Tone.js

簡単に「Web Audio API」を活用できる JavaScript ライブラリ。https://tonejs.github.io/。

「Tone.js」ライブラリは、GitHub（https://github.com/Tonejs/Tone.js）からダウンロードするか、以下の URL から利用可能。https://tonejs.github.io/build/Tone.min.js。例：<script src="Tone.min.js"></script>。

```
var synth = new Tone.Synth().toMaster();
synth.triggerAttackRelease('C5', '2n');
```

「Tone.Synth()」
　音源に接続（この場合は「Synth」という音源）。その音源を「マスター」と呼ばれる場所「toMaster()」に接続。
　これを変数「synth」に格納し、「synth.triggerAttackRelease()」で音を鳴らす。「C5」：第 5 オクターブの「C」（ハ長調のド）。「”2n”」：2 分音符、「”4n”」：4 分音符、「”8n”」：8 分音符、「”1m”」：全音符(1 小節)、相対"+4n"。

「triggerAttackRelease()」を複数行書くことで、「単音」だけでなく、「音階」（単音列）を鳴らすことができるが、音が同時にならないように、時刻（何秒後か）を指定する。

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var synth = new Tone.Synth().toMaster();
synth.triggerAttackRelease('C5', '2n',1);
synth.triggerAttackRelease('D5', '2n',3);
synth.triggerAttackRelease('E5', '2n',5);
		</script>
	</body>
</html>
```

「Tone.js」には「シーケンス制御」が可能な API も提供されている。

```
var melody = new Tone.Sequence(【関数】, 【音階の配列】).start()
```

```
var melodyList = [ 'C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3', 'C4' ];

function setPlay(time, note) {
	synth.triggerAttackRelease(note, '8n', time);
}

var melody = new Tone.Sequence(setPlay, melodyList).start();

Tone.Transport.start();
```

関数の引数「time」に何秒後に音を出すかという情報が入れられた上で呼び出される。音符ごとに時刻を割り振ることで「シーケンス」で制御された「音階」が再生される。

音階は、周波数を表す数字（例：440）、音階＋オクターブを表す文字列（例：”F#6”、"Bb4"）。

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var melodyList = ['C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3', 'C4'];
var synth = new Tone.Synth().toMaster();

function setPlay(time, note) {
	synth.triggerAttackRelease(note, '8n', time);
}

var melody = new Tone.Sequence(setPlay, melodyList);
melody.start();
melody.loop = 2;
Tone.Transport.start();
		</script>
	</body>
</html>
```

コード(和音)

複数の「音」を同時に鳴らすために、「パート制御」ができる専用の API が用意されている。

```
var melody = new Tone.Part(【関数】, 【コードの配列】).start();
```

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var C_chord = ['C4', 'E4', 'G4', 'B4'];
var D_chord = ['D4', 'F4', 'A4', 'C5'];
var G_chord = ['B3', 'D4', 'E4', 'A4'];

var chordMelody = [
   ['0:0:2', C_chord],
   ['0:1:0', C_chord],
   ['0:1:2', D_chord],
   ['0:2:2', C_chord],
   ['0:3:0', C_chord],
   ['0:3:2', G_chord]
];

var synth = new Tone.PolySynth().toMaster();

function setPlay(time, note) {
	synth.triggerAttackRelease(note, '16n', time);
}

var melody = new Tone.Part(setPlay, chordMelody);
melody.start();
melody.loop = 2;
Tone.Transport.start();
		</script>
	</body>
</html>
```

左側に「発音タイミング」を記載している。:「小節:拍:拍内小拍」

「音源」のオプション

new Tone.Synth(option).toMaster();

オプションの中身については、各音源によって異なる。「Synth」の場合：

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var option = {
  oscillator: {
    type: "square"//"sine"//"triangle"
  },
  envelope: {
    attack: 0.005,
    decay: 0.1,
    sustain: 0.3,
    release: 0.5
  }
};

var synth = new Tone.Synth(option).toMaster();
synth.triggerAttackRelease('C5', '2n');
		</script>
	</body>
</html>
```

音源：Synth、MonoSynth、FMSynth など。音源 PolySynth は、和音用。

「エフェクト」

「リバーブ」「フィルター」「コーラス」など。

```
var reverb = new Tone.Freeverb().toMaster();
synth.connect(reverb);
```

```
<!doctype html>
<html>
	<head>
		<meta charset="utf-8">
	</head>
	<body>
		<script src="Tone.min.js"></script>
		<script>
var melodyList = ['C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3', 'C4'];
var synth = new Tone.FMSynth().toMaster();

var reverb = new Tone.Freeverb().toMaster();
synth.connect(reverb);

function setPlay(time, note) {
	synth.triggerAttackRelease(note, '32n', time);
}

var melody = new Tone.Sequence(setPlay, melodyList);
melody.start();
melody.loop = 2;
Tone.Transport.bpm.value = 240;
Tone.Transport.start();
		</script>
	</body>
</html>
```