Programmer’s Manual

Talking Word Puzzles

# Vision Statement

To create an accessible crossword game for all audiences.

# Introduction

This project is intended to fulfill the need for an accessible crossword puzzle game for blind and visually impaired users as requested by the American Printing House for the Blind. We achieved this by creating a web application that allows users to create and play crossword puzzles with accessibility features to foster ease of use for visually impaired users.

# Component Overview

## Puzzle Generation

Puzzles are generated using a properly formatted XWC file and client side Javascript to dynamically create HTML elements which result in the play grid.

### Table Elements

The table input element IDs are named in a way to define where we are in the grid. The columns are letters A, B, C, etc. The rows are number 1, 2, 3, etc. This naming convention gives us the ability to determine where we are within the puzzle which will be used for arrow key navigation which is explained further in this document. Note that only the input elements are given an ID. The grayed out elements which do not represent an answer section are given no ID.

Input elements are given a data attribute (data-clueId) which is loaded with the ID(s) of corresponding clue(s) ID(s). There could be 1 or 2 clues associated with an input element and we load them all into this data attribute. This is used for the CTRL navigation feature which is explained further in the document.

Input elements are given another data attribute (data-answer) which is loaded with the one letter correct answer for that particular input. This value is used to check answers.

All elements within the table are given the CSS class “square.” All input elements are also given the CSS class “letter.”

### Clue Elements

The ID of across/down clues are given an ID based on what number their clue is. For example, the clue element for clue “1 Across” would have an ID of across1. Likewise, the clue element for clue “3 Down” would have an ID of down3. This will be important for arrow key navigation discussed further in this document.

Clue elements are loaded with a data attribute (data-clueStarts) which is loaded with the ID of the input element where this clue starts. This is used for the CTRL navigation which is explained further in this document.

### ARIA Labels

ARIA labels are used to instruct tell a screen reader what to read when focused on an HTML element. ARIA labels are put on the first input of clues. When focused, the ARIA label reads what answer it is (e.g. 1 Across) and the length of the word (e.g. 5 letters). An ARIA label is also placed on the input for the XWC file which reads “Upload XWC”. ARIA labels are not placed elsewhere because most other elements on the page are plain text which a screen reader will read on its own.

### Tab Sections

There are three main sections to the play page; the crossword puzzle, across clues and down clues. Each of these three sections is a container for their respective content. When you focus on a container, it will focus the last saved element for that section. After loading an XWC file, the last saved elements are set to a default of the first puzzle input, the first across clue and the first down clue. These values are updated in the focus listener methods of the elements inside the containers. This means the tab order would take you from the last focused puzzle input to the last focused across clue to the last focused down clue.

### Table Input Arrow Key Navigation

Arrow key navigation is set up in the checkKey method. When you are focused on a puzzle input element, the method will take the currently focused ID and the pressed key to figure out what the ID of the next element to be focused is. For example, if the current element ID is “a1” and the down key is pressed, the next element should be “b1”. The method will take the ID we created and try to focus it (in this case, “b1”). If it is a valid element, it is focused. If it is not valid (think of the case where we are at the edge of a grid and a user tries to navigate past that edge), focus will stay on the same element we started with.

### Clue Arrow Key Navigation

Arrow key navigation for the clue sections is also set up in the checkKey method. If we are focused on an across clue and press the down key, the method will attempt to find the next clue with a greater number. If the up key is pressed, we will try to find a clue that has a lower number. For example, if we are focused on “across1” and the down key is pressed, the method will try to find an “across2,” “across3,” “across4,” etc until we find a valid clue. If we find a valid clue, we focus that. If no valid clue is found (think of the case where we are at the bottom of the clue section), we stay focused on the same element.

### CTRL Navigation

CTRL is used to navigate from a clue to its corresponding input element. CTRL is also used to navigate from an input to related clue(s). This functionality is set up in the checkKey method. If CTRL is pressed when focused on a clue element, the method will look at that clue’s data-clueStarts attribute which is loaded with the ID for the first input for that clue. If CTRL is pressed from an input element, it will look at the input elements data-clueId attribute and go to any clue(s) associated with that input. If there is more than one clue associated with that input, it will go to the across clue then the down clue. It is important to note that the method will cycle through the two clues but after hitting CTRL again, you will be taken to the first input of the down clue that was focused. This feels cumbersome and confusing to users but we could not think of a way to alleviate this issue without stripping functionality of clue-> input.

### Directional Auto-Movement

The puzzle also allows for auto movement when entering answers either to the right or down. By default, it is set to across but the Javascript variable that controls which direction to move in is set when an across hint or down hint is focused. If a user is on an across hint and then goes to the puzzle, the auto-movement will go to the right. If a user is focused on a down hint and then goes to the puzzle, the auto-movement will go down.

### Auto-Highlight Input

When an input element is focused, the value that is currently in that input is highlighted. This is down by the focus listener method for input elements. This allows for easy overtyping of answers that may be incorrect. We wanted input of new answers to be an easy process rather than requiring users to select an input element and press the delete or backspace key.

### Check Answers

There are two buttons used to check answers but they both work in the same way. The listener for these buttons will look at all input elements and compare their inputted value with the value of the data-answer attribute. If these two match, that input is considered correct. If all inputs are correct when the “check answer” button is pressed, the string “Correct!” is displayed to the user. If not all inputs are correct, the string “Incorrect!” is displayed. The “show wrong answers” button works in the same way but instead of outputting a string, the incorrect inputs are given a red background.

### Upload New Puzzle

After a puzzle has been uploaded, a new puzzle can be uploaded after pressing the “upload new puzzle” button. This button will show the upload element again and the dynamically created HTML is erased. This allows users to upload a new puzzle if they desire.

## Puzzle Creation

Users can create XWC files using the puzzle creation page. The resulting XWC file can be uploaded to the puzzle generation page. Users can navigate throughout this page using “Tab” and “Shift+Tab” to reach different sections within the puzzle.

### Puzzle Form

Users can fill out input fields for Author and Puzzle-Title. Author is optional, and Puzzle-Title will be given a default name if input is left blank.

### Size

Users can input a size between 6 and 35. This will result in a puzzle grid size of n rows by n columns. If they try to input a size less than 6, the puzzle will default to a size of 6. Likewise, if they try to input a size greater than 35, the puzzle will default to a size of 35. Agile was used for this to update the grid in real time. The Lock Values button is used to make the input from Size read only. This preserves values entered into the clue grid if the user were to try to change the size of the puzzle.

### Puzzle Grid

Users can use arrow keys to navigate around the puzzle grid. Clicking anywhere in the grid area will take the cursor to where it was focused last. Furthermore, each cell inside the grid is limited to a single character input.

### Clue Box

Users can enter words as they exist in the grid followed by a “/” and then the clue. If a word is not found in the grid, an error page will be displayed. Furthermore, each new clue should be started on a new line. It is important that every clue word in the grid is accounted for in the clue box. Failure to do this may result in an inaccurate or incompatible puzzle.

### Save

The save button executes a PHP script. The values entered in the forms are posted and written to an .xwc file. The values in the grid are checked to see if a cell holds a value. A “1” is used to represent a blank cell. These values are written to the .xwc file as the layout of the puzzle. The values from the clue box are posted. Each clue word is defined before the “/” is met. These words are searched for in the grid determining their location, length, and direction. The corresponding clues are paired with these clue words. Finally, these are written to their corresponding section in the .xwc file

# Tool Overview

We utilized google docs for report management. We used notepad++ as an editor. PHP, HTML5, AngularJS 1.5, CSS, and JavaScript were used to build the project. Moqups.com to create layouts of the application. Microsoft Office for spreadsheets and documents. Batch files and scripts for automated tests.

# Project Repository

Documents were shared either through google docs or email. The project has been hosted and managed from a dedicated server from APH.

# Installation for New Install

Place the Talking Word Puzzles’ folder on a web accessible server that has access to PHP. There is no further setup required.

# Installation for New Platform

There is no data associated with the project at its current state. Follow the installation directions in the above section.

# Further Development Statement

If given more time, we would like to polish and refine the different functionalities and look of the site. The foremost issue that we would like to fix is the cumbersome “input to clue” CTRL functionality. The CTRL press will move a user to a clue associated with that input. Pressing CTRL again will put the focus on the beginning of that clue’s input section. Because we focus the beginning of the input for that clue and not the input we were last focused on, users can get confused. It appears that users prefer to go from clue->input and not the other way around so perhaps stripping the input->clue(s) functionality would be worthwhile. We would also like to save XWC files on the server side so users do not have to already have an XWC stored on their local machine to play. If we could store XWC files on the server, we could have a link on the main menu page which would preload the XWC into the “play” page. Another enhancement would be to allow users to upload their created XWC to the server to allow others to play their creation. This would require moderation of the content within the puzzle. After all, we don’t want users to upload offensive content for others to see.