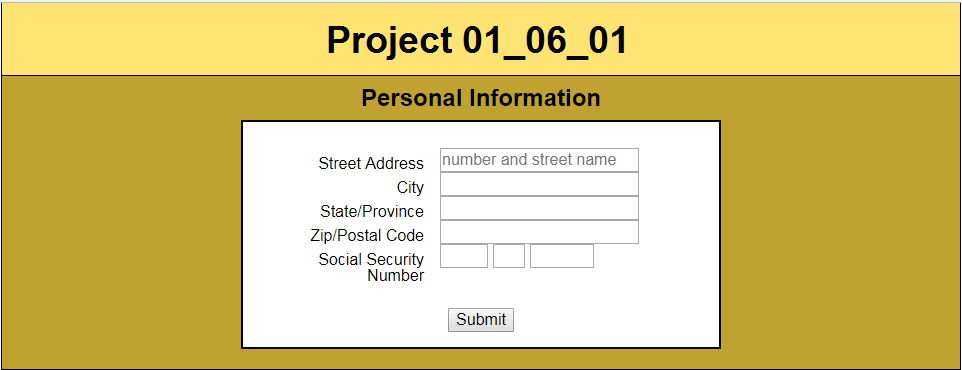
**Project 01\_06\_01**

In this project, you will create a set of validation functions for form submission.

1. Copy all of the completed files from ***Project 01\_06\_01.zip*** into a ***Project 01\_06\_01*** folder; this will contain all of the files you will zip up and submit. Open all of the necessary files in your IDE and complete the documentation at the top.
2. Go to the ***index.html*** file and add a ***<script>*** element just before the closing ***</head>*** tag:  
    ***<script type="text/javascript" src="script.js"></script>***Go to the opening ***<form>*** element and add code to ***disable*** browser-based validation.
3. Save the file and create a new JavaScript file, ***script.js***. Add all of the appropriate documentation to the top of the file, with your name and the date. Add an directive to the file to ***use strict*** JavaScript. Add a ***global*** variable to the file named ***formValidity*** and set it to true.  
   Save the file and run ***index.html*** in the browser. Open the Developer Tools and make sure that everything looks good and there are no Console errors. Submit the file to make sure that browser-based validation is off and the ***<form>*** submits.
4. Let’s scaffold out the structure of the script. Go to the bottom and add event listeners for the ***load*** event. The event handler should be named ***createEventListeners()***. Scaffold the event handler with some test code in it.   
   Do a browser test for both syntax and to make sure the event handler executes.
5. Let’s start building the code for the new event handler. Its purpose is to create any other event listeners that we will need. Our first listener should be for the ***submit*** event. It’s event handler should be named ***validateForm()***. Scaffold the event handler with some test code in it.   
   Do a browser test for both syntax and to make sure the event handler executes. It should execute on ***submit***.
6. Now let’s build out ***validateForm()***. The function will need a parameter, which will receive the ***submit*** event object. We can make use of its properties and methods. We will use one of its methods to disable the default behavior for the ***submit*** event. This must work for both newer and older browsers. Also, we will reset the ***formValidity*** variable to ***true*** for now, until we build it out further:  
   Give this a test in the browser. The ***Submit*** button should not produce a submission page anymore. We should have effectively stopped the form from submitting.
7. We are going to build a bunch of validity tests in this function. But before we do, let’s build a framework that will handle what will be the ***results*** of these tests. For starters, let’s build the code that will execute if the ***<form>*** turns out to be valid after our soon-to-be-built tests:  
   A browser test now should show that ***Submit*** produces results, because we have not built any validation checks yet. That is indicated by the displayed values.
8. Now we can add a call to our first validity check named ***validateRequired()*** into the correct spot in ***validateForm()***.
9. Start building our ***validateRequired()*** validity check by building a function called ***validateRequired()***. Add some test code to it so we can see if it executes with a browser test.
10. Now let’s set up some variables in the function for support. We will need to grab all of the ***<input>*** fields and group them in an array we will be able to loop through. Another will grab the ***<div>*** for the fieldset that will hold error messages. We will also need variables to track the validity of the ***<fieldset>***, the number of elements, and the current element.
11. We will need to set up a ***try*** / ***catch*** structure to handle the validation. Our intention will be to have the ***try*** clause loop through the fields, testing for validity. After looping through the fields for data, it will ***throw*** an error if necessary. The ***catch*** clause will handle the mechanics of the error. For starters, let’s scaffold the structure. We’ll start with building the ***try*** clause with just a piece of test code. We can build out the entire ***catch*** clause.  
    Check in the browser for any syntax errors and to make sure the try clause is executing.
12. Create a test error to ***throw*** in the ***try*** clause to make sure that the ***catch*** clause is operating properly. Give that a test.
13. Let’s build a ***for*** loop in the ***try*** clause to iterate through all of the ***<input>*** fields, checking for validity. In this case, validity means there is a value in each, they are ***required***. Every one that we find that is invalid, we will change its ***background*** property, and we will also mark the ***fieldsetValidity*** variable as ***false***: For testing, let’s use a ***debugger*** command to monitor results.  
    A browser test with Developer Tools on and with no data should give us a number of debugger pauses on submission and we should be able to see the field backgrounds have changed. Now let’s put data into the some fields and give it a check.
14. Now let’s set up throwing the error in the proper place in the ***try*** clause  
      
    If everything appears to be working in the browser test, remove the ***debugger*** statement and let’s give it some significant testing.