

# Module 3 Challenge

[Start Assignment](#)

- Due Monday by 11:59pm
- Points 100
- Submitting a text entry box or a website url

This week's Challenge requires you to create an application that a payroll manager to view and manage employee payroll data.

## JavaScript Challenge: 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

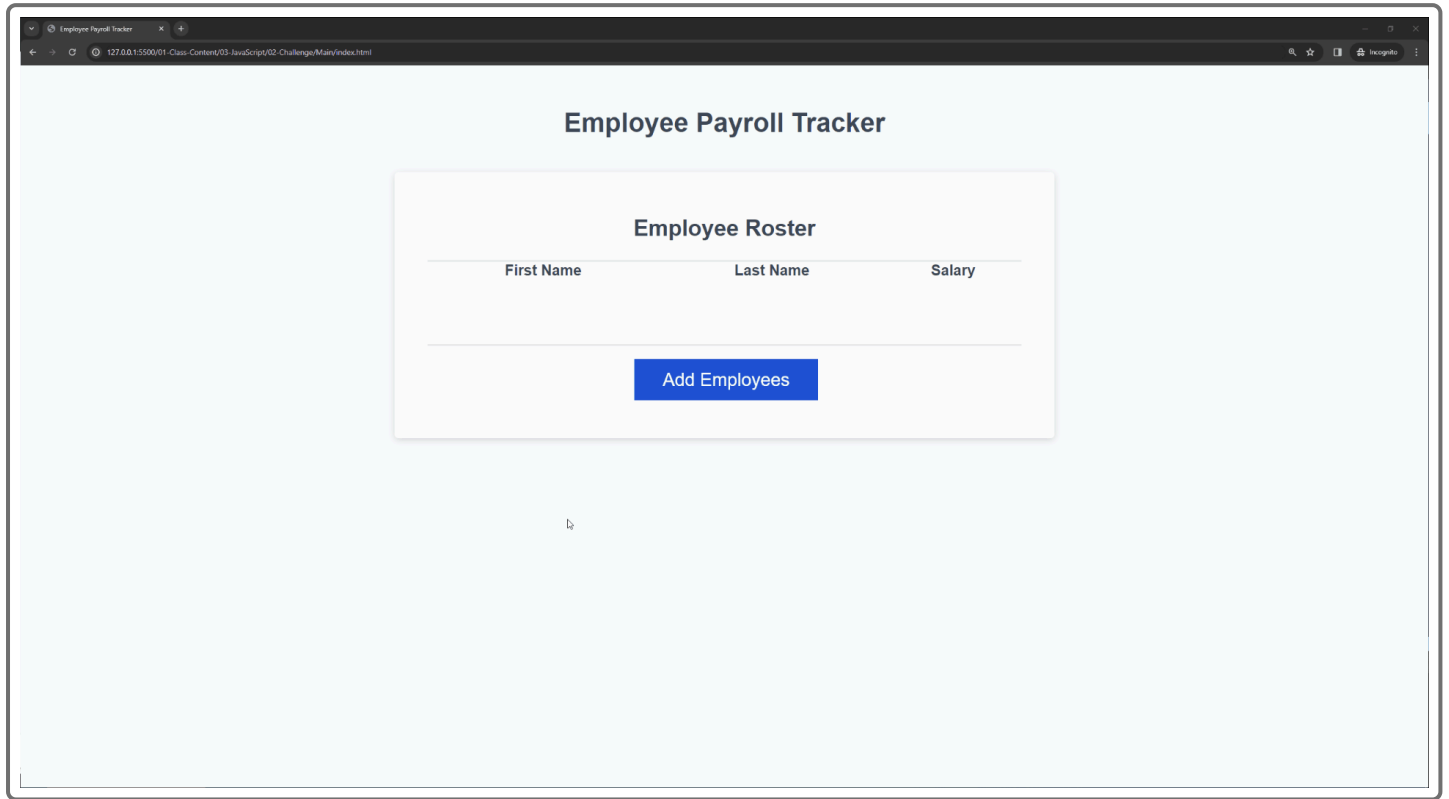
AS A payroll manager  
I WANT AN employee payroll tracker  
SO THAT I can see my employees' payroll data and properly budget for the company

## 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 con

## Mock-Up

The following images show the web application's appearance and functionality:



**Employee Payroll Tracker**

**Employee Roster**

First Name	Last Name	Salary
Jeri	Gulsby	\$34,600.00
Joe	Hills	\$14,000.00
Shana	Smith	\$45,789.00

[Add Employees](#)

Array(3) [script.js:100](#)

The average employee salary between our 3 employee(s) is [script.js:53](#)  
\$31463.00

===== [script.js:104](#)

Congratulations to Jeri Gulsby, our random drawing winner! [script.js:59](#)

## How to Complete the Challenge

Follow these steps to complete the challenge:

### IMPORTANT

Make sure to clone the starter code repository and make your own repository with the starter code. Do not fork the starter code repository!

1. Clone the [starter code](https://github.com/coding-boot-camp/curly-potato) [🔗 \(https://github.com/coding-boot-camp/curly-potato\)](https://github.com/coding-boot-camp/curly-potato).
2. Modify the code to meet the Acceptance Criteria.
3. Ensure that your work meets the full list of grading requirements below.
4. Reach out to your classmates and instructional support team for help if you need it.
5. Follow the submission instructions.

### NOTE

After this week, the preceding steps will not be included in Challenge instructions in order to simulate a typical work experience where the developer determines how to solve an issue on their own.

## Getting Started

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` loops](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while) [↗](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while) (<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 `isNaN`](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN) [↗](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN) ([https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/isNaN](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 `return`](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/return) [↗](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/return) (<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.random`](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random) [↗](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random) ([https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Math/random](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

### NOTE

If a Challenge assignment submission is marked as "0", it is considered incomplete and will not count towards your graduation requirements. Examples of incomplete submissions include the following:

- A repository that has no code
- A repository that includes a unique name but nothing else
- A repository that includes only a README file but nothing else
- A repository that only includes starter code

This Challenge is graded based on the following criteria:

### Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria.

### Deployment: 32%

- Application deployed at live URL.
- Application loads with no errors.
- Application GitHub URL submitted.
- GitHub repository that contains application code.

### Application Quality: 15%

- Application user experience is intuitive and easy to navigate.
- Application user interface style is clean and polished.
- Application resembles the mock-up functionality provided in the Challenge instructions.

### Repository Quality: 13%

- Repository has a unique name.
- Repository follows best practices for file structure and naming conventions.

- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
  - Repository contains multiple descriptive commit messages.
  - Repository contains quality readme file with description, screenshot, and link to deployed application.
- 

## How to Submit the Challenge

You are required to submit BOTH of the following for review:

- The URL of the deployed application.
- The URL of the GitHub repository. Give the repository a unique name and include a README describing the project.

### NOTE


You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next Module.

Comments are disabled for graded submissions in BootCamp Spot. If you have questions about your feedback, please notify your instructional staff or the Student Success Advisor. If you would like to resubmit your work for an improved grade, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

### IMPORTANT

**It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo.** This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a Challenge or any aspect of the curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.
2. AskBCS Learning Assistants exists in your class Slack application.
3. Office hours facilitated by your instructional staff before and after each class session.
4. **[Tutoring Guidelines](https://docs.google.com/document/d/1hTIdEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing)**  —  [\(https://docs.google.com/document/d/1hTIdEfWhX21B\\_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing\)](https://docs.google.com/document/d/1hTIdEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing) — schedule a session in the "Tutor Sessions" section of Bootcampspot - Canvas.
5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section

of your BCS application.

© 2024 edX Boot Camps LLC