# Touch MIDI v2

Flexible HTML5 based touch based control surface for MIDI

View on GitHub Device Layouts

## Developer Guide

This is a brief reference guide to creating a new layout HTML file.

### Getting Started

Create an empty HTML file and paste the contents below.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Touch Midi [My Layout]</title>
    <script src="https://cdn.jsdelivr.net/gh/benc-uk/touchmidi@v2
  </head>
  <body>
    <!-- PUT YOUR CONTROLS HERE -->
  </body>
</html>
```

All you need to include in the HTML is the bundled JavaScript in the HTML head, this is served from the jsDelivr CDN, e.g. `https://cdn.jsdelivr.net/gh/benc-uk/touchmidi/dist/bundle.js`

Note. You can specify any version tag (see releases) after the touchmidi part, e.g. `touchmidi@v2` or `touchmidi@v2.2.0` or use `touchmidi@latest` for the most recent released build

Note. Jan 2020 - There seems to be a bug with jsdelivr CDN and they cache old versions for a VERY long time, it is strongly suggested to use the full version tag, e.g. `touchmidi@v2.2.0` in order to get the correct version.

# Control Reference

## Sliders & Encoders

```
<midi-slider></midi-slider>

<midi-encoder></midi-encoder>
```

Creates a slider or encoder control, which holds an underlying value to be sent as CC or NRPN messages. Sliders and encoders start with their value equal to their minimum (zero by default)

Attributes:

- `cc` - Makes this control send CC messages to the given CC number, e.g. `cc="71"`.
- `nrpn` - Makes this control send NRPN messages to the given NRPN pair, e.g. `nrpn="64,5"`.
- `pitch-bend` - Makes this control send pitch bend. A valueless bool. Note. Sliders only. `<midi-slider pitch-bend>`.
- `min` - Minimum value this control can hold, default is `min="0"`.
- `max` - Maximum value this control can hold, default is `max="127"`.
- `horizontal` - Switches sliders to horizontal orientation, for encoders changes the movement direction to "turn" the encoder to left-right (it's up-down by default)
- All common attributes, see below

One of either `cc` or `nrpn` should be specified or the control will do nothing. Both can be specified and the slider will send the value as both CC and NRPN.
If a slider or encoder set to send NRPN also has a max set higher than 127, then a high resolution (14 bit) NRPN value will be sent

Some examples:

```
<!-- An lime green encoder to control CC 74 -->
<midi-encoder cc="74" label="cutoff" colour="lime"></midi-encoder

<!-- A slider to control CC 7 with a dynamic label -->
<midi-slider cc="7" label="%v"></midi-slider>
```

```
<!-- An encoder which sends a NRPN -->
<midi-encoder nrpn="61,18" colour="#ff00ff"></midi-encoder>

<!-- A horizontal slider with min and max-->
<midi-slider horizontal cc="83" min="20" max="64"></midi-slider>
```

### Buttons

```
<midi-button></midi-button>
<midi-button toggle></midi-button>
```

Creates a button control, which can trigger MIDI messages when pressed and also when it is released. Buttons can be momentary or toggled.

Attributes:

- `cc` - When clicked this button sends the given `value` to this CC number, e.g. `cc="71"`
- `nrpn` - When clicked this button sends NRPN messages with the given `value` to the given NRPN pair, e.g. `nrpn="64,5"`
- `prog` - When clicked this button sends program change messages with the given `value` and a pair bank select messages, e.g. `prog="1,7" value="44"`
- `note` - When clicked this button sends a MIDI note on message to this number, using the `velo` as velocity. e.g. `note="48"`.
  When the button is released a MIDI note off message is sent.
- `velo` - Velocity when sending notes, e.g. `velo="66"`, the default is 127.
- `value` - The value to send when sending CC or NRPN messages, e.g. `value="88"`.
- `value-off` - When specified, the button upon being released sends another MIDI CC or NPRN message with this value. Default is none
- `nrpn-hires` - Boolean. Send the NPRN value using high resolution (14 bit) messages
- `toggle` - Boolean. The button becomes a toggle, and will not send note off or value-off messages when first released, until clicked a second time. This can be used to hold (latch) notes or create a mute type button.
- All common attributes, see below

One of either `cc`, `nrpn` or `note` should be specified or the control will do nothing. If multiple are supplied then multiple messages will be sent (e.g. a CC message *and* and note)

Some examples:

```
<!-- A button that plays middle C -->
<midi-button note="60" velo="88" label="mid-C" colour="#00ff00"><

<!-- A button that opens the filter cutoff fully -->
<midi-button cc="74" value="127" label="filter"></midi-button>

<!-- A mute button toggle -->
<midi-button toggle cc="7" value="0" value-off="127" label="mute'
```

## XY Pad

```
<midi-pad></midi-pad>
```

Creates a 2D touch pad type control, which holds an underlying pair of values to be sent as CC. Pads start with their value equal to their minimum (zero by default)

Attributes:

- `cc-x` - Makes this control's x-axis send messages to the given CC number, e.g. `cc-x="71"`.
- `cc-y` - Makes this control's y-axis send messages to the given CC number, e.g. `cc-y="74"`.
- `min` - Minimum value this control can hold, default is `min="0"`.
- `max` - Maximum value this control can hold, default is `max="127"`.
- All common attributes, see below

Note both x & y values share the same `min` and `max`

Some examples:

```
<!-- A XY pad to control filter cutoff and resonance -->
<midi-pad cc-x="74" cc-y="71" label="filter\n%v" colour="magenta'
```

## Common Attributes

All controls accept the following attributes.

- `chan` - See channel configuration section below.
- `colour` - Colour of this control, can be HTML hex code (e.g. `#ff44bb`) or a colour name, choose bright & light colours. The default colour is white
- `label` - Label to display on the control, can be a static string or a dynamic formatted value. see label formatting section below.
- `label-scale` - Scaling factor to apply to the label font size (e.g. `0.5` or `1.2`)
- `value-offset` - Offsets the value when displayed as a label
- `grow` - Make this control larger than others in the group, must be a positive integer, default is 1.

## Layout Containers & groups

### Row

Create a container which lays out controls in a horizontal row using `<group-row>`. It is essentially a div set to `display: flex` and `flex-direction: row`

```
<group-row>
  <midi-slider cc="1"></midi-slider>
  <midi-slider cc="2"></midi-slider>
</group-row>
```

### Column

Create a container which lays out controls in a vertical column using `<group-column>`. It is essentially a div set to `display: flex` and `flex-direction: column`

```
<group-column>
  <midi-slider cc="1"></midi-slider>
  <midi-slider cc="2"></midi-slider>
</group-column>
```

## Row & Column Attributes

Both `<group-row>` and `<group-column>` share some common attributes.

- `grow` - Make this container larger than others inside it's parent container, must be a

positive integer. Default is 1
- `margin` An additional margin around this container, in `rem` CSS units. Default is 0.
- `border` : Boolean. If specified, draw a border around this group. Default is false and no border.
- `colour` : What colour to draw the border, if border is enabled. Default is white

## Channel Configuration

As described in the user guide layout files can operate in two different ways with respect to MIDI channels. This is likely to change in a future release.

### Basic - Single Channel

If designing a layout to control a single device which responds on a single channel (i.e. is mono-timbral) then you do not need to specify any channels (i.e. `chan="6` ) on the controls. Simply omit them and allow the user to pick a global MIDI channel at startup

### Advanced - Multi Channel

If designing a layout to control either a device which responds on multiple channels (i.e. is multi-timbral) or you want to control multiple devices, then you must specify the channel using `chan=` on every control. Any controls without a channel will default to channel 1. Note. The user can still pick a global channel and override the settings inside the HTML.

## Label Formatting

Labels can take the form of a static string, or display a dynamic value taken the control's parameters. This is done with special flags and specifiers in the label string. These formatters are substituted as follows:

- `\n` becomes a line break
- %v becomes the control's value (on a XY pad will be comma separated)
- %p becomes the control's value as a percentage between min & max
- %h becomes the control's channel parameter
- %c becomes the control's CC parameter
- %n becomes the control's note parameter shown as a number
- %a becomes the control's note parameter shown as a name
- %% becomes the '%' symbol

If omitted a label will fall back to showing the value or note number depending on the control

# Advanced

Local development notes.
This project's dev environment uses Node, NPM and Webpack plus some other tools.

To work locally:

```
npm install
```

NPM scripts |Script Name|Purpose| |-|-| | `npm run watch` |Run webpack bundling with hot reload & dev mode| | `npm run build` |Create production minified bundle| | `npm run lint` |Run linting checks with ESLint| | `npm run format` |Run code format checks with Prettier|

GitHub actions is setup to run a build `npm run build` and push the results to the `latest` branch. No development work should take place on this branch

---

touchmidi is maintained by benc-uk.
This page was generated by GitHub Pages. Using the MS Dark Theme (Ben Coleman, 2018)