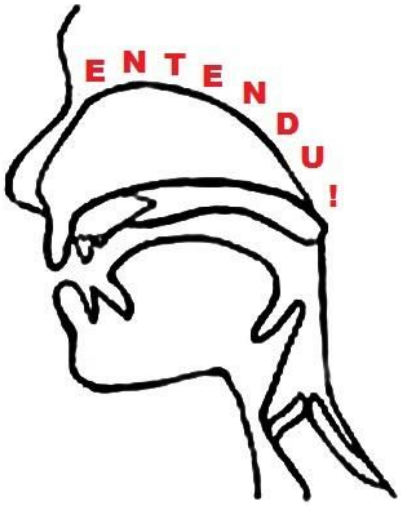
ENTENDU!

A tutorial on French Phonetics



System Design Specification

Nathan Morse and Austin O'Malley

May 4, 2017

Table of Contents

List of Figures

GitHub Repo URL: *https://github.com/nathanzm/ENTENDU*

1. Design Objective

ENTENDU! is a web application designed to be successful in satisfying the needs of both the user as well as the client. The design of *ENTENDU!* not only provides an intuitive and successful way to teach French phonetics, but it is also a convenient tool for the client to use academically and make certain changes to when necessary. College students who want to improve their French speaking skills and overall knowledge of the French language can do so with this design. The application allows students to learn and review phonemes and speech articulation through interactive lessons, quizzes, audio files, and schematic images.

The application makes efficient use of Freelancer, a Bootstrap framework which supplies the basis for design with JQuery plugins, HTML elements, Font Awesome icons, and CSS components.

The interface is cohesive across all sections of the application, so regardless of each division purpose, the theme and design provide smooth and consistent functionality through the entirety of *ENTENDU!*

2. Environment and Implementation Language

All parts of the *ENTENDU!* application were implemented using HTML5, CSS, JavaScript, and JQuery. The implementation was partially provided by Bootstrap's Freelancer framework.

All of the capabilities necessary to implement the code for *ENTENDU!* are present in most popular web browsers, which are in turn available to most operating systems. Known compatible browsers include Chrome, Firefox, and Safari. With the assumption that the browser of the user or developer is up to date with the most recent version, all functional aspects of the application will perform as intended.

If for any reason the user of the application experiences problems with the functionality of a component, this may be due to an out-of-date browser. For example, the event listener calls in a couple of our JavaScript functions, which allow the user to play and listen to audio, are not supported in the browsers Internet Explorer Version 8 and earlier versions as well as Opera Version 6.0 and earlier. We have also found problems playing audio in outdated versions of Firefox, which stems from a similar issue.

Although there are various browsers that provide many similar and comparable developer tools in addition to support for HTML, CSS, and JavaScript, some browsers are considered better than others. An updated version of Google Chrome is most supportive but not required. Mozilla Firefox would be the next most supportive browser but other popular browsers would suffice to maintain stable functionality with *ENTENDU!*'s usage.

3. Design Decisions

1. The current ENTENDU! app, as of May 4, 2017, still possesses many of the same decisions outlined in our detailed functional specifications.
2. *Quiz One* is fully randomized, and creates a new unique quiz upon refreshing the webpage.
3. Upon pressing the play button associated with each of the 6 questions that make up *Quiz One* selects 1 of the 2 possible answers to be correct and 3 of the 9 french words associated with that correct answer. The three selected french words are then played in succession for as many times as the user wants, but only before they select the answer.
4. The integrity of quizzes is maintained until the user leaves or refreshes the quiz page. For instance, if the user hits a play button on *Quiz One* more than one time, the same three french words will be played in the same order.
5. All custom functionality of and related to JavaScript that was added but not included with Freelancer was appended to the provided freelancer.js file.
6. Lessons two and three on the homepage display a “Coming Soon” message with disabled button links upon mouseover, as the content is not yet implemented.
7. Additional web pages that were not included with Freelancer were added to the same hierarchical directory structure as the included index.html and were only added when completely separate content was necessary.
8. The reading of data from image files and audio files for both the lesson and quiz is designed to only be read only from the /audio/ and /img/ directories accordingly. The images must be in Portable Network Graphic format (PNG) and the audio must be in MP3 format in order to be parsed correctly.
9. The client can easily import any audio or images without changing or adding code provided that the same file naming convention are used and the directory structure is maintained.

4. Conventions and Enhancements

All styling with the implementation languages HTML and CSS follow W3Schools HTML5 Style Guide as well as standards in the Google HTML/CSS Style Guide.

All styling with the implementation languages JavaScript and in turn JQuery follow W3Schools JavaScript Style Guide and Coding Conventions.

The guides previously mentioned all cover the topics of standardized commenting, formatting of HTML elements, whitespace, and naming conventions.

5. External Interface

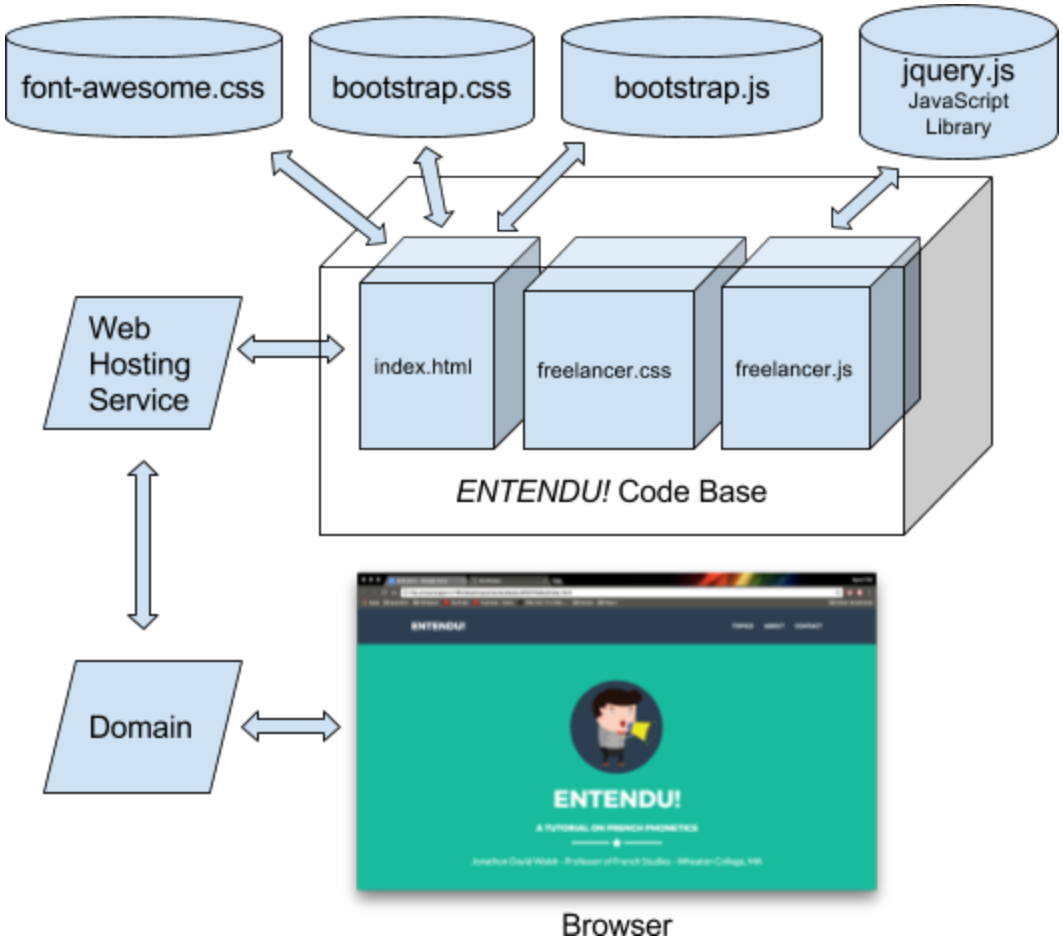


Figure 1. External Interface Diagram

5.1 External Interface - Details

a. font-awesome.css: A font and icon toolkit based on CSS and LESS. It was created with the intention that it would be used in conjunction with Twitter Bootstrap. This is exactly the way we have incorporated the use of Font Awesome in the *ENTENDU!* project.
b. bootstrap.css: Provides countless css designs for typography, forms, buttons, navigation, general formatting, and other interface components. Part of a free and open-source front-end web framework for designing websites and web applications.
c. bootstrap.js: Provides extensions to the css file that pair the styling with interactive features. Part of a free and open-source front-end web framework for designing websites and web applications.
d. jQuery: A cross-platform JavaScript library designed to simplify the client-side scripting of HTML. The main purpose for incorporating this library is to make it easier to navigate the elements in the DOM, perform animations, and to handle events. It is free and open-source using the permissive MIT license.
e. Web Hosting Service: An external service that provides a space where all internal sources and their working dependencies reside. The Web Hosting Service provides a server that enables sharing of application data and resources among multiple clients upon request through Hypertext Transfer Protocols (HTTP).
f. Domain: Serves as an address to the hosted sources; works in conjunction with the web hosting service to provide the user access to any given part of the application upon request. The domain name represents an Internet Protocol (IP), and thus acts as a pointer to the address of the server hosting the web application.
g. Browser: A graphical user interface that the user relies on to both display and use the web application. When a web page is requested and the data has loaded, the browser creates a Document Object Model (DOM) of the page that allows the user to instinctively interact with the application.

6. Structure and Integration

The HTML, CSS, JQuery, and JavaScript components all coalesce into one smooth interface with dynamic functionalities. The JavaScript functions written specifically for *ENTENDU!* perform most of the functionality required and are all located within the freelancer.js file. These functions are described in this section.

6.1 Audio Playback

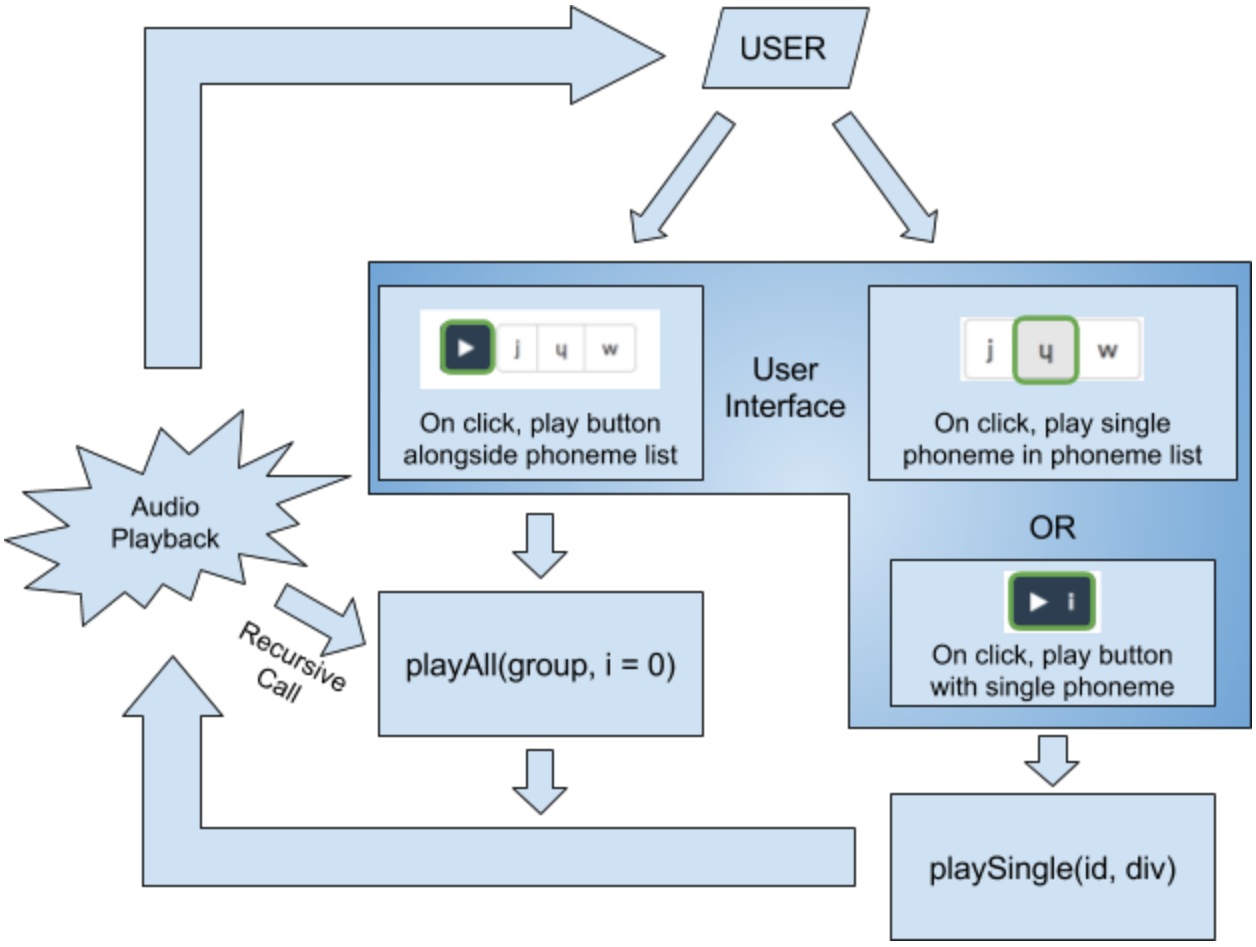


Figure 2. Structure and Integration of Audio Playback

6.1.1 playAll

This recursive function plays all sound files associated with a group that is passed in with an onClick function call. The group should be defined inside the playAll function, located in the freelancer.js file. The audio filename to be played is parsed within the function, an event listener is created to let the function know when the audio has finished playing, and then a recursive call is made.

6.1.1 a. Parameters

- ➔ IN: phoneme group
- ➔ OUT: phoneme group
- ➔ OUT: i + 1

6.1.1 b. Function Call Convention

- ➔ Example: To play vowel sounds, write *onclick="playAll('vowels')"* in the associated play button div in the HTML file, where 'vowels' is the name of the corresponding array of sounds to be played located in the freelancer.js file.
- ➔ Following the play button, the *id* of each list element should be *id="(audio filename w/o extension)(group)"*

6.1.2 playSingle

On click, this function plays a single sound file associated with the *id* value corresponding to the clicked button that is passed into the playSingle function with an onClick function call. The audio filename to be played is parsed within the playSingle function, located in the freelancer.js file. An event listener is created to toggle the button class to and from an active state.

6.1.2 a. Parameters

- ➔ IN: button id
- ➔ IN: div

6.1.2 b. Function Call Convention

- ➔ Example: To play a single audio file, the function call of the clickable HTML element should be *onclick="playSingle('(audio filename w/o extension)', this)"*.

6.2 Quiz Section

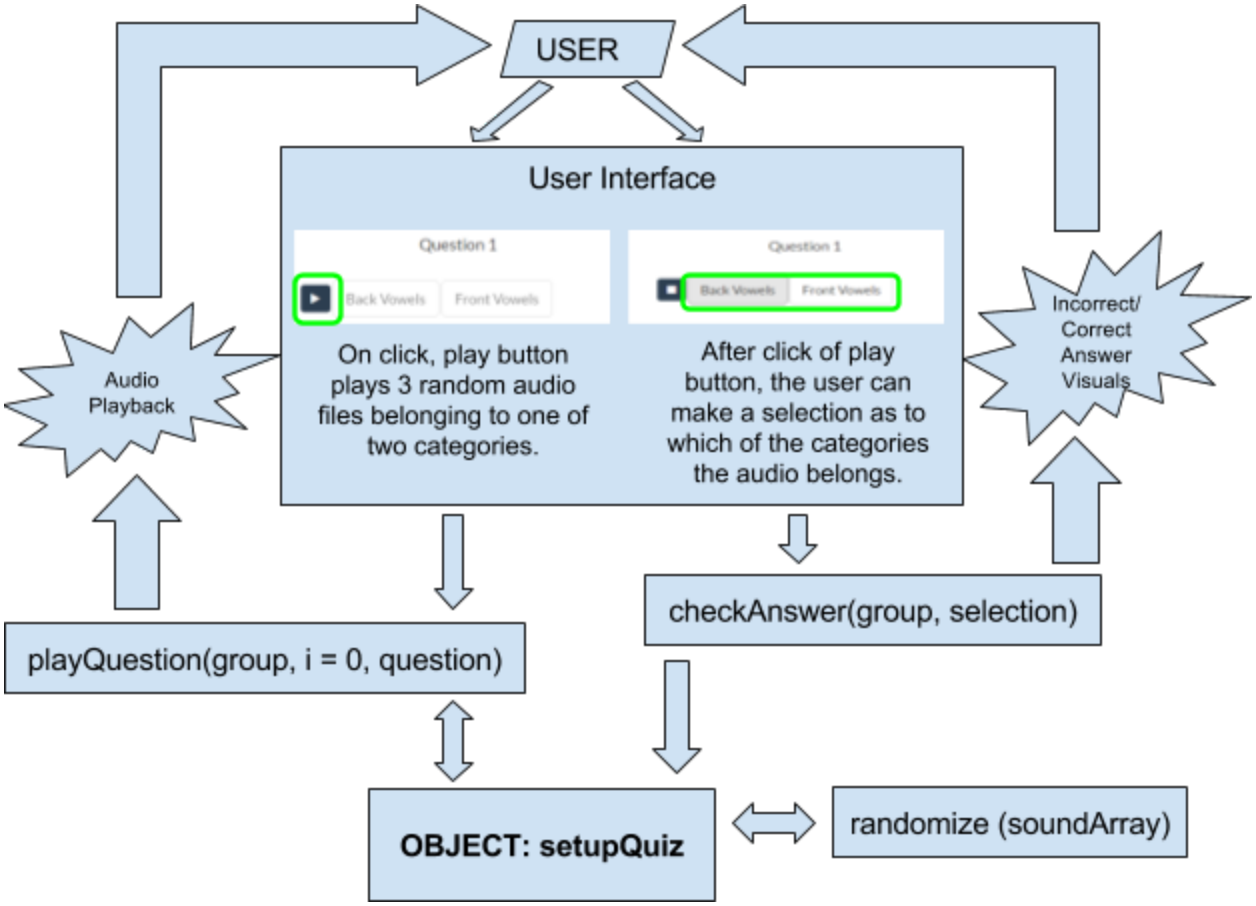


Figure 3. Structure and Integration of Quiz Functionality

6.2.1 setupQuiz

This function is invoked when the playQuestion function is called and it returns an object called returnGroup that contains a randomized set of sounds for a given group that is assigned to each question. The correct answer for the corresponding answer selection buttons is also stored in the object. This function relies on the randomize function to select which audio files are a part of the group. This object remains in memory for as long as the page is current; if the page is refreshed, the object is lost. This way, the quiz can be completely random and unique every time the page is refreshed. This function is a member of the freelancer.js JavaScript file.

6.2.2 randomize

This function takes an array of sound file names as input. It picks three random, unique elements from an array of possible choices and returns an array containing the three selected sound filenames without the extension. This function is a member of the freelancer.js JavaScript file.

6.2.3 playQuestion

This function works similarly to playAll, but bases its duties off of the returnGroup object. On click, this function checks to see if the audio group parameter matches any existing groups in the returnGroup object. If there is a match, the question being played replays the same group members as exists in the returnGroup object instead of creating a new group. If there is no match found in the returnGroup object, the function setupQuiz is called and a unique set of audio filenames is added to the object. This function is a member of the freelancer.js JavaScript file.

6.2.4 checkAnswer

This function returns either a true or false boolean value, and checks to see if the user's selection matches the correct answer for that question contained in the returnGroup object. If there is a match, checkAnswer returns true. Else, the user's answer is incorrect and checkAnswer returns false. This function is a member of the freelancer.js JavaScript file.