next step, the maintenance or alteration of who is speaking the line, the image of the characters, even the playing of certain audio files, etc.

Other functions are smaller scaled, such as the “callCover” function which only specifies when and which animations would occur for one particular transition in the very beginning of the game. This function would only show the successful animation after the function “SortByValue” is called. This is because this function sorts the images within the array used in the callCover function by its value. There were also functions that is created only to simplify the code so not everything is compiled within one local domain. For instance, “secondScene” and “transitionSand1”, “transitionSand2”, “transitionSand3”, “transitionSand4”, “transitionSand5” and “transitionSand6”. Where the first function is one–time called just so the code seems less compacted, and it links in with the other six functions listed respectively. The rest are all hardcoded transitional code that animates the images in the “transitions” folder in an orderly fashion. To this extent, the “timeSwitch” function is an alternate version for the animation, which did not work as expected hence it was never called, only placed there as a guidance and innovation for similar ideas in coding.

In relation to the online database, restdb.io, the two major functions are “getCharacter” and “addUnderstanding”, with an additional minor function, “addCharacter”. “getCharacter” allows for the creation of new div elements for the characters offered during the later user selection phase. This retrieves stored information from the database and uses it during the for loop. As for “addUnderstanding” function, the principles behind are the same, but instead was the storing of information into the database rather than retrieving. And the minor function, “addCharacter” have not been called or modified, but only there as a back–up option if needed in the future.

Rest of the functions have no category it suitably fits into, but rather, all have the intention of cutting down possibilities for repeated code. The function “deleteElements” only purpose is to remove or hide multiple elements at the same time so there will be less repetitive hard code present. The function “creatingCharacter” consist of a compilation of methods relevant when the user–selected character is created. As there is no guarantee for the user to choose the intended character, therefore in order to minimise repetitive code, the function would act as an appropriate method to use. The function “audioGlobal” is coded so whenever a specific audio file that needs to be in the global domain could be easily generated without bothering too much about copy and pasting hard code. And the functions “guestConversation” and “secondCharacter” are continuations of the story progression if the user chooses the first or second character. Purely because the code would become too lengthy if it was included with the section of code straight after the user selects the character. The function “updateArray” allows for identifying the matching description for choosing the wrong choice for different characters, and through updating an empty array, only showing the user information that is relevant. And the last function “conclusion” enables a shortcut for all endings of the user selected character to be the same one without repeating too much of the same code.

Variables

It is a personal habit for all variables and elements to use camel case starting with the second word of the ID, making a more uniformed and organised classification. And all global variables are declared in sections, usually positioned above a new section of code, in which by keeping the variables nice and close, provides a clearer guide as to what is being referenced.

Arrays

All arrays present within this project starts with the prefix “arr”, which categorises and expresses arrays explicitly. There are multiple arrays used, including “arrCover”, “arrTransition”, “arrStudy”, “arrWrong” and “arrDescription”. Where “arrCover” includes the images for animation at the very beginning before the start of the game in order, “arrTransition” includes the images for transition (the files within the transition folder) in order and “arrStudy” are also the following images for a transition of the setting of a study ordered.

Comparatively, “arrStudy” and “arrWrong” refers to an empty array and an array filled with descriptions ready to be updated into that empty array. For instance, if the user chooses the wrong choice for the first character, than the function “updateArray” would be called, identifying which question is wrong and which description in the array for descriptions matches the question number. This description would then be pushed into the empty array and the empty array would then be shown on the user’s menu page (triggered when the colour of the top–right hand corner button changes).

Strings

There is quite a number of string variables within the code for the major project, though there has not been a specific prefix as such to distinguish it out. Most were just to some extent describing the function and purpose for that variable. Examples for local domain string variables and global domain string variables include:

Local

* “createHTML”: Creates HTML line to be appended to the HTML page.
* “coverID”: Represents the ID of an element that could be selected.
* “secondID”: Represents the ID of an element that could be selected.
* “restriction”: Represents the ID of an element that could be selected.
* “titleFull”: Represents the ID of an element that could be selected.
* “character”: Represents the ID of an element that could be selected.
* “hideWhat”: Represents the ID of an element that could be selected.
* “insertHTML”: Creates HTML line to be appended to the HTML page.
* “HTML”: Creates HTML line to be appended to the HTML page.
* “firstElement”: Represents the ID of an element that could be selected.
* “secondElement”: Represents the ID of an element that could be selected.
* “thirdElement”: Represents the ID of an element that could be selected.
* “fourthElement”: Represents the ID of an element that could be selected.
* “fifthElement”: Represents the ID of an element that could be selected.
* “sixthElement”: Represents the ID of an element that could be selected.
* “threeID”: Represents the ID of an element that could be selected.
* “fourID”: Represents the ID of an element that could be selected.
* “characterCropped”: Creates HTML line to be appended to the HTML page.
* “HTML1”: Creates HTML line for the character selected to be appended to the HTML page.
* “displayInfo”: Represents the ID for the identity of the character selected.
* “displayInfo1”: Represents the ID for the surname of the character selected.

Global

* “displayID”: Represents an empty string.
* “apikey”: String for storing the APIKEY of the online database.
* “url”: String for storing the URL of the online database
* “displayID1”: Represents an empty string.
* “url1”: String for storing the URL in a separate online database collection (with the same APIKEY).

There are also other variables left out of the list, generally with the prefix of “audio” at the front because it is an empty string that will later be replaced with an HTML audio element. So technically it only lasts as a string for a short period of time hence it was not included.

Objects/Classes

There are one superclass within the interactive story, which is the class “Character” and a subclass called “Female” that extends off from the superclass. Containing all of the attributes for Character while having unique ones as well. Each including methods that specifies what could be done with the created element. This was called within the function “creatingCharacter”. In using classes and object oriented programming (OOP), the long, repetitive process for creating new characters could be minimised to only a few lines. Accompanied by the presence of a function, reduces the amount of lines needed down to merely one or two lines.

Databases

This game uses one database with two collection, called “ancientchina” and “userunderstanding”. The first includes the fields, “Name” (identifying the ID of the character), “ImageReference” (image source), “Description” (brief description for personality and status), “Value” (signifies the order), “Letter” (the matching letter for the image), “Width” (the width for the image), “Top”, “Left”, “Selection” (the suffix needed to add onto the end of ‘character’ as the new ID for the element as well as the image name for the character’), “Interval” (Determines the preferred x–position of the element created), and “OriginalWidth”. Which is the values needed for generating the characters for the users to select during the selection phase. Where some of the stored values are called, used and reference and others simply act as a record for the original image used to generate the character, not the cropped double–letter images used for user selection. On the other hand, the second database collection only consists of one field, namely the “Learned” field. Which is designated for the inputs of the user appended after the user hits the submit button.