jQuery SideBar Folding Navigation

Using jQuery

Project Objective

This project will end with the exact same fold down menu that we did last week with JavaScript, but this time using jQuery.

This is a great way to see how the syntax of jQuery can lead to shorter more compact scripts, that are perhaps a bit easier to write.

The start file already has the jQuery library from the CDN linked. Just add your script inside the script tags.

```
</main>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
    //your script here
</script>
```

Hiding the Submenus

```
<script>

    $('ul li ul').css("display", "none");

</script>
```

You can use the CSS method we saw before in jQuery to hide the sub menus.

In the version we did of the menus last week, you added the .hide-menu class.

Hide() Helper Function

```
<script>

    $('ul li ul').hide();

</script>
```

jQuery has a bunch of helper functions built in that you can just use. These functions are added to do common tasks, like hiding an element.

So instead, it is simpler to just use the .hide() method to hide the sub menus.

Add the Click Handler

```
<script>

    $('ul li ul').hide();

    $('.menulink').click( function(){} );

</script>
```

Add a click handler for the main menu links, which runs a function when one of those links is clicked.

Get the Next UL

```
$('.menulink').click( function(){

    var thisMenu = $(this).next('ul');

} );
```

You want to get the unordered list that comes directly after the link that was clicked.

$(this) gets the link, and then jQuery has a helper function called next() that will get the next element, in this case, the unordered list.

In the JavaScript version last week, you had to go up a level to the parent and find the UL that was inside that parent list item.

Showing the Hidden Menu

```
$('.menulink').click( function(){

      var thisMenu = $(this).next('ul');

      if(thisMenu.is(':visible')){
            thisMenu.hide();
      }
      else{
            thisMenu.show();
      }

} );
```

You can do this kind of thing. There is an .is() method that will return true or false depending on what is passed in.

You can pass in the :visible filter to see if the menu is currently showing or not.

However, there is an even easier way.

Do This Instead

```
$('.menulink').click( function(){

    var thisMenu = $(this).next('ul');

    thisMenu.toggle();

} );
```

jQuery has a helper function called toggle(); that will toggle visibility. So that makes doing something like this super simple.

You will notice that the menus are almost working properly.

You just need one more line to make it so only one sub menu can be open at a time.

The Whole Script

Here is the whole script. This final line says close all the menus except the one we clicked on. That one will get toggled open or closed on the next line, but the rest of them should be closed.

```
$('ul li ul').hide();

$('.menulink').click( function(){

    var thisMenu = $(this).next('ul');

    $('ul li ul').not(thisMenu).hide();

    thisMenu.toggle();

} );
```

Wrapping Up

Here is the final script with all the best practices in place.

When you move the linked file into the head, be sure to move both the link to your script file and the link to the jQuery library on the CDN and add the defer property to both of them.

```
(function(){
    "use strict";

    $('ul li ul').hide();

    $('.menulink').click( function(){

        const thisMenu = $(this).next('ul');

        $('ul li ul').not(thisMenu).hide();

        thisMenu.toggle();

    } );

}());
```