“Web Design DeCal” Hands-On Session 9 (Design)
User Interface Elements

Introduction

So far, you guys have learned all the essential skills to make a website - HTML for structure, CSS for design, and Jquery for interactions. For HTML/CSS, everything culminated together when we covered positioning, as we covered how to make website layouts by dividing each components into different pieces. For JQuery, we are going to teach you how to create some of the most useful and popular user interface elements via JQuery. These user interface elements greatly enhance user experience, and is the central to make your website look more like a "web application" than a static website.

1. Popovers and Tooltips

Popovers are variations of a dropdown menu. It's used to reveal additional information upon a click or a hover event. Examples of a popover include a share on social media options as you can see in this screenshot, and facebook user profile popovers.

Tooltips are used to convey a caption of a button or image when a user hovers on them. Tooltips are important if you have a menu with just icons, since icons might not be enough to convey what that button does - for example, the menu icon.

How would you create these popovers and tooltips? Let's look at the HTML/CSS structure first. You can roughly divide them up to two sections - the arrow and the actual content. As such, you divide the HTML accordingly - a div for the arrow and a div for the content. You would have additional item divs inside the content if you are making a list popover. For a tooltip, you can just write the caption text itself. And of course, these are stacked vertically, so we don't need to do anything.

Everything looks easy... until you reach the arrow. It's a triangle. Now, you might wonder, how do I create a triangle in CSS? Creating a triangle in CSS is one of those things that require a CSS hack. There are three ways to do it - transparent borders, webkit transform, and SVG. Here, we are going to use transparent borders.

To create a triangle using CSS borders, set width and height to 0px, and set the border to <size> solid transparent for non-base sides, and <size> solid <color> for the base side. For example, if you want to create a white triangle pointing upwards, the base side is bottom, and non-base sides are left and right. Therefore, the CSS for this triangle should be:
width: 0px;
height: 0px;
border-left: 5px solid transparent;
border-right: 5px solid transparent;

border-bottom: 5px solid white;

Now, we can go onto make the actual popover. In your HTML, we filled in the structure of your popover. Uncomment the commented block for popover-content. When the user clicks on a popover-button, the popover-content div will show up.

A. We need to make sure we can position the popover-content relative to the position of the popover, so add position: absolute to #popover-content.

B. Now let's style the popover content itself. Add "padding: 10px; font-size: 13px; font-weight: 700; color: #757c78; width: 120px; border-bottom: #DDD;" to .popover-item. Then, add "background: white; box-shadow: 0px 0px 5px #DDD; border-radius: 5px; to #popover-box. "

C. Let's draw some triangles! Add "width: 0px; height: 0px; border-left: 8px solid transparent; border-right: 8px solid transparent; border-bottom: 8px solid white;" to #popover-arrow. We now have a triangle, but you can see that it's being covered by the popover-box, and is not centered. You can use z-index to rearrange the priority for the layers. Add "z-index: 1; position: relative; margin: 0px auto;" to #popover-arrow to give it a priority over the popover-content, and to center it.

D. Last but not least, we want to place the entire popover somewhat close to the button, and on top of it. Add "top: 32px; left: -12px; z-index: 1;" to #popover-content.

Right now, popover is visible at all times. The desired effect is to make the popover hidden by default, and show it if and only if the user clicks on the button, and then hide if the user clicks on the button again. We can use JQuery to make this work.

In script.js, add:

```
$('#popover-button').toggle(function(){
$('#popover-content').show();
}, function(){
$('#popover-content').hide();
});
```

Finally, add "display: none;" to $('#popover-content') so that it's hidden by default.

Tooltip is extremely similar to the popover. Try to experiment with the tooltip on your own for the next 5 – 10 minutes.

2. JQuery Masonry

If you have ever used Pinterest before, you might have wondered - how do they create this cool tile of images? It's interesting because if you just float items left, or use inline-block, items will just stack like a grid. JQuery Masonry allows you to achieve the same effect on Pinterest for your website. Before we actually get to JQuery Masonry, let me explain a bit more about some subtleties of CSS float.

As you all now, when items are stacked horizontally via float, items will continue stacking next to each other as long as there is space available. When there is not enough space, it moves to the next row, obviously. What's important to mention here is the height of the row generated. The height of the row is equal to the tallest element in that row if you use float. This is why when you use float, as it puts items in the next row, you can see a giant empty space between items with uneven heights.

Jquery Masonry allows you to bypass this property by generating a tile of items automatically, depending on the height of each element. If you would look at the CSS code generated by Masonry, you can see that it designates the entire container as position: relative, and all of the items are positioned absolute. Masonry automatically calculates where each item "should be," and positions the items to its appropriate locations, generating a Pinterest-style UI.

Now, when should you use JQuery Masonry? It's extremeley useful if you want to build an image-heavy website, because it allows you to preserve the proportions of your images. If you want to preserve the proportions of your images, you either have to fix your width or height. This results in uneven sizes of your items. As you saw in our float example, when the sizes of items are uneven, you get a huge empty space. And if you were to create a fix sized grid, you lose the proportions of your image, which means that you will get an addressbook full of fat heads and skinny heads as you can see in this screenshot. With JQuery Masonry, you can preserve the proportions of your images, yet at the same time the spacing between grid items are preserved. And of course, it looks freaking sexy.

A. Before we get to the actual javascript, let’s do some styling first:

```
.item {
  width: 225px;
  float: left;
  position: relative;
  border-radius: 5px;
  overflow: hidden;
  margin-bottom: 15px;
}
```

```
.item-caption {
   position: absolute;
   bottom: 0px;
   width: 205px;
   padding: 10px;
   background: -webkit-linear-gradient(transparent, rgba(0, 0, 0, .7));
   color: white;
   font-size: 24px;
   font-weight: 300;
}
```

B. Because we are using regular float right now, we have some empty spaces here and there. Let's use JQuery Masonry to close the gap.

Add the following code inside the head tag, above your script.js import:
`<script src="assets/jquery_masonry_min.js"></script>`

C. Then, add:

```
$('#container').masonry({
        itemSelector : '.item',
        gutterWidth: 15
});
```

This tells the masonry script that we want to designate all of the divs with the class .item to be a part of the masonry grid on #container.

3. Sticky Nav

For our final tutorial of user interface elements, we will cover sticky navigation. Sticky navigation is popular for landing pages and onepage-scroll website like our Web Design DeCal website. Sticky navigation is a technique that makes a navigation bar that's usually at the bottom and scrolls with your website stick to the top when the navigation bar reaches the top.

It's usually used for landing pages where you put the navigation bar on the bottom, and the cover photo and title on top. For these types of websites, you want to emphasize the emotional appeal as much as possible by showcasing a large cover photo and title, usually accompanied by a tagline. This means that you are going to have to put your navigation bar on the bottom of the cover, if you want to dedicate emphasis as much as possible to the cover. At the same time, however, you don't want to sacrifice usability that a traditional top navigation bar offers. Top navigation bar provies immediate accessibility to commonly accessed navigation items even when the user scrolls the page.

That's why we use a sticky nav bar. The navigation bar is starts off on the bottom, like the usual landing page layout, but when the user scrolls, the navigation bar becomes just like a top navigation bar and remains in the position even when the user scrolls. What's important in building a sticky navigation bar is to simulate the "stick" - effect: when the top bar reaches the top of the page, we instantly make it remain in that position, so it seems as if the navigation bar is glued to the top.

We can formulate this specification that we want into an actual JQuery code.
1. We have to detect when the top bar reaches the top of the page when the user scrolls.
The first part, the top bar reaching the top of the page, can be calculated by finding how much a user has scrolled. Because the top bar is 432px away from the top of the page, if the user has scrolled 432px, the top bar must have reached the top.
You can combine this with JQuery scroll event, which is an event handler just like a click event that detects when a user scrolls.

```
$(window).scroll(function(){
	var offset = $(window).offset().top
		if (offset > 432) {
		 // do something
		}
});
```

2. Now, we want the navigation bar to stick to the top. You have all done this before - position:fixed! Inside the if statement, add a position:fixed; top: 0px; so that it remains in that position, and is on the top of the page.

```
$(window).scroll(function(){
	var offset = $(window).offset().top
		if (offset > 432) {
			$('#cover-nav').css({'position': 'fixed', 'top': '0px'});
		}
});
```

You can now check that it does stick to the top of the page when you scroll. The only problem is that it doesn't come off when you scroll back to the top. To fix this, we are going to change a few parts in our if statement.

3. We want the top bar to remain position:fixed if the user's scrolled position is 432px or greater. Thus, we can add an else statement to put the navigation bar back to its original position.

```
$(window).scroll(function(){
	var offset = $(window).offset().top
		if (offset > 432) {
```

```
                $('#cover-nav').css({'position': 'fixed', 'top': '0px'});
        } else {
                $('#cover-nav').css({'position': 'absolute', 'top': 'auto'});
        }
});
```

If we have time we will go over how to implement the same effect using JQuery waypoints. $(window).scroll() is very taxing on performance (can you guess why?) but waypoints allows you to achieve the same effect, but with greatly improved performance.