ASSIGNMENT 10

Version 21/FA

MAPPINGS:

The following course objectives and/or outcomes are measured in this assignment:

COURSE OBJECTIVES

- 10E: Use your browser to review the response that's returned from a request to a web service.
- 10F: Use the Fetch API to make Ajax requests that update a web page without reloading it.

COURSE OUTCOMES

- 1. Use JavaScript to create interactive web applications.
- 2. Use JavaScript to interact with application programming interfaces (APIs).
- 3. Use Ajax to dynamically update web pages.
- 4. Write clean, consistent code.

OVERVIEW:

Assignment 10 is partially a punch-and-run (i.e., you type code provided by the instructor) and partially a problem-solving exercise. There is an opportunity for extra credit in Assignment 10. The extra credit is defined in Appendix B (page 7).

When inputting the provided code, pay close attention to details.

Continues . . .

GRADING:

Task	Points	Criteria
Part 1		
1.1 Setup		
There are no points in this section of the assignment.		
1.2 Test the Application		
There are no points in this section of the assignment.		
1.3 Linking to External JavaScript Files		
1	4	Pass/Fail
1.4 Punch and Run		
2	2	Pass/Fail
3	2	Pass/Fail
4	20	Pass/Fail
1.5 Enhancement: Format Currency Using `accounting.js`		
3	10	- Subtract10 points if student did not use account.js to format currency. - If student did use accounting.js, deduct points based on any errors that may be present (instructor discretion).
1.6 Comments		
2	10	- 10 points: if student has commented at least ½ of the code base. Comments must demonstrate the student understands what the code base is doing. Superfluous comments (e.g., "add numbers") may be penalized. - Deduct points for meaningless comments, which do not provide any insight into the student's understanding of the code.
Total	**48**	

Penalties
Deduct 50% from entire assignment for the use of the `var` keyword in variable declarations.
Deduct 30% from the entire assignment is the solution does not load and/or if errors appear in the console that are not generated by explicit methods of the console object (in other words, errors that have not been troubleshot and resolved by the student prior to submission).
Deduct 60% from the entire assignment if JavaScript is inline/embedded instead of external.
Deduct a maximum of 22.5% for code that does not comply with the course *Style Guide* and/or which is messy/unorganized, uncommented, or missing semicolons.
Deduct up to 10% per file for any file moved from its original location within the starter files' file structure.

Continues…

TASK:

1.1 SETUP

1. Review the course *JavaScript Style Guide* before starting this assignment. Part of the assignment will be graded on your adherence to the *Style Guide*.
2. Download **assignment10_starter.zip** from the *Module 10: Assignment* drop box in Canvas. The file is located beneath the heading *Assignment Resources*.
3. Extract **assignment10_starter.zip.** The file contains two files and two subfolders:
    a. a single HTML document named *index.html*.
    b. a subfolder named *css*, which contains a single file named *style.css*
    c. a subfolder named *js*, which contains two JavaScript files:
        i. *accounting.min.js*
        ii. *script.js*
    d. a subfolder named *img*, which contains a single file named *ajax-loader.gif*

1.2 TEST THE APPLICATION

1. Open **index.html** in your browser. The rendered page should look what's shown in Figure 1 (below).

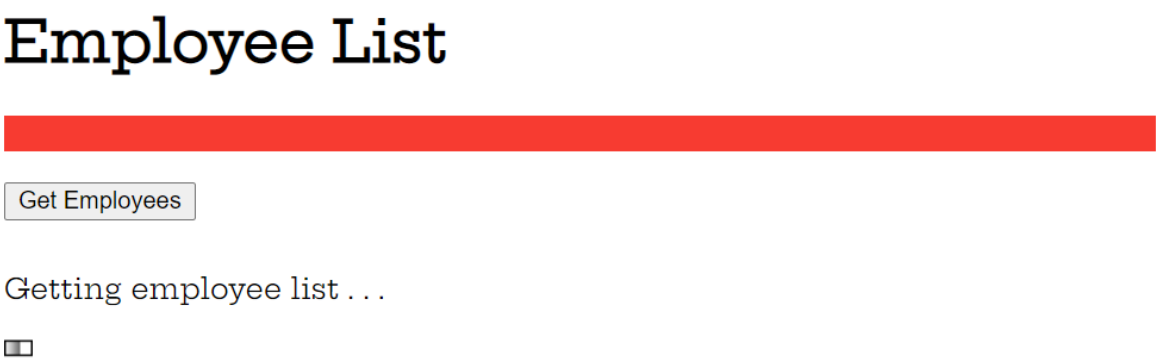

Figure 1

Continues . . .

1.3 Linking External JavaScript Files

1. Open **index.html** and link **accounting.min.js** and **script.js (4 points)**.

1.4 Punch and Run

1. Open **js/script.js**
2. Add the **Standard Opening Comment** to the top of the script **(2 points**).
3. Add "use strict" beneath the **Standard Opening Comment**. **(2 points)**.
4. Beneath "use strict", type the code provided in **Appendix A** (on page 6) **exactly as shown** into **script.js (20 points)**.
    a. This code has also been linked as a large PNG image to *Assignment 10* as *Image of Code for script.js*, which you can find beneath the *Assignment Resources* heading.
    b. No instructor assistance will be provided for task 1.4 – a working solution is in the code, you must pay close attention to the details in the code when typing the code.

1.5 Enhancement: Format Currency Using `accounting.js`

1. Review the documentation for the *accounting.js* library: http://openexchangerates.github.io/accounting.js/
2. Open **js/script.js**
3. Use accounting.js to format `employee_salary` as follows **(10 points)**:
    a. Symbol: `$`
    b. Decimal: `.`
    c. Thousand: `,`
    d. Precision: `2`
    e. Format: `%s%v`

1.6 Comments

1. Open **js/script.js**
2. Review the code; walk through the code. Then, add meaningful comments to the code.

Continues . . .

1.6 CLEANUP

1. Test your application. If necessary, use the browser's developer tools to troubleshoot any errors you may have.
2. Comment and clean up your code:
    a. Make sure to document what functions and blocks of code do.
    b. Ensure consistent alignment, spacing, and carriage returns.
    c. Make sure to remove non-used code; comments should describe your code only.

SUBMISSION

When complete, create a single ZIP file containing your solution for this assignment. The ZIP file should contain all files included in the original starter code or added as part of this assignment.

Attach and upload the ZIP file to Assignment 10 and submit.

NOTE: Canvas is configured to only accept ZIP files, DOC files, and DOCX files; it will not accept ZIPx, 7ZIP, pZip, RAR, etc.

End Assignment.

APPENDIX A

```
$(document).ready(() => {
    const baseUrl = 'https://www.mccinfo.net/epsample/employees';
    const employeeList = document.getElementById('employeeList');
    const employeeInfo = document.getElementById('employeeInfo');

    $('#errorHolder').hide();
    $('#loadingHolder').hide();
    $('#employeeList').hide();
    $('#getEmployees').click((evt)=> {
        $(evt.target).parent().slideUp();
        $('#loadingHolder').slideDown();
        fetch(baseUrl)
        .then ( response => response.json() )
            .then ( (employees) => {
                for(let employee of employees) {
                    let div = document.createElement('div');
                    let a = document.createElement('a');
                        a.href='#';
                        a.id = employee.id;
                        a.innerHTML = `${employee.first_name} ${employee.last_name}`;
                        a.addEventListener('click', onEmployeeClicked);
                        div.appendChild(a);
                        employeeList.appendChild(div);
                };
                $('#loadingHolder').fadeOut(400, ()=> {
                    $('#employeeList').fadeIn();
                });

            })
        });

        const onEmployeeClicked = (e) => {
            $('#errorHolder').hide();
            $('#errorHolder').html('');
            employeeInfo.innerHTML = '';
            fetch(`${baseUrl}\/${e.currentTarget.id}`)
            .then( response => response.json() )
            .then( (employee) => {
                let div = document.createElement('div');

                let img = document.createElement('img');
                    img.src = employee.image_filename;
                    img.alt = `${employee.first_name} ${employee.last_name}`;

                let h1 = document.createElement('h1');
                    h1.innerText = `${employee.first_name}  ${employee.last_name}`;

                let divDepartment = document.createElement('div');
                    divDepartment.innerText = `Department: ${employee.department.name}`;

                let divSalary = document.createElement('div');
                    divSalary.innerText = `Annual Salary: ${employee.annual_salary}`;

                let divHireDate = document.createElement('div');
                    divHireDate.innerText = `Hire Date: ${employee.hire_date}`;

                    div.appendChild(img);
                    div.appendChild(h1);
                    div.appendChild(divDepartment);
                    div.appendChild(divSalary);
                    div.appendChild(divHireDate);

                    employeeInfo.appendChild(div);
            })
            .catch((e) => {
                console.log(e.message);
            });
        }
});
```

Figure 2

APPENDIX B

This section of the writeup outlines the extra credit opportunity available for Assignment 10. This extra credit task requires a working `SuperDate` class, which should have been created in Assignment 8. You will not be able to complete this extra credit task without a working `SuperDate` class.

The instructor will provide no assistance completing the extra credit task.

TASK EC.1: SETUP

1. Copy the **libs/lib_convert.js** file to your working Assignment 10 solution. You should end up with a file structure like what's shown in Figure 3 – **lib_convert.js** must be inside the **libs** folder. Failure to maintain the file structure will render extra credit null and void (meaning no credit).

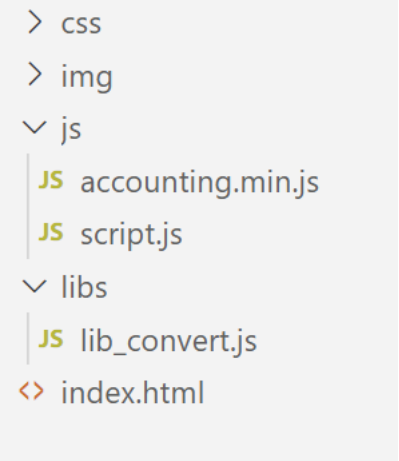

Figure 3

2. Open **index.html** and link **libs/lib_convert.js**.

Continues . . .

TASK EC.2: SUPERDATE()

1. Open **js/script.js**
2. Locate the block of code shown in Figure 4 (below).
    a. Note: Lines 54 and 55 are preserved for reference; however, the block shown in Figure 4 may appear on different lines in your solution depending on how you input the original code.

```
54                  let divHireDate = document.createElement('div');
55                      divHireDate.innerText = `Hire Date: ${employee.hire_date}`;
```

Figure 4

3. Create a new instance of the `SuperDate` class passing in `employee.hire_date` as the date value. Then, rewrite line 55 so that it uses `SuperDate` methods to output the full day name, full month name, the date value, and the full four-year date value in the Hire Date area. An example of this is shown in Figure 5 (below).
   (6 points)

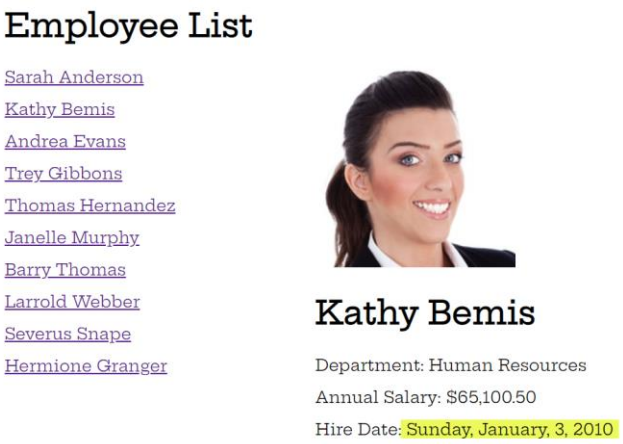

Figure 5

Continues . . .

TASK EC.3: SUPERDATE.GETDATEORDINAL()

An ordinal number is, "A word that expresses the relative position of an item in a sequence."[1] In colloquial English, we typical use ordinal numbers when denoting dates. For example, I might say, "Christmas is always on the 25th of December." 25th is an ordinal number.

In Task 1.2, you will modify the `SuperDate` class to output ordinal numbers for dates!

1. Open **libs/lib_convert.js**
2. Add a new method to the `SuperDate` class called `getDateOrdinal`() **(4 points)**
    a. The method should accept no parameters/arguments.
    b. Based on `this.getDate()` the method should return the date value with the appropriate sequence-suffix after it. For example, if `this.getDate()` returns **1**, then `getDateOrdinal`() should return 1st – if `this.getDate()` returns **23**, then `getDateOrdinal`() should return 23rd etc.
3. An example of the expected behavior for this task is shown in Figure 6 (below).

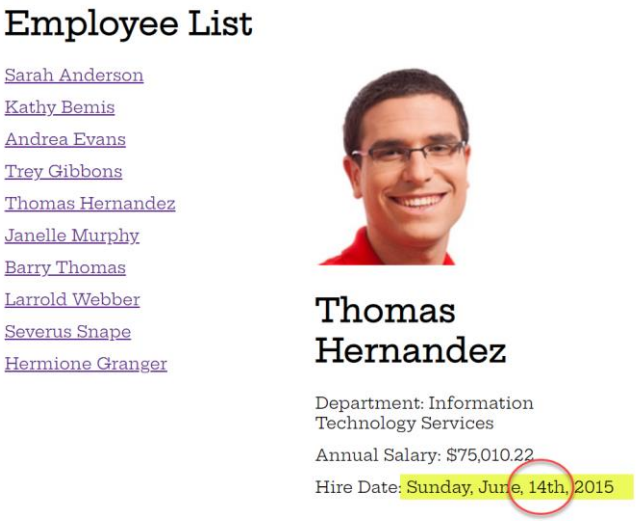

Figure 6

[1] https://en.wiktionary.org/wiki/ordinal_number#English