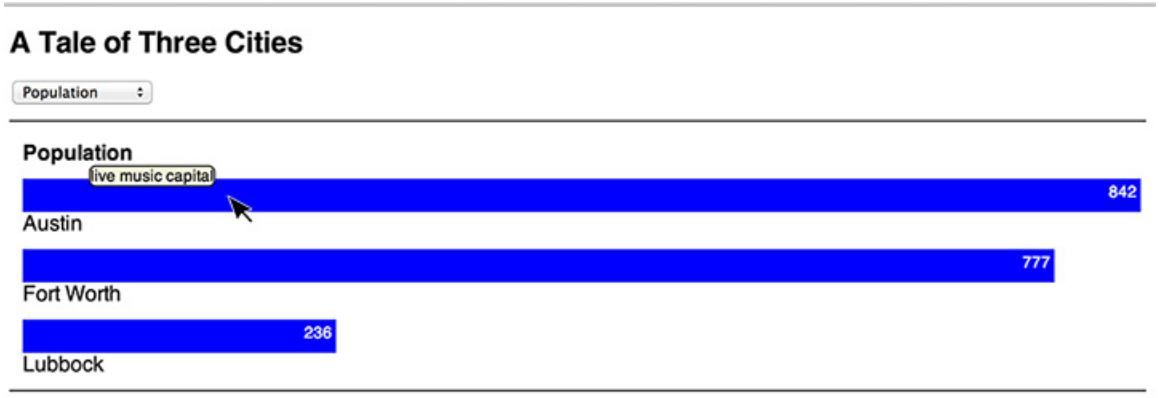
CodeActually
Dr. Cindy Royal
Texas State University
School of Journalism and Mass Communication

Simple Charting Exercise with HTML/CSS and JavaScript

With this exercise, you will learn to use techniques you know now in HTML/CSS/JavaScript to create an interactive chart. When you are finished you will have a chart that looks like this, with animated bars, dropdown to select different measures and tooltip on hover.



We'll be working with census data comparing cities in Texas, but you can feel free to substitute data of your own.

Setting Up a Basic Chart
Let's start with a basic bar chart. We will use a div to create a bar on the page. Create a basic html page with the proper structure and include a div with the id "bar1" and a class of "bars". We will be putting CSS in the head tag within the <style> section, but you could also use an external stylesheet if you needed to use these styles across multiple pages.

Include the style information for boxes to create a blue box that has a width of 300px and height 25px.

Then add a div below with the name of the first city in it. Use an id of "name1".

```
<!DOCTYPE html>
<html>
<head>
<style>
.bars {
```

```
background-color: blue;
width: 300px;
height: 25px;
margin: 10px 0 0 0;
}
</style>
</head>

<body>
<div id=”box1” class=”bars”></div>
<div id="name1">Austin</div>


</body>
</html>
```

Save this page as chart.html and open in a browser. You should see the blue bar labeled with “Austin”.

Let’s go ahead and create two more divs for two additional cities, so that the html in the <body> section looks like this:

```
<div id="bar1" class="bars"></div>
<div id="name1">Austin</div>

<div id="bar2" class="bars"></div>
<div id="name2">Fort Worth</div>

<div id="bar3" class="bars"></div>
<div id="name3">Lubbock</div>
```

Add Some CSS

If you open this page in a browser, you will see three blue bars, all of the same size, each labeled with a different city. But the fact that they are the same size means that we need to identify them separately by id to adjust just the width. Add the following to the style section to correspond to the ids for each div.

```
#bar1 {
width: 832px;
}

#bar2 {
```

```
width: 777px;
}

#bar3 {
width: 236px;
}
```

You have a bar chart with data! We could stop right here, publish this on the Web. You could add some additional styles to control the font choice, margins or headings. For now, let's just add a <body> style that sets the font to Helvetica and size to 16.

Your CSS should look like this:

```
body {
font-family: Helvetica;
font-size: 16px;
}

.bars {
background-color: blue;
height: 25px;
margin: 10px 0 0 0;
}                              d

#bar1 {
width: 842px;
}

#bar2 {
width: 777px;
}

#bar3 {
width: 236px;
}
```

We are using population data from the US Census (quickfacts.census.gov). Let's assume that a pixel represents 1000 people, although we can make this anything we want. You can modify this depending on the data with which you are working. You should add comment lines to your page that provide any information the user needs to understand the data.

Adding Numbers to the Bar

What's the next thing our chart needs? It would be good to have the actual numbers on the chart. Let's get them at the end of each bar. Inside each div, let's add a paragraph that holds the number, using the class "number". Your html should look like this now:

```
<div id="bar1" class="bars"><p class="number">300</p>
</div>
<div id="name1">Austin</div>

<div id="bar2" class="bars"><p class="number">600</p>
</div>
<div id="name2">Fort Worth</div>

<div id="bar3" class="bars"><p class="number">800</p>
</div>
<div id="name3">Lubbock</div>
```

Great. You have your data on the bar, but we need to style it now so that it shows up nicely and to the right. Let's add a style for the "number" class.

```
.number {
color: white;
text-align: right;
}
```

I think those numbers could use some padding and smaller sizing. Let's go with this for the number class and see what it looks like. Feel free to adjust to the desired size and placement you want for your chart.

```
.number {
color: white;
text-align: right;
padding: 3px;
font-size: 12px;
}
```

Pretty nice chart. We have three cities, representing population on separate bars. You can add any headings you want to the chart to help the user better understand what is happening. And, you can add any content below the chart. Let's just add an h1 that tells what the chart is about. Add this above the divs. Create an h1 style to be the size you wish.

```
<h1> A Tale of Three Cities</h1>
```

Adding Data with JSON

This chart is awesome, but it's kind of a pain to have to put the data right into the html and css. Let's put the data into a JSON section and then use JavaScript to pull it in where it is needed. JSON data is an object inside an array that holds the information we want to plot. We did this in the JavaScript introduction. Put the JSON data in a <script >section under the html but before the </body> tag. It is important to put this script below most of the html, because it needs to render after those elements.

```
var cities = [
{name: "Austin", population: 842},
{name: "Fort Worth", population: 777},
{name: "Lubbock", population: 236}
];
```

We also need to add the ids `id="bartext1", id="bartext2", id="bartext3"` to each paragraph that holds the population number. Add to each paragraph like this:

```
<div id="bar1" class="bars"><p id="bartext1"
class="number">300</p>
</div>
```

And, we should remove the original city names that we placed in the divs. These are no longer necessary.

```
<div id="name1"></div>
```

Now we will test the magic of the getElementById method. We'll do the first element in the JSON data, and then we will create a loop to read all the elements.

The getElementById method combined with the innerHTML property allows you to change what is in the html based on the id. The code below changes the innerHTML for "name1" and "bar1" to reflect the first element ([0]) in the JSON data. The second line adds the text to the "bartext1" id. And the third line uses the style.width method to adjust the size of the bar.

```
document.getElementById("name1").innerHTML= cities[0].name;
document.getElementById("bartext1").innerHTML=
cities[0].population;
document.getElementById("bar1").style.width =
cities[0].population + "px";
```

We could do this for all three bars, but it's more efficient and more fun to use a loop to read in the data to each item. Modify this section by adding the for loop statement and the accessor variable i.

```
for (var i = 0; i<cities.length; i++) {
document.getElementById("name" + (i+1)).innerHTML=
cities[i].name;
document.getElementById("bartext" + (i+1)).innerHTML=
cities[i].population;
document.getElementById("bar" + (i+1)).style.width =
cities[i].population + "px";
};
```

Since we are using the JSON data to establish the width, we no longer need the styles for each box. You can delete those styles, unless you want to do something unique to each bar later, like give each a separate color.

Now you can modify the data in the JSON section to anything you want and be able to use this code to chart anything!

Dropdown

Now, let's add some additional data about the cities so we can use a dropdown to change the measure we are viewing. Chances are, you'll want to display more than one set of data per item.

```
<form name="form1">
<select name="sel" onChange="selMeasure()">
  <option value="Population">Population</option>
  <option value="Households">Households</option>
  <option value="Units">Housing Units</option>
</select>
</form>
```

Notice that the select statement uses the onChange event handler with the selMeasure() function. We will write selMeasure() below. Each items has a value that corresponds to the text for each option.

Writing a Function

Let's put the main population items in a function. We need to have the population figures show when the page loads and to also show when "Population" is selected from the dropdown. Rather than writing it twice, we'll write it once in a function, then use the function to call it the subsequent times.

```
function writeData() {
    for (var i = 0; i<cities.length; i++) {
    document.getElementById("name" + (i+1)).innerHTML=
cities[i].name;
    document.getElementById("bartext" + (i+1)).innerHTML=
cities[i].population;
    document.getElementById("bar" + (i+1)).style.width =
cities[i].population + "px";
    };
};

writeData(); // this is where we call the function the
first time.
```

Additional Data

Change your JSON data to include a couple other measures, like units and households, all from the census data. You can also include the "tip" which will be what shows on the bar when we hover over it. We'll use JQuery to add that.

```
var cities = [
{name: "Austin", population: 842, units: 354, households:
322, tip: "live music capital"},
{name: "Fort Worth", population: 777, units: 291,
households: 257, tip: "near Dallas"},
{name: "Lubbock", population: 236, units: 95, households:
86, tip: "home of Texas Tech"}
];
```

Changing the Measure

We will write a function that will be called in the html form for the dropdown. This function will read the selection in the dropdown. To make things more efficient, let's create a variable as an array to store the measures we will be using. These must match the values in your form.

```
var measures = ["Population", "Housing Units", "Households"
];
```

The function below reads the current value in the form and compares it to the items in the array. Then it generates the chart. Notice that each item now has an additional line to display a heading to identify the measure. This changes depending on the variable "m" which is equal to the item selected in the form.

```
document.getElementById("cat").innerHTML = "<strong>" + m +
"</strong>";
```

selMeasure() Function
This function switches from the JSON into the chart, based on which measure was selected. Note that you only have to write the name once, since that is the same for all three measures.

```
function selMeasure() {
var m = document.form1.sel.value;
if (m == measures[0])
{
document.getElementById("cat").innerHTML = "<strong>" + m +
"</strong>";
writeData();

} // end if
else if (m ==  measures[1]) {
document.getElementById("cat").innerHTML = "<strong>" + m +
"</strong>";
for (var i = 0; i<3; i++) {
    document.getElementById("bartext" + (i+1)).innerHTML=
cities[i].households;
    document.getElementById("bar" + (i+1)).style.width =
cities[i].households + "px";

    };
}
else if (m ==  measures[2]) {
document.getElementById("cat").innerHTML = "<strong>" + m +
"</strong>";
for (var i = 0; i<3; i++) {
    document.getElementById("bartext" + (i+1)).innerHTML=
cities[i].units;
    document.getElementById("bar" + (i+1)).style.width =
cities[i].units + "px";

    };

}
}
```

Adding a Border Around the Chart
Include a div id=”main” around the chart below the dropdown form. Then include a style to create some border lines at the top and bottom.

```
#main
{
border-top: 1px solid black;
border-bottom: 1px solid black;
overflow: auto;
padding: 10px;
margin-top: 10px;

}
```

Animation

We can add a simple animated effect to jazz up the chart a little. Let's add a CSS3 transition to draw the bars.

Add this to the .bars class. When you switch measures with the dropown, you will see a drawing effect on the width. For now, you must use the prefix for each browser until the main specification is supported. For browsers that don't support this feature, the user will not see the animated effect.

```
.bars {
background-color: blue;
height: 25px;
margin: 10px 0 0 0;
-webkit-transition:width 1s ease-in;
-moz-transition:width 1s ease-in;
-o-transition:width 1s ease-in;
transition:width 1s ease-in;
}
```

Tooltip

We are going to use a little JQuery to add a tooltip to each bar. We will cover JQuery in more detail in another exercise, but JQuery is a library that provides a variety of JavaScript functionality. The JQuery UI takes that a step further by providing specific User Interface functionality. You can learn more here:
JQuery - jquery.com
JQueryUI – jqueryui.com

To add the tooltip, we just have to add a short function and modify the loop to write the tip. Our tips are included in the JSON data.

First, we need to include the link to the JQuery UI library. You can link to it online or download it to your own server. Put this in the head of the document.

```
<script
src="http://cdn.jquerytools.org/1.2.7/full/jquery.to
ols.min.js"></script>
```

Next, we need to include the Tooltip code in a function. Put this within the <script> tag at the bottom of the page. The first line is the way that all JQuery functions are started.

```
$( document ).ready(function() {
   $("[title]").tooltip();
    });
```

Finally, add this line to the writeData() function. Since the tooltip is the same for each measure, we only have to establish it once.

```
document.getElementById("bar" + (i+1)).title=
cities[i].tip;
```

We need to do some styling with the tooltip as well. Play around with the CSS to get it to look the way you want.

```
.tooltip {

    display:none;
    background: beige;
    font-size:14px;
    height:25px;
    border-radius: 5px;
    padding-top: 2px;
    padding-left: 5px;
    padding-right: 5px;
    padding-bottom: 0px;
    color: black;
    border: 1px solid black;

  }
```

If you don't like the position of the tooltip, you can add the offset to move around the left and top position, like this.:

```
$("[title]").tooltip({offset: [10, -30]});
```

Using Percentages for the Widths of the Bars

In this age of mobile, we really need to consider how the chart would look on smaller devices. Let's try to make this responsive by using percentages instead of

pixels to draw the chart. By making the chart's features draw as a percentage of the window, that should take care of most responsive issues.

Using percentages can also improve the presentation of data that has either been too small or too large to be meaningful by presenting with pixels. The values will remain the same, but the bars will be drawn relative to one another instead of with a fixed pixel size.

To do this, we'll have to use a little math. We need to figure out the largest number that has been input, then find out the percentage that the others represent.

For example:

Population		
Austin	826	100%
Fort Worth	777	94%
Lubbock	236	29%

The city in this list with the largest population is Austin. Dividing Austin's population by itself results in 1 or 100%. The others are shown as a percentage of Austin's population. We will use code to identify the maximum value and calculate the percentages. The largest will draw at 100% of the window and the others will be drawn relatively based on the percentage.

First, we will read all values from the JSON data into an array. Be sure to put these elements before the initial writeData() function.

```
var myPop = [cities[0].population, cities[1].population,
cities[2].population];
```

Then we create a variable to store the maximum number in the array using the Math.max.apply method.

```
var popMax = Math.max.apply(Math,myPop);
```

Now in the loop when you create the width, you will divide the amount by the max. Replace the statement that is currently there with this one.

```
document.getElementById("bar" + (i+1)).style.width =
((cities[i].population/popMax)*100) + '%';
```

You can do the same for the other two measures in the JSON data. Read the numbers into an array, find the max and adjust the calculation.

```
var myHouseholds = [cities[0].households,
cities[1].households, cities[2].households];
var householdsMax = Math.max.apply(Math,myHouseholds);
```

```
var myUnits = [cities[0].units, cities[1].units,
cities[2].units];
var unitsMax = Math.max.apply(Math,myUnits);
```

then in the respective loops…

```
document.getElementById("bar" + (i+1)).style.width =
((cities[i].households/householdsMax)*100) + '%';

document.getElementById("bar" + (i+1)).style.width =
((cities[i].units/unitsMax)*100) + '%';
```

You may have to control other elements of the chart via media queries, which we covered in html/css. Now, when you size the window, the chart adjusts.

Let's take a look at what we have done. We have an html page that reads information from JSON-stuctured data into a chart. The chart demonstrates animation and interactivity by changing the dropdown selection and hovering for the tooltip. It is done completely with code, no plug-ins necessary. And now it is responsive to the size of the screen on which it is displayed. This one page can be uploaded to the Web and anyone can see it in a browser. You can use this as a standalone page or take this code into an existing page. Just make sure you structure the html, css and script areas properly.

This technique works well for a simple bar chart. We will be looking at other types of tools to understand what is available to make other types of charts.