# Function List

## Splash screen:

### Drag & Drop:

dragStartHandler ( event ) {

* When a draggable element is ‘picked up’, the handler invokes the built in dataTransfer API, calling setData()
* The parameters in setData set the type of the element as an image, and the data as the element.

dragOverHandler (event) {

* When the draggable element is over a droppable element, the function calls the built in stopPropogation() function – which ‘stops the bubbling of an event to parent elements’

dropHandler (event) {

* When the draggable element is dropped in the droppable element, the function stops the default result from happening, via the built in preventDefault() function.
* The function also calls the fadeout function
* The function also changes the href from index.html to app.html, delaying the change by 1100ms
* The function also pauses the audio element, and plays a coin sound.

fadeout (el) {

* Fades the element from maximum opacity to nothing

### MP3:

play () {

* Plays the audio file

Pause () {

* Pauses the audio file

Hide () {

* Hides the audio div element, unless it is moused over.

### Canvas:

Init () {

* Calls initScroll() and initFlash()

initScroll () {

* Sets the context, colour of the font, font family, font size, where the font is positioned, and initialises some variables
* Calls scrollText ()

scrollText () {

* Sets the text to scroll from top to halfway down the page

initFlash () {

* Sets the context, colour of the font, font family, font size, where the font is positioned, and initialises some variables
* Calls textFadeUp ()

textFadeUp () & textFadeDown () {

* Sets the text to fade in and out from maximum opacity to minimum opacity

## View

Render ( model ) {

* Sets the innerHTML of all DOM elements in the HTML. Called by the model

## Controller

Init () {

* Called by the App on load
* Creates new Model and View objects
* Calls the functions register (view) and notify () from the Model
* Sets certain DOM elements to be hidden on load.

getResource () , generate () , retrieveEnglish ( optionNumber ) , butWaitTheresMore () , render () , reload () , changeQuiz () , resultsTime () , resultsOrdered () , newFile () , retryQuiz (), setName () {

* Functions called by the View element, to invoke functions held in the Model.
* See the Model function list for functionality.

## Model

### File Handler

fileChangeHandler (event) {

* Fires via an event listener on the ‘file’ input
* Error handling – if the selected file is not a .txt document, it will not load
* Calls loadedHandler (event)

loadedHandler (event) {

* Error handling – if the selected file is not correctly formatted (doesn’t have a title and doesn’t follow the [maoriWord, englishWord] structure, it will not load
* If it is a valid file, calls theUser.beginQuiz () and docChange ()

docChange () {

* Shows and hides certain DOM elements

### User class

beginQuiz () {

* Called by the File Handler
* Creates a new Quiz object
* Calls theQuiz.init()
* Starts the timer of the quiz

setName () {

* Sets the username string
* Error handling: if the input field is null, prompts the user to enter a name

getQuiz () {

* Public function, returns the private quiz

createLSNode () {

* Called automatically after the quiz finishes, and the user has entered their name
* Creates an array to be stored in local storage
* Consists of [title of the quiz, score of the quiz, the time it took, the username]
* Returns the created array (‘node’)

createLocalStorage ( title, array ) {

* Generic function to set a local storage item
* Converts the array input to a string, via the JSON.stringify function

updateLS () {

* Public function, called automatically once the quiz is finished, and the user has entered their name
* Calls createLSNode ()
* Gets the existing local storage object for that particular quiz, or creates a blank array if the quiz hasn’t been sat before
* Pushes the new history node onto the existing history
* Calls createLocalStorage, pushing the updated array back into local storage

processTime ( deltaTime ) {

* Takes the time a user took to finish a quiz, in ms, and converts it to a string of MM:ss:msmsms

createLSHistory () {

* Gets the local storage history for the quiz taken, and converts it into a string format
* Ordered chronologically

createLSRanked () {

* Gets the local storage history for the quiz taken, and converts it into a string format
* Ordered first by score, then by time (if score is the same)

### Quiz Class

setUsername (name) {

* Sets the username

getUsername (name) {

* Gets the protected username string

startTimer () {

* Creates a new Timer object

getTimer () {

* Returns the Timer for the quiz

Init () {

* Calls textSplit ()
* Takes the full array of the processed list, and pushes them into an array of possible answers

textSplit () {

* Converts a txt file into an array
* Splits the string by line breaks, then splits each line by commas
* This returns an array of [maoriWord, englishWord] for each question in the quiz
* Pops the last line of the text file to use as the title

selectResource (array) {

* Selects a random element from the array
* Removes the selected element from the array
* Returns the element

getResource () {

* Used to invoke the selectResource function, using the wordArray

getMaori (aResource) {

* Returns the Maori element of the current question

getEnglish (aResource) {

* Returns the English element of the current question

removeElementFromArray (element, array) {

* Finds the element inside the selected array, then splices it from the array

createTempArray () {

* Creates a temporary array without the current question

createFillerArray (fillerAmount) {

* Creates a filler array, comprised of (fillerAmount, which is set to 3 by default) random answers to the question, none of which are the answer
* There are some Maori words which have the same English Translation, (eg. In the COLOURS quiz, Kākāriki and Māota both translate to Dark Green)
* This function error checks, and doesn’t include multiple answers
* Returns the fillerArray

Shuffle (array) {

* Implementation of the Fisher/Yates shuffle algorithm
* Used to shuffle the possible answers

createQuestionArray () {

* Sets the variable question array to comprise of the 3 dummy answers, and the actual answer
* Shuffles the possible answers

createQuestion () {

* Creates and returns the string of the current question
* Eg. “What is the English word for Māota?”

createHistory (theQuestion, options, theAnswer, correct) {

* Called automatically when the User answers a question
* Creates a History object, saving the question, the possible answers, what the User answered, and whether it was correct

generateQuestion () {

* Function that is called to create each question in the quiz
* Calls getResource(), createFillerArray(3), and createQuestionArray()

getQuestion () {

* Returns the string of the question

retrieveEnglish (questionNumber) {

* Used to return the English element of each element of the Question array

getWordArray () {

* Returns the wordArray – the list of remaining questions that can be asked

getQuestionArray () {

* Returns the question array for a question

getTheResource () {

* Returns the resource, the array of the current question/answer

printResults () {

* Returns a String of the detailed history of the quiz
* Not invoked in the app’s current state, but left in for expansion

printAnswersOnly () {

* Called once the quiz has finished
* Returns a string of the question number, and whether it was correct or incorrectly answered

getScore () {

* Returns a string of the total score

getTitle () {

* Returns the title of the quiz

getCount () {

* Returns the number of correct answers

### History Class

getQuestion, getOptions, getTheAnswer, getCorrect, getAll () {

* Return strings of the corresponding history attributes

isTrue () {

* Boolean checker, sets the colour of the text to be blue if correct, green if incorrect

### Timer Class

Init () {

* Sets the minutes, seconds and milliseconds to 0, and the string ‘out’ to 00:00:00

startTimer () {

* Sets the startTime attribute to Date.now()

stopTimer () {

* Sets the endTime to Date.now()
* Sets the deltaTime attribute to the difference between endTime and startTime

Process () {

* Converts the deltaTime attribute, measured in ms, into a string of mm:ss:msmsms
* Returns the string

showTime () {

* Calls stopTimer () and process ()

createTimer () {

* Calls init() and startTimer()

getTime () {

* Returns a string to be used for the results screen

getDeltaTime () {

* Returns the raw deltaTime attribute, used for localstorage

### Model Functions

setSpan () {

* Sets the spanQuestion and spanTitle DOM elements to the question and its respective title

btnAnswers () {

* Sets the innerHTML for the four possible answer divs

publishQuestion () {

* Calls btnAnswers() and setSpan()

getNextQuestion () {

* Automatically called once a User answers a Question
* If there are questions remaining, get next question
* If no questions remaining, print the quiz recap, display the time taken, and hide some DOM elements

### Button Functions

butWaitTheresMore () {

* Used to generate the question, and set the innerHTML for the interface
* Invokes theUser.getQuiz.generateQuestion(), publishQuestion() and createTimer()
* Hides some DOM elements

retryQuiz () {

* Used to retry the same quiz
* Invokes theUser.beginQuiz() and butWaitTheresMore()

changeQuiz () {

* Displays the changequiz button

setName () {

* Sets the username attribute for the quiz
* Error handling, if setName returns true
* Updates the local storage history by calling theUser.updateLS()

### View Functions

Register (view) {

* Pushes the current state of the model to the view

Notify () {

* Renders the HTML elements of the view

getSpanQuestion (), getSpanOptions() , getSpanTitle (), getLinkOne (), getLinkTwo (), getLinkThree() , getLinkFour() {

* Used to set the innerHTML of the DOM elements defined in the view

clearAll () {

* Clears all of the above DOM elements’ innerHTML

Reload () {

* Controller function, calls clearAll()

### Results Window Functions

getResultsTime () {

* Creates a new window, listing the past results for the particular quiz, with the results ordered sequentially

getResultsOrdered () {

* Creates a new window, listing the past results for the particular quiz, ordered by score, then by speed