Assignment #3
# JavaScript

## CS193C Summer 2012, Young

We now get a chance to explore some of JavaScript's more advanced features. All problems should work in Firefox and Chrome. This assignment is due Thursday August 2$^{nd}$ at 3:15pm.

## College Information

Our first task is to create an online college database. This program should work in IE and in Firefox. Here's a screenshot of this webpage:



As you can see the webpage features a table displaying information on Bay Area Colleges and Universities. At the right are two sets of controls. The Search Criteria allows the user to limit the display to "Public", "Private", or "Public and Private" Universities. It also allows users to specify the Maximum Tuition, Maximum High (75 percentile) SAT score or the Minimum Low (25 percentile) SAT score. A second set of controls tells the webpage what information it should display for each of the universities. Information available includes the university's full name and an alternate nickname, SAT information, Tuition (and Room and Board) information, and system information. When the "Update" button is clicked the table is regenerated.

**Implementation Notes**

I've included a list of Bay Area College data from the "US News and World Reports 2003 Edition of America's Best Colleges" with this assignment's downloads (feel free to update the data with something more current, 2003 is the only year I have a copy of the report for).

**Warning:** Don't forget that the web browsers will create a <tbody> element inside a table, even if one is not explicitly declared. Depending on how you decide to access or generate the table, this may make a difference.

If you're looking for some help implementing, you might want to try implementing in the following order:

- Design the output table using static data, so you know what you want it to look like. In other words, write an HTML file without any JavaScript that displays data in the table format you want.

- Now try generating some sample data in the format of the output table, ignoring the search criteria. In other words, try generating the output table from the previous step by using JavaScript. Ignore the search criteria.

- At this point, you should have convinced yourself that you can dynamically create a table from JavaScript. Now set that work aside and work on making sure you can meet the search criteria. Develop a separate webpage that lists universities which match the search criteria in an "alert" dialog box.

- Now, put the two halves together by generating the output table based on the search criteria.

If you're interested in getting more practice, consider make each of the headers clickable and use them to sort the colleges listed. For example, if the user clicks on the "name" header, list all colleges in alphabetical order. If they click on the "tuition" header order the colleges in order of their tuition. You may also want to display the Tuition and Room and Board information in a more user friendly format (e.g., $27,204 instead of 27204).

## Stanford Maps

We will now create a Stanford-specific version of Google Maps.  Here is a screenshot:
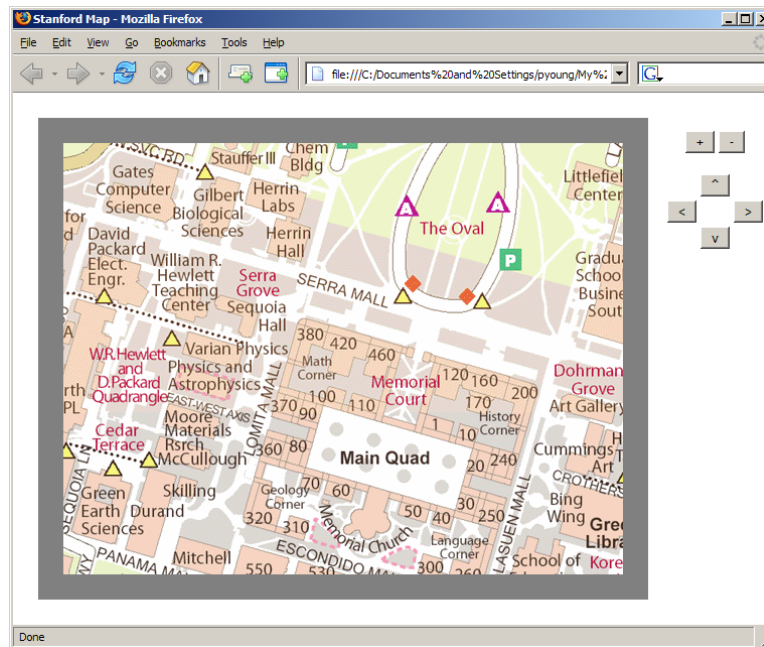


Your webpage should support the following functionality:

- The map should be enclosed within a map frame (as shown above).  The size, color, and style of the frame is up to you.

- Allow the user to click and drag on the map to change the map area which is currently visible within the map frame.

- Set the cursor to the move cursor ✛ when the user begins a drag operation.  Return the cursor back to the standard arrow cursor when the drag operation is complete.  You can set the cursor using the cursor style property.[1]

- When the user double-clicks on the map move the point double-clicked to the center of the view area.

- The map frame should resize as the window is resized.  As the window is enlarged, the map area should enlarge, as the window size is reduced the map area should be reduced.  The map frame should maintain fixed margins on all sides.  These margins should be maintained as the window is resized.  The exact margin sizes are up to you—choose something which you find aesthetically pleasing.

- The map should support zooming in and out.  As the map zooms, the point in the center of the view area should stay fixed.  In other words, if the map is centered on the Gates

---

[1] You'll only have complete control the cursor while it's over your map or map frame.  When the mouse moves over other HTML elements, they will temporarily override the cursor based on their own style settings, even if you set the cursor on the <body> tag.  If you experiment with Google Maps, you'll discover this is the same behavior they have.

Computer Science building, the Gates building should stay in the center regardless of the zoom level. We'll discuss how zooming works in more detail below.

- Provide controls to scroll left, right, up, and down. Left or right scrolling should cause the window to scroll 1/2 of the total width currently visible. Scrolling up or down should cause the window to scroll 1/2 of the total height currently visible.

- To keep things simple, if the user scrolls off the edges of the map it's okay to go ahead and let them keep scrolling even if it means you can no longer see the map.

### Creating the Map and Map Frame

The map itself is represented by a GIF file. Create a <div> for the map frame. Put an <img> tag for the map within the map frame's div. Set the map frame's overflow style property to hidden. The overflow property controls what happens when the contents of an element do not fit within the containing element. Setting the overflow property to hidden tells the web browser to hide any sections of the contents which overflow the map frame.

### Resizing the Map Frame

You can cause a function to execute when the window is resized by assigning a function to the window's onresize property. For example:

```
window.onresize = handleResize;
```

will cause the web browser to call the handleResize function when the window is resized.

To get the size of the window use the following:

```
window.innerWidth and window.innerHeight
```

Once you get the new window size, use the techniques we learned in class (and found in the Dynamic Contents handout) to change the size of the map frame. Don't forget when you assign a new top, left, height, or width to an element's style properties you must include a "px" at the end of the string.

```
elem.style.width = 5;      // wrong, wrong, wrong

elem.style.width = "5px";  // right
```

### Dragging the Map

You should be able support dragging the map around via careful use of onmousedown, onmousemove, and onmouseup events. However, a couple of points will be helpful.

These next two paragraphs are very <u>important</u>. If you ignore these you will be in for great frustration. In order to support dragging in Firefox, your *mousedown* handler <u>**must return false.**</u> If it doesn't, some versions of Firefox will have problems.

Place your event handlers on the document, not map (or whatever specific element you are trying to drag). Placing the handler on the actual element can cause problems if the user moves the mouse faster than you can move the element (if the mouse moves outside of the element you will lose mousemove and mouseup events). In theory you can place the handlers on the <body> tag. However, in my experience you're better off putting it on the document itself. You can do this programmatically like this:

```
document.onmousemove = handleMouseMove;
document.onmousedown = handleMouseDown;
document.onmouseup = handleMouseUp;
```

```
document.ondblclick = handleDblClick;

window.onresize = handleResize;
```

### Navigation Rosette

As previously noted, you need to provide controls for zooming in and out and for scrolling in each of the four cardinal directions (up, down, left, right). The easiest way to provide controls is simply creating push buttons using <input> tags inside an HTML <form>.  You can see a closeup of my controls at right.
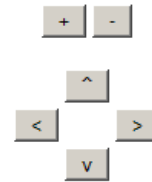
If you want to get fancier, you can replace the buttons with images (or use the <button> tag to combine a pushbutton with an image).  If you're feeling more ambitious you can move the navigation buttons onto the map (actually not that difficult).  You can also create a zoom slider control as on Google Maps.

### Zooming

We support zooming by having four separate GIFs—"map-xl.gif", "map-l.gif", "map-m.gif", and "map-s.gif".  We swap back and forth between the GIFs as the user zooms in and out. You'll need to do some careful arithmetic combined with setting the position of the GIFs to make sure that the same map point stays in the middle of the viewing area as the user zooms. Don't forget to preload the map images for smooth operation when the user first starts zooming.

If you want to get some extra practice, here are some things you can try adding:

- Add a text field and allow the user to enter in building names.  Center on the building or if you want to get fancy add a flag or map pin.

- The default application can instantly jump from one location to another in response to double clicks or scroll requests.  A more gradual scrolling may help the user by allowing them to see the relationship between the original location and the new location (try scrolling in Google Maps for an example of how this might look).  Support smooth scrolling when the user double clicks or clicks on a scroll button.

### Credits

Our map is from the Stanford Maps and Records Department.  The original Map was from:

```
http://www-facilities.stanford.edu/maps/download/TransportationMap.pdf
```