Sound Check Experiment

This is a preliminary test for making sound with javascript. Right now, the code will be tangled together in an attempt to understand how everything works. The actual program will have much more separation of parts.

Global Constants

The Lists will be constant throughout the whole program. The program will randomly select an item from one of these lists depending on the situation, and use that value to affect the sound in some way.

```
const LIST_OF_WAVEFORM_TYPES = [
    'sine', 'square', 'sawtooth', 'triangle'
];
```

Global Functions

During the creation of random sounds, randomly generating numbers is a common occurence. This function generates a random integer between two integers.

More precisely, let $f(m, M)$ be a function $f : (\mathrm{Z}, \mathrm{Z}) \to \mathrm{Z})$ such that $f(m, M) = x$ and $m \leq x \leq M$

```
function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max) + 1;
    return Math.floor(Math.random() * (max - min)) + min;
}
```

Since the most common target is a random value from one of the global Constant Arrays, a useful function will return one of those randomly selected values. This can be achieved by picking a random KEY in the array, with the minimum being 0, and the maximum being the length of the array.

```
function getRandomValueFromArray(name) {
    return name[ getRandomInt(0, name.length - 1) ];
}
```

Audio Context

Quote from Mozilla Developer - Audio Context

> The AudioContext interface represents an audio-processing graph built from audio modules linked together, each represented by an AudioNode. An audio context controls both the creation of the nodes it contains and the execution of the audio processing, or decoding. You need to create an AudioContext before you do anything else, as everything happens inside a context.

```
var audioContext = new (window.AudioContext || window.webkitAudioContext)();
```

Simple Oscillator

For this example, an oscillator node will be created. Given that we have a list of waveforms, we will choose one of them at random upon creation of the oscillator.

```javascript
var buildRandomOscillator = function () {
    var osc;
    var waveformType = getRandomValueFromArray(LIST_OF_WAVEFORM_TYPES);
    osc = audioContext.createOscillator();
    osc.type = waveformType;
    return osc;
};
```

Button Events - Start and Stop the Sound

For now, we want an oscillator to be defined globally, so that we can turn it on or off. Before any buttons are pressed, there is no sound. We just have an empty variable where the oscillator will go when it is turned on.

```javascript
var osc;
var is_sound_on = false;
```

These functions simply start and stop the sound, and are intended to be used by button events on the HTML page. This is part of the interface, and should always be present in some form, just in case somebody wants to shut off the sound.

```javascript
var startSound = function() {
    if (is_sound_on) {return;}
    osc = buildRandomOscillator();
    osc.connect(audioContext.destination);
    osc.start(audioContext.currentTime);
    is_sound_on = true;
    changeSoundStatus('True');
    changeTextWaveform('Highlight', osc.type);
};
var endSound = function() {
    if (osc == undefined) {return;}
    osc.stop(audioContext.currentTime);
    is_sound_on = false;
    changeSoundStatus('False');
    changeTextWaveform('NoHighlight', 'None');
};
```

If we are using this script with the HTML webpage (which we are), then starting or stopping the sound can change the text in the interface. If, for some reason, we aren't in the webpage, then that function simply won't return anything.

```javascript
var changeSoundStatus = function (status) {
    var span = document.getElementById('sound_status');
    span.innerHTML = status;
    span.className = status;
};
var changeTextWaveform = function (status, waveform) {
    document.getElementById('waveform_type').innerHTML = waveform;
    document.getElementById('waveform_type').className = status;
};
var humanHearingCheck = function(frequency) {
    return;
};
```