

CS 338: Graphical User Interfaces

Assignment 2

Deadline

Due Tuesday, March 1, 11:59pm

Submit electronically on Drexel Learn. You must include a **self-assessment form** with your submission.

Goals

This assignment asks you to build a user interface for the web. In recent lectures, you learned about HTML, CSS and JavaScript. In this assignment, you will combine all these skills to build a complete, albeit small, web-based GUI application. We will keep a simple “To-Do List” to track tasks that need to be done. The design of the website is specified below, and you will go through the process of using HTML and CSS to implement it. We will then add interactive behavior using HTML, JavaScript and CSS. You **should not use any additional frameworks or toolkits** (e.g. JQuery, Backbone, etc.). You must test your website on **two** browsers from this list: **Firefox, Chrome, Safari, Opera**.

Resources

There are numerous resources on the web for learning HTML, CSS, and JavaScript. There are several listed on the Resources tab of the course webpage:

<http://cci.drexel.edu/faculty/esolovey/courses/CS338-W16/resources.html>

In particular, you may find these useful:

About CSS Selectors: http://www.w3schools.com/cssref/css_selectors.asp

Try out different CSS Selectors: <http://www.w3schools.com/cssref/tryse1.asp>

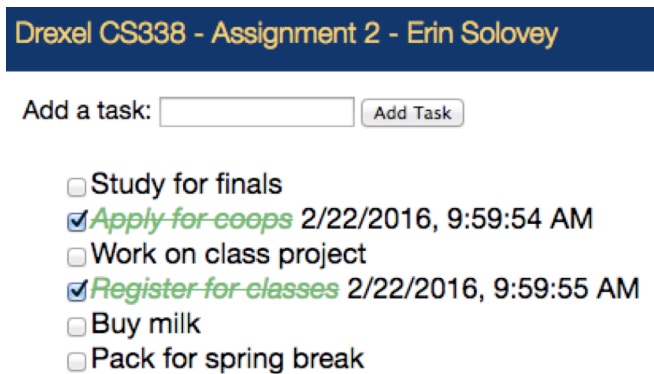
CSS Tutorial: <http://www.w3schools.com/css/default.asp>

HTML & CSS Lessons: <https://www.codecademy.com/learn/web>

You may use any development environment that you would like. The most basic requirement is that you have a text editor. We recommend Sublime Text. However, there are many others: Notepad++ (Windows), TextWrangler, vi, emacs, etc.

All of the browsers listed above have developer tools that can help you debug your code. For example, in Chrome, you would select View->Developer->Developer Tools. It is highly recommended that you explore the options for the browser.

Final Application



Drexel CS338 - Assignment 2 - Erin Solovey

Add a task:

- ☐ Study for finals
- ☒ ~~Apply for coops~~ 2/22/2016, 9:59:54 AM
- ☐ Work on class project
- ☒ ~~Register for classes~~ 2/22/2016, 9:59:55 AM
- ☐ Buy milk
- ☐ Pack for spring break

Figure 1. The completed user interface

Provided Files

We have provided you with CS338-Asst2.zip.

This contains the following:

- PartOne.html – The HTML file for the first part of the assignment.
- ToDoList.html – The HTML file with placeholders for your code
- ToDoList.js -- The JavaScript file that you will fill in to make your application interactive
- ToDoList.css – the file where you will set up your user interface CSS code

You will start with these files, and edit them as appropriate to complete the assignment.

Basic Layout

This part does not require any interactive functionality. You will just be creating the Layout to look similar to the figure. You will do this in **PartOne.html** and **ToDoList.css**. You should not need to use ToDoList.html or ToDoList.js for this part of the assignment.

1. Start by familiarizing yourself with PartOne.html in the browser and the HTML code. There are todo list items that are hard coded in there. You are welcome to change them, but you should keep 6 items in the list.
2. Set up the stylesheet to point to ToDoList.css.
3. Add your name in two places: the title and the Navigation Bar. In both places, the text should be "Drexel CS338 Assignment 2 – <YourFirstName> <YourLastName>
4. Set up the Navigation Bar at the top, by editing the CSS file. The height should be 50 pixels, the background color should be #003875 and there should be no border. In addition, you need to update the text style. Please add a left margin of 10 pixels, a font that is 18 pixels and the font color should be mapped to: # ffd65a.

5. Add a place to enter new tasks. This will not be functional in this section, but we will build a placeholder and work to style it. Above the ToDo list items, create a text field for entering text and a button for submitting the task to be added.
6. Use `ToDoList.css` to add style elements to the webpage.
 - a. Add a style to the check box elements so that when an item is checked, the text is 1) italicized, 2) (any) green color, 3) strike through. When the item is unchecked, the text is changed back.
 - b. There should be padding and/or margins around the text of 15-30 pixels.
 - c. The fonts should be 18px.
 - d. Use the CSS as needed to format page to look like figure above.

That's it! You no longer need `PartOne.html` for the remainder of the assignment, but it should be turned in. You will use the same `.css` file for the rest of the assignment.

Interactive User Interface

In this part, we will build from the work in the previous section, and will use the same CSS file. However, **we will not use `PartOne.html`**. Instead, we are now using **`ToDoList.html`**, **`ToDoList.js`** and **`ToDoList.css`**. These files have skeletons of the code required to finish the assignment. You can edit any of them as needed.

The code in **`ToDoList.html`** should contain only **HTML that is static** and will not change throughout user interaction. Dynamic code will be written into the document object model using JavaScript in `ToDoList.js`. This code will be inserted into the `<container>` section dynamically using JavaScript. Here are the tasks that are required:

1. Enter your full name in the title and the heading as in the previous section.
2. Take the code from **Step 5 above** and put it in this file to add a (non-interactive) text field and button for adding tasks.
3. Create the “**Model**” for the application in `ToDoList.js`. The **To Do List** is an array of **Tasks**. Each Task has a **Task Name** and a **Task Status** (complete or incomplete), and a **Task Completion Date**. Any user interaction should update this model (e.g. add the task to the list, change the Task Status). **Note:** The `view()` function will then be responsible for rendering the list any time the model is changed. These two parts of your code must be separated, as it is a key part of the assignment. Any updates of the model should be done through functions.
 - a. **Constructor:** There should be a constructor for the Task object that takes a Task Name and creates a new task with that name. It should also set the Task Status to incomplete, and the Task Completion Date to an empty string by default.
 - b. **Validation:** The model should also take care of any validation. For this assignment, you just need to check for a blank Task Name and create an alert that the Task Name is blank.
 - c. **Update:** There should be an **`updateStatus(taskIndex)`** function that changes the Task located at `taskIndex` in the To Do List from complete to incomplete and vice versa. When a task is set to complete, the completion date should be set to the current date and time automatically. When a task is set to incomplete, the completion date should be set back to empty.
 - d. There should also be a data structure for the To Do List. It is simply an array of Tasks.

4. Create the “**View**” for the application in **ToDoList.js**. This should go in `view()`. The view function should be making changes to the **container section** in the `ToDoList.html` dynamically. **Note:** In `PartOne.html`, the checklist items were static and hard-coded. Here you will dynamically create them, based on user input and any changes to the model. So, your code should find this section in the DOM and:
 - a. **Update View:** Get the `ToDo List` model and go through each item in list.
 - i. Create a check box input item with the Task Name and Task Completion Date (if applicable)
 - b. **Update CSS:** In **`ToDoList.css`**, set up check list items so that checked items have a text color of green, are shown with a line through them, and are in italics. Unchecked items should be shown as regular text.
5. Add an event handler function called **`AddTask()`** that is called when button is clicked. Register the event handler function to the button in **`ToDoList.html`**. The code for the function should be in **`ToDoList.js`**. There is a skeleton there for you to fill in. This function must:
 - a. **User Input:** Get "to do" item from user input using `getElementById()`
 - b. **Update the model:** Create new task. As noted above, the model should take care of setting task status to incomplete, and the completion date to empty. Add new task to To Do List model
 - c. **Update view:** Call `view()` to update view
6. Add an event handler function called **`UpdateStatus()`** that is called when a checkbox item is checked or unchecked. Update your `ToDoList.js` to register the event handlers with each check box. The **`UpdateStatus()`** function must do the following:
 - a. Update the model’s Task Status and Task Completion Date.
 - b. Call **`view()`** to update the view

Additional Notes

Besides getting the basic functionality correct, this assignment emphasizes an important aspect of GUI building. We want to aim for separation of model, view and controller. While some frameworks enforce this or at least make it easier, we are going to work through it in the basic JavaScript. Sometimes, this is discussed in terms of “state” and “render,” in which the state (or model) gets updated independently and the render is called to update any views of that model.

Please follow all instructions carefully to make grading easier; failure to follow them will result in a lower grade. Your program will be graded on Firefox and/or Chrome, and thus you are strongly encouraged to test your program on two browsers (e.g. Firefox, Chrome, Safari, Opera etc.) to ensure that the grader will see the same UI and functionality that you’re seeing on your system.

Documentation

It is expected that all code written for this assignment is properly commented. Please document each function you write/modify, especially in places where you have made particular choices about data structures and/or algorithms to employ. Also, please add an identification header to every file with the following format:

Your Name
Your-Email
CS338:GUI, Assignment [#]

Collaboration

This is an **individual** assignment. This means that all code should be your own, other than the code we provide. While it is okay to discuss the assignment with classmates, you may not view other students' code. Similarly, you should not let classmates view your code. Your submission should include a list of all people that you discussed the assignment with. If you did not collaborate with anyone, you should state that you did not discuss the assignment at all with anyone. This should appear in the submission comments on Drexel Learn. For more details, please see the course policies on the [syllabus](#).

Submission

Please zip your project files, and submit this package as an attachment for Assignment 2 on Drexel Learn. (A link to Drexel Learn is on the left sidebar of the main course home page.)