“Web Design DeCal” Hands-On Session 6 (Design)
User Experience: Sidebar

In this week’s hands-on session, you will be creating a sidebar similar to that of Medium (http://www.medium.com) This will give you a demonstration of UX principle #5 (Less is good) and #6 (Design for performance) mentioned in the lecture.

I. CSS

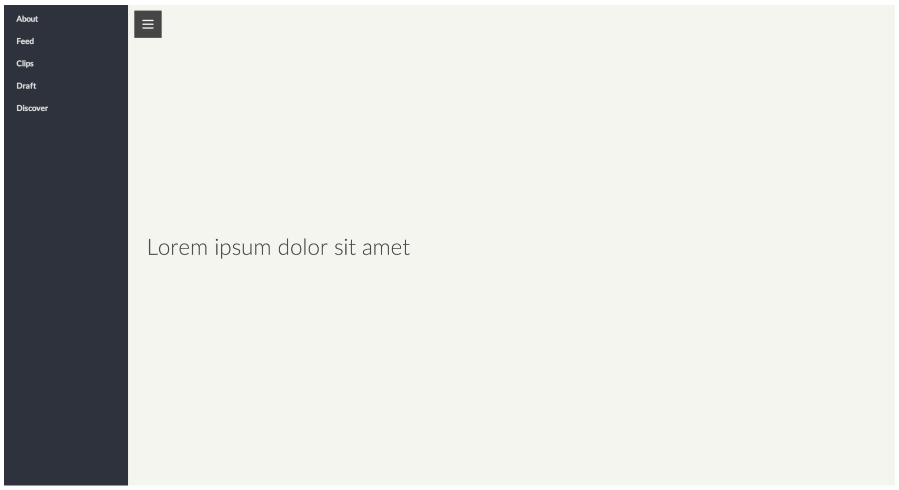
Before we begin, take a look at the screenshots “state1.jpg” and “state2.jpg.” We will be creating a page where if the user presses the toggle button on the top-left corner, a hidden sidebar will appear.

The content section is already created for you. In Part I, you will be creating the CSS rules for the sidebar and the toggle button.

A. Let’s start off with the toggle button. Let’s adjust the size of the button and make it look like a button by setting “width: 24px; padding: 10px; background: #4a4a4a; cursor: pointer;” and put it on top of the content, on an appropriate location: “position: absolute; top: 10px; left: 10px;”

B. For the sidebar, we need to first make it a layer that can go on top of other elements by setting “position: absolute; top: 0px;” Then, we can customize the styles to make it look like a sidebar by setting “background: #2f353f; width: 200px; height: 100%;” Note that the styles for individual sidebar items are already filled in for you.

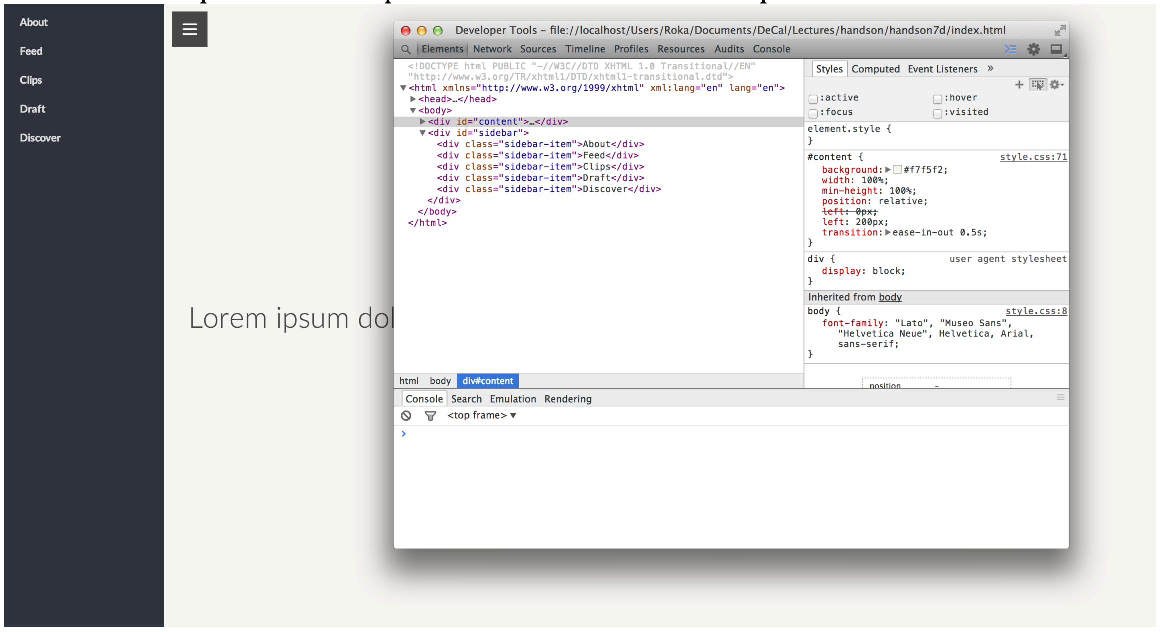
C. Finally, we want the sidebar to be hidden by default, and when a user presses the toggle button, the content slides to reveal the sidebar underneath. To do this, add “z-index: -1;” to #sidebar.



II. Animation and Performance

Now, we will make the content actually move using CSS transforms and JQuery. Most of the transition itself is handled by the CSS, and we use JQuery to respond the click event. As such, the typical workflow is to modify the CSS first for each state, see if it works, and then implement the JQuery.

A. So far, we have coded the initial state of the page. What would be the toggled state be? It should be that the content slides to the right, revealing the sidebar underneath. To do this, add "left: 200px;" to the content – which will push the content to the left by 200px. Keep the "left: 0px;" so that we can experiment the performance in the next steps.


About
Feed
Clips
Draft
Discover
Lorem ipsum dol
Developer Tools – file://localhost/Users/Roka/Documents/DeCal/Lectures/handson/handson7d/index.html
Elements Network Sources Timeline Profiles Resources Audits Console
Styles Computed Event Listeners
:active :hover
:focus :visited
html body div#content
Console Search Emulation Rendering
<top frame>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
▼<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  ▶<head>…</head>
  ▼<body>
    ▶<div id="content">…</div>
    ▼<div id="sidebar">
      <div class="sidebar-item">About</div>
      <div class="sidebar-item">Feed</div>
      <div class="sidebar-item">Clips</div>
      <div class="sidebar-item">Draft</div>
      <div class="sidebar-item">Discover</div>
    </div>
  </body>
</html>
```

```
element.style {
}
#content {                                  style.css:71
  background: ▶ #f7f5f2;
  width: 100%;
  min-height: 100%;
  position: relative;
  ~~left: 0px;~~
  left: 200px;
  transition: ▶ ease-in-out 0.5s;
}
div {                            user agent stylesheet
  display: block;
}
Inherited from body
body {                                       style.css:8
  font-family: "Lato", "Museo Sans",
     "Helvetica Neue", Helvetica, Arial,
     sans-serif;
}
```

B. So far, the content changed its position without actually animating. To add a transition effect, add "transition: ease-in-out 0.5s" Try turning the "left: 200px" rule that you just added on and off to simulate the CSS effect.

Save your progress so far before moving on to the next steps!

C. It might seem that the transition looks fine, but it's because you are testing on Chrome, which has the best Javascript rendering engine. In this step, you will see that there's a better method to do the animation that provides consistently fast performance for even other browsers.

If you have Safari, open up the web page in Safari and open the Web Inspector. Try turning the "left: 200px;" on and off as you did in Chrome in the last step. You will see that the transition stutters a little bit.

We can make the transition a lot smoother by using –*webkit-transform* rather than *left*. –webkit-transform uses hardware acceleration, meaning it uses your computer's graphic card to render the animation. To do this, remove "left: 200px;" and add "-webkit-transform: translateX(200px;)" This accomplishes the same effect that "left: 200px" did, moving the content by left 200px, but uses hardware acceleration to make the transition a lot smoother.

The gist is: We will use "-webkit-transform: translateX(200px)" to make the transition.

III. JQuery

A. Now that we know what we should do to accomplish the transition, all we need to do is to implement the JQuery. Notice that to make the transition, you just have to add in one style rule: "-webkit-transform: translateX(200px)" But how do we set this programmatically?

   To do this, we can use JQuery .click() and .css() functions.
   First, inside $(document).ready() in script.js, add:

```
$('#toggle').click(function(){
alert('!');
});
```

   This will add a click event handler for #toggle button. $('#toggle') selects an element named with id "toggle" and .click() makes sure that when a user clicks on that element, whatever is inside function() will execute. Right now, we have it so that when a user clicks on #toggle, the browser will alert an exclamation point.

B. Let's now add JQuery code that does the actual transition. Inside the function, add

```
$('#content').css({'-webkit-transform': 'translateX(200px)'});
```

   This adds a CSS rule "-webkit-transform" of value "translateX(200px)" when a user presses the toggle button.

C. We are not quite done yet! Notice that if you click the toggle button again, the content does not go back. This is a clear user experience issue! How do we make sure that each time a user "toggles" a button, it executes different functions? JQuery has a toggle() function to take care of that.

   Remove the click function you just created and replace it with:

```
$(‘#toggle’).toggle(function(){
$(‘#content’).css({‘-webkit-transform’: ‘translateX(200px)’});
}, function(){
$(‘#content’).css({‘-webkit-transform’: ‘translateX(0px)’});
});
```

JQuery toggle() function takes in <u>two</u> functions – one for when a user clicks on the button *odd number of times* and the other for *even number of times*. We can use this to have two functions that slides the content to the right, and back to the original position.