Lab 3: Client Side JS

The goal for this lab is to wire up JavaScript into the index.html and calendar.html pages. You'll learn how to include external scripts into your webpage and how to read documentation.

Deliverable

1. 3 webpages: a Home page, a Grades page, a Calendar page all uploaded to the ECC servers.
2. Your lab should be accessible at the following paths:
    a. http://students.engr.scu.edu/~aelahi/coen-161/lab-3/index.html
    b. http://students.engr.scu.edu/~aelahi/coen-161/lab-3/calendar.html
    c. http://students.engr.scu.edu/~aelahi/coen-161/lab-3/grades.html

index.html

By the end of the lab, your page should look similar to the screenshot below.

Updating the CSS

Before getting started with the JavaScript portion, write the CSS needed to make index.html look like the above screenshot.

1. The background color of each of those charts is #eee
2. Each chart is actually an HTML <canvas> element. The canvas must be the only child of its parent for the charting library to work so you'll need a structure like this for each chart.

```
<figure>
  <canvas></canvas>
</figure>
```

3. Extra Hints
    a. Just like an image, make the canvas have a size of 100% height and width of its parent
    b. The wrapping elements for the two smaller charts (ex. the figure in the above HTML) should have a height and width of 200px. The larger one shouldn't have any size restriction
    c. The left column should take up 50% of the space of its parent. The right column should take up 30. Neither column should grow.
4. The rest is up to you

Integrating the chart.js library

Now that the structure and style of our page is all set up, we'll use JavaScript to insert dynamic content. These charts are going to be very interactive but we're not actually going to do much work to set that up. One of the cool things about JavaScript is that it's really easy to include other people's code into our web pages.

For any line that says Question N, please type your answer into the top of your index.js page as a comment.

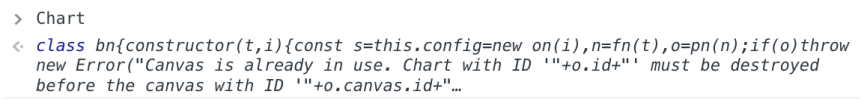
Updating index.html

1. In your index.html page, include the chart.js from this URL: https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js
    a. You'll need to use a script tag to do so. Make sure this script **does not block the browser while parsing the JavaScript**

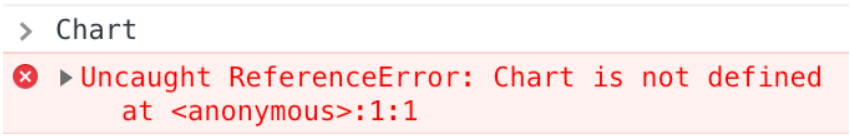
i. Question 1 - What attribute did you use to not block the browser while parsing JavaScript?

b. Open the developer console and in the Console tab, type the word `Chart` (case sensitive)

i. If you've added the script, it should look like this:



ii. If you haven't added the script, it should look like this



2. Add a unique id attribute to each of the <canvas> elements. We'll use this id to uniquely select each of our charts from JavaScript.

Setting up index.js

1. Create a new file called index.js. Include it in your HTML just like you included the Chart.js library.
    a. Inside, use the console.log function to print out what the Chart object is. This is the same thing we did in the last step, just in script for now.
        i. Refresh the page a couple times. Sometimes you should see the Chart is not defined error message and others you should see the Chart object printed to the console.
            1. If you never see the error message, try using Cmd/Ctrl + Shift + R.
            2. If you don't see the error message, then your answer to 1.a.i *may* be incorrect
        ii. Open the network tab in the Developer Console. Look at the size, and waterfall columns for the index.js and chart.min.js entries.
            1. Compare and contrast the values of these columns when you see the error message and when you don't.
            2. Question 2 - What do you think is happening when you see the error message?
    b. To make sure we never see the error message, place the console.log() in a call to setInterval() as shown below

```
let interval = setInterval(function() {
  if (!Chart){
    console.log(Chart)
  } else {
    clearInterval(interval)
  }
}, 100);
```

   i. Question 3 - What does the setInterval function do?
   ii. Question 4 - What would happen if the clearInterval call didn't exist?
   iii. Question 5 - What does the 100 in the above code snippet signify?

c. Replace the anonymous function inside of setInterval with a named version of the function (`main` in this handout)
d. Create 3 empty functions: createGradeByWeekChart(), createPointsInClassChart(), and createPointsByCategoryChart(). Call each of these functions from `main`.

Building the charts

Chart.js uses the HTML canvas element to paint really nice dynamic charts onto the screen using JavaScript. We usually don't need that level of precision when laying out webpages. For static images, we'd just use an img tag. But because we want the interactivity in these charts, we'll be using chart.js and the canvas element. We won't study the canvas element too much though.

This section guides you through the createPointsByCategoryChart function, everything should happen inside that function.

1. Create a variable to hold a reference to the HTMLChartElement
   a. `const canvas = document.querySelector(/* fill this in */);`
2. To instantiate a chart, Chart.js has a constructor with two parameters (1) the context to paint into (2) options for the chart.
   a. Because there's so many options for how to create charts, Chart.js takes an options object. If a property of that object is not given, then Chart.js would use a default value
   b. Create an empty object called options
   c. `const chart = new Chart(canvas.getContext("2d"), options);`
3. Fill in the options object using the following code block. There are some questions, but you don't have to answer them in the lab handout. Thinking about them should help you finish this page.

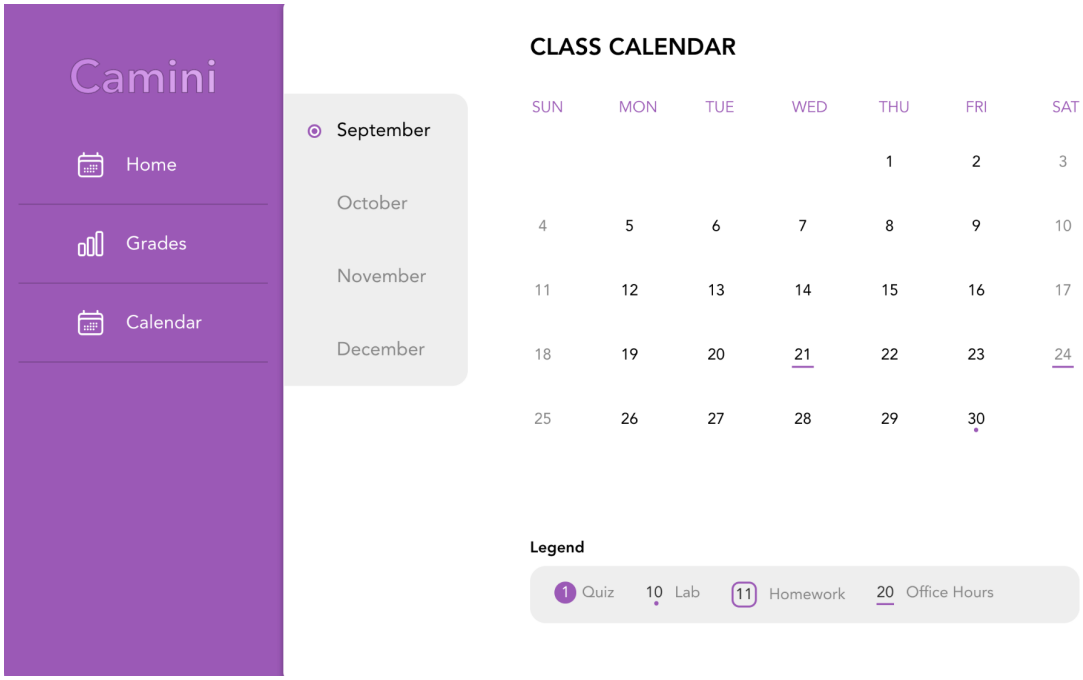
```
{
    // the type of chart we want to use
    type: "bar",
    data: {
      // which axis shows these labels?
      labels: ["Quizzes", "Labs", "Theory", "Practice"],
      datasets: [
        // is this the top or bottom data set?
        // what happens if there's only 1 dataset?
        {
          label: "Earned",
          data: [65, 59, 80, 81, 56, 55, 40],
          backgroundColor: "rgba(155, 89, 182, 1.0)",
        },
        {
          label: "Missed",
          data: [65, 59, 80, 81, 56, 55, 40],
          backgroundColor: "rgba(155, 89, 182, 0.5)",
        },

        {
          label: "Ungraded",
          data: [65, 59, 80, 81, 56, 55, 40],
          backgroundColor: "#eee",
        },
      ],
    },
    options: {
      scales: {
        y: {
          stacked: true,
          grid: {
            display: false,
          },
        },
      },
      x: {
        stacked: true,
        grid: {
          display: false,
        },
      },
      aspectRatio: 1,
      plugins: {
        title: {
          display: true,
          text: "Points by Category",
        },
```

```
      legend: {
        display: false,
      },
    },
  },
}
```

4. After you've gotten 1 chart down, use the [chart.js documentation] to build the other two charts and complete this page.

calendar.html



1. In calendar.html, include the given date-fns.js file. This has utilities to help you handle dates since they're notoriously annoying.
2. Create a new calendar.js file that gets included (just like above) in calendar.html.
3. Paste the [events code block] into the file at the top of calendar.js.
4. Initially populate the days of the month using functions from the date-fns file.
    a. Make sure to use an event listener on the window's load function.
5. Whenever someone clicks one of the radio buttons for a month
    a. make sure the days for that month reflect what's currently shown in the calendar
    b. any events on that date should reflect through the styling of the calendar

Tips

1. Use a data-attribute for your months. There's only 4 of them, you can hard code that into your HTML
2. When you update the calendar, have these distinct steps
    a. Reset the calendar
    b. Figure out the boundaries of the calendar. Which table cells do you skip and which do you fill?
        i. The DateFns object has functions like startOfMonth() which can help you figure that out. **Read** that file
    c. Start from the first cell you need to fill and create a span element with the date
    d. Populate the events by iterating through that month's entry for the events object

Grading

The following table shows the point breakdown per section. Each day (including weekends) past the due date for the lab will be -5 points. You must submit your lab on Camino as well as upload it to your website.

Criteria	Points
index.html - Updated CSS for index.html	10
index.html - questions (submitted in index.js code as a comment)	20
index.html - charts dynamically generated using Chart.js library	20
index.html - large chart resizes dynamically	10
calendar.html - radio buttons have the correct styling applied when the value changes	15
calendar.html - calendar days update to reflect the current month	15
calendar.html - calendar dates are correctly labeled by the legend	10

Events Code Block

Paste this into your calendar.js file when you get to that point.

```
const EVENTS = {
  // september
  8: {
      21: {
      type: "office-hours",
      },
      22: {
      type: "homework",
      description: "Homework 1 Due",
      },
      24: {
      type: "office-hours",
      },
      28: {
      type: "homework",
      description: "Homework 2 Due",
      },
      30: {
      type: "lab",
      description: "Lab 1 Due",
      },
  },
  // october
  9: {
      3: {
      type: "office-hours",
      },
      4: {
      type: "homework",
      description: "Homework 3 Due",
      },
      5: {
      type: "lab",
      description: "Lab 2 Due",
      },
      10: {
      type: "office-hours",
      },
      13: {
      type: "quiz",
      description: "Quiz 1",
      },
      16: {
      type: "homework",
      description: "Homework 4 due",
      },
      17: {
      type: "office-hours",
```

```
    },
    19: {
    type: "lab",
    description: "Lab 3 Due",
    },
    24: {
    type: "office-hours",
    },
    28: {
    type: "homework",
    description: "Homework 5 due",
    },
},
// november
10: {
    6: {
    type: "office-hours",
    },
    7: {
    type: "homework",
    description: "Homework 6 due",
    },
    9: {
    type: "lab",
    description: "Lab 4 Due",
    },
    13: {
    type: "office-hours",
    },
    16: {
    type: "lab",
    description: "Lab 5 Due",
    },
    21: {
    type: "homework",
    description: "Homework 7 due",
    },
    22: {
    type: "quiz",
    description: "Quiz 2",
    },
    30: {
    type: "office-hours",
    },
},
// december
11: {
    1: {
```

```
      type: "lab",
      description: "Lab 6 Due",
      },
      2: {
      type: "quiz",
      description: "Quiz 3",
      },
      4: {
      type: "homework",
      description: "Homework 8 due",
      },
  },
};
```