**Requirements**

Build your Web Project based on the minimum requirements listed below to earn the most points.  These features are not listed in any particular order.

* Plan a website containing at least three pages with a common layout and navigation system.  Feel free to add additional pages as needed, but please try not to exceed 5 pages.
* Create wireframes for your pages.
* Add a feature to one of the web pages in your personal site that prompts the user to enter their name and greet them by their name.  Store the user's name to the localStorage.  Next time the user returns to the site, your site should check the localStorage and display the user's name.  If no localStorage exists, then prompt the user to enter their name.
* Add a feature to one of the web pages in your personal site that incorporates content or functionality created by a series of if, if/else, and/or else if statements, or by a switch statement.
* Add event listeners to interact with the DOM.  Examples:  click a button to perform a task, display content to a specific location on a web page, remove content based on a button selection, etc.
* Add a web form to one of your pages.  The form must include at least four fields for user to enter data.
* Add exception handling to the code for one of the forms on your personal web site. If your site does not include a form, add one first. Your code should display one or more relevant error messages in an appropriate location. After you finalize your code, write a summary of the debugging methods from this chapter that you used in this project, describing how you used each one in your code. (See page 288)
* Add validation to the code for one of the forms on your individual website. First, ensure that your form uses at least three of the following field types: check boxes, text boxes, option buttons, selection lists, and text areas. Then, program validation for your form ensuring that users enter values or make selections in all fields, and verifying at least one other aspect of at least one of the fields. Provide appropriate feedback to users when the form fails validation. Test your completed program until all validation works reliably with different combinations of valid and erroneous data. (See page 450)
* Expand your individual website to include a page that calculates the time elapsed since a date entered by a user. The page should include a form that allows users to enter a day, month, and year. The page should then calculate and display the elapsed time in years, months, and days. Note that your program must include code to convert day values in excess of into months, and months in excess of into years. (See page 531)
* Enhance the feedback form in your project to enable users to choose one or more options from a list of at least five options. Include code that adds user selections to either an array or an object, and ensure that if a user deselects one of the options, it is removed from the array or object. Add code to convert the array or object to a string.  (See page 617)
* Identify data provided by an AJAX service that you’d like to include in your personal website. You should choose a web service other than those used in the chapter and the Hands-on Projects. If you have an idea for data you’d like to access but are unsure what service might provide that data, perform a web search on a description of the data plus “API.” For instance, if you were looking for a source of tide tables, you might search for “tide tables API.” Use the documentation for the web service to construct an AJAX request and to display selected data from the service on your website. Note that if you don’t have experience with writing PHP, you may need to examine a number of potential APIs to identify one that allows JSON-P or CORS requests, which don’t require you to run a proxy. (See page 811)
* In your individual website, revise a function to use jQuery selectors and methods. Identify a function that contains at least three selectors that you can replace with jQuery selectors, and that performs at least one DOM traversal or CSS change that you can replace with a jQuery method. Comment out the code you replace rather than deleting it. Be sure to link to the jQuery library in all HTML documents that link to the .js file you’ve updated. When your revisions are done, test all pages that use the function to ensure they still perform as they did when the function was written in plain JavaScript. (See page 839)
* Your site should use at least one example of JavaScript modules.
* Create and use custom classes to create your own objects in your project.  You may use either the **class approach** or **function constructor approach**.
* Create and user JSON data in your project.
* Provide code documentation.  Document your code where necessary to explain what the code does.  For example, document your function by explaining what it does, purpose of the parameters used (if any), and what data the function returns (if any).
* /\*\*
* \* Calculate the sum of two numbers. If the parameters are no numeric, return 0
* \* @param {\*} a - the first numeric operand
* \* @param {\*} b - the second numeric operand
* \* @returns the total of the two numbers or 0 if not numeric
* \*/
* function addTwoNumbers(a,b){
* total = a+b;
* if (isNaN(total)){
* return 0;
* } else {
* return total;
* }

}