Event Handling Lab

Correcting the wireups

We may or may not be wiring up our buttons at the best time or in the best way. If you put them at the bottom, there is a chance that they may run before the buttons are loaded into the DOM. Let's make sure.

1. Add some code to the top of your JavaScript. Put in a comment "Wireups" or something similar.
2. Add an event listener to the *document* element. The event should be the *DOMContentLoaded* event. In the handler function, just put a console.log() for testing. Here's some code that should work:

```
document.addEventListener("DOMContentLoaded", function () {
  console.log("Document was loaded.  Yay!");
}, false);
```

3. Run and test to make sure you can see the message.
4. Now add your button wireups in that DOMContentLoaded event handler. Make sure they're done with addEventListener() instead of any of the other methods.
5. Run and test to make sure the buttons still work as expected.

Creating a right-click menu

Let's expand our graph element a little by allowing the user to get even more information from us. We're going to allow them to interact with the canvas via a right-click menu.

6. Get a reference to the canvas using getElementById() or another method like it.
7. Add an event listener to it for the 'contextmenu' event. That's really the right-click event.
8. Make it console.log that the event fired. Run and test to make sure your handler wired up properly.
9. Get rid of the current context menu by calling preventDefault() from the event object. (Hint: the event object will be passed in to your event handler.)
10. Run and test. Did you get a console message and no default right-click menu? If so, rock on.
11. Add a floating div to your page.

```
<div class='menu'>
  <a href="#" id='getDataLink'>Get the data</a>
  <a href="#" id='getPictureLink'>Save a copy</a>
</div>
```

12. And in the CSS, do this:

```
.menu {
  visibility: hidden;
  position: absolute;
  z-index: 100;
  width: 100px;
  height: 100px;
  background-color: grey;
  border: 1px solid black;
  border-radius: 5px;
  box-shadow: 3px 3px 3px grey;
  margin: 4px;
}
```

You've got your div hidden to start with. Now wire up your right-click function to show your new context menu:

13. Get a reference to your hidden and change its style.

```
var d = document.querySelector('.menu');
d.style.visibility = "visible";
```

14. Run and test. When you right-click, your div should appear and have two links.

Lastly, let's make one of the links do something. The getPicture button should allow the user to save a snapshot of the graph. Fortunately, the canvas tag allows that.

15. In the right-click event handler, get references to your canvas (theCanvas) and to the link (theLink). Then add code that looks something like this:

```
var dataURL = theCanvas.toDataURL("image/png");
theLink.href=dataURL;
theLink.download="results.png";
```

16. Run and test. When you right-click on your graph, a menu should pop up. If you click it's "Save" link, it should immediately download a snapshot of the graph to the user's downloads folder. (Note: If you get a security message, try commenting out your logo.)

When it is doing that, you can be finished.

17. Bonus!! Make the popup menu appear to the right of the mouse. You'll use the event object's clientX and clientY properties to set the top and left properties of the div. Something like this will work:

```
theDiv.style.top = e.clientY + "px";
theDiv.style.left = e.clientX + "px";
```

18. Extra bonus!! Make the div's visibility hidden again when you click anywhere on the page.

19. Extra extra bonus!! Give the mouse a tooltip that says the Y-value of the graph on mousemove. This way the user can move the mouse to find an exact value for a point on the graph.