**Flex Panel Gallery**

We have got this thing called flex panel and when the pagagraph inside is clicked it grows and or moves. So the file we are given has some css but it currently doesn’t move. And so first we are going to put our css panels going vertical side by side instead of stacking them on top of each other.

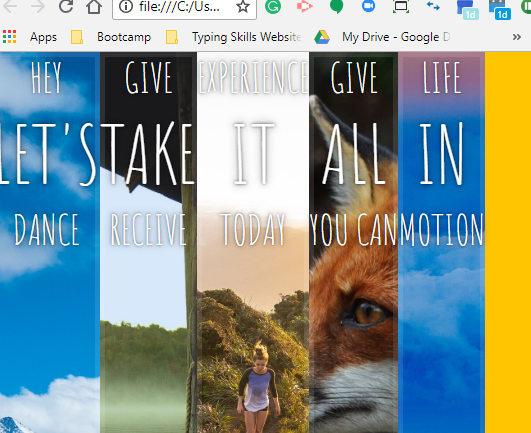
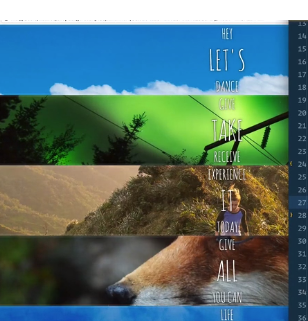
We go to .panels div and add display: flex;

.panels {

min-height:100**vh**;

overflow: *hidden*;

display: *flex*;



So now the flex box is currently taking as much space as the words are so what we want to tell it to do is to tell the children to split the space amongst yourselves. So we go to out .panel div and add flex: 1; which will do this

font-size: 20**px**;

background-size:*cover*;

background-position:*center*;

flex: 1;

}

Next to just visually help us lets just put a small border on each of the flex children

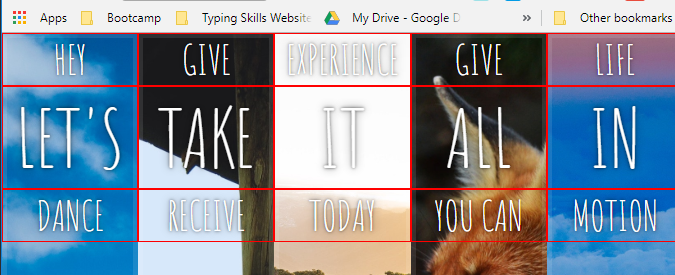
.panel **>** \* {

margin:0;

width: 100**%**;

transition:transform 0.5**s**;

border: 1**px** *solid* red;



Currently now they are at the top of the page so we go into the .panel div

And add justify-content: center; which centers the content horizontally

And align-items: center; which doesn’t get us much but center the words vertically in the div. so to get it where we want we want to display: flex; on this div as well

flex: 1;

justify-content: *center*;

align-items: *center*;

display: *flex*;

an element can both be a flex item as well as a flex container and by doing that it displays them left to right as the default



So to get them vertical we add flex-direction: column;

display: *flex*;

flex-direction: *column*;

but we also want each of these flex children to take up 1/3 of the page. So on the flex panel div

flex: 1 0 *auto*;

which divides it evenly however the text is back. So we add

flex: 1 0 *auto*;

display: *flex*;

justify-content: *center*;

align-items: *center*;



Next we want to hide the top divs up and the bottom divs down. So what we are essentially going to be doing is when you take off the translate Y its going to animate itself in and out of the page. And how that’s going to work is when the panel is active we translate y back to zero from where we will set it at 100 (off the page)

.panel **>** \*:first-child {transform: translateY(-100**%**); }

.panel.open-active **>** \*:first-child {transform: translateY(0); }

.panel **>** \*:last-child {transform: translateY(100**%**); }

.panel.open-active **>** \*:last-child {transform: translateY(0); }

So this means (which we can experiment in the dev tools) if we add a class of open active it will animate itself in to its new position. Take it off and it will animate off.

we can take the border off sense we kinda know where the elements are on the page

The last bit of the puzzle is that one of these panels also want to have a class of open. So that it will get bigger. So in .panel.open we will give it a flex of 5

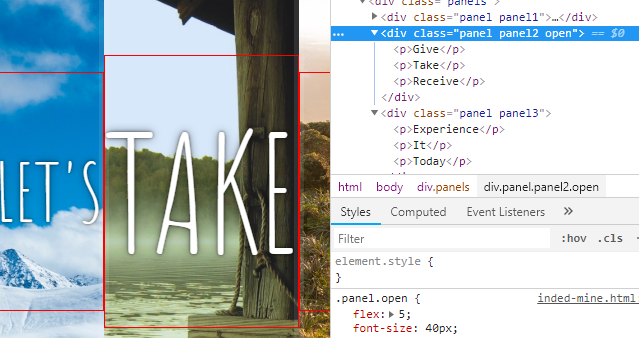
.panel.open {

flex: 5;

font-size:40**px**;

}

Earlier when we gave it a it a flex of one means that each of them are going to distribute the extra room. So here by giving it a flex of five, what that means is that when it has a class of open its going to take 5 times the amount of extra room as the rest of them.

 so when you add the class in it will animate itself in and when you take it out it will animate(transition rather) itself out.

Now that we have all the CSS in place all we need to do is wright a little JavaScript so that when we click them we will add and remove the classes

So first we want to grab all the panels. And we will do document.querySelectorAll which will give us a node list of the ones we have.

const panels **=** **document**.querySelectorAll('.panel');

and then we want to wright a function that will toggle that class of on /or open.

function toggleOpen() {

this.classList.toggle('open');

}

And then on each of the panels we will listen for a click

panels.forEach(panel => panel.addEventListener('click', toggleOpen));

so we are going to take each of the panels, loop over each one of them, listen for a click on each of them and then run the toggle function when it is run

now that we have this div transitioning open we want to bring the other transition too into play

so we are going to do the same thing and instead listen for a transitionend and change the toggleOpent to toggleActive

panels.forEach(panel => panel.addEventListener('transitionend', toggleActive));

so now we have to wright the function for it

function toggleActive(a) {

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

**if**(e.propertyName.includes('flex')){

this.classList.toggle('open-active');

}

}

So again what is happening here is we are listening for a click on each panel. When someone clicks we are going to open it. That’s going to then intern trigger a css open. And when that finishes the second one transition end will fire. And that’s when we are going to toggle open active. And it will just go the other way when you click it again (off)