**DrumKit**

http://keycode.info/ is a little website that shows the keyCode of a key when pressed

data attribute

data-key is not standard. In this example the data-key="65" is used like a class to link the audio attribute to the div attribute

<div data-key="65" class="key">

<kbd>A</kbd>

<span class="sound">clap</span>

</div>

and

<audio data-key="65" src="sounds/clap.wav"></audio>

we want to listen to the keydown event and when this happens we are going

to run this function (e) ((the event))

(in the script tag)

window.addEventListener('keydown', function (e) {

console.log(e);

<script>

**window**.addEventListener('keydown', function (e) {

**console**.log(e.keyCode)

});

</script>

when we run this event and press say A we will see in the console the keyboard event and then its object

with all its information.

if we clicked "A" and open the console and open the object we will see the keyCode for A to be 65.

(now if we change the script to find the keyCode)

**window**.addEventListener('keydown', function (e) {

**console**.log(e.keyCode)

Now when we press a key, only the keyCode will be displayed in the console.

Now we want to say if there is an audio element on the page that has a data-key of 65

And since there is no class of key 65 (class=”key-65)((which is more like a id))

So we want to use a attribute selector

const audio **=** **document**.querySelector('audio[data-key=65]')

but that is going to be a variable (65) that is going to live in the event (e) so instead we will use

ES6 template strings ( changing ‘ ‘ to backtic ` ` and using ${ } ((`audio[data-key=”${e.keyCode}”]`

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`)

now we can console log audio and see if we have selected an actual audio element.

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`)

**console**.log(audio)

now when we look in our console log and press A we get the correct audio element (as well as all the other keys that have an associated audio element)

if we tried to press Q we would get null because there is now audio element associated with 81 ((Q)).

We can right there add an if statement that says if there is no audio element then return (which will stop the function)

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

now since we have the audio element, we can play it with adding

audio.play();

I now kind of works but it can’t start over when you push it. once pushed it takes its full 2whatever seconds to play the whole sound clip.

What we want to do is be able to start the clip over again if the key is played again so before we play it we set the clip at 0 when we press the key

audio.currentTime **=** 0; *//rewind to the start*

so

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

We have successfully selected the audio element but want to also select the key so that we can add the animation feature

To do this we add a query selector (which is pretty much copied from the audio but changed for the key)

const key **=** **document**.querySelector(`key[data-key="${e.keyCode}"]`);

so

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

**console**.log(key);

we now can see in the console the class key and its corresponding keycode

now we can add the class of playing (line 34 in .css )

key.classList.add('playing');

(this is vanilla JS. If we were using Jquery it would be key.addClass(‘playing’);

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

key.classList.add('playing');

now the highlight animation engages but it doesn’t turn off to be able to engage again when the key is struck again

if you wanted to mess with the transition time and scale you can change

transition in the key class on line 26

and

transform: scale in the playing class on line 35

so now we are correctly adding the class playing to get the animation but it stays on until we remove the class

We could do it by making say, a setTimeout function that times out after .07 sec but then we are setting a timeout here in the JS as well as in the css key class transition but if we did this they could get out of sync and so we don’t want to code it this way because if we wanted to change it we would always be needing to change it in two places.

So instead we want to code in a Transition End Event

We are going to want another event listener to listen when this event happens and in this case we want it listening to All the keys

const keys **=** **document**.querySelectorAll('.key');

and I can copy paste

**document**.querySelectorAll('.key');

and put it in my console and press enter I get an array of the node list object of all the keys

so I now want to listen for the event called transition end on each one

so each key gets an event listener added to it, which is transitionend (when a transition is ending) and then it removes it

keys.forEach(key => key.addEventListener('transitonend', removeTransition));

so now we need to make that function

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

key.classList.add('playing');

});

function removeTransition(e) {

**console**.log(e);

}

const keys **=** **document**.querySelectorAll('.key');

keys.forEach(key => key.addEventListener('transitonend', removeTransition));

now when we test it and press A in our console we get 6 different transitions

but we are really only wanting typically the longest one but in our case we want to specify the transitionend

so we make an if statement where

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

function removeTransition(e) {

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

**console**.log(e.propertyName);

}

const keys **=** **document**.querySelectorAll('.key');

keys.forEach(key => key.addEventListener('transitionend', removeTransition));

as of now if we test it the console will console log the word transform because that is the property that is being ended

now if you went messed with the css again and changed the transform you would see the word transform in the console log show however much time you set it as.

So know that it transforms IN we then want to REVERSE it and transform it OUT.

So we can use THIS which is always got called against it (key.addEventListener)

A easy way to find out what THIS is, is to console log it.

function removeTransition(e) {

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

**console**.log(this);

we then add

this.classList.remove('playing');

run it

**IT WORKS!!**

<script>

**window**.addEventListener('keydown', function (e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

key.classList.add('playing');

});

function removeTransition(e) {

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

this.classList.remove('playing');

}

const keys **=** **document**.querySelectorAll('.key');

keys.forEach(key => key.addEventListener('transitionend', removeTransition));

</script>

Now we can clean it up

(dude says he’s not a fan of attaching a function to a keydown)

Place the event function in its own function

(function playSound)

<script>

**window**.addEventListener('keydown');

function playSound(e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**;*//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

key.classList.add('playing');

};

function removeTransition(e) {

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

this.classList.remove('playing');

}

const keys **=** **document**.querySelectorAll('.key');

keys.forEach(key => key.addEventListener('transitionend', removeTransition));

</script>

Then move window.addeventListener down to the bottom (highlighting and then using Alt+ up or down

Alt+Down

And then say when someone keys down then play the sound

**window**.addEventListener('keydown');

to

**window**.addEventListener('keydown', playSound);

Finished script

<script>

function playSound(e) {

const audio **=** **document**.querySelector(`audio[data-key="${e.keyCode}"]`);

const key **=** **document**.querySelector(`.key[data-key="${e.keyCode}"]`);

**if** (**!**audio) **return**; *//stops the function from running*

audio.currentTime **=** 0; *//rewind to the start*

audio.play();

key.classList.add('playing');

};

function removeTransition(e) {

**if** (e.propertyName **!==** 'transform') **return**; *//skip it if its not a transform*

this.classList.remove('playing');

}

const keys **=** **document**.querySelectorAll('.key');

keys.forEach(key => key.addEventListener('transitionend', removeTransition));

**window**.addEventListener('keydown', playSound);

</script>