The web site is four pages, one at front, and three interconnected at back.

1. The front page is composed of an h1, an html form, and a login button. CSS styles the doc, giving it a pleasing palate, and form.
2. JavaScript applies functionality, creating the accepted login information, the relative path to the page to be opened should the entered information prove correct.
3. Additionally, the default text in the form is made to “dance”, by moving it into a span, and referencing the letters one at a time. They are then moved upwards through a transformation, creating a wave effect.
4. Resources and sound files are kept in two second level folders, making relative reference to them easy.
5. Each second level index is supported by a CSS and Javascript sheet. JavaScript creates an array with an index for each of the buttons the page will call.
6. Each button is given an event listener, which triggers the mp3 file associated with the button.
7. A separate method stops whatever file is playing when a new file is called.
8. Each one of the second level CSS files is unique, because one has a GIF, and two have h1 tags. In each case, the additions completely messed up the formatting dependencies. Unfortunately, the changes were apparently random in nature. They were eventually solved by individual attention, but uniquely solved.
9. The Basic library page is different from its compatriots by way of a “sound board” which has been dropped in without ceremony. In the html it exists only as it’s div tags. No other code has been placed between them. CSS defines how code should behave were it to exist in the div, and defines a new item, called ‘square’. It also defines hover behavior for this new item. JavaScript injects ‘square’ code into the div, up to a limit defined in a for loop.
10. Navigation between second level pages is handled by a standard box with three links in it. A button is created on them which is responsible for a transformation, which shrinks the box, and applies maximum opacity to the links. A problem I ran into was that the links continue to exist after their opacity is at full. I solved this problem by deactivating pointer events when the links were invisible, and reactivating them on call.