Notes about the “WordFlash” animated text effect HTML files by Jonray

This text animation shows a short paragraph, sentence or phrase appearing word-by-word in a brief artistic animation. Similar text effects are used frequently in documentaries and TV news reports. I’ve called it “WordFlash”.

The HTML file referred to in this document is:

WORDFLASH text effect 01 TEMPLATE ANTONIO VIVALDI slide from LEFT then exit STAGGERED vertically (AT KEYFRAMES method).HTML

TO VIEW THE HTML ANIMATION -

The HTML file can be previewed just by double-clicking it.

Your browser will open and the animation will play. It will have a solid purple background with moving white and yellow text.

If the text effect looks big and goes off your screen, hit CTRL/- (Ctrl and minus) a few times until the whole purple rectangle just fills your screen. Alternatively, go to the menu (top right of your browser (Chrome)), scroll down to “Zoom” and zoom out a few times.

After the animation has played you can hit “replay” (top left, CHROME) to play it again.

TO VIEW / EDIT THE HTML TEXT CONTENT -

I suggest you install Notepad++ on your computer (or another HTML editor), and view the file this way. In Notepad++ you can then select “Run” (top menu) then “Launch in Chrome” (or a browser of your choice).

Alternatively, to could change the extension of the file to “.txt” and open it in Windows standard Notepad or any other text editor. You can edit it. However, you won’t be able to preview it without re-saving it with the extension “.html”.

A WORD ABOUT FONTS -

For the animation to look as intended, you’ll need to have the font “Calisto MT” installed on your computer.

If not, the text will default to Times New Roman. It will still work, and it may look nearly right, but the layout of the words may be incorrect.

You might like to Google “Calisto font”, download it and install it by dragging the file to your fonts folder.

You can easily change the font to any installed on your computer. Just type the name of your font after the “font-family” attribute (line 51). Use just lower-case letters. For example, typing “arial” will result in the non-serif “arial” font. But again be aware that the layout may look wrong.

Getting into detail of the HTML text.

COLOURS -

The background colour of the animation will be purple. That's because of the attribute "background:purple;" in line 10:

```
1   <html><head><style>
2
3   /* Below: Leave this as it is!     */
4   /*~~~~~~~~~~~~~~~~~~~~~~~~*/
5   div                 {position:absolute; }
6
7   /* Below: You can make the purple and light blue colours transparent by ad
8   /*~~~~~~~~~~~~~~~~~~~~~~~~*/
9   body                {background:lightblue;}
10  #fullscreen_box     {background:purple;         width:1920px;   height:108
```

I chose purple because it is generally easy on the eye, not to glaring.

You could make the purple change to TRANSPARENT by adding a # after it (or indeed any letter or symbol). It's a quirky method I use to be able to swap between a background colour and transparent.

Usually I will make boxes coloured to begin with to make it easy to position and lay them out, then make them transparent later by adding #s.

By making all the colours transparent, except for the text itself, then you can overlay the text to show over a video.

Please note that if you do make the purple transparent, you will now see a light blue background. That's because of "body {background:lightblue;}" in line 9. To make the background transparent, add a # to "lightblue" as well (lightblue#).

However, if you make these two colours transparent then preview this in a browser, you won't see the white text because your browser has a white background, too.

RIGHT-ALIGNED TEXT -

Because the words in this animation are right-aligned, I've used the attribute "text-align:right " for all the "line_xxword_xx" elements (see line 51). This makes each word align to the right-hand edge of its box.

LAYOUT

Back to line 10. This gives a large (1920 pixels by 1080 pixels) purple box.

```
1   <html><head><style>
2
3   /* Below: Leave this as it is!     */
4   /*~~~~~~~~~~~~~~~~~~~~~~~~*/
5   div                {position:absolute; }
6
7   /* Below: You can make the purple and light blue colours transparent by adding a # after them.   */
8   /*~~~~~~~~~~~~~~~~~~~~~~~~*/
9   body               {background:lightblue;}
10  #fullscreen_box    {background:purple;        width:1920px;   height:1080px;  margin:auto;top:0px;bottom:0px;left:0px;right:0px;overflow:hidden#;}
```

1920 x 1080 is the size of a full screen video in HD. I gave this box an ID of “fullscreen_box”. The code after “margin” centers the purple box in the browser.

Look at line 10. This is called “Container” because it has inside it (“contains”) some orange rectangular boxes (“text containers”) and inside THOSE are nested the actual text boxes with the white and yellow text.

```
10      #fullscreen_box     {background:purple;         width:1920px;   height:1080px;  margin:auto;top:0px;bottom:0px
11
12      /* Below: You can change the top and right attributes to position the whole animation on the page.       */
13      /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
14      #container          {background:dodgerblue#;    width:80%;      height:80%;     top:300px; right: 10%; }
15
```

Because this “container” is nested inside the purple box, its size is: width 80% of the purple box and 80% of the height of the purple box. It’s 300px from the top of the purple box and a distance of 10% of the purple box’s width from the right-hand edge of the purple box.

```
182   <div id="yellow_box_container">
183   <div id="yellow_box"></div>
184   </div>
185
186
187       <div id="line00_template_zeros">0000<span
188
189       <div id="line_01_container">
190       <div id="line01_word01" >Antonio</div>
191       <div id="line01_word02" >Vivaldi</div>
192       <div id="line01_word03" >was</div>
193       <div id="line01_word04" ></div>
194       <div id="line01_word05" ></div>
195       <div id="line01_word06" ></div>
196       </div>
197
198
199       <div id="line_02_container">
200       <div id="line02_word01" >ordained</div>
201       <div id="line02_word02" >as</div>
202       <div id="line02_word03" >a</div>
203       <div id="line02_word04" >priest,</div>
204       <div id="line02_word05" ></div>
205       <div id="line02_word06" ></div>
206       </div>
207
208       <div id="line_03_container">
209       <div id="line03_word01" >though</div>
210       <div id="line03_word02" >he</div>
211       <div id="line03_word03" >instead</div>
212       <div id="line03_word04" ></div>
213       <div id="line03_word05" ></div>
214       <div id="line03_word06" ></div>
215       </div>
216
217       <div id="line_04_container">
218       <div id="line04_word01" >chose</div>
219       <div id="line04_word02" >to</div>
220       <div id="line04_word03" >follow</div>
221       <div id="line04_word04" >his </div>
222       <div id="line04_word05" ></div>
223       <div id="line04_word06" ></div>
224
225       </div>
```

Look at lines 89 - 120.

This is where I have used DIVS (divisions) to specify the layout of the screen and which elements to include.

It’s important to note how I’ve NESTED divs inside each other, because this can dramatically alter the layout of the animation.

For example, the divs named “line01_wordxx” are all nested inside the div named “line_01_container”.

Note I have typed in the words of the animation when I want the text to appear. The template includes enough space for 6 lines with up to 6 words in each. This of course can be increased by adding more divs.

Obviously in this animation there are only 3 or 4 words in each line, so I’ve left the divs below blank - they won’t show.

Or you can delete the unused divs.

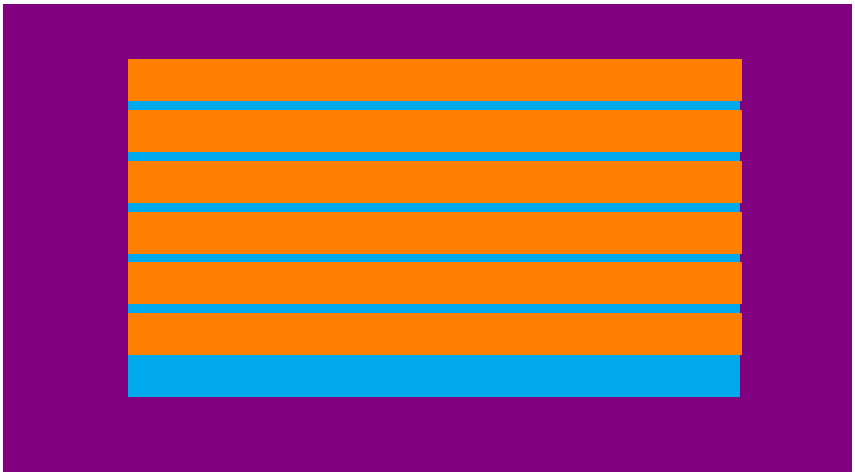
MORE ABOUT THE LAYOUT

SO - basically, the layout (with colours applied) looks like THIS in stages:

1. Large purple box (“fullscreen_box”) :

2. Blue “container” (“#container”):

3. 6 orange “#line_containers” numbered 1 - 6:

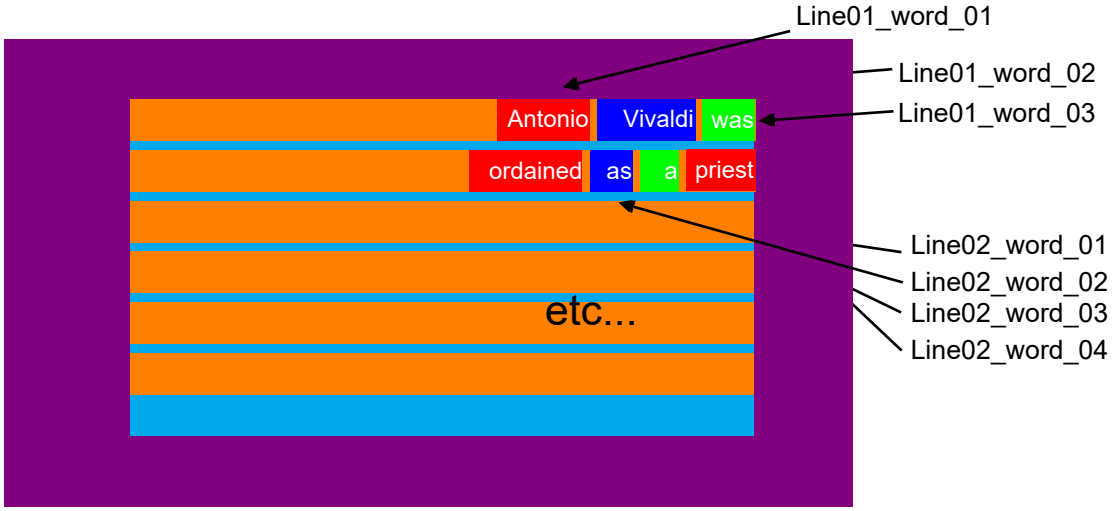


These are nested inside the blue box and take up 100% of its width.

Each box is 93 pixels high and they are all stacked vertically at a distance of 100 pixels. The height of the boxes can be changed (in conjunction with changing the font size), but make sure the orange box is slightly larger than the size of the font, otherwise any “ys” “ps” or “gs” may be cut off at the bottom.

3. Individual word boxes: (Line01_word_01, Line01_word_02,

Line02_word_01, Line02_word_02, Line02_word_03, ... etc...



These boxes are nested inside the orange boxes.

THE HORIZONTAL POSITIONING OF THE WORDS

In the animation, each word will have to be animated separately.

This, of course, is one of the limitations of this system - you'll need to position the text boxes manually and individually, which can take some time (although this process will become faster once you get used to doing it). The distance between each word will obviously depend on the physical width of each individual word.

1. USING PIXELS

This is a screenshot of a previous version of the HTML file I created, when I was testing it. I've used the "margin-right" attribute and pixel units to position the words horizontally.

```
34
35 #line01_word01 { background:red#    ;   margin-right: 210px; width: 200px;  }
36 #line01_word02 { background:blue#   ;   margin-right: 000px; width: 200px;  }
37 #line02_word01 { background:red#    ;   margin-right: 300px; width: 200px;  }
38 #line02_word02 { background:blue#   ;   margin-right: 140px; width: 200px;  }
39 #line02_word03 { background:green#  ;   margin-right: 000px; width: 200px;  }
40 #line03_word01 { background:red#    ;   margin-right: 170px; width: 500px;  }
41 #line03_word02 { background:blue#   ;   margin-right: 000px; width: 200px;  }
42 #line04_word01 { background:red#    ;   margin-right: 400px; width: 200px;  }
43 #line04_word02 { background:blue#   ;   margin-right: 340px; width: 200px;  }
44 #line04_word03 { background:green#  ;   margin-right: 120px; width: 200px;  }
45 #line04_word04 { background:yellow# ;   margin-right: 000px; width: 200px;  }
46 #line05_word01 { background:red#    ;   margin-right: 000px; width: 400px;   color:yellow;}
47 #line06_word01 { background:red#    ;   margin-right: 000px; width: 600px;   color:yellow;}
48
```

Note that the text boxes "line01_word_02" and "line02_word_03" both have the attribute "margin-right:000px;", meaning that they are aligned to the right edge of the orange box.

2. USING the “ch” ATTRIBUTE

Since using pixels as the unit of movement is really fiddly, difficult to utilise and time-consuming, I experimented with using “ch” values.

What is the “ch” attribute?

In HTML, “”ch” or “character” is a unit of distance equal to the WIDTH of the “0” character of the font you’re using. So the physical distance changes according to what font you are using.

This should be a better solution than pixels for helping to position the words. In essence, if you count the number of letters in each word, and allow another 1 or 2 for a space, you should quickly be able to choose appropriate values to position each word.

I tried it. However it wasn’t quite as easy as I expected. Mainly because in non-monospace fonts each letter has a different width, and the width of the “0” character is generally about 20-30% wider that the average width of other characters.

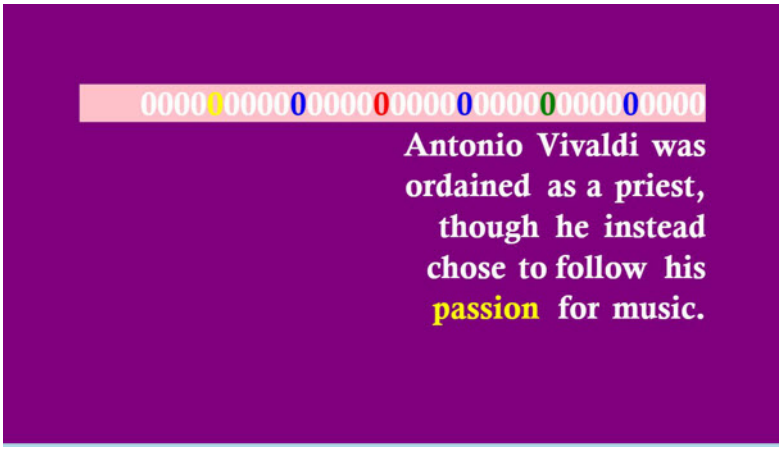
This screenshot shows the margin-right attributes with the ch units. Note it is possible to have fractions of whole units (eg. 18.5, 6.2 etc) for fine placement.

```
57      #line01_word01 { background:red#    ;   margin-right: 11ch;     width:  800px; }    /* antonio */
58      #line01_word02 { background:blue#   ;   margin-right: 4ch;      width:  800px; }    /* vivaldi*/
59      #line01_word03 { background:red#    ;   margin-right: 0ch;      width:  800px; }    /* was*/
60      #line01_word04 { background:red#    ;   margin-right: 0ch;      width:  800px; }
61      #line01_word05 { background:red#    ;   margin-right: 0ch;      width:  800px; }
62      #line01_word06 { background:red#    ;   margin-right: 0ch;      width:  800px; }
63
64      #line02_word01 { background:red#    ;   margin-right: 10.5ch;     width:  800px; }    /* ordained */
65      #line02_word02 { background:blue#   ;   margin-right: 7.8ch;      width:  800px; }    /* as */
66      #line02_word03 { background:green#  ;   margin-right: 6.2ch;    width:  800px; }      /* a */
67      #line02_word04 { background:green#  ;   margin-right: 0ch;      width:  800px; }      /* priest,*/
68      #line01_word05 { background:red#    ;   margin-right: 0ch;      width:  800px; }
69      #line01_word06 { background:red#    ;   margin-right: 0ch;      width:  800px; }
```

Because it’s so difficult to work out the exact positions, I thought of another idea. My method involves adding a new line of text, directly above line 1 of the animation, and filling it with just 0s (zeros) - and using them as a template.

Further, I made each 5th zero (counting from the RIGHT) BLUE, the 10th zero GREEN, the 20th zero RED, and so on. This made it quite easy for me to judge where the position of each word should go.

Here’s a screenshot:



This is the code producing the above template line of noughts. First, the HTML:

```html
<div id="line00_template_zeros">0000<span style="color:yellow">0</span>0000<span style="color:blue">0</span>0000<span style="color:red">
0</span>0000<span style="color:blue">0</span>0000<span style="color:green">0</span>0000<span style="color:blue">0</span>0000</div>
```

... and now the CSS:

```css
47      /* Below: This provides a template (row of noughts) to help you position the words quickly.
48      Switch ON (opacity:1) when setting the spacing, switch off (opacity:0) afterwards to hide it.*/
49      /*~~~~~~~~~~~~~~~~~~~~~~~~~*/
50     #line00_template_zeros  {background:pink; width:100%; height:93px; top:-100px; right: 0%;  overflow:hidden;
51     font-family:calisto mt; color:white; font-size: 80px; font-weight:bold; text-align:right;              opacity:1;}
52
```

Having turned ON this row of noughts using “opacity:1”, to make it easy to position the words, you could DISABLE the ANIMATION and make the words static while you adjust the ch values.

To do this:

1) add a # or X (or any other symbol) after the animation names in the @keyframes section of the code, and

```css
 /* Below: KEYFRAMES*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~*/
@keyframes move1# {
0%   { opacity: 0;      right: -100px;       top: 0px;}
10%   { opacity: 1;      right: 0px;           top: 0px;}
90%  { opacity: 1;      right: 0px;          top: 0px;}
95%  { opacity: 1;      right: 0px;          top: -100px;}
100% { opacity: 1;      right: 0px;          top: -100px;}
}

@keyframes slide1# {
0%   { opacity: 0;      left:100%;          top: 0%;}
6%   { opacity: 1;      left:0%;            top: 0%;}
90%  { opacity: 1;      left:0%;            top: 0%;}
93%  { opacity: 1;      left:0%;            top: -100%;}
100% { opacity: 1;      left:0%;            top: -100%;}
}
```

2) change “opacity:0” to “opacity 1” in line 51 (to make the line of noughts visible):

```css
47      /* Below: This provides a template (row of noughts) to help you position the words quickly.
48      Switch ON (opacity:1) when setting the spacing, switch off (opacity:0) afterwards to hide it.*/
49      /*~~~~~~~~~~~~~~~~~~~~~~~~~*/
50     #line00_template_zeros  {background:pink; width:100%; height:93px; top:-100px; right: 0%;  overflow:hidden;
51     font-family:calisto mt; color:white; font-size: 80px; font-weight:bold; text-align:right;              opacity:1;}
52
```

You can then work on positioning the text boxes, using the noughts as a visual guide, and, when done, activate the animation again, by 1) removing the #s and making opacity = 0 again (line 51).

Don’t forget the last word in each line will have the value “0ch” - because it will always be right-aligned to the edge of the orange box. To help you keep track of which lines have words, and how many, you could add a short comment and type the words at the ends of the “linexx_word_xx” entries - for example:

```css
#line01_word01 { background:red#    ;  margin-right: 11ch;     width:  800px; }    /* antonio */
#line01_word02 { background:blue#   ;  margin-right: 4ch;      width:  800px; }    /* vivaldi*/
#line01_word03 { background:red#    ;  margin-right: 0ch;      width:  800px; }    /* was*/
#line01_word04 { background:red#    ;  margin-right: 0ch;      width:  800px; }
#line01_word05 { background:red#    ;  margin-right: 0ch;      width:  800px; }
#line01_word06 { background:red#    ;  margin-right: 0ch;      width:  800px; }

#line02_word01 { background:red#    ;  margin-right: 10.5ch;     width:  800px; }    /* ordained */
#line02_word02 { background:blue#   ;  margin-right: 7.8ch;      width:  800px; }    /* as */
#line02_word03 { background:green#  ;  margin-right: 6.2ch;    width:  800px; }      /* a */
#line02_word04 { background:green#  ;  margin-right: 0ch;      width:  800px; }      /* priest,*/
#line01_word05 { background:red#    ;  margin-right: 0ch;      width:  800px; }
#line01_word06 { background:red#    ;  margin-right: 0ch;      width:  800px; }
```

HOW I MADE THE TEXT MOVE AROUND

THE “@KEYFRAMES” METHOD -

In my first HTML file, I used the usual (standard) way of creating an HTML animation - by the use of the “@keyframes”.

Browsers understand this method, so the animation will work in all browsers, but however Shotcut does not recognise this way of doing things. That’s why, once I made the animation this way I then used Elusien’s HTML/CSS animation generator to convert the code so the second HTML file will enable the animation to be applied to a clip in Shotcut.

HOW THE “@KEYFRAMES” METHOD WORKS -

```
/* Below: KEYFRAMES*/
/*~~~~~~~~~~~~~~~~~~~~~~~~~*/
@keyframes move1 {
0%   { opacity: 0;      right: -100px;      top: 0px;}
10%   { opacity: 1;      right: 0px;         top: 0px;}
90%  { opacity: 1;      right: 0px;         top: 0px;}
95%  { opacity: 1;      right: 0px;         top: -100px;}
100% { opacity: 1;      right: 0px;         top: -100px;}
}

@keyframes slide1 {
0%   { opacity: 0;      left:100%;          top: 0%;}
6%   { opacity: 1;      left:0%;            top: 0%;}
90%  { opacity: 1;      left:0%;            top: 0%;}
93%  { opacity: 1;      left:0%;            top: -100%;}
100% { opacity: 1;      left:0%;            top: -100%;}
}
```

Look at the code on the left.

Specifically look at the first chunk of code.

This tells the browser to perform an animation.

I’ve given the animation a name - “move1”.

Let’s concentrate on just the “opacity” attribute.

At 0%, opacity is 0 (ie invisible); then at 10%, opacity is 1 (ie solid). The opacity stays solid all the way through to 100%.

SO - this results in the animation fading in (from 0 - 10%), and staying solid (opaque) for the rest of the animation time. (I’ll show you how to set the animation duration later...).

But there are other things going on. At 0%, the “right” attribute is -100px, and at 10% it becomes 0px.

This means that the element which the animation is applied to will start 100 pixels to the right and at 10% it will have moved left to the position of 0 pixels to the right (ie its default right position).

And one more thing ... At 90% the “top” attribute is 0px (default), but by 95% the element moves upwards by a distance of 100 pixels. This is the end or “exit” animation, where the text moves upwards and disappears.

By the way, the reason the words disappear as they move upwards is because of the attribute “overflow:hidden” applied to the (orange) text containers:

```
    /* Below: You can change the height if you want a larger font-size but do it in conjunction with
    changing the text font-size (below). Make the orange boxes slightly larger than the font-size, otherwise
    any "ys" and "gs" may be cut off at the bottom, depending on your font.*/
    #line_01_container      {background:orange#;    width:100%; height:93px;    top:000px; right: 0%; overflow:hidden;}
    #line_02_container      {background:orange#;    width:100%; height:93px;    top:100px; right: 0%; overflow:hidden;}
    #line_03_container      {background:orange#;    width:100%; height:93px;    top:200px; right: 0%; overflow:hidden;}
    #line_04_container      {background:orange#;    width:100%; height:93px;    top:300px; right: 0%; overflow:hidden;}
    #line_05_container      {background:orange#;    width:100%; height:93px;    top:400px; right: 0%; overflow:hidden;}
    #line_06_container      {background:orange#;    width:100%; height:93px;    top:500px; right: 0%; overflow:hidden;}
```

The attribute “overflow:hidden” applied to the (orange) text containers means that when the text boxes (nested inside them) move out of the boundaries of the text containers, they become invisible. It creates a very pleasant artistic effect when the viewer sees the words disappearing as if by magic!

HOW TO CONTROL THE DURATION OF THE ANIMATION -

Look at lines 107 onwards:

```
    /* Below: ANIMATIONS.*/
    /*~~~~~~~~~~~~~~~~~~~~~~*/
    #yellow_box     {animation: slide1 6s ease  0.05s   1   normal forwards;}

    #line01_word01  {animation: move1 6s ease   0.0s    1   normal forwards;}
    #line01_word02  {animation: move1 6s ease   0.05s   1   normal forwards;}
    #line01_word03  {animation: move1 6s ease   0.1s    1   normal forwards;}
    #line01_word04  {animation: move1 6s ease   0.1s    1   normal forwards;}
    #line01_word05  {animation: move1 6s ease   0.1s    1   normal forwards;}
    #line01_word06  {animation: move1 6s ease   0.1s    1   normal forwards;}

    #line02_word01  {animation: move1 6s ease   0.2s    1   normal forwards;}
    #line02_word02  {animation: move1 6s ease   0.25s   1   normal forwards;}
    #line02_word03  {animation: move1 6s ease   0.3s    1   normal forwards;}
    #line02_word04  {animation: move1 6s ease   0.3s    1   normal forwards;}
    #line02_word05  {animation: move1 6s ease   0.1s    1   normal forwards;}
    #line02_word06  {animation: move1 6s ease   0.1s    1   normal forwards;}
```

Notice I’ve separated out the “animation” attribute for each of the “linexx_wordxx” divs.

Here’s a brief description of what each part of the animation means:

1) Move 1 — The user-created name of the animation.

2) 6s — The animation’s duration is 6 seconds.

3) ease — The movement’s acceleration speeds up slowly at the start and slows down at the end.

4) .2s — The animation is delayed by 0.2 seconds.

5) 1 — The animation is played once

6) normal — The animation’s direction - as opposed to reverse or alternate, for example.

7) forwards — The element retains its style values of the last keyframe (ie the animations stops and stays visible).

Notice how I've made the numbers at 4) (animation delay) gradually increase when you read down the list (of the active words). For the blank lines, the animation doesn't matter, because the words aren't going to show anyway).

This creates the effect where each word of the text moves in sequence, each one moving a split-second after the other - AKA a "staggered" motion. This achieves a very interesting and artistic effect.

HOW TO CONVERT THE ANIMATION SO IT CAN BE APPLIED IN SHOTCUT

As I mentioned earlier, the "@keyframes" method (which is the common way of achieving HTML animations) will not work in Shotcut.

Fortunately, Shotcut Forum member Elusien has devised a tool for converting an "@keyframes" HTML animation into HTML code which can be used in Shotcut.

To begin:

1) Check that your "@keyframes" animation works OK in you browser by double-clicking on the HTML file. As mentioned earlier, you may have to change the zoom factor of the browser to see the full screen.

2) Go to www.elusien.co.uk/shotcut, then select the button "CSS3 animations".

3) Follow the instructions on the webpage.

Basically, you copy and paste a) the @keyframes data, and b) the animations data into the text fields (this explains why I separated out the "animation" code from the other CSS code in the HTML file), then select how many frames per second you'd like the animation to run at.

4) Copy the automatically generated code and paste it back into your HTML document just after the </head> (end of head tag). (I suggest you save it with a different file-name).

5) As per Elusien's instructions, make sure you have downloaded the two .js files and put them in either the parent folder or the same folder as the HTML file.

Important - If you put them in the SAME folder, you should change the line of code below:

```
82 <!--##########################################-->
83 <script src="../jquery.min.js"          ></script>
84 <script src="../elusien_cssanims2js.js"></script>
85 <script>!window.jQuery && document.write(unescape('%3Cscript
86 <!--##########################################-->
```

to:

```
82 <!--##########################################-->
83 <script src="jquery.min.js"          ></script>
84 <script src="elusien_cssanims2js.js"></script>
85 <script>!window.jQuery && document.write(unescape('%3Cscript
86 <!--##########################################-->
87
```

(ie. remove the text "../" from both lines).

HOW TO APPLY THE ANIMATION TO A VIDEO CLIP IN SHOTCUT

Follow these steps:

1) Open Shotcut.

2) Decide which clip you want to apply the HTML file to. You can apply it to a video clip, a colour clip or a transparent clip.

(Applying it to a transparent clip, placing it on a higher track than V1 (ie V2), then making all the background colours transparent so that just the text shows, is probably the best method, since you can then move the clip horizontally on the timeline to where you want it to appear and it gives you great control this way).

3) Add a “text:HTML” filter to your chosen clip.

4) Tick the “Use Webvfx javascript extension” box, and hit “Yes” to confirm.

5) Make sure that you have the HTML file in a convenient folder , and that the two .js files (downloaded from Elusien’s site) are in the SAME FOLDER as the HTML file. Also, change the code from:

```
82 <!--##########################################-->
83 <script src="../jquery.min.js"          ></script>
84 <script src="../elusien_cssanims2js.js"></script>
85 <script>!window.jQuery && document.write(unescape('%3Cscript
86 <!--##########################################-->
```

to:

```
82 <!--##########################################-->
83 <script src="jquery.min.js"          ></script>
84 <script src="elusien_cssanims2js.js"></script>
85 <script>!window.jQuery && document.write(unescape('%3Cscript
86 <!--##########################################-->
87
```

(ie just remove the “../” from each line).

6) Select “Open” and point to the “converted” HTML file. If all is well, when you hit “play” in Shotcut the animation will appear.

TWO FINAL TIPS -

1) Although you can edit the HTML file directly in Shotcut (by selecting “Edit” in the text:HTML filter pane), I find it a good idea to keep the HTML file open in Notepad++. Then if I need to edit this file, I do it in Notepad++, save, then select “Reload” in the filter pane.

2) If you select a MINUS figure (eg -25) for the framerate (when converting the file in Elusien’s site), you can then control the speed (rate) of the animation by altering the length of the clip on the timeline by click/hold/dragging the right-hand edge of the clip. The shorter the clip, the faster the animation.

GOOD LUCK!