This is the steps I take in order to create this to do app:

1. I straight away remember CRUD individually. This means:

* Create = addTodo
* Read = displayTodo
* Update = changeTodo
* Delete = deleteTodo

After that, add the toggleCompleted and toggleAll. Stay calm, use logic for each of the 6.

1. *(This is a good place to create a new object just to handle the handlers. Remember objects – more specific = better)* Start adding buttons for the CRUD. Naturally we need text boxes too. Give the boxes an id so we can grab it using javascript.
2. *(This is a good place to create a new object just to handle the view. Remember objects – more specific = better)*

When you click on the buttons in step 2, it displays things in the console only. Now we escape from console so that when you click on buttons, it displays things in the HTML. **First**, create **<ul>** element. **Second** do a for loop that matches the number of items in the todos array. Inside it create **<li>** and set the **textContent** to be the todo object. **Third,** append it to the **<ul>**. Lastly, to truly escape from the console, call the displayTodos method from view object at the end of each handler method while also deleting the displayTodos from the todoList object. This one prints to the console. We don’t need it, already got one that prints to the DOM.

So **value** is simply the value of a text box etc. **innerHTML** is the content inside an element, like body, div, etc., IN STRING. **textContent** is similar but better performance and can protect from XSS attacks

1. Clicking to delete. Firstly, create a function in View object whose job is to create delete buttons. Don’t forget to add **className**. Then append to each **<li>**. Then each **<li>** should have an id that has the todo position. Then create a function to setup the eventlisteners. Here, the logic is, check what is clicked. If what is clicked is the className then call the **handlers.deleteTodo()**

Instead of adding an **eventListener** to each **<li>** item, you can add a single **addEventListener** to the enclosing **<ul>**, and use the **event** object to find out which item was clicked. This is called **event delegation.**

**var view = {**

**displayTodos: function() {**

**var todosUl = document.querySelector('ul');**

**todosUl.innerHTML = '';**

**todoList.todos.forEach(function(todo, position) { <------------------------------------2**

**var todoLi = document.createElement('li');**

**var todoTextWithCompletion = '';**

**if (todo.completed === true) {**

**todoTextWithCompletion = '(x) ' + todo.todoText;**

**} else {**

**todoTextWithCompletion = '( ) ' + todo.todoText;**

**}**

**todoLi.id = position; <-------------------------------------------------------------------------4**

**todoLi.textContent = todoTextWithCompletion;**

**todoLi.appendChild(this.createDeleteButton()); <------------------------------------3**

**todosUl.appendChild(todoLi);**

**}, this) <------------------------------------------------------------------------------------------1**

Look at 3. The **this** cannot refer to the **view** object because this whole thing is inside a forEach function (2). (**createDeleteButton** is a method of **view**). So to fix this, give a second argument, **this**(1). Also, noticed I used a second argument called **position**(2) to replace the **[i]** of giving the **li** the **id** (4). This second argument, p**osition,** represents the index, starting at 0. Its like the **each\_with\_index** of ruby