Lab 8: Smooth CSS3 panel TRANSITIONS

In this lab we will create a responsive 100% width/height layout with some smooth page transitions. The idea is to have some content panels and a navigation which will allow us to navigate between the panels. We’ll use radio buttons for the navigation and animate the content to the right position with a transition, creating a “smooth scrolling” effect. This layout idea could be useful for web pages or web apps where the content should be strictly the size of the screen (width and height).

*Note: most vendor prefixes are excluded for expediency – add them if you want to use this in production across multiple browsers*

THE MARKUP

The structure will consist of a main container with the class **st-container** which will contain radio buttons and links, and the wrapper with the class **st-scroll** for the panels. Each panel will have some content elements:

1. Create a new folder for your project called Lab 15 CSS Page Transitions
2. Add html document called index.html with the usual html, head, and body tags.
3. Add the following between the opening and closing <body> tags

<div class="container">

<div class="st-container">

</div><!-- // st-container -->

</div> <!-- // container -->

1. Let’s create the Navigation elements.

As we are using only CSS and html mark-up for our menus, we need something to respond to button clicks on the navigation menu. We can avoid using JavaScript by using radio buttons which are clickable. We will “hide” the radio buttons by giving them 0 opacity. Add the following radio buttons directly below <div class="st-container">

<input type="radio" name="radio-set" checked="checked" id="st-control-1"/>

<a href="#st-panel-1">About</a>

<input type="radio" name="radio-set" id="st-control-2"/>

<a href="#st-panel-2">Games</a>

<input type="radio" name="radio-set" id="st-control-3"/>

<a href="#st-panel-3">Illustrations</a>

<input type="radio" name="radio-set" id="st-control-4"/>

<a href="#st-panel-4">Character Design</a>

<input type="radio" name="radio-set" id="st-control-5"/>

<a href="#st-panel-5">Contact</a>

1. Now we add the content panels – 1 for each menu link created in the last step

Notice that some of the section elements have 2 class selectors associated; the second class colours the header text pink to provide variation in text colour.

<div class="st-scroll">

<section class="st-panel st-color" id="st-panel-1">

<h2>Eye Candy</h2>

</section>

<section class="st-panel" id="st-panel-2">

<h2>Games</h2>

<p>

Art party readymade beard labore cosby sweater.

</p>

</section>

<section class="st-panel st-color" id="st-panel-3">

<h2>Illustrations</h2>

<p>

Sint aute occaecat id vice. Post-ironic godard.

</p>

</section>

<section class="st-panel st-color" id="st-panel-4">

<h2>Character Design</h2>

<p>

Mixtape fap leggings art party, cosby sweater

</p>

</section>

<section class="st-panel" id="st-panel-5">

<h2>Contact Us</h2>

<p>

Fixie ad odd future polaroid nulla irony.

</p>

</section>

</div><!-- // st-scroll -->

What we want to do is basically move the panel wrapper by changing its top value and bringing the respective panel into the viewport.

We can do this by selecting the ***sibling*** of a checked radio button, the **st-scroll** division, with the sibling combinator and target the correct panel inside.

Because of this technique, we need to keep the radio buttons on the same z-index level as the **st-scroll** and on top of the links (they will be invisible though, since we’ll give them 0 opacity).

In order for us to to select the correct panel, we give IDs to the links and and to the radio buttons.

The reason why we use links and not, labels, is that we want to be able to create some kind of “fallback” for non-supportive browsers (sibling combinators don’t work in older browsers).

The links have the href value of the panels’ IDs, so for the fallback, we’ll simply hide the radio buttons, making the links clickable which will make it possible to “jump” to the right panel.

### THE CSS

The trick is to make the main container absolute with a width and height of 100% while setting the panels and their wrapper to position relative. But they will also have a width and height of 100%. This will make each panel exactly the size of the screen (since the main container and the panel wrapper are) but allow an overflow of the content, stacking the panels in the classic way.

1. Add the following style block and content to the html document: (Choose your own hi-res background images and create an *images* folder in the same location as *index.html* to store them in)

<style>

#st-panel-1 {

background: url(images/aye.jpg) no-repeat;

background-position: 50% 50%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

#st-panel-2 {

background: url(images/mario.jpg) no-repeat;

background-position: 50% 50%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

#st-panel-3 {

background: url(images/carmelite.jpg) no-repeat;

background-position: 50% 50%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

#st-panel-4 {

background: url(images/hitman.jpg) no-repeat;

background-position: 50% 50%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

#st-panel-5 {

background: url(images/contact.jpg) no-repeat;

background-position: 50% 50%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

</style>

1. Create a new style sheet called style.css and ad a link to the sheet within the head tags of index.html. We will add all further CSS styles to this style sheet.

<link rel="stylesheet" type="text/css" href="css/style.css" />

1. We will do the content navigation by animating the panel wrapper, so we’ll set the body overflow to hidden:

body {

overflow: hidden;

}

1. Let’s style the main container:

.st-container {

position: absolute;

width: 100%;

height: 100%;

top: 0;

left: 0;

font-family: 'Josefin Slab', 'Myriad Pro', Arial, sans-serif;

color: #ffffff;

}

1. : We’ll put the “navigation” at the bottom of the page by giving it a fixed position. Note that we are setting the same width and height for both, the input and the link. The idea is to overlay the radio button on the link elements so that they are clickable, but giving them 0 opacity so that they are not visible. And because of that it’s also important that we set the z-index of the radio buttons higher than the one of the link elements:

.st-container > input, .st-container > a {

position: fixed;

bottom: 0px;

width: 20%;

cursor: pointer;

font-size: 16px;

height: 34px;

line-height: 34px;

}

.st-container > input {

opacity: 0;

z-index: 1000;

}

.st-container > a {

z-index: 10;

font-weight: 700;

background: #666666;

color: #fff;

text-align: center;

text-shadow: 1px 1px 1px rgba(151,24,64,0.2);

}

1. Since we are using percentages to spread the links and inputs across the width of the screen, we might get into some rounding trouble that will make some gaps appear. In order to hide that, well use a pseudo element that will be under the links and inputs. It will have the same background color like the link elements:

.st-container:before {

content: '';

position: fixed;

width: 100%;

height: 34px;

background: #e23a6e;

z-index: 9;

bottom: 0;

}

1. Our links and inputs are still not positioned, so let’s give them their respective left values. The percentage differences allow us to equally space our 5 navigation links across the screen. Notice the styles apply to both the radio button and its adjacent link element

#st-control-1, #st-control-1 + a {

left: 0;

}

#st-control-2, #st-control-2 + a {

left: 20%;

}

#st-control-3, #st-control-3 + a {

left: 40%;

}

#st-control-4, #st-control-4 + a {

left: 60%;

}

#st-control-5, #st-control-5 + a {

left: 80%;

}

1. As you can see, we are using the adjacent sibling selector to “reach” the direct sibling of an input which is the related link element. Using the same principle, we will define a “selected” state for the link elements. Once we click on an input, we will give the sibling link element a different background colour:

.st-container > input:checked + a,

.st-container > input:checked:hover + a {

background: #821134;

}

1. Let’s also add a little triangle using the pseudo-class *:after* and give it the same color:

.st-container > input:checked + a:after,

.st-container > input:checked:hover + a:after

{

bottom: 100%;

border: solid transparent;

content: '';

height: 0;

width: 0;

position: absolute;

pointer-events: none;

border-bottom-color: #821134;

border-width: 20px;

left: 50%;

margin-left: -20px;

}

1. Let’s define a hover state for the link element:

.st-container > input:hover + a {

background: #AD244F;

}

.st-container > input:hover + a:after {

border-bottom-color: #AD244F;

1. The wrapper for the panels and the panels will have relative position and we’ll give them a width and height of 100%. The panel wrapper will also get a top and left position of 0 while we don’t touch the values for the panels (it will be auto).

The transition will be for animating the **transform** property value to the respective position:

.st-scroll, .st-panel {

position: relative;

width: 100%;

height: 100%;

display: block;

-moz-background-size: cover;

-webkit-background-size: cover;

background-size: cover;

}

.st-scroll {

top: 0;

left: 0;

transition: all 0.6s ease-in-out;

/\*enforce some hardware acceleration \*/

-webkit-transform: translate3d(0, 0, 0);

-webkit-backface-visibility: hidden;

}

.st-panel {

background: #fff;

overflow: hidden;

}

1. Define the positions for the **st-scroll** wrapper for each checked radio button. Since we know that every panel has a height of 100% we know the exact positions. We will use the transform property to translate the panel wrapper in the Y-dimension (up and down):

#st-control-1:checked ~ .st-scroll

transform: translateY(0%);

}

#st-control-2:checked ~ .st-scroll {

transform: translateY(-100%);

}

#st-control-3:checked ~ .st-scroll {

transform: translateY(-200%);

}

#st-control-4:checked ~.st-scroll {

transform: translateY(-300%);

}

#st-control-5:checked ~.st-scroll {

transform: translateY(-400%);

}

1. The heading will be placed in the center of the screen with a negative top margin in order to “pull” it up a bit:

.st-panel h2 {

color: #e23a6e;

text-shadow: 1px 1px 1px rgba(151,24,64,0.2);

position: absolute;

font-size: 154px;

font-weight: 900;

width: 80%;

left: 10%;

text-align: center;

line-height: 50px;

margin: -70px 0 0 0;

padding: 0;

top: 50%;

-webkit-backface-visibility: hidden;

}

1. Every time we click on an input, we want the respective heading to run an animation. It will animate a bit from the top and fade in at the same time. In order to select the correct heading, we will use the general sibling combinator:

#st-control-1:checked ~ .st-scroll #st-panel-1 h2,

#st-control-2:checked ~ .st-scroll #st-panel-2 h2,

#st-control-3:checked ~ .st-scroll #st-panel-3 h2,

#st-control-4:checked ~ .st-scroll #st-panel-4 h2,

#st-control-5:checked ~ .st-scroll #st-panel-5 h2

{

animation: moveDown 0.6s ease-in-out 0.2s backwards;

}

@-webkit-keyframes moveDown

{

0% {

-webkit-transform: translateY(-40px);

opacity: 0;

}

100% {

-webkit-transform: translateY(0px);

opacity: 1;

}

}

@-moz-keyframes moveDown

{

0% {

-moz-transform: translateY(-40px);

opacity: 0;

}

100% {

-moz-transform: translateY(0px);

opacity: 1;

}

}

1. The paragraph will have the following style:

.st-panel p {

position: absolute;

text-align: center;

font-size: 16px;

line-height: 22px;

color: #8b8b8b;

z-index: 2;

padding: 0;

width: 50%;

left: 25%;

top: 50%;

margin: 10px 0 0 0;

-webkit-backface-visibility: hidden;

}

1. While the heading of a panel will move down, the paragraph will move up:

#st-control-1:checked ~ .st-scroll #st-panel-1 p,

#st-control-2:checked ~ .st-scroll #st-panel-2 p,

#st-control-3:checked ~ .st-scroll #st-panel-3 p,

#st-control-4:checked ~ .st-scroll #st-panel-4 p,

#st-control-5:checked ~

.st-scroll #st-panel-5 p {

animation: moveUp 0.6s ease-in-out 0.2s backwards;

}

@-webkit-keyframes moveUp

{

0% {

-webkit-transform: translateY(40px);

opacity: 0;

}

100% {

-webkit-transform: translateY(0px);

opacity: 1;

}

}

@-moz-keyframes moveUp

{

0% {

-moz-transform: translateY(40px);

opacity: 0;

}

100% {

-moz-transform: translateY(0px);

opacity: 1;

}

}

1. In order to make out layout a bit more fun, we’ll add a colour class and “invert” the colours for some panels and their content elements: (Pink Heading Text)

.st-color {

background: #fa96b5;

}

.st-color h2 {

color: #fff;

text-shadow: 1px 1px 1px rgba(0,0,0,0.1);

}

.st-color p {

color: #fff;

}

1. Finally, we will add some media queries to control the position and font size of the elements for smaller screens:

@media screen and (max-width: 520px)

{

.st-panel h2

{

font-size: 42px;

}

.st-panel p

{

width: 90%;

left: 5%;

margin-top: 0;

}

.st-container > a

{

font-size: 13px;

}

}

@media screen and (max-width: 360px)

{

.st-container > a

{

font-size: 10px;

}

}

1. Run and test
2. Fin.