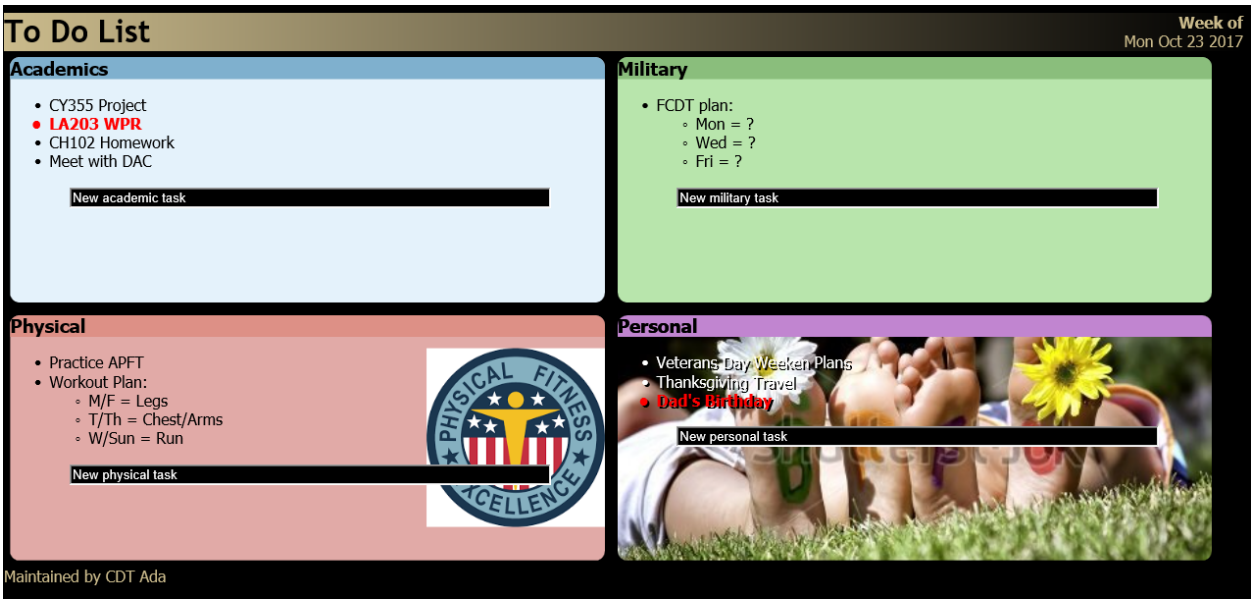
JavaScript ICE 2

JavaScript provides many powerful features for integrating data with webpages. This requires you to understand how to use basic language structures (like variables, arrays, objects, and loops), but is greatly simplified with more powerful abstractions like JSON (JavaScript Object Notation) and operations that modify the DOM (Document Object Model).

In addition to writing your own JavaScript code, you can take advantage of freely available JavaScript libraries, such as W3.JS from W3Schools.com. The purpose of this ICE is to introduce the concepts of understanding a library (API) and then using that understanding to add more capability to your web site.

We will do this by adding JavaScript functionality to our previous HTML/CSS ICE. At the end of the HTML/CSS lessons, our Weekly Planner web page looked like the following:



Using the W3.JS Library

Before we can use a JavaScript library, we first need to determine whether we are going to link to the library remotely or download a local copy. The benefit of linking to a library/API (such as JQuery, Bootstrap, etc. from a content delivery network (CDN) is that users may be able to load your web site faster if they have accessed a site using that library (i.e. the file is already cached) or if the CDN server is actually located closer to the user (which is a good chance) than your website. The problem is if there is a connectivity issue to that library. A local copy removes that problem, but may make your website a little slower to load.

For our ICE, we will download a local copy of **`w3.js`** and store it in a subfolder called `scripts`. The URL of the w3.js file is https://www.w3schools.com/lib/w3.js.

Let's link to this local copy of w3.js by placing it within the <head> element:

```
<script src="scripts/w3.js"></script>
```

Using the W3.JS Include

Our current webpage is a single HTML file. However, we can definitely see that logically we can divide our webpage into separate components (will be called templates in Meteor). Each of these components are now smaller and easier to maintain. Also, this allows us to reuse components. Looking at the CY355 Web Site (http://www-internal.eecs.usma.edu/courses/cy355/), we see several components reused over and over (header, navigation, and footer). If the contact information for LTC Harvie changed in the footer, this change would have to be changed in multiple files. It would be better to have the footer included as a separate HTML file so that LTC Harvie would only have to update one file if he changes contact information versus approximately 50 files.

For our ICE, select all the content within the `<section class="military list">` element and save it to a file named **military_list.html**.

Modify the <section> element by giving it the newly defined attribute w3-include-HMTL to point to our newly created header.html file with the following code:

```
<section class="military list" w3-include-html="military_list.html">
</section>
```

However, we are not done yet because we have not called the **w3.includeHTML()** function that will look for the w3-include-HTML attribute in our HTML document. Right before the `</body>` tag at the end of main page insert the following JavaScript:

```
<script>
    w3.includeHTML();
 </script>
```

NOTE: We will need to view this using Firefox. The `includeHTML()` function uses xhttp.send() which assumes that the file is on a web server. Firefox will still let you see the file even though it is not on a web server, Chrome and Edge will not. However, you will be able to see the snippet on both browsers when you upload the file to the web server.

Using the W3.JS Display

We can read in data from a JavaScript object that is store in a separate file. Updating that JavaScript object will allow us to update what is displayed. This is how the schedule page of the CY355 Web Site displays. In this example, let's move the academic list items from the HTML file to a separate JavaScript file that we can call.

Let's first create the academic_tasks.js file with the following code:

```
var academicTaskList = {"academicTaskItems":[
  {"AcademicTask":"CY355 Project"},
  {"AcademicTask":"LA203 WPR"},
  {"AcademicTask":"CH102 Homework"},
  {"AcademicTask":"Meet with DAC"}
]};
```

We need to link to this new JS file by using a `<script>` inside the `<head>` element.

```
<script src="academic_tasks.js"></script>
```

Now, that we have our JavaScript Object to reference, we now need to give our `<ul>` element and id that we can reference when we call the `w3.displayObject()` function.

```
<ul id="academicTasks">
```

We want to repeat the academic tasks in the `<li>` elements, so we insert the w3-repeat attribute into the first `<li>` element and set it equal to the JSON object (academicTaskItems). Within the `<li>` element we use `{{AcademicTask}}` to indicate that we want to display the value of the key `AcademicTask` here:

```
<ul id="academicTasks">
        <li w3-repeat="academicTaskItems">{{AcademicTask}}</li>
  </ul>
```

Finally, in our `<script>` element at the bottom of our web page, we need to tie the `<ul id="academicTasks">` to the JavaScript Object (`academicTaskList`) with the following code:

```
<script>
    w3.includeHTML();
    w3.displayObject("academicTasks", academicTaskList);
</script>
```

You will notice that LA203 WPR is no longer red because we got rid of the class="important". We can bring that back by modifying our JS file to have another key called Category. We will set the Category for LA203 WPR to important, but leave the other items empty. Here is our modified academic_tasks.js file:

```
var academicTaskList = {"academicTaskItems":[
  {"AcademicTask":"CY355 Project", "Category":""},
  {"AcademicTask":"LA203 WPR", "Category":"important"},
  {"AcademicTask":"CH102 Homework", "Category":""},
  {"AcademicTask":"Meet with DAC", "Category":""}
]};
```

Now, we just need to modify the <li> element so that the class attribute is set to the value of Category:

```
<li w3-repeat="academicTaskItems" class={{Category}}>{{AcademicTask}}
</li>
```

Using the W3.JS Sort

We can also insert a button that will allow us to sort our Academic Task alphabetically. Beneath our `<ul>` element, we insert the following code:

```
<button onclick="w3.sortHTML('#academicTasks','li')">Sort Tasks
</button>
```