Employee Payroll Tracker

**Employee Payroll Tracker**

This week's Challenge requires you to modify starter code to create an application that enables a payroll manager to view and manage employee payroll data. This app will run in the browser and will feature dynamically updated HTML and CSS powered by JavaScript code that you write. It will have a clean and polished, responsive user interface that adapts to multiple screen sizes.

**User Story**

AS A payroll manager

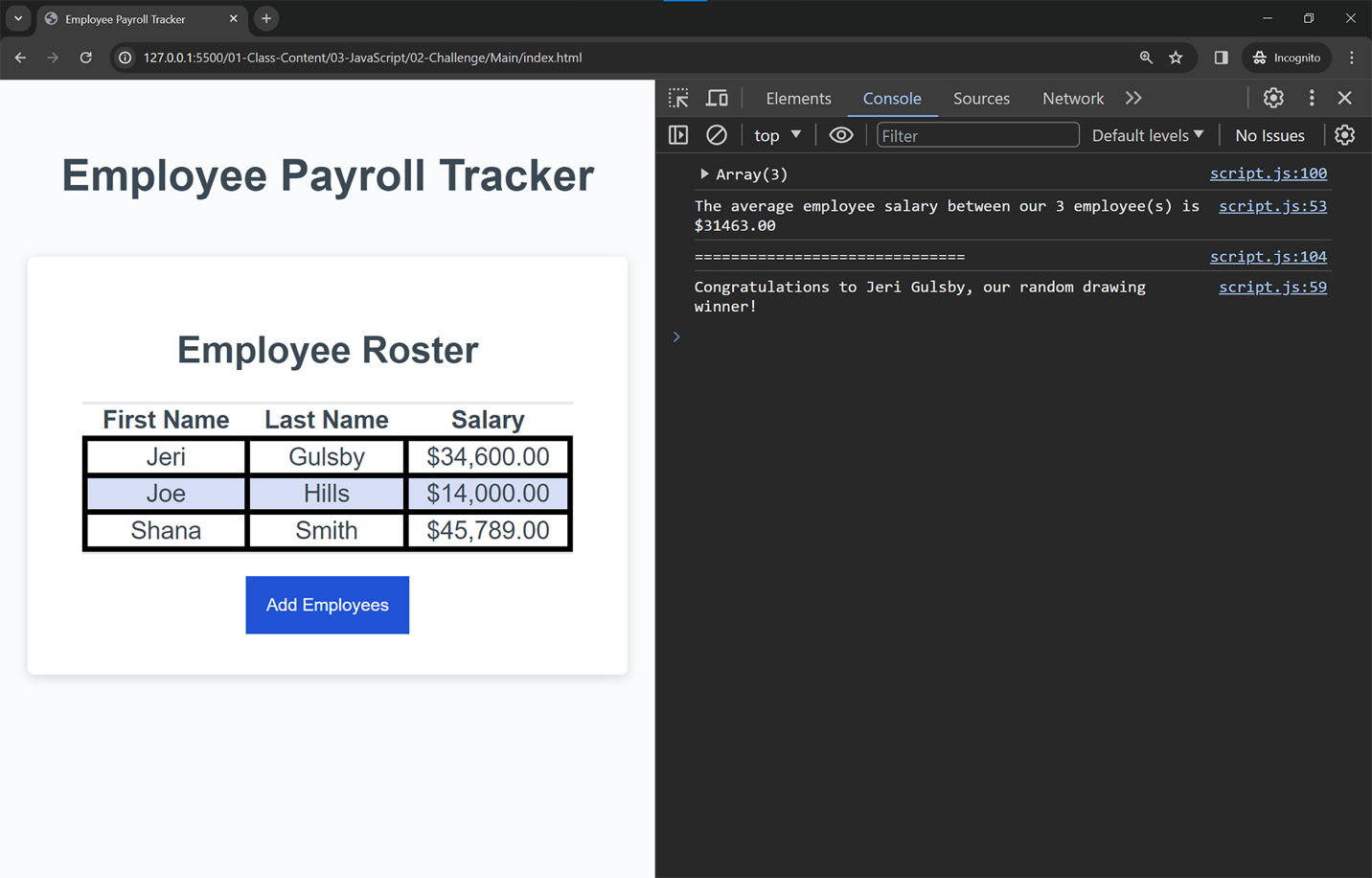
I WANT AN employee payroll tracker

SO THAT I can see my employees' payroll data and properly budget for the company

**Acceptance Criteria**

GIVEN an employee payroll tracker

* WHEN I click the "Add employee" **button**  
  THEN I am presented with a series of prompts asking for first name, last name, and salary
* WHEN I finish adding an **employee**  
  THEN I am prompted to continue or cancel
* WHEN I choose to **continue**  
  THEN I am prompted to add a new employee
* WHEN I choose to **cancel**  
  THEN my employee data is displayed on the page sorted alphabetically by last name, and the console shows computed and aggregated data



You will be responsible for filling out the following functions:

* **collectEmployees**: This function will allow a user to add multiple employees to display on the page. The user will need to enter the first name, last name, and salary of each employee, then have the option to keep adding employees until they choose to stop. A **while loop** will be needed here ([MDN Web Docs on while loopsLinks to an external site.](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while)) The salary will need to be entered as a number, otherwise it should default to $0. The **isNaN** function can help with this: ([MDN Web Docs on isNaNLinks to an external site.](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN)) This function should return an array of objects, like the following example. Reference the [MDN Web Docs on returnLinks to an external site.](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/return):

[

{

firstName:"John",

lastName:"Smith",

salary:12345

},

{

firstName:"Jane",

lastName:"Doe",

salary:54321

}

]

* **displayAverageSalary**: This function will take in the generated array of employees and log the average salary and number of employees to the console. You should use a template literal string for this task.
* **getRandomEmployee**: This function will take in the generated array of employees, randomly select one employee, and use a template literal to log their full name to the console. The built in Math object can help with random number generation: ([MDN Web Docs on Math.randomLinks to an external site.](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random))

The provided starter code includes the **displayEmployees** and **trackEmployeeData** functions. These functions are complete and working. You do not have to modify any code for the following functions:

* displayEmployees: This function will take in an array of employees and render each employee to an HTML table.
* trackEmployeeData: This function will execute when the "Add Employees" button is clicked. It will take the array generated in your collectEmployees function, sort the employees by last name, and place them on a table on the page using the provided displayEmployees function. Additionally, the function will execute the displayAverageSalary function to log the average employee salary to the console, and execute the getRandomEmployee function to log a random employees information to the console.

**Grading Requirements**

This Challenge is graded based on the following criteria:

**Function to Collect Employees 40%**

* The collectEmployees() function must implement the following:
  + Create a new employee object by collecting first name, last name, and salary using prompt(). (20 points)
  + Create multiple employee objects by collecting first name, last name, and salary for each employee using prompt() and confirm(). (20 points)

**Function to Display Average Salary 30%**

* The displayAverageSalary() function must implement the following:
  + Calculate the average salary and log "The average employee salary between our <numberOfEmployees> employee(s) is $<averageSalaryWithTwoDecimals>" when given salaries with no decimals. (15 points)
  + Calculate the average salary and log "The average employee salary between our <numberOfEmployees> employee(s) is $<averageSalaryWithTwoDecimals>" when given salaries with decimals. (15 points)

**Function to Choose a Random Drawing Winner 30%**

* The getRandomEmployee() function must implement the following:
  + Choose an employee at random and log "Congratulations to <employeeFirstName> <employeeLastName>, our random drawing winner!". (15 points)
  + Include a random selection method that allows for all employees to be chosen in the drawing. (15 points)