# **Angular Lab: Part 3**

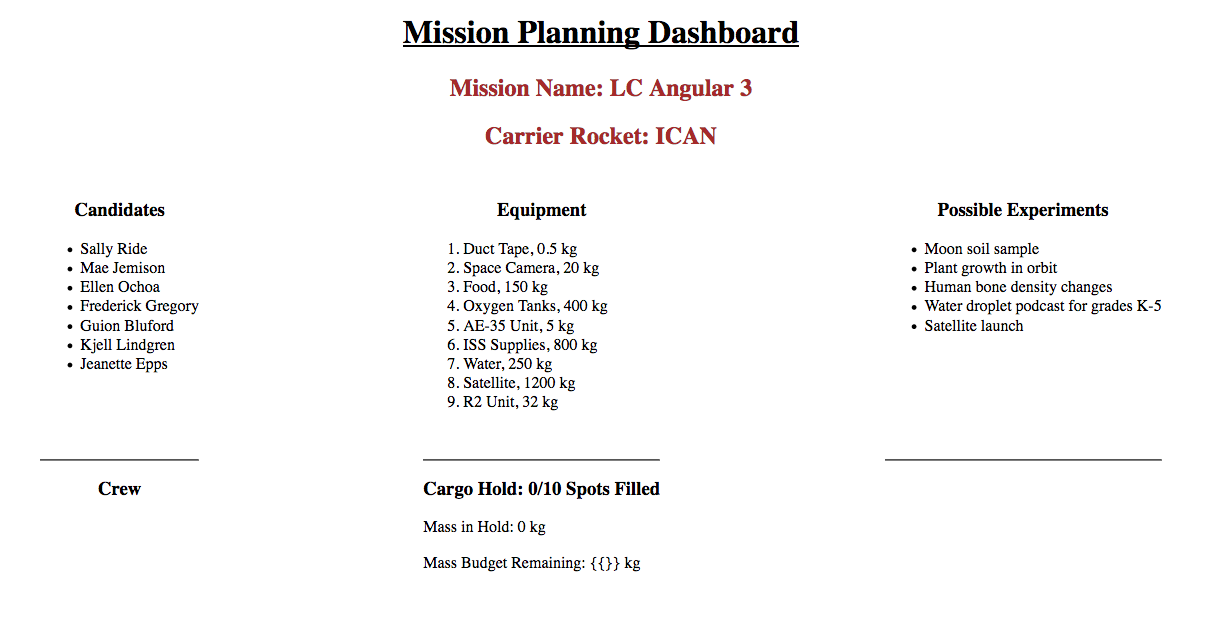
In this lab, you must combine data-binding and attribute directives to dynamically change the appearance or behavior of HTML elements.

## **1. Getting Started**

This lab uses the same mission planner repository as Angular Lab part 2.

1. Unzip the downloaded starter project in this folder and move it to the angular\_practice directory you are using for your Lab Assignments.
2. Run npm install on the root directory of the project.
3. Run **npm install** to download dependencies.
4. Run **ng serve** to build and serve the project.

When viewed in your browser, the project should look like this:



Mission Name and Carrier Rocket are still editable, but the functionality for the crew, equipment, and experiments have been removed.

## **2. Part 1: Select Cargo**

We want to add items from the equipment list to the cargo hold, but we must NOT exceed the values of the **maximumAllowedMass** or **maxItems** variables.

The interactive equipment list will eventually behave as follows:

1. When clicked, an Add to Cargo Hold button adds the selected equipment to the **cargoHold** array and updates the **cargoMass** variable.
2. As items are added to the hold, their names should appear in the Cargo Hold section of the page. Also, the Mass in Hold, Mass Budget Remaining, and Spots Filled values should update.
3. The Add to Cargo Hold buttons should be disabled if all of the spots in the hold have been filled.
4. If the mass of a particular item will push the cargo hold over **maximumAllowedMass**, that item's button should be disabled.
5. If **cargoMass** comes within 200 kg of **maximumAllowedMass**, then the Mass Budget Remaining text should turn red.

The following sections will guide you through completing Part 1.

### **2.1. Code the addItem Function**

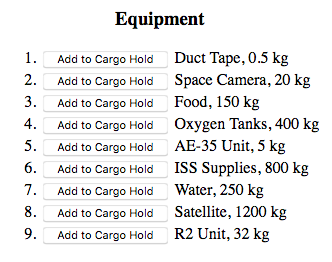
Open **equipment.component.ts** and create the **addItem** function:

1. It should take an equipment object as a parameter.
2. It should add the equipment object to the **cargoHold** array.
3. It should increase the **cargoMass** variable by the mass of the new equipment.
4. It should return **true** or **false** depending on whether **cargoMass** is within 200 kg of **maximumAllowedMass**.

### **2.2. Make the Add to Cargo Hold Buttons Work**

Open **equipment.component.html** and do the following:

1. Include an Add to Cargo Hold button within each list item.



1. Add a **(click)** event to each button that calls the **addItem** function and passes the selected equipment as the argument.
2. Bind the **disabled** attribute to the following conditions:
   1. If all of the cargo hold spots are full (**cargoHold.length === maxItems**), disable the button.
   2. If adding the item to the cargo hold would exceed **maximumAllowedMass**, disable the button.
3. If active, make the button an attractive color.



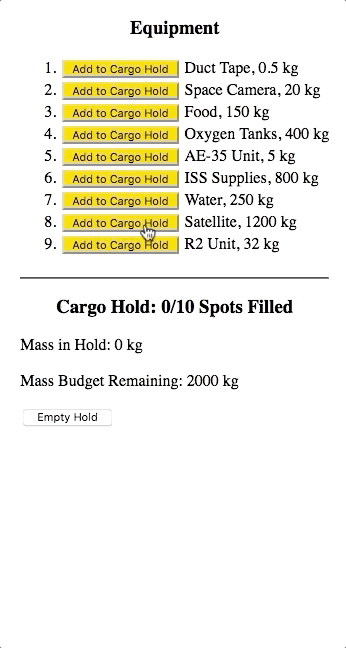
### **2.3. Update the Cargo Hold Display**

Open **equipment.component.html** and **equipment.component.css** and do the following:

1. Add a **nearMaxMass** class in the CSS file that styles the text to be:
   1. Bold OR italic,
   2. Red.
2. Bind **[class.nearMaxMass]** to a boolean that will change the style of the Mass Budget Remaining text whenever the cargo hold gets within 200 kg of **maximumAllowedMass**.
3. Add an Empty Hold button that clears the **cargoHold** array and resets **cargoMass**. As a side effect, clearing the hold should reactivate all of the buttons and return Mass Budget Remaining to its original style.

### **2.4. Status Check**

At this point, the equipment component should behave something like:



## **3. Part 2: Select Crew Members**

We want to add up to three astronauts to the mission crew, and we want to do this by clicking on their names rather than creating more buttons.

The interactive candidates list will eventually behave as follows:

1. When clicked, the candidate's name will change color and will appear in the Crew list.
2. If a candidate is already part of the crew, clicking their name again in the Candidates list will remove them from the crew.
3. When the mouse pointer hovers over an astronaut's name in the Crew list, their photo appears below the list. When the pointer leaves their name, their photo disappears.
4. When the crew size reaches 3 members, the heading changes to Crew Full and clicking on more candidate names will not do anything.

The following sections will guide you through completing Part 2.

### **3.1. Code the addCrewMember Function**

Open **crew.component.ts** and create the **addCrewMember** function:

1. It should take a candidate object as a parameter.
2. It should check if the candidate is already part of the crew.
3. If the crew size is less than 3 AND the candidate is not part of the crew, then their data should be added to the **crew** array.
4. If the candidate is already part of the crew, then their data should be removed from the **crew** array.

### **3.2. Update the Candidates List**

Open **crew.component.html** and **crew.component.css** and do the following:

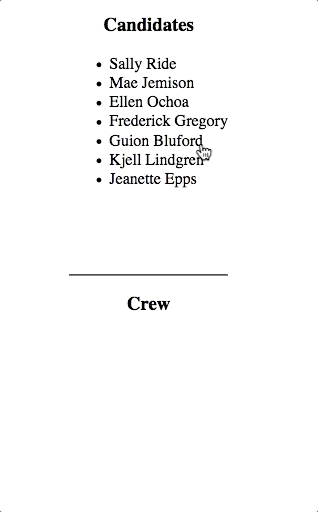
1. Add a **(click)** event to each **li** element that calls the **addCrewMember** function and passes the selected candidate as the argument.
2. Add a **selected** class in the CSS file that styles the text to be a different color from the other list items.
3. Bind **[class.selected]** to a boolean statement that will change the color of a candidate's name when they are selected or de-selected for the crew.

### **3.3. Update the Crew List**

1. When the crew size reaches 3, the heading should change to "Crew Full".
2. Add **(mouseover)** and **(mouseout)** events to the **li** tags to determine if the mouse pointer is currently over a name in the Crew list.
3. If a crew member is selected by moving the mouse over their name:
   1. Use an **img** tag with **\*ngIf** to display a photo of the astronaut below the crew list.
   2. Bind the **.photo** property of the astronaut to the **src** attribute.
   3. When the mouse pointer moves off of a name, the photo should disappear.

### **3.4. Status Check**

At this point, the crew component should behave something like:



## **4. Bonus Missions**

To boost your Angular skills, add one or more of the following features:

1. Update the CSS files to make the web page look a little less bland.
2. Don't allow more than two of the same item in the cargo hold.
3. Allow the user to remove individual items from the hold.
4. Complete the **experiments** component with features similar to the **crew** and **equipment** components.
5. Add other data to the astronaut objects, and center this data below the crew photo.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*