



CSE 316 - Fall 2019

Fundamentals of Software

Developing the **Wireframer** App

HW 1: 1st Technology Learning Assignment - GUIs & JavaScript

The Big Picture

This semester you will be creating CRUD Web Application for creating Wireframe GUI diagrams. It is a program we hope will be useful to students taking this very course in future semesters. The program will let the user login to an online account, create a new diagram, edit that diagram by adding various GUI controls, and export the diagram to image formats. Note that the site will store all created diagrams in an online database. This will be a semester-long project, so before we code our solution, we first will have to learn all about the necessary technologies to use. This is a common first step in Software Engineering, so get used to it. We will then design our solution, and *then* code. The HWs are divided up as follows:

- **HW 1 - 1st Technology Learning Assignment - GUIs & JavaScript:** In this assignment you will learn about building a GUI using HTML/CSS/JSON/JavaScript, i.e. front-end Web standards.
- **HW 2 - 2nd Technology Learning Assignment - Frameworks:** Here you'll learn about using a front-end Framework to make front-end GUI creation easier. You will also learn how to make your own Framework to manage a Transaction Processing System.
- **HW 3 - 3rd Technology Learning Assignment - Back-End Database:** Here you'll learn about using a back-end platform that

will provide a login system and a database such that we may persist all our work to an online database.

- **HW 4 - UML Design:** Now you'll design all of your code for your application using UML class and sequence diagrams. You will specify all necessary classes, variables, and methods, as well as the relationships between all types and the results of all triggered events.
- **HW 5 - Implementation Stage 1:** In this stage you will layout your full GUI and setup all test beds.
- **Final Project - *Wireframer*** - In this assignment you will complete the project.

To get @todo up and running do the following:

- **Download** and install Visual Studio Code. This is the IDE we will be using this semester for authoring source code.
- **Download** and install **Firefox**. The TAs will use this Web browser for testing this semester so it is to your advantage to use it as well.
- **Download** and install **Web Server for Chrome**. In order to test your work you'll need a Web server that you can run locally to make sure all linked files are served properly to your browser. The Teaching Assistants will use this tool for grading purposes, so I would advise doing the same.
- Create a local directory called **todo_hw1** on your computer where you will do your work for HW 1. Note that you will use a different directory for the next assignment.
- **Download/Clone** the **todo_hw1** project. Put the full project into the directory you have chosen to work. Note that you will be required to use **Git** starting in HW 2, so you don't have to employ it just yet, but if you are familiar with it, I thoroughly advise it. We will cover use of Git in an upcoming lecture and will get everyone

in the course skilled in its use soon enough.

- Open Web Server for Chrome and choose the folder where you put the downloaded project, then Start the Web Server. You'll notice in this program that your @todo Web application will be provided via <http://127.0.0.1:8887>, so all you'll then need to do is open that link in Firefox and you'll be able to run the @todo program.
- Once the application is up and running in Firefox you'll want to get accustomed to debugging using a Web browser. This is a **very** important skill. To do so, go to the ☰ menu in the top right corner and select **Web Developer** and then click on **Debugger**. There are tabs for viewing and editing anything and everything related to the page. **Inspector** has the DOM, **Console** displays output. **Debugger** lets you actively debug your program. There are also tabs for keeping track of network communications, threads, style, and all manner of performance metrics like CPU usage and memory. The first thing you should do is go to the Debugger Tab and put a breakpoint at the first point of entry for the program, which is line 22 inside **index.html**. Put a breakpoint there and refresh the page. See if you can walk through the complete initialization of the site. Be sure to put breakpoints inside callback functions as well so that you may get to see them operate.

@todo - Becoming Code Literate

In this assignment I'll be giving you a whole heap of code that is partially working but incomplete. It is vital that you start by learning how it all works. Do your best to figure out how all the pieces fit together and what each piece does. Writing carefully designed, modular code is one of the most important skills for you to develop this semester, and the code I have provided tries to encourage you to do just that.

In this first assignment we will be creating a front-end only Web app that will provide the user with a means for making and editing todo lists. It will let the user make a list, add action items to the list, edit items, edit the list, and delete a list. Note again, it is front-end only. So, once the page is loaded, we are storing all data in JavaScript variables,

but should the user press the browser's **Refresh** button the full page will be reloaded and all those variables will be lost.

Once you start the program you'll find that two screens are currently working and that the application can do couple of simple things. You can create a new list or view an existing one. You can also change a list name and navigate back to the home screen. Note that the views of your initial GUI are shown below in Figures 1 & 2:

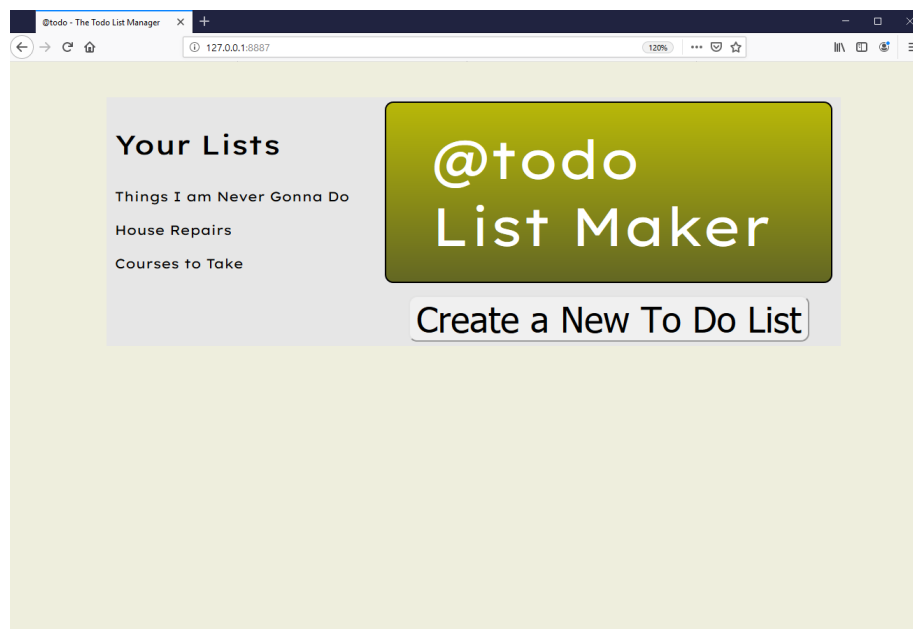


Figure 1: Initial Home Screen

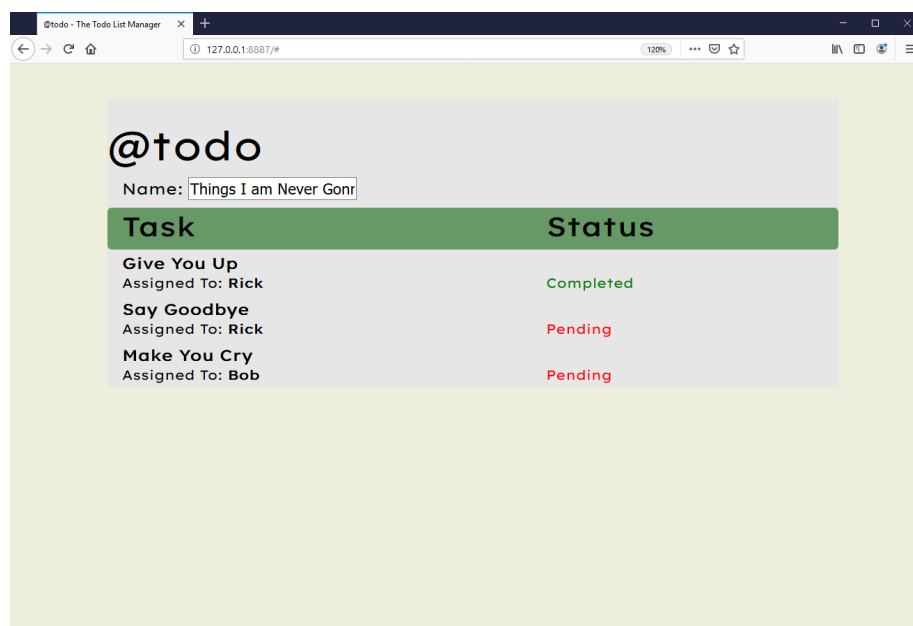


Figure 2: Initial List Screen

How are we using JSON

JSON stands for **JavaScript Object Notation** which like XML, is a Web standard for storing data. In fact, it is starting to replace XML in many respects. Note that we'll use both of these formats as they are things you should become familiar with. In this first assignment we will use a JSON file to load test data so that we can easily test all required functionality.

The @todo App - Required Functionality

Learn how *all* the code I have provided you works before getting started. This is essential for software engineering. Going in using guesswork to make changes is no way to build high quality software. Note that at the bottom of this page I have provided screen shots of how your application should look when it's done. To complete this application, do the following:

1. **Each List Should Have an Owner** - each list currently has a name, make sure each list also has an Owner (i.e. a person's name) that can be edited in a text field beside the list name.
2. **Add a Delete List Button to List Editing Screen** - add a trash can button to the list edit screen such that when pressed it slides in a dialog to verify if one wishes to delete the list. If the user presses **Yes**, the list will be deleted and the user will be brought back to the Welcome screen. If the user presses **No**, the dialog will simply close. Note that your dialog must animated in and off screen. Also note your trash can should be located as shown in the GUI figures.
3. **Each List Item Should Have a Due Date** - make sure each item in the list has a due date, which should be presented as shown in the UI diagrams. Make sure your items properly format item

dates.

4. **Move Up/Move Down/Delete Item Buttons** - add three buttons on the right side of each list item. One moves an item up the list, one down, and one deletes an item from the list.
5. **Add New Item Button** - add a button to the bottom of the list that when pressed, will bring the user to an item edit screen where they may enter information about a new item. This butt should look like a big plus.
6. **Select Item to Edit** - change the app such that when the user clicks on an item in the list the user may edit the item in a separate screen (as shown in Figure 5).
7. **Edit Item Screen & Controls** - add a new screen where the user may edit an Item's fields. Be sure to use a Date picker control for the due date, text fields for the description and assigned to fields, and a checkbox for the completed field. Add submit and cancel controls for the user to press to verify or cancel the edit. Note this screen must be used for both creating a new item and editing an existing one.
8. **Workspace Styling** - make sure your user interface looks just like the ones in the images provided.
9. **Fool Proof Design** - One of the best ways to prevent the user from doing illegal operations is to enable/disable controls in order to prevent the user from having illegal options. So, make sure that your application employs this approach. This means disable the move up button for the top item in the list and the move down button for the bottom item in the list.

And that's it! Note that there are a number of things we haven't done, like we have no mechanism for changing TA names or removing them, but don't worry about that for now. You'll need to look through a ton of my code to figure out what's going on here, but in fact, you don't have nearly as much code to write as I have provided. Don't take the assignment lightly, however. When dealing with a large code base like this, everything takes time. Also, try to understand as much as you can about all of my code as well, since you'll want to use much of it later in

your future app designs.

Once you've completed your work, your List screen should look like the image in Figure 3, your verification dialog should look like the image in Figure 4, and your Item screen should look like the image in Figure 5 as shown below:

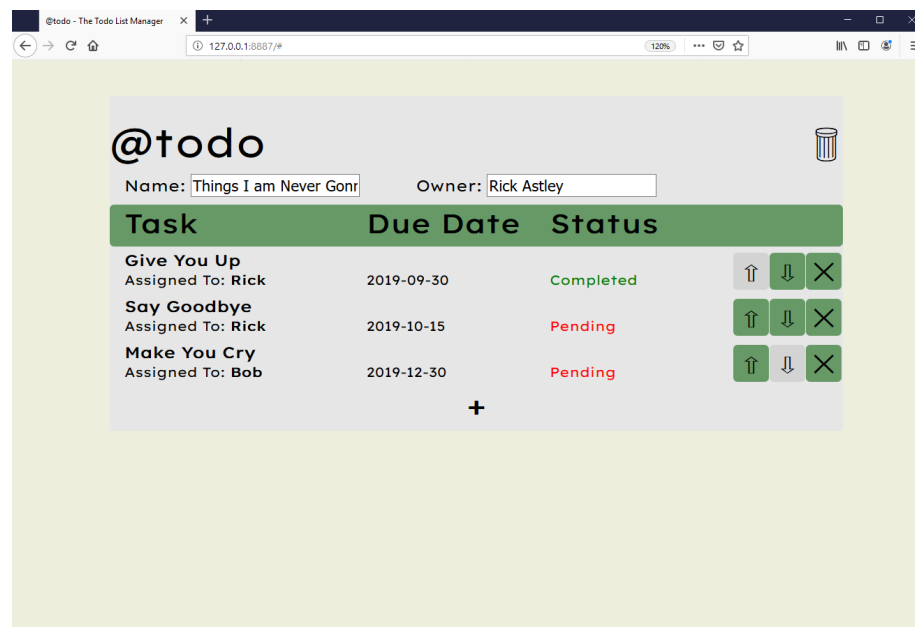


Figure 3: Completed List Screen

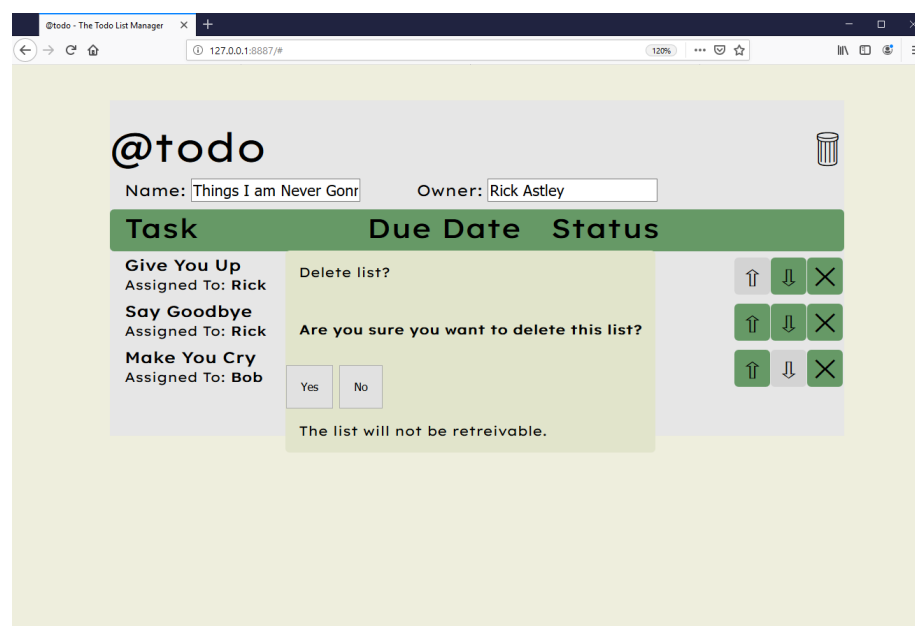
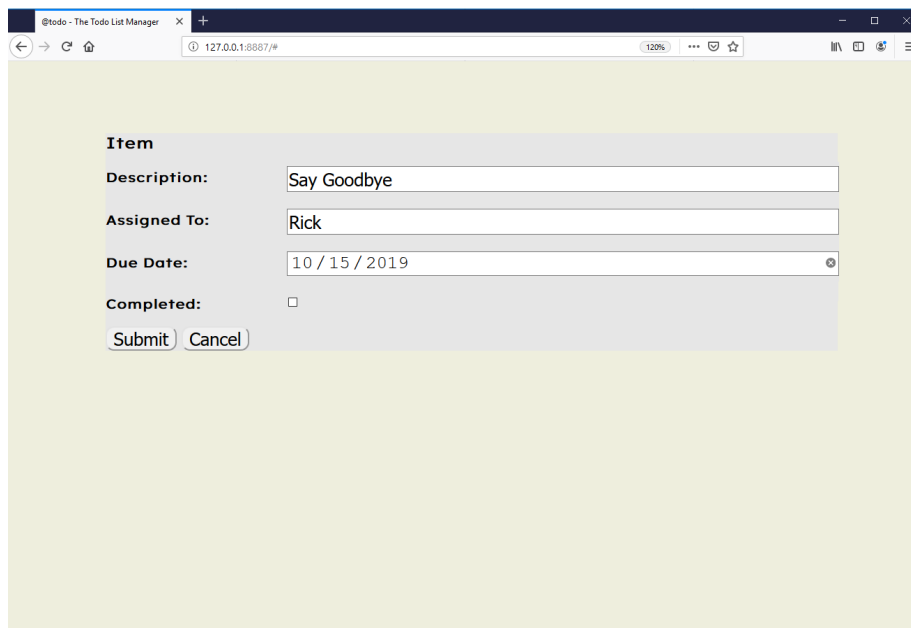


Figure 4: Completed Verification Dialog

The screenshot shows a web browser window with the title '@todo - The Todo List Manager'. The address bar shows '127.0.0.1:8887/#'. The main content area has a light yellow background. A modal dialog box is centered on the screen with a light gray background. The dialog has the title 'Item' and contains the following fields: 'Description:' with the value 'Say Goodbye', 'Assigned To:' with the value 'Rick', and 'Due Date:' with the value '10 / 15 / 2019'. There is a 'Completed:' checkbox which is currently unchecked. At the bottom of the dialog are two buttons: 'Submit' and 'Cancel'.

Figure 5: Completed Item Screen

Finally

Remember, one of the most important principles for you to learn this semester is to plan, then do. You **must** learn how all of the code works and why it is arranged as is before starting, and you **must** plan your approach before coding. Don't just march off to the north pole in your underwear and hope for the best. Note that demonstrations of my solution are being given during lecture.

Handin Instructions

When you are done, zip up the entire **todo_hw1** directory and submit that single ZIP file via **Blackboard**. Note, it **must** be a .zip file, **not** a .rar file.

Grading

Note that grading will be based on program performance and will only be done by appointment with each student's prescribed Teaching Assistant. This part of your grade will be based on how well your program performs specific required tasks.



Stony Brook University

Web page created and maintained
by



Stony Brook University
Computer Science Department