app.js

```
const audioModel = new AudioModel();
const model = new Model(audioModel);
const view = new View(audioModel);
const controller = new Controller(model, view);
const midiController = new MidiController(controller);
const touchController = new TouchController(controller);
```

MidiController

```
controller = controller;
midiAccess = null;
deviceName = "Arturia MiniLab mkII";
```

```
initializeMIDI(): null
setupMIDIInput(): null
handleMIDIMessage(event): null
```

Controller

```
model;
view;
----------
mainFrame = document.getElementById('mainFrame');
display = document.getElementById('display');
knobs = document.querySelectorAll('.knob');
displayMenu = document.getElementById('displayMenu');
knobMenu = document.getElementById('knobMenu');
----------
presetOptions = document.getElementById('presetOptions');
pedalOptions = document.getElementById('pedalOptions');
presetSelect = document.getElementById('presetSelect');
pedalSelect = document.getElementById('pedalSelect');
knobElements = document.querySelectorAll('.knob');
----------
keys = document.querySelectorAll('.key');
blacKeys = document.querySelectorAll('.blacKey');
displayPads=document.querySelectorAll('.pad');
```

```
- flipLed(id): null
- flipKey(id): null
- flipPad(idx): null
----------
- preventRightClick(): none
- displayContextMenu(): none
- knobContextMenu(): none
- presetSelectClick(): none
- pedalSelectClick(): none
- documentClick(): none
- handleContextMenu(event, view, contextMenu): none
----------
```

View

```
audioModel=audioModel;
canvas = document.getElementById('signalCanvas');
ctx = this.canvas.getContext('2d');
frequency = 0.02;
phase = 0;
centerX = this.canvas.width / 2;
```

```
flipLed(led): null
flipKey(keyElement, keyType): null
flipPad(padElement): null
----------
hideContextMenu(contextMenu): null
showContextMenu(contextMenu, x, y): null
----------
rotateKnob(knob, rotation): null
----------
drawSinusoidalSignal(): null
animateSinusoid(): null
```

TouochController

```
controller = controller;
initialValue = 0;
attachKeysEventListeners(this.controller.keys);
attachKeysEventListeners(this.controller.blacKeys);
```

```
attachKnobEventListeners(): null
attachKeysEventListeners(keys): null
attachPadsEventListeners(): null
```

Model

```
audioModel = audioModel;
knobsLevel = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
pads = [0, 0, 0, 0, 0, 0, 0, 0];
pressedKeys = {};
sustainedNotes = {};
isSustainPedalDown = false;
```

```
updateKnobLevel(idx, value): null
handleControlChangeEvent(controllerNumber, value): { knob }
refreshAudioParameters(): null
----------
handleSustain(note): null
handleNoteOn(note): { display, ledSelector, keySelector }
handleNoteOff(note): { display, ledSelector, keySelector }
handlePadOn(note): boolean
handlePadOff(note): { display, padIndex }
```

AudioModel

```
context = new AudioContext();
----------
mainGain = this.context.createGain();
keyGain = this.context.createGain();
drumGain = this.context.createGain();
analyser = this.context.createAnalyser();
analyser.fftSize = 256;
compressor = this.context.createDynamicsCompressor();
----------
keyGain.connect(this.mainGain);
drumGain.connect(this.mainGain);
mainGain.connect(this.analyser);
analyser.connect(this.compressor);
compressor.connect(this.context.destination);
----------
bufferLength = this.analyser.frequencyBinCount;
dataArray = new Uint8Array(this.bufferLength);
attackNote = 0.01;
releaseNote = 0.10;
```

```
getAmplitude(): amplitude
setMainGain(value): null
setKeyGain(value): null
setDrumGain(value): null
----------
playNote(note): { oscillator, gainNode }
stopNote(note): null
playKick(): null
playSnare(): null
playClosedHiHat(): null
playCrashCymbal(): null
```