Web Design DeCal

Homework 8 November 19th

Carousels and Sidebars in jQuery

Required Tools: Text Editor & Google Chrome

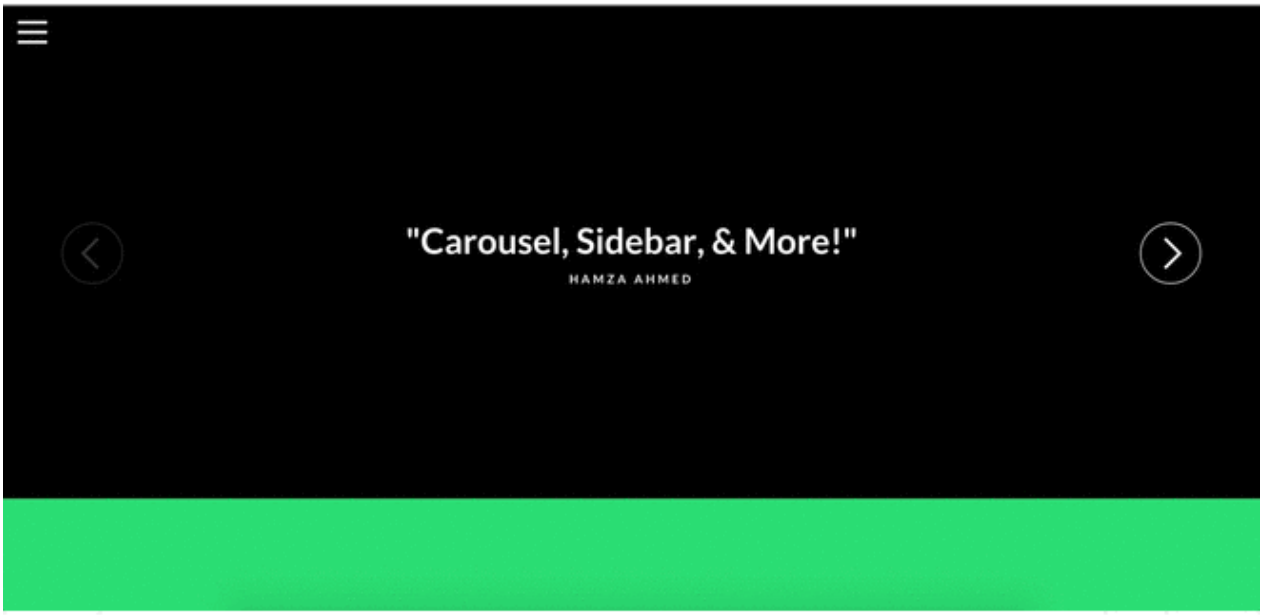
You're going to build a four-section page, with a working carousel and sidebar!

Assignment

First, we're going to build a sidebar and carousel from scratch using HTML and CSS. Next, we're going to implement them using the power of jQuery. This homework will take jQuery a step further. Please read through the entire assignment before you begin.

Download Assignment

This is what you will be creating for this homework:



Question 1: Write HTML for the Sidebar

Sidebars are very common in web design. They are essentially navigation bars that are tucked away when not in use in order to draw the user's attention to the actual content.

Our sidebar will consist of three main components:

1. The **Sidebar Container** which, just like any other container, will be the parent of the inner elements
2. The **Sidebar Unordered List (ul)** which we will use to create the sidebar items in list form
3. The **Sidebar List Items** which will be the buttons that the user can click to jump to a desired section of the page

Take some time to look over the HTML structure of the page. The first thing you'll notice is that there is a sidebar button which is already coded for you. The sidebar button will be used to active/open the sidebar. Additionally, follow the comments in the HTML docuement and add the necessary information in each of the sections. Make sure to uncomment the information you provide.

Let's build our sidebar in HTML:

- In the HTML element called ".sidebar-container", add an unordered list (use the ul tag) with no id or class
- Within that unordered list, add four list items (use li tags) with the class "sidebar-item", referring to the four sections of our page
- Our four sections will be Home, Name, Major, and Fact. Go ahead and add one section title inside each list item (Home in the first li, Name in the second li, etc.)

Don't worry if you can't see any changes yet.

Question 2: Write HTML for the Carousel

We want the first section of our page to be a carousel/slider. A carousel usually consists of a collection of images that are displayed one at a time and can be navigated through using "next" and "prev" buttons.

Carousels usually consist of three main components:

1. **Carousel Container**
2. **Carousel**
3. **Carousel Items**

The **Carousel Container** is what defines the viewing space of the entire carousel. In other words, it is the area that is visible in the carousel at any given time.

The **Carousel** itself will contain all the carousel items. You can think of it as a film reel that has a width equal to the sum of the widths of the carousel items.

The **Carousel Items** are the individual slides/sections of the carousel.

Let's code the carousel in HTML inside the HTML element called ".carousel-container":

- Add a div with an id of "carousel"
- Inside "#carousel", add another div with the class "carousel-item"

We want 5 carousel items for our carousel. Before you duplicate your code five times, let's add some content to the carousel item itself. In this homework, the carousel will be used to display a selection of your favourite quotes.

Add the following code inside the HTML element ".carousel-item":

- Create a div with a class called "quote"
- Add another div inside ".quote" with a class of "author", which we will use to mention the author of the quote

Now that each carousel item is ready, duplicate the code pertaining to the carousel item 5 times and add 1 of your favorite quotes and its author inside each carousel item.

Finally, we want to have two buttons, "next" and "prev" in order to be able to navigate through the carousel. If you take a look at your HTML document, you'll notice these buttons have been provided for you. Please make sure you understand how they've been implemented.

You should have something like this:

Question 3: Write CSS for the Sidebar

Now, let's make that sidebar actually look like a true sidebar.

The way we're going to make our sidebar work is we're going to give it a fixed width of 280px, and have be *off the screen* by default (yes, you can do that!). To be more specific, we're going to assign it a **transform: translateX(-280px)** to shift it 280px to the left of the left edge of the screen, so that it it

touches the left edge of the screen but is not immediately visible. In your CSS stylesheet, cretate a CSS element called ".sidebar-container" and add the following code to it:

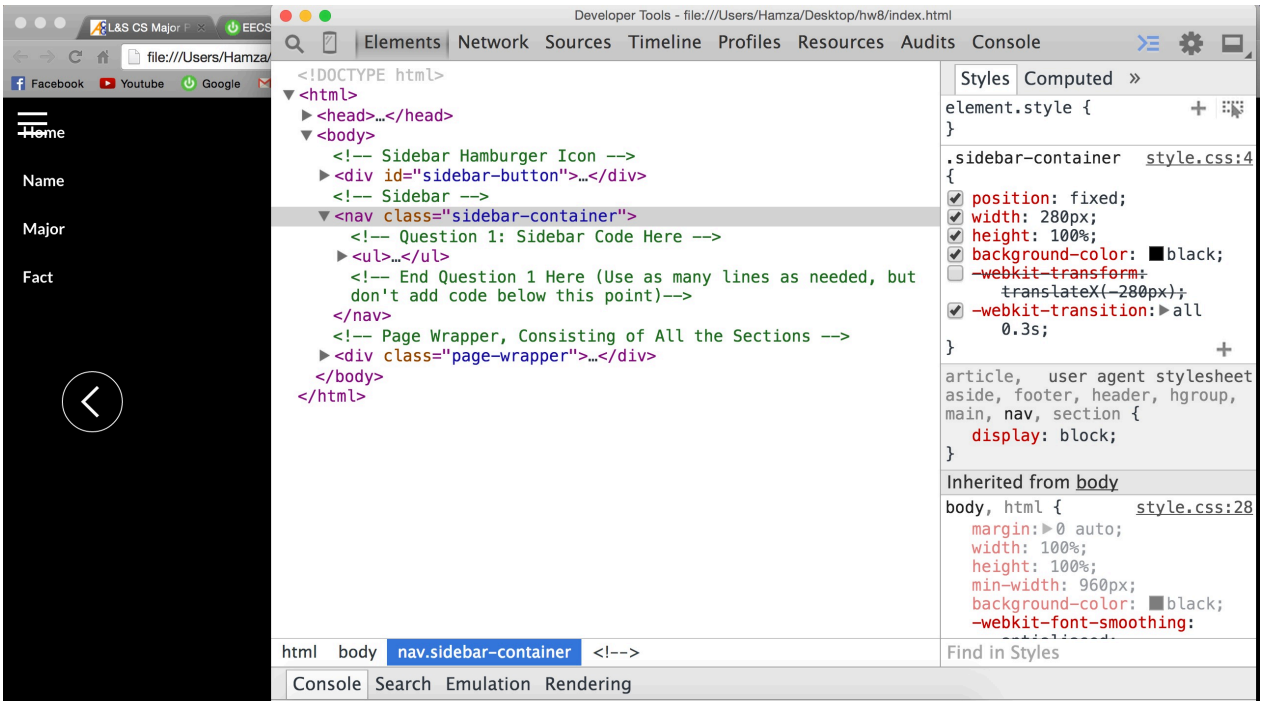
- Make it fixed on the screen (Hint: positioning)
- Give it a width of 280px
- Make it occupy the full height of the screen
- Give it a background color of black
- Have it be 280px to the left of the left edge of the screen (-webkit-transform: translateX(-280px))
- Define a transition of 0.3s to animate the "slide out" effect (-webkit-transition: all 0.3s;)

Becareful! Remember that transforms and transitions need to be optimized for each browser via **vendor prefixes**. (Optional) Add vendor prefixes to enable browser support for WebKit browsers, Mozilla Firefox, and the default browser.

Next, lets style the sidebar items. Create a CSS element for the class ".sidebar-item" and add the following code:

- Give it a font weight of 700
- Give it a font size of 16px
- Set its font color to be white
- Give it a top and bottom padding 15px and a left and right padding of 20px

Remember that your sidebar is sitting 280px to the left of the left edge of the screen. If you open up the Web Inspector, and deselect the transform property on the HTML element ".sidebar-container", this is what you should see:



Question 4: Write CSS for the Carousel

There are two ways to make a working carousel:

1. Using position absolute/relative: A little trickier, but allows you to click "next" on the last item to get back to the first item
2. Using overflow: Easier, but you cannot switch to next item after reaching the last item

In this assignment, we will be using method #2: overflow.

Before explaining carousels, let's first go over the CSS overflow property. There are three values that are frequently used for CSS overflow:

1. **overflow: visible**: This is the default. If content goes outside of a div, the content will still be visible. For example, if you have a width 40px element inside a width 20px element, the inner content will be visible.
2. **overflow: scroll**: If content goes outside of a div, the content outside the boundary of the parent div will not be visible. However, you will be able to scroll to the content. For example, if you have a height 60px element inside a height 30px element, the content will initially be invisible, but you can scroll through the parent element and see the remaining element.
3. **overflow: hidden**: If content goes outside of a div, the content outside the boundary of the parent div will not be visible. You cannot even scroll.

In our carousel, we will have a carousel container with a width of 960px. This will be the section of the carousel that is visible at all times. Inside it, there will be 5 slides of quotes. Initially, only one quote will be visible; however, when the user clicks on "next", the currently visible image will scroll to the left and the user will see the second quote, and so on.

Lets begin by styling the carousel container. Create a CSS element for the class ".carousel-container" and add the following code:

- Give it a width of 960px
- Make it horizontally centered (use the margin trick!)
- Give it a height of 500px
- Hide the overflow so that the user can only see one quote at a time
- Make it position: relative

Next, let's style the actual carousel. Our carousel will consist of 5 quote slides, with each one having a width of 960px. Create a CSS element for the ID "#carousel" and add the following code:

- Give it a width of (5 slides x 960px each) = 4800px
- Make it occupy the full height of the carousel container
- Give it a margin-left of 0px (we will manipulate this later)
- Define a transition of 0.3s to animate the sliding effect (-webkit-transition: all 0.3s;)

Finally, we need to style the individual carousel items. Create a CSS element for the class ".carousel-item" and add the following code:

- Give it a width of 960px
- Set it to be position: relative
- Make it occupy the full height of the carousel
- Stack the items horizontally by using float: left

By setting overflow: hidden on the carousel container, any element that goes outside of the carousel container will not be visible. Since we have a carousel with width 4800px, we can only see the first carousel item, and any remaining item will be invisible.

As mentioned before, the carousel requires "prev" and "next" buttons in order to navigate through it. These buttons are already CSS-styled for you, as are the quotes and authors of each section in the carousel.

This is what it should look like:

Question 5: Other CSS and Class Manipulation

Let's quickly add some styling to the other sections in our page. Since this is not the main focus of the assignment, this question will be very straightforward. The text in each section is already styled for you.

Each section of our page has a class called section. We want the sections to look similar to eachother. Create a CSS element for the class ".section" and add the following code:

- Make it full width
- Give it a height of 500px
- Make it position: relative

Finally, let's give each section a distinct background color:

- Give #name-section a background color of #2bde73

- Give #major-section a background color of #8bb2af
- Give #fact-section a background color of #fff
- Give #fact-text a font color of black

This is what part of your page should look like:

Great! When we use jQuery, we're going to use classes to manipulate the state of the sidebar, page wrapper, and sidebar button. Each of these has two states: **"active"** and **"inactive"**, or in other words, "open" and "not open". We've defined three classes for you which can be found at the end of the stylesheet:

- **sidebar-active**: when added to the sidebar, this will bring the sidebar back into view by giving it setting its translateX value back to 0.
- **wrapper-active** and **button-active**: these classes will push the wrapper and button to the right by 280px to accomodate for the sidebar when in an active state
- **no-scroll**: this will prevent the user from scrolling when the sidebar is active/open

Please make sure you understand the active state classes that we provided for you, as they are crucial to implementing the jQuery questions below.

Question 6: Using jQuery to Create a Functional Sidebar

As of right now, our sidebar does not respond when the user clicks on the sidebar button. Let's use the power of jQuery to change that.

Our sidebar button has two main purposes:

1. If the sidebar is active, the sidebar button should close the sidebar
2. If the sidebar is inactive, the sidebar button should open the sidebar

Our sidebar button has two roles to play depending on the state of the sidebar. How can we implement it? We can use a click event listener on the sidebar button, along with some conditionals!

Let's first think about the case when the sidebar is inactive. What all do we need to do in order to change its state from inactive to active?

The active classes we defined earlier come in handy here. We simply need to add the respective active class to each of the wrapper, button, and sidebar!

We also need to stop the user from scrolling when the sidebar is active. We can achieve this by adding the no-scroll class to the body tag! However, want this to happen ***after*** the sidebar opens. In other words, we want this to take effect only after the entire sidebar transition has finished (remember from the CSS code that the sidebar transition is 0.3 seconds long). How do we apply a delay to one particular line of code in jQuery? We can use the setTimeout() function!

The following code adds the no-scroll class to the body tag with a delay of 300 milliseconds:

```
setTimeout(function() {
    $('body').addClass('no-scroll');
}, 300);
```

Now let's think about the case when the sidebar is active. How do we go make the sidebar go from an active state to an inactive state? We simple remove the active classes from the wrapper, button, and sidebar!

The **hasClass()**, **addClass()**, and **removeClass()** methods will be extremely useful in writing these functions. Use the jQuery API if you're unsure about what each one does.

Now its time to put all the pieces together and write the sidebar button click event listener. Here is some psuedo-code to help you write the body of the function:

```
(When the user clicks on an the sidebar button)
If the sidebar container has the "active" class:
    Remove the "no-scroll" class from the body tag
    Remove the "active" class from the sidebar button
    Remove the "active" class from the sidebar container
    Remove the "active" class from the page wrapper
Otherwise:
    Add the "active" class to sidebar button
    Add the "active" class to sidebar container
    Add the "active" class to the page wrapper
    Add the "no-scroll" class to the body tag with a delay of 300 ms (code above)
```

Clarification: In the JavaScript file, the comment for question #6 should read "//Implement the showing and hiding of the sidebar when the user clicks on #sidebar-button here". If your file already says that, you can just ignore this.

After implementing the code above, you should have a working sidebar! You should see something similar to this:

We left the implementation of the sidebar links for the optional Extra for Experts section.

Question 7: User Experience and jQuery

We know that in terms of good user experience, we should close the sidebar when the user clicks anywhere but the sidebar. This is similar to the idea expressed in last week's modal overlay click.

In this assignment, "anywhere but the sidebar" refers to the ".page-wrapper" div. Thus, if the sidebar is active, a click on the page wrapper should change the sidebar's state from active to inactive. As far as implementation goes, this is the same idea as clicking on the sidebar button when the sidebar is open.

Implement the page wrapper click even listener that exibits the same functionality as the button when the sidebar is active. Note that you can reuse part of your code from question #6 to implement this function.

Question 8: Powering the Carousel with jQuery

As of now, the user can only see the first quote slide, and there's no way to see the remaining quote slides. We will add some jQuery to make this work.

Here's how our carousel will work:

- When a user clicks on the "prev" button, the carousel will *slide to the right*, revealing the previous quote slide

- When a user clicks on the "next" button, carousel will *slide to the left*, revealing the next quote slide

Now, the question is: "How can we achieve slide to the right/left effect?"

To do this, we simply adjust the margin-left property of the carousel (which has width of 4800px). It is important to note that you can have a margin-left of negative value. For example, if you have margin-left: 100px, the div will make 100px of spacing from its current left position, pushing it to the right. If you have margin-left: -100px, the div will make 100px of spacing toward the opposite direction, pushing it to the left.

- To achieve the slide to the left effect, we add -960 to the current value of margin-left. This will push the carousel by 960px, which is the width of each carousel item, toward the left.
- Likewise, to achieve slide to the right, we add 960 to the current value of margin-left. This will push the carousel by 960px toward the right.

To get the current value of margin-left in an integer format, use:

```
parseInt($('#carousel').css('margin-left').replace("px", ""));
```

This will get the current CSS value of margin-left from "#carousel", take out the trailing "px" keyword, and then convert it to an integer.

To achieve the "slide" effect, simply modify the CSS margin-left value of the carousel via the css() function. Remember that we added a transition of 0.3 seconds to our carousel, so everytime we modify the margin-left value, the change will be animated over 0.3 seconds.

You should also make sure that you have not reached the start or the end of the carousel. If the user is on the first slide and clicks the "prev" button, the carousel should not slide to the right. To do this, you will need to make use of conditionals.

Use the following pseudo-code to write the "#carousel-next" click function:

```
(When the user clicks on an the carousel-next button)
Get the margin left value of the carousel and store it in a JavaScript variable
If the current margin is equal to -3840 (i.e. we are at the end)
    return false
Otherwise:
    Update the margin-left of the carousel to be the current margin - 960
```

Use the following pseudo-code to write the "#carousel-prev" click function:

```
(When the user clicks on an the carousel-prev button)
```

```
Get the margin left value of the carousel and store it in a variable
If the current margin is equal to 0 (i.e. we are at the beginning)
    return false
Otherwise:
    Update the margin-left of the carousel to be the current margin + 960
```

Note: A cool feature to add would be to dim the "prev" and "next" buttons if you reach the end/beginning of the carousel. This will be left as an exercise.

Extra for Experts (Optional)

The links in our sidebar still do not work. That means there's room for improvement!

Here's the expected behavior: when the user clicks on a particular link, the page should animate and scroll to the desired section smoothly. After the animation is done, the sidebar should collapse.

Before we write some jQuery, each sidebar item needs to know where exactly it needs to jump to. To do that, we need to wrap each list item in the sidebar with an anchor tag and an **href** attribute. For example, if we wanted to jump to the name section, we'd do the following:

```
<a href = "#name-section">
    <li class = "sidebar-item"> Name </li>
</a>
```

The following function allows you to animate the html and body tags of a page to a particular section:

```
$('html, body').animate({
    scrollTop: $( $.attr(this, 'href')).offset().top
}, 300);
```

To do this, we need to create a click event listener for anchor tags. The following pseudo-code will help you write the expected function:

```
(When the user clicks on an anchor tag)
remove the no scroll class on the body
animate the html and body tags using the above function
Do the following code after a delay of 0.3 seconds:
    Remove the active class from the sidebar container
    Remove the active class from the sidebar button
    Remove the active class from the page wrapper
return false
```

Submission

Submit the following files as a **zip** file through Assignments under the Assignments tab of the WDD Portal:

- index.html
- assets
  - style.css
  - script.js
  - jquery-1.9.1.js
  - arrowleft.jpg
  - arrowright.jpg

Your assignment must be submitted as a **zip** file. Submission will automatically fail if your submission does not contain the index.html or if is not a zip file.

This assignment is due by Thursday, **November 19th** at 5PM.